uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería Electrónica Industrial y Automática
Curso Académico 2017-2018

Trabajo Fin de Grado

Montaje y programación de la mano robótica Dextra

Daniel Jason Castillo Patton

Tutor

Álvaro Villoslada Peciña

Escuela Politécnica Superior, Leganés, Julio 2018

Esta obra se encuentra sujeta a la licencia Creative Commons **Reconocimiento – No Comercial – Sin Obra Derivada**

Resumen

El presente Trabajo de Fin de Grado tiene como objetivo el montaje y programación de la mano robótica perteneciente al proyecto de *open-source* (código abierto) Dextra, con el objetivo de realizar nuevas implementaciones en la mano para mejorar su diseño y ampliar su funcionalidad.

Consiste en la construcción de una mano robótica-protésica impresa en 3D, siendo realizado el montaje en su totalidad incorporando todos los elementos necesarios para el funcionamiento posterior de la mano (tendones, motores, superficie antideslizante, etc.) y la electrónica necesaria soldando todos los elementos (motores, cables, drivers, etc.) en un circuito impreso.

Tras realizar el montaje de la mano en su conjunto, se usará una API de Python ya creada que permite el control de la mano Dextra, para realizar los ajustes y pruebas necesarias para evaluar su correcto funcionamiento. Posteriormente, se realizará la programación de varios códigos para ejecutar una serie de agarres predefinidos.

Como conclusión se obtiene una mano robótica funcional, que puede ser usada como prótesis, con un amplio rango de movimientos y de bajo coste.

Abstract

The objective Final Degree Project is the building and coding of a robotic hand that belongs to the Dextra open-source project, with the aim of doing new implementations to the hand itself in order to upgrade it in functionality.

The project consists in the building of a 3D-printed robotic-prosthetic hand, the assembly being made in its totality incorporating all the necessary elements for the subsequent operation of the hand (tendons, motors, anti-slip surface, etc…) and the necessary electronics soldering all the elements (motors, cables, drivers, etc…) in a printed circuit board.

After assembling the hand, we will use an existing Python API that allows the control of the Dextra hand, to make the necessary adjustments and tests to evaluate its correct functioning. Subsequently, the programming of several codes will be performed or execute a series of predefined grasps.

In conclusion, we will obtain a functional robotic hand, which can be used as a prosthesis, with a wide range of movements and a low-cost price.

Agradecimientos

"Mucha gente opina de cosas que desconoce. Y cuanto más ignorantes son, más opiniones tienen"- Thomas Hilderin

A mis padres, Antonio y Carole, por darme la oportunidad de poder estudiar esta carrera,

a mi hermano, Andy, por su apoyo durante todos estos años

a mi tutor, Álvaro, ya que sin él este trabajo no sería posible, y por toda su ayuda a lo largo de este proyecto,

y a mis amigos, por estar ahí cuando más los necesitaba.

Índice

Índice de figuras

Índice de tablas

1. Introducción

1.1. Motivaciones

La principal motivación que lleva a la realización de este trabajo es el análisis de la viabilidad de la mano Dextra en cuestiones de diseño y su alcance para ser una mano de bajo coste, realizando nuevas implementaciones como el cambio de los tendones, o la eliminación de material para introducir distintos tipos de conectores para mejorar el diseño y la modularidad de esta.

A su vez, se desea crear una serie de códigos que ejecuten distintos agarres predefinidos (se seguirá la anatomía de Cutkosky, que define los movimientos básicos de la mano humana) para que puedan ser utilizados por un futuro usuario.

1.2. Introducción

Durante la historia de la humanidad, las enfermedades, las guerras y los accidentes han provocado numerosas pérdidas de miembros, tanto inferiores como superiores. Esto ha llevado a la necesidad de crear prótesis para reemplazarlos, que en los últimos años han tenido un gran desarrollo. Aunque las prótesis sacadas al mercado son muy avanzadas, los precios de muchas son desorbitados por lo que personas con recursos limitados no pueden acceder a estas.

De estas premisas surge la creación de proyectos de *open-source*, donde desarrolladores suben sus esquemas y sus códigos a Internet bajo una licencia especial que permite la utilización de los elementos desarrollados con el fin de mejorarlos, para que cualquiera sea libre de usarlos. Esto conlleva un desarrollo exponencial de la tecnología, ya que cualquier idea exitosa es rápidamente absorbida por el resto de desarrolladores, y a la reducción drástica de los precios, ya que se tiende a usar elementos baratos, pasando de los decenas de miles de euros a precios incluso inferiores a los mil euros.

Nuestra mano robótica Dextra, será creada desde cero con una impresora 3D, obteniendo los archivos de las piezas de la web del creador, a las cuales se les harán una serie de modificaciones durante el montaje, con el fin de probar nuevos añadidos. Se espera que estas modificaciones supongan una mejora para futuros desarrolladores, así como para el creador de la mano, de acuerdo a la filosofía de colaboración en comunidad que es la base de los proyectos *open-source*.

Una serie de movimientos basados en la taxonomía de Cutkosky serán implementados en forma de distintos códigos con el fin de intentar recrear todos los agarres de la mano humana a lo largo de la vida diaria.

1.3. Objetivos

El proyecto se enmarca en un proyecto *open-source*, donde el alumno fabricara y montara una mano Dextra. Una vez operativa, el alumno trabajará en la programación de la mano usando una API de Python que permite el control de Dextra, generando una serie de agarres y movimientos predefinidos que puedan usarse posteriormente para las aplicaciones para las que está diseñada la mano robótica. Durante este proceso se pretende profundizar en cada uno de los procesos que dan como resultado la mano

operativa, como realizar la impresión de todas las piezas mediante impresión 3D, realizar el montaje con las modificaciones pertinentes y la creación de códigos de agarres predefinidos, comprobando la efectividad de los mismos.

Otro objetivo, aparte de la realización de la construcción desde cero de una mano robótica, es crear una guía para que cualquier persona sea capaz de fabricarla sin necesidad de ayuda técnica y con unos conocimientos técnicos mínimos ya que, siguiendo las premisas de la filosofía *open-source*, se desea crear un proyecto que sea viable tanto para desarrolladores como para gente con una amputación que requieran una prótesis que sea modificable, fácilmente reemplazable y de bajo coste. Las necesidades socio-económicas de muchas de estas personas, que analizaremos más en detalle, pueden impedir su acceso a cualquier tipo de prótesis o tienen la falta de los recursos y/o conocimientos necesarios para la realización de una mano de estas características.

1.4. Marco regulador

Este trabajo de fin de grado, como se ha mencionado ya, hace un gran énfasis en la modalidad de la propiedad intelectual, ya que trabajamos siempre bajo el paraguas de la filosofía del *open-source* [1,2].

Todo software normalmente está constituido por un código fuente el cual siempre, al igual que un libro, tiene un autor y por lo tanto unos derechos o copyright. El software, al ser una parte fundamental del mundo en el que vivimos ya que es una parte esencial de muchos de los aparatos que usamos cada día, está suscrito a unos derechos comerciales. A diferencia de los libros, aunque la autoría es del creador del código, dichos derechos comerciales son mantenidos por la empresa u organización bajo el cual han sido creados.

Como cualquier otro derecho comercial sobre un bien, el software está bajo los derechos de reproducción, transformación, distribución y publicación, pudiendo ser este explotado comercialmente. Al igual que cualquier otro bien de una empresa, normalmente la copia o distribución de dicho software se encuentra restringida, o simplemente no puede ser explotado comercialmente por terceros. En resumen, el software puede ser tratado como cualquier otro bien comercial.

El *open-source*, o código abierto, sacrifica los derechos comerciales exclusivos de la obra, siendo ésta libre de explotación y modificación. Existen varios tipos de licencias *open-source*, siendo las más habituales las que obligan a las nuevas publicaciones (modificadas o no) a utilizar la misma licencia que la usada en la obra original, dando crédito al autor de ésta.

El uso de licencias libres no supone la renuncia de los derechos de autor ni de los derechos comerciales de las obras publicadas, ya que es posible (y se hace) realizar actividades lucrativas, siendo únicamente obligatorio el uso de las licencias mencionadas anteriormente.

En nuestro proyecto hacemos uso de la licencia *GPL v3* en cuanto al código, dando libre uso de usar el software de cualquier modo, hacer los cambios pertinentes y de los derechos de distribución [3].

En cuanto a las piezas, elementos electrónicos y documentación, utilizamos una licencia del tipo *Creative Commons Attribution-ShareAlike 4.0 International License*. Esta licencia establece la libertad de copia y distribución en cualquier plataforma y/o formato, junto al libre derecho de uso, transformación y comercial. Las únicas restricciones dadas son que el uso de obras (sujeto a transformaciones) con este tipo de licencias debe ser bajo el mismo tipo de licencia, así como dar crédito al creador original [4].

1.5. Entorno socioeconómico

Para hablar del impacto de la realización de un proyecto de nuestras características, vamos a dimensionarlo en dos partes: el propio impacto que tienen las prótesis avanzadas (es decir, las que no son cosméticas ni mecánicas) y el impacto económico y social de realizar una mano robótica/protésica de un coste bajo [5].

Normalmente, la pérdida de un miembro genera traumas psicológicos severos, ya que son situaciones extremas de anormalidad para las cuales cualquier ser humano no está preparado para ello. Más allá del impacto emocional y psicológico que tiene la pérdida de un miembro, conllevando todos los problemas que ello supone, este hecho causa una estigmatización social sobre cada individuo, el cual pierde gran capacidad de realizar una vida “normal” cuando no es capaz de realizar tareas tan sencillas como abrir una botella. Esto implica que directamente no puede ser partícipe de cualquier puesto de trabajo, siendo parte de la cultura de la vergüenza y siendo excluido como un miembro de la sociedad, ya que incluso algunas personas sienten rechazo frente a individuos con ciertas patologías de este tipo.

Con la llegada de las prótesis robóticas, que permiten hacer tareas de la vida cotidiana de manera más sencilla, llega un impacto psicológico que puede suponer contrarrestar todos los impactos negativos que hemos mencionado anteriormente; además, dado el contexto social en el que vivimos actualmente, donde hay una gran admiración por la ciencia ficción y la tecnología, tanto en la literatura como en las producciones cinematográficas y de entretenimiento, la reacción normal ante un brazo biónico suele de ser de asombro y de interés, dándole al usuario la capacidad de pasar de una cierta marginalidad al protagonismo en escenas sociales. Todo esto sin olvidar que el usuario de una prótesis puede volver a realizar las acciones de una vida normal, siempre bajo alguna restricción por la tecnología actual (todavía no existen prótesis que reproduzcan el funcionamiento de un miembro humano completamente, y examinar los impactos de prótesis así o superiores queda fuera de este análisis).

Uno de los problemas de las prótesis avanzadas, que al no ser productos de uso común y necesidades tecnológicas muy específicas, suele ser su alto precio. Por eso, aparecen iniciativas de realizar prótesis de bajo coste, como la que se presenta en este trabajo. Una de las razones que motivan este hecho es que muchas personas bajo circunstancias de necesidad de una prótesis no disponen de un poder económico

suficiente para costear prótesis de varios miles de euros, y posiblemente se contenten con prótesis menos avanzadas pero que reduzcan drásticamente el precio. Asimismo, una reducción drástica de los precios puede generar el aumento de la demanda de dichas prótesis, y obligar a los competidores del sector a realizar diseños más baratos o lo suficientemente costeables para poder competir con dichos productos, siendo esta la premisa del mercado competitivo que vivimos en nuestros días, que ha llevado al desarrollo de las avanzadas tecnologías que poseemos hoy.

El impacto de este trabajo influye principalmente esta última parte, ya que se centra en el abaratamiento de las manos robóticas (incluyendo a las protésicas) creadas por impresión 3D. Las ventajas frente a las prótesis avanzadas de otros materiales, aparte de la gran diferencia de coste, es que los reemplazos de las piezas solo necesitan del acceso a una impresora 3D para realizar repuestos. Además, en nuestro caso, la mano robótica Dextra además es fácilmente desmontable y fácil de arreglar, por lo que instigamos la cultura del ahorro y la reparación frente a la cultura del consumo.

A su vez, otro impacto que realiza es la democratización de la tecnología y poner el conocimiento al alcance de todas las personas. Al ser un diseño abierto, tanto la parte mecánica, electrónica y la programación están accesibles, a diferencia de la gran parte de los dispositivos tecnológicos que se utilizan normalmente. De esta forma, la gente puede aprender cómo funciona y aplicar estos conocimientos adquiridos en proyectos similares o distintos.

2. Estado del arte

2.1. Precedentes de las manos protésicas

A lo largo de la historia se han desarrollado e implementado varios modelos de prótesis de miembro superior, ya que la pérdida de miembros por enfermedades, accidentes o guerras dejaba mucha gente tullida. Por este motivo, los antiguos inventores ya trataron de recrear prótesis más allá de la función estética, como la mano de *Götz von Berlichingen* en 1504 [6], con un diseño sorprenderte para la época, como puede verse en la Figura 1.

Figura 1: Diseño de la mano de Götz von Berlichngen

Aun así, e inevitablemente, con estas prótesis su única capacidad de actuación era mecánica mediante unos movimientos del usuario. Otro ejemplo lo tenemos en 1860, donde el *Conde de Beaufort*, debido al gran número de amputados que trajeron las guerras de Crimea e Italia al Imperio francés, hizo diseñar una prótesis la cual podía ser controlada mediante un arnés en el hombro, permitiendo una mínima articulación de la mano [7].

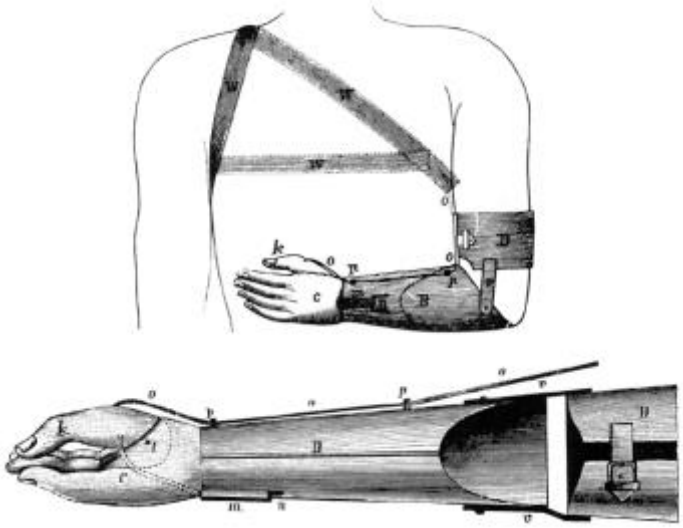

Figura 2: Prótesis diseñada por el Conde de Beaufort

Este tipo de prótesis se estableció hasta entrado el siglo XX, ya que la tecnología para realizar manos actuadas por otros medios que no fueran el propio cuerpo humano (como actuadores eléctricos o neumáticos) aún no estaba disponible o no era lo suficientemente madura. La mayor barrera que tenían los ingenieros era el tamaño desproporcionado de los motores eléctricos y las baterías (por lo que el peso de la mano era demasiado elevado para poder ser llevada). En el caso de los actuadores neumáticos, la problemática era dónde llevar las bombas para introducir el aire que moviese los mecanismos. Como consecuencia de esto, la mayoría de prótesis fueron pasivas o mecánicas hasta la aparición de los microcontroladores y de las tecnologías que permitieron el desarrollo de actuadores cada vez más pequeños y potentes.

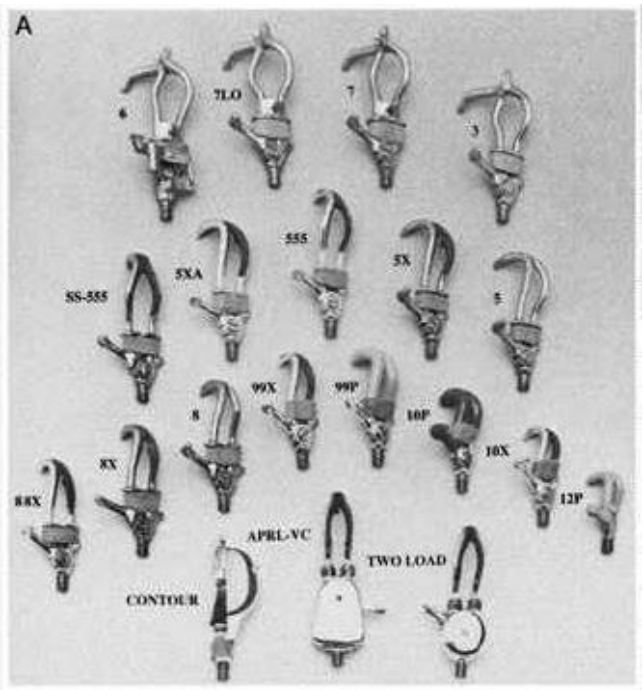

Figura 3: distintos tipos de prótesis de gancho del siglo XX

Junto al interés humano (y económico) de crear prótesis para los amputados, paralelamente en la revolución de la era digital, el desarrollo de la tecnología de las telecomunicaciones con el cada vez mayor aumento de las velocidades de transmisión empezó a ampliar y crear nuevos campos de manipulación remota de sistemas móviles en sectores aeroespaciales (por ejemplo, el control de los robots en misiones espaciales), industriales (cualquier tipo de fábrica), militares (por sus propios intereses) y médicos (la creación de sistemas más precisos para realizar procedimientos quirúrgicos). Esto supuso el interés de crear actuadores eléctricos y mecánicos que luego se beneficiarían las prótesis, ya que algunos de estos sectores les interesa tener una mano funcional humana incorporada a un robot para realizar manipulaciones remotas en cualquiera de los ámbitos mencionados anteriormente.

Con el paso de los años, se ha logrado la tecnología para crear todo tipo de prótesis de manos, con un amplio grados de libertad y numero de actuadores para elegir a elección del usuario, aunque obviamente ligado al precio que escala exponencialmente con la complejidad del sistema.

2.2. Manos robóticas antropomórficas

2.2.1. Shadow Dexterous Hand

Una de las ramas de desarrollo de los manipuladores robóticos se centra en la industria de manipulación para procesos delicados. Para manipular cosas delicadas uno de los mejores manipuladores es una mano humana, debido a su complejidad y rango de movimientos, y a su capacidad de realizar movimientos altamente precisos. Las manos robóticas antropomórficas surgen como una forma de replicar en una herramienta artificial la funcionalidad de este miembro tan complejo.

La Shadow Hand es una mano robótica antropomórfica con 20 grados de libertad actuados, realimentación de posición absoluta y sensores de fuerza, accionada mediante un grupo de tendones actuados desde su base, confiriendo gran precisión y seguridad de movimientos [8]. En la actualidad existen dos modelos de la Shadow Dexterous Hand, en función de su sistema de actuación. Uno de ellos usa *smart motors*: unos actuadores que integran motores de corriente continua con reductora, controles de fuerza y posición y comunicaciones en un módulo compacto, integrados todos ellos en la base de la mano. El otro modelo usa un sistema neumático que integra electrónica de control de posición, electrónica de acondicionamiento de la válvula, colector, sensores de presión y comunicación por 80 válvulas en la base. La mano intenta hacer un diseño antropomorfo lo más cercano posible a la mano humana, aproximando la biomecánica y cinemática de esta.

Uno de los problemas que presenta esta mano es su propia base, ya que es bastante voluminosa debida al gran número de actuadores que contiene. El peso de la mano junto al antebrazo es de un total de 4.3 kg. La mano posee una capacidad de carga de hasta 5kg. Incorpora una serie de sensores táctiles de alta precisión y un control de posición absoluto, para realizar agarres similares a los que realiza un humano.

Esta mano tiene gran relevancia, ya que fue de las primeras manos con un alto grado de precisión y adaptabilidad.

Figura 4: Shadow Hand sujetando una bombilla

2.2.2. Gifu Hand III

La predecesora de la Gifu Hand II, Gifu Hand III [9] es una mano robótica antropomórfica compuesta por cuatro dedos y un pulgar. El pulgar tiene 4 articulaciones con 4 grados de libertad mientras que cada dedo tiene 4 articulaciones con 3 grados de libertad. Cada cuarta articulación está compuesta de un mecanismo de unión de cuatro barras planares.

La diferencia distintiva entre el pulgar y el resto de dedos es que la cuarta articulación del pulgar contiene un servomotor que permite el movimiento perpendicular frente al resto de dedos.

La mano tiene un control PD con una responsividad muy alta, ya que el ancho de banda llega a los 10.4 Hz. Comparativamente, una mano humana se mueve, como mucho, a 5.5 Hz, por lo que Gifu Hand III se mueve mucho más rápido que una mano humana.

La mano incorpora una distribución de sensores táctiles que se distribuyen por la palma y cada uno de los dedos de la mano. En conjunto, todos los sensores táctiles son capaces de detectar hasta 859 puntos. Estos se distribuyen de modo que se cubre la mayor área posible de cada una de las partes, por lo que el área sin capacidad de

medir es del 49.1%. La responsividad de estos sensores es de más de 1 kHz. La carga de trabajo máxima es de 2.2x10-3 N/m2, la resolución de medida de 8 bits y el ciclo de muestreo de 10ms por imagen.

Los resultados experimentales de la mano son una serie de agarres los cuales realizan un par parecido al deseado para cada agarre, con un retardo de 10 ms desde el PC con los sensores táctiles. El ciclo de muestreo del controlador de la mano es de 2 ms.

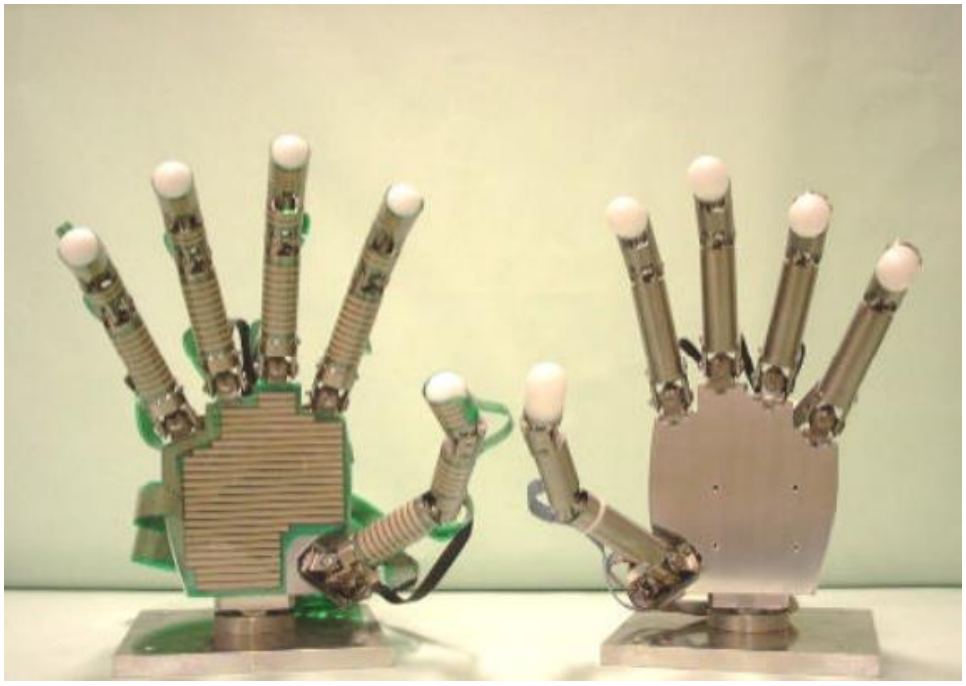

Figura 5: Gifu Hand III con sensores táctiles (izquierda), sin sensores (derecha)

2.2.3. Pisa/IIT SoftHand PLUS

Diseñada por el Centro E. Piaggio de la universidad de Pisa, *SoftHand* es una mano robótica diseñada para la manipulación y el agarre [10, 11].

Figura 6: Representación 3D de la SoftHand PLUS

La mano tiene un único actuador, lo que a priori parece restarle capacidad respecto a otras manos robóticas del mercado, pero su avanzado diseño le permite agarrar casi cualquier objeto. Esto es posible gracias a que la mano tiene 18 ligamentos artificiales y se basa en el funcionamiento de "sinergia blanda" (del inglés, *soft synergies*), que es un tipo de subactuación (menos actuadores que grados de libertad) basado en la combinación de un único motor y principios de control [12, 13]. Esto da como resultado que el agarre no es una posición predeterminada, sino que la mano se adapta al entorno mediante la interacción física con este. A pesar de tener un único grado de actuación, la mano tiene 19 grados de libertad.

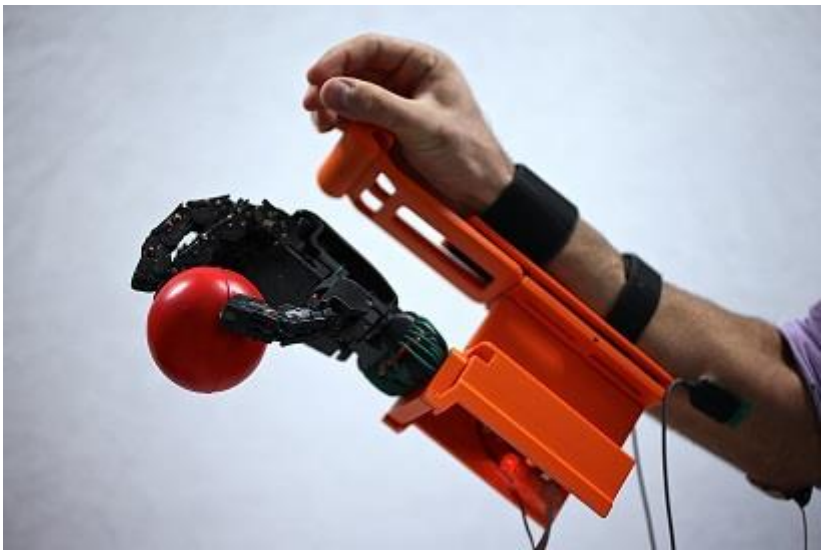

Figura 7: Softhand sujetando una pelota

De momento la mano solo se ha implementado como mano robótica, aunque se está investigando su uso como prótesis gracias al exquisito control que tiene. A su vez, se está desarrollando su siguiente versión, la *PISA/IIT SoftHand 2*, que contendrá dos motores, para que en vez de solo poder realizar el agarre de objetos se pueda realizar la manipulación compleja de nuevos elementos [14].

2.2.4. Problemática de las manos robóticas antropomórficas

La problemática general que encontramos en este tipo de manos suele ser su elevado precio y el volumen que ocupan todos los elementos que forman estas manos, siendo ejemplos de esto la Shadow Hand y Gifu Hand III. Esto hace que en algunas aplicaciones no se pueden emplear de manera práctica y tienen un peso en ocasiones superior a un límite de confort para ser usadas, por ejemplo, como prótesis.

A su vez, los actuadores/baterías suelen ser muy voluminosos, ya que la finalidad de este tipo de manos es de un uso de investigación o de manipulación industrial, o en su defecto, van instaladas en otras plataformas que son capaces de cargar con todos los elementos necesarios (robots humanoides o bases móviles).

Con la finalidad de contrarrestar todos estos problemas, surgen iniciativas como la mano SoftHand que está diseñada para evitar todas estas problemáticas.

2.3. Manos protésicas

Como hemos mencionado anteriormente, el desarrollo de las prótesis eléctricas se vio muy limitado hasta el desarrollo de motores eléctricos pequeños que tuvieran suficiente potencia para poder articular y sostener pesos razonables [15].

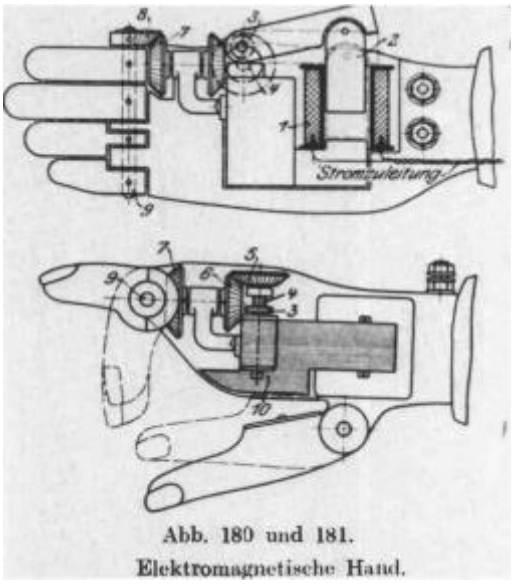

Figura 8: Diseño de prótesis de prótesis eléctrica del siglo XX

Gracias al desarrollo de las tecnologías de los campos mecánicos, físicos, eléctricos, electrónicos y químicos, a día de hoy se disponen de múltiples prótesis avanzadas y otras en vías de desarrollo/investigación que aseguran la aproximación a una vida “normal” de la gente que las necesite. Es notable resaltar el salto tecnológico que supuso la creación de los microprocesadores, fruto de los avances de las tecnologías de fabricación y producción masiva, que permitieron la creación de plataformas tipo Arduino, ya que el coste de algunos microprocesador pasaban de los miles de euros a los 20-30€, a su vez acercando el uso de estos dispositivos a usuarios no expertos, potenciando increíblemente el desarrollo de proyectos personales que luego se vuelcan con la comunidad y se retroalimentan con centros de investigación.

La gran diferencia entre una mano robótica y una protésica es que las protésicas están designadas a reemplazar miembros humanos (como su propio nombre indica) y todas las manos protésicas robóticas son, como su nombre indica, manos robóticas, pero no todas las manos robóticas son manos protésicas. Las manos robóticas están diseñadas para realizar patrones de movimiento humanos pero más orientados hacia manipulación industrial y robots humanoides. A continuación vamos a mencionar varias manos protésicas actuales, analizando brevemente sus funcionalidades y sus características, a vista de hacer una observación objetiva de las mismas.

2.3.1. i-limb

Creada por *Touch Bionics*, esta mano articulada tiene gran variedad de características que la hacen una opción muy buena para cualquier persona que la necesita [16].

Figura 9: Distintos modelos de i-limb

El pulgar se mueve entre la posición natural y de agarre opuesto de una mano (movimientos del día a día de una mano normal). Cada dedo está subactuado: controlado individualmente por un motor que acciona todas sus articulaciones. Incorpora cuatro métodos de control distintos: control por gestos, control por músculos, control por aplicación móvil y control de proximidad. Éste le permite al usuario un gran rango de movimientos, incluyendo la preprogramación de 24 movimientos junto con los 12 por defecto.

Figura 10: Distintos agarres con la mano i-limb

2.3.2. BeBionic

BeBionic, creada por la empresa Steeper, es una de las primeras manos protésicas en salir al mercado con un objetivo comercial de un ciudadano medio, ya que el coste de esta mano ronda los 3.000€. Tiene una amplia gama de tamaños para ajustar el rango a la mayoría de usuarios que la necesiten [17, 18].

Figura 11: Distintos modelos de la mano BeBionic

Incorpora una serie de patrones de agarre preprogramados para hacer actividades de la vida diaria (como agarrar llaves, tarjetas, manejar un ratón de ordenador, etc.). Aunque la mano no tiene funcionalidades avanzadas como sensores de tacto en los extremos como otros prototipos de prótesis, la mano se controla mediante un control de electromiografía e incluye 14 distintos tipos de agarre en total. Cada dedo está actuado por un motor individual. También incluye cuatro tipos distintos de muñeca, ofreciendo desde muñecas simples que simplemente pueden girar 360 grados sobre el brazo, hasta muñecas avanzadas que permiten al usuario una rotación de hasta 30º en cualquier dirección, permitiendo bloqueos en las posiciones de extensión, neutral y flexión. Para proporcionar un movimiento más natural a la mano, cada uno de los dedos realiza un seguimiento del resto de los dedos cada 15 ms, de modo que los dedos ajustan la velocidad constantemente durante la actuación de la mano.

El peso de la mano va de un rango de 369 g para el modelo más pequeño y simple hasta los 690 g para el más grande y complejo. La mano puede soportar cargas estáticas de hasta 45 kg y soporta hasta 90 kg de carga vertical. Algunas posiciones de agarre, como la *key grip* para sujetar llaves o tarjetas, ejercen hasta 36.6 N de fuerza, que es un equivalente de un agarre de 3.73 kg. Los tiempos para abrir y cerrar la mano son de 1 segundo, siendo bastante rápido para el tamaño y características de la mano.

2.3.3. DARPA Prosthetic Arm – Modular Prosthetic Limb

La protésica ha generado interés en todos los campos, tanto médicos, como civiles y militares. Una gran parte del desarrollo tecnológico de nuestro tiempo se encuentra en manos de corporaciones militares, ya que inyectan enormes cantidades

de dinero con el fin de obtener superioridad táctica y técnica sobre el resto de potencias. Aun así, el objetivo de muchos de sus proyectos es el de la recuperación de miembros perdidos para veteranos de guerra, así como de manipulación de entornos peligrosos para reducir el coste de pérdidas humanas, desde situaciones de campo de batalla como en rescate durante desastres naturales [19].

El Modular Prosthetic Limb (aunque no es una mano como tal, el brazo incluye una mano muy avanzada tecnológica) ha sido desarrollado por parte del Applied Physics Lab (APL) de la John Hopkins University con financiación de DARPA (Defence Advanced Research Project Agency) con el objetivo de desarrollar manos y brazos artificiales para soldados amputados [20]. Estos dispositivos deberían poder integrarse de forma natural tanto estéticamente como desde el punto de vista de su control. Por esto deben ser capaces de responder a las señales neuronales del usuario (a través de la electromiografía, o sea, la técnica de medición de las señales eléctricas que descargan las neuronas motoras sobre las fibras musculares provocando su contracción) y además devolver información, como el tacto, al mismo. El resultado de este proyecto ha supuesto un gran desarrollo en el campo de las prótesis, al ser uno de los sistemas brazo-mano más complejos jamás creados.

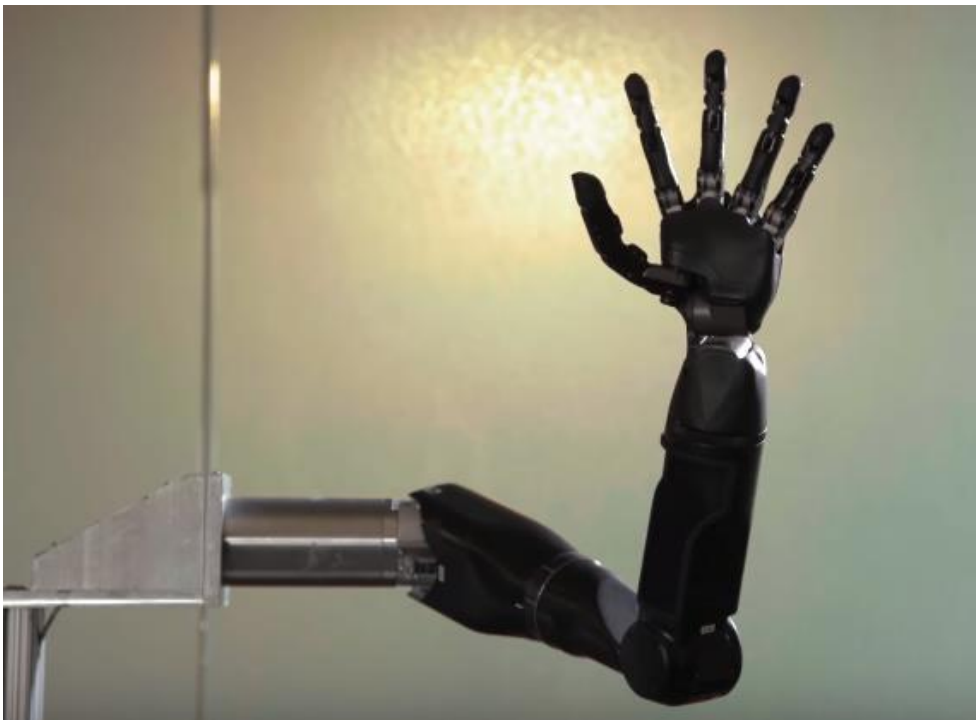

Figura 12: DARPA prosthetic arm

Este brazo consiste en el control mediante 2 sensores *Myo Gesture Control Armband* (es una banda que se compone de 8 sensores de electromiografía), junto a una técnica revolucionaria que se llama reinervación muscular. En estudios, voluntarios con parálisis o falta de miembros han demostrado un control multi-dimensional usando electrodos ligados al cerebro, así como la capacidad de tacto con la prótesis a través de sensores en esta.

Todos los actuadores, electromecánicos, se encuentran en la mano, mientras que la fuente de alimentación está situada en el interior del antebrazo. Contiene un único

driver con 15 salidas para cobótica. La mano está compuesta de una mezcla extensa de distintos materiales de alta calidad (polímeros, fibra de carbono, cerámicas piezoeléctricas). El sonido que produce el funcionamiento de brazo protésico es muy bajo (50 dB). El consumo aproximado es de unos 50 W-h para un día de actividad utilizando una masa de unos 200-300gr [21].

La financiación que ha recibido el proyecto a lo largo de los últimos 10 años asciende a los 120 millones de dólares.

2.4. Manos protésicas impresas en 3D

El mayor problema al que se enfrentan las prótesis robóticas, aparte del desafío técnico, es el factor económico. Las prótesis siguen siendo extremadamente caras (en el rango de los miles de euros a los cientos de miles de euros, o incluso superiores), no pudiendo ajustarse al presupuesto de muchas de las personas que las necesitan, y más teniendo en cuenta que la gente a que le faltan miembros superiores reduce drásticamente su capacidad para obtener un trabajo (debido al impedimento físico que supone) y normalmente las ayudas que reciben son lo suficiente para sobrevivir. Debido a esto, gracias a la expansión de las impresoras 3D, debido al abaratamiento de su precio, y a los avances en la tecnología, empezaron a surgir numerosas iniciativas de crear manos robóticas de bajo coste fabricables mediante impresión 3D, consiguiendo reducir su precio a las pocas centenas de euros.

2.4.1. OpenBionics

Una de los primeros grupos de desarrollo de prótesis que nacieron bajo la premisa del *open-source*. Los creadores de esta iniciativa tienen como objetivo el desarrollo manos robóticas y protésicas baratas, ligeras, modulares y adaptables, que estén compuestas de materiales convencionales y sean fáciles y rápidas de montar. Usando estas técnicas se pueden conseguir dichos tipos de manos por valores inferiores a los 200 dólares y que pesen menos de 300 gramos [22, 23].

Figura 13: Mano robótica desarrollada por *OpenBionics*

Los sistema de actuación y transmisión están inspirados en el diseño de una mano humana, que reproduce los movimientos humanos de flexión (mediante tendones a través de unos tubos) y extensión (mediante el uso de materiales elásticos). La estructura de cada dedo está construida con Plexiglas y las articulaciones mediantes láminas de silicona.

El pulgar está compuesto por un mecanismo que sustituye los 3 grados de libertad del pulgar humano por un solo grado de libertad. El mecanismo propuesto es completamente rígido en cada posición con el fin de no verse afectado por torsiones externas.

La actuación de la mano está realizada por un único motor, el cual acciona cada dedo a través de un elaborado sistema de engranajes. En total, hasta 16 combinaciones se pueden realizar con el cojuntos de los dedos y usando el mecanismo diferencias. Estas, combinadas con las 9 posiciones disponibles del pulgar, suman hasta 144 diferentes posturas de agarres.

2.4.2. In-Moov Hand

Esta mano se enmarca en un proyecto en el que se quiere crear un robot humanoide completo mediante impresión 3D [24].

Figura 14: Vista anterior de la mano de In-Moov

Como el desarrollo se enmarca completamente en el marco de las manos de código abierto, todas las piezas que componen la estructura de la mano son imprimibles con cualquier impresora 3D. El funcionamiento de la mano se basa en 5 servomotores estándar (TowerPro MG995) incorporados dentro del antebrazo, como se puede ver en la Figura 15.

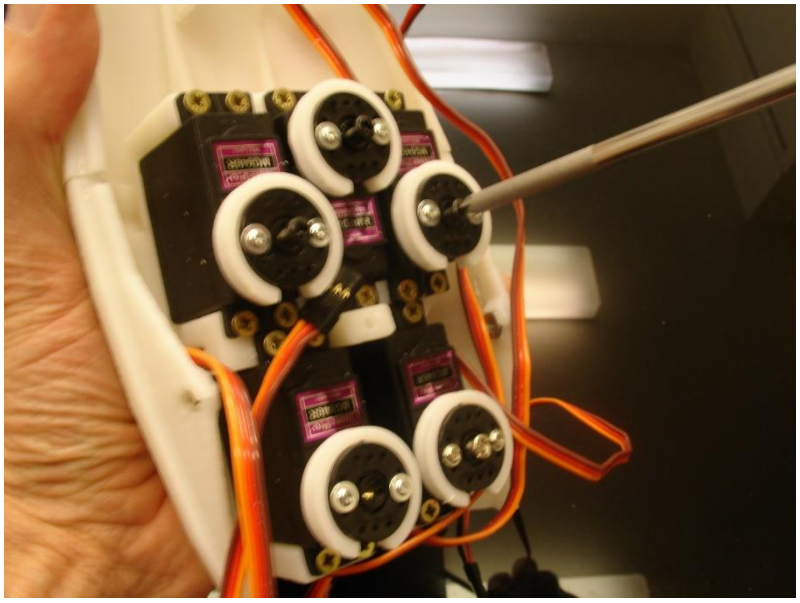

Figura 15: Vista interior del antebrazo de In-Moov con la fijación de los actuadores

Esto supone un problema, ya que el sistema de actuadores es muy pesado y no puede ser usada como una prótesis, pero se incluye en este apartado ya que es una de las primeras manos funcionales impresas en 3D y un referente en este campo.

Cada uno de estos servomotores controla uno de los dedos de la mano a través de un par de tendones artificiales. El control de estos procesos se realiza con una Arduino en la cavidad del antebrazo también.

2.4.3. Open Bionics

Open Bionics fue una de las primeras iniciativas privadas en construirse alrededor de la creación de prótesis de mano imprimibles de bajo coste. También tienen como potenciales usuarios a investigadores y aficionados, contando que cada vez más gente tiene acceso a una impresora 3D. Aunque existen varias manos robóticas o protésicas disponibles en el mercado, tienen precios muy altos. La decisión de formar una empresa de estas características es hacer prótesis biónicas atractivas más accesibles. Los creadores afirman que se pueden realizar prótesis en un tiempo menor de 48 horas y que pueden llegar a costar hasta 30 veces menos que otras prótesis del mercado [25, 26].

2.4.3.1. Dextrus

Dextrus es una de las primeras manos diseñadas para cumplir los objetivos de las prótesis de bajo coste [26]. Su diseño permite que todos los motores y electrónica quepan en la palma de la de la mano, sirviendo así incluso para personas con una amputación al nivel de la muñeca.

Figura 16: Mano Dextrus

La mayor parte de la mano se compone de piezas se harán mediante impresión 3D usando ABS (Acrilonitrito Butadieno Estireno), un plástico muy duradero (por ejemplo, las piezas de lego están hechas de este material). Aunque lo ideal sería usar algunas partes de titanio y fibra de carbono para las partes más importante que no se gasten en unos pocos meses, el diseño y coste de dichas partes no son viables.

El diseño de la mano permite que al agarrar un objeto los dedos se adecuen a la forma del objeto, ya que cada uno de los dedos está accionado por un único tendón artificial que sigue el agarre natural de un dedo humano, permitiendo un agarre firme. Cada tendón es un cable de 7 hilos de titanio con una capa de nylon por encima para que se deslicen suavemente que puede aguantar una carga de hasta 18 Kg de peso por dedo.

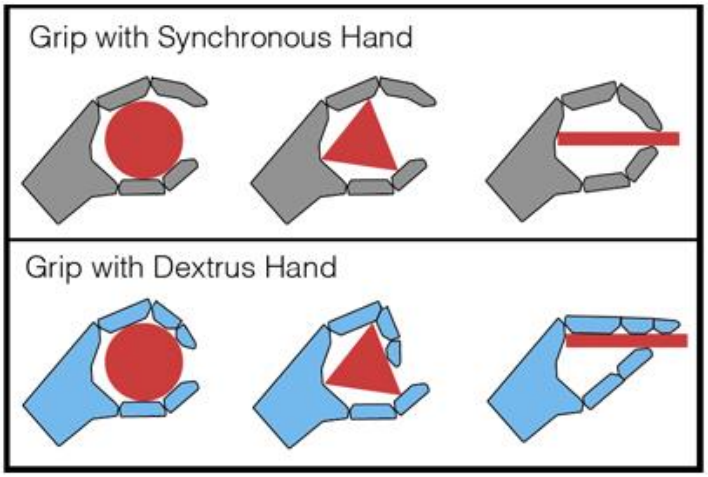

Figura 17: Comparación de agarres mediante una mano robótica de articulaciones acopladas (superior) frente a la mano Dextrus (inferior)

La mano articula cada dedo por separado para cumplir el objetivo de sujetar distintos elementos de la vida diaria (Figura 17), y cada dedo es capaz de reconocer cuando se está agarrando un objeto usando unos sensores de retroalimentación de la mano, haciendo que el usuario solo tenga que pensar en abrir y cerrar la mano gracias a la incorporación de sistemas de control mioeléctrico con la prótesis. La fuerza necesaria para realizar esto se consigue con motores de engranajes epicicloidales o planetarios (consisten en una serie de engranajes externos que giran sobre un engranaje central). Gracias a este ingenioso sistema, la mano Dextrus permite de 8 a 12 horas de uso con cada carga usando baterías de litio.

2.4.3.2. Brunel Hand

Primera mano a nivel comercial de *Open Bionics*, diseñada para gente que quiera introducirse al mundo de las prótesis. Esta mano articulada tiene 9 grados de libertad y 4 grados de actuación [28].

Figura 18: Vista anterior de la mano robótica Brunel

El controlador de la mano está basado en el entorno de Arduino, haciendo que pueda ser usada y programada tanto por expertos como por aficionados a la robótica. La estructura está totalmente impresa en 3D, haciendo que sea fácilmente replicable. Al estar hecha de PLA (poliácido láctico) es bastante ligera (371 g en su conjunto). Su sistema de actuación está basado en servomotores lineales, ubicados en el interior de la zona de la palma, y tendones de hilo de pescar trenzado. Para aumentar la adherencia de los agarres usa almohadillas de uretano en los dedos y una capa de este material en la palma. La mano puede ser controlada desde un PC a través de una conexión USB o usando un *MyoWare sensor*, un sensor mioeléctrico de bajo coste que mide las señales eléctricas generadas en la contracción muscular.

2.4.3.3. Hero Arm

Hero Arm es la última creación de *Open Bionics*, siendo un brazo biónico con multi-agarre que los creadores afirman que cuesta la mitad que sus competidores. Una de las características que diferencia esta prótesis de los anteriores diseños de esta empresa, es que es un dispositivo puramente comercial que no ha sido publicada bajo una licencia *open-source*, ya que la mano no es modificable ni su código fuente está disponible para realizar tareas de desarrollador [29, 30].

Figura 19: *Hero Arm* agarrando un vaso

El brazo incorpora sensores mioeléctricos para usar las contracciones de los músculos del usuario como señales de control que permiten abrir y cerrar los dedos. El diseño del brazo está adecuado a cada paciente, haciéndolo lo más cómodo posible. Además es personalizable, pudiendo darle un diseño especial cosmético haciendo que el efecto psicológico de llevar una prótesis sea incluso algo positivo. Una de sus características principales es que puede ser usado por niños a partir de 8 años.

La mano tiene una amplia capacidad de agarre, pudiendo coger un gran número de objetos normales de la vida cotidiana. Dispone de una función que deja los dedos en posiciones estáticas, para que agarrar un vaso o una bolsa de compra no suponga una fatiga importante por tener realizar una contracción mantenida. La muñeca puede girar 180 grados. La velocidad de los dedos se puede regular con un control proporcional para agarrar objetos delicados (por ejemplo, un huevo).

El peso de la prótesis es menor de 1 kg, y puede levantar hasta 8 kg. La batería de larga duración da un día entero de vida al brazo antes de cargarlo, convirtiéndose en una de las prótesis más versátiles del mercado.

2.4.4. Dextra

Dentro de las prótesis hechas mediante impresoras 3D, saliéndonos de las organizaciones o empresas, numerosos desarrolladores de manera individual han tomado los principios del *open-source* y han creado sus propios proyectos.

Este es el caso de la mano robótica Dextra, una mano imprimible de tamaño humano que ha sido creada por un ingeniero español de la Universidad Carlos III de Madrid, Álvaro Villoslada, como parte de un proyecto que se ajusta perfectamente a los objetivos de este trabajo de fin de grado, que es la realización de una prótesis robótica de bajo coste y de *open-source*.

Las claves de la mano, y la razón por la que la hemos elegido para nuestro trabajo, son la capacidad de agarre adaptativo, el tamaño compacto, la sencillez mecánica y la facilidad de replicación y de modificación, así como su bajo coste de fabricación. La mano es completamente modular, lo que es una gran ventaja para trabajar con ella.

No entramos más en detalle de esta mano ya que al basarse nuestro trabajo en este diseño, haremos hincapié en cada una de sus características a lo largo del proyecto.

2.1. Síntesis estado del arte

Como hemos podido observar durante nuestro proceso de investigación para analizar el estado actual de las manos robóticas, en especial en las protésicas, en estos últimos años es una tecnología que ha alcanzado un gran desarrollo, pero que aún tiene mucho camino por recorrer.

Se ha visto que tanto en la parte comercial como en proyectos *open-source*, hay una gran variedad de manos y prótesis. Los modelos más avanzados tienen un coste muy elevado, en ocasiones inasumible por el ciudadano medio. Gracias al trabajo de algunos grupos de investigación e iniciativas personales de aficionados a la robótica, están surgiendo prótesis cada vez más baratas y con unas características cada vez más próximas a los dispositivos comerciales.

Esto, en gran medida, es gracias al abaratamiento de los componentes mecánicos y electrónicos y al mayor acceso a tecnologías de fabricación de bajo coste como la impresión 3D, ya que incluso permiten fabricar estos dispositivos en casa, con una reducción drástica de los costes de material y del tiempo de realización de estos. Estas tecnologías también permiten abaratar el coste del mantenimiento de las prótesis y simplificar su reparación, ya que ante la rotura de una parte de la mano, la pieza afectada se puede reemplazar fácilmente si se tiene acceso a una impresora 3D. A su vez, el tener acceso al código y poder realizar una reprogramación de una mano da más libertad de uso al usuario, pudiendo este programar su comportamiento a medida.

Estos avances sobre las manos robóticas imprimibles no solo son útiles para cualquier usuario con un poder económico bajo en nuestro entorno socio-económico, sino que tienen un gran impacto sobre países en vías de desarrollo, ya que, debido

principalmente al coste, el acceso a recursos como prótesis avanzadas es prácticamente imposible.

Realizando una comparación de las opciones presentadas durante este apartado (yendo éstas en orden cronológico), no existe una opción que supere a todas las demás, ya que cada una tiene unas características que deberán de ser contempladas y probadas por cada usuario de una prótesis, para comprobar si esta se adapta a las necesidades de cada persona.

3. Desarrollo del proyecto: Mano Dextra

3.1. Descripción

La mano robótica Dextra, como hemos mencionado brevemente anteriormente, es una mano que sigue las premisas del *open-source*, por lo que es totalmente modificable y todos los diseños necesarios para su fabricación y montaje y el código necesario para su manejo son accesibles para cualquier usuario. Esto nos permitirá la construcción de la mano desde cero, realizando la impresión y montaje de todas las piezas, para posteriormente realizar nuestra propia programación para darle la funcionalidad deseada, que detallaremos en el apartado correspondiente.

Dextra es una mano robótica (que puede ser utilizada como prótesis) compuesta por cinco dedos en la misma disposición que una mano humana, para emular su funcionamiento de la manera más fiel posible. Cada dedo se encuentra subactuado por un micromotor de corriente continua con reductora incorporada, el cual tiene acoplado en su eje un carrete que al girar, enrolla un tendón artificial alojado a lo largo del dedo y sujeto a la última falange de este. Al enrollarse el tendón, este se acorta, tirando del extremo del dedo provocando su cierre. Los tendones están constituidos por hilos de sedal de pesca, para proporcionar una gran resistencia a la tensión y evitar que se partan durante su uso. La extensión de cada dedo es pasiva, siendo una serie de gomas elásticas situadas en la parte superior de cada articulación las que generan la fuerza que hace que el dedo se abra cuando el motor desenrolla el tendón.

Los motores cuentan con unos *encoders* para medir su posición angular. Esta medida se usa como señal de realimentación del control de posición que se ejecuta en la placa de control. Estos *encoders* también conectan la entrada de alimentación de los motores con su *driver* correspondiente, alojado en la placa de control de la mano. Esta placa cuenta con un microcontrolador que ejecuta los lazos de control de posición y controla los *drivers* de los motores.

El pulgar se encuentra alojado en una pieza que hace de abductor de la mano, responsable de mover el dedo pulgar de manera perpendicular frente al resto de la mano. La abducción está controlada mediante un servomotor.

Todos los elementos de la mano se unen mediante tornillos, por lo que no es necesario ningún tipo de adhesivo en su montaje. Una de las principales características de la mano es la modularidad de esta, ya que el diseño del conjunto motor-dedo es el mismo para los cuatro dedos de la mano y para el pulgar (con ligeras variaciones), haciendo que los dedos sean intercambiables y se reduce el número de piezas únicas que se usan en la fabricación y el montaje.

Por ejemplo, el conjunto de un dedo (las piezas del dedo junto a la parte que aloja el motor con su *encoder* y conexiones), se encuentra unida al resto de la mano mediante dos tornillos, por lo que en caso de rotura de un componente o si se desea realizar una operación con dicho dedo, este es fácilmente extraíble de la mano sin necesidad de desmontar todo el dispositivo. Este factor es muy importante, ya que simplifica la fabricación y montaje de la mano y facilita su reparación.

La implementación de unos nuevos conectores de tipo IDC (*insulation-displacement connector*) permitirá realizar el conexionado de la mano de manera más eficiente.

3.2. Impresión 3D de la mano robótica Dextra

3.2.1. Impresora utilizada

Para la realización de este proyecto, se ha utilizado una impresora da Vinci Mini de la marca XYZ. Las características generales de la impresora son las siguientes:

- Plataforma de impresión de aluminio de 5.9”x5.9”x5.9”, que equivale a 15x15x15 cm
- Tecnología de impresión: modelado por deposición fundida
- Grosor de capa de impresión según velocidad:
  - Fina: 0.1 mm
  - Estándar: 0.2 mm
  - Rápida: 0.3 mm
  - Ultrarrápida: 0.4 mm
- Cabezal de impresión: Boquilla simple
  - Diámetro de la boquilla: 0.4 mm
- Filamento:
  - Material: PLA (poliácido láctico) derivado del almidón de maíz
  - Diámetro del filamento: 1.75 mm
- Conectividad: USB 2.0/ Wi-Fi (802.11 b/g/n)
- Formatos soportados: archivos .stl, formato XYZ (.3w), 3mf

3.2.2. Software 3D – Piezas utilizadas

Para realizar la impresión de las piezas, se ha utilizado el software *XYZware* de la marca XYZ, que es el software de los fabricantes de la propia impresora.

Los archivos .stl para realizar la impresión de las piezas están disponibles en cualquiera de las webs de licencia abierta donde el creador las ha incorporado, como GitHub, Hackaday y Thingiverse [31, 32, 33].

Vamos a ver todas las piezas que vamos a utilizar dentro del entorno de *XYZware*, como su equivalente de la mano humana, así como su función:

Elemento *abductor*:

El abductor, tal como indica su nombre, se refiere al músculo abductor corto del pulgar, situado en la parte anterior de la mano, siendo un músculo que recorre lateralmente desde la base de la muñeca hasta alojarse en la base primera falange del dedo pulgar [34].

Su función es realizar la abducción del pulgar, que es el movimiento perpendicular del pulgar respecto a la palma de la mano. También es capaz de realizar la extensión y flexión del pulgar para alejarlo/acercarlo a la mano.

En nuestro caso, nuestra mano robótica solo cumple la primera función, la del desplazamiento perpendicular del pulgar, ya que realizar la extensión y flexión del

abductor respecto a la mano requeriría otro actuador, lo cual aumentaría el tamaño y complejidad del diseño existente.

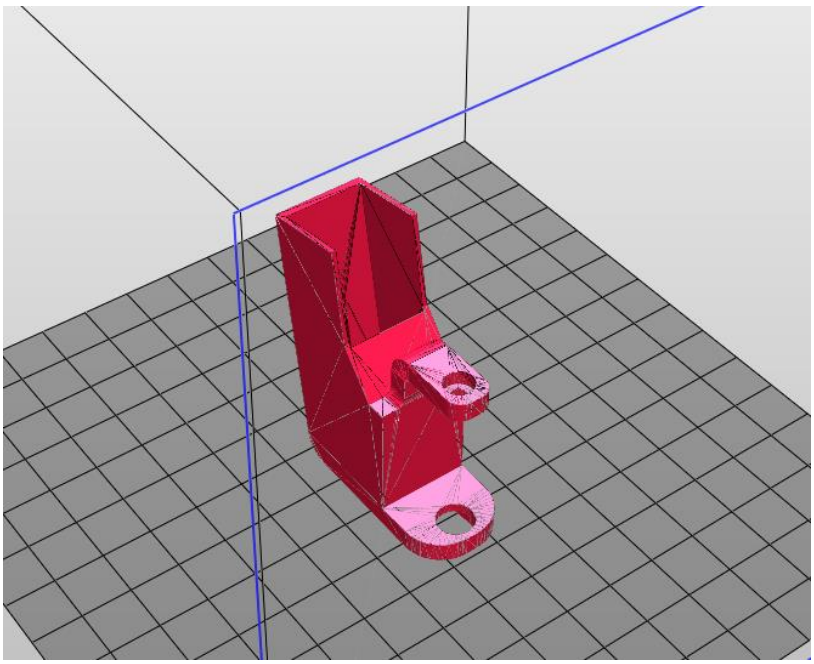

Figura 20: Representación *abductor* en el software XYZware

Elemento *servo_housing*:

Como hemos dicho en el apartado anterior, el abductor permite el movimiento perpendicular de la mano. Para realizar ese movimiento necesitamos un elemento activo, un actuador, que en este caso será un servomotor compacto comercial (modelo Turnigy TGY-EX5252MG Twin BB Digital Micro Servo).

Para alojar el servomotor y que este realice únicamente el movimiento perpendicular sin holgura alguna en otras direcciones, se atornillara dicho servomotor a una pieza (*servo_housing)* que irá unida al dorsal de la mano y a su vez servirá de estructura de movimiento al abductor de la mano, ya que el servo en esta posición moverá el conjunto del abductor con el dedo pulgar.

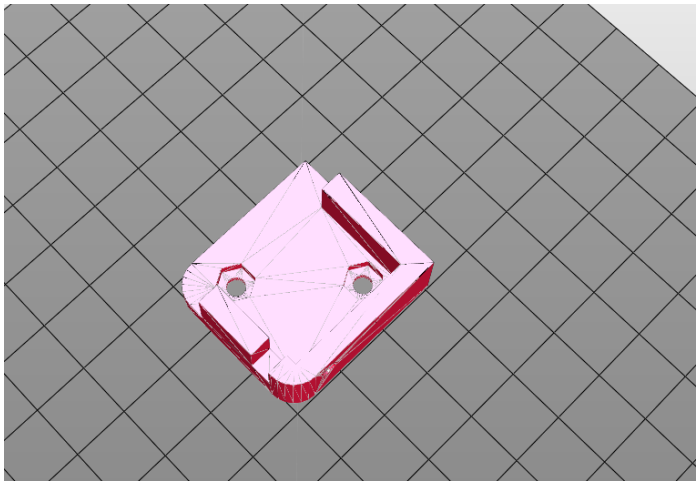

Figura 21: Representación *servo_housing* en el software XYZware

Elementos *distal, middle y proximal*:

Todos los dedos de la mano humana, a excepción del pulgar, se componen de tres huesos, los cuales son desde la punta a parte inferior: falange distal, falange media y falange proximal, las cuales están unidas por articulaciones interfalángicas. El proximal se aloja en el hueso metacarpiano correspondiente de cada dedo. La función de estos huesos es formar la estructura de cada dedo, para poder agarrar los objetos y que no se deforme la mano (con ayuda de los músculos) [35].

En la mano robótica, las piezas correspondientes cumplen la misma función que las falanges humanas, prescindiendo de la estructura exterior de algunos músculos ya que la mano se sostendrá estirada sola gracias a unas gomas en la parte posterior (su equivalente seria el tendón posterior del dedo) que tira de este en todo momento. En nuestro caso, las articulaciones sólo contaran con un grado de libertad (flexión/extensión), ya que no se dispone de la capacidad mecánica para realizar otros movimientos. Dentro de cada una de las piezas que componen los dedos hay unas pequeñas canalizaciones, donde irá alojado el hilo que servirá para transmitir la actuación de los motores a los dedos. Estas canalizaciones están diseñadas de tal manera que la flexión los dedos se realiza siguiendo una secuencia en la que primero se flexiona la falange proximal, luego la medial y por último la distal, de tal forma que la flexión del dedo se adapte a la forma de casi cualquier objeto. Al igual que en la mano humana, el pulgar cuenta con dos falanges: proximal y distal [36].

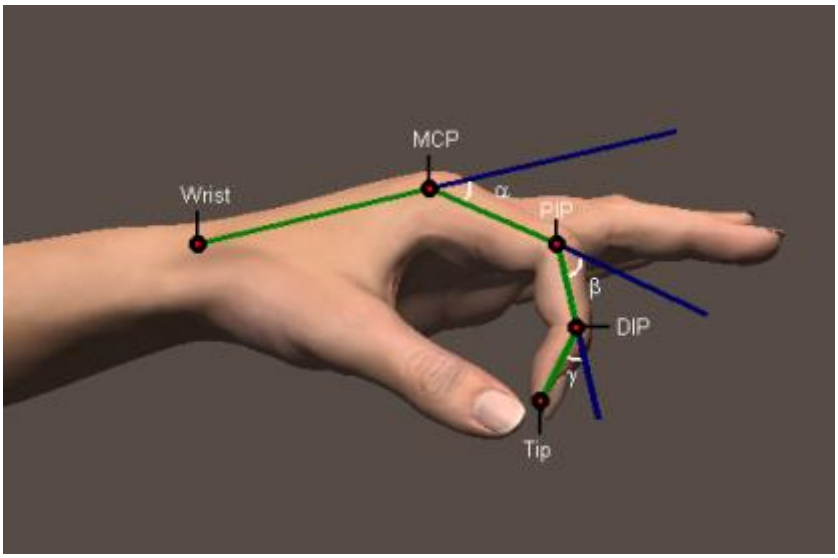

Figura 22: Posición de las falanges de un dedo durante el cierre de la mano

Aunque una mano humana es característica porque la longitud de las falanges no es el mismo para cada dedo (el tamaño del dedo meñique no es igual que el del dedo corazón), debido al diseño modular de Dextra, las falanges con el mismo nombre son iguales para los cuatro dedos; por ejemplo la falange proximal es la misma para el dedo índice que para el anular. De esta forma, los dedos de la mano están diseñados como módulos intercambiables, simplificando su fabricación debido al menor número de piezas individuales. El pulgar, al tener un tamaño muy distinto al del resto del dedo, cuenta con sus propias falanges proximal y distal. Para dar la característica forma de

"montaña" de la mano humana, cada módulo de dedo está colocado a una altura distinta en la pieza dorsal de Dextra.

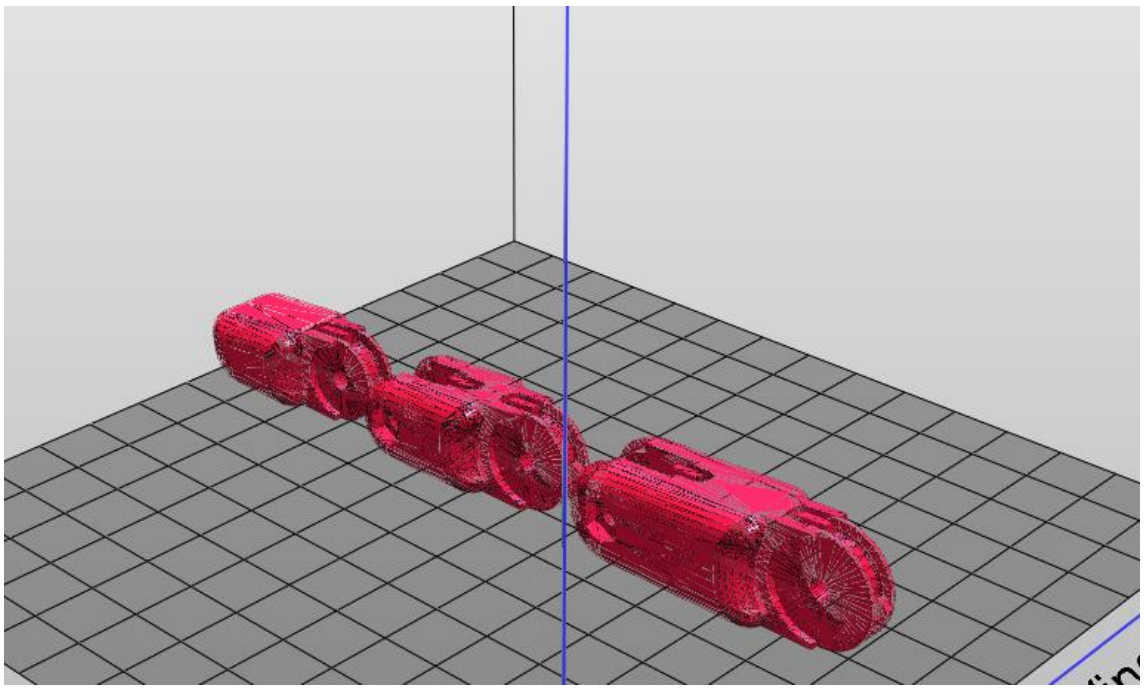

Figura 23: Representación de la estructura de un dedo en el software XYZware (*distal*, *middle* y *proximal*, de izquierda a derecha)

Elemento *motor_holder*:

Los dedos se alojan en la mano humana a través de la unión de la falange proximal con el metacarpiano correspondiente a través de la articulación metacarpofalángica. El metacarpo se compone de cinco huesos largos (los metacarpianos) y forman la estructura del esqueleto de la palma y el dorsal de la mano.

Para la mano robótica Dextra, los dedos se unirán del mismo modo que en la mano humana, es decir, mediante una articulación entre la falange proximal y el metacarpiano, que en nuestro caso es uniendo el elemento *proximal* al elemento *motor_holder*. A su vez, tal como se ve en la imagen inferior (figura 24), nuestro metacarpiano tiene un hueco cuya finalidad es la de alojar el conjunto del carrete, motor y *encoder* que controlan el movimiento de cada dedo.

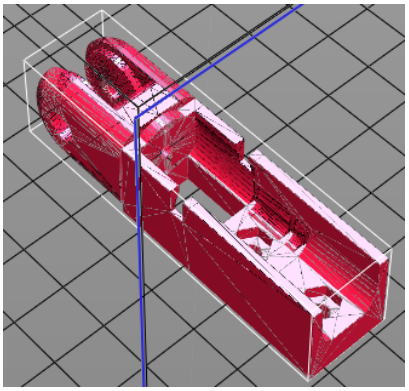

Figura 24: Representación *motor_holder* en el software XYZware

Elemento *spool*:

El control de los movimientos de la mano se hará a través de unos hilos que harán la función de los tendones que realizan la flexión y extensión de los dedos, tal y como hemos mencionado. Para realizar la flexión de un dedo, debemos de recoger el hilo a través de un carrete, la pieza *spool*, que irá unido al eje del motor. De forma inversa, para realizar la extensión del dedo, se desenrollará el hilo girando el carrete en sentido contrario, siendo las gomas elásticas de las articulaciones las encargadas de generar la fuerza que hace que se abra el dedo.

Como se observa en la figura 25, el hilo se introducirá en el agujero que tiene el carrete en el saliente externo, haciendo un nudo en la parte inferior lo suficientemente grande para que no se suelte.

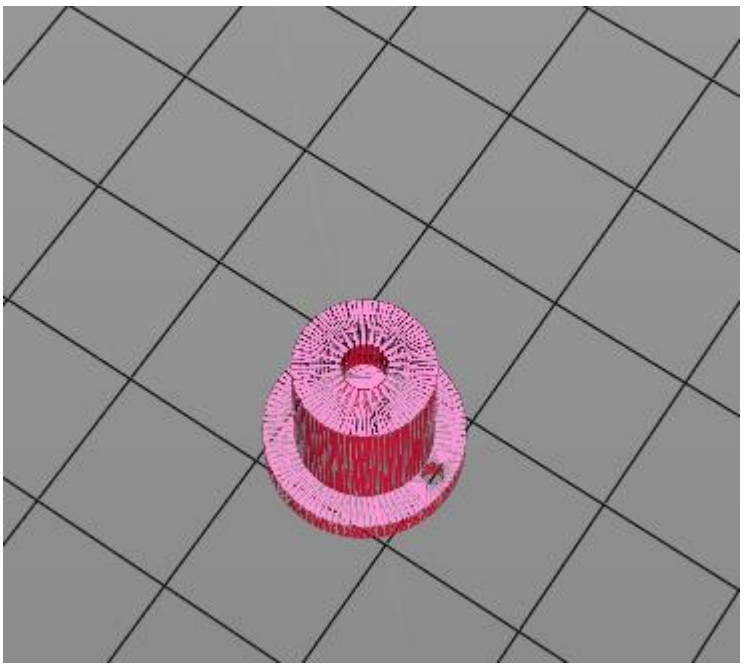

Figura 25: Representación *spool* en el software XYZware

Elemento *dorsal*:

El dorsal de la mano es el elemento estructural al cual se fijan los elementos principales de la mano, los módulos de los cuatro dedos y el abductor con el dedo pulgar. Cada módulo se instala en la pieza dorsal insertándolo en su hueco correspondiente y fijándolo por medio de dos tornillos, uniendo el *motor_holder* de cada módulo y el *servo_housing* del pulgar a la parte interior de la pieza dorsal.

Los orificios rectangulares que se encuentran debajo de los huecos donde se alojan los módulos son para tener una salida de los cables de control y de alimentación del *encoder* y de los propios motores.

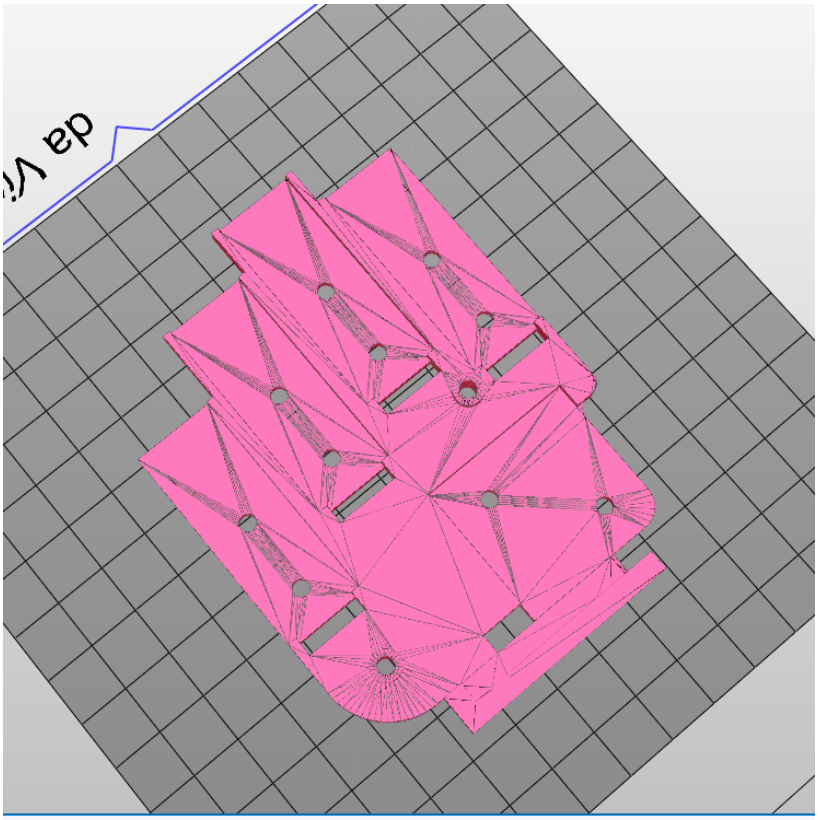

Figura 26: Representación *dorsal* en el software XYZware

Elemento *palm*:

La funcionalidad de la palma de la mano es la de cerrar el conjunto de la mano, protegiendo los motores de los módulos de los dedos. Además, la palma es el elemento que proporciona una superficie de agarre para la manipulación de objetos con la mano. La palma se une al dorsal mediante dos tornillos.

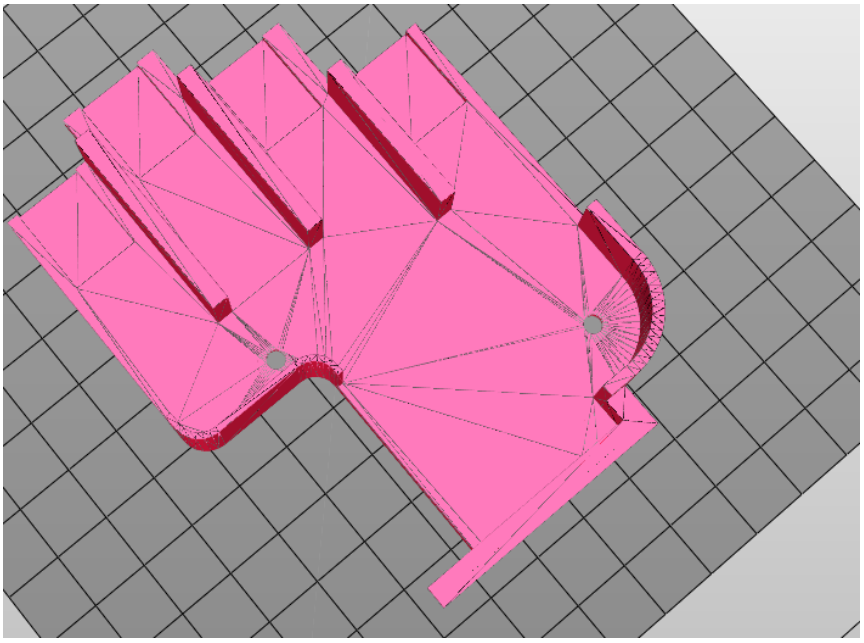

Figura 27: Representación *palm* en el software XYZware

Elemento *wrist*:

La muñeca humana es una de las articulaciones más complejas del cuerpo humano. En nuestro proyecto, la muñeca no tiene una función de articulación; su cometido es el de acoplar la mano a otros elementos (el encaje de una prótesis, el brazo de un robot humanoide…). Además, proporciona solidez estructural a la mano, asegurando la unión entre las piezas *palm* y *dorsal*. Para realizar el acople entre la mano y otros elementos, la muñeca contiene un hueco hexagonal para la cabeza de un tornillo de métrica M10, que es el tamaño estándar las interfaces entre manos protésicas y sus encajes al brazo. De esta forma, la mano es compatible con los encajes protésicos comerciales.

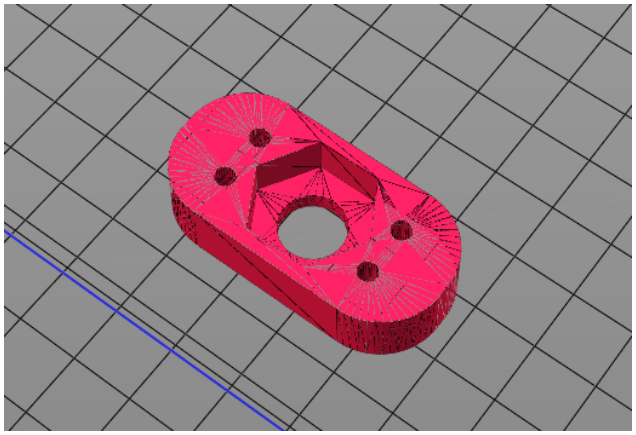

Figura 28: Representación *wrist* en el software XYZware

Elemento *stand*:

Para trabajar con la mano y poder hacer los ensayos y pruebas pertinentes, necesitamos un elemento que permita apoyar la mano verticalmente sobre una superficie. Esta es la función de la pieza *stand*, donde se atornilla el conjunto de la mano a través del tornillo de la muñeca. Esto nos permitirá realizar todas las operaciones y ensayos de manera cómoda con ella.

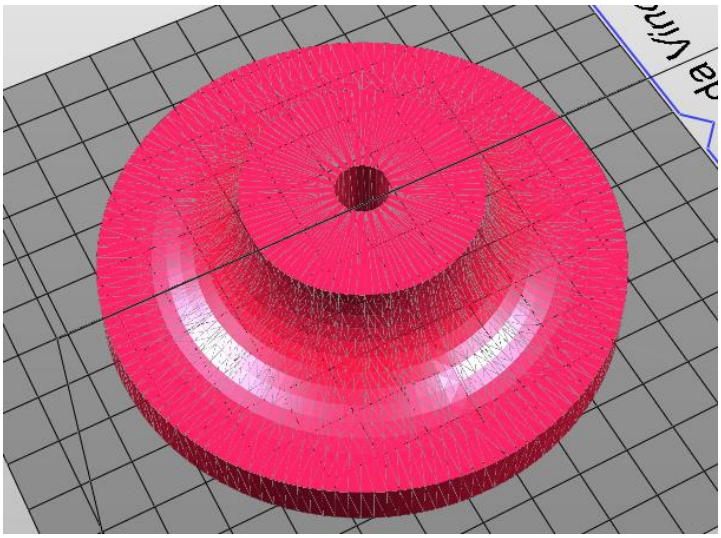

Figura 29: Representación *stand* en el software XYZware

3.2.3. Proceso de impresión

Nuestra impresora da Vinci Mini usa la tecnología de Modelado por Deposición Fundida (llamada comúnmente FDM o FFF, por sus siglas en inglés), que consiste en la deposición de un material previo calentamiento a través de un extrusor, el cual tiene la capacidad de moverse en un espacio de tres dimensiones (en los tres ejes cartesianos), mediante el uso de motores paso a paso. El material situado se deposita en forma de hilos, capa a capa, que se solidifican al contacto con el aire de manera rápida [37], como ilustra la Figura 30.

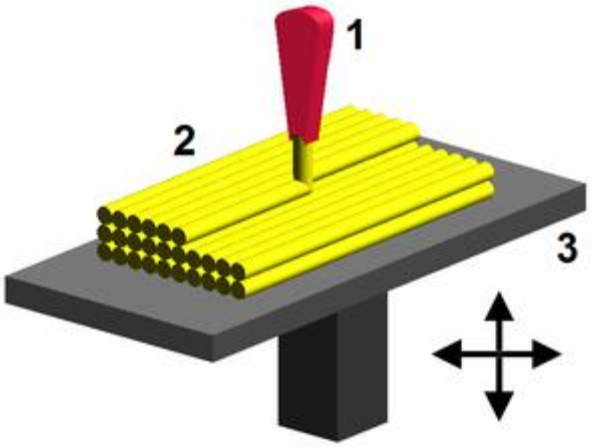

Figura 30: Representación de la técnica de Modelado por Deposición Fundida

Dado que no disponemos de una cama caliente que ayude a la adhesión de las primeras capas a la superficie de la cama, se debe poner una capa de pegamento adhesivo de barra, o algún tipo de laca adherente, la cual aplicamos justo antes de realizar la impresión en el área aproximada donde hemos indicado que se imprimirán las piezas.

Layer shifting

El *layer shifting* es un problema común de las impresoras 3D FDM; supone un desplazamiento de las capas en los ejes X o Y con respecto a la que debería ser su posición según el modelo 3D a imprimir. Este efecto puede verse en la parte derecha de las piezas de la Figura 31.

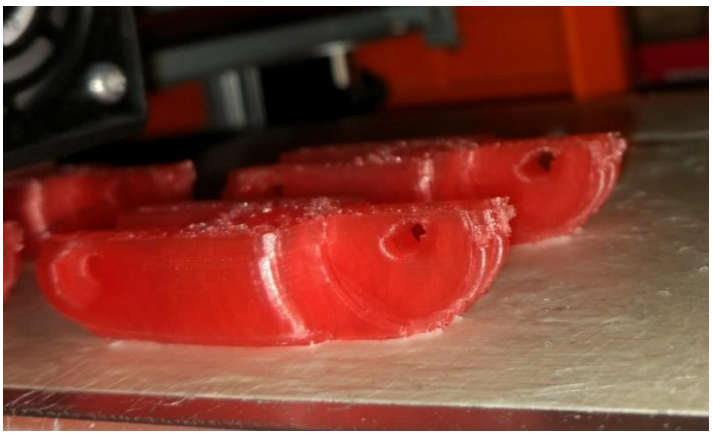

Figura 31: Piezas afectadas por layer shifting

Dicho problema aparece por varios motivos:

- Sobrecalentamiento de los *drivers* de los motores paso a paso de la impresora, que hace que estos se salten algunos pasos, introduciendo una desviación permanente en las posiciones de los ejes correspondientes a dichos motores.
- Error durante el fileteado de la pieza (el proceso por el cual se pasa de un diseño 3D a una serie de comandos de desplazamiento en el espacio cartesiano y de cantidad de extrusión de material).
- Tensionado erróneo de las correas del extrusor, que hace que se desplace involuntariamente hacia uno de los lados.
- Falta de movilidad de la cama de la impresora o del extrusor, ya sea por un problema mecánico, suciedad o falta de aceite.

En nuestro caso, este efecto se producía tras varias horas de funcionamiento y, realizando un análisis de funcionamiento, se descartaron todos los errores menos el primero. Este error se podría solucionar cambiando la electrónica de potencia (inviable en este caso al tratarse de una impresora cerrada y no modificable), incorporando un sistema de refrigeración para esos componentes o no realizar muchas horas seguidas de trabajo para la impresora. La solución fue no proporcionar a la impresora una carga de trabajo demasiado continua, ya que al utilizar la impresora en periodos inferiores a las 2-3 horas continuadas no aparecía este problema.

3.3. Montaje de la mano robótica Dextra

A continuación, se enumeran y describen brevemente los distintos componentes que componen la mano y su electrónica de control y potencia. Además, se documenta de forma detallada el proceso de montaje de la mano.

3.3.1. Componentes de la mano

El listado de componentes de la mano, junto a una descripción de funciones que realizan es la siguiente:

- Motores de los dedos: se han usado cinco *Pololu Micro Metal Gearmotor 1000:1 HP with extended motor shaft*. Estos motores son muy compactos y gracias a su reductora incorporada con una relación 1000:1 son capaces de generar un par máximo de 12 Kg*cm. Incluyen un pequeño eje trasero para poder acoplar un disco magnético para medir la posición mediante *encoders* de efecto Hall.
- Encoders: se han usado cinco *encoders* de cuadratura *Polulu Magnetic Encoder for Micro Metal Gearmotors* compatibles con los motores seleccionados. Estos *encoders* generan dos trenes de pulsos desfasados que permiten saber la dirección de giro del motor. Estos pulsos se generan al pasar por encima de dos sensores de efecto Hall los polos norte-sur alternativos del disco magnético acoplado al motor. Un sencillo código ejecutado por el microcontrolador decodificará estos trenes de pulsos convirtiéndolos en la posición del motor.
- Servomotor: el encargado de actuar la abducción del pulgar es un *Turnigy TGY-EX5252MG Twin BB Digital Micro Servo*, un servomotor digital compacto con reductora de engranajes metálicos, capaz de generar un par de 2.8 Kg*cm.

- Tendones: para realizar la función de tendón artificial se han empleado dos tipos de hilo de pescar distintos. Primero se probó con un hilo trenzado como una mejora con respecto al diseño original, que usaba un filamento de nylon de 0.6 mm de diámetro. El hilo trenzado es más deformable, e inextensible, con lo que se pretendía conseguir una ligera mejora en el rendimiento de la mano, reduciendo rozamiento y mejorando el control. Debido a algunos problemas que se explicarán más adelante, se tuvo que descartar esta opción y usar hilo de nylon de 0.6 mm como en el diseño original.
- Cables y conectores: a diferencia del diseño original en el que se usan unos conectores hembra soldados a los pines de los *encoders*, en esta versión se han usado cables planos soldados directamente a dichos pines, para mejorar la robustez del dispositivo. Para la conexión de estos cables con la electrónica de control se han usado conectores IDC (*insulation-displacement connector*) de seis terminales, por la facilidad de uso con el tipo de cable elegido.
- Elementos elásticos: como ya se ha comentado, la apertura de los dedos se realiza de forma pasiva gracias a unos elementos elásticos colocados sobre cada articulación de los dedos. Se han utilizado gomas elásticas de ortodoncia por su disponibilidad, bajo coste y durabilidad.
- Tornillería:
    - 14 tornillos de M3x14 mm para las articulaciones de las falanges de todos los dedos.
    - 2 tornillos de M3x12 mm para unir el dorsal de la mano a la palma, introduciendo estos en los espaciadores.
    - 10 tornillos M3x8 mm para realizar la unión de cada módulo de dedo al dorsal de la mano, 2 de ellos para unir el módulo del dedo pulgar al abductor.
    - 3 tornillos M3x6 mm para unir el abductor al lugar donde va alojado el servomotor.
    - 2 separadores de M3x12 mm para realizar la unión de la palma con los tornillos del dorsal de la mano.
    - 26 tuercas de M3.
    - 1 tornillo de M10x35 mm para la muñeca.
    - 1 tuerca de M10.

3.3.2. Electrónica de control

- Microcontrolador: el elemento principal de la electrónica de control, encargado de la ejecución de los lazos de control de la posición de cada uno de los dedos, de la lectura de los *encoders* de los motores y de la recepción de los comandos de posición de los dedos por parte de un control de alto nivel (desde un PC u otro dispositivo). Para esta finción se ha elegido una placa de desarrollo *Teensy 3.2 USB Development Board* que contiende un procesador ARM de 32 bits, compatible con el entorno de programación Arduino.
- Drivers: se han usado cinco drivers DRV8838 Single Brushed DC Motor Driver Carrier, consistentes en un circuito tipo puente H que controla la velocidad y dirección de giro en función de las señales de control generadas por el microcontrolador.

- Circuito de alimentación del servomotor: este elemento consiste simplemente en un regulador de tensión para proporcionar una alimentación constante de 6 V al servomotor y dos condensadores de 10 µF y 100 µF para estabilizar las tensiones de entrada y salida del regulador.
- Placa de circuito impreso (PCB): se ha modificado el diseño de la PCB original para poder usar los terminales hembra de los nuevos conectores IDC que incluimos en la mano.
- Fuente de alimentación: en el diseño original, la electrónica de control y los motores de la mano se alimentan por medio de una batería de polímero de litio (LiPo). En este proyecto se ha optado por usar una fuente de alimentación *Mean Well S-25-5* conectada directamente a la red eléctrica, que transforma los 230 V de la red en una salida de 5 V con hasta 5 A. Aunque en el diseño original el circuito se alimenta con una tensión de 7.4 V, en este caso se usa una alimentación de 5 V, lo cual se ha comprobado que es suficiente para alimentar la mano. Debido a la menor tensión, los motores funcionan un poco más lento, haciendo que los movimientos de la mano sean también más lentos.

3.3.3. Proceso de montaje

Aunque las instrucciones de montaje de la mano se encuentran en la web de su creador [31, 32], creemos pertinente detallar todos los pasos seguidos durante el montaje de Dextra, debido a que se realizan cambios respecto al diseño original. Como se están realizando modificaciones que no se encuentran descritas en ningún sitio, es necesario describir el proceso de montaje añadiendo las nuevas implementaciones, de tal forma que la construcción de esta versión modificada de Dextra esté debidamente documentada, siendo este además uno de los objetivos de este proyecto. Se destacarán todos los aspectos que creemos convenientes para que no solo el montaje se realice de forma correcta, sino para prevenir posteriores errores de funcionamiento.

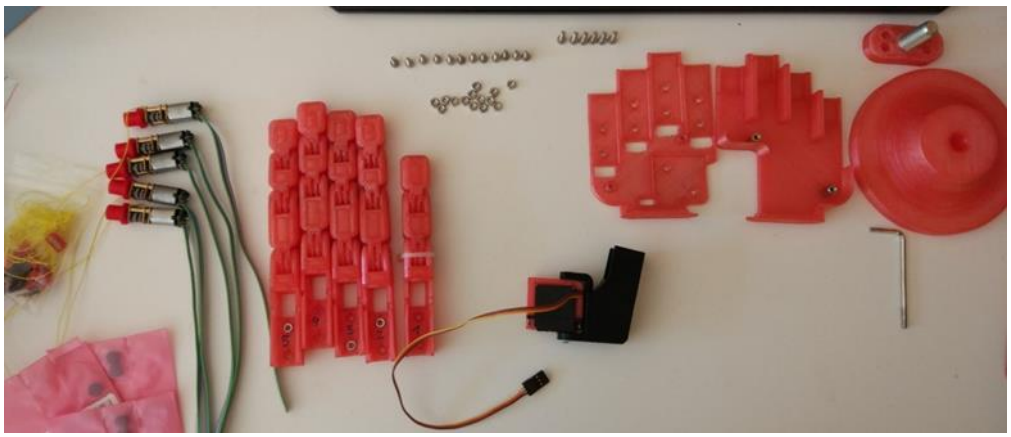

Figura 32: Disposición de todas las piezas que componen la mano Dextra

3.3.3.1. Soldadura de los motores

El proceso de soldadura de los componentes empieza por los cables de control y alimentación de cada *encoder*.

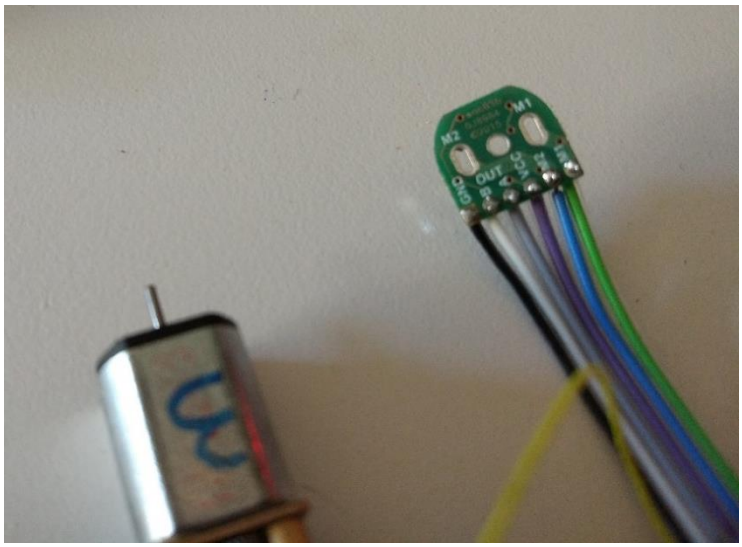

Figura 33: Soldadura de los cables al *encoder*

Como se ve en la Figura 33 se ha seguido el dicho proceso, el mismo que se hará con los otros 4 *encoders*:

- GND (tierra): cable negro
- OUT B (salida de datos B): cable blanco
- OUT A (salida de datos A): cable gris
- VCC (alimentación de los sensores Hall): cable morado
- M2 (alimentación del motor): cable azul
- M1 (alimentación del motor): cable verde

El siguiente paso es soldar el *encoder* al propio motor. Para realizar este proceso usamos un sargento de viga para sujetar el motor y asegurar la sujeción de este. También debemos de sujetar el *encoder* al motor mediante cinta adhesiva, ya que los agarres del motor apenas sobresalen y, al tener los cables soldados, estos realizan una torsión que saca al *encoder* de su sitio.

Por último se introduce el imán del *encoder* en el eje del motor.

3.3.3.2. Dedos

Una vez están los motores soldados, el siguiente paso es montar cada uno de los dedos. El montaje de estos es sencillo, ya que solo hay que unir las falanges, es decir, las piezas proximal, intermedia y distal. Las falanges a su vez van unidas al elemento *motor_holder* que contiene el motor que controla el dedo. Para esto se usan tres tornillos de M3 de 14 mm con una tuerca empotrada, teniendo precaución de no atornillarlos demasiado, ya que en caso contrario puede ejercerse presión sobre la articulación de las falanges, haciendo que estas no giren con facilidad.

Después de tener unidas todas las falanges, se colocan las gomas dentales en sus huecos correspondientes en la parte dorsal de las articulaciones.

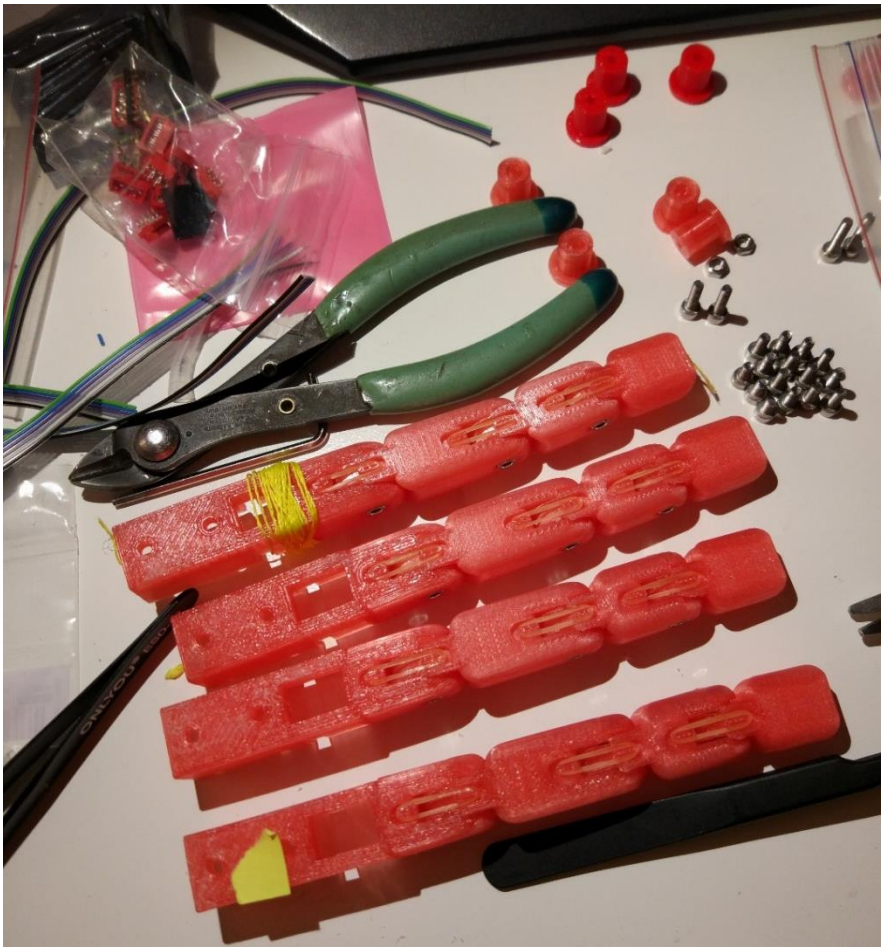

Figura 34: Dedos índice, corazón, anular y meñique montados

Tras esto, se acopla el carrete donde se enrolla el tendón al eje del motor. Se introduce el eje del motor en el hueco correspondiente del carrete hasta que haga tope. En algún caso es necesario el uso de una herramienta para desbastar el interior de este orificio, debido a algún fallo de impresión en la cavidad para introducir el eje. El siguiente paso es fijar el tendón al carrete. Para esto se cortan unos 10-15 cm de hilo para cada dedo y se realiza un nudo en uno de sus extremos. Se pasa el hilo por el pequeño orificio del saliente del carrete hasta que el nudo haga tope.

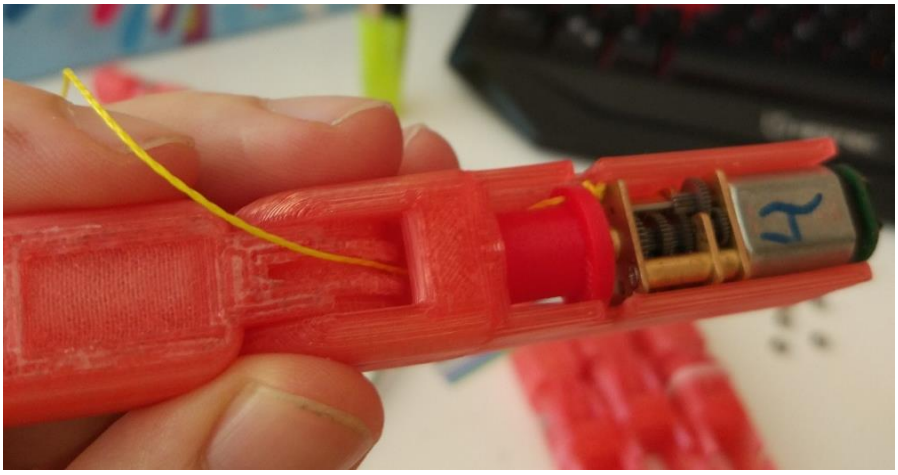

Figura 35: Introducción del sedal de hilo trenzado en el dedo

A continuación, se enruta el tendón por las canalizaciones que tiene cada dedo para ello, asegurando de que el hilo realice una transición suave por los huecos y no se quede atascado con ningún grumo o imperfección que pueda haber, para prevenir complicaciones futuras durante el funcionamiento de la mano. Cuando el extremo del hilo salga por el final de la falange distal del dedo, se realiza un nudo de suficiente grosor para asegurar que hace tope en el orificio de salida de la falange distal y evitar que se pueda deslizar hacia dentro del propio dedo al flexionarlo por la acción del motor.

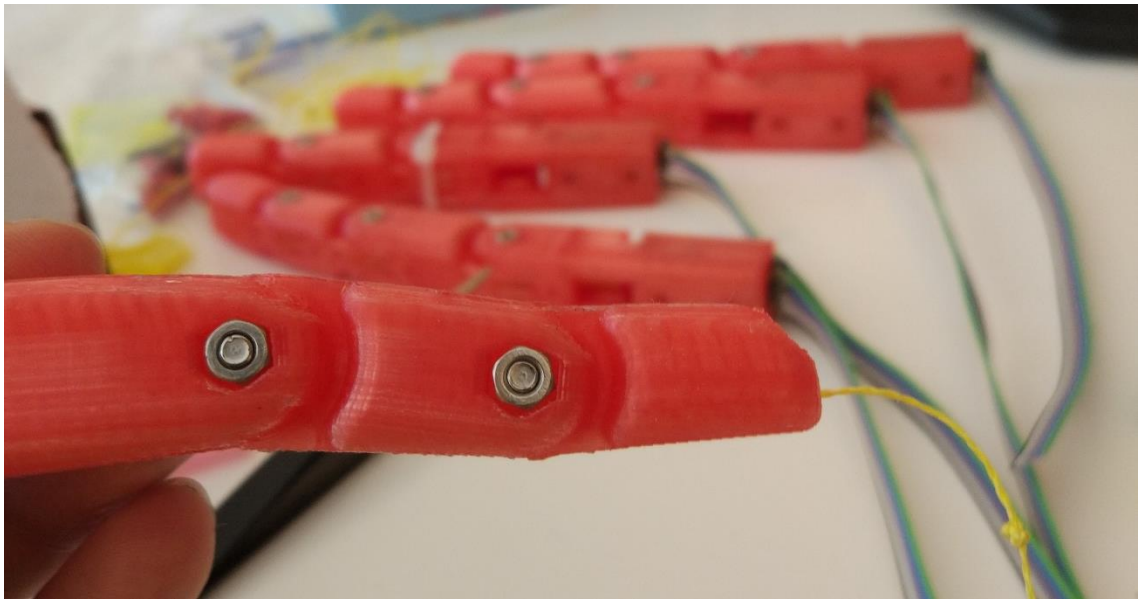

Figura 36: Sedal de hilo trenzado introducido en un dedo con el nudo realizado al final de este

El último paso es introducir el conjunto del actuador (*encoder* + motor + carrete) en la cavidad de cada dedo, concretamente en el elemento *motor_holder*. En alguno de los casos fue necesario lijar la parte superior del carrete un par de milímetros, ya que rozaba una de las superficies de la cavidad del actuador, impidiendo que se introdujera el conjunto del motor de manera adecuada o causando un rozamiento elevado entre el carrete y la pieza *motor_holder*, que impediría o dificultaría el giro del motor.

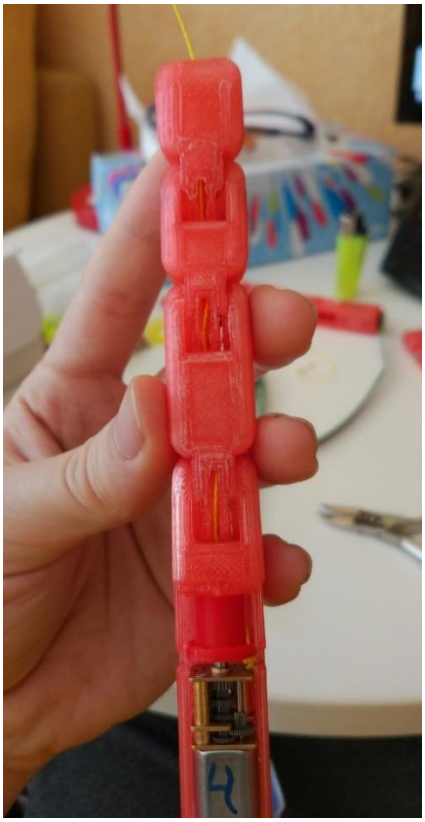

Figura 37: Módulo de dedo completamente montado

3.3.3.3. Dedo pulgar y abductor

El módulo del pulgar se monta siguiendo los mismos pasos anteriormente descritos (con la única diferencia de que tiene una falange menos). Los pasos que se detallan a continuación se refieren al montaje del conjunto abductor.

Figura 38: Vista de los elementos que componen el conjunto abductor. De izquierda a derecha: elementos *abductor, servo_housing* y servomotor

El conjunto abductor incluye la pieza *servo_housing* que aloja al servomotor y que acopla el conjunto al resto de la estructura de la mano, la pieza *abductor* que es el elemento que mueve el servomotor y el módulo del dedo pulgar que se inserta en dicha pieza.

En primer lugar, se introduce el servomotor en la pieza *servo_housing*, de tal forma que el cable del servo salga por la pequeña muesca que hay en la parte derecha de la pieza. El servo se atornilla a los dos pequeños orificios de la pieza los tornillos que vienen incluidos con el propio servo.

A continuación, hay que unir la pieza *abductor* al ensamblaje anterior. Esta pieza se ha modificado, ampliando 2 mm el orificio de salida de los cables del motor del dedo, ya que el diseño original no está preparado para los cables y conectores que se han empleado en este proyecto. Para ensamblar este conjunto, primero se introduce la pieza de acople que viene incluida con el servomotor en su hueco correspondiente en la pieza *abductor*. Se introduce esta pieza en el conjunto *servo_housing* + servomotor, de tal forma que la pieza de acople del servo coincida con el eje de este. Finalmente, se atornilla la pequeña bisagra de la pieza *abductor* a su orificio en la pieza *servo_housing* con un tornillo M3x6 mm.

Una vez ensamblado el módulo del dedo pulgar, se introduce este en su correspondiente hueco en la pieza *abductor*, teniendo especial cuidado de no dañar la soldadura de los cables con el *encoder*, ya que el tamaño para introducir el dedo es justo.

Figura 39: Montaje final del dedo pulgar junto con el abductor

3.3.3.4. Dorsal

Este es el principal elemento estructural de la mano, donde se fijan los módulos de los dedos, el conjunto pulgar-abductor, la palma de mano y la mitad de la muñeca. Al igual que en la pieza *abductor*, se ha ampliado 2 mm el orificio de salida de los cables de cada módulo de dedo para poder pasar los nuevos conectores IDC.

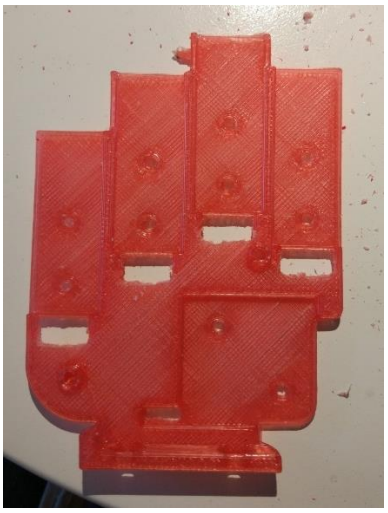

Figura 40: Elemento *dorsal* con los agujeros ampliados mediante la dremel

Una vez realizados los orificios, se coloca cada módulo de dedo en su hueco correspondiente y se fijan a la pieza *dorsal* por medio de dos tornillos M3x8 mm por cada módulo. A continuación, se coloca el conjunto pulgar-abductor en su espacio correspondiente y se atornilla mediante dos tornillos de M3x6 mm. La Figura 41 muestra cómo quedaría el dorsal de la mano por delante cuando unimos todos estos elementos.

Figura 41: Módulos de dedo y conjunto abuctor fijados al elemento dorsal visto de frente

En la parte inferior izquierda (del dorsal) se deberá de quitar un poco de material ya que se produce un gran rozamiento al introducir la palma de la mano, y no permite el cierre adecuado de esta.

3.3.3.5. Palma

La palma de la mano contiene dos orificios donde se atornillas dos espaciadores de M3x12 mm, a los cuales se atornilla la pieza *dorsal*. Debemos de realizar el atornillado de manera delicada, alternando cada uno de los dos tornillos ya que sino la torsión que se crea empujará la palma en dirección contraria al dorsal y no se realizará el cierre de manera correcta.

Figura 42: Palma de la mano añadida, vista de frente

3.3.3.6. Muñeca y base

Con la mano ensamblada, sólo queda acoplar la muñeca y atornillarla a la base. En primer lugar, hay que alojar la cabeza hexagonal del tornillo M10 en su correspondiente hueco. A continuación, se acopla la pieza *wrist* a la mano atornillándola a los cuatro orificios en la base del conjunto dorsal-palma. Por último, se atornilla el tornillo M10 a la base, completando así el montaje de la mano Dextra.

3.3.3.7. Electrónica de control

Utilizando el esquemático del circuito de control que puede encontrarse en el repositorio de Dextra [32], se han soldado todos los componentes que forman la electrónica de control a la PCB de esta, como puede verse en la Figura 43.

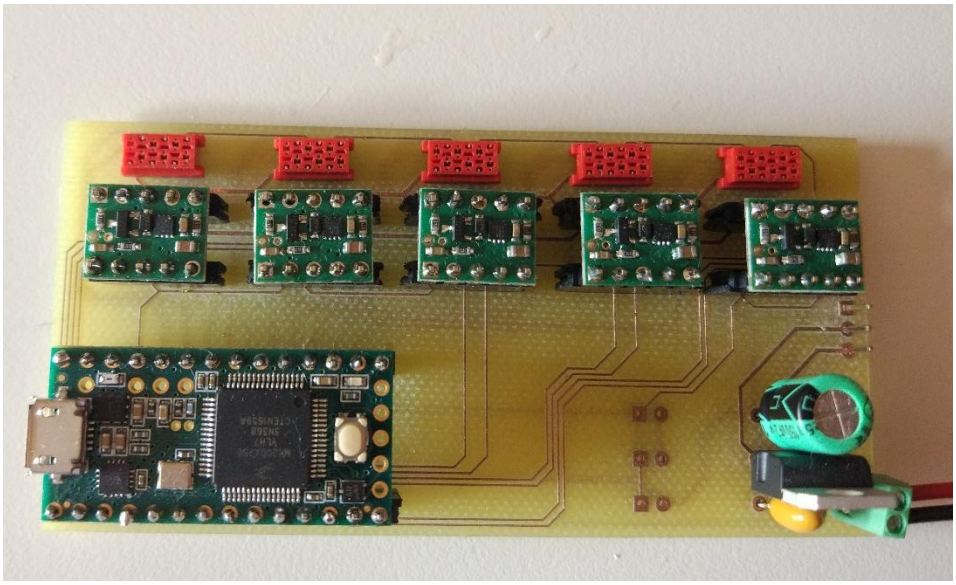

Figura 43: Circuito impreso con todos los elementos añadidos y soldados

4. Control de la mano robótica Dextra

4.1. Control de bajo nivel

En este apartado vamos a analizar el funcionamiento de los elementos que realizan las funciones de bajo nivel, es decir, el control de los elementos de hardware del sistema.

4.1.1. Microcontrolador Teensy 3.2.

Para empezar, se enumeran las características principales de la placa microcontroladora que hemos utilizado:

- Procesador Cortex-M4 72 MHz (modelo MK20DX256VLH7).
- Memoria Flash de 256 KB.
- RAM 64 KB.
- 34 entradas/salidas digitales (3.3 V). Todas soportan funcionalidad de interrupción externa.
- 21 entradas analógicas de 16 bits de resolución.
- 1 salida analógica de 12 bits de resolución.
- 12 salidas PWM.
- Comunicaciónes USB, serie, SPI, I2C y CAN bus.
- Coste: 19€.

Una de las premisas del proyecto Dextra es la facilidad en la programación y control del dispositivo. De esta forma, usuarios no expertos pueden experimentar con su comportamiento y desarrollar nuevas funcionalidades gracias a la posibilidad de reprogramarla de forma sencilla. Esto hace que Dextra tenga también un gran potencial de ser utilizada en un ámbito educativo, ya sea en institutos, universidades o centros de investigación. Por este motivo, el microcontrolador encargado de las tareas de bajo nivel de la mano robótica es compatible con el entorno Arduino.

Arduino tiene una serie de ventajas para la programación de microcontroladores por parte de usuarios no expertos, como son el uso de funciones de alto nivel para poder usar funcionalidades de bajo nivel del microcontrolador de forma sencilla, la existencia de multitud de librerías en las que están implementadas gran cantidad de funcionalidades extra que facilitan al programador el desarrollo de software para estos dispositivos, y la existencia de una gran comunidad de usuarios que permiten encontrar respuestas a casi cualquier duda que se pueda tener a la hora de programar en este entorno.

Las razones de haber escogido la placa de desarrollo Teensy de entre la gran cantidad de microcontroladores compatibles con Arduino, oficiales y no oficiales, son las siguientes:

- Se necesitan un mínimo de 5 pines que soporten la función de interrupción externa, es decir, que ante un cambio en el nivel lógico del pin se ejecute inmediatamente una función determinada. Esto es debido a cómo funciona la lectura de los *encoders*: los pulsos que generan disparan una interrupción que incrementa o decrementa la medida de posición (en función del sentido de giro). Por este motivo se necesitan al menos 5 pines de interrupción, uno por cada

encoder. Algunas placas oficiales Arduino soportan este número de interrupciones externas [36]. Por ejemplo, el Arduino Mega cuenta con 6 pines de este tipo, pero su gran tamaño y su velocidad (16 MHz) descartan su uso.

- Dentro de las placas Arduino de tamaño reducido que cuentan con suficientes pines de interrupción externa encontramos aquellas basadas en el microcontrolador ATmega32U4 (Micro, Leonardo…) tienen 5 pines de este tipo, pero su velocidad (16 Mhz) es insuficiente para la ejecución de todas las tareas de bajo nivel (lectura de los 5 *encoders* a través de las interrupciones, ejecución de 5 lazos de control PID de los motores y comunicación con el PC que envía los comandos de posición).
- Por último, el coste es otro factor determinante. El Arduino Zero o el Arduino MKR1000 no son excesivamente grandes y su velocidad de procesamiento (48 MHz) podría ser suficiente para la ejecución de todas las tareas de bajo nivel. Sin embargo, ambas placas cuestan alrededor de 28€, mientras que la Teensy 3.2 puede adquirirse por unos 19€.

No sólo el coste de la Teensy 3.2 es inferior a placas de características similares existentes con la marca Arduino. Es superior a todas ellas en su funcionalidad, es más potente (72 Mhz pudiendo llegar hasta los 96 MHz), cuenta con más entradas/salidas digitales, analógicas, PWM…, está basada en la popular arquitectura ARM Cortex, etc. Además, su tamaño es muy reducido, lo que hace que sea posible de integrar en un futuro circuito de control que esté integrado en la propia mano.

A través de la página de Hackaday o GitHub del proyecto Dextra se puede descargar el *firmware* de la mano, el código que implementa todas las funciones de bajo nivel como el control, la lectura de sensores y la comunicación. Este *firmware* se programará en la placa usando la herramienta multiplataforma Platformio, un compilador y depurador de código, con gestión de librerías y compatible con multitud de microcontroladores y placas de desarrollo, su principal ventaja [39]. Una vez programada la placa, podemos comprobar el funcionamiento correcto de la mano.

4.1.2. Control de la posición de los dedos

El control de la posición de los dedos se hace mediante un controlador de lazo cerrado tipo proporcional-integral-derivativo (PID). Este control PID tiene un bucle de realimentación en el que se mide la posición del motor de cada dedo por medio de su *encoder*, calcula el error entre el valor medido y el deseado y genera la señal de control en función de este valor. En la mano Dextra, esta señal de control es una señal modulada por ancho de pulso (PWM) con la que se controla la velocidad de los motores de cada uno de los dedos.

La implementación del PID discreto en pseudocódigo es la siguiente:

$$Señal\ control = K_P \cdot Error + K_I \cdot (Suma\ errores + Error) + K_D \cdot (Error - Error_{anterior})$$

donde K_P es la ganancia proporcional, K_I es la ganancia integral y K_D es la ganancia derivativa.

Para asegurar el correcto funcionamiento del PID, este se ejecuta de forma periódica cada 10ms gracias a una interrupción que lanza un *timer* configurado con este tiempo de ejecución.

4.1.3. Protocolo de comunicaciones

El protocolo de comunicaciones permite recibir los comandos de posición de los dedos a través de un puerto serie, por lo que cualquier dispositivo con esta funcionalidad puede comunicarse con Dextra. En este caso la comunicación se realiza desde un PC a través de un puerto serie emulado por USB. Para realizar la comunicación, se usa un protocolo de comunicaciones llamado "Synapse", creado específicamente para esta aplicación. El código de este protocolo, en sus versiones para Arduino y en lenguaje Python, se encuentra disponible en un repositiorio web [32].

El funcionamiento del protocolo "Synapse" se basa en un método denominado *framing*, en el que se envían una serie de paquetes de datos, o *frames*, en formato binario. Dichos paquetes se encuentran delimitados por un byte de cabecera y un byte de terminación, estableciendo el inicio y final de la comunicación. Después del byte de cabecera se encuentran seis grupos de cinco bytes cada uno. Cada uno de estos grupos consiste en un byte de identificación que indica a qué dedo (incluyendo el abductor) corresponde la posición contenida en los 4 bytes siguientes. La posición ocupa 4 bytes ya que es el resultado de convertir un número en punto flotante *float* de 32 bits a binario. Por último, tras los datos posición pero antes del byte de terminación, tenemos un byte de comprobación de la integridad del paquete CRC (*cyclic redundancy check*) que sirve para comprobar que los datos recibidos son los mismos que los enviados. Este byte CRC se calcula realizando la operación lógica XOR sobre todos los bytes del paquete de datos, excluyendo los bytes de cabecera y terminación.

4.2. Control de alto nivel

El control de alto nivel se refiere a la generación y el envío de los comandos de posición desde un dispositivo, en este caso un PC, y la electrónica de control de Dextra. En este proyecto, la generación de los comandos de posición será manual, es decir, se preprogramarán las posiciones deseadas para realizar determinados agarres. Sin embargo, este control de alto nivel podría ser automático, mediante algoritmos de *grasping*, o por medio de un sistema de control mioeléctrico, como los usados en las prótesis robóticas comerciales revisadas en el estado del arte. El envío de los datos de posición se realiza a través de la versión Python de protocolo "Synapse" ejecutándose en el PC.

A continuación, se detallan los distintos pasos que hay que realizar para la programación de un código que envíe una serie de posiciones a la electrónica de control de Dextra. Lo primero que debemos de hacer es importar las librerías que vamos a usar en el programa; en este caso la mencionada librería "Synapse" y también el módulo "time", el cual es un módulo estándar de Python que se encarga de gestionar temporizadores, retardos, etc…

```
1. import synapse as sy #importamos las librerias necesarias, llamamos sy para ab
   reviar
2. import time
```

Lo siguiente es escanear los puertos serie del PC, para identificar el puerto serie al que está conectada la placa Teensy. Todos los puertos serie disponibles en el PC se guardan en una lista (el equivalente en Python a un array), donde cada elemento es a su vez una lista de tres elementos: el nombre del puerto serie, el nombre del dispositivo conectado y un código de identificación.

```
1. serial_port_list = sy.serial_port_scan() #escaneamos los puertos serie del ord
   enador
2.
3. print serial_port_list[0][0]  # Primera lista [0] y primer elemento de la list
   a [0]
4. print serial_port_list[0][1]  # Primera lista [0] y segundo elemento de la lis
   ta [1]
5. print serial_port_list[0][2]  # Primera lista [0] y tercer elemento de la list
   a [2]
6. print "\n"
```

La función *serial_port_select()* muestra al usuario en una terminal los nombres de los dispositivos conectados (almacenados en la variable *serial_port_list*), pudiendo éste elegir a qué puerto conectarse. El puerto elegido se guardará en una variable llamada *serial_port*, que se pasará como argumento a la función *serial_port_connect()* encargada de configurar y realizar la conexión. Esta función nos devuelve un objeto *ser* con el puerto serie y una variable *status* que indica el estado de la conexión.

```
1. serial_port = sy.serial_port_select(serial_port_list)
2.
3. ser, status = sy.serial_port_connect(serial_port)
```

Si la conexión se ha realizado de manera correcta, *status* devuelve un valor *True* (1) y se ejecuta el código principal. Si se devuelve un *False* (0), se imprimirá por pantalla un mensaje de error.

```
1. if status:
2.  .
3.  .
4.  .
5.  .
6.  .
7.  .
8.  .
9. else:
10.     print("Connection error") #Si no conseguimos cominicar con el puerto serie
   , sacamos un mensaje de error
```

Para realizar el control de la mano desde el programa, se envía una lista con seis valores correspondientes al ángulo del abductor y a las posiciones del pulgar, índice, corazón, anular y meñique respectivamente. Para enviar la lista de valores al microcontrolador se usa la función *write_setpoint_list()*. Es necesario pasarle como argumentos objeto que almacena el puerto serie al cual queremos enviar los datos y la lista de posiciones.

```
1.      setpoint_list[0] = 84  # Colocamos el pulgar a 90 grados
```

```
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)
    setpoint_list[1] = 8  # Flexionamos el pulgar del todo hacia atras
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    raw_input("Close")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[84,-8,5.2,6.5,7,6.5]) # Enviamos posiciones
    time.sleep(1)
```

La manera más correcta de proceder para ejecutar cada movimiento es dividirlo en dos o más "submovimientos". Por ejemplo, para agarrar un objeto cilíndrico, como una botella, hay que colocar el pulgar en una posición adecuada antes de cerrar el resto de dedos en torno al objeto. Por esta razón, es recomendable enviar algunas posiciones por separado con retardos de 1 o 2 segundos entre envíos, para dar tiempo a los dedos a completar su movimiento, o bien usar funciones como *raw_input()* que hacen que el código quede a la espera de que el usuario pulse *intro* para continuar su ejecución.

Tras realizar el agarre deseado, se seguirá el proceso inverso para devolver la mano a su posición original. Utilizando *raw_input*, el código esperará a que el usuario pulse la tecla *intro* para iniciar la secuencia de apertura en la que la mano sotará el objeto (en caso de que este agarrando uno) y volverá a su posición de reposo.

```
     raw_input("Open")  # El programa espera hasta que pulses intro
    setpoint_list[2:6] = [0,0,0,0]  # Colocamos el pulgar en su posicion de re
poso
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(3)
    setpoint_list[0:2] = [10,0]  # Colocamos el pulgar en su posicion de repos
o
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)
```

Por último se debe cerrar la conexión serie. Si acaba el programa sin cerrar la conexión, la próxima vez que se quiera conectar a la placa posiblemente dará un error, ya que la conexión estará ocupada (el microcontrolador tendrá asignado ya un puerto serie y se deberá realizar la desconexión física de este).

```
    ser.close() #Cerramos la conexion con el puerto serie
```

4.3. Interfaz de control

Para facilitar el manejo de la mano desde el ordenador, se puede usar la interfaz gráfica de control de usuario (GUI) que viene en el repositorio de Dextra [32]. Consiste en una aplicación que permite controlar la posición de cada dedo y el servomotor de forma gráfica, usando controles deslizantes o introduciendo directamente el valor numérico de la posición deseada, como se puede ver en la Figura 44.

Figura 44: Panel de control de la mano Dextra (Interfaz gráfica de control)

En este proyecto se ha usado esta interfaz para realizar un control manual de la mano, moviendo cada dedo a las posiciones necesarias para realizar los distintos agarres que se han implementado y que se mostrarán en el capítulo de resultados. Una vez completado el agarre, se han utilizado estas posiciones en los distintos códigos que se han desarrollado para realizar los agarres de forma automática.

5. Resultados

5.1. Introducción

En este apartado se van a mostrar los distintos agarres implementados ejecutados por la mano Dextra. Estos agarres tratan de reproducir los definidos en la taxonomía de Cutkosky, que se explica a continuación.

5.2. Principios anatómicos y taxonómicos

Una de las características distintivas de los seres humanos frente a otros animales es la capacidad de uso que tenemos con las manos. La anatomía de la mano humana es única, ya que nuestro pulgar es completamente oponible al resto de los dedos, lo que nos permite realizar una gran variedad de agarres y acciones que requieren de gran destreza y precisión.

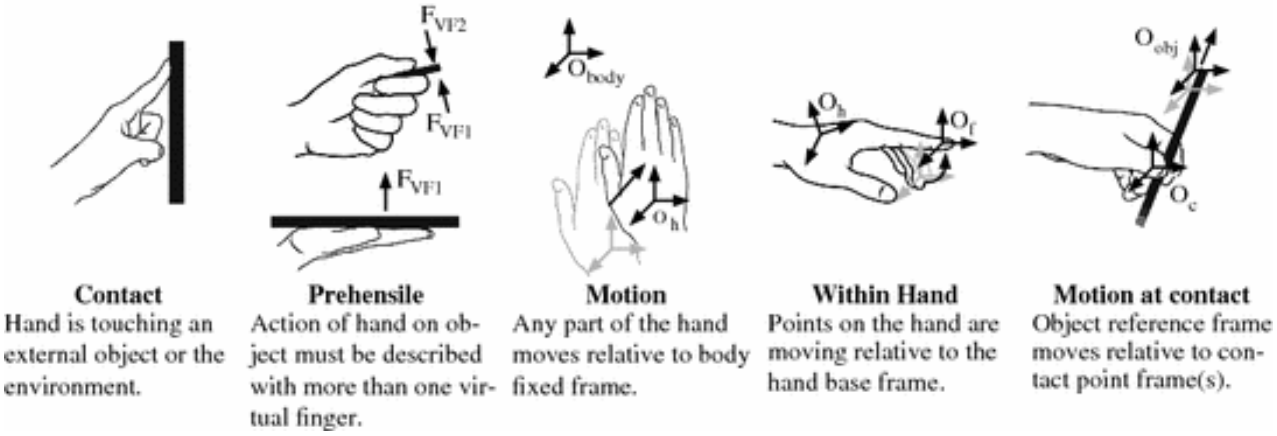

Figura 45: Ejemplos de usos generales de la mano definidos por subclases

Se han dedicado varios estudios al análisis de los agarres que se realizan para llevar a cabo las actividades de la vida diaria, o ADLs por sus siglas en inglés (*activities of daily living*) [40]. Dichas actividades son las acciones básicas que realizamos en nuestro día a día como comer, vestirse, lavarse… Cuando se evalúa la capacidad de manipulación de una mano robótica, especialmente si es una prótesis, es muy habitual comparar los agarres que es capaz de realizar con los necesarios para realizar dichas ADLs.

En este sentido, una de las clasificaciones más usadas para definir los agarres básicos que es capaz de realizar un ser humano y con la que se comparan los agarres que es capaz de realizar una mano robótica es la taxonomía de Cutkosky, que incluye 16 tipos de agarre [40]. Esta será la clasificación que se usará como referencia para la implementación de los agarres en este proyecto.

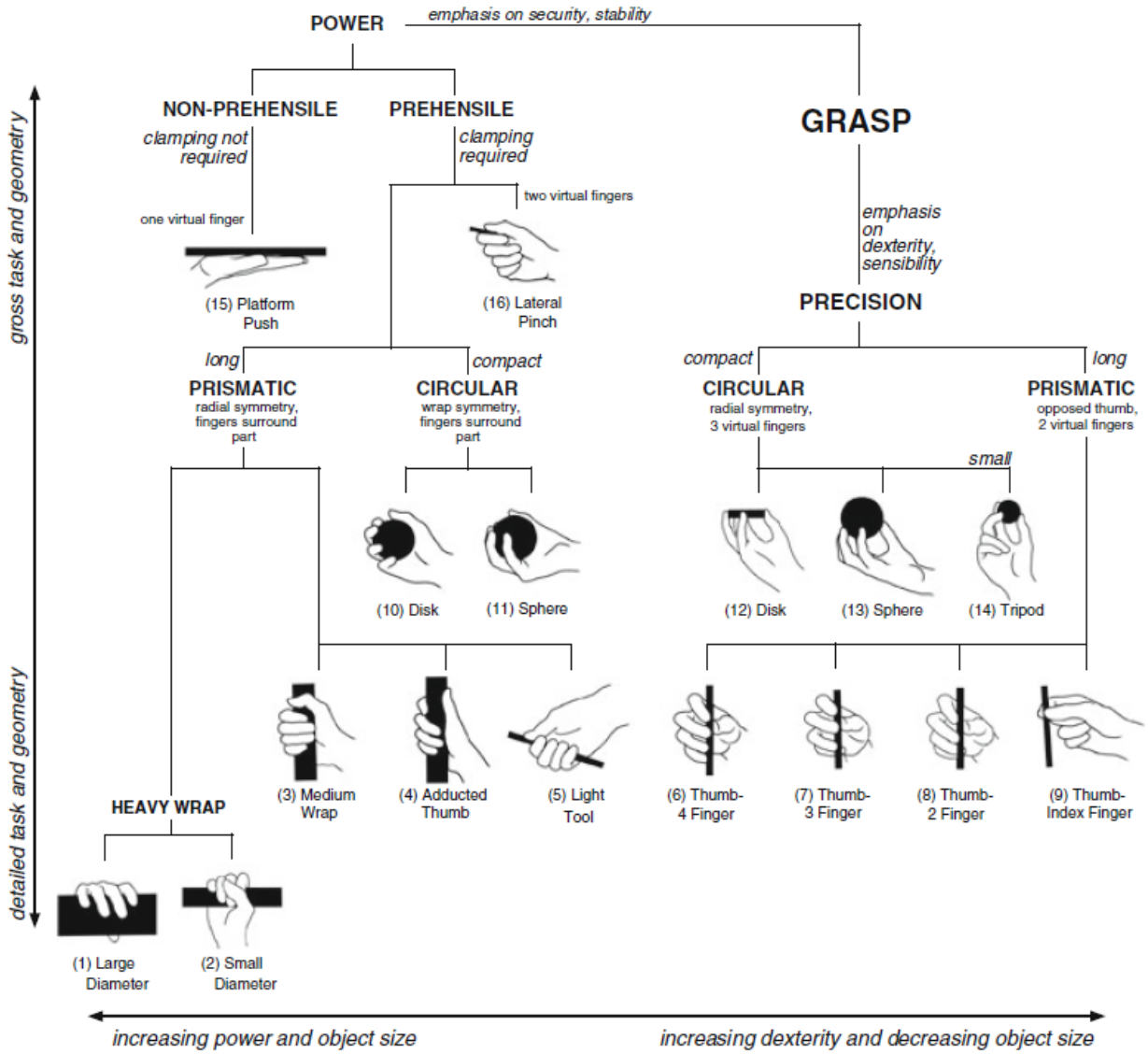

Figura 46: Taxonomía de Cutkosky

Aunque los seres humanos somos capaces de realizar agarres más complejos combinando varios de los agarres básicos identificados por Cutkosky, la destreza de la mano Dextra no es equiparable a la de una mano humana, entre otros motivos porque la capacidad de abducción del pulgar es muy limitada. Por este motivo, los agarres que van a implementarse en este proyecto van a limitarse únicamente a los agarres básicos definidos en la taxonomía de Cutkosky.

5.3. Clasificación de los movimientos

Una vez vista la taxonomía de Cutkosky, debemos de identificar las posiciones de los dedos necesarias para reproducir esos agarres. Para eso, como se ha mencionado, se ha empleado la GUI de control de Dextra para llevar los dedos a las posiciones necesarias, que se listan en la Tabla 1. Hay que mencionar que estas posiciones no son las posiciones angulares de las articulaciones de los dedos sino que, por cómo está programado el *firmware* de Dextra, representan el desplazamiento lineal de los tendones en mm. En el caso del abductor, la posición si está en grados.

		Dedos (mm)						
Nombre del agarre	Abductor(°)	Pulgar	Índice	Corazón	Anular	Meñique	Clasificación del agarre	Tipo de agarre
Platform Push	0	0	0	0	0	0	Non-Prehensile	Power
Lateral Pinch	5	4,5	9,9	14	12	14	Prehensile	
Tool	56	-1,6	20	18,2	20	20	Prismatic	
Large Diameter	81	-8	7,9	8	7,8	7,3	Prismatic Heavy Wrap	
Small Diameter	60	3	20,6	19,9	20,2	20	Prismatic Heavy Wrap	
Sphere Complete	80	-6,7	9,4	10,1	12	17	Circular Prehensile	
Disk	80	-5,5	6,5	6,8	7,6	16	Circular Precision	Grasp
Sphere Small	80	-7	9,1	9,9	12	17	Circular Precision	
Tripod	80	-4	9,1	0	0	0	Circular Precision	
Thumb-Index Finger	82	-5	11,4	15,8	17,5	19,5	Prismatic Precision	
Thumb-2 Finger	82	-5	11,6	12,3	16	16	Prismatic Precision	
Thumb-3 Finger	82	-5	11,1	11,8	10,7	12,6	Prismatic Precision	

Tabla 1: Posiciones de los dedos en cada uno de los distintos agarres

Durante las pruebas se ha observado que el servomotor produce un pitido cuando se lleva a una posición de 90 grados. Esto es debido a que, en esta posición, la pieza *abductor* choca contra la estructura donde se acopla la pieza *wrist*, lo cual le impide llegar a dicha posición de 90 grados. Esto provoca un exceso de consumo de corriente por parte del servomotor, lo que genera dicho pitido. Para evitar dañar este componente, se ha limitado la posición máxima de abducción a 84 grados.

5.4. Resultados de las pruebas

Para ilustrar los resultados obtenidos, vamos a reproducir el árbol jerárquico de la taxonomía de Cutkosky con los agarres realizados por nuestra mano. Estos resultados se muestran en la Figura 47.

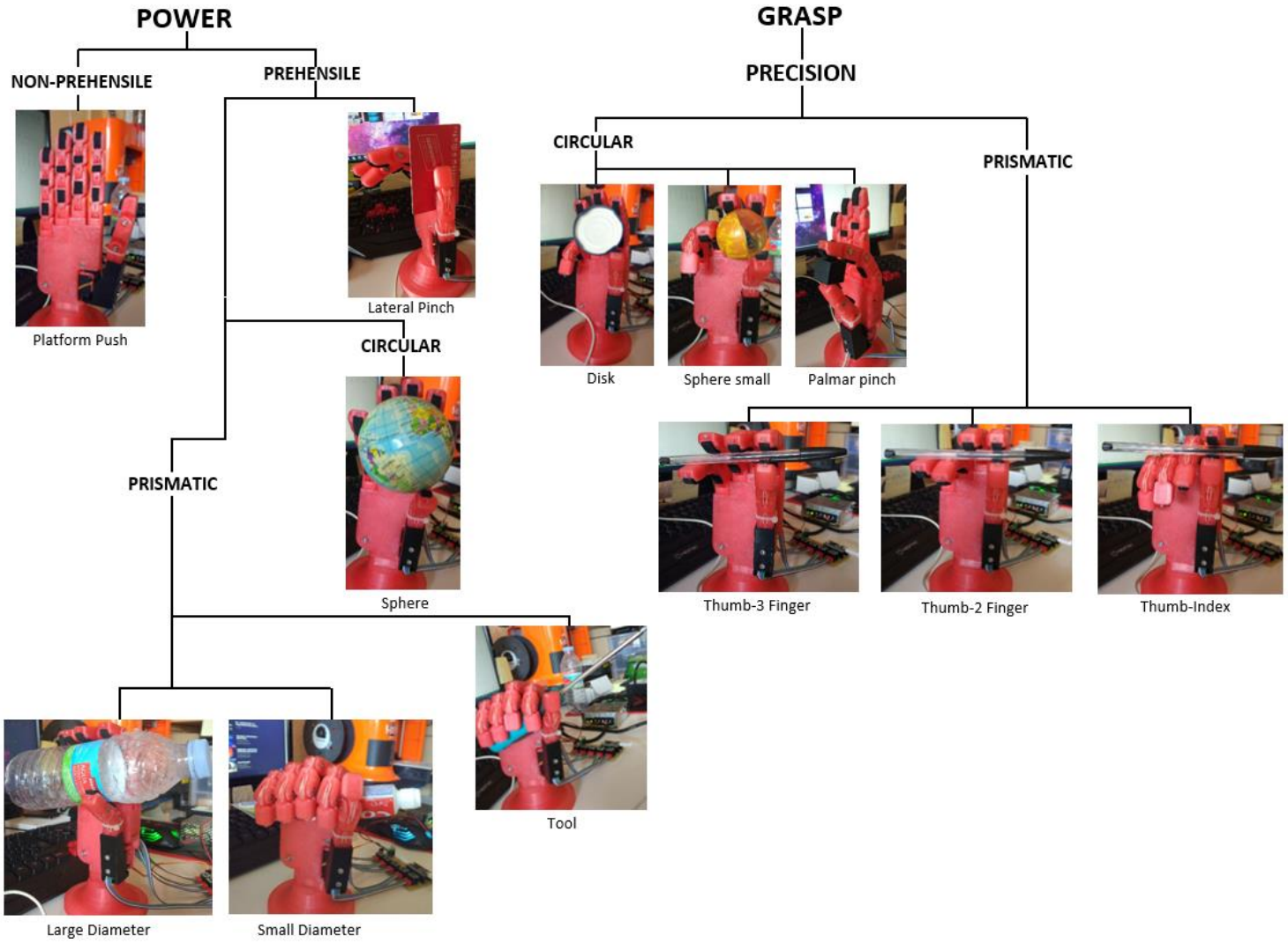

Figura 47: Reproducción de la taxonomía de Cutkosky con nuestros movimientos

Como podemos observar en la figura 47, la recreación de la taxonomía de Cutkosky con nuestra mano se puede realizar de manera casi completa, a excepción de los movimientos *(10) Disk*, *(3) Medium Wrap, (4) Adducted Thumb*, según vienen dados en la figura 46. Estos no han sido posibles debido a las limitaciones físicas en el diseño de la mano, ya que algunas de dichos movimientos implican posiciones imposibles para la mano Dextra, ya sea porque el pulgar no alcanza dichas posiciones o la taxonomía de la mano no lo permite.

Aunque hay agarres que aún no se pueden ejecutar, se pueden realizar futuras modificaciones de la mano para poder realizar la totalidad de los movimientos. La mano Dextra es viable para ser usada como una prótesis robótica sencilla, ya que al ser capaz de reproducir la mayoría de la taxonomía de Cutkosky, se realiza una evaluación positiva de la mano para la realización de las ADLs.

5.4.1. Problemas encontrados

Durante el periodo de pruebas de la mano, hemos encontrado una serie de problemas derivados de algunas de las propuestas dadas para mejorar el diseño de la mano.

Obstrucción de los tendones con los carretes

Durante el proceso de pruebas de movimientos de la mano, uno de los problemas encontrados ha sido que al realizar movimientos rápidos, o simplemente al llegar a ciertas posiciones, el sedal de hilo trenzado se quedaba atascado entre el carrete y la pieza *motor_holder*, tal y como muestra la Figura 48.

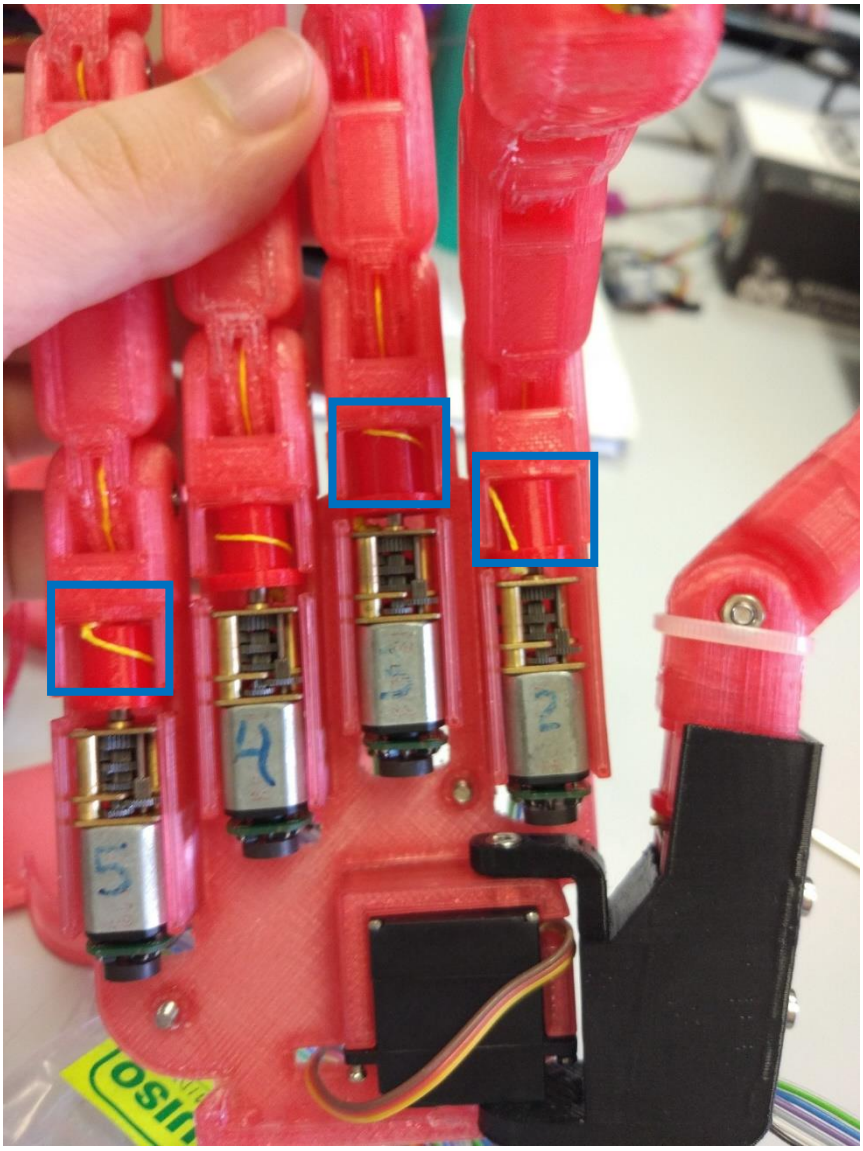

Figura 48: Vista interior de la mano con el sedal de hilo trenzado atascado (señalados con los rectángulos azules)

Como resultado, algunos de los dedos se quedaban permanentemente flexionados a pesar de estar el motor en la posición correspondiente al dedo en reposo, o directamente no realizaban flexión tras quedarse obstruidos. Dichos problemas se pueden observar en la Figura 49, donde los dedos índice y corazón, a pesar de no estar

conectados y habiendo sido desconectados en la posición de reposo, se encuentran en otra posición.

Figura 49: Vista lateral de la mano con el sedal de hilo trenzado atascado

Esto sucede debido al uso del sedal de hilo trenzado frente a un sedal de nylon. El hilo trenzado, a pesar de tener una mayor resistencia a la torsión y a ser más deformable, debido a su estructura de fibras muy finas trenzadas se aplana y, al tensarse por la acción del motor, se introduce en los huecos remarcados en la Figura 49. La única manera de solucionar este atasco era desmontando el módulo de dedo afectado y sacando el motor de su posición. Tras varios ensayos, se repetía el mismo problema, por lo que se decidió sustituir el hilo trenzado propuesto por el hilo de nylon del diseño original.

Rotura de los cables de conexionado de los *encoders*

Con el diseño del nuevo cableado de los motores, con los nuevos conectores y los cables planos, estos van soldados directamente a cada *encoder* (en el diseño original son unos pines hembra los que se sueldan a cada *encoder*). Aunque a priori esto no debería ser causa de posibles fallos, durante el montaje y testeo de la mano se encontró que algunas de las uniones de las soldaduras fallaban y se partían. En el caso del pulgar, ocurría la rotura total de las conexiones, al estar sometido a un mayor esfuerzo de

torsión por la movilidad de la pieza abductora. La causa de este fallo posiblemente sea la calidad de los cables, ya que su sección es muy pequeña y se rompen con facilidad.

La solución encontrada ha sido descargar la tensión en esos puntos, reforzando las soldaduras a través de un compuesto plástico. En este caso, se ha aplicado pegamento termofusible en todas las uniones del cable con el *encoder*, creando una acumulación de pegamento que protege las uniones. Esto hace que cuando se mueve el cable, la fuerza aplicada es absorbida por esta capa protectora en vez de sobre la conexión del cable con el *encoder*.

6. Conclusiones

6.1. Conclusiones

Para concluir con la presente memoria, realizando un análisis objetivo del conjunto del proyecto:

- Se ha realizado la impresión de todos los componentes mediante una impresora 3D.
- Se ha realizado el montaje de la mano Dextra, realizando los ajustes necesarios y añadiendo las implementaciones deseadas al diseño original, modificando algunos de los elementos originales con la intención de realizar una mejora en el conjunto de la mano.
- Utilizando las herramientas de configuración ya creadas, se ha realizado la configuración y ajuste de la mano Dextra.
- A través del lenguaje de programación Python, se han creado una serie de códigos para replicar los agarres definidos por la taxonomía de Cutkosky.
- Se ha documentado todo lo anterior de modo que pueda ser usado como un manual de instrucciones por cualquier desarrollador que quiera realizar el mismo trabajo, mejorarlo o modificarlo.

Como resultado general, se ha construido una mano robótica de bajo coste, con un gran rango de movimientos, sin gran complejidad de construcción, modular y de fácil reparación.

La facilidad de la programación de los movimientos de la mano permite a desarrolladores con poca experiencia implementar nuevos agarres e integrar este dispositivo en sus propios proyectos.

Todo esto supone que esta mano robótica tiene un gran potencial tanto para ser usada como prótesis (debido a su tamaño, diseño antropomórfico y capacidad de manipulación) como para ser usada como mano robótica de propósito general en robots humanoides, brazos robóticos, manipuladores móviles… Su coste, replicabilidad y diseño abierto también la hace muy atractiva para ser usada en un ámbito educativo en institutos, universidades o centros de investigación.

6.2. Trabajos futuros

Conforme a lo que hemos observado en el estado del arte, las manos robóticas actuales incorporan algunas funcionalidades que son aplicables a la mano Dextra. Vamos a analizar brevemente algunas propuestas que se podrían realizar de manera inmediata:

- Control por EMG (mioeléctrico): cualquier prótesis robótica hoy en día incluye un control de electromiografía para tener un uso más natural. Este tipo de control podría implementarse en la mano Dextra, conectando un sistema de electromiografía a un dispositivo capaz de comunicarse con la mano e implementando dicho sistema de control.
- Rediseño de la anatomía de la mano: aunque el diseño original, incluyendo nuestras modificaciones, es bastante antropomórfico, se podría realizar una modificación del diseño original de la mano para mejorar su antropomorfismo, inclinando el abductor frente a la palma para adquirir una posición más natural y añadir nuevos movimientos a la mano. Posiblemente este cambio de posición necesitaría una recolocación de los demás elementos.
- Realizar un rediseño de la electrónica de control para poder integrarlo en la estructura de la mano, colocándolo sobre la pieza dorsal de la mano, mejorando su portabilidad y modularidad.
- Imprimir todas las piezas en material con filamentos flexibles.
- Añadir nuevas funciones a la GUI de Dextra, como algunos botones que tengan agarres predefinidos de ejemplo.

7. Referencias

[1] "Alternativas de licenciamiento de software libre y open source. Análisis legal", *Derechos Digitales*, 16/05/2005. **[En línea]. Disponible en:** https://www.derechosdigitales.org/29/alternativas-de-licenciamiento-de-software-libre/ (última visita: 24/05/2018)

[2] José M. Fidel Santiago Luque, "Legislación europea y el Open Source / Free Software". **[En línea]. Disponible en:** http://es.tldp.org/Presentaciones/200309hispalinux/2/2_htmldoc.pdf (última visita: 24/03/2018)

[3] "A quick Guide to GPLv3", *GNU Operating System.* **[En línea]. Disponible en:** https://www.gnu.org/licenses/quick-guide-gplv3.html (última visita: 31/05/2018)

[4] "Attribution-ShareAlike 4.0 International", *Creative Commons.* **[En línea]. Disponible en:** https://creativecommons.org/licenses/by-sa/4.0/ (última visita: 31/05/2018)

[5] "Aesthetic prothetics and its Social, Cultural and Psychological Impacts", *Curatorial Ambition*, 08/05/2014. **[En línea]. Disponible en:** https://jwmelville17.wordpress.com/2014/05/08/aesthetic-prosthetics-and-its-social-cultural-and-psychological-impacts/ (última visita: 27/05/2018)

[6] "16th Century Prothetic iron Hand: The Story of Gotz von Berlichingen", *Ancient Origins*. **[En línea]. Disponible en:** http://www.ancient-origins.net/history-famous-people/16th-century-prosthetic-iron-hand-story-gotz-von-berlichingen-006153 (última visita: 14/06/2018)

[7] "Historia de las prótesis de mano", *La Reunión*. **[En línea]. Disponible en:** http://revistalareunion.blogspot.com/2008/09/historia-de-las-prtesis-de-mano.html (última visita: 14/06/2018)

[8] "Shadow Dexterous Hand", *Shadow Robot Company*. **[En línea]. Disponible en:** https://www.shadowrobot.com/products/dexterous-hand/ (última visita: 28/05/2018)

[9] T. Mouti, H. Kawasaki, K. Yoshikawa, J. Takai, S. Ito, "Anthropomorphic Robot Hand: Gifu Hand III", *Proc. Int. Conf. Control Autom. Syst.,* pp 1288-1293, octubre 2002.

[10] "The PISA/IIT SoftHand", *Centro E. Piaggio bioenginnering and robotics research center*. **[En línea]. Disponible en:** http://www.centropiaggio.unipi.it/pisaiit-softhand (última visita: 10/05/2018)

[11] A. Bicchi, M. G. Catalano, A. Ajoudani, S.B Godfrey, M. Bianchi, G. Grioli, A. Saerio, E. Farnioli, C. Piazza, M. Bonilla, M. Garabini, M. Gabiccini, "What can a simple hand do?", *Mobile Manipulation*. **[En línea]. Disponible en:** http://mobilemanipulation.org/rss2014/images/Talks/4brock.pdf (última visita: 11/05/2018)

[12] G. Grioli, M. G. Catalano, M. Silvestro, E. Tono, S. Bicchi, Antonio. (2012). "Adaptive synergies for the Desing and Control of the Pisa/IIT SoftHand", *International Journal of Robotics Research*, col. 33, Issue 5, pp. 768-782.

[13] M.G. Catalano, G. Grioli. E. Farnioli, A. Serio, C. Piazza, M. Gabiccini, A. Bicchi. "From Soft to Adarptative Synergies: The Pisa/IIT Softhand", *Humans and Robot Hands,* pp 101-125, 2014.

[14] "SoftHands", *SoftHands*. **[En línea]. Disponible en:** http://www.softhands.eu/ (última visita: 11/05/2018)

[15] D. S. Childress, "Historical Aspects of Powered Limb Prostheses", *Clinical Prosthetics & Orthotics*, vol. 9, no. 1, pp. 2-13, 1985.

[16] "i-limb", *Touch Bionics*. **[En línea]. Disponible en:** https://www.touchbionics.com/ (última visita: 28/05/2018)

[17] "BeBionic Hand", *Ottobock*. **[En línea]. Disponible en:** https://www.ottobock.co.uk/ (última visita: 28/05/2018)

[18] Motherboard, "Living With Future Prosthetics", *YouTube*, 27/10/2016. **[Vídeo en línea]. Disponible en:** https://www.youtube.com/watch?v=1gIIega50To (última visita: 27/05/2018)

[19] Motherboard, "The Mind-Controlled Bionic Arm With a Sense of Touch", *YouTube*, 18/08/2016. **[Vídeo en línea]. Disponible en:** https://www.youtube.com/watch?v=F_brnKz_2tI&t (última visita: 27/05/2018)

[20] C. W. Moran, "Revolutionizing Prosthetics 2009 Modular Prosthetic Limb-Body Interface: Overview of the Prosthetic Socket Development", *Johns Hopkins Apl Technical Digest,* vol. 30, no. 3, pp. 240-249, 2011.

[21] M. S. Johannes, J. D. Bigelow, J. M. Burck, S.D. Harshbarger, M. V. Kozlowski, T. V. Doren, "An Overviem of the Developmental Process for the Modular Prosthetic Limb", *Johns Hopkins Apl Technical Digest,* vol. 30, no. 3, pp. 207-216, 2011.

[22] "Open-Source Robotic Bionic Hands", *OpenBionics*. **[En línea]. Disponible en:** http://openbionics.org (última visita: 26/03/2018)

[23] G. P. Kontododis, M. V. Liarokapis, A. G. Zisimatos, C. I. Mavrogiannis, K. Kyriakopoulos,"Open-Source,Antropomorphic, Underactuated Robot Hands with a Selectively Lockable Differential Mechanism: Towards Affordable Protheses", *Intelligent Robots and Systems (IROS)*, 2015.

[24] G. Langevin, "InMoov Open Source 3D printed life size robot", *Inmoov*. **[En línea]. Disponible en:** http://inmoov.fr/ (última visita: 28/03/2018)

[25] TechRadar, "3D printing yourself a hand: Deux Ex's bionics limbs made for real by Open Bionics", *YouTube*, 28/07/2016. **[Vídeo en línea]. Disponible en:** https://www.youtube.com/watch?v=3nnrstBxomk&t (última visita: 27/05/2018)

[26] "Open Bionics", *Open Bionics*. **[En línea]. Disponible en:** https://openbionics.com/ (última visita: 28/03/2018)

[27] J. Gibbard, "Dextrus", *Open Hand Project,* 2013. **[En línea]. Disponible en:** http://www.openhandproject.org/dextrus.php (última visita: 26/03/2018)

[28] "Brunel Hand", *Open Bionics.* **[En línea]. Disponible en:** https://www.openbionics.com/shop/brunel-hand (última visita: 26/03/2018)

[29] "Hero Arm", *Open Bionics*. **[En línea]. Disponible en:** https://openbionics.com/hero-arm/ (última visita: 10/05/2018)

[30] H. Watkin, "Open Bionics Releases Incredibly Affordable and Customizable "Hero Arm" Prosthetic", *All About 3D Printing*, 24/04/2018. **[En línea]. Disponible en:** https://all3dp.com/open-bionics-releases-3d-printed-affordable-hero-arm/ (última visita: 24/05/2018)

[31] A. Villoslada, "Dextra Project, Open-source myoelectric hand prosthesis", *Hackaday*. **[En línea]. Disponible en:** https://hackaday.io/project/9890-dextra (última visita: 14/06/2018)

[32] Alvipe, "Dextra", *Github*. **[En línea]. Disponible en:** https://github.com/Alvipe/Dextra/ (última visita: 14/06/2018)

[33] Alvipe, "Dextra", *Thingiverse*. **[En línea]. Disponible en:** https://www.thingiverse.com/thing:1538508 (última visita: 14/06/2018)

[34] "Músculo Separador o Abductor Corto del Pulgar", *musculos.org*. **[En línea]. Disponible en:** https://www.musculos.org/musculo-separador-abductor-corto-pulgar.html (última visita: 01/04/2018)

[35] "Mano", *Wikipedia La Enciclopedia libre*. **[En línea]. Disponible en:** https://es.wikipedia.org/wiki/Mano (última visita: 01/04/2018)

[36] T. Pham, P. N. Pathirana, H. Trinh, P. Frasy,"A Non-Contact Measurement System for the Range of Motion of the Hand", *Sensors*, 2015.

[37] "FDM-FFF o modelado por deposición fundida", *Todo 3D*. **[En línea]. Disponible en:** http://todo-3d.com/fdm-fff-modelado-deposicion-fundida/ (última visita: 22/05/2018)

[38] "External Interrups", *Arduino*. **[En línea]. Disponible en:** https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/ (última visita: 13/06/2018)

[39] "PlatformIO is an open source ecosystem for IoT development", *Platformio*. **[En línea]. Disponible en:** http://docs.platformio.org (última visita: 14/05/2018)

[40] A. M. Dollar, "Classifying Human Hand Use and the Activities of Daily Living", *The Human Hand as an Inspiration for Robot Hand Development*, capítulo 10, pp. 201-216, 2014.

[41] "Ácido Poliláctico (PLA)", *Textos científicos.* **[En línea]. Disponible en:** https://www.textoscientificos.com/polimeros/acido-polilactico (última visita: 26/03/2018)

[42] "Myo Techspecs", *Myo Armband*. **[En línea]. Disponible en:** https://www.myo.com/techspecs (última visita: 29/03/2018)

[43] "Engranaje planetario", *Wikipedia La Enciclopedia libre*. **[En línea]. Disponible en:** https://es.wikipedia.org/wiki/Engranaje_planetario (última visita: 30/03/2018)

[44] "3D-Printable Prothetic Devices", *National Institutes of Health 3D Print Exchange*. **[En línea]. Disponible en:** https://3dprint.nih.gov/collections/prosthetics (última visita: 27/05/2018)

[45] "Da Vinci Mini", *XYZ Printing*. **[En línea]. Disponible en:** http://eu.xyzprinting.com/eu_es/Product/da-Vinci-Mini (última visita: 14/05/2018)

[46] "Kivy User's Guide >> Installation on Windows", *Kivy*. **[En línea]. Disponible en:** https://kivy.org/docs/installation/installation-windows.html (última visita: 17/05/2018)

[47] A. Pîrjan, D. M. Petroşanus,"The impact of 3D printing technology on the society and economy", *Journal of information systems & operations management,* vol. 7, no.2, pp 360-370, diciembre 2013.

[48] P. Shenoy, K. J. Miller, B. Crawford, R. P. N. Rao, "Online Electromyographic Control of a Robotic Prosthesis", *IEEE Transactions on biomedical engineering*, vol. 55, no. 3, marzo 2008.

[49] L. Zollo, S. Roccella, E. Guglielmelli, M.C. Carrozza, P. Dario, "Biomechatronic Design and Control of an Anthromorphic Artificial hand for Prothetic and Robotic Applications", *IEEE/ASME Transactions on mechatronics*, vol. 12, no. 4, pp 418-429 agosto 2007.

[50] I. M. Bullock, R. R. Ma, A. M. Dollar, "A Hand-Centric Classification of Human and Robot Dexterous Manipulation", *IEEE Transactions on Haptiics*, vol. 6, no. 2, pp 129-144, 2013.

8. Anexos

8.1. Presupuesto

El presupuesto del proyecto teniendo en cuenta todos los materiales y componentes necesarios se ilustra en la siguiente tabla:

Unidades	Componentes/Materiales	Medición	Precio unitario (€)	Precio total (€)
Ud.	Turnigy TGY-EX5252MG Twin BB Digital Micro Servo	1	8,84	8,84
Ud.	x2 Polulu Magnetic Encoder for Micro Metal Gearmotors	3	7,67	23,01
Ud.	Polulu Micro Metal Gearmotor 1000:1 HP	5	20,54	102,70
Ud.	DRV8838 Single Brushed DC Motor Driver Carrier	5	2,56	12,80
Ud.	Sedal de pesca Line 4x4 250 metros Caperlan	1	1,99	1,99
Ud.	Sedal de pesca Line Resist Fluo Yellow 150 m Caperlan	1	3,49	3,49
Ud.	Bolsa 20 tornillos M3x14 mm	1	2,14	2,14
Ud.	Bolsa 25 tornillos M3x12 mm	1	4,47	4,47
Ud.	Bolsa 20 tornillos M3x8 mm	1	1,70	1,70
Ud.	Bolsa 20 tornillos M3x6 mm	1	1,83	1,83
Ud.	Tornillo M10x35 mm	1	0,29	0,29
Ud.	Bolsa 100 tuercas M3	1	5,02	5,02
Ud.	Tuerca M10	1	0,09	0,09
Ud.	Bolsa 10 espaciadores M3x12	1	1,00	1,00
Ud.	Tensy 3.2 USB Development Board	1	16,98	16,98
Ud.	x5 Condensador 100 uF	1	1,56	1,56
Ud.	x5 Condensador 10 uF	1	1,811	1,81
m.	Cable plano Micro-Match de paso 1.27 mm, 16 bandas, 150 mm	3	2,37	7,11
Ud.	Fuente de alimentación Mean Well S-25-5	1	7,70	7,70
Ud.	Regulador LDO de tensión 6.25 V Texas Instruments	1	1,31	1,31
Ud.	Pack 5 Conector IDC TE Connectivity serie Micro-Match de 6 contactos y paso 1.27mm	1	3,97	3,97
Ud.	Barra pegamento termofusible (pack de 50)	1	3,13	3,13
Ud.	Bandas elásticas de goma de ortodoncia 1/8" pulgadas (Pack de 100)	1	4,71	4,71
Ud.	Antideslizante INOFIX ADHESIVO RECORTABLE 100x85mm	1	1,40	1,40
Ud.	Soldador de plomo RS Pro 1.27mm 250g	1	14,17	14,17
m.	Filamento XYZ PLA	71,1	0,15	10,67
Ud.	Placa de circuito impreso	1	50,00	50,00
			Total (€)	**293,89**

Tabla 2: Presupuesto del proyecto

Alcanzando un coste total de **doscientos noventa y tres euros con ochenta y nueve céntimos**.

8.2. Fases del proyecto

Este proyecto se ha dividido en las varias fases, cuyos periodos de ejecución son los siguientes:

TAREA	FECHA DE INICIO	DURACIÓN	FECHA FINAL
Impresión de las piezas en 3D	10/05/2018	4	14/05/2018
Montaje de la mano Dextra	14/05/2018	2	16/05/2018
Periodo de ajuste y cambios principales	16/05/2018	3	19/05/2018
Programación	19/05/2018	13	01/06/2018
Análisis de resultados	20/05/2018	12	01/06/2018

Tabla 3: Duración del proyecto

Dichas fases se pueden representar en el siguiente diagrama:

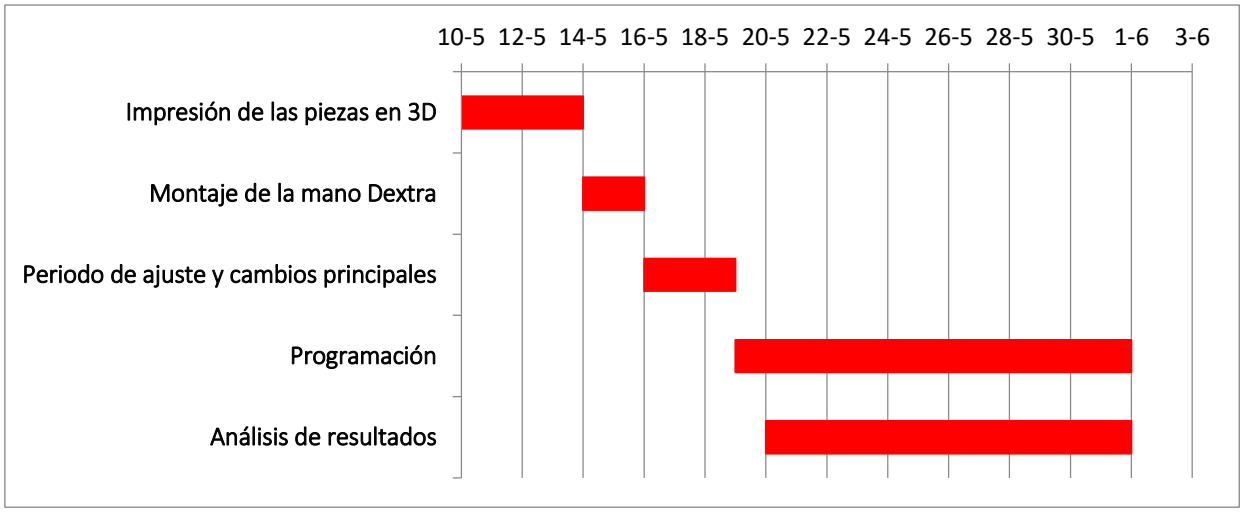

Figura 50: Diagrama de Gantt

8.3. Código de Python

```
"""
Created on Wed May 23 17:05:23 2018
name: Platform Push
@author: Daniel Jason Castillo Patton
"""
import synapse as sy #importamos las librerias necesarias, llamamos sy para abreviar
import time

serial_port_list = sy.serial_port_scan() #escaneamos los puertos serie del ordenador

print serial_port_list[0][0]  # Primera lista [0] y primer elemento de la lista [0]
print serial_port_list[0][1]  # Primera lista [0] y segundo elemento de la lista [1]
print serial_port_list[0][2]  # Primera lista [0] y tercer elemento de la lista [2]
print "\n"

serial_port = sy.serial_port_select(serial_port_list)

ser, status = sy.serial_port_connect(serial_port)

if status:

    setpoint_list = [10,0,0,0,0,0] #Posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list) #Enviamos la posicion de reposo siempre
    time.sleep(2)  # Esperamos 2 segundos

    raw_input("Start")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[0,0,0,0,0,0])  # Enviamos posiciones
    time.sleep(1)

    raw_input("Open")  # El programa espera hasta que se pulse intro
    setpoint_list[0] = [10]  # Valores para volver a la posicion inicial
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)

    ser.close() #Cerramos la conexion con el puerto serie
else:
    print("Connection error") #Si no conseguimos cominicar con el puerto serie, sacamos un mensaje de error
```

```
"""
Created on Wed May 23 17:10:42 2018
name: Lateral Pinch
@author: Daniel Jason Castillo Patton
"""
import synapse as sy #importamos las librerias necesarias, llamamos sy para abreviar
import time

serial_port_list = sy.serial_port_scan() #escaneamos los puertos serie del ordenador

print serial_port_list[0][0]  # Primera lista [0] y primer elemento de la lista [0]
print serial_port_list[0][1]  # Primera lista [0] y segundo elemento de la lista [1]
print serial_port_list[0][2]  # Primera lista [0] y tercer elemento de la lista [2]
print "\n"

serial_port = sy.serial_port_select(serial_port_list)

ser, status = sy.serial_port_connect(serial_port)

if status:

    setpoint_list = [10,0,0,0,0,0] #Posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list) #Enviamos la posicion de reposo siempre
    time.sleep(2)  # Esperamos 2 segundos

    raw_input("Start")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[10,-2,9.9,14,12,14])  # Enviamos posiciones
    time.sleep(1)
    raw_input("Close")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[5,4.5,9.9,14,12,14])  # Enviamos posiciones
    time.sleep(1)
    raw_input("Open")  # El programa espera hasta que pulses intro
    setpoint_list = [5,0,9.9,14,12,14]  # Abrimos el pulgar
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(3)
    setpoint_list = [10,0,0,0,0,0]  # Colocamos el pulgar en su posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)

    ser.close() #Cerramos la conexion con el puerto serie
else:
    print("Connection error") #Si no conseguimos cominicar con el puerto serie, sacamos un mensaje de error
```

```
"""
Created on Wed May 23 18:10:34 2018
name: Tool
@author: Daniel Jason Castillo Patton
"""
import synapse as sy #importamos las librerias necesarias, llamamos sy para abreviar
import time

serial_port_list = sy.serial_port_scan() #escaneamos los puertos serie del ordenador

print serial_port_list[0][0]  # Primera lista [0] y primer elemento de la lista [0]
print serial_port_list[0][1]  # Primera lista [0] y segundo elemento de la lista [1]
print serial_port_list[0][2]  # Primera lista [0] y tercer elemento de la lista [2]
print "\n"

serial_port = sy.serial_port_select(serial_port_list)

ser, status = sy.serial_port_connect(serial_port)

if status:

    setpoint_list = [10,0,0,0,0,0] #Posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list) #Enviamos la posicion de reposo siempre
    time.sleep(2)  # Esperamos 2 segundos

    raw_input("Start")  # El programa espera hasta que se pulse intro
    setpoint_list[0] = 56  # Colocamos el pulgar a 56 grados
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)
    setpoint_list[1] = -3  # Flexionamos el pulgar del todo hacia atras
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    raw_input("Close")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[56,-1.6,20,18.2,20,20])  # Enviamos posiciones
    time.sleep(1)
    raw_input("Open")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[56,-4,20,18.2,20,20])
    time.sleep(3)
    sy.write_setpoint_list(ser,[56,-4,0,0,0,0])
    time.sleep(3)
    setpoint_list[0:2] = [10,0]  # Colocamos el pulgar en su posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)

    ser.close() #Cerramos la conexion con el puerto serie
else:
    print("Connection error") #Si no conseguimos cominicar con el puerto serie, sacamos un mensaje de error
```

```
"""
Created on Sun May 27 11:25:29 2018
name: Large Diameter
@author: Daniel Jason Castillo Patton
"""
import synapse as sy #importamos las librerias necesarias, llamamos sy para abreviar
import time

serial_port_list = sy.serial_port_scan() #escaneamos los puertos serie del ordenador

print serial_port_list[0][0]  # Primera lista [0] y primer elemento de la lista [0]
print serial_port_list[0][1]  # Primera lista [0] y segundo elemento de la lista [1]
print serial_port_list[0][2]  # Primera lista [0] y tercer elemento de la lista [2]
print "\n"

serial_port = sy.serial_port_select(serial_port_list)

ser, status = sy.serial_port_connect(serial_port)

if status:

    setpoint_list = [10,0,0,0,0,0] #Posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list) #Enviamos la posicion de reposo siempre
    time.sleep(2)  # Esperamos 2 segundos

    raw_input("Start")  # El programa espera hasta que se pulse intro
    setpoint_list[0] = 81  # Colocamos el pulgar a 90 grados
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)
    setpoint_list[1] = -8  # Flexionamos el pulgar hacia atras
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    raw_input("Close")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[81,-8,7.9,8,7.8,7.3])  # Enviamos posiciones
    time.sleep(1)
    raw_input("Open")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[81,-8,0,0,0,0])
    time.sleep(3)
    setpoint_list[0:2] = [10,0]  # Colocamos el pulgar en su posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)

    ser.close() #Cerramos la conexion con el puerto serie
else:
    print("Connection error") #Si no conseguimos cominicar con el puerto serie, sacamos un mensaje de error
```

```
"""
Created on Sun May 27 11:31:12 2018
name: Small Diameter
@author: Daniel Jason Castillo Patton
"""
import synapse as sy #importamos las librerias necesarias, llamamos sy para abreviar
import time

serial_port_list = sy.serial_port_scan() #escaneamos los puertos serie del ordenador

print serial_port_list[0][0]  # Primera lista [0] y primer elemento de la lista [0]
print serial_port_list[0][1]  # Primera lista [0] y segundo elemento de la lista [1]
print serial_port_list[0][2]  # Primera lista [0] y tercer elemento de la lista [2]
print "\n"

serial_port = sy.serial_port_select(serial_port_list)

ser, status = sy.serial_port_connect(serial_port)

if status:

    setpoint_list = [10,0,0,0,0,0] #Posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list) #Enviamos la posicion de reposo siempre
    time.sleep(2)  # Esperamos 2 segundos

    raw_input("Start")  # El programa espera hasta que se pulse intro
    setpoint_list[0] = 52  # Colocamos el pulgar a 90 grados
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)
    setpoint_list[1] = -7  # Flexionamos el pulgar hacia atras
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    raw_input("Close")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[52,-5.5,20,20,20,20])  # Enviamos posiciones
    time.sleep(1)
    raw_input("Open")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[81,-8,20,20,20,20])
    time.sleep(3)
    sy.write_setpoint_list(ser,[81,-8,0,0,0,0])
    time.sleep(3)
    setpoint_list[0:2] = [10,0]  # Colocamos el pulgar en su posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)

    ser.close() #Cerramos la conexion con el puerto serie
else:
    print("Connection error") #Si no conseguimos cominicar con el puerto serie, sacamos un mensaje de error
```

```
"""
Created on Wed May 23 17:46:56 2018
name: Sphere Complete
@author: Daniel Jason Castillo Patton
"""
import synapse as sy #importamos las librerias necesarias, llamamos sy para abreviar
import time

serial_port_list = sy.serial_port_scan() #escaneamos los puertos serie del ordenador

print serial_port_list[0][0]  # Primera lista [0] y primer elemento de la lista [0]
print serial_port_list[0][1]  # Primera lista [0] y segundo elemento de la lista [1]
print serial_port_list[0][2]  # Primera lista [0] y tercer elemento de la lista [2]
print "\n"

serial_port = sy.serial_port_select(serial_port_list)

ser, status = sy.serial_port_connect(serial_port)

if status:

    setpoint_list = [10,0,0,0,0,0] #Posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list) #Enviamos la posicion de reposo siempre
    time.sleep(2)  # Esperamos 2 segundos

    raw_input("Start")  # El programa espera hasta que se pulse intro
    setpoint_list[0] = 72  # Colocamos el pulgar a 90 grados
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)
    setpoint_list[1] = -8  # Flexionamos el pulgar del todo hacia atras
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    raw_input("Close")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[72,-8,5.7,5.5,6.1,8.5])  # Enviamos posiciones
    time.sleep(1)
    raw_input("Open")  # El programa espera hasta que pulses intro
    setpoint_list = [72,-8,0,0,0,0]  # Colocamos el pulgar en su posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(3)
    setpoint_list = [10,0,0,0,0,0]  # Colocamos el pulgar en su posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)

    ser.close() #Cerramos la conexion con el puerto serie
else:
    print("Connection error") #Si no conseguimos cominicar con el puerto serie, sacamos un mensaje de error
```

```
"""
Created on Wed May 23 18:22:03 2018
name: Disk
@author: Daniel Jason Castillo Patton
"""
import synapse as sy #importamos las librerias necesarias, llamamos sy para abreviar
import time

serial_port_list = sy.serial_port_scan() #escaneamos los puertos serie del ordenador

print serial_port_list[0][0]  # Primera lista [0] y primer elemento de la lista [0]
print serial_port_list[0][1]  # Primera lista [0] y segundo elemento de la lista [1]
print serial_port_list[0][2]  # Primera lista [0] y tercer elemento de la lista [2]
print "\n"

serial_port = sy.serial_port_select(serial_port_list)

ser, status = sy.serial_port_connect(serial_port)

if status:

    setpoint_list = [10,0,0,0,0,0] #Posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list) #Enviamos la posicion de reposo siempre
    time.sleep(2)  # Esperamos 2 segundos

    raw_input("Start")  # El programa espera hasta que se pulse intro
    setpoint_list[0] = 80  # Colocamos el pulgar a 90 grados
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)
    setpoint_list[1] = -6  # Flexionamos el pulgar hacia atras
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    raw_input("Close")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[80,-5.5,6.5,6.8,7.6,16])  # Enviamos posiciones
    time.sleep(1)
    raw_input("Open")  # El programa espera hasta que se pulse intro
    setpoint_list[2:6] = [0,0,0,0]  # Colocamos los dedos en su posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(3)
    setpoint_list[0:2] = [10,0]  # Colocamos el pulgar en su posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)

    ser.close() #Cerramos la conexion con el puerto serie
else:
    print("Connection error") #Si no conseguimos cominicar con el puerto serie, sacamos un mensaje de error
```

```
"""
Created on Wed May 23 18:00:56 2018
name: Sphere small
@author: Daniel Jason Castillo Patton
"""
import synapse as sy #importamos las librerias necesarias, llamamos sy para abreviar
import time

serial_port_list = sy.serial_port_scan() #escaneamos los puertos serie del ordenador

print serial_port_list[0][0]  # Primera lista [0] y primer elemento de la lista [0]
print serial_port_list[0][1]  # Primera lista [0] y segundo elemento de la lista [1]
print serial_port_list[0][2]  # Primera lista [0] y tercer elemento de la lista [2]
print "\n"

serial_port = sy.serial_port_select(serial_port_list)

ser, status = sy.serial_port_connect(serial_port)

if status:

    setpoint_list = [10,0,0,0,0,0] #Posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list) #Enviamos la posicion de reposo siempre
    time.sleep(2)  # Esperamos 2 segundos

    raw_input("Start")  # El programa espera hasta que se pulse intro
    setpoint_list[0] = 80  # Colocamos el pulgar a 90 grados
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)
    setpoint_list[1] = -7  # Flexionamos el pulgar del todo hacia atras
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    raw_input("Close")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[80,-6.7,9.4,10.1,12,17])  # Enviamos posiciones
    time.sleep(1)
    raw_input("Open")  # El programa espera hasta que se pulse intro
    setpoint_list[2:6] = [0,0,0,0]  # Colocamos los dedos en su posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(3)
    setpoint_list[0:2] = [10,0]  # Colocamos el pulgar en su posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)

    ser.close() #Cerramos la conexion con el puerto serie
else:
    print("Connection error") #Si no conseguimos cominicar con el puerto serie, sacamos un mensaje de error
```

```
"""
Created on Wed May 23 18:34:01 2018
name: Tripod
@author: Daniel Jason Castillo Patton
"""

import synapse as sy #importamos las librerias necesarias, llamamos sy para abreviar
import time

serial_port_list = sy.serial_port_scan() #escaneamos los puertos serie del ordenador

print serial_port_list[0][0]  # Primera lista [0] y primer elemento de la lista [0]
print serial_port_list[0][1]  # Primera lista [0] y segundo elemento de la lista [1]
print serial_port_list[0][2]  # Primera lista [0] y tercer elemento de la lista [2]
print "\n"

serial_port = sy.serial_port_select(serial_port_list)

ser, status = sy.serial_port_connect(serial_port)

if status:

    setpoint_list = [10,0,0,0,0,0] #Posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list) #Enviamos la posicion de reposo siempre
    time.sleep(2)  # Esperamos 2 segundos

    raw_input("Start")  # El programa espera hasta que se pulse intro
    setpoint_list[0] = 84  # Colocamos el pulgar en la posicion deseada
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)
    setpoint_list[1] = -5.5  # Flexionamos el pulgar hacia atras
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    raw_input("Close")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[80,-4,9.1,0,0,0])  # Enviamos posiciones
    time.sleep(1)
    raw_input("Open")  # El programa espera hasta que se pulse intro
    setpoint_list[2:6] = [0,0,0,0]  # Colocamos los dedos en su posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(3)
    setpoint_list[0:2] = [10,0]  # Colocamos el pulgar en su posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)

    ser.close() #Cerramos la conexion con el puerto serie
else:
    print("Connection error") #Si no conseguimos cominicar con el puerto serie, sacamos un mensaje de error
```

```
"""
Created on Wed May 23 18:44:16 2018
name: Thumb-Index Fingers
@author: Daniel Jason Castillo Patton
"""
import synapse as sy #importamos las librerias necesarias, llamamos sy para abreviar
import time

serial_port_list = sy.serial_port_scan() #escaneamos los puertos serie del ordenador

print serial_port_list[0][0]  # Primera lista [0] y primer elemento de la lista [0]
print serial_port_list[0][1]  # Primera lista [0] y segundo elemento de la lista [1]
print serial_port_list[0][2]  # Primera lista [0] y tercer elemento de la lista [2]
print "\n"

serial_port = sy.serial_port_select(serial_port_list)

ser, status = sy.serial_port_connect(serial_port)

if status:

    setpoint_list = [10,0,0,0,0,0] #Posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list) #Enviamos la posicion de reposo siempre
    time.sleep(2)  # Esperamos 2 segundos

    raw_input("Start")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[82,-5.5,0,15.8,17.5,19.5])  # Enviamos posiciones
    time.sleep(1)
    raw_input("Close")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[82,-5,11.4,15.8,17.5,19.5])  # Enviamos posiciones
    time.sleep(1)
    raw_input("Open")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[82,-5,0,15.8,17.5,19.5])
    time.sleep(3)
    sy.write_setpoint_list(ser,[82,-5,0,0,0,0])
    time.sleep(3)
    setpoint_list[0:2] = [10,0]  # Colocamos el pulgar en su posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)

    ser.close() #Cerramos la conexion con el puerto serie
else:
    print("Connection error") #Si no conseguimos cominicar con el puerto serie, sacamos un mensaje de error
```

```
"""
Created on Wed May 23 18:48:48 2018
name: Thumb-2 Finger
@author: Daniel Jason Castillo Patton
"""
import synapse as sy #importamos las librerias necesarias, llamamos sy para abreviar
import time

serial_port_list = sy.serial_port_scan() #escaneamos los puertos serie del ordenador

print serial_port_list[0][0]  # Primera lista [0] y primer elemento de la lista [0]
print serial_port_list[0][1]  # Primera lista [0] y segundo elemento de la lista [1]
print serial_port_list[0][2]  # Primera lista [0] y tercer elemento de la lista [2]
print "\n"

serial_port = sy.serial_port_select(serial_port_list)

ser, status = sy.serial_port_connect(serial_port)

if status:

    setpoint_list = [10,0,0,0,0,0] #Posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list) #Enviamos la posicion de reposo siempre
    time.sleep(2)  # Esperamos 2 segundos

    raw_input("Start")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[82,-5.5,0,0,16,16])  # Enviamos posiciones
    time.sleep(1)
    raw_input("Close")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[82,-5,11.6,12.3,16,16])  # Enviamos posiciones
    time.sleep(1)
    raw_input("Open")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[82,-5,0,0,16,16])
    time.sleep(1.5)
    sy.write_setpoint_list(ser,[82,-5,0,0,0,0])
    time.sleep(2)
    setpoint_list[0:2] = [10,0]  # Colocamos el pulgar en su posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)

    ser.close() #Cerramos la conexion con el puerto serie
else:
    print("Connection error") #Si no conseguimos cominicar con el puerto serie, sacamos un mensaje de error
```

```
"""
Created on Wed May 23 18:48:48 2018
name: Thumb 3-Finger
@author: Daniel Jason Castillo Patton
"""
import synapse as sy #importamos las librerias necesarias, llamamos sy para abreviar
import time

serial_port_list = sy.serial_port_scan() #escaneamos los puertos serie del ordenador

print serial_port_list[0][0]  # Primera lista [0] y primer elemento de la lista [0]
print serial_port_list[0][1]  # Primera lista [0] y segundo elemento de la lista [1]
print serial_port_list[0][2]  # Primera lista [0] y tercer elemento de la lista [2]
print "\n"

serial_port = sy.serial_port_select(serial_port_list)

ser, status = sy.serial_port_connect(serial_port)

if status:

    setpoint_list = [10,0,0,0,0,0] #Posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list) #Enviamos la posicion de reposo siempre
    time.sleep(2)  # Esperamos 2 segundos

    raw_input("Start")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[82,-5.5,0,0,0,12.6])  # Enviamos posiciones
    time.sleep(1)
    raw_input("Close")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[82,-5,11.1,11.8,10.7,12.6])  # Enviamos posiciones
    time.sleep(1)
    raw_input("Open")  # El programa espera hasta que se pulse intro
    sy.write_setpoint_list(ser,[82,-5,0,0,0,12.6])
    time.sleep(3)
    setpoint_list[5] = 0  # Colocamos los dedos en su posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(3)
    setpoint_list[0:2] = [10,0]  # Colocamos el pulgar en su posicion de reposo
    sy.write_setpoint_list(ser,setpoint_list)  # Enviamos posiciones
    time.sleep(1)

    ser.close() #Cerramos la conexion con el puerto serie
else:
    print("Connection error") #Si no conseguimos cominicar con el puerto serie, sacamos un mensaje de error
```