Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

"Prototipado y control de una mano protésica de bajo coste"

Autor: ***Iglesias García, David***

Tutor: ***Mansilla Gallo, Alberto***

Dpto. Ciencia de los Materiales e Ingeniería Metalúrgica, Expresión Gráfica en la Ingeniería, Ingeniería Cartográfica, Geodesia y Fotogrametría, Ingeniería Mecánica e Ingeniería de Procesos de Fabricación.

Valladolid, Junio del 2022.

1 RESUMEN Y PALABRAS CLAVE

En muchas ocasiones, se puede dar el caso de que, para mejorar la vida de un determinado paciente sea necesario realizar la amputación de un miembro de su cuerpo. Consecuentemente, este perderá así toda la movilidad que antes tenía en ese lugar.

En este trabajo, se desarrollará una prótesis robótica manual de bajo coste para aquellas personas que hayan podido sufrir una amputación de su mano derecha. Se incluirá el montaje y la programación del código que permitirá el control y movimiento de los dedos índice y pulgar. a través de dos sensores mioeléctricos colocados a lo largo del antebrazo.

ABSTRACT

On many occasions, it may be the case that to improve the life of a certain patient it is necessary to amputate a member of his body. Consequently, all the mobility that he previously had in that place is lost.

In this work, a low-cost manual robotic prosthesis will be developed for those who may have suffered an amputation of their right hand. It will include the assembly and programming of the code that will allow the control and movement of the index finger and thumb, through two myoelectric sensors placed along the forearm.

PALABRAS CLAVE

Mano protésica, sensor EMG, Arduino Nano, bajo coste, control.

KEYRORDS

Prosthetic hand, EMG sensor, Arduino Nano, low cost, control.

AGRADECIMIENTOS

A mi familia y mi pareja por su cariño y apoyo constante en todo lo que hago.

A mi tutor Alberto y al laboratorio por la ayuda prestada y la oportunidad de realizar este proyecto.

ÍNDICE

ÍNDICE DE FIGURAS

2 INTRODUCCIÓN Y OBJETIVOS

En estos momentos, la humanidad vive en un momento de auge tecnológico, la mayor parte de los utensilios que utiliza el ser humano en su día a día funcionan con algún tipo de ella. Se da por hecho que, a la hora de comprar cualquier tipo de producto, este será mejor y de más calidad si su precio es mayor, pero ¿es completamente cierto esto?

Por regla general, las prótesis robóticas suelen tener un elevado precio, debido normalmente a la necesaria aplicación de materiales ligeros y duraderos en su fabricación y, sobre todo, por el uso de tecnologías propietarias, es decir aquellas tecnologías en las que el usuario tiene limitadas sus posibilidades de usarlo y modificarlo, además de que su licencia suele conllevar un coste a mayores.

Las prótesis robóticas más avanzadas tienen un precio entre los 30000$ y los 70000$, pero se ha de tener en cuenta que tanto sus registros de movimientos, como materiales y modo de detección para el posterior accionamiento son mucho más complejos. Por otro lado, se encuentran las prótesis más básicas, las cuales son más baratas, pero también se ven limitadas generalmente en el rango de movimientos y calidad de los componentes. Tienen un precio entre los 2000$ y 5000$ [1].

Ambos tipos (que posteriormente veremos en más profundidad en el siguiente capítulo) incluyen en su precio generalmente los movimientos tanto de la muñeca, como de los dedos, y además de los materiales de la mano, los del antebrazo completo o parte de él, ya que la mayor parte de las amputaciones de las extremidades superiores son transradiales, es decir a cualquier altura del antebrazo.

La pérdida de una de las extremidades ya sea completa o parcial impide muchas de las actividades más frecuentes en el día a día. Esta situación no solo afecta al funcionamiento motriz del miembro en cuestión, sino que, el proceso de adaptación a este nuevo estado puede afectar psicológicamente al individuo.

Si se analizan los precios es fácil llegar a la conclusión de que no todas las personas pueden permitirse una prótesis robótica. Además de esto, si se tiene en cuenta las implicaciones a nivel de salud anteriormente vistas, se llega al objetivo principal del proyecto.

Se pretenderá desarrollar una prótesis robótica por un precio reducido, de modo que así pueda ser accesible al mayor número de personas, sin sacrificar calidad en los materiales ni en el modo de funcionamiento. Aunque cabe

destacar que en este proyecto solo se realizará lo equivalente para una persona con una amputación de tipo desarticulación de muñeca, es decir para aquellos pacientes a los que se le amputa la mano a la altura de la muñeca.

3 DESARROLLO DEL TFG

A lo largo del desarrollo del TFG se verán los modelos a encontrar en el mercado, la anatomía de la mano y el antebrazo, los distintos componentes electrónicos utilizados en la prótesis, el montaje de la propia, las conexiones de los componentes, la programación necesaria para realizar los movimientos oportunos de los dedos índice y pulgar y por último unas pruebas de funcionamiento.

3.1 Estado del arte.

En este apartado se analizarán los distintos tipos de prótesis que se pueden encontrar en el mercado. Se hará un análisis exclusivamente de las prótesis robóticas en miembros superiores, específicamente, las que se encuentran desde una amputación por desarticulación de codo a otra por desarticulación de muñeca, tal como indica la **figura 1.**

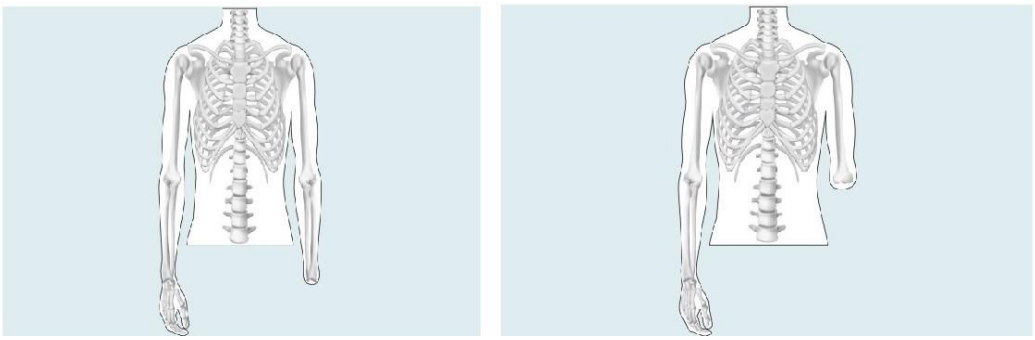


Figura 1. Izq. Desarticulación de muñeca, Dcha. Desarticulación de codo [2].

3.1.1 Tipos de prótesis de miembro superior

Existen 2 grandes tipos de prótesis de miembro superior en la actualidad, las estéticas y las funcionales. En este segundo grupo diferenciaremos distintos tipos dependiendo de la clase de alimentación necesaria para su funcionamiento.

Prótesis de mano Pasivas o Estéticas

Son meramente estéticas, replican la apariencia de una mano humana común pero no pueden realizar ningún tipo de movimiento. Suelen estar compuestas por materiales duraderos que reducen el mantenimiento de estos.

Por otro lado, los dedos se colocan en una posición de descanso, es decir ligeramente flexionados, tal como podemos ver en la **figura 2**.

Figura 2. Prótesis pasiva [3].

Prótesis de mano funcionales

Se centran en dar al usuario una experiencia funcional lo más parecida a una mano real. En cuanto a la estética de la prótesis, existe la posibilidad de añadir una malla o guante que dé una textura y apariencia parecida a una mano real. Aunque cabe destacar que las prótesis más avanzadas tienen un diseño muy similar al de una mano puramente estética. Pueden realizar distintos movimientos accionados de distintas maneras:

- **Prótesis mecánicas:** no tienen ningún componente eléctrico o electrónico, únicamente realizan los movimientos de acuerdo con un movimiento detonante como por ejemplo el movimiento de una articulación o un determinado músculo del cuerpo.

Figura 3. Prótesis mecánica [4]

Por esta razón también se las conoce como prótesis impulsadas por el cuerpo. El movimiento detonante hará que se produzca el estiramiento

de una cuerda, arnés o engranaje que producirá en la prótesis el único movimiento disponible. Tienen la desventaja de que el usuario puede experimentar fatiga en los músculos implicados en el movimiento detonante (**Figura 3**).

- **Prótesis eléctrica:** requieren de motores eléctricos y baterías para su uso y generalmente tienen forma de gancho para un mejor agarre. El usuario debe pulsar un interruptor cada vez que desee realizar un determinado movimiento. Suelen ser más costosas, pesadas, no permiten el contacto con un medio húmedo y requieren de un mantenimiento más constante, pero, por otro lado, también son más precisas y eficaces (**Figura 4**).

Figura 4. Prótesis eléctrica [5]

- **Prótesis mioeléctrica:** cada vez que un músculo se mueve genera una señal que es detectada por unos sensores, los cuales las traducen para que posteriormente se envíen los motores asociados y estos produzcan el movimiento correspondiente a dicha señal.

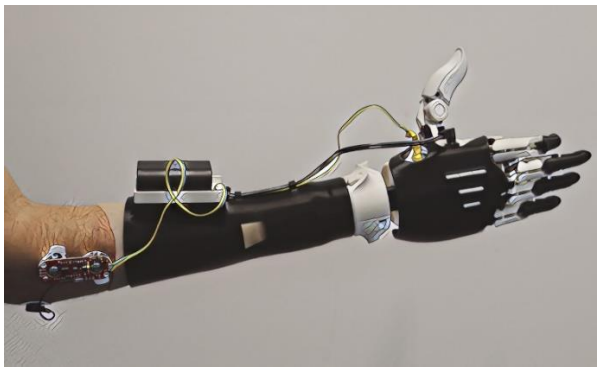


Figura 5. Prótesis mioeléctrica [6]

Son precisas, pero a su vez más complejas y caras. Pueden tener forma de gancho o de mano convencional, de forma que cada dedo disponga de un motor que lo mueva. Precisan de baterías para su funcionamiento por lo que esto desemboca en un incremento del peso y un mantenimiento constante. Dan la capacidad al usuario de poder controlar la fuerza y velocidad de los movimientos, por lo que son lo más parecido a una mano funcionalmente (**Figura 5**).

3.1.2 Las prótesis robóticas en el mercado actual

Ya que este proyecto describe una prótesis robótica mioeléctrica, se hará un análisis de mercado de este tipo en concreto, destacando finalmente tecnologías muy interesantes.

- **Michelangelo**
  Creada por la empresa Ottobock, es una prótesis para personas con una amputación transradial. Su estructura interna está formada por acero y duraluminio, mientras que para evitar oxidaciones e incrementar la resistencia al agua se recubre externamente con elastómero de silicona [7]. Tiene 7 posiciones diferentes de agarre, gracias a la posible rotación de la muñeca y la colocación del pulgar en oposición y abducción. Todos los dedos permiten movimientos de flexoextensión con abducción/aducción [8] (**Figura 6**).

Figura 6. Prótesis Michelangelo de Ottobock. [7]

- **BeBionic3 de RSLSteeper**
  Construida por la empresa RSLSteeper, es una prótesis de nuevo para pacientes que presentan una amputación transradial. Dispone de 5 motores lineales, uno para cada dedo. El pulgar se puede ajustar a dos posiciones diferentes manualmente, abducción y aducción. A través de un botón ubicado en la cara opuesta a la palma se cambia entre las 14 diferentes posiciones de agarre. La prótesis es de fibra de carbono y la empresa permite a los usuarios elegir un guante o recubrimiento total de silicona para una mejor estética. Por último, destacar que sus dedos

estás diseñados con curvatura para una mayor semejanza a unos dedos naturales [8] (**Figura 7**).

Figura 7. Prótesis Bebionic3 de RSLSteeper. [8]

- <u>i-Limb Quantum</u>
  Fabricada por la empresa puntera en prótesis Touch Bionics. Formada por titanio el cual proporciona resistencia a la corrosión a la vez que una buena relación fuerza-peso. Dispone de 5 dedos motorizados individualmente, con el pulgar que gira electrónicamente. El pulgar, a diferencia de los casos anteriores se coloca automáticamente en las posiciones de abducción y aducción. Tiene un total de 24 movimientos preprogramados y 12 personalizables a gusto del usuario. Se asocia cada movimiento a cada una de las posiciones que mantiene la muñeca al rotar o al flexionarse manualmente. Otras formas de tomar una posición concreta se llevan a cabo mediante la aplicación móvil desarrollada por la empresa, o por otro lado, colocando chips bluetooth en objetos con los que interactúa la prótesis al acercarse a ellos tomando el agarre asociado a ese chip [9] (**Figura 8**).

Figura 8. Prótesis i-limb Quantum de Touch Bionics. [9]

- Cyberhand

  Hasta ahora hemos visto prótesis mioeléctricas cuyos sensores estaban en contacto con la piel del paciente, en este caso encontraremos los electrodos conectados a las propias terminaciones nerviosas. De esta manera, el dispositivo permite al paciente sentir la presión y la temperatura a la que está sometida la prótesis, realizando así una realimentación [7]. Esto es beneficioso psicológicamente para el amputado, pues le ayudará a asemejar la prótesis a un brazo sin amputación, todo ello haciendo uso de la información que le llega al cerebro a través de los sensores. Este tipo de tecnología se encuentra en desarrollo actualmente (**Figura 9**).

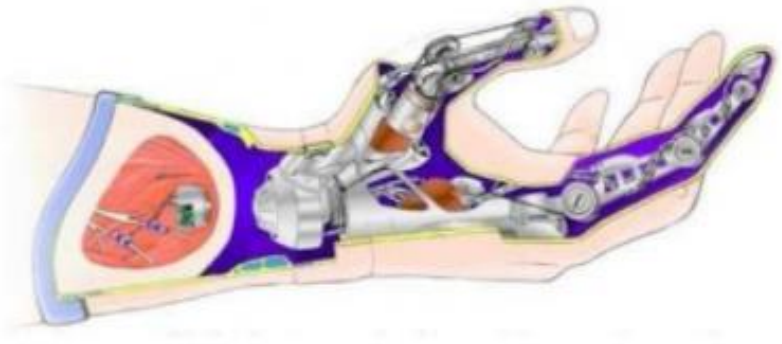


Figura 9. Prótesis Cyberhand. [7]

Por el momento, las prótesis mostradas son prótesis comerciales construidas con materiales costosos y con tecnologías realmente punteras. Por ello, los precios aproximados se encuentran entre los 40.000$ y 75.000$ (sin tener en cuenta la Cyberhand, cuya tecnología es aún más costosa) [10]. A continuación, se mostrarán una serie de modelos construidos en impresión 3D que permiten reducir el precio, facilitando así la accesibilidad a mayor número de pacientes económicamente hablando.

- Mark5s

  Esta prótesis está construida por la empresa surcoreana Mand.ro y tiene un precio aproximado de 1000$ sin incluir extras (no incluye sensores mioeléctricos, cargador y tampoco guante estético), lo cual, para posicionarlo en comparación con otros modelos, el precio final superará los 2000$. Cubre un gran rango de amputaciones, desde parcial de mano hasta por encima del hombro. En este caso se hará referencia al modelo de amputación transradial, el cual tiene 15 articulaciones, entre 5 y 8 patrones de agarre gracias a su giro de muñeca y el movimiento de abducción y aducción manual del dedo pulgar. Cuenta con un peso inferior a 1 kg y un cargador magnético sin necesidad de cableado. Por

último, cabe destacar que permite la incorporación de un guante cosmético ya sea de silicona o de tela [11] (**Figura 10**).

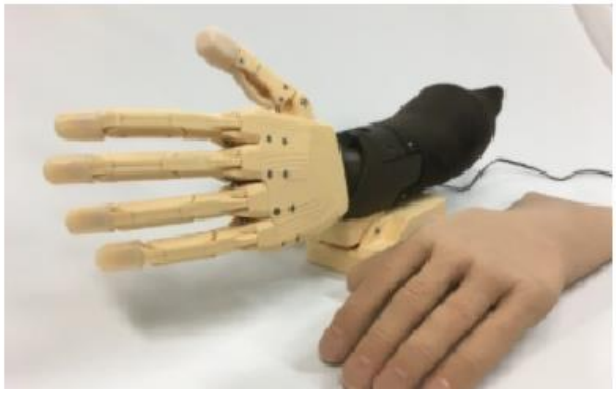


Figura 10. Prótesis Mark5s de Mand.ro. [11]

- <u>Hero Arm</u>
  Fabricada por Open Bionics, tiene un precio de 11.000€ [10]. Tiene un peso de 340 gramos y es capaz de sostener un peso de 8 kg, lo cual es comparable con la fuerza de sujeción de prótesis de coste alto. Por otro lado, dispone de 6 distintas posiciones diferentes de agarre, intercambiables a través de un botón localizado en la parta opuesta a la palma. Cuenta con pulgar y muñeca articulados, pudiendo esta última de realizar una rotación manual de 180º. Permite regular la velocidad de los movimientos al usuario e incorpora un modo de congelación de posición. Como novedad respecto al fabricante anterior, da al usuario información sobre errores y el estado de la prótesis a través de actuadores vibrotáctiles, luces y sonidos. Por último, mencionar que existe la posibilidad de intercambiar la apariencia de la prótesis mediante un total de 50 cubiertas magnéticas disponibles [12]. (**Figura 11**).

Figura 11. Prótesis Hero Arm de Open Bionics. [12]

Estas prótesis tienen la ventaja de que son más accesibles, pero por consiguiente sus funcionalidades se ven recortadas, ya sea en la duración de la batería, la calidad de los materiales, la fuerza soportada, la disminución del número de agarres disponibles o bien la precisión y comodidad de uso.

3.2 Anatomía de la mano y el antebrazo.

En este apartado se analizarán las distintos músculos, articulaciones, tendones y huesos que permiten el movimiento de los dedos de la mano. Es importante destacar que no todos ellos se encuentran en la propia mano, sino que algunos se localizan en la zona del antebrazo.

La mano es una de las partes del cuerpo más importantes en referencia a movilidad y manipulación de objetos. Cuando se encuentran sanas, permiten realizar gran variedad de tareas, desde movimientos delicados a otros que requieren del empleo de fuerza. Además de la movilidad, es uno de los más preciados medios de recopilación de información que tiene el cuerpo humano. Dicha captación se realiza a través del tacto, ya que las yemas de los dedos contienen infinidad de nervios. De ahí que cuando una persona pierde una de ellas se acude a las prótesis anteriormente analizadas.

La mano es la extremidad más distal del miembro superior, comprende desde la punta de los dedos hasta la muñeca, articulación a través de la cual se une al antebrazo. Las articulaciones son los lugares en los que se produce la unión de dos o más huesos, también permiten el movimiento.

Por otro lado, el antebrazo es otra de las partes en las que se divide las extremidades superiores. Está limitada por dos articulaciones, siendo una de ellas la muñeca y la otra el codo, a través del cual se une al húmero.

3.2.1 Huesos

El cuerpo humano tiene que estar constantemente en oposición a la gravedad para mantenerse erguido, por ello una de las principales funciones de los huesos es la estabilidad o sostén del cuerpo, dándolo así forma. Además de ello también forma parte del aparato locomotor junto con los músculos. Aunque los huesos son duros, también se trata de tejidos vivos pues tienen la capacidad de regenerarse al producirse alguna fisura o ruptura en ellos.

A) **Mano:** consta de un total de 27 huesos divididos en tres partes diferentes, tal como muestra la **figura 12.** Primero se pueden observar las falanges, se encuentran 3 en cada dedo, a excepción del dedo pulgar que contiene únicamente 2 falanges. Por otro lado, se encuentran los 5 huesos de la palma o el metacarpo, cada uno correspondiente a un dedo. Finalmente se pueden ver los 8 huesos del carpo o muñeca, divididos en dos filas.

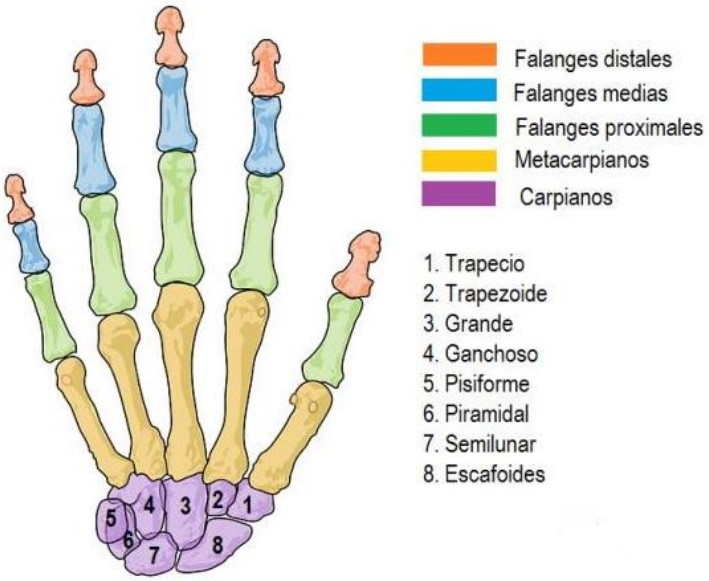


Figura 12. Distribución de los huesos de la mano [13].

B) **Antebrazo:** se compone de dos huesos, el cúbito que es el hueso más interno y el radio que es el más externo (**figura 13**). La cabeza del cúbito está unida con la muñeca mientras que la cabeza del radio hace lo propio con el húmero. En los movimientos de pronación y supinación es el radio el que gira alrededor del cúbito.

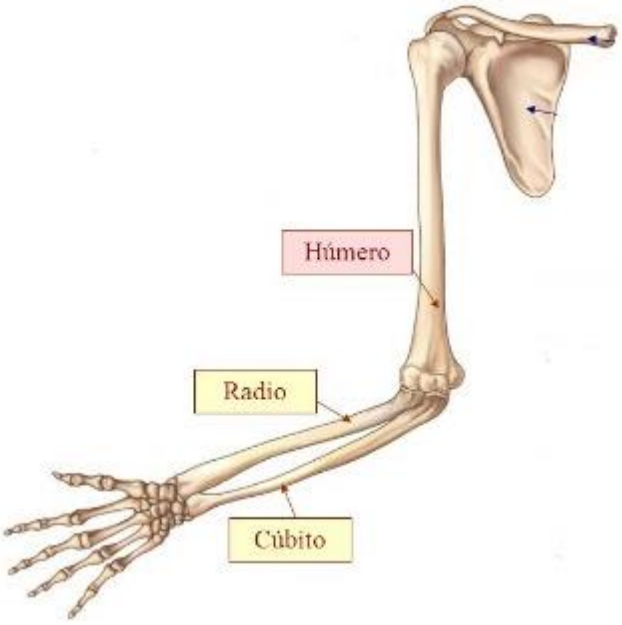


Figura 13. Distribución de los huesos del antebrazo [14].

3.2.2 Músculos

El otro gran componente del aparato locomotor lo forma el tejido muscular. Los músculos están formados por fibras musculares y se pueden clasificar en voluntarios e involuntarios, los primeros son los que se controlan conscientemente (Ej: los músculos que permiten el movimiento de la mano) mientras que los segundos son controlados inconscientemente por el sistema nervioso (Ej: el corazón). En este apartado se hará un análisis de los músculos voluntarios, ya que son los músculos que permiten el movimiento de los dedos.

Sus principales funciones son [15]:

- El movimiento del cuerpo.
- La protección de los huesos ante golpes.
- El mantenimiento de la postura.
- La generación de calor corporal.

Al igual que los huesos son tejidos vivos por lo que tienen la capacidad de regenerarse e incluso crecer durante la mayor parte de la vida de un ser humano. Los músculos están unidos a los huesos a través de los tendones, por lo que pueden tirar de ellos, pero no empujarlos de nuevo a su posición original, por ello los músculos suelen aparecer por pares flexores y extensores, ya que solo permiten el movimiento del hueso en un único sentido. Por ejemplo, para realizar un movimiento de flexión y extensión del codo, se necesitan dos músculos antagonistas, el bíceps (en este caso el flexor) y el tríceps (el extensor), tal como describe la **figura 14.**

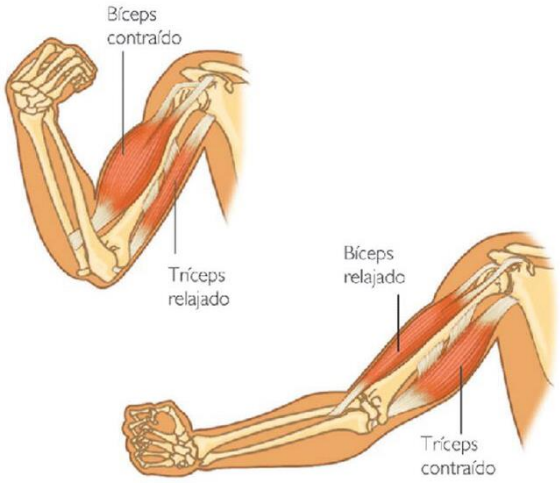


Figura 14. Flexión y extensión del codo [16].

Al realizar el movimiento de flexión el bíceps, se contraen sus fibras, disminuyendo su tamaño y aumentando su grosor, tirando así en este caso de los huesos del antebrazo. Mientras tanto el tríceps se encuentra relajado. Por otro lado, cuando se estira el codo, la contracción se realiza en las fibras del tríceps, tirando en sentido contrario al anterior de los huesos del antebrazo mientras el bíceps se relaja.

En la mano existen dos tipos de músculos clasificados por su origen y terminación. Encontramos por un lado los que se originan y terminan en la propia mano (**músculos intrínsecos**) y por el otro lado los que se originan en la muñeca y terminar en la mano (**músculos extrínsecos**). Como se ha comentado en apartados anteriores, el proyecto pretende desarrollar una prótesis que realice determinados movimientos con el dedo índice y pulgar de la mano derecha. Se hará un análisis de los músculos que permitan los movimientos más útiles de estos dedos, es decir los movimientos de abducción y aducción para el dedo pulgar y los de flexión y extensión para el dedo índice (**Figura 15**).

Figura 15. Izq. Movimientos del pulgar [17], Dcha. Movimientos del índice [18]

A) DEDO PULGAR

En la abducción y aducción del dedo participan músculos intrínsecos (algunos de los llamados músculos tenares) y extrínsecos [17]:

- Abducción: abductor largo del pulgar y abductor corto del pulgar.
- Aducción: aductor del pulgar y primer interóseo dorsal.

B) DEDO ÍNDICE

En la flexión y extensión del dedo participan músculos intrínsecos y extrínsecos [18]:

- Flexión: flexor de los dedos superficial y profundo y por último el primer lumbrical.
- Extensión: extensor común de los dedos, el extensor propio del índice y el primer interóseo tanto palmar, como dorsal.

En las **figuras 16, 17 y 18** se verán estos músculos en su ubicación concreta.

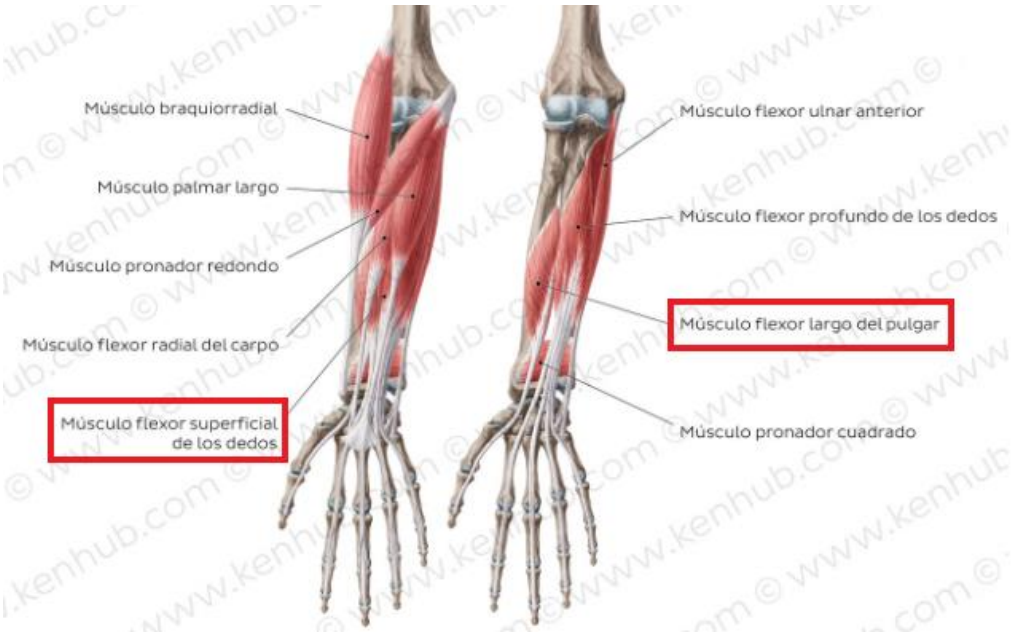


Figura 16. Vista anterior de la musculatura del antebrazo derecho [19].

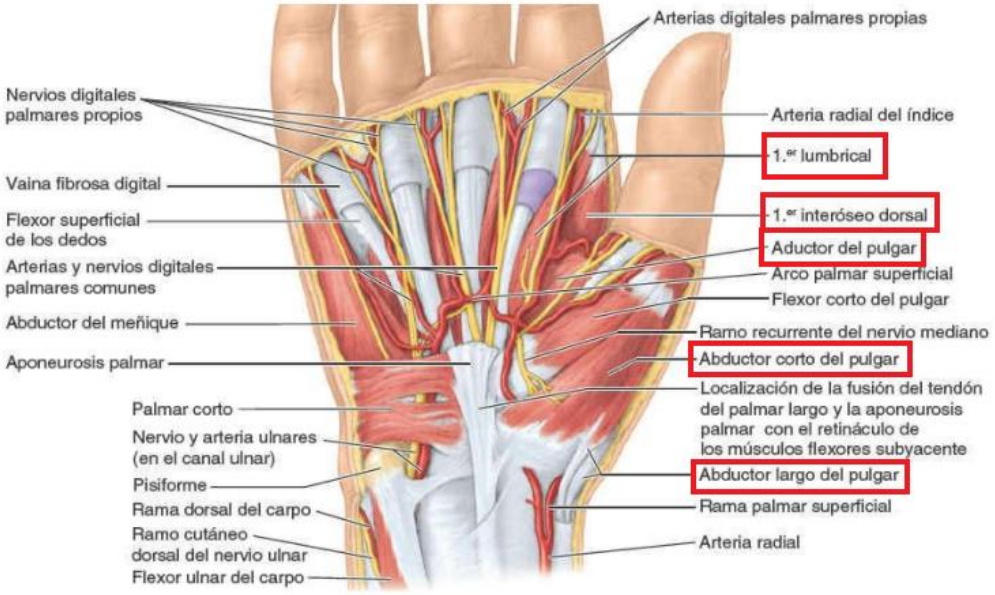


Figura 17. Vista de la musculatura de la mano derecha [17].

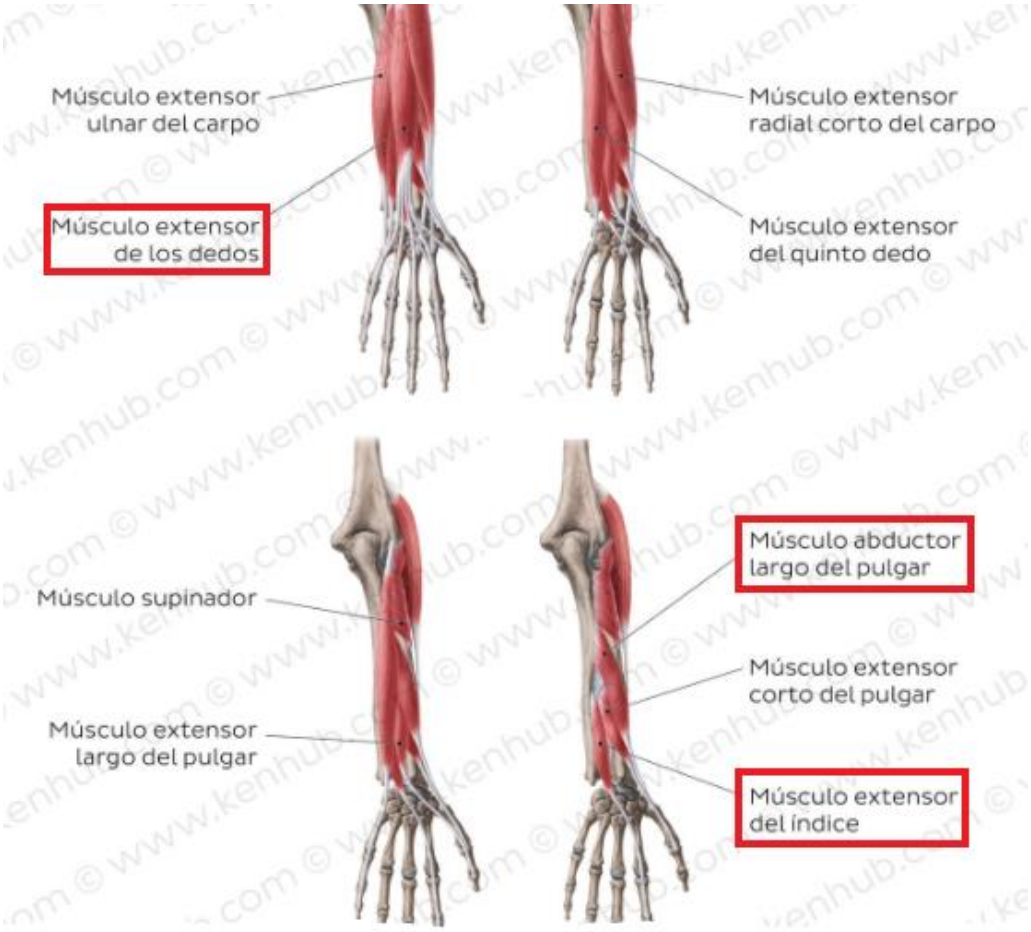


Figura 18. Vistas posterior superficial y profunda de la musculatura del antebrazo derecho [19].

El sujeto para el que está diseñada la mano, se le ha realizado una desarticulación de muñeca por lo que no dispone de los músculos intrínsecos de la mano, pero sí de los extrínsecos. Uno de los objetivos de los sensores mioeléctricos deberá ser la detección de la actividad muscular de dichos músculos para saber cuándo se han de mover los motores de la prótesis.

3.3 Componentes electrónicos.

En este capítulo se comentarán y analizarán detalladamente cada uno de los componentes electrónicos de la mano protésica. El fin principal del proyecto será mover los dedos índice y pulgar de una prótesis robótica a partir de las señales recogidas por unos sensores colocados en puntos del antebrazo que captan la actividad muscular. Utilizaremos motores para mover los dedos y un microprocesador para indicar los movimientos específicos en función de las señales de los sensores.

3.3.1 Arduino Nano

Antes de entrar en detalle con la placa de Arduino Nano es importante saber qué es Arduino. Se trata de una plataforma de creación electrónica de código abierto, que dispone de tanto software como hardware libre.

- Software libre: tipo de software en el que el usuario tiene libertad de uso, libertad para copiar o modificar (acceso al código fuente) y libertad para su distribución (con o sin modificaciones).

- Hardware libre: suelen manejarse a través de software libre y se trata de dispositivos hardware en los que el usuario tiene acceso a todo tipo de información sobre este.

El uso de este tipo de placas es muy beneficioso, ya que puedes basar tus proyectos en otros anteriormente compartidos a través de la comunidad de Arduino, de este modo existe una retroalimentación constante.

Cualquier placa Arduino facilita a través de su software y hardware la programación de un microcontrolador. Este se encargará básicamente de la lectura de los sensores que conectaremos a sus entradas (digitales o analógicas), el posterior procesamiento de la señal recogida y finalmente la escritura en las salidas, que normalmente están conectadas a todo tipo de actuadores.

En este proyecto se utilizará la placa Arduino Nano (**Figura 19**). Como su propio nombre indica dispone de unas dimensiones reducidas comparado con el resto de las placas disponibles en el mercado, lo cual es necesario ya que ocupará mucho menos espacio en la prótesis.

Entre las especificaciones técnicas del producto (**Figura 20**) se destaca el voltaje de entrada que puede ser entre 7V y 12V en continua, a pesar de que su tensión de funcionamiento será de 5V. Esto puede ser útil para utilizar a largo plazo como alimentación una batería (ya sea recargable o por pilas) ya que se dispone de más margen en el voltaje de entrada. Inicialmente se

alimentará a través de un puerto mini-USB que conectaremos al ordenador, este puerto suministra 5V.

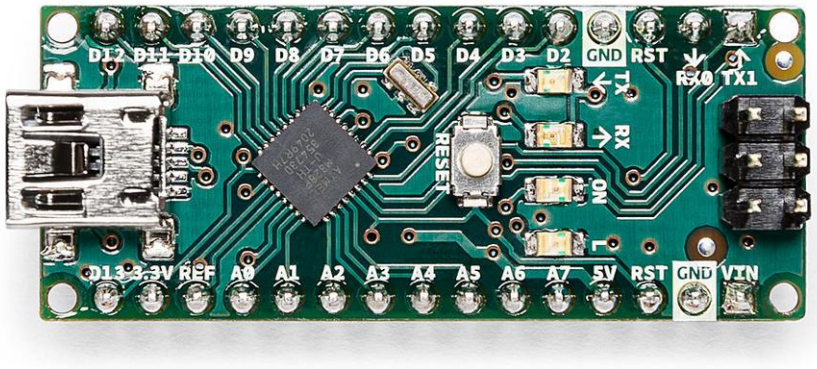


Figura 19. Vista superior de la placa Arduino Nano. [20]

MICROCONTROLADOR	ATmega328
ARQUITECTURA	AVR
TENSIÓN DE FUNCIONAMIENTO	5 V
MEMORIA FLASH	32 KB de los cuales 2 KB son utilizados por el gestor de arranque
SRAM	2 KB
VELOCIDAD DE RELOJ	16 MHz
PINES ANALÓGICOS IN	8
EEPROM	1 KB
CORRIENTE CC POR PINES DE E / S	40 mA (pines de E / S)
VOLTAJE DE ENTRADA	7-12 V
PINES DE E / S DIGITALES	22 (6 de los cuales son PWM)
SALIDA PWM	6
EL CONSUMO DE ENERGÍA	19 mA
TAMAÑO DE PCB	18 x 45 mm
PESO	7 g
CÓDIGO DE PRODUCTO	A000005

Figura 20. Especificaciones técnicas del Arduino Nano. [20]

Por otro lado, es muy ligero y cómodo pues todo lo necesario para que la placa funcione te viene incorporado en el kit de compra, es decir, con la placa, el cable mini USB y el programa gratuito “Arduino IDE” (entorno de desarrollo de Arduino) se pueden desarrollar infinidad de proyectos.

3.3.2 Motores.

En los dedos índice y pulgar de la prótesis se establecerán dos motores (**Figura 21**) para realizar los movimientos de abducción y aducción, en el caso del pulgar y de flexión y extensión, en el caso del índice. Se utilizará el mismo tipo de motor en ambos dedos para realizar dichos movimientos.

Figura 21. Actuador lineal eléctrico PQ12-63-12-P de ACTUONIX. [21]

El actuador eléctrico elegido es el PQ12-63-12-P de ACTUONIX, que se trata de un motor lineal eléctrico cuyas características son las indicadas en la **figura 22**:

PQ12 Specifications			
Gearing Option	30:1	63:1	100:1
Peak Power Point	15N@15mm/s	30N @ 8mm/s	40N @ 6mm/s
Peak Efficiency Point	8N @ 20mm/s	12N@12mm/s	20N @ 8mm/s
Max Speed (no load)	28mm/s	15mm/s	10mm/s
Max Force (lifted)	18N	45N	50N
Max Side Load	5N	10N	10N
Back Drive Force	9N	25N	35N
Stroke	20 mm		
Input Voltage	6 or 12 VDC		
Stall Current	550mA @ 6V, 210mA @ 12V		
Mass	15g		
Operating Temperature	-10°C to +50°C		
Positional Repeatability	±0.1mm		
Mechanical Backlash	0.25 mm		
Audible Noise	55dB @ 45cm		
Ingress Protection	IP-54		
Feedback Potentiometer	5kΩ±50%		
Limit Switches	Max. Current Leakage: 8uA		
Maximum Duty Cycle	20%		

Figura 22. Especificaciones del actuador PQ12-63-12-P. [21]

La relación de transferencia es de 63:1, con esto se hace referencia a la relación que existe entre los engranajes internos del motor. Para clarificar este concepto se utilizará como ejemplo el mecanismo de la **figura 23.** En la relación de transmisión intervienen dos engranajes, uno impulsor y otro impulsado, en el caso de la figura la relación es 2:1, por lo que cada dos vueltas del impulsor el impulsado dará una. Con esta afirmación es lógico llegar a la conclusión de que el engranaje impulsor gira más deprisa, pero por consiguiente el engranaje impulsado aportará un torque (par) mayor ya que su brazo de aplicación de fuerza es mayor. Se trata de una relación de reducción ya que la velocidad angular del engranaje motor es mayor que la del engranaje movido.

En el caso del motor seleccionado, la relación es de 63:1, por lo que comparado con los otros dos modelos con relaciones de transmisión 30:1 y 100:1, se encuentra en un término medio entre una buena velocidad de giro del motor y un buen par aplicado. El motor 30:1 proporcionará mayor velocidad de movimiento, pero un torque inferior, mientras que el motor 100:1 proporcionará menor velocidad de movimiento, pero un torque mayor.

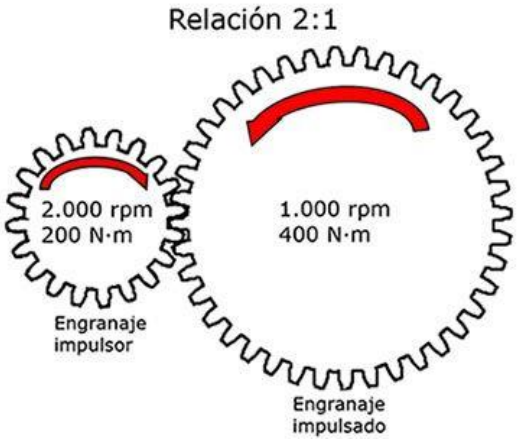


Figura 23. Relación de transmisión entre dos engranajes. [22]

Por otro lado, en la **figura 24**, se puede observar la relación existente entre la fuerza que proporciona el motor con la velocidad lineal y la intensidad eléctrica requerida. Cuando se menciona la fuerza en dichas gráficas, se hace referencia a la fuerza necesaria para mover la carga en cuestión, en el caso de este proyecto se trata de la fuerza necesaria para mover los dedos y la carga que puedan estar soportando los mismos. Proporcional a la fuerza necesaria para mover la carga, se demandará una corriente eléctrica que aumentará con el valor de dicha fuerza, mientras que, en el caso de la velocidad lineal, también existirá una relación proporcional con la carga, pero ésta disminuirá con el aumento del valor de la carga.

El motor PQ12 está diseñado para empujar o tirar de una carga a lo largo de su trayectoria lineal, pero si la energía eléctrica aplicada al motor se desconecta,

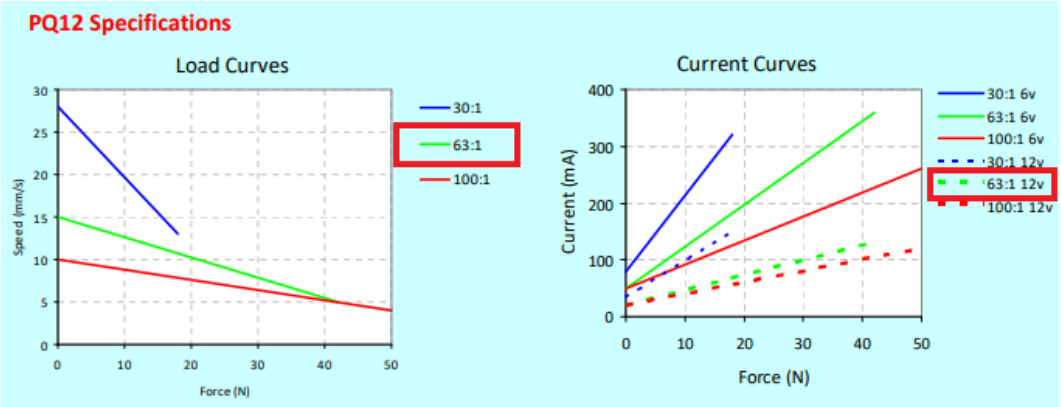


Figura 24. Curvas de Velocidad-Fuerza e Intensidad-Fuerza. [21]

el motor mantendrá su posición a no ser que la carga aplicada exceda la fuerza de retroceso ("**Back drive force"**), que tal y como nos indica la tabla de características de la **figura 22** es de 25N. Por otro lado, la fuerza máxima de empuje (fuerza máxima en extensión del motor) es de 45N. Tiene su punto de máxima potencia en los 30N de carga y 8mm/s de velocidad y la longitud de su carrera es de 20mm.

También cabe destacar que la **corriente de bloqueo** o más conocida como "**Stall Current**" es de 210mA. Este valor indica la corriente que demanda el dispositivo cuando la velocidad de movimiento es nula pero el par es máximo, es decir, el actuador se encuentra bloqueado. En este estado el dispositivo se sobrecalienta y si se consume esta corriente durante un tiempo lo suficientemente largo (que sobrepase el ciclo de trabajo) el motor puede dañarse. Como se puede observar en la **figura 24**, los 210mA son bastante superiores a la corriente que demanda el motor con la carga máxima.

Por otro lado, se pasará a describir el **ciclo de trabajo** del motor o más conocida como "**Duty Cycle**" que en este caso se indica que es del 20% como máximo. El ciclo de trabajo indica el porcentaje de tiempo que está en funcionamiento el motor en comparación con el tiempo de reposo.

Se calcula dividiendo el tiempo de trabajo del actuador (T-on) entre la suma del tiempo de trabajo más el tiempo en reposo (T-on + T-off) y multiplicando el resultado por 100, por ejemplo, si se tiene un actuador trabajando durante 25 segundos, pero luego descansa a lo largo de 75 segundos, el ciclo de trabajo será del 25%. Sobrepasar el ciclo de trabajo continuamente y con cargas altas puede reducir considerablemente la vida útil del motor (**Figura 25**).

De los varios tipos de actuadores PQ12, se ha elegido el PQ12-63-12-P. El número 63 corresponde a la relación de transferencia que ya ha sido explicada, por otro lado, el número 12 correspondo al voltaje de alimentación del motor, que será como indica el nombre de 12V y, por último, se encuentra la letra P.

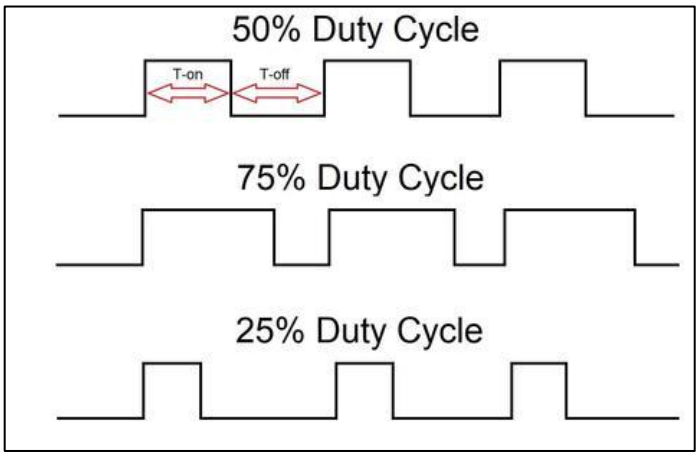


Figura 25. Ejemplo de ciclo de trabajo. [23]

Existen 3 tipos de letras R, S y P, cada una correspondiente a un tipo de control del motor. Se ha elegido el modelo P, ya que es muy útil la retroalimentación que da el dispositivo a cerca de la posición, mediante la variación del voltaje asociado a un potenciómetro cuyo valor cambia con la posición de la carrera del actuador. Se detallará este funcionamiento posteriormente cuando se analice la programación.

3.3.3 L293D.

El componente L293D es un circuito integrado y tiene las funciones de puente H. Este tipo de dispositivos son útiles para permitir que los motores de corriente continua giren en ambos sentidos (en este caso para realizar la extensión y contracción del dedo robótico). En este caso podemos manejar motores que se alimenten entre 4.5 y 36V, suministrando 600mA de corriente de salida, con la capacidad de aguantar picos de 1.2A. Su temperatura de operación se encuentra entre los -40 y los 150ºC.

La descripción del funcionamiento general de un puente H se analizará a partir de la **figura 26**. En el diagrama, el componente J1 controla los pines 1 y 2. Ambos tienen la posibilidad de ponerse a nivel alto o bajo y en función de dichos valores el motor girará en un sentido u otro. Pero antes de nada se deberá aclarar cuál es el funcionamiento de un transistor (se indican en la imagen como Q1, Q2, Q3 y Q4). Los transistores son dispositivos electrónicos que a grandes rasgos se componen de 3 secciones semiconductoras, cada una unida a un pin, el emisor, el colector y la base (C, E y B en la **figura 27**) dejan pasar la corriente desde el emisor al colector únicamente cuando le llega la suficiente

corriente a la base. Los diodos se utilizan para proteger a los transistores de las contracorrientes provocadas por el motor, ya que en el fondo se trata de una carga inductiva.

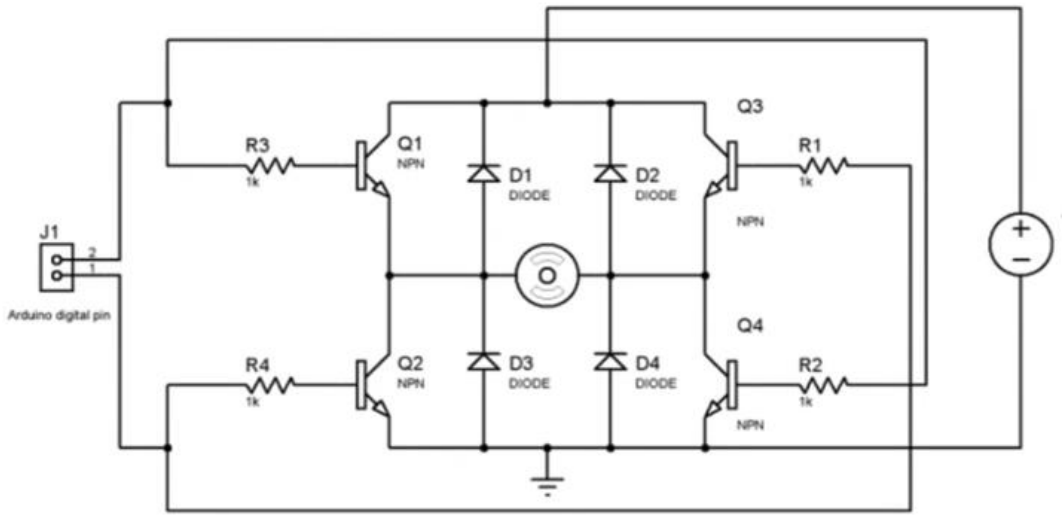


Figura 26. Diagrama general de un puente H. [24]

Se verán los desencadenantes de los dos sentidos de giro por separado:

- Nivel alto en el pin 1 y nivel bajo en el pin 2: los transistores Q1 y Q4 dejan pasar la corriente ya que sus bases están conectadas al pin 1, el cual está a nivel alto, por lo que el motor gira en uno de los sentidos.
- Nivel alto en el pin 2 y nivel bajo en el pin 1: los transistores Q2 y Q3 dejan pasar la corriente ya que sus bases están conectadas al pin 2, el cual está a nivel alto, por lo que el motor gira en uno de los sentidos.

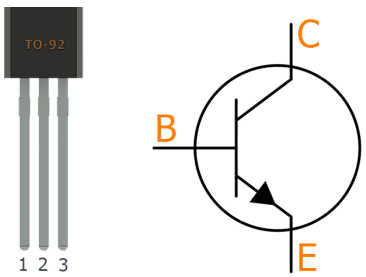


Figura 27. Pines de un transistor. [25]

Una vez aclarado el funcionamiento de un puente H, se analizará el circuito integrado L293D. La **figura 28** nos muestra la disposición real de los 16 pines. Con este dispositivo se tiene la posibilidad de controlar 2 motores con giro en ambas direcciones, es decir disponer de dos puentes H completos o, por otro

lado, el control de 4 motores con giro en una sola dirección, es decir utilizar para cada motor la mitad de un puente H. En este proyecto interesa la primera posibilidad para poder realizar los movimientos complementarios de cada dedo. El funcionamiento de cada pin es el siguiente (los pines pertenecientes a cada lateral del circuito integrado serán propios de cada motor, a excepción de los pines 8 y 16 que son comunes para ambos motores):

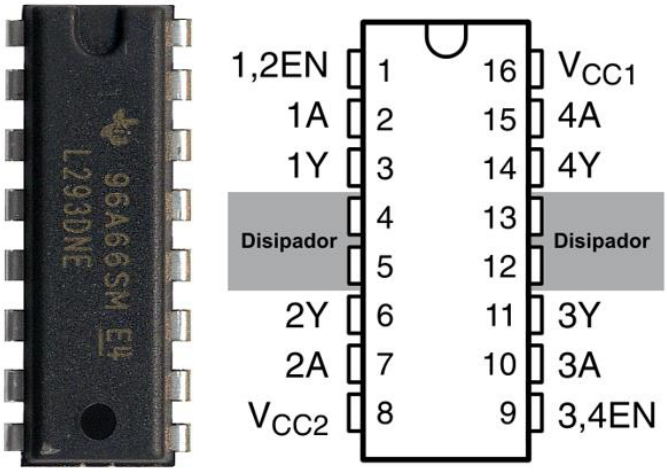


Figura 28. Pines y aspecto del circuito integrado L293D. [26]

- Pines **1** y **9**: son dos entradas nivel alto-bajo, que permitirán habilitar los puentes H. Se tiene la posibilidad de realizar también un control de la velocidad de los motores a través de la tensión entregada al motor. Esto se conseguirá mediante la modulación por ancho de pulso o más bien conocido como PWM, conectaremos pines de salida PWM de Arduino a estos pines.

- Pines **2**, **7**, **10** y **15**: son entradas nivel alto-bajo de nuevo y sirven para controlar los sentidos de giro de ambos motores, así como sus frenados rápidos.

- Pines **3**, **6**, **11** y **14**: son pines de salida y se encargan de hacer llegar potencia a los motores.

- Pines **4**, **5**, **12** y **13**: en ellos se encuentra la conexión a tierra de ambos motores. También se pueden conectar disipadores a estos pines, pero en este caso no será necesario pues la intensidad demandada que genera un calentamiento no es lo suficientemente grande como para provocar el deterioro del dispositivo.

- Pin **8**: pin de alimentación de los dos motores, en este caso se administrarán 12V.

- Pin **16**: pin de alimentación del circuito electrónico interno, básicamente el que regula los transistores. Se conectará a 5V.

Por otro lado, la **figura 29** muestra el interior del dispositivo detalladamente. Relacionando lo visto anteriormente con el funcionamiento del puente H se explicará el funcionamiento para un solo motor, pues la extensión al segundo motor es obvia, ya que dispone de los mismos pines, pero con diferente número, los cuales han sido analizados y relacionados anteriormente.

Comparando la **figura 29** con la **figura 26**, los pines 2 y 7 serán la analogía del componente J1, de manera que al establecerlos en valor alto o bajo a través de su conexión con la placa Arduino, los transistores dejarán o no pasar corriente. Los pines 3 y 6, por otro lado, serán las conexiones del motor con los

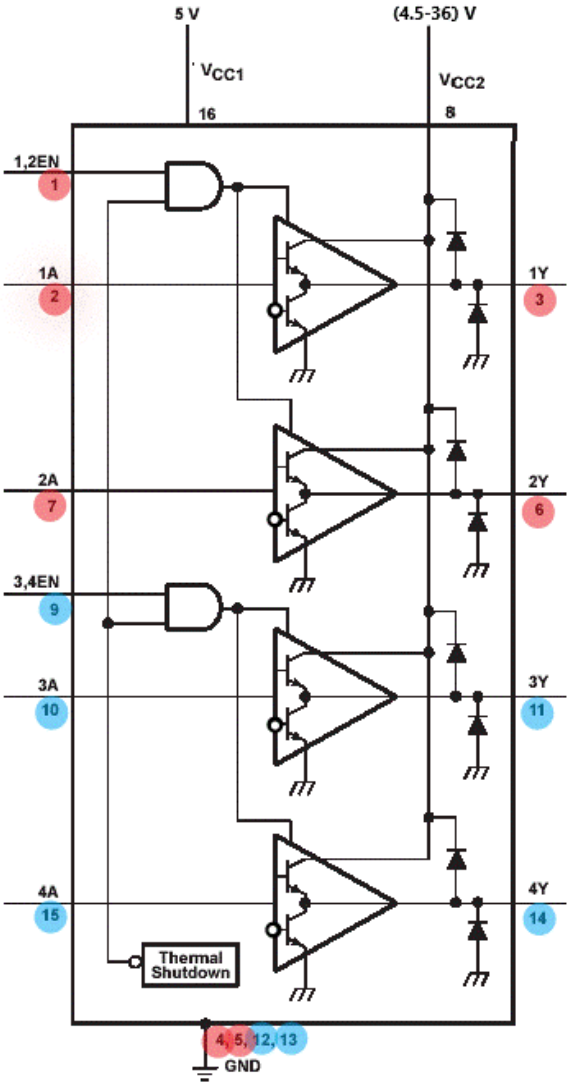


Figura 29. Diagrama del circuito interno del L293D. [26]

transistores. La señal del pin 1 está unida a la señal de apagado térmico (thermal shutdown) en una puerta AND. Esta unión permite habilitar las entradas 2 y 7 únicamente cuando pongamos en valor alto el pin 1 y la señal del apagado térmico esté activa (cuando la temperatura del circuito quede fuera del rango establecido por el fabricante la señal de apagado térmico se pondrá en nivel bajo y los transistores no conducirán).

Finalmente se indica un esquema de la lógica de entradas y salidas que permite establecer dicho dispositivo (**Figura 30**). Aplicándolo al proyecto en cuestión el giro a la derecha o izquierda supondrá la flexión o extensión para el caso del dedo índice, y por otro lado la abducción o aducción del pulgar.

PIN 1/9	PIN 2/10	PIN 7/15	FUNCIÓN
H	L	H	Giro a la derecha
H	H	L	Giro a la izquierda
H	L	L	Detención rápida
H	H	H	Detención rápida
L	X	X	Detención rápida

L= bajo, **H** = alto, **X** = no afecta

Figura 30. Lógica control de movimientos del motor. [26]

3.3.4 Sensores mioeléctricos.

El cuerpo humano se controla por el sistema nervioso a través de impulsos eléctricos, que se envían desde el cerebro hasta el órgano en cuestión a través de los nervios. Funcionalmente podemos hacer una analogía entre los nervios y conductores eléctricos, cerebro y un microprocesador como Arduino o los músculos y unos motores.

Cuando se tiene la intención de mover por ejemplo un dedo, el cerebro manda un impulso eléctrico a través de los nervios hasta llegar a los músculos del dedo, que harán el movimiento requerido. El problema se encuentra cuando una persona pierde la mano, ya que el impulso no puede llegar hasta los músculos del dedo. Por suerte, existen músculos relacionados con los movimientos de la mano y sus dedos que no se encuentran en la propia mano, sino que se extienden a lo largo del antebrazo, estos son los músculos extrínsecos, que ya se vieron en la sección 3.2.2.

El objetivo de los sensores mioeléctricos es detectar la actividad de estos músculos, para posteriormente indicar al microprocesador que se deben mover

los motores de los dedos. En este proyecto se utilizarán los sensores 13E200=50 de Ottobock, **figura 31**.

Generalmente, la señal captada por este tipo de sensores debe recibir un posterior tratamiento para adaptarla al tipo de señal con la que opera un controlador, se verán a continuación cada uno de ellos.

Figura 31. Sensores mioeléctricos 13E200 de Ottobock. [27]

MEDICIÓN

Durante el proceso de contracción y relajación de un determinado músculo, se produce una señal electromiográfica (básicamente, la señal eléctrica de un músculo). Los sensores se colocan en las zonas del antebrazo donde se encuentran estos músculos intrínsecos, posteriormente al producirse el movimiento en el músculo, la señal mioeléctrica atraviesa la piel que, aunque no conduce la electricidad como un metal, también es un material relativamente conductor al estar formado por tejido con solución electrolítica. Seguidamente la señal llega a los electrodos del sensor formados por una superficie conductora metálica.

Un problema es que como la piel posee resistencia al paso de la corriente, para reducirla y poder captar mejor la señal, es necesario aplicar un gel conductor sobre la zona donde se coloque el sensor.

AMPLIFICACIÓN Y FILTRADO

La señal recogida por los electrodos del sensor es de rango muy bajo, entre los µV y los mV, por lo que es normal que interferencias ya sea de origen técnico o biológico puedan afectar a la señal y provocar una interpretación falsa de los resultados. La solución está en la realización de un tratamiento de

amplificación de la señal para llegar a los voltajes manejados por Arduino (entre 0V y 5V) y un filtrado de ruido.

La amplificación se debe hacer adecuadamente, ya que los propios circuitos amplificadores pueden afectar a la señal original al ser esta tan débil. Se suelen utilizar amplificadores de instrumentación, los cuales amplifican únicamente la diferencia entre las señales recogidas por los electrodos, siendo estos diseñados específicamente para aplicaciones en la medicina. Con esta manera de proceder, si dichos electrodos se encuentran relativamente cerca, los ruidos serán captados por ambos dos y por lo tanto serán eliminados.

Por otro lado, se deben eliminar determinadas señales conocidas, pero antes se llevará la señal original del dominio del tiempo al de la frecuencia. Para ello hay que entender que cualquier señal, por muy compleja que sea se puede expresar como composición de senos y cosenos, es decir funciones periódicas de distintas frecuencias y amplitudes. Las señales con pendientes grandes o cambios rápidos de voltaje se representarán como frecuencias muy altas, mientras que las señales más suaves son indicadoras de frecuencias más bajas.

Se sabe que hay señales de ruido e interferencias conocidas que operan a unas frecuencias específicas, además de que las frecuencias entre las que se encuentran las señales electromiográficas se encuentran entre los 10 y los 500Hz. Existen señales de ruido inferiores a los 15Hz que corresponden al movimiento de objetos que no interesan, al igual que las conocidas señales de 50Hz y 60Hz, correspondientes por un lado a la corriente de la red eléctrica y por el otro a una gran parte del ruido ambiente que absorbe la piel.

Aplicando filtros se consigue reducir el ancho de banda de las frecuencias de la señal, consiguiendo así la señal óptima para analizar posteriormente en el microcontrolador.

En el caso concreto del sensor escogido en este proyecto presenta todas estas etapas de tratamiento de la señal. Sus características se muestran en la **figura 32**. El ancho de banda de frecuencia se encuentra entre los 90Hz y los 450Hz, por lo que los filtros internos evitan las señales de ruido anteriormente mencionadas. Los electrodos son de titanio puro, lo que los hace aptos para personas alérgicas. Además, el voltaje de operación del sensor se encuentra entre los 4.8 y los 7.2V por lo que es compatible con el voltaje de 5V que nos da Arduino. Finalmente hay que destacar que, según el fabricante, funcionan bien incluso con entrada de señal muscular baja, ya que tienen sensibilidad ajustable [27].

Número de artículo	**13E200 = 50**
Peso	4,5 g
HZ	50
Tensión de funcionamiento	4,8 - 7,2 V
Dimensiones LxAnxAl	27 x 18 x 9,5 mm
Ancho de banda de frecuencia	De 90 a 450 Hz
Temperatura ambiente	-15 a +60 grados C
	No apto para uso en América del Norte.

Figura 32. Características del sensor 13E200=50. [27]

3.4 Montaje.

En cuanto al montaje de la prótesis, se han aprovechado los ligeros materiales que proporciona la impresión 3D y que a su vez dan fiabilidad necesaria para aguantar ciertas condiciones de tensión, compresión y temperatura. Se utilizarán dos tipos de materiales en función de las necesidades:

- **PLA** o también denominado ácido poliláctico. Es un material termoplástico formado a partir del almidón de maíz. Consta de gran dureza y buena resistencia tanto a la tracción como a la compresión. Es muy cómodo de usar, pues su temperatura de impresión es baja comparada con otros materiales. Destaca por ser reciclable, biodegradable y no producir ningún tipo de emisión al medio ambiente. Como inconvenientes se encuentran su baja resistencia a la temperatura (entorno a los 60-70ºC) y baja flexibilidad.

- **TPU** o también conocido como poliuretano termoplástico. Se caracteriza por ser un material flexible, aunque también tiene buena resistencia a la abrasión y es duradero. Se imprime a una temperatura baja en comparación a otros materiales, pero su flexibilidad dificulta su impresión. A partir de los 80ºC comienza a perder su flexibilidad. Es también reciclable.

Figura 33. Piezas impresas con impresora 3D, Brunel Hand. Recuadradas en azul se encuentran las piezas impresas con material TPU.

Como sus descripciones indican, se utilizará el primero de ellos para las piezas rígidas y el segundo para piezas flexibles. Una vez impresas todas ellas (ver **figura 33**) se procederá al montaje de las mismas según nos indica la guía de ensamblaje [28]. No se detallarán todos los pasos del montaje ya que estos se pueden encontrar en la guía anteriormente citada, se hará un análisis de los aspectos clave y más importantes. Para ello, se necesitarán a mayores una lista de componentes entre los que se incluyen distintos tipos de tornillos, muelles, inserciones, pasadores y un hilo de pesca trenzado que sea resistente. Se pueden diferenciar un mecanismo diferente para cada dedo, aunque ambos están basados en la utilización del hilo de pesca a modo de ligamento.

La idea principal es hacer los movimientos de flexión y extensión para el dedo índice y, de abducción y aducción para el dedo pulgar. Para ello, se ha de tener en cuenta que una cuerda atada a un lugar móvil permite realizar solo un movimiento de tensión. La fuerza que provoca la tensión se hará desde el motor, por lo que para devolver al dedo a su posición de reposo se utilizarán distintos tipos de resortes.

3.4.1 Dedo índice.

Se colocará en el motor correspondiente al dedo índice un resorte cónico tal y como indica la **figura 34** izquierda. La función de este será evitar vibraciones en los mecanismos internos del motor.

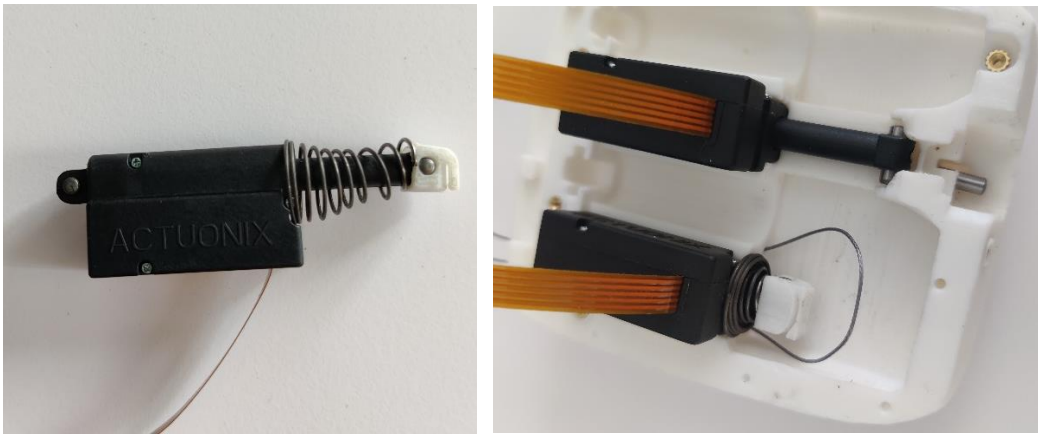


Figura 34. Ensamblaje dedo índice 1.

Además, se unirá una de las piezas al motor a través de un pasador, de modo que el hilo de pesca se amarrará a esta pieza para realizar los movimientos del dedo (**Figura 34** derecha).

Por otro lado, se va pasando el hilo por el interior de las piezas impresas que forman el dedo, ya que estas se diseñaron con un hueco interior con este propósito. Donde se encontraría la yema del dedo se amarra el hilo a un tornillo

plano y este se aprieta para asegurarse de que no quede libre la cuerda (**Figura 35**).

Figura 35. Ensamblaje dedo índice 2.

Como se ha comentado antes, para volver a la posición en la que el dedo está extendido una vez se ha flexionado, se han de emplear resortes de extensión entre las piezas que forman las falánges del dedo (**Figura 36**).

Figura 36. Ensamblaje dedo índice 3.

3.4.2 Dedo pulgar.

En este caso no se utilizará un resorte cónico unido al motor, pero si un pasador y una cuerda amarrada a este. La diferencia con el mecanismo utilizado en el dedo índice se encuentra en que en este caso la cuerda atravesará otros tres pasadores, uno de ellos en el lado de la palma (**Figura 37** derecha) y los restantes en el lado contrario (**Figura 37** izquierda). La función de estos pasadores será girar cuando el motor tense la cuerda, de modo que el hilo no roce con las piezas, evitando así que este se pueda romper.

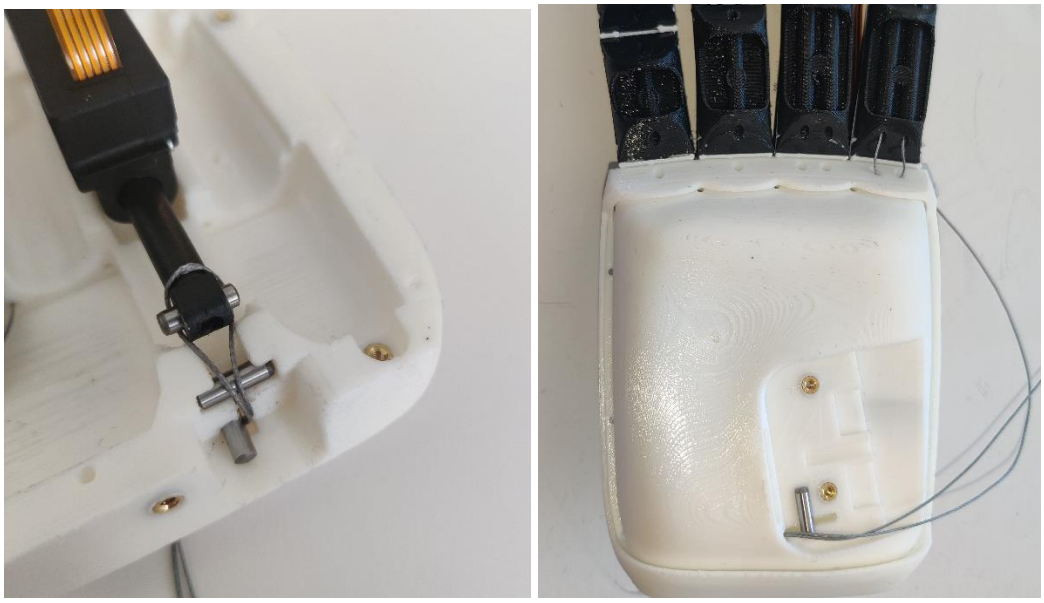


Figura 37. Ensamblaje dedo pulgar 1.

Por otro lado, se irá realizando el ensamblaje de las piezas del dedo pulgar de modo que la pieza que forma la única falange del dedo se unirá a la pieza que simula al metacarpiano. Hemos de tener en cuentra que el movimiento buscado del dedo pulgar es el de abducción y aducción, por lo que a pesar de que el dedo humáno tenga dos falanges, con una única podremos realizar el movimiento. La unión de las piezas anteriormente mencionadas se hace como en el caso del dedo índice con un resorte de extensión, para poder así restablecer la posición tras la abducción (**Figura 38**).

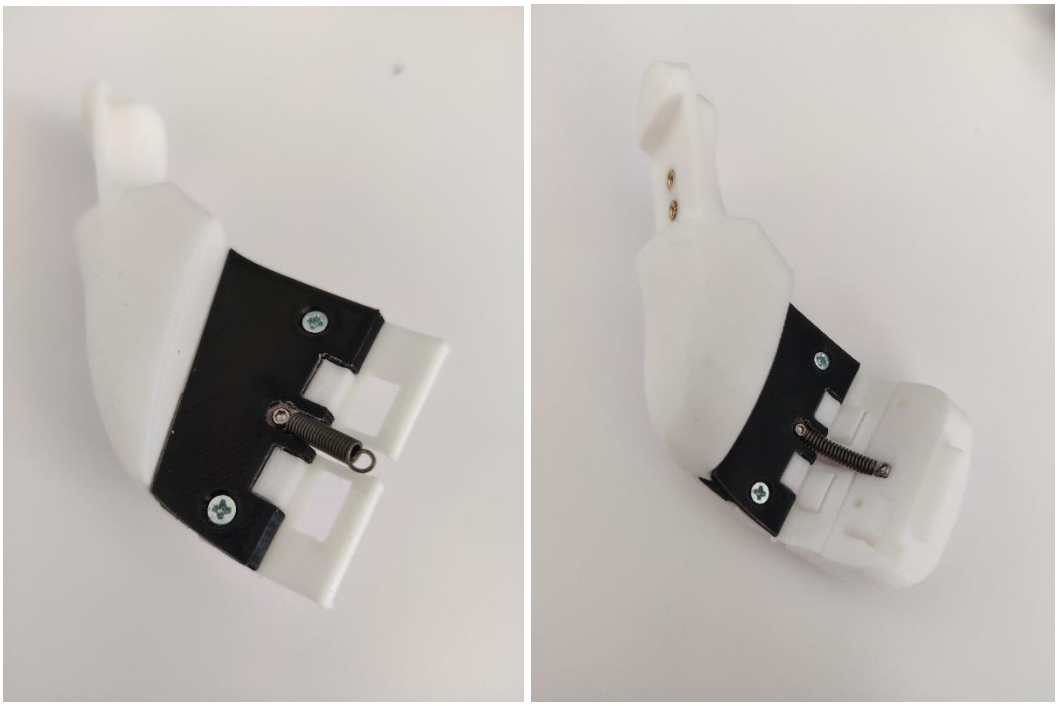


Figura 38. Ensamblaje dedo pulgar 2.

Para terminar, se unirá el dedo pulgar al resto de la mano y simultáneamente se introduce el hilo de pesca a lo largo del dedo por los orificios habilitados. De la misma forma que en el dedo índice, se fijará la cuerda a través de un tornillo plano (**Figura 39**).

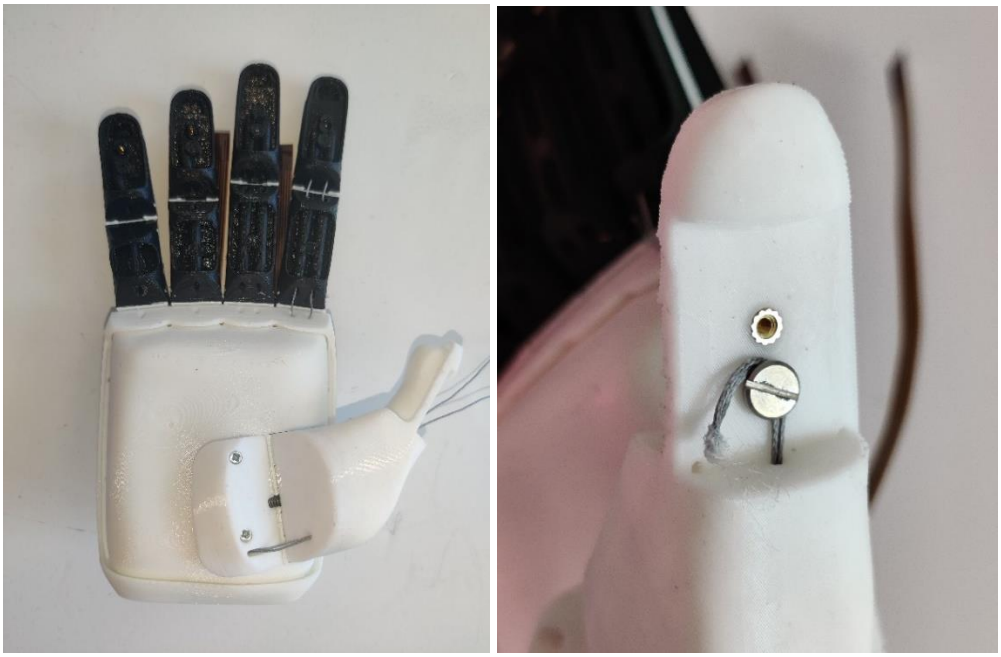


Figura 39. Ensamblaje dedo pulgar 3.

Una vez explicado los dos mecanismos, se procederá a finalizar el montaje de la mano protésica. Para ello se taparán los motores utilizando para lograrlo unas piezas a modo de tapa. A través de un orificio, podremos sacar los cables de conexión de los motores, de manera que se puedan conectar a la placa de Arduino sin necesidad de retirar la tapa (**Figura 40**).

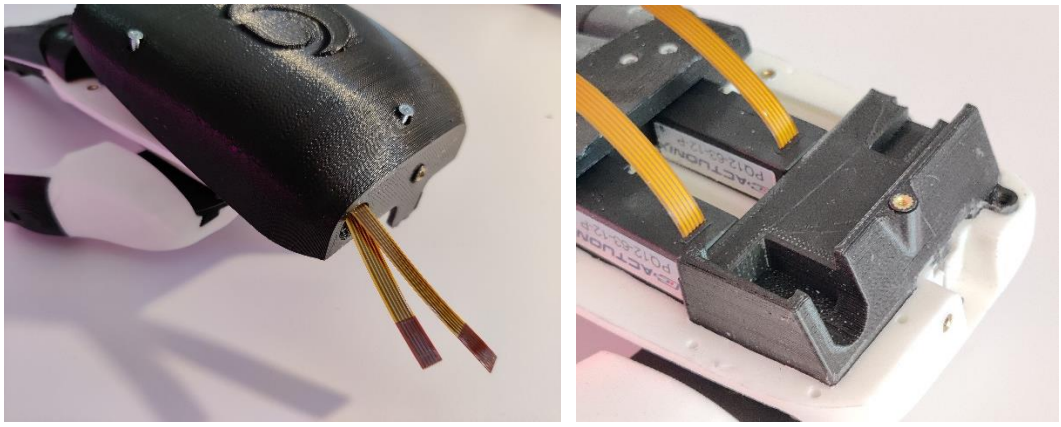


Figura 40. Ensamblaje de la mano protésica.

Como resultado final se obtiene una mano protésica con materiales reciclables y resistentes a las condiciones de uso que puede experimentar cualquier humano en su vida cotidiana. El resultado final se puede observar en la **figura 41.**

Figura 41. Aspecto final de la prótesis.

3.5 Conexiones.

Como se ha descrito en el apartado 3.3, es necesario el uso de seis dispositivos distintos para que la prótesis funcione a la perfección (dos motores, dos sensores, la placa Arduino nano y el circuito integrado L293D). En este apartado se tratarán las conexiones a realizar entre componentes.

Para una mayor comodidad se utilizará una protoboard como la que muestra la **figura 42**. Se trata de una placa de pruebas que sirve como soporte para el montaje de circuitos sin necesidad de soldar componentes, por lo que es reutilizable. Está formada por matrices de agujeros conectados eléctricamente en los que se introducen los terminales de los componentes. Existen cuatro secciones diferentes, dos centrales y dos extremas. En las centrales, cada línea vertical de cinco agujeros es una pieza metálica, mientras que en los extremos + y – de cada lado, cada línea horizontal de veinticinco agujeros es una pieza metálica.

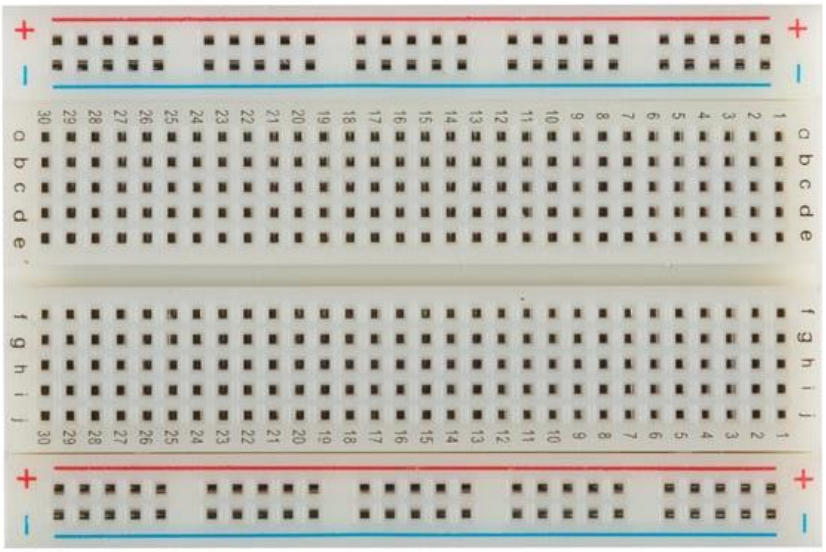


Figura 42. Placa Protoboard. [29]

De esta forma, se pueden conectar a los agujeros de las líneas de los extremos pines de componentes con voltajes comunes, tales como los 5V de alimentación o voltaje a tierra. Por otro lado, se podrán insertar directamente circuitos integrados o placas pequeñas, como el L293D o la Arduino Nano en este caso. Los componentes quedarán todos en un espacio muy recogido, facilitando tanto las conexiones como su almacenamiento (ver anexo 6.1).

Los motores PQ12-P se conectarán a un adaptador como el que muestra la **figura 43**. Este permite separar los distintos pines del motor, para poder posteriormente aislarlos en un cable y poder conectarlos a la protoboard en el lugar adecuado. Por otro lado, también permite aumentar la longitud del cable

y poder así llevar las conexiones a una distancia mayor como, por ejemplo, desde el interior de la prótesis donde se encuentran los motores a el exterior de ella, donde se encontrará la protoboard.

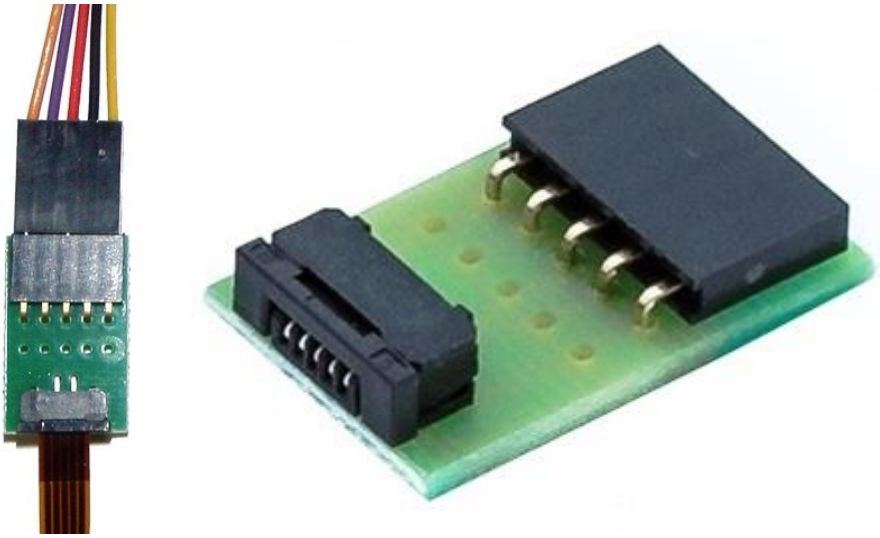


Figura 43. Adaptador motor PQ12-P. [30]

Antes de mostrar el esquema de conexiones, se ha de indicar el modo de alimentación de los motores. Como se ha visto en la descripción del circuito integrado L293D, el pin 8 va conectado eléctricamente a 12V. Este voltaje se puede conseguir de varios modos, a través de baterías, pilas en serie (8 pilas alcalinas AAA) o con una fuente de alimentación. Para este proyecto se empleará la fuente, pero sería necesario utilizar cualquiera de los otros modos de alimentación cuando la prótesis fuera funcional y aplicable a un paciente determinado.

El esquema de conexiones se encuentra en el anexo 6.1 y nos detalla específicamente donde se conecta cada pin de cada uno de los componentes de la prótesis. Se analizarán las conexiones por componente, explicando el funcionamiento de cada uno de los pines.

SENSORES MIOELÉCTRICOS 13E200

Los sensores tienen 3 pines de conexión, los cuales se enumerarán de izquierda a derecha tomando como lado referencia al ver la cara azul del cable en vez de la gris.

- **Pin 1:** se conectará a los pines de entrada analógicos de Arduino A3 y A5. Este pin transfiere al microcontrolador la señal que el sensor ha recogido y posteriormente amplificado y filtrado. Como dicha señal no toma un único valor es necesario conectarla a una entrada analógica en lugar de una digital.

- **Pin 2:** conexión a tierra.
- **Pin 3:** conexión a 5V.

MOTORES PQ12-P

En este caso encontraremos un total de 5 pines a conectar, los cuales se enumerarán de izquierda a derecha tomando como referencia el sentido de la flecha amarilla apuntando hacia arriba.

- **Pin 1**: pin de referencia negativa del potenciómetro de retroalimentación. Se conectará a tierra.
- **Pines 2 y 3**: pines de potencia del motor. En el caso del motor correspondiente al dedo índice se conectarán a los pines 3 y 6 del L293D, respectivamente, mientras que en el motor restante se conectarán a los pines 11 y 14 del mismo componente.
- **Pin 4**: pin de lectura del valor del potenciómetro de retroalimentación. Se conectará a la entrada analógica A0 (motor dedo índice) y A3 (motor dedo pulgar) del microcontrolador Arduino nano.
- **Pin 5**: pin de referencia positiva del potenciómetro de retroalimentación. Se conectará a 5V.

CIRCUITO INTEGRADO L293D

El funcionamiento de cada pin se analizó en el apartado 3.3.3, ya que era necesario para explicar el objetivo del uso del componente en la prótesis. En este apartado solo se detallará el lugar de conexión concreto de cada uno de los 16 pines.

- **Pin 1**: conexión a la entrada digital D9 de Arduino.
- **Pin 2**: conexión a la entrada digital D10 de Arduino.
- **Pin 3**: conexión al pin 2 del motor del dedo índice.
- **Pin 4**: conexión a tierra.
- **Pin 5**: conexión a tierra.
- **Pin 6**: conexión al pin 3 del motor del dedo índice.
- **Pin 7**: conexión a la entrada digital D11 de Arduino.
- **Pin 8**: conexión a 12V.

- **Pin 9**: conexión a la entrada digital D6 de Arduino.
- **Pin 10**: conexión a la entrada digital D5 de Arduino.
- **Pin 11**: conexión al pin 2 del motor del dedo pulgar.
- **Pin 12**: conexión a tierra.
- **Pin 13**: conexión a tierra.
- **Pin 14**: conexión al pin 3 del motor del dedo pulgar.
- **Pin 15**: conexión a la entrada digital D3 de Arduino.
- **Pin 16**: conexión a 5V.

3.6 Programación.

En este apartado se realizará un análisis del código de programación, cuyo objetivo será poder realizar los respectivos movimientos de los dedos índice y pulgar protésicos, pero únicamente cuando los sensores mioeléctricos detecten actividad en los músculos extrínsecos del antebrazo que se encuentren relacionados con el movimiento de dichos dedos.

El código está desarrollado en el entorno propio de Arduino, es decir el Arduino IDE. El lenguaje de programación utilizado es propio de la plataforma, pero está basado en C++. Se puede encontrar dicho código al completo en el Anexo 6.2.

Se dividirá la programación en varios apartados, cada uno correspondiente a un objetivo concreto del código general.

3.6.1 Pines y variables

Antes de comenzar con el programa, en muchos lenguajes de programación se parte realizando una declaración de variables que luego se usarán a la hora de programar. En el caso de Arduino es necesario a mayores, asociar a cada pin de salida o entrada del que se vaya a hacer uso (ya sea este analógico o digital), el nombre de una constante que sea lo suficientemente descriptiva para representar la función del pin. Los números corresponden a pines digitales, mientras que los que comienzan por A seguido de un número pertenecen a pines analógicos.

Se dará nombres a los pines de los sensores (**Figura 44**), del motor perteneciente al dedo índice y del motor perteneciente al dedo pulgar. La descripción de la función de cada uno de ellos se ha realizado en el apartado 3.5 de conexiones.

```
//Pines de los sensores
const int sensorIndice = A3;
const int sensorPulgar = A5;

//Pines del motor del dedo indice
const int habilita_Indice = 9;
const int control_A_Indice = 10;
const int control_B_Indice = 11;
const int pin_pos_Indice = A0;

//Pines del motor del dedo pulgar
const int habilita_Pulgar = 6;
const int control_A_Pulgar = 5;
const int control_B_Pulgar = 3;
const int pin_pos_Pulgar = A1;
```

Figura 44. Fragmento del código donde se declaran los pines.

Por otro lado, se deben declarar las distintas variables a utilizar a lo largo del programa. Las correspondientes a la **figura 45**, se describirán al analizar la función de procesamiento de la señal de los sensores mioeléctricos.

```
// Valores de la señal sin procesar
int valueMuscleI;
int valueMuscleP;

// Valores de la señal procesada
float muscleActivityI = 0;
float muscleActivityP = 0;

// Tiempo de ejecucion (ms)
long t = 0;

// Variables y parametros filtro paso alto
float fc1 = 100; // Frecuencia de corte
float tau1 = 1 / (2 * 3.1415 * fc1);
float y1_1 = 0;
float x1_1 = 0;
float y2_1 = 0;
float x2_1 = 0;

// Variables y parametros filtro paso bajo
float fc2 = 5; // Frecuencia de corte
float tau2 = 1 / (2 * 3.1415 * fc2);
float z1_1 = 0;
float z2_1 = 0;

// Variables calculo de la media movil
int buffLength = 20; // aprox. 20*2 ms
int i1 = 0;
float buffer1 [20] = {0.0};
float sum1 = 0;
int i2 = 0;
float buffer2 [20] = {0.0};
float sum2 = 0;
```

Figura 45. Fragmento del código donde se declaran las variables del procesamiento de la señal mioeléctrica.

Además de las variables anteriores, se disponen de otras correspondientes a los controles de los dos motores:

- posicion_X. En ella se almacenará la posición del dedo X (pulgar o índice). Inicialmente estará igualada a 0, pero tomará un valor entre 0 y 1023 (rango de las entradas analógicas de Arduino) según se encuentre el motor en una posición a lo largo de su recorrido.
- contraMAX_X / extenMAX_X. Los motores tienen un rango de movimiento con dos posiciones extremas, una corresponderá al valor 1023 y otra al 0 de la variable de posición. Pero se ha de tener en cuenta que la colocación de los motores en la prótesis no permitirá llegar a esos valores extremos por razones físicas, por lo que se deberán establecer unos valores de extensión y contracción máxima para que no se produzcan choques con otras piezas.

Como se puede observar en la **figura 46** los valores de las contracciones y extensiones máximas se diferencian de un motor a otro. Al echar un vistazo a la colocación de los motores en la **figura 34** derecha y teniendo en cuenta el funcionamiento de los mecanismos de movimiento de cada dedo, se puede llegar a una justificación de los valores anteriormente mencionados. Tal y como muestra la **figura 47**, la extensión máxima del dedo índice (extensión) se corresponde con la contracción máxima de la carrera del motor, mientras que

la contracción máxima del dedo índice (flexión) hace lo propio con la extensión máxima de la carrera del motor.

```
// Valores de control del motor del dedo indice
int posicion_Indice = 0; // posicion del potenciometro
int contraMAX_Indice = 50; // valor del potenciometro en la contracion maxima
int extenMAX_Indice = 800; // valor del potenciometro en la extension maxima

// Valores de control del motor del dedo pulgar
int posicion_Pulgar = 0; // posicion del potenciometro
int contraMAX_Pulgar = 750; // valor del potenciometro en la contracion maxima
int extenMAX_Pulgar = 200; // valor del potenciometro en la extension maxima
```

Figura 46. Fragmento de código donde se encuentran las variables de control del movimiento de los motores.

En el caso del dedo pulgar, su extensión máxima (aducción) corresponde a la extensión máxima de la carrera del motor, mientras que, por otro lado, la contracción máxima del dedo pulgar (abducción) coincide con la contracción máxima de la carrera del motor.

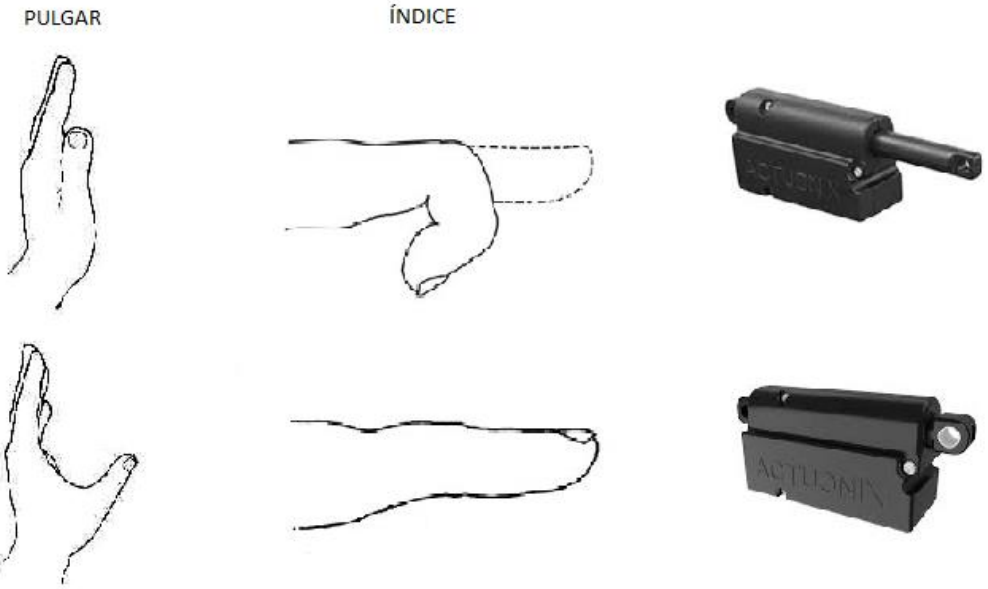


Figura 47. Posiciones de los dedos en función de la contracción o la extensión de la carrera del motor.

3.6.2 Programa. *void setup()*

En este apartado se realizan las configuraciones de los pines (**Figura 48**), indicando si funcionarán como salida o entrada, para ello se utilizará el comando "pinMode(pin, INPUT/OUTPUT)". Los pines de los sensores se configurarán como entradas, mientras que los pines de control de ambos dedos como salida.

```
void setup() {

  //abre el puerto Serie, configura los datos a 9600 bps
  Serial.begin(9600);
  //Se configuran las entradas y salidas
  pinMode(sensorIndice,INPUT);
  pinMode(sensorPulgar,INPUT);
  pinMode(control_A_Indice, OUTPUT);
  pinMode(control_B_Indice, OUTPUT);
  pinMode(control_A_Pulgar, OUTPUT);
  pinMode(control_B_Pulgar, OUTPUT);
  pinMode(habilita_Indice, OUTPUT);
  pinMode(habilita_Pulgar, OUTPUT);

  //Se habilita el puente H
  digitalWrite(habilita_Indice, HIGH);
  digitalWrite(habilita_Pulgar, HIGH);

}
```

Figura 48. Fragmento de código en el que se programa la configuración de los pines.

Por otro lado, en esta función se pueden programar aquellas acciones que necesiten una sola ejecución. Por ello se puede encontrar el "Serial.begin(9600)" y los "digitalWrite" que habilitan el puente H del L293D.

El primero de ellos se utiliza para indicar al microprocesador que se requiere de una comunicación de datos con el puerto en serie a lo largo de la ejecución. Dicha comunicación se realizará a una velocidad de 9600 baudios, es decir, 9600 bits por segundo.

En segundo lugar, los pines habilitadores del puente H tal y como hemos expresado antes, se configuran como salidas ya que son pines de control de los motores de los dedos. Recordando lo mencionado en el apartado 3.3.3, donde se explica el funcionamiento del puente H, los pines 1 y 9 deben establecerse a nivel alto para permitir el giro de los motores en ambos sentidos (**Figura 30**). Por ello con el comando "digitalWrite(pin, LOW/HIGH)", se ponen a nivel alto.

3.6.3 Programa. *void loop ()*

En la función loop colocaremos aquellas acciones que queremos que se repitan en bucle constantemente. Cuando se enciende la placa de Arduino, se comienza por la ejecución única del setup y seguidamente se pasa a repetir constantemente la ejecución del loop un número infinito de veces, hasta que la placa se apague o se cargue otro programa distinto.

En dicha función bucle se pueden encontrar otros tipos de funciones, cuya llamada y ejecución se realizan en el propio loop, pero su código se encuentra externamente a este.

3.6.3.1 FUNCIONES

Las funciones son utilizadas principalmente para mostrar más claridad a la hora de analizar el código en cuestión. Disponemos de 2 tipos de funciones, una relacionada con el procesamiento de la señal de los sensores mioeléctricos y otras 5 relacionadas con los distintos movimientos del dedo. Para el análisis de estas 5, será de gran utilidad fijarse en la lógica de control de movimientos de la **figura 30**.

Función *void parar()*

Se trata de una función que sirve para que el motor se detenga en el momento en el que sea ejecutada. Para ello se establece un nivel bajo en ambos pines de control, los cuales pasaremos como argumento en el caso de que queramos parar el dedo índice o el pulgar (**Figura 49**).

```
void parar (int pin_1, int pin_2)
{
  digitalWrite(pin_1, LOW);
  digitalWrite(pin_2, LOW);
}
```

Figura 49. Función void parar().

Función *void contraerIndice()* y *void extenderIndice()*

En este caso se provoca una extensión o contracción del dedo índice mediante la variación del sentido de giro de los engranajes internos del motor. Para llevarlo a cabo se juega de nuevo con la escritura a nivel bajo/alto (contracción) o alto/bajo (extensión) en los pines de control del dedo índice. A mayores se establece una velocidad (la cual se pasa como argumento de la función) de movimiento del motor mediante el comando "analogWrite(pin,valor)" (**Figura 50**).

```
void contraerIndice (int velocidad)
{
  analogWrite(habilita_Indice, velocidad);
  digitalWrite(control_A_Indice, LOW);
  digitalWrite(control_B_Indice, HIGH);
}
```

```
void extenderIndice (int velocidad)
{
  analogWrite(habilita_Indice, velocidad);
  digitalWrite(control_A_Indice, HIGH);
  digitalWrite(control_B_Indice, LOW);
}
```

Figura 50. Funciones void contraerIndice() y void extenderIndice()

Función *void contraerPulgar()* y *void extenderPulgar()*

Para el caso del dedo pulgar, la programación de las funciones y su descripción es similar a la del dedo índice, con un ligero cambio. La contracción y extensión se producen de manera opuesta al caso del dedo índice, es decir, nivel alto/bajo (contracción) o bajo/alto (extensión) (**Figura 51**). Esto es provocado por el hecho descrito en la **figura 47**.

```
void contraerPulgar (int velocidad)
{
  analogWrite(habilita_Pulgar, velocidad);
  digitalWrite(control_A_Pulgar, HIGH);
  digitalWrite(control_B_Pulgar, LOW);
}
```

```
void extenderPulgar (int velocidad)
{
  analogWrite(habilita_Pulgar, velocidad);
  digitalWrite(control_A_Pulgar, LOW);
  digitalWrite(control_B_Pulgar, HIGH);
}
```

Figura 51. Funciones void contraerPulgar() y void extenderPulgar()

Función *void processEMG()*

Por último, se analiza la función encargada del filtrado de la señal proporcionada por los sensores. Como se ha visto en apartados anteriores, los sensores ya se encargan de hacer un filtrado de la señal, pero para mejorar el resultado se realizará un filtrado digital que elimine las frecuencias que puedan introducir los componentes electrónicos.

Las entradas digitales de Arduino disponen de 10 bits, lo que se traduce en una resolución total de 1024 valores. A medida que la señal va tomando valores entre 0-5V, se mapean entre 0-1023, de manera que se trabajan con esos valores en la programación. El conjunto de algoritmos que se utilizarán en la función tratará de filtrar la señal correspondiente a los valores ubicados en los márgenes anteriormente nombrados. Para ello se harán 3 procesos distintos: un filtro paso alto, un filtro paso bajo y un ajuste de media móvil.

Se sacará provecho del código empleado en la programación de la mano High Five [31].

```
void processEMG()
{
  // Measure time step
  float dt = 0.001 * (millis() - t);
  t = millis();

  // Read raw myoelectric signals
  valueMuscleI = analogRead(sensorIndice);
  valueMuscleP = analogRead(sensorPulgar);
```

Figura 52. Inicio de la función void processEMG().

Los filtros electrónicos suelen emplear una resistencia de valor R ohmios y un condensador de capacidad C faradios, que se conectarán como indica la **figura 53** en función de si se demanda un filtro paso bajo o paso alto. En este código se pretende hacer un simil digital a este tipo de filtros electrónicos.

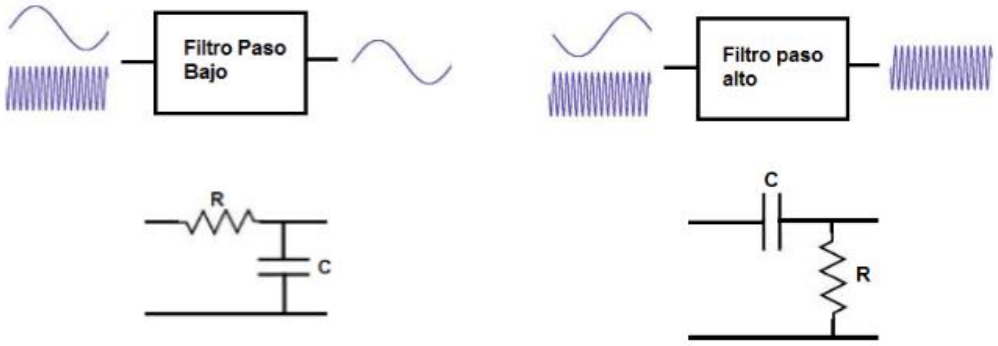


Figura 53. Izquierda, filtro paso bajo. Derecha, filtro paso alto. [32]

Para realizar el filtrado se aplicará el método del filtro paso bajo y/o alto de media móvil exponencial [33]. Inicialmente se definirá un diferencial de tiempo (dt), el cuál cambiará su valor cada vez que se llame a la función. También se calculará el tiempo transcurrido desde que se inició la ejecución del programa a través de la función "millis()"y seguidamente se leerán las señales de los sensores mioeléctricos, tal y como muestra la **figura 52**.

Filtro paso alto y rectificación.

Los filtros paso alto son menos comunes y se utilizan para eliminar las posibles frecuencias bajas que pueda tener la señal (**Figura 54**). Para ello, definimos el valor de "tau1", el cual se trata de la constante de tiempo medida en segundos y dependiente de la frecuencia de corte (100Hz). En los filtros electrónicos, tau es el valor de la resistencia multiplicada por la capacidad del condensador (tau=RC).

Aplicamos el filtro a través del cálculo de "value1filt" y "value2filt", donde "y1_1" e "y2_1" son los valores filtrados anteriores y "x1_1" y "x2_1" son los valores muestreados por los sensores, aunque inicialmente en la primera iteración los 4 valores están inicializados a 0. Seguidamente hacemos una rectificación de las señales elevando al cuadrado sus valores filtrados.

```
// High-pass filtering (remove signal mean value)
float valuelfilt = taul / (taul + dt) * yl_1 + taul / (taul + dt) * (valueMuscleI - xl_1);
yl_1 = valuelfilt;
xl_1 = valueMuscleI;
float value2filt = taul / (taul + dt) * y2_1 + taul / (taul + dt) * (valueMuscleP - x2_1);
y2_1 = value2filt;
x2_1 = valueMuscleP;

// Signal rectification and squaring
float valuelfiltrect = pow(valuelfilt, 2);
float value2filtrect = pow(value2filt, 2);
```

Figura 54. Filtro paso alto y rectificación, función processEMG().

Filtro paso bajo.

Lo que se pretenderá es eliminar las frecuencias altas de la señal, es decir los picos pronunciados, consiguiendo así una curva más suave (**Figura 55**). Para ello, se define el valor "tau2", cuyo fin es el mismo que en el del filtro de paso alto, pero en este caso la frecuencia de corte es de 5Hz.

El filtro se aplica sobre la señal ya rectificada, y se realiza calculando "value1env" y "value2env".

```
// Low-pass filtering
float valuelenv = tau2 / (tau2 + dt) * zl_1 + dt / (tau2 + dt) * valuelfiltrect;
zl_1 = valuelenv;
float value2env = tau2 / (tau2 + dt) * z2_1 + dt / (tau2 + dt) * value2filtrect;
z2_1 = value2env;
```

Figura 55. Filtro paso bajo, función processEMG().

Ajuste de media móvil.

Finalmente se aplicará un ajuste de media móvil (**Figura 56**). Para ello, en los vectores "buffer1" y "buffer2" se van acumulando los valores de las señales procedentes del filtro paso bajo. Previamente dichos valores se van sumando en los sumatorios "sum1" y "sum2", hasta que los buffers llegan a su longitud límite, que en ambos casos es 20. Para calcular la media basta con dividir cada sumatorio entre 20.

Las variables "muscleActivityP" y "muscleActivityI" guardarán estas medias de valores, cuyo muestreo nos detallará la forma de la señal filtrada.

```
  // Moving average (compute signal RMS value)
  suml = suml - buffer1[il];
  suml = suml + valuelenv;
  buffer1[il] = valuelenv;
  il++;
  if (il >= buffLength)
  {
    il = 0;
  }
  muscleActivityI = suml / buffLength;
  sum2 = sum2 - buffer2[i2];
  sum2 = sum2 + value2env;
  buffer2[i2] = value2env;
  i2++;
  if (i2 >= buffLength)
  {
    i2 = 0;
  }
  muscleActivityP = sum2 / buffLength;
}
```

Figura 56. Ajuste de media móvil, función processEMG().

3.6.3.2 MOVIMIENTOS

Se comienza con dos lecturas, la primera de ellas de los sensores, a través de la llamada a la función "processEMG()". Para visualizar los valores de los sensores gráficamente se utilizará el Serial Plotter (**Figura 57**). Con el comando "Serial.print()" se escriben los valores a representar, en este caso, los valores de los sensores ya procesados (se establecen dos líneas de referencia en 90 y -10 para visualizar los valores a la escala aproximada de la señal procesada).

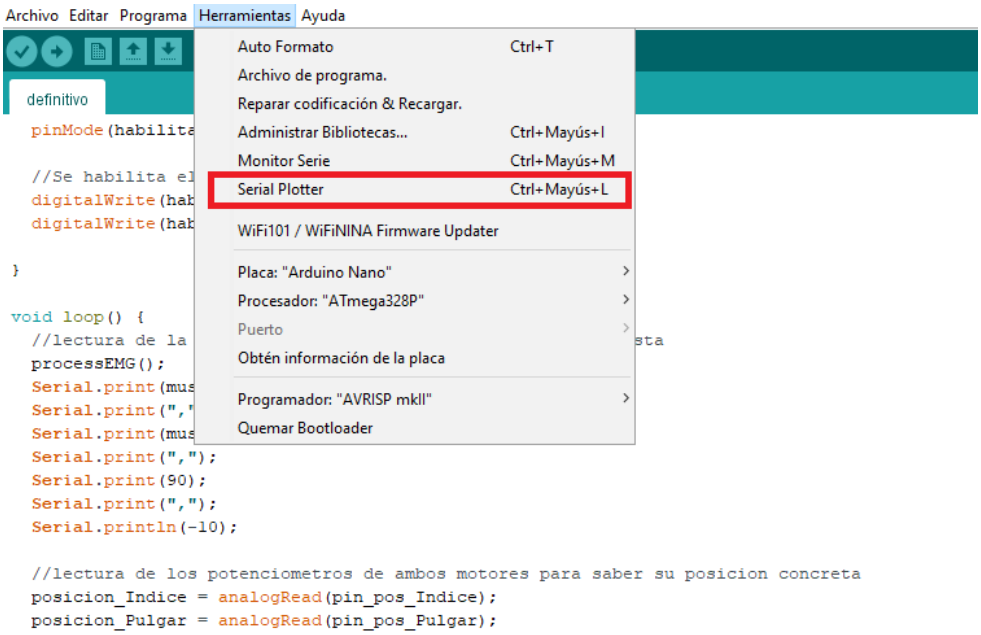


Figura 57. Serial Plotter en el entorno de Arduino.

Por otro lado, la segunda lectura será del valor de los potenciómetros de realimentación de posición, para ello, se utiliza la función “analogRead(pin)”, cuyo valor devuelto se igualará a las variables “posicion_Indice” y “posicion_Pulgar” (**Figura 58**).

```
void loop() {
  //lectura de la señal de los sensores y procesamiento de esta
  processEMG();
  Serial.print(muscleActivityI);
  Serial.print(",");
  Serial.print(muscleActivityP);
  Serial.print(",");
  Serial.print(90);
  Serial.print(",");
  Serial.println(-10);

  //lectura de los potenciometros de ambos motores para saber su posicion concreta
  posicion_Indice = analogRead(pin_pos_Indice);
  posicion_Pulgar = analogRead(pin_pos_Pulgar);
```

Figura 58. Fragmento del código donde se realizan las lecturas de los sensores y los potenciómetros de realimentación.

Los valores recogidos en dichas lecturas se emplearán a continuación para elegir el movimiento. A través de la sentencia condicional “if(condiciones)” para diferenciar los valores de los sensores procesados diferenciamos 3 movimientos, tal y como se indica en la **figura 59**. Por orden, los “if” describen los movimientos del dedo índice únicamente, dedo pulgar únicamente y ambos dedos a modo de pinza.

```
if (muscleActivityI > 60 && muscleActivityP < 30)
 {
   ⋮
 }

if (muscleActivityP > 55 && muscleActivityI < 30)
 {
   ⋮
 }

if (muscleActivityI > 30 && muscleActivityP > 30)
 {
   ⋮
 }
```

Figura 59. Fragmento del código donde se muestran los 3 movimientos principales en función de los valores de la señal procesada de los sensores (los tres puntos del interior de los corchetes hacen referencia a que dentro existe código).

El contenido de los tres if es bastante similar, ya que el modo de operación en la contracción y extensión del motor se basa en los valores de la señal

procesada del sensor y los valores máximo y mínimo de la carrera del motor. Se comenzará describiendo el movimiento individual del dedo índice y a continuación se darán unas pinceladas sobre el movimiento individual del dedo pulgar, dejando para el final el análisis de la pinza.

MOVIMIENTO DEL DEDO ÍNDICE

Se entrará en su respectivo if cuando se produzca un pico mayor a 60 en el sensor del dedo índice, mientras que el sensor del dedo pulgar deberá encontrarse en un valor inferior a 30, tal y como describe la **figura 59**.

Seguidamente a la entrada al if, se utiliza un bucle "while(condición de mantenimiento en el bucle)", **figura 60**. En la ejecución, el programa se mantendrá dentro del bucle hasta que la condición indicada por el while sea falsa.

En este caso, se realizará una lectura y procesamiento de la señal de los sensores, seguida de una escritura en la gráfica "Serial Plotter" y una nueva lectura, pero en esta ocasión del valor del potenciómetro de posición del motor y por último, se contraerá el índice, estableciendo una velocidad de 150 en la función contraerIndice(). Todos estos pasos se realizarán constantemente hasta que la posición del dedo índice alcance su contracción máxima o, por otro lado, cuando se produzca un pico en la señal del sensor "muscleActivityP" superior a 40, ya que el sensor recogerá las variaciones musculares del músculo extensor de los dedos (es decir se quiere realizar una extensión del dedo).

```
while(posicion_Indice > contraMAX_Indice && muscleActivityP < 40)
{
  processEMG();
  Serial.print(muscleActivityI);
  Serial.print(",");
  Serial.print(muscleActivityP);
  Serial.print(",");
  Serial.print(90);
  Serial.print(",");
  Serial.println(-10);

  posicion_Indice = analogRead(pin_pos_Indice);
  //Serial.print("Me contraigo indice: ");
  //Serial.println(posicion_Indice);
  contraerIndice(150);
}
```

Figura 60. Fragmento del código donde se realiza la contracción del dedo índice.

Al salir del bucle anterior, se puede observar otro bucle while, como el que indica la **figura 61**. Su condición de salida está relacionada con la extensión del dedo índice. Por ello, este bucle será útil cuando se llega a la contracción máxima y se quiere seguir manteniendo dicha posición hasta que se realice la

extensión del dedo. Se hará una llamada a la función “parar()” para conseguir esta detención en el motor.

```
while(muscleActivityP < 40)
{
  processEMG();
  Serial.print(muscleActivityI);
  Serial.print(",");
  Serial.print(muscleActivityP);
  Serial.print(",");
  Serial.print(90);
  Serial.print(",");
  Serial.println(-10);

  parar(control_A_Indice, control_B_Indice);
  //Serial.print("Me paro indice: ");
  //Serial.println(posicion_Indice);
}
```

Figura 61. Fragmento de código donde se para el motor después de la contracción máxima del dedo índice.

Finalmente se realizará una extensión del dedo índice a través del bucle while de la **figura 62**, llegando así de nuevo a la posición de descanso de la mano. Para ello, de manera similar a la contracción, se realiza una lectura de las señales ya procesadas de los sensores, y de la posición del motor, para posteriormente realizar la extensión del dedo a través de la llamada a la función “extenderIndice()”. Se abandonará el bucle al llegar a la extensión máxima.

En la conclusión del código de la **figura 62** se realiza una parada del motor del dedo índice, la cual es recomendada por el fabricante para evitar futuros problemas en el funcionamiento del motor.

```
while(posicion_Indice < extenMAX_Indice)
{
  processEMG();
  Serial.print(muscleActivityI);
  Serial.print(",");
  Serial.print(muscleActivityP);
  Serial.print(",");
  Serial.print(90);
  Serial.print(",");
  Serial.println(-10);

  posicion_Indice = analogRead(pin_pos_Indice);
  //Serial.print("Me extiendo indice: ");
  //Serial.println(posicion_Indice);
  extenderIndice(150);
}

parar(control_A_Indice, control_B_Indice);
```

Figura 62. Fragmento de código en el que se describe la extensión del dedo índice.

MOVIMIENTO DEL DEDO PULGAR

La programación del movimiento del dedo pulgar es muy similar a la del dedo índice, por lo que se hará un análisis de las diferencias existentes en el código, obviando los pasos que se encuentran indistintamente en ambos códigos.

Inicialmente la entrada al if correspondiente al movimiento de dicho dedo se produce al encontrar un pico mayor a 55 en el sensor del dedo pulgar, mientras que el sensor del dedo índice deberá encontrarse en un valor inferior a 30, tal y como describe la **figura 59**.

En el primer bucle while donde se produce la contracción, tenemos dos condiciones de mantenimiento, **figura 63**. Una de ellas el no llegar a la contracción máxima establecida y la otra, que no se produzca una extensión. La extensión a diferencia del caso del dedo índice (donde se analizaba la señal de un músculo para la contracción y otro para la extensión), se utiliza la señal del mismo músculo que desencadena principalmente el movimiento del dedo en la contracción, ya que se analiza la actividad muscular del abductor largo del pulgar, recogida por el sensor "muscleActivityP". Concretamente, cuando este sensor detecta una señal inferior a 2, es decir el músculo se deja de contraer, se sale del bucle.

También encontramos una diferencia en la llamada a la función "contraerPulgar()" ya que se pasa como argumento una velocidad de 200 en vez de 150, para poder así apreciar la diferencia de velocidad en la contracción de ambos dedos.

```
while(posicion_Pulgar < contraMAX_Pulgar && muscleActivityP > 2)
{
  processEMG();
  Serial.print(muscleActivityI);
  Serial.print(",");
  Serial.print(muscleActivityP);
  Serial.print(",");
  Serial.print(90);
  Serial.print(",");
  Serial.println(-10);

  posicion_Pulgar = analogRead(pin_pos_Pulgar);
  //Serial.print("Me contraigo pulgar: ");
  //Serial.println(posicion_Pulgar);
  contraerPulgar(200);
}
```

Figura 63. Fragmento del código donde se produce la contracción del dedo pulgar (abducción).

Como en la descripción del movimiento del dedo índice, también es necesario otro bucle while que detenga el motor una vez se ha llegado a la contracción máxima del pulgar mientras no se inicie la extensión del dedo. Como condición de mantenimiento del bucle tenemos una de las utilizadas en el bucle anterior, el cese de actividad en el sensor que recoge la actividad del abductor largo del pulgar (**Figura 64**).

```
while(muscleActivityP > 2)
{
  processEMG();
  Serial.print(muscleActivityI);
  Serial.print(",");
  Serial.print(muscleActivityP);
  Serial.print(",");
  Serial.print(90);
  Serial.print(",");
  Serial.println(-10);

  parar(control_A_Pulgar, control_B_Pulgar);
  //Serial.print("Me paro pulgar: ");
  //Serial.println(posicion_Pulgar);
}
```

Figura 64. Fragmento del código donde se produce la parada del motor después de la contracción máxima del dedo pulgar.

Por último, al igual que en el dedo índice se llevará el dedo pulgar a una posición de reposo, donde se encuentra totalmente aducido. Para ello se utilizará como novedad la función "extenderPulgar()". Se realiza una parada del motor para evitar futuros fallos (**Figura 65**).

```
while(posicion_Pulgar > extenMAX_Pulgar)
{
  processEMG();
  Serial.print(muscleActivityI);
  Serial.print(",");
  Serial.print(muscleActivityP);
  Serial.print(",");
  Serial.print(90);
  Serial.print(",");
  Serial.println(-10);

  posicion_Pulgar = analogRead(pin_pos_Pulgar);
  //Serial.print("Me extiendo pulgar: ");
  //Serial.println(posicion_Pulgar);
  extenderPulgar(150);
}

parar(control_A_Pulgar, control_B_Pulgar);
```

Figura 65. Fragmento del código donde se produce la extensión del pulgar (aducción).

MOVIMIENTO DE PINZA

El inicio del movimiento de pinza lo desencadena un valor superior a 30 en ambos sensores "muscleActivityI" y "muscleActivityP", tal y como indica la **figura 59**. En este caso los movimientos de ambos dedos deben estar coordinados. Por ello, tanto en las condiciones de mantenimiento en los bucles como en las lecturas de potenciómetros, intervendrán los sensores y motores asociados a cada dedo.

En el primer bucle while realizaremos la contracción del dedo índice y la abducción del dedo pulgar, de manera que cuando ambos motores hayan llegado a su posición final se haya agarrado el objeto oportuno haciendo una forma de pinza con los dedos. Inicialmente se realizará una lectura de los sensores y potenciómetros de ambos motores, ya que como en casos anteriores, sus valores servirán para salir del bucle. Seguidamente se llama a las funciones "contraerIndice()" y "contraerPulgar()". Habrá una salida del bucle al llegar en ambos dedos a la posición de máxima contracción, **figura 66**.

```
while(posicion_Indice > contraMAX_Indice || posicion_Pulgar < contraMAX_Pulgar)
{
  processEMG();
  Serial.print(muscleActivityI);
  Serial.print(",");
  Serial.print(muscleActivityP);
  Serial.print(",");
  Serial.print(90);
  Serial.print(",");
  Serial.println(-10);

  posicion_Indice = analogRead(pin_pos_Indice);
  posicion_Pulgar = analogRead(pin_pos_Pulgar);
  //Serial.print("Me contraigo: ");
  //Serial.print(posicion_Indice);
  //Serial.print("//");
  //Serial.println(posicion_Pulgar);
  contraerIndice(150);
  contraerPulgar(200);
}
```

Figura 66. Fragmento del código donde se realiza el movimiento de pinza.

En el siguiente bucle while (**figura 67**) la mano se mantiene en la posición de pinza mientras no se produzca un pico superior a 40 en el valor del sensor "muscleActivityP". Esto pico será provocado por un movimiento en los músculos extensores de los dedos de la mano, el cual es recogido por el sensor anteriormente mencionado. Llamando a la función "parar()" dos veces consecutivas, pero cada una con sus respectivas entradas, conseguiremos detener ambos motores. De esta forma, el usuario se mantendrá en esa posición el tiempo demandado para sujetar el objeto en cuestión.

```
while(muscleActivityP < 40)
{
  processEMG();
  Serial.print(muscleActivityI);
  Serial.print(",");
  Serial.print(muscleActivityP);
  Serial.print(",");
  Serial.print(90);
  Serial.print(",");
  Serial.println(-10);

  parar(control_A_Indice, control_B_Indice);
  parar(control_A_Pulgar, control_B_Pulgar);
  //Serial.print("Me paro: ");
  //Serial.print(posicion_Indice);
  //Serial.print("//");
  //Serial.println(posicion_Pulgar);
}
```

Figura 67. Fragmento del código donde se programada el mantenimiento en posición de pinza.

Para terminar con la programación, se utilizará el último bucle while (**figura 68**) para extender el dedo índice y realizar una aducción del dedo pulgar. De esta manera, se llegará a una posición neutra de la mano donde posteriormente se puede repetir el mismo movimiento de pinza u otro. Para llegar a cabo este proceso se realizará una nueva lectura de los sensores y los potenciómetros. Seguidamente se hará una llamada a las funciones “extenderIndice()” y “extenderPulgar()” para extender ambos dedos. El programa se encontrará realizando estos pasos hasta que ambos dedos lleguen a su extensión máxima.

Llegados a este punto, se finalizará parando ambos motores a través dos llamadas consecutivas a la función “parar()”.

```
while(posicion_Indice < extenMAX_Indice || posicion_Pulgar > extenMAX_Pulgar)
{
  processEMG();
  Serial.print(muscleActivityI);
  Serial.print(",");
  Serial.print(muscleActivityP);
  Serial.print(",");
  Serial.print(90);
  Serial.print(",");
  Serial.println(-10);

  posicion_Indice = analogRead(pin_pos_Indice);
  posicion_Pulgar = analogRead(pin_pos_Pulgar);
  //Serial.print("Me extiendo: ");
  //Serial.print(posicion_Indice);
  //Serial.print("//");
  //Serial.println(posicion_Pulgar);
  extenderIndice(150);
  extenderPulgar(150);
}

parar(control_A_Indice, control_B_Indice);
parar(control_A_Pulgar, control_B_Pulgar);
```

Figura 68. Fragmento del código donde se programa la vuelta a la posición neutra de la mano después del movimiento de pinza.

3.7 Resultados.

En este apartado se demostrará visualmente el funcionamiento de la mano protésica programado en el apartado anterior. Para ello, se han muestreado las señales de los sensores mioeléctricos y posteriormente se han representado en una gráfica a lo largo del tiempo.

Una vez conectados los sensores y colocados en los puntos específicos del antebrazo, se comienza con la realización de los tres movimientos programados. Estos deberán iniciarse y detenerse según lo mostrado en el código. En las gráficas que se mostrarán a continuación, se requiere prestar especial atención a las líneas azul y roja. La primera de ellas recoge la señal del sensor ubicado en el flexor de los dedos, mientras que la segunda se encuentra recogiendo la señal de los músculos extensor de los dedos y el abductor largo del pulgar. Las líneas horizontales ubicadas a los 90 y los -10, simplemente se han utilizado para que la escala de la gráfica se mantenga en valores visibles en el caso de no haber señal.

DEDO ÍNDICE

Se puede observar en la **figura 69** que cuando la señal azul supera el umbral de los 60 y la roja se encuentra por debajo de los 30, el dedo índice se contrae ya que se ha producido movimiento en el flexor de los dedos, tal y como se ve en la **figura 70** (derecha).

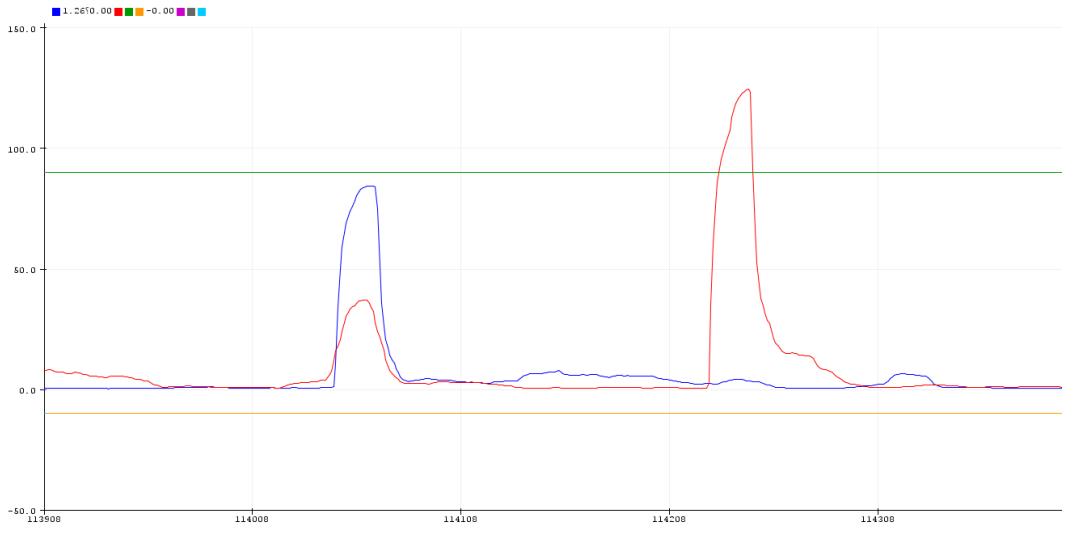


Figura 69. Muestreo de la señal de los sensores para el movimiento del dedo índice.

Seguidamente en la gráfica podemos ver un pico superior a 40, lo que nos indica que se ha detectado señal en el extensor de los dedos, por lo que el dedo índice se extiende (**figura 70** izquierda).

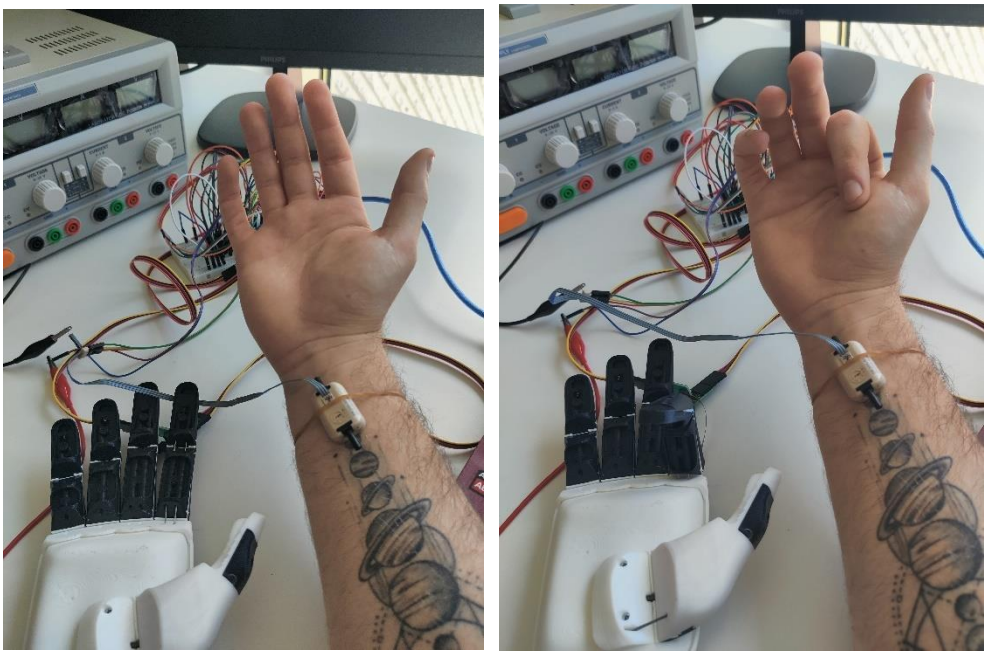


Figura 70. Izquierda, contracción del índice. Derecha, extensión del índice.

DEDO PULGAR

En este caso solo habrá pico en la señal roja, pues se recoge el movimiento del abductor del dedo pulgar, **figura 71**. Cuando dicha señal supere el umbral de los 55 y la azul se encuentre por debajo de los 30, se producirá la abducción del pulgar (**figura 72** izquierda).

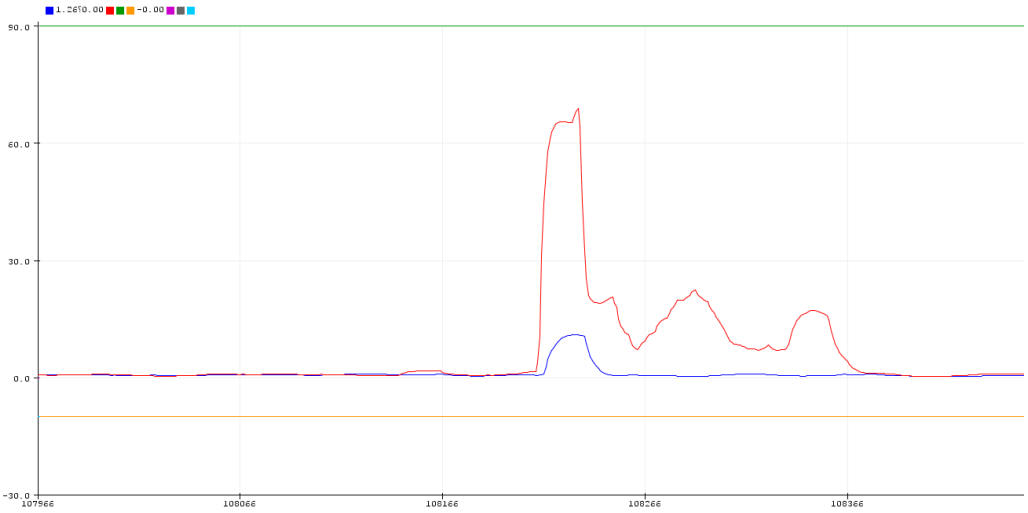


Figura 71. Muestreo de la señal de los sensores para el movimiento del dedo pulgar.

A diferencia del caso anterior, nos fijaremos en la propia señal roja para detener la abducción. Cuando se deje de detectar actividad en el músculo abductor la

señal caerá por debajo de 2, por lo que se producirá la aducción, **figura 72** derecha.

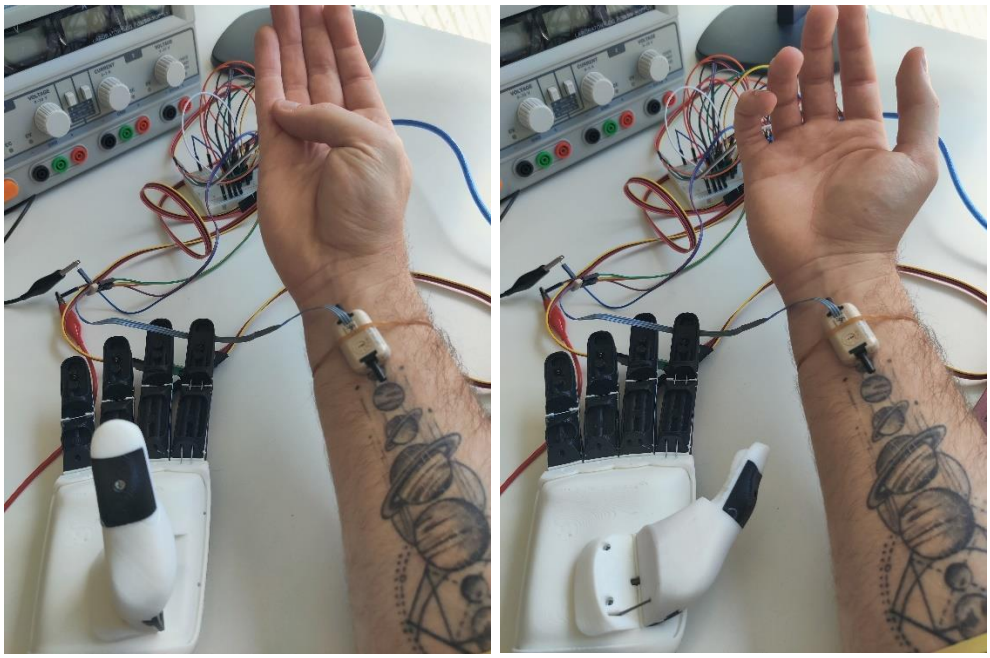


Figura 72. Izquierda, abducción del pulgar. Derecha, aducción del pulgar.

PINZA

Este último caso es fácil de deducir habiendo tratado los dos anteriores. El movimiento de pinza se realizará cuando se encuentre actividad en los dos sensores a la vez, es decir, si en ambas señales se presentan valores superiores a 30.

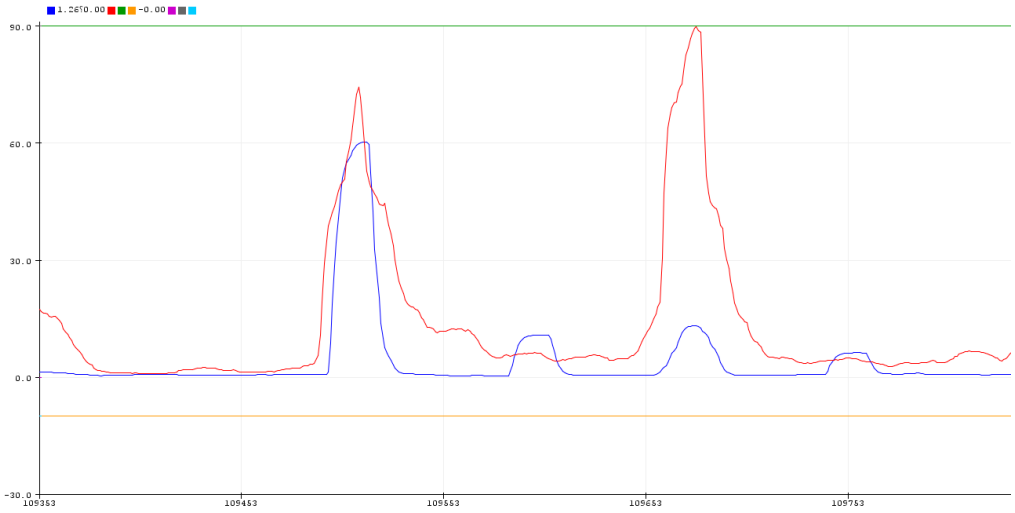


Figura 73. Muestreo de la señal de los sensores para el movimiento de pinza.

Esto se produce en el primer pico doble ubicado en la **figura 73** y tiene como consecuencia el movimiento de ambos dedos a la posición de pinza, tal como muestra la **figura 74** izquierda.

Por último, para volver a la posición neutra de la mano se buscará un pico en la actividad del músculo extensor de los dedos, que desemboca en un pico superior a 40 en la señal roja. Este se puede apreciar en la **figura 73** y su resultado en la **figura 74** derecha.

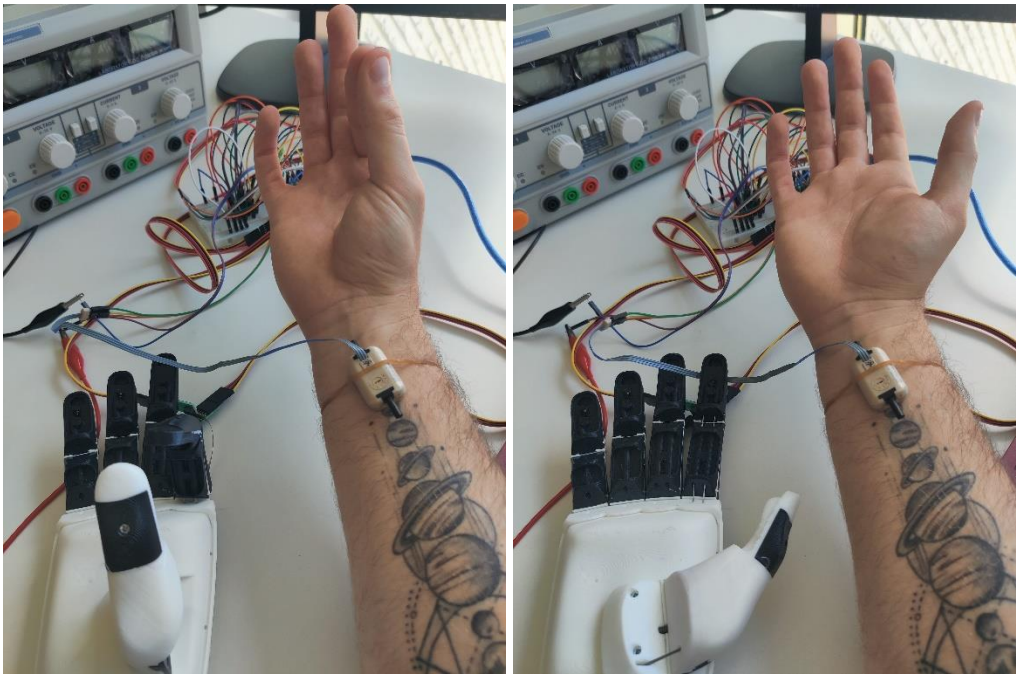


Figura 74. Izquierda, posición de pinza. Derecha, posición neutra.

4 CONCLUSIONES

El objetivo principal de este trabajo es la implementación de una prótesis para un usuario con desarticulación de mano, de manera que su posible venta sea a un precio reducido en comparación a los modelos actuales del mercado.

Como en este caso se ha realizado la construcción por piezas de la mano protésica al completo, pero solo se han motorizado y programado dos dedos de cinco, el coste será evidentemente menor que el de una prótesis completa. Aun así, extrapolando los costes se puede aproximar el precio final.

En la siguiente tabla se reflejarán los costes de la prótesis.

	REQUERIMIENTOS	UNIDADES	€/UNIDAD	PRECIO (€)
ELEMENTOS ELECTRÓNICOS	Cables de conexión	37	0,05825	2,15525
	Arduino nano	1	9,66	9,66
	L293D	2	1,499	2,998
	Sensor mioeléctrico	2	49,99	99,98
	Protoboard	1	2,95	2,95
	Motor PQ12-P	4	55	220
	Adaptador de cable PQ12-P	4	6,11	24,44
PIEZAS MONTAJE	Resorte cónico	1	14,5	14,5
	Resorte de tracción	9	6,25	56,25
	Pasador din-6325 3x10mm	11	0,056	0,616
	Insertos M2	22	0,0595	1,309
	Tornillo din-965 M2x5mm	2	0,015	0,03
	Tornillo din-965 M2x8mm	2	0,0139	0,0278
	Tornillo din-965 M2x10mm	15	0,0145	0,2175
	Tornillo din-965 M2x16mm	4	0,016	0,064
	Tornillo din-921 M2x5mm	5	1,0786	5,393
	Tornillo din-912 M1.4x5mm	14	0,8048	11,2672
		METROS	€/METRO	PRECIO (€)
	Nylon de pesca	1,8	1,5	2,7
IMPRESIÓN Y PROGRAMACIÓN		Kg	€/Kg	PRECIO (€)
	Impresión 3D	0,188	20	3,76
		HORAS	€/HORA	PRECIO (€)
	Consumo máquinaria	49,37	1	49,37
	Coste operario	40	15	600
			TOTAL (€):	**1107,68775**

Si comparamos el coste resultante con los precios del mercado, la prótesis cumple con los principales requerimientos de un usuario con desarticulación

de muñeca a un coste inferior al resto del mercado. Todo ello, sin sacrificar funcionalidad y utilizando materiales reciclables y biodegradables.

Finalmente se debe destacar que este trabajo tiene un objetivo meramente académico, con ello se quiere aclarar que puede ser base de futuros desarrollos, pero no está preparado para ser comercializado. Para ello, sería necesario incluir mejoras en relación con el control de la mano y el procesado de la señal de los sensores.

Como futuras líneas de trabajo se propone:

- Montaje del resto de los dedos: comprando otros dos motores PQ12-P se pueden mover los dedos restantes, pues los dedos anular y meñique se mueven conjuntamente.
- Programación del resto de los dedos: programando los dedos restantes de la mano protésica se podría conseguir una mano completa funcional (a excepción de los movimientos de la muñeca).
- Utilización de una Raspberry pi como controlador: en lugar de Arduino se podría utilizar otro controlador que permita la programación y ejecución en tiempo real, ya que la comunicación entre Arduino y los dispositivos electrónicos tiene un pequeño retraso. Utilizando la Raspberry pi esto podría solucionarse.
- Utilización de otro controlador que incluya comunicación Wifi o Bluetooth: a través del uso del Wifi y el Bluetooth se podría manejar la prótesis con un smartphone, a través del desarrollo de una aplicación que lo permita o una página web.
- Definición de distintos mecanismos de agarre: en este trabajo se ha programado la pinza, pero se pueden incluir infinidad de agarres teniendo todos los dedos operativos. Se puede hacer un estudio de los más comunes e incluso personalizarlos para el usuario.
- Inclusión de sensores de presión en las yemas de los dedos protésicos: añadiendo este tipo de sensores se podrá regular la fuerza con la que se agarra un objeto.
- Análisis del mejor modo de alimentación para la prótesis: en este proyecto se ha utilizado como alimentación de la placa Arduino un ordenador portátil y para los motores una fuente de alimentación. Pero, se puede incluir una batería o una alimentación por pilas, de forma que se utilice una única fuente para todos los dispositivos.
- Combinar la mano protésica con otra prótesis: para poder tener una mano o brazo protésico completamente funcional, habría que combinar la prótesis con otra que controle los movimientos de la muñeca y/o los movimientos del codo.

5 BIBLIOGRAFÍA

[1] EsMachina, «Prótesis de mano en la actualidad,» ESMACHINA, 27 Enero 2020. [En línea]. Available: https://www.esmachina.com/que-es-protesis-de-mano-tipos-y-precios/. [Último acceso: 9 Septiembre 2021].

[2] Ottobock, «Nivel de amputación,» 2014. [En línea]. Available: https://www.ottobock.es/protesica/informacion-para-amputados/de-la-amputacion-a-la-rehabilitacion/altura-de-la-amputacion/. [Último acceso: 2021 Septiembre 9].

[3] Mediprax, «Tipos de prótesis para miembro superior,» [En línea]. Available: https://mediprax.mx/tipos-de-protesis-para-miembro-superior/. [Último acceso: 2021 Octubre 11].

[4] O. Albufera, «Productos, prótesis,» 2018. [En línea]. Available: https://www.ortopediaalbufera.com/. [Último acceso: 11 Octubre 2021].

[5] MedicalExpo, «Prótesis de mano con control de motor,» 2021. [En línea]. Available: https://www.medicalexpo.es/prod/rslsteeper/product-74956-458252.html. [Último acceso: 11 Octubre 2021].

[6] G. D. I. E. R. A. Y. BIOMECÁNICA, «Diseño, desarrollo e implementación de prótesis mioeléctricas personalizadas de mano con retroalimentación háptica empleando tecnologías de fabricación digital en filamentos de plástico PET reciclado de bajo costo,» 31 Diciembre 2020. [En línea]. Available: https://investigacion.pucp.edu.pe/grupos/girab/proyecto/protesis-mioelectricas-personalizadas-de-mano-con-retroalimentacion-haptica-empleando-fabricacion-digital-en-filamentos-plastico-pet/. [Último acceso: 2021 Octubre 11].

[7] J. L. Q. M. X. C. D. &. C. J. I. Brito, «Estudio del estado del arte de las prótesis de mano,» 2013. [En línea]. Available: https://dspace.ups.edu.ec/handle/123456789/8447. [Último acceso: 12 03 2022].

[8] J. ARENAS MOLES, «Diseño funcional de una prótesis de mano,» 2020. [En línea]. Available: https://upcommons.upc.edu/handle/2117/331167. [Último acceso: 12 Marzo 2022].

[9] O. P. Garo, «Quantum i-limb,» 2018. [En línea]. Available: https://garotecnica.com/protesica/miembro-superior/manos/quantum-i-limb/. [Último acceso: 13 03 2022].

[10] P. SÁNCHEZ CARRATALÁ, «Construcción y evaluación de una prótesis mioeléctrica de mano de bajo coste,» 2020. [En línea]. Available: http://rua.ua.es/dspace/handle/10045/108353. [Último acceso: 13 3 2022].

[11] Mand.ro, «Mand.ro - A Low-cost, Light-weight 3D Printed Prosthetic Hand,» 2020. [En línea]. Available: http://mand.ro/#feature. [Último acceso: 13 3 2022].

[12] OpenBionics, «Hero Arm,» 2021. [En línea]. Available: https://openbionics.com/hero-arm/. [Último acceso: 13 3 2022].

[13] Angels, «Los huesos de la mano,» 16 Enero 2017. [En línea]. Available: https://www.mundodeportivo.com/uncomo/educacion/articulo/como-se-llaman-los-huesos-de-la-mano-40009.html. [Último acceso: 10 Septiembre 2021].

[14] VIVES, «AFECCIONES DEL CODO (PARTE 1),» [En línea]. Available: http://clinicavives.com/2016/10/18/afecciones-del-codo-parte-1/. [Último acceso: 10 Septiembre 2021].

[15] Ensure, «¿Sabes cuál es el rol de los músculos en nuestro cuerpo?,» [En línea]. Available: https://www.ensure.abbott/mx/blog/sabes-cual-es-el-rol-de-los-musculos-en-nuestro-cuerpo-.html. [Último acceso: 10 Septiembre 2021].

[16] FISIODELASERNA, «¿Qué es una contracción muscular?,» 23 Septiembre 2018. [En línea]. Available: https://www.fisioterapiadelaserna.com/fisioterapia-de-la-serna-que-es-una-contraccion-muscular/. [Último acceso: 11 Septiembre 2021].

[17] EnfermeríaTop, «Anatomía-Miembro Superior- Mano,» [En línea]. Available: https://enfermeria.top/apuntes/anatomia/miembro-superior/mano/. [Último acceso: 12 Septiembre 2021].

[18] FisioStar, «Articulaciones de la mano,» [En línea]. Available: https://fisiostar.com/anatomia/articulaciones-de-la-mano. [Último acceso: 12 Septiembre 2021].

[19] C. Serrano, «Extremidad superior (Anatomía),» KENHUB, 2021. [En línea]. Available: https://www.kenhub.com/es/library/anatomia-es/anatomia-de-la-extremidad-superior. [Último acceso: 12 Septiembre 2021].

[20] Arduino.cc, «Arduino Nano,» [En línea]. Available: https://store.arduino.cc/products/arduino-nano. [Último acceso: 23 Septiembre 2020].

[21] Actuonix, «PQ12-P Linear Actuator with Feedback,» [En línea]. Available: https://www.actuonix.com/Actuonix-PQ-12-P-Linear-Actuator-p/pq12-p.htm. [Último acceso: 23 Septiembre 2021].

[22] W. M. d. s. d. transmisión, «Relacion de engranes,» [En línea]. Available: https://mantenimiento-de-sistemas-de-transmision.fandom.com/es/wiki/Relacion_de_engranes. [Último acceso: 24 Septiembre 2021].

[23] F. A. Team, «DUTY CYCLE en un actuador lineal,» 23 Enero 2020. [En línea]. Available: https://www.firgelliauto.com/es/blogs/news/what-is-a-duty-cycle-in-a-linear-actuator. [Último acceso: 24 Septiembre 2021].

[24] L. Union, «Explicación PUENTE H - Funcionamiento - Inversión de giro con transistores.,» [En línea]. Available: https://www.youtube.com/watch?v=CdIQL2LQyWA. [Último acceso: 25 Septiembre 2021].

[25] UNITELECTRONICS, «BC547B Transistor BJT NPN TO-92 45V,» [En línea]. Available: https://uelectronics.com/producto/bc547b-transistor-bjt-npn-to-92-45v/. [Último acceso: 25 Septiembre 2021].

[26] RobotsDidácticos, «Manejo de potencia para motores con el integrado L293D,» 5 Noviembre 2019. [En línea]. Available: http://robots-argentina.com.ar/didactica/manejo-de-potencia-para-motores-con-el-integrado-l293d/. [Último acceso: 25 Septiembre 2021].

[27] Ottobock., «13E200 MyoBock electrode,» [En línea]. Available: https://professionals.ottobock.com.au/Products/Prosthetics/Prosthetics-Upper-Limb/Adult-Terminal-Devices/13E200-MyoBock-electrode/p/13E200. [Último acceso: 2 Noviembre 2021].

[28] o. bionics, «BRUNEL HAND 2.0 Assembly Guide,» [En línea]. Available: https://usermanual.wiki/Document/BrunelHand20AssemblyGuide.1170707177. [Último acceso: 15 Diciembre 2021].

[29] Peysanet, «PLACA DE PRUEBAS PROTOBOARD BOARD DE ALTA CALIDAD ELECTRONICA -> 400 CONTACTOS,» [En línea]. Available: https://www.peysanet.com/articulo-25452-PLACA%20DE%20PRUEBAS%20PROTOBOARD%20BOARD%20DE%20ALTA%20CALIDAD%20ELECTRONICA%20%20400%20CONTACTOS. [Último acceso: 18 Diciembre 2021].

[30] I+DElectrónica, «Adaptador de cable para actuadores PQ12,» [En línea]. Available: https://didacticaselectronicas.com/index.php/elementos-electromecanicos/motores-y-solenoides-1/actuadores-lineales/realimentados/adaptador-de-cable-para-actuadores-pq12-cable-adapter-lineales-servoactuadores-actuador-lineal-pistones-micro-servo-pq12-actuo. [Último acceso: 18 Diciembre 2021].

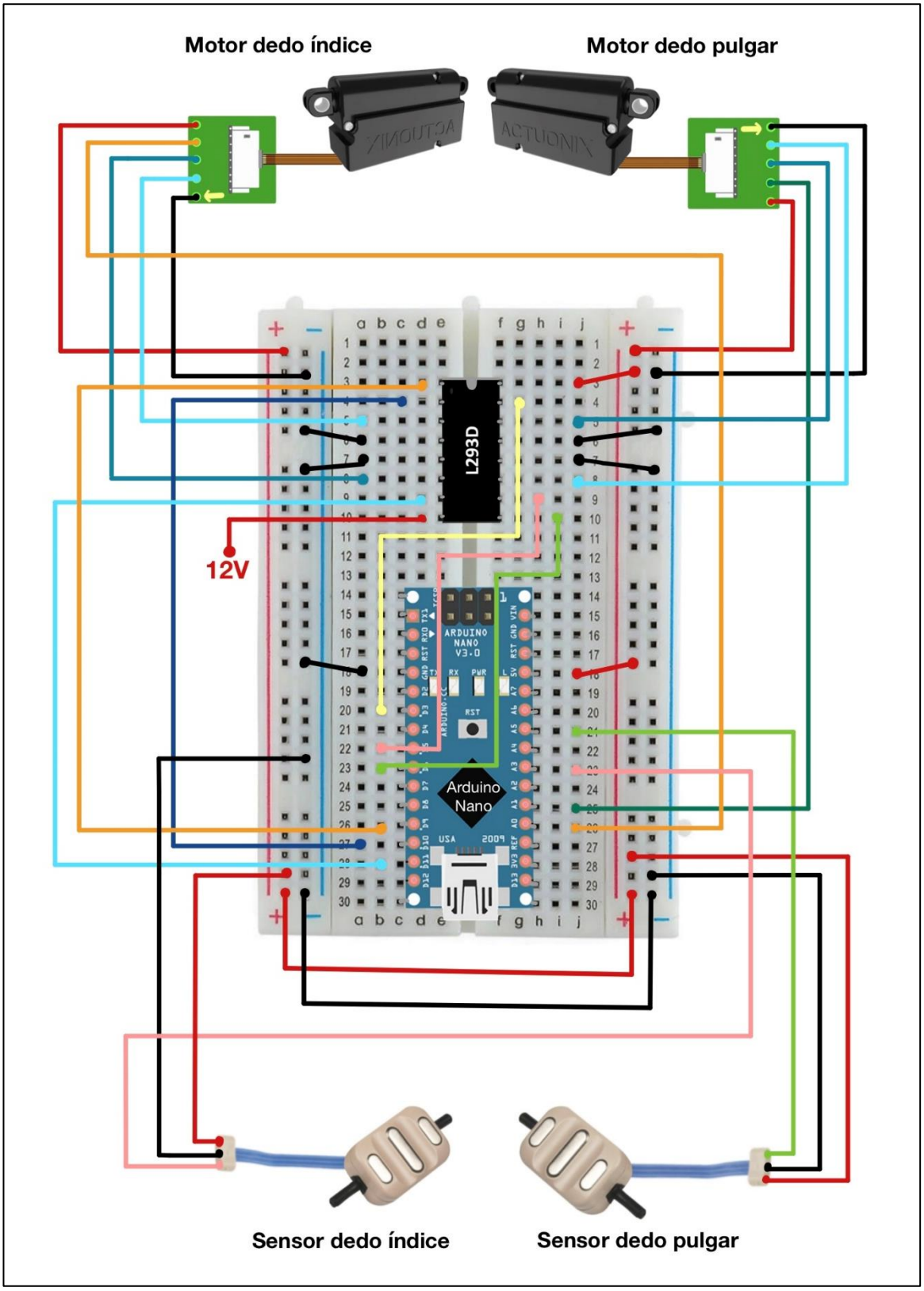
[31] V. O. H. H. Luis Hernán Cubillos, «Project Hub,» 11 Noviembre 2018. [En línea]. Available: https://create.arduino.cc/projecthub/high-five/protesis-mioelectrica-de-mano-1159a1. [Último acceso: 25 03 2022].

[32] Solectro, «Todo lo que necesitas saber sobre filtros RC,» 25 Septiembre 2020. [En línea]. Available: https://solectroshop.com/es/blog/todo-lo-que-necesitas-saber-sobre-filtros-rc-n52. [Último acceso: 22 Marzo 2022].

[33] L. Llamas, «Luis Llamas,» 24 Marzo 2017. [En línea]. Available: https://www.luisllamas.es/arduino-paso-bajo-exponencial/. [Último acceso: 22 Marzo 2022].

6 ANEXOS

6.1 Esquema de conexiones



6.2 Código de programación.

```
/*****************************************************************
 ** Movimiento de ambos dedos a traves de los valores del sensor *
 *****************************************************************/

/************
 *** Pines ***
 ************/

//Pines de los sensores
const int sensorIndice = A3;
const int sensorPulgar = A5;

//Pines del motor del dedo indice
const int habilita_Indice = 9; // pin de salida digital para
habilitar el puente H
const int control_A_Indice = 10; // pin A de salida para control
del sentido del motor del dedo indice
const int control_B_Indice = 11; // pin B de salida para control
del sentido del motor del dedo indice
const int pin_pos_Indice = A0; // pin Analogico de entada que
indica la posicion del motor

//Pines del motor del dedo pulgar
const int habilita_Pulgar = 6; // pin de salida digital para
habilitar el puente H
const int control_A_Pulgar = 5; // pin A de salida para control
del sentido del motor
const int control_B_Pulgar = 3; // pin B de salida para control
del sentido del motor
const int pin_pos_Pulgar = A1; // pin Analogico de entada que
indica la posicion del motor


/*****************
 *** Variables ***
 *****************/

// Valores de la señal sin procesar
int valueMuscleI;
int valueMuscleP;

// Valores de la señal procesada
float muscleActivityI = 0;
float muscleActivityP = 0;

// Tiempo de ejecucion (ms)
long t = 0;

// Variables y parametros filtro paso alto
float fc1 = 100; // Frecuencia de corte
float tau1 = 1 / (2 * 3.1415 * fc1);
float y1_1 = 0;
float x1_1 = 0;
float y2_1 = 0;
float x2_1 = 0;

// Variables y parametros filtro paso bajo
```

```
float fc2 = 5; // Frecuencia de corte
float tau2 = 1 / (2 * 3.1415 * fc2);
float z1_1 = 0;
float z2_1 = 0;

// Buffer variables for computing moving average
int buffLength = 20; // aprox. 20*2 ms
int i1 = 0;
float buffer1 [20] = {0.0};
float sum1 = 0;
int i2 = 0;
float buffer2 [20] = {0.0};
float sum2 = 0;

//Valores de control del motor del dedo indice
int posicion_Indice = 0; // posicion del potenciometro
int contraMAX_Indice = 50; // valor del potenciometro en la
contracion maxima
int extenMAX_Indice = 800; // valor del potenciometro en la
extension maxima

//Valores de control del motor del dedo pulgar
int posicion_Pulgar = 0; // posicion del potenciometro
int contraMAX_Pulgar = 750; // valor del potenciometro en la
contracion maxima
int extenMAX_Pulgar = 200; // valor del potenciometro en la
extension maxima


/****************
 *** Programa ***
 ****************/

void setup() {

  //abre el puerto Serie, configura los datos a 9600 bps
  Serial.begin(9600);
  //Se configuran las entradas y salidas
  pinMode(sensorIndice,INPUT);
  pinMode(sensorPulgar,INPUT);
  pinMode(control_A_Indice, OUTPUT);
  pinMode(control_B_Indice, OUTPUT);
  pinMode(control_A_Pulgar, OUTPUT);
  pinMode(control_B_Pulgar, OUTPUT);
  pinMode(habilita_Indice, OUTPUT);
  pinMode(habilita_Pulgar, OUTPUT);

  //Se habilita el puente H
  digitalWrite(habilita_Indice, HIGH);
  digitalWrite(habilita_Pulgar, HIGH);

}

void loop() {
  //lectura de la señal de los sensores y procesamiento de esta
  processEMG();
  Serial.print(muscleActivityI);
  Serial.print(",");
  Serial.print(muscleActivityP);
```

```
  Serial.print(",");
  Serial.print(90);
  Serial.print(",");
  Serial.println(-10);

  //lectura de los potenciometros de ambos motores para saber su
posicion concreta
  posicion_Indice = analogRead(pin_pos_Indice);
  posicion_Pulgar = analogRead(pin_pos_Pulgar);

  /*
   * Movimiento del dedo INDICE:
   * contraccion --> cuando se llegue a un pico en la señal del
sensor del indice (mayor a 50)
   * extension --> cuando se llegue a un pico en la señal del
sensor del pulgar (mayor a 40)
   */
  if (muscleActivityI > 60 && muscleActivityP < 30)
   {
    //no saldremos del bucle hasta que se contraiga del todo o
extendamos
    while(posicion_Indice > contraMAX_Indice && muscleActivityP <
40)
    {
      processEMG();
      Serial.print(muscleActivityI);
      Serial.print(",");
      Serial.print(muscleActivityP);
      Serial.print(",");
      Serial.print(90);
      Serial.print(",");
      Serial.println(-10);

      posicion_Indice = analogRead(pin_pos_Indice);
      //Serial.print("Me contraigo indice: ");
      //Serial.println(posicion_Indice);
      contraerIndice(150);
    }

    //en el caso de que el bucle concluya porque se llegó a la
contracción maxima
    //se mantiene en esa posicion hasta que no se vuelva a
extender el dedo
    //en cualquier otro caso ni entrara en el bucle
    while(muscleActivityP < 40)
    {
      processEMG();
      Serial.print(muscleActivityI);
      Serial.print(",");
      Serial.print(muscleActivityP);
      Serial.print(",");
      Serial.print(90);
      Serial.print(",");
      Serial.println(-10);

      parar(control_A_Indice, control_B_Indice);
      //Serial.print("Me paro indice: ");
      //Serial.println(posicion_Indice);
    }
```

```
    //independientemente de porque se salga del bucle extendemos
el dedo
    while(posicion_Indice < extenMAX_Indice)
    {
      processEMG();
      Serial.print(muscleActivityI);
      Serial.print(",");
      Serial.print(muscleActivityP);
      Serial.print(",");
      Serial.print(90);
      Serial.print(",");
      Serial.println(-10);

      posicion_Indice = analogRead(pin_pos_Indice);
      //Serial.print("Me extiendo indice: ");
      //Serial.println(posicion_Indice);
      extenderIndice(150);
    }

    parar(control_A_Indice, control_B_Indice);

   }


  /*
   * Movimiento del dedo PULGAR:
   * contraccion --> cuando la señal del sensor del pulgar de un
pico (mayor a 55)
   * extension --> cuando la señal del sensor del pulgar vuelva a
un valor inferior a 5
   */
  if (muscleActivityP > 55 && muscleActivityI < 30)
   {
    //no saldremos del bucle hasta que se contraiga del todo o
extendamos
    while(posicion_Pulgar < contraMAX_Pulgar && muscleActivityP >
2)
    {
      processEMG();
      Serial.print(muscleActivityI);
      Serial.print(",");
      Serial.print(muscleActivityP);
      Serial.print(",");
      Serial.print(90);
      Serial.print(",");
      Serial.println(-10);

      posicion_Pulgar = analogRead(pin_pos_Pulgar);
      //Serial.print("Me contraigo pulgar: ");
      //Serial.println(posicion_Pulgar);
      contraerPulgar(200);
    }

    //en el caso de que el bucle concluya porque se llegó a la
contracción maxima
    //se mantiene en esa posicion hasta que no se vuelva a
extender el dedo
    //en cualquier otro caso ni entrara en el bucle
```

```
    while(muscleActivityP > 2)
    {
      processEMG();
      Serial.print(muscleActivityI);
      Serial.print(",");
      Serial.print(muscleActivityP);
      Serial.print(",");
      Serial.print(90);
      Serial.print(",");
      Serial.println(-10);

      parar(control_A_Pulgar, control_B_Pulgar);
      //Serial.print("Me paro pulgar: ");
      //Serial.println(posicion_Pulgar);
    }

    //independientemente de porque se salga del bucle extendemos
el dedo
    while(posicion_Pulgar > extenMAX_Pulgar)
    {
      processEMG();
      Serial.print(muscleActivityI);
      Serial.print(",");
      Serial.print(muscleActivityP);
      Serial.print(",");
      Serial.print(90);
      Serial.print(",");
      Serial.println(-10);

      posicion_Pulgar = analogRead(pin_pos_Pulgar);
      //Serial.print("Me extiendo pulgar: ");
      //Serial.println(posicion_Pulgar);
      extenderPulgar(150);
    }

    parar(control_A_Pulgar, control_B_Pulgar);

   }

  /*
   * Movimiento de pinza:
   * contraccion --> cuando ambas señales lleguen un pico
   * extension --> cuando la señal del sensor del pulgar llegue a
un pico
   */
  if (muscleActivityI > 30 && muscleActivityP > 30)
   {
    //no saldremos del bucle mientras no se produzcan las
contracciones maximas
    while(posicion_Indice > contraMAX_Indice || posicion_Pulgar <
contraMAX_Pulgar)
    {
      processEMG();
      Serial.print(muscleActivityI);
      Serial.print(",");
      Serial.print(muscleActivityP);
      Serial.print(",");
      Serial.print(90);
      Serial.print(",");
```

```
      Serial.println(-10);

      posicion_Indice = analogRead(pin_pos_Indice);
      posicion_Pulgar = analogRead(pin_pos_Pulgar);
      //Serial.print("Me contraigo: ");
      //Serial.print(posicion_Indice);
      //Serial.print("//");
      //Serial.println(posicion_Pulgar);
      contraerIndice(150);
      contraerPulgar(200);
    }

    //se mantiene en esa posicion hasta que no se vuelvan a
extender los dedos
    while(muscleActivityP < 40)
    {
      processEMG();
      Serial.print(muscleActivityI);
      Serial.print(",");
      Serial.print(muscleActivityP);
      Serial.print(",");
      Serial.print(90);
      Serial.print(",");
      Serial.println(-10);

      parar(control_A_Indice, control_B_Indice);
      parar(control_A_Pulgar, control_B_Pulgar);
      //Serial.print("Me paro: ");
      //Serial.print(posicion_Indice);
      //Serial.print("//");
      //Serial.println(posicion_Pulgar);
    }

    //independientemente de porque se salga del bucle extendemos
el dedo
    while(posicion_Indice < extenMAX_Indice || posicion_Pulgar >
extenMAX_Pulgar)
    {
      processEMG();
      Serial.print(muscleActivityI);
      Serial.print(",");
      Serial.print(muscleActivityP);
      Serial.print(",");
      Serial.print(90);
      Serial.print(",");
      Serial.println(-10);

      posicion_Indice = analogRead(pin_pos_Indice);
      posicion_Pulgar = analogRead(pin_pos_Pulgar);
      //Serial.print("Me extiendo: ");
      //Serial.print(posicion_Indice);
      //Serial.print("//");
      //Serial.println(posicion_Pulgar);
      extenderIndice(150);
      extenderPulgar(150);
    }

    parar(control_A_Indice, control_B_Indice);
    parar(control_A_Pulgar, control_B_Pulgar);
```

```
    }

}

/*****************
 *** Funciones ***
 *****************/

/*
 * funciones que indican los distintos movimientos del dedo
 */
void parar (int pin_1, int pin_2)
{
  digitalWrite(pin_1, LOW);
  digitalWrite(pin_2, LOW);
}
void contraerIndice (int velocidad)
{
  analogWrite(habilita_Indice, velocidad);
  digitalWrite(control_A_Indice, LOW);
  digitalWrite(control_B_Indice, HIGH);
}
void contraerPulgar (int velocidad)
{
  analogWrite(habilita_Pulgar, velocidad);
  digitalWrite(control_A_Pulgar, HIGH);
  digitalWrite(control_B_Pulgar, LOW);
}
void extenderIndice (int velocidad)
{
  analogWrite(habilita_Indice, velocidad);
  digitalWrite(control_A_Indice, HIGH);
  digitalWrite(control_B_Indice, LOW);
}
void extenderPulgar (int velocidad)
{
  analogWrite(habilita_Pulgar, velocidad);
  digitalWrite(control_A_Pulgar, LOW);
  digitalWrite(control_B_Pulgar, HIGH);
}


/*
 * funcion que procesa la señal del sensor mioelectrico
 */
void processEMG()
{
  // Measure time step
  float dt = 0.001 * (millis() - t);
  t = millis();

  // Read raw myoelectric signals
  valueMuscleI = analogRead(sensorIndice);
  valueMuscleP = analogRead(sensorPulgar);

  // High-pass filtering (remove signal mean value)
  float value1filt = tau1 / (tau1 + dt) * y1_1 + tau1 / (tau1 +
dt) * (valueMuscleI - x1_1);
```

```
  y1_1 = value1filt;
  x1_1 = valueMuscleI;
  float value2filt = tau1 / (tau1 + dt) * y2_1 + tau1 / (tau1 +
dt) * (valueMuscleP - x2_1);
  y2_1 = value2filt;
  x2_1 = valueMuscleP;

  // Signal rectification and squaring
  float value1filtrect = pow(value1filt, 2);
  float value2filtrect = pow(value2filt, 2);

  // Low-pass filtering
  float value1env = tau2 / (tau2 + dt) * z1_1 + dt / (tau2 + dt) *
value1filtrect;
  z1_1 = value1env;
  float value2env = tau2 / (tau2 + dt) * z2_1 + dt / (tau2 + dt) *
value2filtrect;
  z2_1 = value2env;

  // Moving average (compute signal RMS value)
  sum1 = sum1 - buffer1[i1];
  sum1 = sum1 + value1env;
  buffer1[i1] = value1env;
  i1++;
  if (i1 >= buffLength)
  {
    i1 = 0;
  }
  muscleActivityI = sum1 / buffLength;
  sum2 = sum2 - buffer2[i2];
  sum2 = sum2 + value2env;
  buffer2[i2] = value2env;
  i2++;
  if (i2 >= buffLength)
  {
    i2 = 0;
  }
  muscleActivityP = sum2 / buffLength;
}
```