MEJORAS EN UNA PRÓTESIS MIOELÉCTRICA DE MANO PARA SU USO EN REHABILITACIÓN

IMPROVEMENTS IN A MYOELECTRIC HAND FOR USE IN REHABILITATION

Víctor Rodríguez González

UNIVERSIDAD DE VALLADOLID

Tutelado por Alonso Alonso Alonso

RESUMEN

Las prótesis mioeléctricas están cada vez más evolucionadas. Por ello resulta interesante poder desarrollar uno de estos dispositivos con un presupuesto ajustado, para que así todo el mundo pueda acceder a una prótesis mioeléctrica.

En el presente documento se realizará un recorrido por el proceso de creación de una prótesis mioeléctrica asequible, sin dejar de ser funcional. Se comenzará con una introducción teórica, para después continuar con el desarrollo del hardware, separando cada una de las componentes. Posteriormente se verá el desarrollo del software, así como todas las pruebas que han sido necesarias para el correcto funcionamiento de la prótesis. También se ha tenido en consideración la necesidad de incluir un sistema de sensores en la prótesis que nos ayuden a que el manejo de esta sea más natural y eficiente.

El resultado final es una mano mioeléctrica que, si bien no es funcional, tiene mucho potencial para serlo, tras aplicar unas pequeñas mejoras, las cuales también han sido reflejadas en la memoria en el apartado "líneas de mejora".

Palabras claves: Mano, prótesis, mioeléctrica, Arduino y sensor.

ABSTRACT

Myoelectric prothesis are increasingly evolved. That is why is very interesting to develop a low-cost prothesis, so that everyone can afford to have one.

This document is a guide of the creation process of a functional and affordable myoelectric prothesis. It begins with a theoretical introduction, and then it continues with the hardware development, dividing it in its components. Later, software development and all of the functionality test required for the correct operation of the prothesis will be seen. It has been considered the need of a sensor system in the prothesis that help us to use the hand more naturally and efficiently.

The result of this word is a myoelectric hand, that is no functional. However, it has great potential to be functional by applying some improvements that have been indicated in this document in the section 'Improvement lines'.

Keywords: Hand, prothesis, myoelectric, Arduino and sensor.

AGRADECIMIENTOS

En primer lugar, agradecer a mi tutor Alonso, por ofrecerme la posibilidad de realizar este trabajo, y por ofrecerme toda su ayuda y conocimientos en esos momentos en los que las cosas no salían como esperaba. También me gustaría agradecer tanto a Alonso cómo a Ramón esos ratos de descanso que tan necesarios eran tras varias horas de duro trabajo.

También me gustaría agradecer a todos los amigos y compañeros que han estado a mi lado y me han acompañado en este camino que está a punto de acabar y que comenzó hace ya casi 5 años.

Por supuesto, y por encima de todo agradecer a mi familia, a mis padres y a mi hermana, porque sin ellos nada de esto hubiera sido posible, por aguantarme en los momentos en los que parecía que nada iba bien y por tener siempre esas palabras de ánimo. Finalmente, agradecer a Paula ser mi apoyo incondicional en todo momento, por estar a mi lado en los peores momentos, y por nunca abandonarme cuando todo falla.

A todos, muchas gracias. Porque gracias a vosotros me he convertido en la persona que soy ahora.

ÍNDICE

ÍNDICE DE CONTENIDOS

ÍNDICE DE IMÁGENES

INTRODUCCIÓN

1.1. JUSTIFICACIÓN DEL TFG

Hay un número importante de personas que carecen de alguno de sus miembros, ya sea de nacimiento, o a causa de algún accidente automovilístico, laboral… Antiguamente, estas personas no eran aptas para realizar un trabajo, y por ello quedaban 'apartadas' del resto, sufriendo una discriminación pasiva.

Con el paso del tiempo, se fueron desarrollando prótesis mecánicas que, aunque rudimentarias, resultaban lo suficientemente funcionales para que los amputados pudieran comenzar poco a poco a integrarse en el mundo laboral. Hoy en día se han desarrollado prótesis mucho más avanzadas, que permiten un enorme grado de integración para el paciente, sin embargo, tienen un gran inconveniente, su precio. Estas prótesis pueden llegar a costar 90000€, precio que muchas de las personas amputadas no se pueden permitir.

Enfocando ese problema, en este TFG se tratará de buscar una prótesis lo más funcional posible, pero reduciendo el presupuesto lo máximo posible, tratando que su precio final quede por debajo de los 1000€.

Para conseguir una mano funcional a un precio tan reducido, se deberá renunciar a algunas características de las prótesis punteras, pero tratando que sean cosas secundarias que no afecten a la funcionalidad de la mano. Ejemplos de cosas a las que se podría renunciar son una piel artificial y baterías portátiles. Sin embargo, características imprescindibles serían el poder cargar algo de peso con la mano, dedos controlados independientemente unos de otros…

Tras un estudio de cómo realizar una prótesis de mano con dichas características, se descubrió un mundo que ofrecía una inmensa variedad de posibilidades, las prótesis impresas con impresoras 3D.

Ante la gran cantidad de posibilidades que este mundo ofrecía, decidí ahondar más en él, analizando varias de las prótesis disponibles, para encontrar la que mejor se adaptara a nuestras exigencias.

***Imagen 1**. Prótesis de mano creada con una impresora 3D.*

Hay una gran cantidad de manos, cuyos modelos e instrucciones de montaje se ofrecen de forma gratuita en internet, por lo que tras un análisis superficial de varias de ellas decidí centrarme en un proyecto, el cual había creado varias manos, el proyecto 'Open Hand Project'. Se optó por este proyecto por dos motivos: Por un lado, contaban con algo de experiencia al haber creado más de una mano, y por otro lado, sus manos contaban con tres articulaciones móviles para cada dedo, lo cual da a la prótesis un aspecto más humano. Este último motivo es importante pues la mayoría de las manos disponibles sólo cuentan con dos articulaciones móviles, mientras que la tercera es fija.

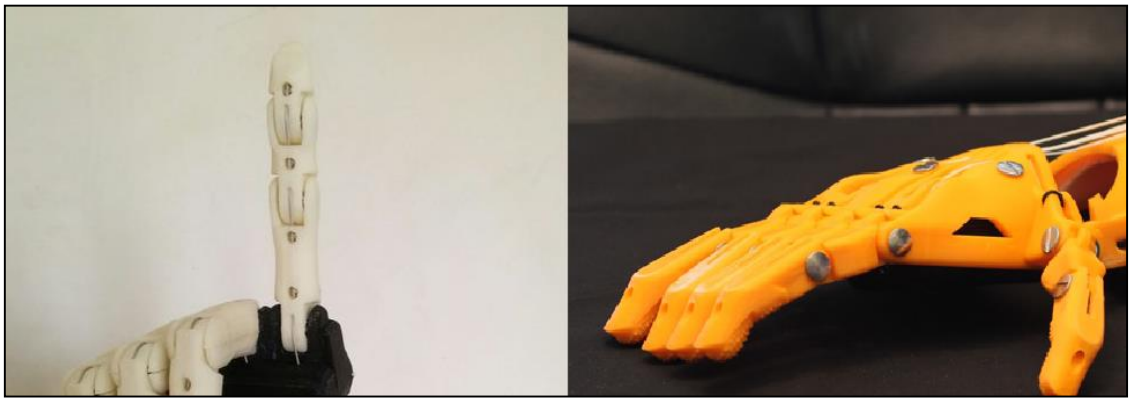


***Imagen 2**. Dedos de las prótesis Dextrus (Izquierda) y Cyborg Beast (Derecha).*

Centrándonos ya sobre el proyecto Open Hand Project, se ve que este cuenta con dos modelos de mano:

- **Dextrus**: Disponible en versiones 1.0, que fue la inicial; 1.1 que mejoraba la 1.0 en diversos aspectos y la 2.0 que cambiaba totalmente el enfoque de los dos primeros modelos, empleando incluso materiales diferentes. Dextrus 1.X emplea en su construcción materiales sólidos como ABS y PLA, mientras que Dextrus 2.0 emplea un material flexible, el Filaflex. El motivo que me llevó a escoger la versión 1.1 es que una de nuestras exigencias es que pudiera coger objetos de un cierto peso, y para ello es más adecuado un material rígido que uno flexible.
- **Ada**: Es una versión nueva, lanzada junto con un lavado de cara del proyecto, que pasaría a llamarse 'Open Bionics'. Ada es un modelo nuevo, mientras que Dextrus v1.1 es una revisión del modelo inicial, este hecho aporta una fiabilidad extra al diseño, y es el motivo que me ha llevado a trabajar con el modelo Dextrus v1.1 en detrimento del modelo Ada.

Recapitulando, buscábamos cómo construir una prótesis de mano barata y funcional y para ello nos fijamos en el mundo de las prótesis impresas con impresoras 3D. Dentro de estas nos centramos en el proyecto Open Hand Project (Open Bionics), del cual escogimos crear la prótesis Dextrus v1.1

Esta prótesis tiene los modelos 3D disponibles para que cualquiera se los pueda descargar en el siguiente enlace:

http://www.thingiverse.com/thing:287638

Y las instrucciones para su ensamblaje en el siguiente:

http://www.instructables.com/id/Dextrus-v11-Robotic-Hand/

1.2. PROBLEMÁTICA Y OBJETIVOS DEL TFG

En el presente TFG pretendemos construir una mano mioeléctrica que sea perfectamente funcional, y que pueda ser fabricada con un bajo presupuesto, menos de 500€ (Sin contar la impresora 3D). La prótesis de mano pretenderá tener similares funcionalidades a las que nos ofrecen las prótesis más avanzadas del mercado, aunque eso sí, implementadas de una forma mucho más rudimentaria debido al bajo presupuesto disponible.

Para lograr el objetivo del bajo presupuesto renunciaremos a elementos como una piel artificial, elemento estético de gran importancia, pero cuya ausencia no restará funcionalidad a nuestra mano. Por otro lado, nuestra mano estará construida a partir de unos modelos 3D disponibles en la red. Esto hará que, aunque nuestra mano tenga una forma antropomórfica, sus dimensiones no se ajusten del todo a las de una mano humana, teniendo un tamaño significativamente mayor al de estas, sobre todo en el grosor de la palma.

Tampoco se prestará demasiada atención a lo que podríamos considerar los últimos pasos de la implantación de la prótesis, los cuales incluyen: Reducción del tamaño de las placas electrónicas, búsqueda de un sistema de alimentación portátil y ensamblado de la mano al muñón de los pacientes. Estos aspectos son de gran importancia, pero se alejan del objetivo de este TFG, además son procesos cuyo desarrollo llevaría mucho tiempo, por lo que los dejaremos relegados al apartado 'Líneas de mejora' para que puedan ser desarrollados en un futuro.

Además del problema del presupuesto, deberemos prestar atención a la velocidad y fuerza de la mano. El objetivo es que nuestra mano pueda agarrar cosas medianamente pesadas, como pueda ser un botellín de agua, y que no emplee demasiado tiempo en abrirse y cerrarse para que la experiencia de su uso sea medianamente similar a la del empleo de una mano humana, pudiéndose trabajar con ella prácticamente en tiempo real.

También se tratará de crear un sensor de contacto que nos mejore el control de la mano. Para esto realizaremos varias pruebas con diferentes posibles sensores, viendo cuál es el que mejor se adapta a nuestros requerimientos. Uno de los sensores que se estudiarán es uno basado en grafeno. Para crear el sensor, se realizarán varias pruebas con dicho material, gracias a las diversas muestras que se han podido conseguir.

Finalmente, el último problema importante con el que deberemos lidiar es la implementación de un sistema de control mioeléctrico mediante el cual cualquier persona pueda manejar la prótesis. Se adaptarán los parámetros de ganancia a mis señales mioeléctricas, pero se intentará que el sistema cuente con una cierta flexibilidad para que el control sea suficientemente preciso con independencia del usuario que lo maneje.

1.3. ETAPAS DEL DESARROLLO DEL TFG

Para el desarrollo del TFG se divide el proceso en varias etapas bien definidas, y hasta cierto punto, independientes entre sí. De este modo podremos fragmentar el problema que afrontamos en varios problemas de menor dimensión, los cuales nos resultarán más sencillos de abordar y resolver.

El primer paso de todos es la documentación. Durante este proceso se revisará gran cantidad de información acerca de impresoras 3D, prótesis, señales mioeléctricas, modelado 3D… Esta información será comparada entre sí y clasificada, viendo que puntos comunes y que discrepancias existen entre los diferentes documentos revisados, para así poder tomar decisiones lo más fundamentadas posible. Tras el proceso de documentación, revisión y clasificación de la información deberemos tomar dos decisiones importantes: Qué impresora 3D utilizar y qué prótesis mioeléctrica construir.

El siguiente paso será informarse en profundidad acerca de la impresora 3D y la prótesis escogidas, para poder afrontar los siguientes pasos con un conocimiento de ambas lo más extenso posible.

Llegados a este punto, comenzaremos con el montaje de la mano. Para empezar, montaremos tan solo la parte mecánica de la mano, sin control mioeléctrico, sin electrónica, sin Arduino… Es decir, ensamblaremos tan solo la mano con sus motores.

Para ello, lo primero que deberemos hacer es montar nuestra impresora 3D con los manuales disponibles en la web, y una vez montada imprimir las piezas necesarias para nuestro montaje. Durante este proceso deberemos comprar el resto de piezas necesarias para nuestra mano, para así poder comenzar el propio montaje de la mano disponiendo ya de todos los materiales necesarios, evitando así tener que parar nuestro trabajo para esperar la llegada de alguna pieza. Una vez que dispongamos de todos los materiales, podremos comenzar con el montaje de la mano en sí. Este proceso será largo y complicado, y en algún momento requerirá alejarse un poco de los manuales disponibles en internet debido a naturaleza no exacta de las piezas impresas en 3D.

Cuando hallamos terminado el paso anterior ya deberíamos contar con nuestra mano completamente montada, cuyos dedos deberían abrirse y cerrarse al conectar sus motores a la corriente. Por ello, el siguiente paso sería la creación de un montaje electrónico que nos permita controlar la mano desde el ordenador a través de una placa de desarrollo del tipo ‘Arduino’.

Aquí de nuevo tendremos que llevar a cabo una búsqueda de información. Esta búsqueda será más breve que las llevadas a cabo en pasos anteriores, pues tan solo deberemos buscar qué placa de Arduino es la que mejor se adapta a nuestras necesidades y qué integrados son los que nos pueden ayudar para resolver nuestro problema. Una vez resueltas estas cuestiones podremos llevar a cabo el montaje sobre una placa de pruebas, para poder modificarlo en caso de que se presente algún error.

Con el montaje ya listo, procederemos a probarle. Para ello deberemos crear un programa en Arduino, lo más sencillo posible, en el cual probaremos cada motor individualmente, para cerciorarnos de que todos funcionan correctamente. Una vez realizada esta comprobación

deberemos pasar nuestro montaje de una placa de pruebas a una placa definitiva, soldando los componentes para que no puedan desengancharse. Con el montaje ya puesto sobre la placa definitiva, emplearemos de nuevo nuestro programa de Arduino para ponerle a prueba y ver que todo funciona correctamente.

En este paso ya tendremos la mano con toda su electrónica correctamente montada, por lo que podríamos pasar a la parte software. Diseñaremos un programa, con el cual podremos escoger entre los diferentes movimientos de la mano disponibles con un ordenador, a través del puerto serie de Arduino. Este programa nos permitirá realizar todas las pruebas necesarias con la mano, hasta que la incorporemos el sensor mioeléctrico, momento en el cual deberemos cambiar el programa, para crear uno nuevo en el que la mano se controle con dicho sensor.

Una vez comprobado el programa, y antes de incorporar a este el control mioeléctrico, se pasará el montaje de la placa de pruebas a una placa de pistas, en la cual el montaje quedará fijado definitivamente.

El siguiente paso sería la creación de un sistema de control mioeléctrico y la incorporación del mismo a la prótesis. En primer lugar, se realizará un solo canal del control mioeléctrico y una vez comprobado el correcto funcionamiento de este, se replicará para así poder realizar las pruebas pertinentes con los dos canales simultáneamente. Una vez comprobado el correcto funcionamiento de los dos canales, se pasarán estos a una placa de pistas para que nuestro montaje quede fijado. Con nuestros dos canales ya fijados sobre una placa de pistas pasaremos a integrarlos en el Arduino.

Para ello en primer lugar crearemos un programa de Arduino con el que evaluaremos el funcionamiento de los canales y estableceremos de manera óptima los umbrales de decisión para poder controlar la mano con nuestras señales EMG.

Una vez fijados estos, podremos pasar a la creación del software final. Dicho software combinará el control de los motores realizado anteriormente con el mando de control mioeléctrico creado. Nos serviremos de los umbrales fijados para que cada uno de los canales sea capaz de distinguir entre tres niveles de esfuerzo, de esta manera, podremos acceder a 9 movimientos distintos. Estos movimientos serán tomados del software en el que seleccionábamos distintos movimientos de la mano mediante el PC.

De este modo quedará creado un sistema mediante el cual podremos controlar los movimientos de la prótesis por medio de nuestras señales EMG. Todo lo que nos quedará será alojar la electrónica en una caja para que quede resguardado de los daños que pudiera recibir durante su transporte y manipulación.

Paralelamente a esto, se investigará la creación de un sensor de contacto que nos asista en el control de la mano. Una de las vías que se estudiarán es la de crear el sensor explotando las propiedades del grafeno, gracias a unas muestras de este material que se han podido conseguir.

1.4. IMPRESORAS 3D

Definimos impresión 3D cómo la reproducción de objetos con volumen a partir de un prototipo diseñado por ordenador.

1.4.1. EVOLUCIÓN HISTÓRICA DE LAS IMPRESORAS 3D

Aunque su enorme presencia actual nos podría hacer pensar que la impresión tridimensional es algo reciente, esto no es así. Realmente, la impresión 3D nació en 1984, año en que Chuck Hull creó la primera impresora tridimensional, para poco después, en 1986, fundar la empresa 3D Systems en California y desarrollar la primera impresora tridimensional comercial, basada en la estereolitografía (SLA), que fue lanzada al mercado en 1988.

Ese mismo año aparecieron dos técnicas nuevas de impresión 3D:

- FDM que fue desarrollado por Scott Crump, que fundó en 1989 Stratasys, otra de las grandes empresas dedicadas a la impresión 3D hoy en día, la cual lanzó en 1992 su impresora 3D Modeler.
- SLS que fue creado por los doctores Carl Deckard y Joe Beaman de la Universidad de Texas, en Austin, y con patrocinio del gobierno de los EEUU a través de DARPA. Posteriormente fundaron la conocida compañía DTM, otra de las empresas pioneras en la impresión 3D, que en 1992 también sacó al mercado su propia impresora 3D con bastante éxito. Pero finalmente en 2001 3D Systems terminó comprando DTM y acabando con parte de su competencia más directa.

En 1993 un grupo de estudiantes del MIT concibe la impresión 3D por inyección. La empresa Z Corporation se hizo con la licencia en exclusiva de esta tecnología en 1995, convirtiéndose en sus fabricantes y distribuidores a nivel mundial. Fue en 1996, con la llegada de los principales representantes de cada una de estas tecnologías, cuando se consolidó el término impresora tridimensional.

Todas estas técnicas e impresoras manejaban materiales relativamente poco resistentes, como plásticos, polímeros o yeso, pero paralelamente en Alemania empezaban a surgir también otras tecnologías de impresión tridimensional que tenían como objetivo el uso de metales y aleaciones metálicas. Surgen así el fundido láser selectivo (SLM) cuyo desarrollo comenzó en el Fraunhofer Institute ILT en Aachen y en el que colaboraron los doctores Dieter Schwarze, Matthias Fockele, Wilhelm Meiners y Konrad Wissenbach. De aquí surgieron las empresas alemanas SLM Solutions y Realizer. Existen otras técnicas de impresión 3D de metales, así como varias empresas que han ido desarrollando mejoras e innovaciones, pero su importancia se centra en el sector industrial y no han sido partícipes en la creciente popularización de las impresoras 3D que se está viendo actualmente.

En 2005 el doctor Bowyer, de la Universidad de Bath, Reino Unido, desarrolla la primera máquina 3D autorreplicante: la RepRap, que supone un salto adelante en la normalización de las impresoras tridimensionales, facilitando el acceso del gran público a estas. Finalmente destacar que en el 2009 la empresa Organovo crea la impresora 3D MMX Bioprinter, la primera capaz de fabricar tejidos orgánicos, suponiendo esto un importante avance para el empleo de las impresoras 3D en campos como la medicina.

1.4.2. TIPOS DE IMPRESORAS 3D [1] [2]

Para continuar, veremos los tipos de impresora 3D que existen, así como el funcionamiento básico de cada uno de ellos. Esto nos ayudará a comprender mejor la evolución de las impresoras 3D y nos proporcionará una información importante para entender mejor este mundo.

Impresoras 3D por Estereolitografía (SLA): Fue la primera técnica que se usó, y la que empleaban las primeras impresoras 3D comerciales, lanzadas al mercado por la compañía 3D Systems. Es un método 'capa a capa' en el cual se hace incidir un haz de luz UV sobre una base de resina fotosensible, que al contacto con la luz UV se va solidificando capa a capa. La base que soporta la estructura se desplaza hacia abajo para que la luz vuelva a ejercer su acción sobre el nuevo baño, y así hasta que el objeto alcance la forma deseada. Con este método se consiguen piezas de altísima calidad, aunque se desperdicia cierta cantidad de material en función del soporte que sea necesario fabricar para la pieza. Algunos ejemplos de impresoras 3D que funcionan por estereolitografía son: Projet 1500, 1200 o 3510 de 3D Systems.

https://www.youtube.com/watch?v=NM55ct5KwiI

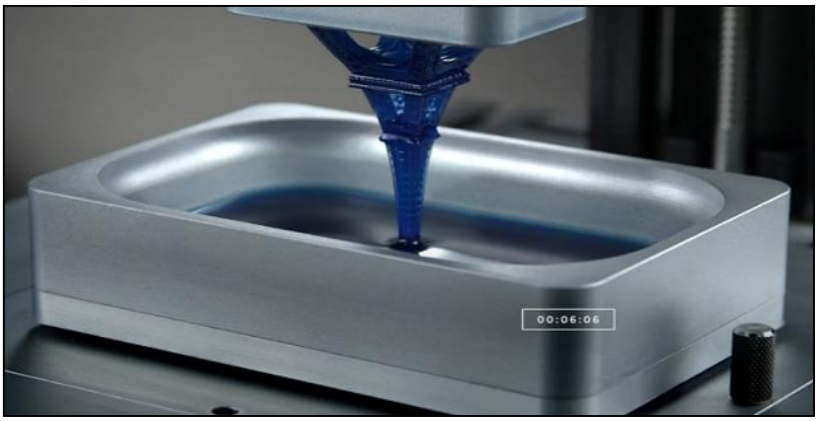


***Imagen 3**. Impresión por Estereolitografía.*

Impresoras 3D de Sinterización Selectiva por Láser (SLS): Esta tecnología guarda ciertas similitudes con la SLA consiguiendo también piezas de alta calidad, sin embargo, la tecnología SLS permite utilizar un gran número de materiales en polvo (Poliestireno, materiales cerámicos, cristal, nylon y materiales metálicos). El láser impacta en el polvo, funde el material y este se solidifica. Todo el material que no se utiliza queda almacenado, por lo que no se desperdicia nada. Una de las impresoras 3D más famosas que utilizan esta tecnología de impresión 3D es la EOS.

https://www.youtube.com/watch?v=9E5MfBAV_tA

Impresión por deposición de material fundido (FDM): La tecnología FDM estaba protegida por patentes. El vencimiento de determinadas de esas patentes y el nacimiento de una tecnología muy similar, Fused Filament Fabrication (FFF) han permitido el boom reciente de la impresión 3D personal. Se trata de la técnica más común en cuanto a impresoras 3D de escritorio y usuarios domésticos.

Es una tecnología que permite conseguir piezas utilizando plástico ABS (similar al material de los juguetes Lego) o bien PLA (un polímero biodegradable que se produce desde un material orgánico). Esta tecnología consiste en depositar polímero fundido sobre una base plana, capa a capa. El material, que inicialmente se encuentra en estado sólido almacenado en rollos, se funde

y es expulsado por la boquilla en minúsculos hilos que se van solidificando conforme se depositan en la base, en su posición correspondiente de la capa que se está imprimiendo.

Aunque los resultados pueden ser muy buenos, no suelen ser comparables con los que ofrecen las impresoras 3D por SLA o por SLS. La ventaja principal es que esta tecnología ha permitido poner la impresión 3D al alcance de cualquier persona con impresoras como la CubeX o la Prusa.

https://www.youtube.com/watch?v=8_vloWVgf0o

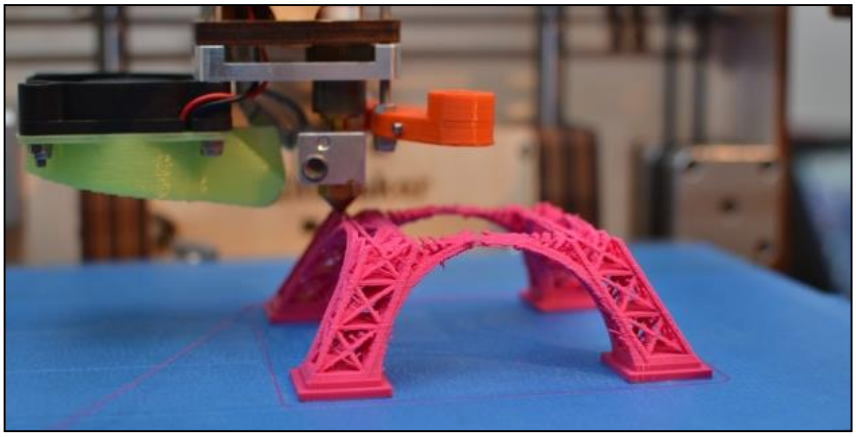


***Imagen 4**. Impresión por deposición de material fundido.*

Impresoras 3D por Laminated Object Manufacturing (LOM): Donde distintas capas de material (Papel adhesivo, plástico o láminas de metal) son situadas una encima de otra, se pegan y son cortadas con la forma apropiada empleando un láser. Este proceso recuerda en parte al modo de fabricación de la fibra de carbono.

https://www.youtube.com/watch?v=jNJbusVcVQc

Impresoras 3D por Inyección: Es el sistema más parecido a una impresora habitual (de tinta en folio), pero en lugar de inyectar gotas de tinta en el papel, inyectan capas de fotopolímero líquido que se curan en la bandeja de construcción. Muy similar a la tecnología de impresión por láser, su diferencia con ésta radica en que, en lugar de emplear un láser, el material (Que estará en polvo y a una temperatura cercana a la fundición) se compactará mediante inyección de un aglomerante (tinta). Esta tecnología permite imprimir en color, ya que el aglomerante utilizado puede tener un color u otro. Como ejemplo de impresoras 3D por inyección destacamos X60 de 3D Systems o la Zprint 450.

https://www.youtube.com/watch?v=1C7hkG5lwjY

Impresoras 3D por Syringe Extrusion: No es más que un sistema de impresión que hace uso de cualquier tipo de material en formato cremoso o viscoso, haciendo uso de un extrusor a modo de jeringuilla, situando el material en la posición adecuada. Según el material utilizado se requerirá que el extrusor caliente (Por ejemplo, con chocolate) o no (Silicona).

***Imagen 5**. Impresora 3D por Syringe Extrusion.*

1.4.3. APLICACIONES DE LAS IMPRESORAS 3D

Las impresoras 3D tienen incontables usos, tanto industriales como domésticos. A continuación, veremos algunos de los usos más destacados de estas:

- *En educación*: Las aplicaciones en educación son infinitas, ayudando a los alumnos a entender y visualizar conceptos abstractos. Las aplicaciones en sociales, ciencias de la naturaleza, matemáticas, arte, historia y, por supuesto, tecnología, son capaces de revolucionar la actividad pedagógica.
- *En arqueología*: El uso de tecnologías de escaneo 3D, permite la réplica de objetos reales sin el uso de procesos de moldeo, que en muchos casos pueden ser más caros, más difíciles y demasiado invasivos para ser llevados a cabo; en particular, con reliquias arqueológicas de alto valor cultural donde el contacto directo con sustancias de moldeo puede dañar la superficie del objeto original.
- *En arte*: Artistas han usado impresoras 3D de diferentes maneras. Durante el Festival de Diseño de Londres, un montaje, desarrollado por Murray Moss y dirigido a la impresión 3D tuvo lugar en el Museo de Victoria y Alberto. La instalación fue llamada Industrial Revolution 2.0: How the Material World will Newly Materialise.
- *En biotecnología*: La tecnología de impresión 3D está siendo actualmente estudiada en el ámbito de la biotecnología, tanto académico como comercial, para su posible uso en la ingeniería de tejidos, donde órganos y partes del cuerpo son construidas usando técnicas similares a la inyección de tinta en impresión convencional. Capas de células vivas son depositadas sobre un medio de gel y superpuestas una sobre otra para formar estructuras tridimensionales. Algunos términos han sido usados para denominar a este campo de investigación, tales como impresión de órganos, bio-impresión e ingeniería de tejidos asistida por computadora.
- *En prótesis*: El uso de impresión 3D en prótesis está avanzando rápido. En el año 2012 por primera vez hicieron un trasplante de una mandíbula impresa en titanio. Hoy, trasplantes de este tipo ya no son una novedad y existen muchos casos exitosos de trasplantes de caderas, mandíbulas y otras partes del cuerpo. Más importante aún es que desde que aparecieron las impresoras 3D de bajo costo, las personas fuera del

campo médico empezaron a experimentar e inventar sus propias soluciones, cómo es el caso de este TFG.

- *En farmacología*: El proceso de personalización no tiene que ser limitado solamente a crear las formas precisas, también puede ser usado para imprimir pastillas personalizadas directamente en las farmacias. Uno de los desafíos de la medicina geriátrica es que el paciente a veces tiene que tomar hasta diez pastillas por día y es fácil de olvidar o tomar la dosis equivocada. Con la llegada de la farmacología digital, tu médico va a poder indicar online exactamente qué combinación de medicamentos y en qué dosis debería tomar el paciente, y la farmacia de al lado va a imprimir una sola pastilla que contenga todos los tratamientos indicados para esta persona.
- *En construcción*: Ya existen varias empresas y equipos académicos que están en las etapas avanzadas de llevar las tecnologías aditivas a la industria de la construcción. Una de las empresas líderes se llama D-Shape, y desarrolló una impresora 3D con la capacidad de imprimir cualquier estructura arquitectónica que encaje en un cubo de 6 metros por lado.
- *En alimentación*: Se han modificado impresoras Open Source como RepRap para crear máquinas que imprimen galletitas, chocolates y mucho más. Pastelería es una de las primeras aplicaciones en impresión de comida porque estos productos suelen estar hechos de un solo material, y por lo tanto es fácil modificar una impresora 3D existente para adaptarla a este propósito.
- *Domésticos*: Un usuario particular con una impresora 3D y conocimientos en modelado 3D, puede encontrar tantos usos cómo su imaginación le permita: Podría imprimir desde figuras de sus personajes favoritos, hasta fundas para su teléfono, pasando por el escudo de su equipo favorito.

***Imagen 6**. Personaje de animación creada con una impresora 3D.*

1.5. EVOLUCIÓN, TIPOS Y ESTADO DEL ARTE DE LAS PRÓTESIS DE MANO.

1.5.1. EVOLUCIÓN HISTÓRICA

La mano humana realiza principalmente dos funciones: La prensión y el tacto, las cuales permiten al hombre realizar todo tipo de tareas. Adicionalmente, la mano nos ayuda a comunicarnos, añadiéndonos una expresividad que va más allá de las palabras, como en el caso de las esculturas o de los sordomudos. El sentido del tacto desarrolla totalmente las capacidades de la mano, pues sin este sería imposible medir la fuerza prensora, o detectar temperaturas que podrían dañarnos. Por último, es importante mencionar que el dedo pulgar representa el miembro más importante de la mano, pues sin este la capacidad funcional de la mano se reduce enormemente.

Sustituir por pérdida alguno de los miembros humanos es una acción que se lleva realizando desde hace más de dos mil años. Sin embargo, fue durante el siglo XX cuando las prótesis lograron mayores avances, desarrollándose con el objetivo de que los amputados pudieran regresar a la vida laboral.

Inicialmente el objetivo propuesto se alcanzó por el médico francés Gripoulleau, quien fabricó distintos accesorios que podían ser utilizados como unidad terminal. En 1912, Dorrance, en los Estados Unidos, desarrolló una unidad terminal llamada Hook que podía abrirse y cerrarse activamente mediante movimientos de la cintura escapular combinados con un tirante de goma.

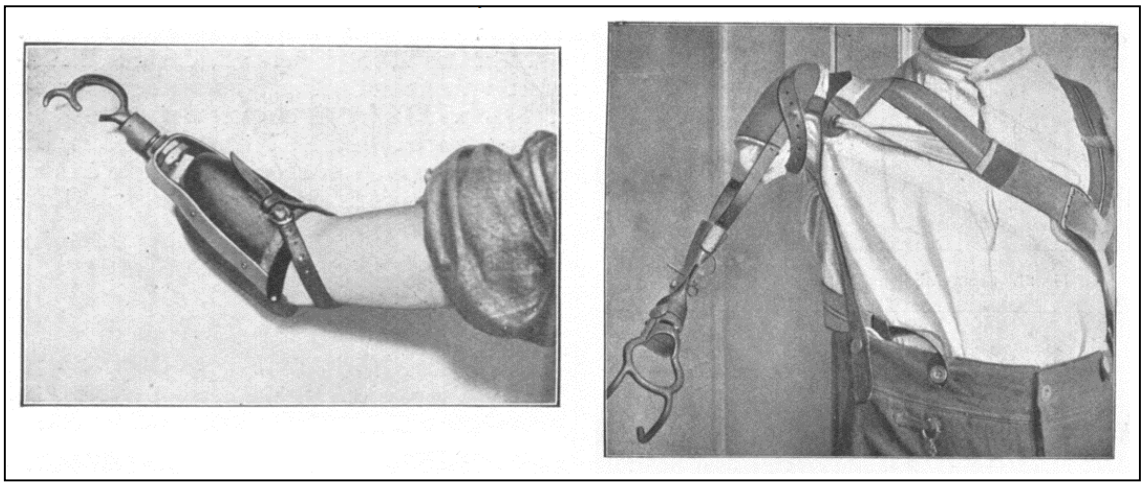


***Imagen 7**. Prótesis Hook desarrollada por Dorrance en 1912.*

La prótesis con mando mioeléctrico tiene su origen en Rusia durante la década del 60. Esta opción protésica basa su control en los pequeños potenciales eléctricos extraídos de las masas musculares de una parte sana del brazo, siendo conducidos y amplificados para poder controlar los movimientos de la prótesis con dichos potenciales. A finales del siglo XX las funciones de las prótesis con mando mioeléctrico, estaban limitadas al cierre y apertura de una pinza. Las diferencias entre los distintos modelos consistían en el tipo de control que emplean, pero la mayoría realizaban básicamente las mismas funciones.

En los últimos años, con los avances recientes en electrónica y materiales inteligentes, se ha vivido de nuevo un gran desarrollo de las prótesis de mano, consiguiendo importantes avances como un pulgar controlado activamente o retroalimentación para informar al paciente de la fuerza que se está ejerciendo o de la temperatura a la que está el objeto que se está agarrando. Actualmente, los países con mayor avance tecnológico en investigación y desarrollo de prótesis son: Alemania, Estados Unidos, Francia, Inglaterra y Japón [1].

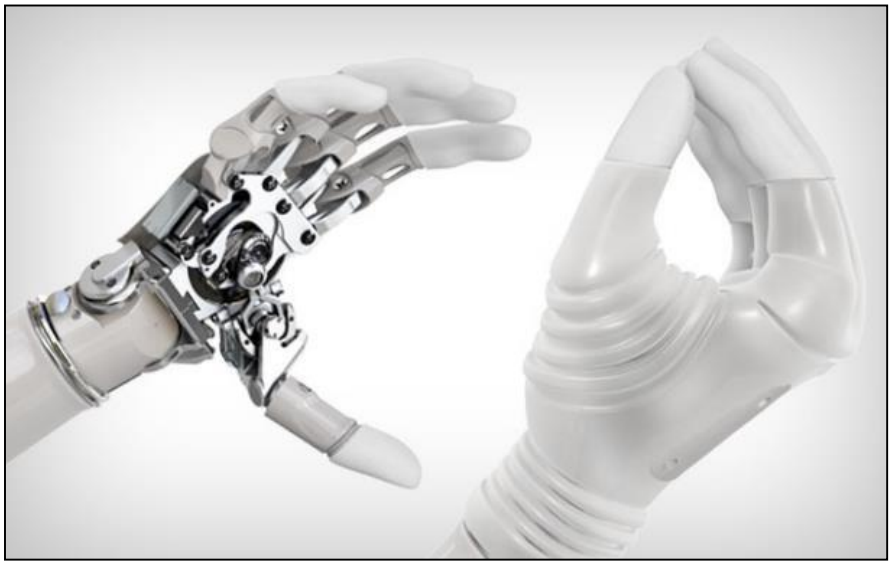


***Imagen 8**. Prótesis de mano 'Michelangelo'. Una de las prótesis más avanzadas del mercado.*

1.5.2. TIPOS DE PRÓTESIS

A la hora de escoger la prótesis a utilizar, se deben de tener en cuenta varios factores, cómo el nivel de amputación, el tipo de displasia, la funcionalidad de la misma y las posibilidades económicas del paciente. A continuación, veremos los tipos de prótesis que existen actualmente, las cuales se han desarrollado utilizando diferentes tecnologías y conocimientos.

PRÓTESIS ESTÉTICAS

Este tipo de prótesis, conocidas también cómo prótesis pasivas, no tienen movilidad y sólo cubren el aspecto estético del miembro amputado. En su fabricación se emplean polímeros cómo el PVC rígido, el látex flexible o la silicona, pues son materiales livianos y requieren de menos mantenimiento, ya que no disponen de piezas móviles.

***Imagen 9**. Prótesis estética.*

PRÓTESIS MECÁNICAS

Estas prótesis cumplen funciones básicas cómo la apertura y el cierre de la mano, pero están limitadas al agarre de objetos grandes y movimientos imprecisos. El control mecánico es obtenido de otro miembro cómo el codo o el hombro, para ello se implementa un arnés colocado en la espalda, con el cual se generará la movilidad de la prótesis a través de una liga, cómo se aprecia en la Imagen 10.

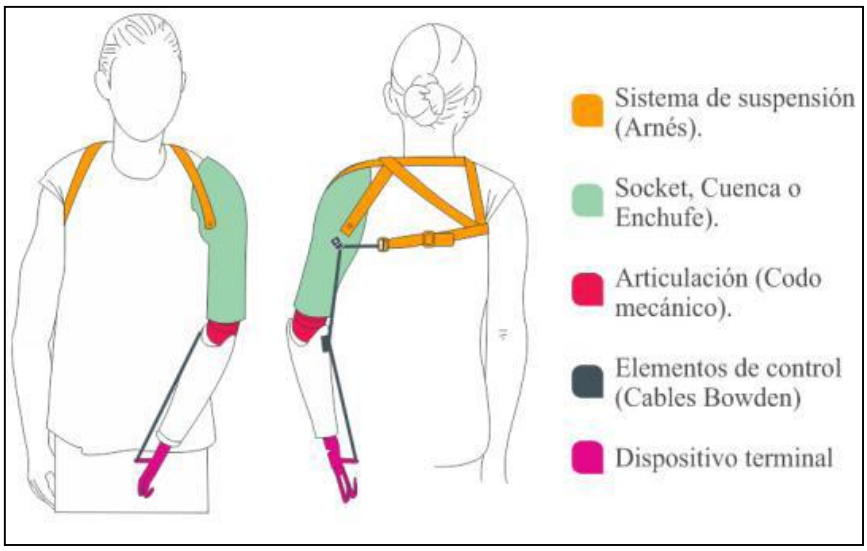


***Imagen 10**. Prótesis mecánica.*

PRÓTESIS ELÉCTRICAS

Las prótesis eléctricas se basan en el uso de motores eléctricos, que pueden ser controlados por medio de servo-controles, pulsantes o interruptores. Su principal desventaja es su mantenimiento, su alto coste, que su exposición a ambientes hostiles puede dañarlas y su peso. Por otro lado, su principal ventaja es el agarre de objetos rápidamente y con precisión de forma activa gracias a los sensores en los dedos.

***Imagen 11**. Prótesis eléctrica.*

PRÓTESIS NEUMÁTICA

Las prótesis neumáticas hacen uso de aire a presión obtenido por medio de un compresor. Su ventaja principal es que proporcionan una gran fuerza y rapidez de movimientos, mientras que sus desventajas principales son el tamaño de los dispositivos que se implementan para su control y funcionamiento y que su mantenimiento es costoso y dificultoso.

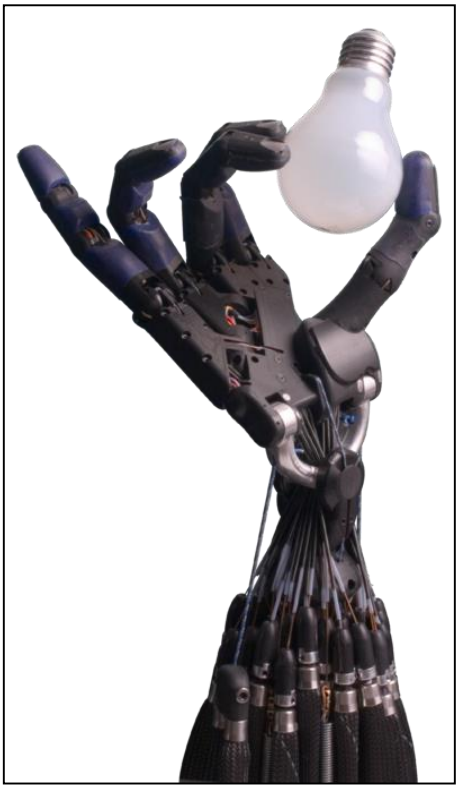


***Imagen 12**. Prótesis neumática.*

PRÓTESIS MIOELÉCTRICAS

Las prótesis mioeléctricas son en la actualidad las de mayor aplicación en el mundo, ya que aúnan estética, precisión y fuerza. Se basan en la obtención de señales musculares (EMG)

mediante el uso de electrodos que permiten la extracción de la señal, la cual es amplificada, procesada y filtrada al control para el manejo de la prótesis. Eliminan el arnés de suspensión, usando una de las siguientes técnicas para mantener la prótesis en el lugar correspondiente: Bloqueo de tejidos blandos-esqueleto o succión. Las desventajas fundamentales son la necesidad de una fuente externa de energía eléctrica para la potencia, su peso y su coste. En la Imagen 13 tenemos las partes de una prótesis mioeléctrica.

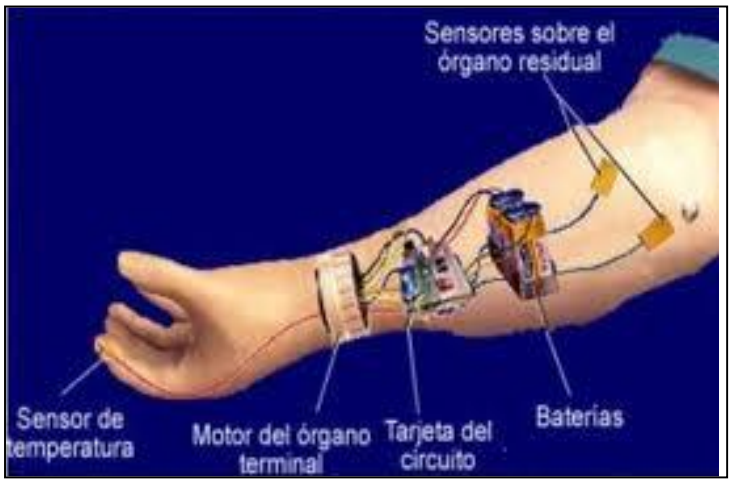


***Imagen 13**. Partes de una prótesis mioeléctrica.*

PRÓTESIS HÍBRIDAS

Las prótesis híbridas son utilizadas por personas que tienen amputaciones desde arriba del codo, ya que combinan la acción del cuerpo con el accionamiento por electricidad. Es muy frecuente en las prótesis híbridas que utilicen un codo accionado mediante el cuerpo y un dispositivo al final controlado en forma mioeléctrica, que puede ser un gancho o una mano.

***Imagen 14**. Prótesis híbrida.*

A modo de conclusión, decir que actualmente los sistemas protésicos mioeléctricos son los que proporcionan el más alto grado de rehabilitación y los más empleados. Esto, añadido al hecho de disponer previamente de un mando mioeléctrico, son los motivos por los cuales el presente TFG se centrará en las manos mioeléctricas y se tratará de crear una prótesis mioeléctrica funcional.

1.5.3. ESTADO DEL ARTE Y PRÓTESIS MÁS DESTACADAS

Para el diseño y construcción de una prótesis de mano se involucran diferentes áreas de la ingeniería, requiriendo procesos tan diversos como: Diseño de mecanismos, mecanizado de materiales, diseño del control, programación de este, diseño de la interfaz entre el hombre y la máquina, e incluso encontrar un diseño para la prótesis similar al de una mano real.

En los últimos años, el hombre, en su incesante búsqueda de soluciones para resolver los problemas que se presentan en el día a día de la sociedad, ha logrado grandes avances en el campo de las prótesis. Se ha conseguido avanzar mucho, facilitando así las condiciones de vida de las personas que necesitan la ayuda de una prótesis de mano, creando prótesis que emulan un gran porcentaje de los movimientos que realiza la mano humana.

A continuación, vamos a realizar un recorrido por las prótesis más destacadas del mercado actual, viendo las principales características de cada una:

PRÓTESIS BIÓNICA I-LIMB

Es la más extendida actualmente. Cuándo hace unos años se comenzó a controlar los dedos independientemente unos de otros, se produjo una revolución en el mundo de las prótesis de mano. Touch Bionics fue la primera empresa en incorporar este avance en una de sus manos en concreto en su prótesis I-Limb.

Esta prótesis está basada en que cada dedo tiene su propio motor, electrónica… lo cual resulta básico para diseñar prótesis con proporciones similares a las de una mano humana. Es conveniente destacar que los amputados de uno o varios dedos podrían emplear prótesis individuales de dedo, en vez de emplear un implante completo de mano.

La prótesis I-Limb tiene en la base de cada dedo una articulación y en la primera articulación un punto de pivotaje, todo esto le aporta a I-Limb 5 grados de libertad. I-Limb es controlada por el usuario mediante sensores mioeléctricos situados en una parte del brazo amputado. El control está diseñado para ser intuitivo y que el paciente pueda abrir y cerrar la mano empleando movimientos similares a los que realizaría al abrir y cerrar la mano real. Esta prótesis cuenta con hasta 14 patrones de agarre.

Sin embargo, a pesar de sus numerosas ventajas, I-Limb no tiene una forma activa de controlar el pulgar, debe ser el usuario, con su otra mano el que modifique la posición de este para adecuarlo a la acción que va a realizar. Otra desventaja que debe ser mencionada es que, al no tener un indicador de fuerza para el usuario, esta prótesis puede ser poco eficiente para tareas de precisión, pudiendo dejar caer objetos por no estar correctamente agarrados.

El coste completo de esta prótesis es de unos 55000€.

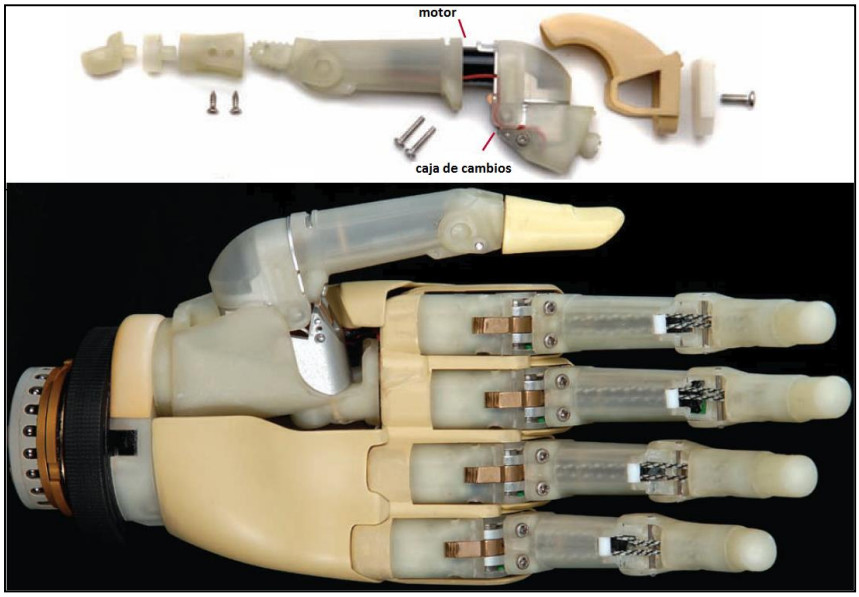


***Imagen 15**. Prótesis biónica I-Limb.*

PRÓTESIS BEBIONIC

Es una prótesis muy similar a la I-Limb, construida por la empresa RSL Steeper con componente mecánicos y funcionalidades muy similares a esta. Esta prótesis tiene como finalidad emular la mayoría de las funciones de I-Limb, pero a un precio significativamente menor, unos 9000€.

***Imagen 16**. Prótesis BeBionic.*

PRÓTESIS MICHELANGELO

La mano Michelangelo, construida por la empresa alemana "Otto Bock" es actualmente la prótesis más avanzada del mercado.

Se caracteriza por realizar una gran variedad de movimientos precisos, gracias al control y a los diferentes mecanismos de fuerzas y velocidades de agarre. Al igual que en las manos "I-Limb" y "BeBionic" los dedos son controlados independientemente, sin embargo, en esta el pulgar y la

muñeca disponen de movilidad, siendo la única prótesis del mercado que dispone de esta funcionalidad.

Internamente está construida con acero y duraluminio (Aleación ligera de aluminio con magnesio, cobre y manganeso que es tan duro como el acero y tiene gran resistencia mecánica) de alta resistencia, mientras que externamente está recubierta por elastómero de silicona. Esta prótesis está caracterizada por ofrecer hasta 6 grados de libertad y ser resistente al agua.

El precio de esta prótesis ronda los 90000€, lo cual hace que, a pesar de ser la prótesis más avanzada del mercado, quede fuera del alcance de muchos de los pacientes que la necesitan.

***Imagen 17**. Prótesis Michelangelo*

PRÓTESIS DEXTRUS

Para poder entender esta prótesis después de haber hablado de las anteriores, es necesario un cambio de enfoque: Ya no nos centraremos en las funcionalidades de las prótesis sino en el precio. Dextrus pretende ser una mano barata y funcional. Alejándose de las prótesis avanzadas de miles de euros, Dextrus intenta crear una mano que sea útil y eficaz, pero al menos precio posible.

Esta mano cuenta con dedos individualmente controlados, y un pulgar que se puede controlar activamente. Además, es una mano que se puede crear en casa, pues sus planos están disponibles en la red y son gratuitos, pudiendo introducir mejoras o nuevas funcionalidades, creándola a nuestro gusto de tamaño y color. También podemos programarla nosotros mismos, para que se adapte mejor a nuestras necesidades concretas. Sin embargo, en caso de no desear, o no tener la capacidad de realizar esa personalización, siempre podremos comprarla por un precio muy reducido, menos de 1000€.

La mano está construida principalmente con plásticos ABS y PLA, imprimidos con una impresora 3D, lo cual hace muy sencillo sustituir piezas en caso de rotura o desgaste. Para manejar la mano se emplean unos sensores mioeléctricos, igual que en otras manos más avanzadas. El control electrónico de la mano se realiza empleando una placa de desarrollo tipo Arduino, lo cual cómo ya se ha dicho ofrece innumerables posibilidades de programación diferentes para esta prótesis.

A pesar de sus limitaciones, en cuanto a sensores, precisión, aspecto… Dextrus ofrece una alternativa barata y muy funcional a las prótesis de miles de euros que hay en el mercado. Es por ello que en este TFG creará una mano Dextrus, se programará partiendo desde cero y se incorporará alguna mejora.

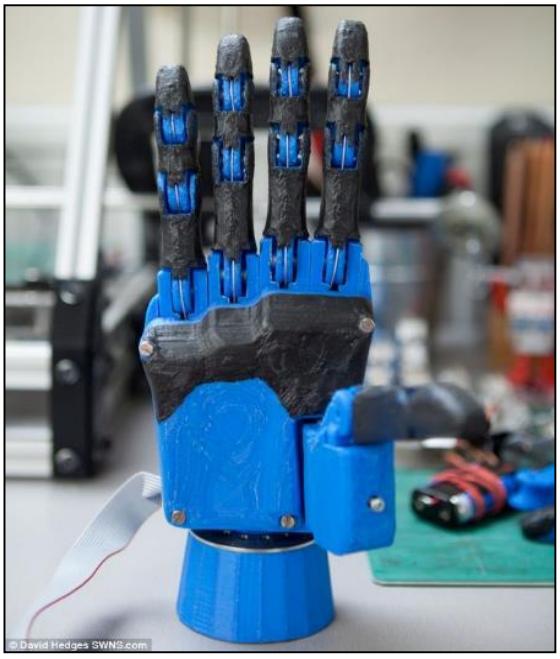


***Imagen 18**. Prótesis Dextrus v1.1.*

MATERIALES INTELIGENTES

En la actualidad el término inteligente se utiliza como una definición para calificar y describir una serie de materiales que presentan la capacidad de cambiar sus propiedades físicas (rigidez, viscosidad, forma, color, etc.) en presencia de un estímulo concreto. Para controlar dicha respuesta de una forma predeterminada, se diseñan mecanismos de control y selección. El tiempo de respuesta es corto y el sistema comienza a regresar a su estado inicial tan pronto como el estímulo cesa. En la Tabla 1 se presentan un grupo de materiales inteligentes empleados en el desarrollo de prótesis de mano.

Materiales con memoria de forma	Aleaciones con memoria de forma Polímeros con memoria de forma Cerámicos con memoria de forma Aleaciones con memoria de forma ferromagnéticos
Materiales electro y magnétoactivos	Materiales electro y magnetereológicos Materiales piezoeléctricos Materiales electro y magnetorestrictivos
Materiales fotocromáticos	**Fotoactivos** Electroluminiscentes Fluorescentes Fosforescentes **Cromoactivos** Fotocromáticos Termocromáticos Electrocromáticos

***Tabla 1**. Materiales inteligentes [3] [4] [5] [6] [7].*

Los alambres musculares, delgados y de alta resistencia mecánica, son elaborados con aleaciones de Níquel y Titanio llamadas “Nitinol”, la cual es una de las aleaciones con memoria más utilizadas. Uno de los aspectos críticos durante la fase de diseño de una prótesis de mano

es el relacionado con la selección de los actuadores y en esta dirección los alambres musculares han mostrado una gran complementariedad con estos últimos [3].

La tendencia a futuro será incrementar la investigación y desarrollo en nuevos materiales que posean un buen comportamiento en cuanto a respuesta, compatibilidad, resistencia y durabilidad. Lo cual, junto al empleo de sistemas de control y accionamientos más potentes, robustos y compactos, posibilitará un mayor acercamiento de las prótesis de mano hacia su equivalente natural.

CONCLUSIONES

La fabricación de prótesis de mano se ha venido desarrollando desde hace siglos atrás, pero en los últimos años con el avance de la tecnología se ha logrado grandes avances con métodos y técnicas de modelamiento y diseño de mecanismos, acompañado de la gran variedad de materiales y control automatizado que juntamente con el interfaz máquina-hombre permiten desarrollar y obtener prótesis que emulen en funcionalidad los movimientos que desempeña la mano del ser humano.

En los recientes años los avances en la biomecánica humana han permitido la fabricación de prótesis de mano con un alto desempeño en simulación de movimientos y con gran apariencia natural, siendo capaz de recibir señales desde el cerebro del ser humano. Sin embargo, los costos de estas prótesis son elevados e inalcanzables para los usuarios, por lo que se opta en utilizar prótesis estéticas, prótesis mecánicas con un gancho o definitivamente aceptar la displacía y no utilizar ningún medio prostético.

La evolución hacia la fabricación de una prótesis que cumpla todas las expectativas tanto en funcionalidad como en costo, ocurrirá posiblemente en un plazo de pocas décadas; para lograr una prótesis de esa magnitud se requerirán de grandes esfuerzos interdisciplinarios de la ingeniería, que junto con la tecnología permitirán superar debilidades y desventajas presentes en las prótesis actuales.

1.6. EL ELECTROMIOGRAMA (EMG) [8]

La información presentada en este apartado ha sido cogida del PFC "Sistema de control de dispositivos externos mediante electromiograma: Aplicación a una mano robótica" desarrollado por Javier García Rodríguez en octubre de 2014 ([8]). Tomaremos este apartado con el fin de evitar volver a realizar un trabajo que ya está hecho en [8], pues los conocimientos de señales mioeléctricas que necesitamos adquirir para ambos trabajos serán los mismos.

Para el correcto diseño y desarrollo del sistema es requisito previo e indispensable el conocimiento y manejo de las señales que vamos a emplear para su funcionamiento, así como el resto de señales fisiológicas que pueden interferir en nuestro dispositivo. Es por ello que debemos adquirir una serie de conocimientos teóricos, desde qué son, cómo se producen las señales y cuáles son sus principales características, hasta cómo debemos adquirir y tratar a las mismas.

la ciencia y la ingeniería han realizado grandes avances en el área biomédica, como por ejemplo, las señales electromiográficas superficiales (EMG) y sus aplicaciones en el control de dispositivos activos como las prótesis mioeléctricas. Este tipo de prótesis son cada vez más aceptadas por personas con amputación de mano, ya que permite a la persona que la utiliza su rehabilitación para desempeñarse activamente en su campo laboral.

1.6.1. SEÑALES BIOELÉCTRICAS

Cada una de las señales bioeléctricas que podemos observar en la superficie corporal, tales como electrocardiograma o electromiograma, tienen su origen en las membranas de las células del sistema con el que están relacionado (muscular, miocardio o neuronal). Podemos asemejar el comportamiento de algunas de ellas a determinados componentes electrónicos, aunque obviamente, presuponiendo una complejidad altamente superior.

Uno de los campos elementales de los que se ocupa la bioelectrónica es el de la interpretación teórica de las señales biológicas. Dicho campo abarca desde el estudio de la generación y propagación de las mismas, hasta la investigación sobre las distintas posibilidades acerca de su adquisición, relacionándolos con conceptos análogos a los empleados en la electrónica básica.

Podemos hacer una gran división cuando hablamos de señales bioeléctricas: las señales captadas a nivel intracelular, potenciales de acción, y las captadas a nivel extracelular, tales como electrocardiograma, electroencefalograma o electromiograma. En ambos casos, la captación de las mismas se lleva a cabo mediante unos dispositivos denominados electrodos (en los cuales profundizaremos posteriormente), los cuales convierten las corrientes iónicas asociadas a la distribución de potencial en los fluidos orgánicos en corrientes electrónicas que pueden ser tratadas de forma adecuada por medio de instrumentación electrónica convencional.

En la tabla de más abajo, obtenida de [9], podemos apreciar las principales similitudes y diferencias entre las señales biológicas más representativas, así como los métodos de medición empleados en cada una de ellas.

Tipo de señal	Amplitud	Banda	Métodos de medida
Potencial de acción	50mV-150mV	0.1Hz-1KHz	Micro electrodos metálicos o de vidrio con puntas de 0.1 a 1µm
ECG (Electrocardiograma)	0.5mV-4mV	0.01Hz-250Hz	Electrodos de superficie en puntos normalizados sobre miembros y torso
EEG (Electroencefalograma)	5µV- 300µV	0.01Hz-150Hz	Electrodos de superficie en puntos normalizados sobre el cuero cabelludo
EMG (Electromiograma)	100µV-5mV	0.01Hz-500Hz	Electrodos de superficie en puntos localizados o electrodos de aguja insertados en músculos

***Tabla 2**. Comparativa de señales bioeléctricas.*

1.6.1.1. MEDIDAS DEL SISTEMA MUSCULAR

La adquisición de medidas en el sistema muscular aplicada a las señales bioeléctricas va a ser uno de los puntos más destacados en el diseño de nuestro sistema, pues adquiere una alta relevancia, para, posteriormente, poder procesar y acondicionar las señales pertinentes. Una señal bioelectrica es la consecuencia de la captación de los potenciales de acción de un conjunto de células que componen parte de un músculo. Dicha recogida se realiza mediante unos electrodos, que describiremos con detalle más adelante.

Los músculos son tejidos con una naturaleza excitable, y como células nerviosas que son, tienen la capacidad de generar potenciales de acción. Podemos asemejar nuestros músculos a un mecanismo capaz de transformar energía química en calor y trabajo. De esta forma, cuando a un músculo, desde una perspectiva unitaria, se le aplica una carga y posteriormente se le estimula eléctricamente con un pulso de breve periodo, éste se contrae. Si se produce una contracción de este tipo, en las que el músculo se comprime bajo la aplicación de una carga constante, se denomina contracción isotónica. Por otro lado, si el tendón del músculo correspondiente se mantiene fijo para impedir el acortamiento, se definirá como contracción isométrica.

La unidad contráctil de la musculatura es la fibra muscular [10]. Se trata de células cilíndricas, de aproximadamente, unos 25 µm de radio, que al recibir un estímulo se contraen desempeñando una fuerza. A su vez, las fibras musculares están compuestas por miofibrillas (haces de filamentos contráctiles de aproximadamente 1 µm de diámetro).

1.6.2. ELECTROMIOGRAMA (EMG)

El primer estudio, con relativa profundidad y rigurosidad, del electromiograma (EMG), fue llevado a cabo por Piper en 1912, quien registró potenciales durante la contracción voluntaria empleando electrodos de superficie y un galvanómetro de hilo.

Posteriormente, en 1922, el médico estadounidense Joseph Erlanger, acompañado de su pupilo, el fisiólogo Herbert Spencer Gasser, consiguieron amplificar las señales eléctricas originadas al estimular una fibra nerviosa y representarlas gráficamente en un osciloscopio de rayos catódicos. De esta manera, descubrieron que las fibras nerviosas conducen impulsos a diferentes velocidades según su espesor y que cada una posee su propio umbral de excitabilidad, trabajo galardonado con el premio Nobel de Medicina en 1944.

En 1929 Adrián y Broke introdujeron el electrodo concéntrico de aguja que hizo posible, conjuntamente con el osciloscopio de rayos catódicos y los amplificadores electrónicos, el estudio de potenciales de acción de unidades motrices y de fibras únicas.

Durante las siguientes décadas y debido a las continuas mejoras de los aparatos de EMG, la electromiografía superficial fue utilizada cada vez más para el estudio de la función del músculo.

A finales de los años 50 y principios de los 60, George Whatmore utilizó la electromiografía para aumentar la técnica de relajación progresiva. Había nacido el biofeedback, utilizado en el tratamiento de algunos trastornos y enfermedades como el dolor de cabeza, el asma, la hipertensión, el estrés, la ansiedad, la úlcera, etc. Los sensores se quedan en contacto con los músculos cuya tensión se quiere medir y controlar. Estos músculos a su vez transmiten unas señales eléctricas que son absorbidas por el aparato de biofeedback que, a modo de respuesta, envía señales audibles y visuales para que la persona conozca su nivel de tensión muscular y pueda aprender a controlarla, obteniendo así un alivio de los síntomas.

1.6.2.1. FUNDAMENTOS FISIOLÓGICOS

La electromiografía es el estudio de la actividad eléctrica de los músculos. Resulta de una gran utilidad pues nos proporciona información muy útil sobre su estado fisiológico y el de los nervios que los activan. Por ejemplo, en nuestro caso, nos permite tras su correcta adquisición y pertinente procesado, contralar un dispositivo externo.

La membrana de las células excitables se encuentra polarizada, siendo el interior de la célula negativo con respecto al exterior. En la célula muscular, con un electrodo situado en el interior de la fibra y otro en el exterior se puede detectar una diferencia de potencial de reposo de unos 90 mV. Este potencial es producido por diferencias existentes en la concentración de diversos iones (Na+, K+, Ca++, Cl-, etc.). Además, obedeciendo a señales procedentes de otras células, la fibra muscular puede sufrir despolarizaciones transitorias (potenciales de acción) que determinan la actividad de la maquinaria contráctil de la fibra. En el músculo podemos distinguir dos tipos de unidades, las anatómicas y las funcionales. La unidad anatómica es la llamada fibra muscular y la unidad funcional es la unidad motora.

Una unidad motora es un grupo de fibras musculares inervado por una única motoneurona de la médula espinal o de un núcleo motor del tallo cerebral. Este concepto fue introducido por Liddell y Sherrington y comprende una motoneurona, su axón, las ramificaciones de éste y el conjunto de fibras musculares sobre los que estos hacen contacto sináptico. Si la motoneurona sufre una despolarización, ésta recorre todo el axón hasta las terminaciones sinápticas y provoca la despolarización, casi sincrónica, en todo el conjunto de fibras musculares de la unidad motora.

La unidad contráctil de la musculatura del esqueleto es la fibra muscular, que es una célula cilíndrica de unos 50 pm de diámetro, que al ser estimulada se contrae desarrollando fuerza. Un músculo consiste en haces paralelos de fibras musculares. La activación de cada fibra muscular se hace a través del axón de la fibra nerviosa motriz que la inerva. Según la posición y la función del músculo, el número de fibras musculares inervadas por un mismo axón puede variar entre uno o más de mil.

El conjunto formado por la célula nerviosa motriz en la espina dorsal, su axón y las fibras musculares que éste inerva constituye la unidad funcional básica del sistema muscular y se conoce por unidad motora (UM) tal y como podemos apreciar en la Imagen 19.

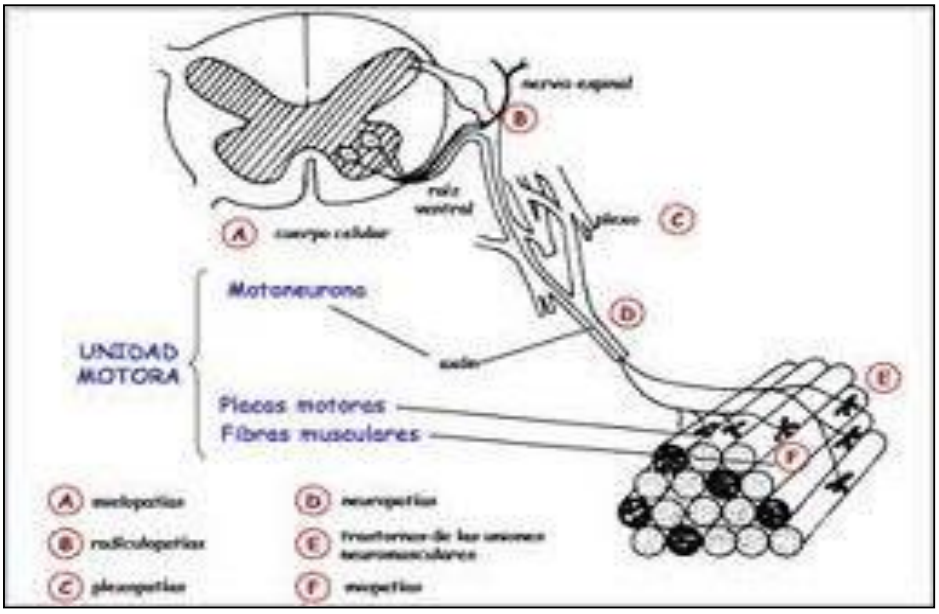


***Imagen 19**. Diagrama de unidad motora.*

Cuando el potencial de acción del nervio alcanza el punto en el que éste se une con el músculo, se libera una cierta cantidad de un transmisor químico (la acetilcolina) que origina la despolarización local de la membrana de la fibra muscular. El transmisor es rápidamente neutralizado por una sustancia denominada estereato de acetilcolina, quedando la unión mioneuronal libre para una nueva excitación. Al potencial complejo que originan las fibras de una UM se le conoce por potencial de la unidad motriz (PUM) y es la suma de los potenciales de acción de las distintas fibras de la UM (potenciales de acción cuasi-sincrónicos en UM normales).

Con todo esto, un músculo puede considerarse como un conjunto de unidades motoras dispuestas en paralelo, entre las cuales se encuentran otras fibras musculares modificadas llamadas Husos musculares que contienen elementos sensoriales que perciben las tracciones en el músculo y sirven para el servocontrol de la posición del músculo, como podemos apreciar en la Imagen 20.

El significado de una orden de excitación dirigida a un músculo es que un número mayor o menor de unidades motoras recibirán una orden de despolarización. Una orden motora puede ser mínima, si solo ordena actividad a una única unidad motora, o máxima, si ordena la contracción completa del músculo. La actividad de una unidad motora es el elemento individualizable mínimo de la contracción muscular. En el electromiograma (EMG) se registra la actividad del músculo y en él se puede distinguir la activación de sus unidades motoras, las variaciones características de estas activaciones y las relaciones de unas unidades con otras.

Se comprende que el número de fibras musculares que contiene cada unidad motora determina la finura o la delicadeza de los movimientos que puede ejecutar. Este número de unidades recibe el nombre de «tasa de inervación» y cuanto menor sea (es decir, muchas motoneuronas y pocas fibras musculares) más flexibilidad motora tendrá el músculo. Por lo tanto, la fuerza de la contracción muscular se gradúa controlando el número de axones que se estimulan y la frecuencia de estimulación de cada axón.

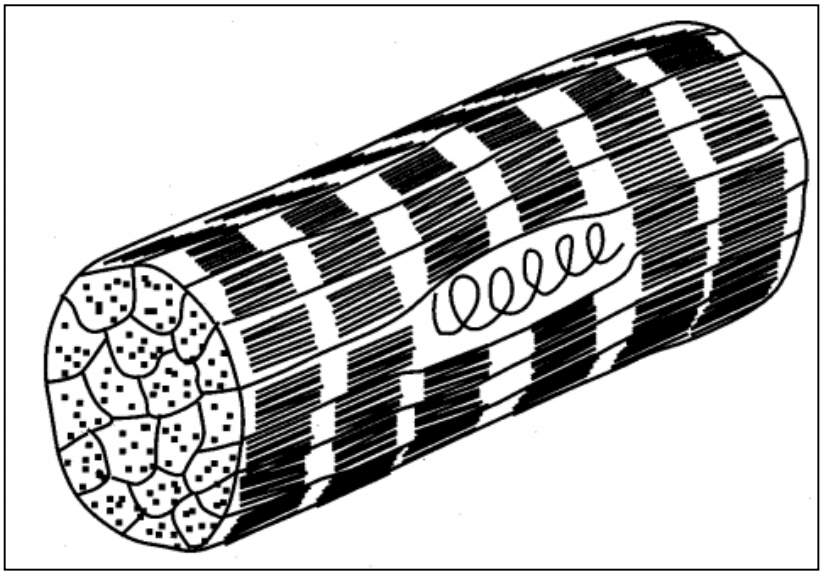


***Imagen 20**. Segmento de un haz de fibrillas musculares mostrando un huso con la terminación sensitiva anuloespinal.*

Cada unidad motora ocupa un territorio en el cual es posible registrar su actividad. Este territorio es algo mayor que el que físicamente ocupan sus fibras. De hecho, las unidades no se agavillan unas junto a otras, ni hay una frontera definida entre ellas; por el contrario, sus fibras o grupos de fibras se entrelazan con las de unidades vecinas de tal forma que en una reducida sección transversal de músculo conviven varias unidades motoras. En líneas generales, se puede afirmar que una unidad motora de un miembro superior se corresponde con un área de unos 5- 7 mm de radio. En los miembros inferiores son 7 -10 mm. Estos valores se han obtenido por medios electrofisiológicos. [11]

1.6.2.2. CARACTERÍSTICAS DEL EMG

Generalmente, las señales bioeléctricas se caracterizan por su reducida amplitud y, por lo tanto, suelen aparecer corrompidas con interferencias, como pueden ser las procedentes de la red eléctrica (50 Hz). Por este motivo, dichas señales han de ser amplificadas hasta niveles adecuados para su correcta adquisición y/o posterior procesamiento. La etapa de amplificación es una de las partes más sensible del diseño. La razón de ello es que en dicha etapa debemos de conseguir amplificar la amplitud de la señal y de manera simultánea rehusar las señales no deseadas.

Como se puede intuir, en ocasiones, el nivel de las señales de interferencia es de varios órdenes de magnitud superior a los de las señales que son objeto de estudio, en este caso, las señales electromiográficas. En las pruebas realizadas, se pudo comprobar cómo, en la adquisición de las señales, la señal de electromiograma, siempre por debajo de 1 mV, podía estar mezclada con una señal de varios voltios a frecuencias próximas o superiores a los 50 Hz. Todo ello nos incrementa la dificultad, y nos presenta un hándicap, que es cómo eliminar estas interferencias mediante filtros respetando al máximo la componente original de la señal.

Si trabajásemos en un entorno ideal, la naturaleza de las señales sería voluntaria, es decir, cuando de manera consciente se efectuará la acción de contracción o relajamiento, existiría una respuesta muscular. Como se puede esperar, en un ámbito práctico, esto no sucede así. En la Imagen 21 de más abajo, existen una gran cantidad de señales de una naturaleza espontánea

que no se pueden predecir de modo alguno, que afectan en la etapa de adquisición de las señales y que hay que tenerlo en cuenta en la fase de procesado de las mismas.

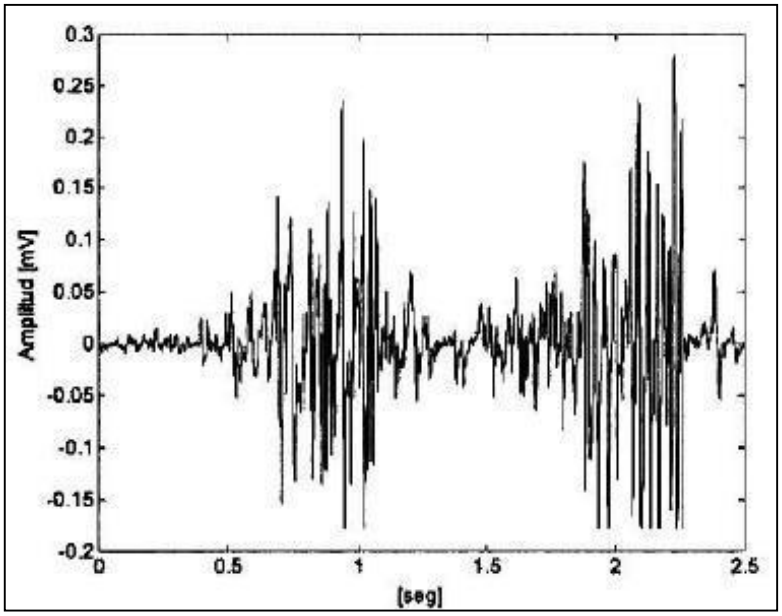


***Imagen 21**. Señal EMG durante contracciones intermitentes del músculo extensor de la muñeca.*

La actividad de inserción es un tipo de actividad espontánea que se origina al colocar el electrodo en el músculo y que en sujetos sanos permanece presente aproximadamente el tiempo en que está en movimiento el electrodo.

Los potenciales de fibrilación son de pequeña amplitud (80 a 50 mV) con una duración de apenas unos milisegundos y con una frecuencia de repetición entre 2 y 10 veces por segundo. Son una consecuencia de la contracción espontánea de las fibras musculares y son característicos de músculos en los que la continuidad entre el axón motor y la fibra muscular se ha interrumpido.

Las fasciculaciones son contracciones espontáneas de fibras musculares o unidades motrices lo suficientemente potentes para producir una contracción visible del músculo, pero sin que la articulación se mueva. Pueden presentarse en sujetos sanos y en enfermedades degenerativas muy graves de las neuronas motrices, lo que hace difícil el diagnóstico.

Existen también otros tipos de actividad espontánea como pueden ser: la respuesta miotónica, los calambres, los espasmos musculares, etc. No se entrará en detalle ya que a efectos prácticos no es el tema principal de la presente sección.

1.6.3. ELECTRODOS

Las señales EMG, son colectadas, típicamente, mediante electrodos bipolares de superficie, ubicados sobre la piel. Éstas han sido utilizadas para el control de prótesis de miembros superiores desde 1948 [12]. Estas señales proveen información sobre la actividad neuromuscular que las origina, siendo esencial esta información en: diagnóstico clínico, rehabilitación y como fuente de control para dispositivos activos y esquemas de estimulación eléctrica funcional [13]. Las señales EMGS son generadas por la contracción muscular, por lo que su adquisición requiere de una correcta identificación de las regiones musculares comprometidas en la ejecución de los movimientos a clasificar. Las señales recogidas serán demasiado débiles, por lo que se hace necesario un procesamiento previo de filtraje y

amplificación antes de su análisis. Así mismo, según la complejidad del dispositivo externo será necesario disponer de un mayor o menor número de canales o electrodos de recolección. En nuestro caso, trabajaremos con dos canales y cinco electrodos, dos para cada uno de los canales y el restante será el correspondiente a la masa del sistema.

En este campo, las últimas dos décadas se han conseguido avances que priman el carácter no invasivo de las medidas. Es decir, se ha trabajado en realizar medidas con interferencias mínimas sobre el proceso biológico en estudio. Como se ha comentado anteriormente, las señales bioeléctricas se captan, tanto a nivel intracelular como a nivel extracelular mediante dispositivos denominados electrodos que convierten las corrientes iónicas asociadas a la distribución de potencial en los fluidos orgánicos en corrientes electrónicas que pueden ser tratadas en forma adecuada por medio de instrumentación electrónica convencional.

Para la captación de las señales bioeléctricas extracelulares, los electrodos que se utilizan son macroscópicos y habitualmente están formados por placas metálicas que entran en contacto con la piel directamente o a través de un gel conductor o bien en agujas que se insertan en el tejido en estudio. La tendencia actual es a integrar en los propios electrodos la circuitería electrónica de amplificación y acondicionamiento previo de la seña en forma de pequeños circuitos analógicos diseñados ad-hoc. Existen dos grandes tipos de electrodos habitualmente utilizados para la captación de señales EMG, electrodos de aguja y electrodos de superficie.

1.6.3.1. ELECTRODOS DE AGUJA

Los electrodos de aguja son principalmente utilizados en electromiografía clínica ya que son invasivos y es recomendable que sean manipulados por personal cualificado. Consisten en una cánula similar a una aguja hipodérmica típicamente fabricada de platino debido a que cuando forma aleación con una pequeña cantidad de iridio (entre un 10% y un 20%) se convierte en un material muy fuerte y difícil de doblar o fracturar. En su interior se pueden encontrar uno o dos hilos dependiendo si el electrodo es unipolar o bipolar, tal y como podemos ver en la Imagen 22.

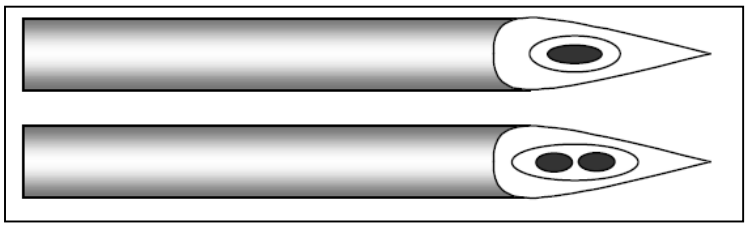


***Imagen 22**. Diferentes tipos de electrodos de aguja.*

1.6.3.2. ELECTRODOS DE SUPERFICIE

En contraposición a los electrodos de aguja, los electrodos de superficie no son invasivos y pueden ser utilizados sin necesidad de ningún tipo de práctica o cualificación especial. En las últimas décadas se han desarrollado un gran número de diferentes tipos de electrodos de superficie. Los más comunes son los electrodos de lámina de metal, que consisten básicamente en una lámina de metal en contacto directo con la piel, provistos con un adhesivo para facilitar su fijación a la superficie cutánea. Este tipo de electrodos tiene una desventaja y es que aparecerán ciertas interferencias a la hora de registrar las señales debidas al movimiento del electrodo. En la Imagen 23 podemos observar diferentes tipos de electrodos de lámina de metal.

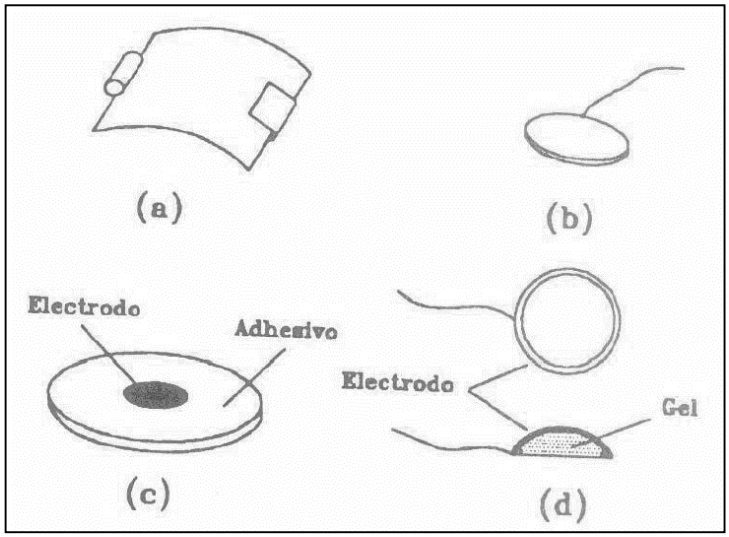


***Imagen 23**. Diferentes tipos de electrodos de lámina de metal.*

Con el objetivo de paliar dichas interferencias, se emplean los electrodos flotantes, los cuales sitúan el metal en el fondo de una cavidad rodeado de electrolito conductor no llegando a estar en contacto directo con la piel, de esta forma se minimiza el movimiento relativo entre el electrodo y el electrolito. En la Imagen 24 podemos ver la estructura de un electrodo de este tipo.

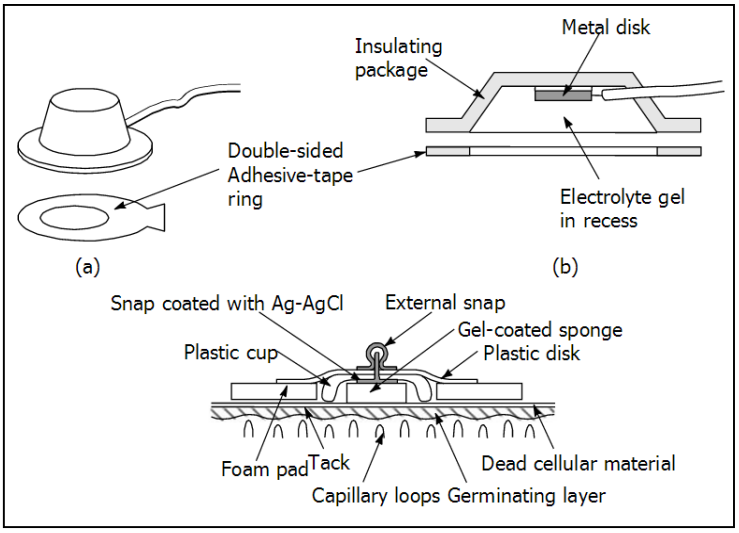


***Imagen 24**. Descripción electrodos de superficie.*

Este tipo de electrodo, presenta un potencial de contacto, inmersos en una solución salina, mucho más estable que el observado en los electrodos fabricados únicamente con plata. Además, son muy sencillos de fabricar por procedimientos electrolíticos a partir de un electrodo de plata sumergido en una solución que contenga iones Cl-.

También es importante mencionar que existen una serie de adversidades adicionales provocadas por el uso de electrodos de Ag/AgCl. Una de ellas es que pueden perder sus propiedades por envejecimiento debido a que la capa de cloro es fotosensible y se deteriora por efecto de la luz. También son muy utilizados los electrodos de acero inoxidable. Dicho material posee buena conductividad eléctrica, es fuerte mecánicamente, no se corroe fácilmente y no es tóxico para los tejidos vivos en su utilización como electrodo de registro. Tiene la ventaja de ser

un material considerablemente más barato que los metales nobles. Por ello cada vez viene siendo más utilizado llegándose incluso a extender su uso para el registro de potenciales internos. Este tipo de electrodos que acabamos de detallar, es el elegido para emplear en nuestro sistema.

Cuando las señales bioeléctricas se registran en la superficie de la piel, se debe considerar, además de la interfase existentes entre el sistema electrodo-electrolito, la piel propiamente dicha. La piel está compuesta por 2 capas, la dermis y la epidermis. Dada su composición, la capa interna de la piel: la dermis, presenta una baja resistencia al paso de la corriente eléctrica. Pero no se puede decir lo mismo de la capa externa de la piel. La epidermis está formada por 4 capas o estratos de células irregulares y escamadas. Es sobre esta capa sobre la que se disponen los electrodos; su irregularidad, así como las secreciones de aceite y sudor hacen que la piel en su conjunto presente una alta resistencia al paso de la corriente eléctrica.

La mencionada elevada impedancia dificulta el registro de potenciales bioeléctricos a nivel superficial. Con el objetivo de minimizar el efecto de la impedancia de la piel sobre los registros, se han desarrollado diversas pastas y geles que reciben el nombre genérico de electrolitos. La función de dichos electrolitos es establecer un camino de baja impedancia entre el electrodo situado en la superficie y la capa de la piel que presenta baja resistencia eléctrica, denominada dermis. Además, asegura como se ha comentado anteriormente, que el área total de la piel bajo el electrodo hace un buen contacto eléctrico con el metal del electrodo y que pelos y otras superficies irregulares no disminuyen la superficie de contacto efectiva. Llegado el momento de realizar la adquisición de las señales para su posterior tratamiento, debemos tener en cuenta la impedancia de la interfase electrodo-electrolito-piel. Dichos valores dependen motivos muy diversos, como por ejemplo, grosor y color de la piel.

1.7. NUEVOS MATERIALES: GRAFENO

1.7.1. QUE ES EL GRAFENO

Los nanomateriales han acaparado el interés de la investigación científica de las últimas dos décadas, debido al descubrimiento de propiedades diferentes a las que ofrecen los macromateriales, dando lugar al advenimiento de una nueva rama del saber científico: la nanotecnología. El espectro de posibilidades de su aplicación es de amplitud y versatilidad tal que inauguran una verdadera revolución tecnológica. [14]

Nanomateriales es el nombre genérico con que se designa a las partículas de una dimensión igual o menor a una millonésima de milímetro. El carbono, por ser el elemento más conocido e intrigante de la Tabla Periódica, es el que ha focalizado en mayor grado la atención científica a este respecto. El carbono tiene varias formas alotrópicas. Alotropía, en química, es la existencia, especialmente en el estado sólido, de dos o más estructuras moleculares o cristalinas de un elemento [14]. Los alótropos del carbono pueden ser:

- Tridimensionales – Diamante, grafito…
- Bidimensionales – Grafeno.
- Monodimensionales – Nanotubos.
- Cero dimensionales –Fullerenos.

Esta alotropía tan extensa se debe a la capacidad de los átomos de carbono para formar redes muy complicadas y numerosas diversas estructuras.

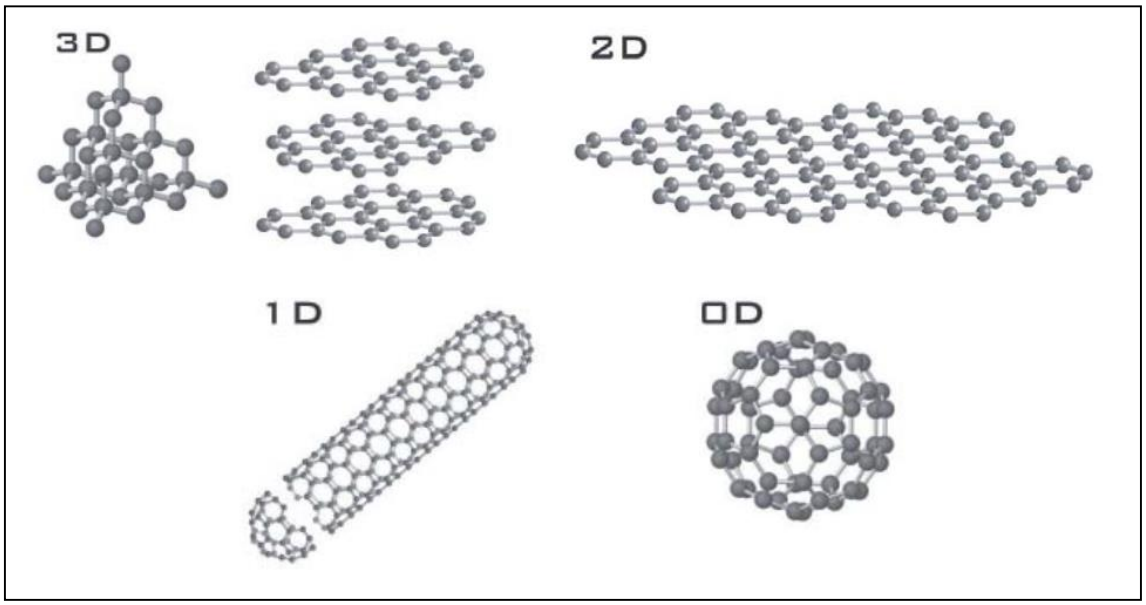


***Imagen 25**.Formas alotrópicas del carbono. Diamante y grafito(3D), grafeno(2D), nanotubos(1D) y fullerenos (0D).*

Por lo tanto, se puede decir que el grafeno es una capa monoatómica de átomos de carbono. En el año 2010 el premio Nobel de Física fue concedido a Andre K. Geim y Konstantin Novoselov por sus experimentos relacionados con este material bidimensional. Con su trabajo

respondieron a la pregunta de si era posible la existencia de cristales bidimensionales de tamaño macroscópico en nuestro mundo tridimensional.

A pesar de que era teóricamente imposible, cómo demostraron Piers y Landau en los años 30 del siglo XX, numerosos investigadores habían tratado de crear capas individuales de grafito, pero sin éxito. No fue hasta 2004, cuándo Geim y Novoselov de la Universidad de Manchester, junto con colegas del Instituto de Microelectrónica y Tecnología de Materiales de Chernogolovka, Rusia, informaron que habían encontrado una forma sencilla de obtener una sola capa de grafito. La técnica, que por su extrema sencillez dejó a la gente atónita, consiste en poner un trozo de grafito entre dos capas de cinta adhesiva de celofán. Cuando se separan las cintas se queda pegado en cada una de ellas un fragmento más delgado de grafito. Repitiendo el proceso varias veces, al final se termina teniendo fragmentos de una sola capa atómica de espesor. Este método fue descrito por primera vez en un artículo publicado en la revista Science el 22 de octubre de 2004. [15]

A raíz de esto, comenzó una carrera para descubrir las propiedades del grafeno, produciéndose una explosión en lo que a su estudio se refiere. En la gráfica de abajo podemos ver una evolución del número de artículos publicados con la palabra grafeno en el título entre los años 2000 y 2010. Más allá de lo que muestra esta gráfica, en los últimos años el número de artículos publicados ha ido creciendo a un ritmo sorprendente.

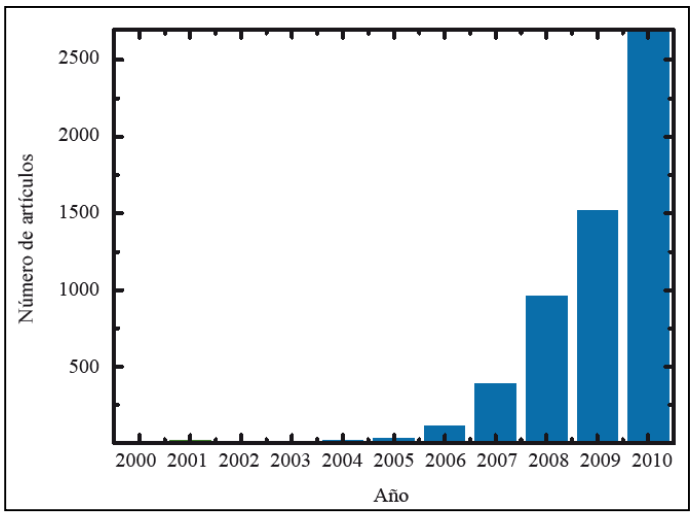


***Imagen 26**. Evolución del número de artículos publicados con la palabra grafeno en el título.*

1.7.2. ESTRUCTURA DEL GRAFENO

El grafeno es una estructura nanométrica, bidimensional, de átomos de carbono fuertemente cohesionados en una superficie uniforme, ligeramente plana, con ondulaciones, de un átomo de espesor, con una apariencia semejante a una capa de panal de abejas por su configuración atómica hexagonal.

De esta configuración o arreglo peculiar se desprenden propiedades electrónicas, mecánicas y químicas excepcionales del grafeno. Tomando un fragmento de la figura anterior, mostramos algunas relaciones importantes que se presentan entre sus átomos de carbono.

Lo que confiere al grafeno una singular importancia -entre otros aspectos- es que puede considerarse como el bloque constructor a partir del cual se forman todos los demás materiales

grafíticos. Si se le envuelve como el forro de un balón, proporciona fullerenos; si se le enrolla cilíndricamente, nanotubos; si se le superpone tridimensionalmente, grafito.

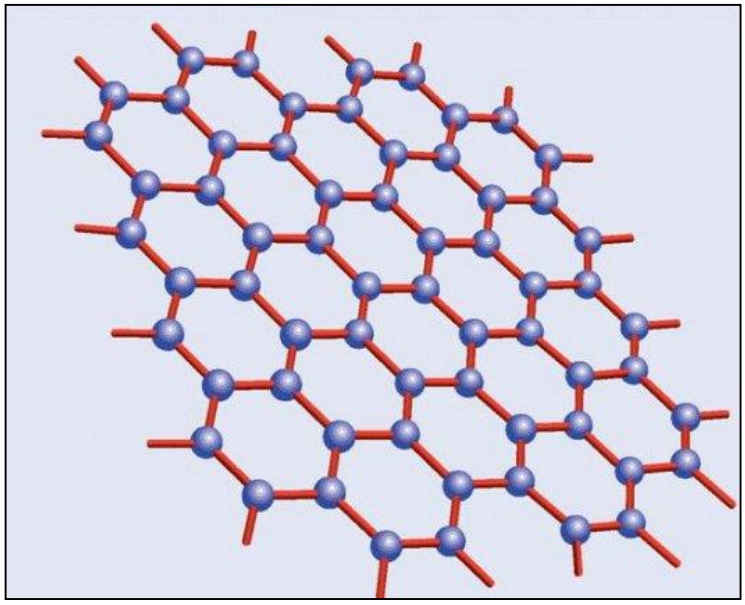


***Imagen 27**. Estructura del grafeno.*

1.7.3. PROPIEDADES

El grafeno tiene muchas propiedades interesantes para la industria: Flexibilidad, resistencia, conductividad… A continuación, veremos brevemente alguna de las propiedades más importante y que más utilidad tienen. [16]

- *Flexibilidad*: El grafeno es muy flexible, gracias a su cualidad de ser una capa monoatómica y a su estructura hexagonal. Tiene la capacidad de aumentar hasta un 20% su tamaño y volver a su forma inicial.
- *Transparencia*: El grafeno tan solo absorbe el 2% de la luz que lo atraviesa. Esto le aporta una gran transparencia haciéndolo prácticamente invisible a simple vista.
- *Resistencia*: El grafeno es el material más resistente conocido por el hombre, siendo el doble de resistente que el acero. Esta capacidad se da porque cada átomo de grafeno esta unido a otros 3 por enlaces muy fuertes, repitiéndose esta estructura a lo largo de toda la capa de grafeno.
- *Conductividad eléctrica*: Tenemos al cobre por un material muy conductor, sin embargo, el grafeno es incluso más conductor que el cobre. Esto es porque gracias a su estructura los electrones se pueden mover increíblemente rápido a través de él.
- *Conductividad térmica*: El grafeno es el material con mayor conductividad térmica conocido por el hombre.
- *Ligereza*: El hecho de que el grafeno tenga un grosor monoatómico le proporciona una enorme ligereza. En la imagen inferior se puede observar la extrema ligereza del grafeno en forma de aerogel de grafeno, material desarrollado en un laboratorio de la Universidad de Zhejiang.

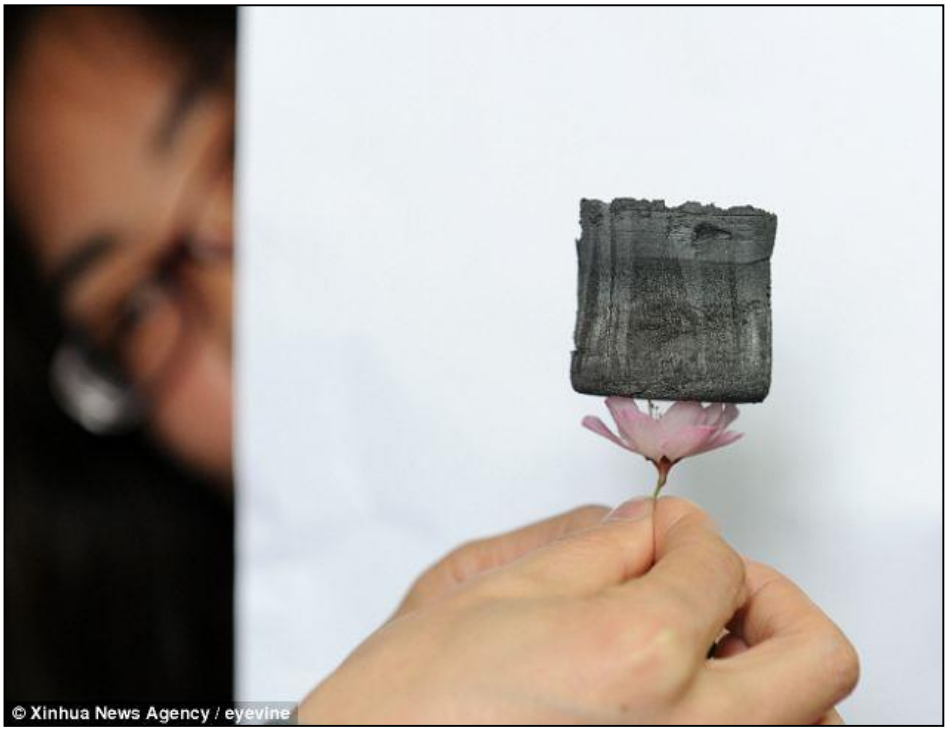


***Imagen 28**. Pieza de aerogel de grafeno, situada sobre una flor de cerezo.*

1.7.4. APLICACIONES

De sus innovadoras propiedades podemos extraer todo un mundo de ámbitos de aplicación del grafeno. Empezando por el ámbito empresarial, donde muchas empresas pueden tener intereses, hasta el campo de la investigación, donde sus propiedades punteras pueden ayudar a lograr avances muy importantes.

A continuación, nombraremos algunas de las aplicaciones más interesantes de este material:

- *Industria textil*: Las propiedades de flexibilidad y conducción térmica del grafeno nos pueden ayudar a diseñar ropa que incorpore tecnología. Ejemplos de ello serían crear ropa que reaccione a cambios de temperatura del exterior, o ropa deportiva que conduzca el calor lejos del cuerpo, para aumentar el rendimiento durante la práctica deportiva.
- *Industria energética*: Gracias a sus propiedades capacitivas, el grafeno permitirá construir baterías de larga duración. Además, hará que la industria de las energías renovables se vea impulsada, ya que al recubrir placas solares con este material la eficiencia de estas se ve aumentada en gran medida.
- *Electrónica*: Este material permitirá la construcción de microchips y transistores de grafeno, elementos imprescindibles en prácticamente cualquier dispositivo electrónico. Estos dispositivos serían mucho más rápidos y energéticamente eficientes que los actuales, construidos con silicio. Además, el grafeno nos abre la puerta al desarrollo de dispositivos electrónicos, cómo ordenadores o smartphones, que puedan ser flexibles y transparentes.
- *Industria automovilística y aeronáutica*: Las características de resistencia y ligereza del grafeno los hace ideales para aplicarlos en la industria automovilística y aeronáutica. Se

podría emplear en aplicar grafeno sobre los chasis de los coches y la estructura de los aviones aumentando así su resistencia, y mejorando la seguridad en estos.

- *Investigación*: La capacidad del grafeno de reaccionar con otros materiales químicos es una propiedad que interesa en gran medida a la comunidad científica. Un ejemplo de esto es el grafano, que surge de la adición de hidrógeno al grafeno, lo que da como resultado un nuevo material con propiedades aislantes.
- *Biomedicina*: En este campo el grafeno es realmente prometedor, permite avances muy interesantes cómo: Focalizar el tratamiento contra el cáncer en una zona concreta, crear implantes neuronales o incluso el desarrollo de implantes musculares y óseos.
- *Otros*: Fuera de los campos anteriores, el grafeno también tiene numerosas aplicaciones interesantes. Ejemplos de estas aplicaciones podrían ser la mejora de la eficiencia de los combustibles al añadirles grafeno y el aumento de la velocidad de algunos procesos industriales de fabricación al emplear grafeno.

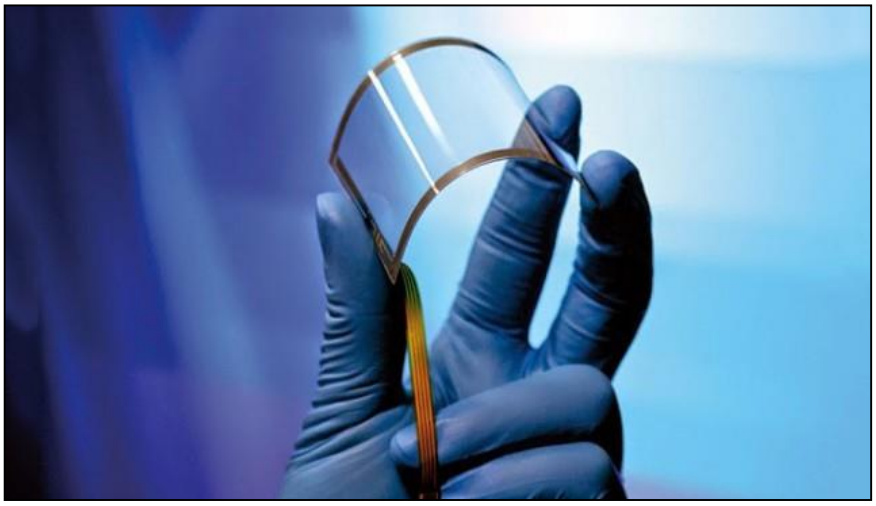


***Imagen 29**. Prototipo de pantalla flexible diseñada con grafeno.*

1.7.5. OXIDO DE GRAFENO

El óxido de grafeno (r-GO) es grafito oxidado en el que se intercalan moléculas de oxígeno entre las capas de carbono, que al reducirse se separan en láminas de pocas capas superpuestas.

El grafito es carbono con una estructura tridimensional con millones de capas de grafeno, que al oxidarse separa las capas de C con moléculas de O, lo que aumenta la separación entre capas y lo convierte en hidrófilo, es decir, soluble en agua. Aprovechando las propiedades hidrófilas del óxido de grafito, se exfolia en agua mediante sonicación resultando una estructura de unas pocas capas superpuestas, conocida como óxido de grafeno.

El óxido de grafeno fue presentado en 2010 en la Universidad de Manchester por Andre Geim, profesor de Física y su equipo (Rahul Nair e Irina Grigorieva).

1.7.5.1. PROPIEDADES Y EJEMPLOS DE APLICACIÓN

- Impermeable a gases y líquidos, pero no al agua. Esta propiedad lo hace idóneo para la destilación del alcohol.
- Propiedades eléctricas: la capacidad de conducir electrones depende de la cantidad de oxidación y del método de síntesis, por lo que el óxido de grafeno altamente oxidado

sería un mal conductor, pero se puede modificar químicamente para desarrollar propiedades y adaptarse a aplicaciones específicas.

- Flexibilidad: dada su estructura atómica, la forma en que se entrelazan las distintas capas lo convierte en un material muy flexible, idóneo para fabricar láminas conductoras, las películas de óxido de grafeno se pueden depositar sobre cualquier sustrato convirtiéndolo en conductor, para electrónica flexible, dispositivos de cristal líquido, sensores químicos...
- Resistencia a la rotura: el óxido de grafeno tiene una resistencia de 55 Newtons por metro. Por poner un ejemplo, una hoja del espesor de una bolsa de plástico soporta un peso de 2.000kg. Se está estudiando su uso para la protección del cuerpo, como los chalecos antibalas.
- Debido al pequeño tamaño de sus poros, se está estudiando su utilización para la obtención de gas natural y separación de gases en general.
- Producción de grafeno: aunque el óxido de grafeno en sí mismo ya cuenta con propiedades que pueden ser desarrolladas para diferentes aplicaciones, la obtención de grafeno a gran escala es una de las más importantes ya que se está estudiando la manera de reducir el coste de producción maximizando la calidad del producto resultante. Existen varios métodos de reducción:
    - Químicamente: mediante reductores como hidrato de hidrazina o borohidruro de sodio. Este último con mejores resultados ya que el hidrato de hidracina es un reductor fuerte, venenoso y deja impurezas en el producto resultante como oxígeno y nitrógeno.
    - Térmicamente
    - Electroquímicamente

1.7.5.2. FUNCIONALIDADES

Por ejemplo, la sustitución de colectores de corriente, basados en metales pesados con películas delgadas, ligeras, flexibles y altamente conductores mejorará aún más la densidad de energía de este tipo de dispositivos. Películas con bloques de construcción de dos dimensiones, como nanoláminas óxido de grafeno reducido (RGO) o grafeno, son particularmente prometedores debido a su bajo umbral de percolación con una alta relación de aspecto, excelente flexibilidad y bajo costo.

Sin embargo, la conductividad eléctrica de estas películas es bajo, típicamente menos de 1000 S / cm. En este trabajo, para el primer informe de tiempo una película RGO con una conductividad eléctrica de hasta 3112 S / cm. Nos lograr una alta conductividad en películas RGO a través de un proceso de recocido inducida corriente eléctrica a alta temperatura de hasta 2750 K en menos de 1 min de tiempo de recocido. Hemos estudiado en detalle el proceso de calentamiento Joule única a temperatura ultra alta.

A través de una combinación de estudios experimentales y computacionales, se investigó el mecanismo fundamental detrás de la formación de una estructura tridimensional altamente conductora compuesta de capas RGO bien conectados. La película RGO altamente conductor con una conductividad alta corriente continua, de bajo espesor (~ 4 m) y baja resistencia de la lámina (0,8 Ω / sq.) Se utilizó como un colector de corriente liviano en baterías de ion-litio.

1.7.5.3. APLICACIONES

Los PCB son contaminantes ambientales altamente estable que se lixivian de productos tales como aislamiento eléctrico, retardantes de llama y papel de copia sin carbón. Altamente soluble en grasa, que se sabe que se acumulan en el cuerpo humano y causar alteraciones endocrinas y neurotoxicty. Como un nanomaterial base de carbono bidimensional, óxido de grafeno tiene aplicaciones potenciales en diversos campos, tales como la electrónica, la biomedicina, la protección del medio ambiente, debido a sus propiedades estructurales, mecánicas, ópticas y electrónicas únicas.

En particular, la gran superficie y la abundancia de grupos funcionales que contienen oxígeno de óxido de grafeno se piensa para que sea un buen absorbente de contaminantes tales como los PCB. Para investigar si el óxido de grafeno podría reducir el impacto de la contaminación de PCB, los investigadores utilizaron células híbridas-hámster humana (A L células) como modelo de alta sensibilidad de detección de mutaciones. Ellos encontraron que el tratamiento previo con óxido de grafeno disminuyó drásticamente la muerte celular y mutación genética causada por la PCB 52.

Por otra parte, los investigadores demostraron que la adherencia y la absorción de nanoláminas óxido de grafeno por un L células causaron la producción de especies reactivas del oxígeno. En conjunto, se encontró que la captación de nanoláminas y el estrés oxidativo para inducir un proceso celular conocido como autofagia, donde las células digieren y productos no deseados reciclados. Los autores especulan que la degradación acelerada de los orgánulos con discapacidad, proteínas y PCB en sí que se desencadenan por pre-tratamiento con óxido de grafeno establece una barrera de protección y la mejora de la capacidad de las células para resistir las sustancias tóxicas.

DESCRIPCIÓN DEL HARDWARE

2.1. PROCESO DE MONTAJE DE LA MANO

Este epígrafe abarca el proceso de construcción de la parte mecánica de la mano, es decir, la mano en sí, sin tener en cuenta el mando mioeléctrico ni el control electrónico con Arduino ni la placa de alimentación…

Este proceso, aunque parece simple, es largo y en algunos momentos muy complicado, ya que abarca desde el montaje de la impresora 3D, hasta el ensamblaje de las distintas piezas para formar finalmente la mano.

A continuación, vamos a ver detalladamente cada uno de los pasos a seguir hasta tener completamente montada la parte mecánica de nuestra mano, para más adelante poder desarrollar el mando mioeléctrico y la parte electrónica, quedando así completamente construida nuestra prótesis mioeléctrica.

El primer paso de todos es el montaje y calibración de la impresora 3D, procesos que quedan perfectamente descritos en los anexos 1 y 2 del presente documento. Una vez montada y calibrada nuestra impresora pasaremos a imprimir las piezas necesarias para el proceso de montaje de nuestra mano.

2.1.1. IMPRESIÓN DE LAS PIEZAS DE LA MANO

Una vez tengamos el archivo .zip que contiene todas las piezas de la Dextrus v1.1 vemos que contiene 4 carpetas: 'Arduino' con el código para la electrónica de la mano, 'Blender' para poder ver en este visor partes completas de cada mano, 'OBJ Files' similar a la anterior pero esta vez siendo archivos .stl y 'STL Files' que es la que usaremos, pues contiene los modelos de cada una de las piezas que necesitaremos imprimir.

En primer lugar, debemos fijarnos en la lista que nos da el creador de Dextrus, para saber cuántas unidades de cada pieza necesitamos. Es posible imprimir varias piezas a la vez, pero no es recomendable, dado que podríamos perder calidad en estas por ahorrarnos unos minutos de impresión. A continuación, pondremos la lista de cuántas unidades necesitaremos de cada pieza, junto con el tiempo que nos llevará esa parte de la impresión:

- Hand-Back.stl - 2 horas → Una unidad → Tiempo total: 2 horas.
- Hand-Back_2.stl - 1 hora → Una unidad → Tiempo total: 1 hora.
- Hand-Palm.stl - 1 hora 40 minutos → Una unidad → Tiempo total: 1 hora 40 minutos.
- Hand-Palm_2.stl - 50 minutos → Una unidad → Tiempo total: 50 minutos.
- Hand-Thumb.stl - 50 minutos → Una unidad → Tiempo total: 50 minutos.
- Hand-Thumb_Cover.stl - 30 minutos → Una unidad → Tiempo total: 30 minutos.
- Thumb-Joint_1.stl - 30 minutos → Una unidad → Tiempo total: 30 minutos.
- Thumb-Joint_2.stl - 20 minutos → Una unidad → Tiempo total: 20 minutos.
- Finger-Joint 1.stl - 30 minutos → 4 unidades → Tiempo total: 2 horas.
- Finger-Joint 2.stl - 20 minutos → 4 unidades → Tiempo total: 1 hora 20 minutos.
- Finger-Joint 3.stl - 15 minutos → 4 unidades → Tiempo total: 1 hora.
- Hand-Motor_Housing.stl - 30 minutos → 4 unidades → Tiempo total: 2 horas.
- Hand-Tensioner.stl - 15 minutos → 5 unidades → Tiempo total: 1 hora 15 minutos.
- Spool-Spool.stl - 20 minutos → 5 unidades → Tiempo total: 1 hora 40 minutos.

Sumando todos los tiempos tenemos un total de 16 horas y 55 minutos de impresión, sin embargo, estos tiempos son orientativos que nos ofrece Cura, y suelen verse incrementados en la realidad un 15%-30% dependiendo del tiempo de impresión de cada pieza, pues cuánto mayor es, más crece el error.

Aunque no nos centraremos en el proceso de impresión, dado que está descrito perfectamente en el anexo 2 de este documento, es necesario indicar la calidad de impresión (Layer height) con la que se deben imprimir las piezas. Todas ellas pueden ser imprimidas con una calidad de 0,25mm, excepto los tornillos (Spool-Spool.stl) que se deberán imprimir con una calidad de 0,1mm o incluso 0,06mm.

Adicionalmente a esto, y aprovechando los conocimientos adquiridos en impresión 3D, se ha imprimido una parte de un sistema 'Eyedrivomatic', el cual será ensamblado por un profesor. Este dispositivo permite realizar el control de una silla de ruedas con los ojos, lo cual puede ser de una gran utilidad para personas que sufren de distrofia muscular. No se construirá el dispositivo entero, sino que sólo trabajaremos con las piezas que se encargan de mover el joystick de la silla. Dichas piezas serán probadas por un profesor en una silla que se está construyendo en el 'Laboratorio de Electrónica y Bioingeniería' y posteriormente serán entregadas a un enfermo de distrofia muscular que ha solicitado ayuda para conseguir dichas piezas.

2.1.2. OBTENCIÓN DE LAS PIEZAS NO IMPRIMIBLES DE LA MANO

Esta parte es la más sencilla, pero la más cara. En primer lugar, debemos encontrar la lista de materiales, la cual se encuentra en el .zip que contiene los modelos 3D, en el documento Bill_Of_Materials.ods. Con la lista de materiales simplemente deberemos buscar en internet los componentes tratando de encontrar el mejor precio para cada uno de ellos, puesto que los enlaces que nos indican en el documento, en muchos casos no es el mejor precio disponible, a causa, sobre todo de los gastos de envío a España. Para la búsqueda lo mejor es buscar en eBay España, en Aliexpress si no tenemos prisa, o directamente en Google.

Para componentes que sea difícil encontrar con exactamente las mismas características, cómo los motores o los muelles, es conveniente realizar la compra en el proveedor que se indica, RS-Online en este caso, pues contar con las mismas piezas que el creador nos aporta una gran fiabilidad, además, este proveedor en concreto ofrece una muy buena relación calidad - precio. Cabe destacar que en este TFG no se han empleado los motores que se recomendaban (417-9706), sino unos con las mismas dimensiones, pero algo más lentos y fuertes (752-1970), dado que en el momento de la compra no había stock de los motores recomendados.

Es muy importante mencionar, antes de comenzar con el proceso de ensamblaje de la mano, que, a la hora de montar los motores en sus carcasas, los dos tornillos que realizan esta deben ceñirse estrictamente a las dimensiones de largura recomendadas (6mm). Para mayor seguridad en este tema, se podría incluso recortar 1 o 2 mm a los tornillos, quedando así con 4 o 5 mm de largo. Esto es así debido a que, al introducir el tornillo en el motor, si este fuera demasiado largo podría presionar y llegar a doblar algún componente interno del motor, y estropearlo. Colocar tornillos demasiado largos nos ha ocasionado una pérdida grande de fuerza en 2 de los motores, hecho que nos ha llevado a la necesidad de comprar 2 nuevos, con el aumento de presupuesto que ello conlleva.

Componente	Cantidad	Precio (€)	Total (€)	Tienda	Enlace
Rodamientos de bolas ranurado	20	0.27	5.37	eBay	http://goo.gl/JQ4Aeb
Tornillo M2x6mm	25	0.10	2.40	eBay	http://goo.gl/Mm578r
Servo 9g SG90	1	3.10	3.10	eBay	http://goo.gl/byyMLD
Tuerca M2	40	0.10	4	Electrónica San Francisco	Tienda física
Tornillo prisionero M2x3mm	10	0.71	7.09	eBay	http://goo.gl/YEczAm
Cable de acero	6	2.47	14.80	TecniCable	http://goo.gl/eBENP1
Virolas dobles	13	0.73	9.50	TecniCable	http://goo.gl/9vcge6
Tornillos M3x45mm	10	0.48	4.78	eBay	http://goo.gl/s0SXMw
Tuerca M3	20	0.10	2	Electrónica San Francisco	Tienda física
Ejes de 3mm	Se tomaron unos ejes de unas impresoras estropeadas y se cortaron.				
Bobina PLA	1	20	20	BQ Store	https://goo.gl/hVc6Vf
Arandela de nylon M3	1	5.8	5.8	eBay	http://goo.gl/jhI13m
Rodamientos de bolas	14	1.72	24.08	RS-Online	http://goo.gl/vz11oQ
Muelle	10	0.86	8.56	RS-Online	http://goo.gl/mMS6i7
Motor	7	22.47	157.29	RS-Online	http://goo.gl/JfFLcO
TOTAL					**268.77€**

Tabla 3. *Componentes empleados en el montaje de la mano.*

En la tabla superior podemos ver la tienda en la que se ha comprado, para este TFG, cada uno de los componentes, con su precio y un enlace para acceder al sitio concreto en el que se puede realizar la compra del componente.

Una vez tengamos en nuestro poder todos los materiales de la lista, podremos pasar al último paso del proceso de montaje de la mano, el ensamblaje de la misma.

2.1.3. ENSAMBLAJE DE LA MANO

Para ensamblar todas las piezas de la mano, debemos seguir un manual, aportado por el creador de Dextrus. En él se indican detalladamente todos los pasos a seguir, herramientas necesarias etc.

Se indica que el tiempo aproximado de montaje son dos horas, sin embargo, este tiempo puede verse incrementado en gran medida por complicaciones que pueden ir surgiendo a lo largo del proceso de ensamblaje, debido a la naturaleza imperfecta de las piezas impresas con la impresora 3D o al hecho de que los materiales empleados no sean exactamente los mismos que se recomiendan. En concreto, en este TFG he empleado entre 15 y 20 horas en el ensamblaje de la mano.

Este proceso consta de 11 pasos, los cuales explicare brevemente a continuación. La descripción que se hará de cada uno de los pasos servirá para entender mejor el proceso, y optimizarlo en la medida de lo posible, sin embargo, para realizar el ensamblaje se deberá recurrir a la guía que nos da el creador de Dextrus [17], ya que esta será infinitamente más completa que este breve resumen de ella.

Cómo veremos, en el manual se comienza montando un dedo y se continúa montando la carcasa del motor y uniendo esta al dedo montado. Después, se monta la carcasa del pulgar. Posteriormente se repiten los tres primeros pasos (Montar el dedo, la carcasa del motor y unir

ambas) para el resto de dedos, ahorrando así un tiempo considerable, al estar ya familiarizados con el proceso a seguir. Por último, se terminarán de ensamblar el resto de las piezas de la mano.

PASOS A REALIZAR PARA EL ENSAMBLAJE DE LA MANO

- **PASO 1 – PREPARAR LOS RODAMIENTOS:** En este paso simplemente se introducirán los ejes en los rodamientos correspondientes. Aunque no se indique, es conveniente introducir también un eje de 6mm en cada uno de los 10 rodamientos ranurados que tendremos, para ahorrar tiempo posteriormente.
- **PASO 2 – ENSAMBLAJE DE UN DEDO:** Este paso consiste en montar uno de los dedos. Es sencillo, pero hay que poner atención al orden en el que se montan las falanges, pues si se hace de forma desordenada, puede que no encajen correctamente las piezas. En este paso es posible que necesitemos la ayuda de un pico loro o unos alicates para prensar las dos partes de cada falange y poder atornillarlas bien, pues al encajar el rodamiento entre ellas, es posible que haya que forzarlas un poco para que se junten completamente.

***Imagen 30**. Montaje de un dedo de la Dextrus v1.1.*

- **PASO 3 – MONTAJE DE LA CARCASA DEL MOTOR**: Aquí encajaremos el motor en su carcasa, y engancharemos los tensores a dicha carcasa. Aparte de esto, también encajaremos los rodamientos ranurados (En los cuales, recordemos, hemos metido los ejes de 6mm) en su lugar correspondiente dentro de los tensores.

***Imagen 31**. Carcasa unida al motor y a los tensores.*

- **PASO 4 – COLOCACION DEL TORNILLO**: Este paso consta de colocar el tornillo impreso, junto con el cable de acero que hará de tendón, en el eje del motor. A la hora de embutir la tuerca dentro del tornillo, es conveniente retirar previamente los restos de plástico que pudiera haber en el hueco de la tuerca con un destornillador, o una herramienta similar.

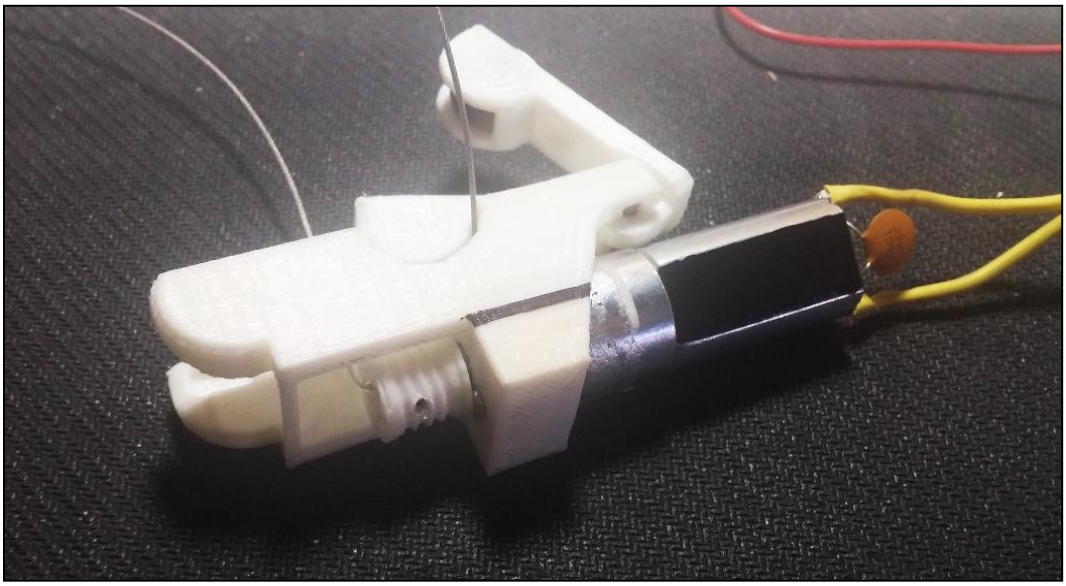


***Imagen 32**. Tornillo ya colocado en el eje del motor.*

- **PASO 5 – COLOCACIÓN DEL TENDÓN**: Ahora colocaremos el tendón, que es nuestro hilo de acero, a través del dedo. Hay que prestar especial atención a cómo pasa el tendón a través de los rodamientos.

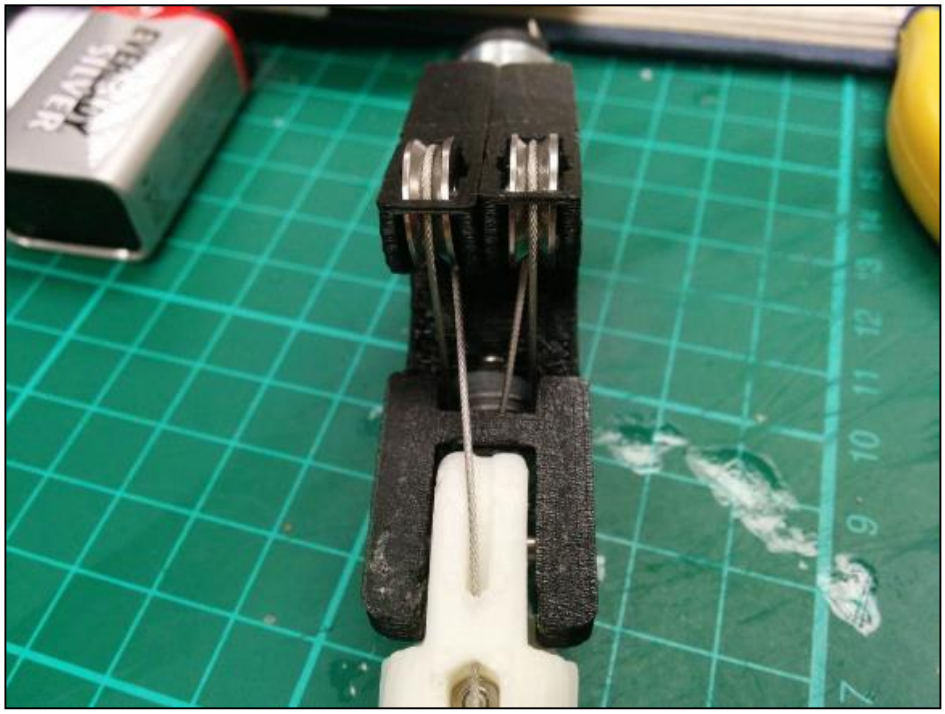


***Imagen 33**. Posición que debe tener el cable sobre los rodamientos.*

- **PASO 6 – TENSADO DEL TENDÓN**: Aquí se tensará el tendón y se colocará una virola al final de este para que quede tensado. En el manual se recomienda apretar los tensores con un trozo de cable, y dando vueltas a este, sin embargo, he visto que es mucho más sencillo apretarles con un tornillo de banco, si se dispone de él, o con un pico de loro. Tras este paso ya tendremos un dedo completamente montado.

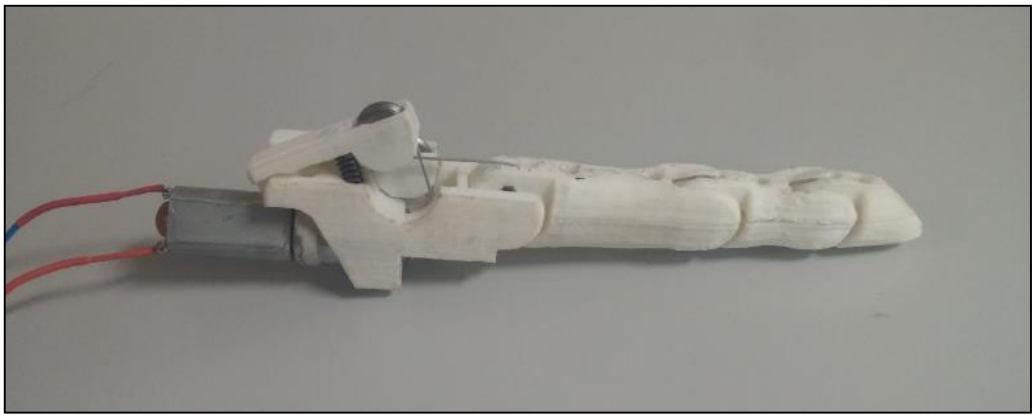


***Imagen 34**. Mecanismo del dedo completamente montado.*

- **PASO 7 – MONTADO DEL RESTO DE DEDOS**: En este paso montaremos el resto de dedos. Repetiremos los pasos del 1 al 6 tres veces más para los otros dedos, y posteriormente los pasos 1 y 2 de nuevo para el pulgar.

***Imagen 35**. Los cinco dedos montados.*

- **PASO 8 – ENSAMBLADO DEL PULGAR A SU CARCASA**: En este paso continuaremos montando el dedo pulgar. Colocaremos el este en su carcasa, y pondremos en ella los tensores. En la carcasa también se colocará el servo que nos permitirá girar el pulgar. En este paso hay varias cosas importantes a mencionar:

- He encontrado más cómodo arrancar las pestañas que tiene el servo para fijarlo y simplemente pegarlo con un pegamento fuerte, que anclarlo con tornillos.
- Al ir a colocar el servomotor en su posición, nos daremos cuenta de que este no encaja correctamente. Esto es debido a un fallo que provoca la impresión de una capa extra de PLA la cual no debería estar ahí. Esta capa hay que arrancarla con cuidado. La forma más eficiente que se ha encontrado es arrancar la parte central con un taladro, y posteriormente con un formón pequeño ir quitando los bordes.
- Para montar los tensores es útil agrandar el agujero en el que se colocan los ejes de los tensores pasando una broca de 2,5mm de lado a lado. Una vez introducida la broca hay que mover esta un poco, para facilitar la colocación de los ejes.

***Imagen 36**. Pulgar ensamblado.*

- **PASO 9 – MONTAJE DE LA PALMA**: Este paso no es demasiado complicado. Simplemente pegaremos las dos partes de la palma de la mano, situaremos el motor de continua en su posición (Habiendo soldado los cables antes), colocaremos el servo, y atornillaremos el brazo del servo en su lugar correspondiente. Se ha visto que es más cómodo atornillar primero el brazo del servo a la palma de la mano y posteriormente acoplar este al servo.

***Imagen 37**. Servo con su brazo ya instalado.*

- **PASO 10 – COLOCACIÓN DEL TENDÓN DEL PULGAR**: En este paso colocaremos los tendones del pulgar, les tensaremos y colocaremos la cobertura del pulgar. A pesar de parecer un paso relativamente sencillo, hay varios problemas que es necesario tener en cuenta:
    - Debido a la capa extra que se imprime en la carcasa del pulgar, los agujeros por los que pasan los tendones de este están tapados. Para destapar estos agujeros es necesario pasar por ellos un taladro con una broca fina y a baja velocidad para que no se funda el plástico.
    - El agujero para el tornillo de la carcasa no coincide con el agujero de la cobertura, debido, una vez más, a la capa extra que se imprime. Esto se soluciona simplemente tomando como referencia el agujero de la cobertura, haciendo un nuevo agujero en la carcasa, para poder fijar esta a la cobertura con un tornillo.

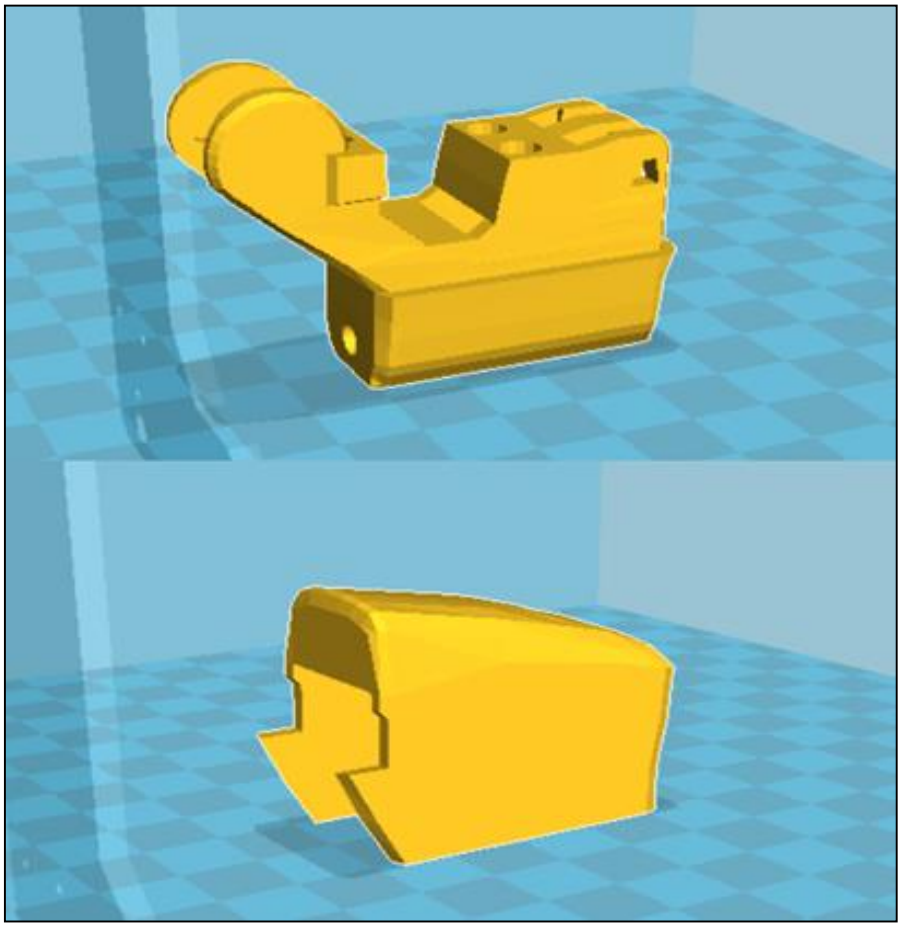


Imagen 38*.Carcasa del pulgar (Arriba) y cobertura de la carcasa del pulgar (Abajo).*

***Imagen 39**. Pulgar con el tendón montado y tensado.*

- **PASO 11 – ENSAMBLANDO LA MANO**: Este es el paso final. En el colocaremos todos los motores en su sitio y cerraremos la mano, juntando las partes 'Palm' y 'Back'. En este paso las piezas encajan un poco mal, por lo que deberemos tener en cuenta las siguientes consideraciones:
    - Para que los motores encajen en la pieza 'Back', es necesario limar los laterales de las carcasas de todos los motores (Menos el del pulgar y el del servo) pues de lo contrario no encajan adecuadamente y no llegarían a hacer tope al final de la pieza.
    - También es necesario limar los laterales de la pieza 'Palm' para que al juntarla con la pieza 'Back' el encaje se produzca correctamente.
    - Si los tendones no se han apretado con la suficiente fuerza (Lo cual es muy posible, debido a la dificultad de hacerlo bien), los tensores de los motores pueden quedar demasiado levantados y tocar en la pieza 'Palm'. Si no se produce demasiado contacto se podría ignorar este hecho. Esto produciría que la palma de la mano no cerrara completamente, quedando aproximadamente 1mm que deberá ser rellenado con algún plástico termorrectráctil o con algún pegamento tipo Araldite. Si el contacto que se produce es demasiado grande, podría llevar a que los tornillos perdieran la rosca y habría que cortar el tendón de ese dedo para colocar un nuevo tendón, esta vez más prieto que el anterior.

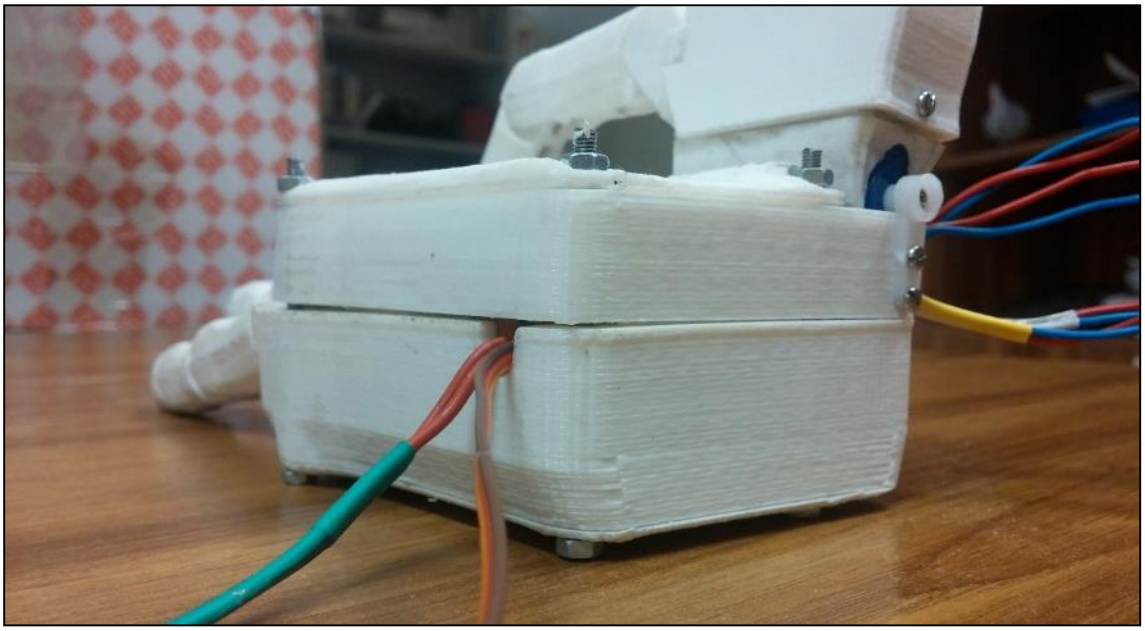


***Imagen 41**. Hueco que debe ser rellenado de alguna manera para que la palma de la mano quede totalmente cerrada.*

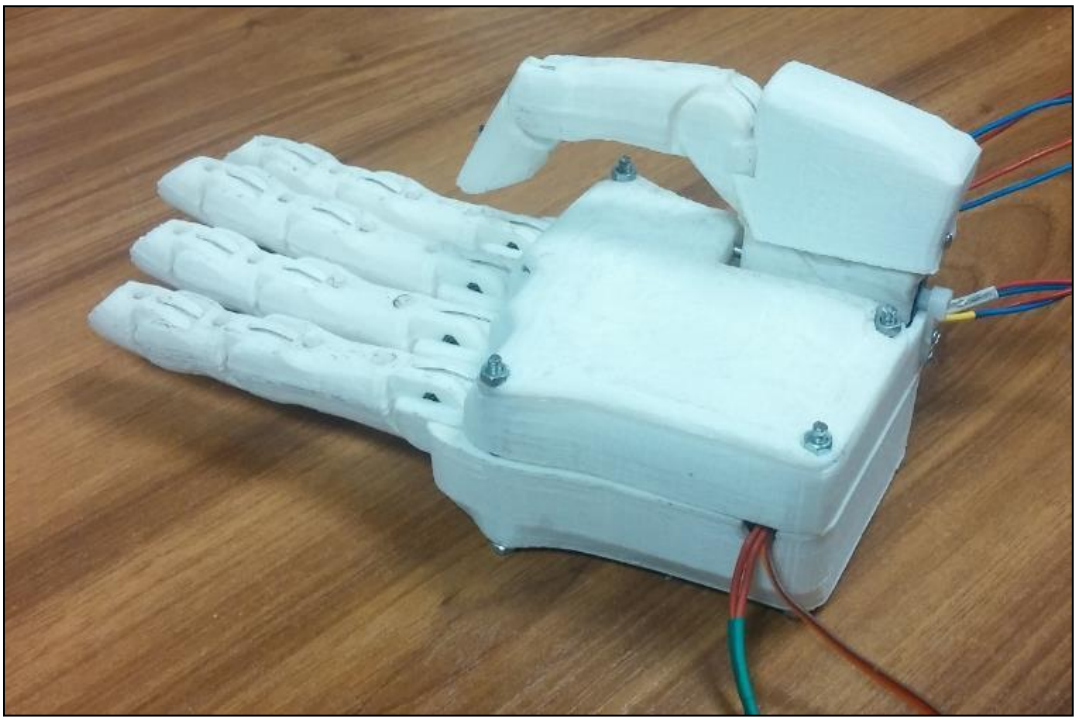


***Imagen 40**. Mano una vez finalizado el proceso de ensamblaje.*

2.2. SENSORIZACIÓN DE LA MANO

Una correcta sensorización de la mano es de vital importancia para un correcto funcionamiento de esta. Los sensores tienen la capacidad de actuar sobre la electrónica, a través del Arduino, para ayudar a automatizar el control de la prótesis. Además, también pueden actuar sobre el usuario, dotándole de propiocepción de la prótesis a través de actuadores.

Las prótesis comerciales cuentan con una gran cantidad de sensores que ayudan al funcionamiento óptimo de la mano. Hay gran cantidad de sensores, de temperatura, de contacto, de fuerza, de deslizamiento… La acción conjunta de todos estos hace que la experiencia de llevar la prótesis se aproxime lo máximo posible a la de poseer una mano real.

Podríamos considerar los sensores de contacto los más importantes de todos, siendo los que más funcionalidad nos restan de no estar presentes, pues al carecer de ellos no podríamos agarrar cosas frágiles ya que las deformaremos con facilidad. Los sensores de temperatura tienen una función puramente de seguridad, para que la mano no se dañe al coger cosas demasiado calientes o que el usuario no se lleve a la boca una bebida que queme, por ejemplo. Los sensores de deslizamiento son muy útiles, pero no son imprescindibles, su funcionalidad es principalmente detectar cuándo algo se está escurriendo de nuestra mano y aumentar la fuerza que se está ejerciendo para evitar que ese algo se caiga al suelo. Finalmente, los sensores de fuerza son los que menos funcionalidad nos añaden, pero aun así son bastantes interesantes, pues nos permiten regular la fuerza que se está ejerciendo con la prótesis.

El sensor de fuerza sería muy sencillo de implementar, mediante alguno de los sensores de presión comerciales que se venden, mientras que el feedback se le podría dar al usuario por medio de una serie de LED. Sin embargo, estos sensores son considerablemente caros, y no hay que perder de vista que uno de los objetivos de este TFG es desarrollar una prótesis funcional, pero con el menor presupuesto posible.

***Imagen 42**. Sensor de fuerza comercial.*

Implantar un sensor de deslizamiento sería muy complicado ya que no existe como tal, y debería hacerse implementando varios sensores de contacto a lo largo de toda la mano, hecho que nos

aumentaría en gran medida la complejidad de la mano y su precio, por lo que también se descartará implementar un sensor de deslizamiento en la mano.

***Imagen 43**. Termistores.*

El sensor de temperatura no es demasiado complicado de implementar. Bastaría con colocar un termistor en uno o varios de los dedos, en los puntos en los que se quiere medir la temperatura. Para el feedback se podría emplear una termorresistencia para la sensación de calor y una célula Peltier para la sensación de frío. Se podría emplear también una célula Peltier para la sensación de calor, pero esta es mucho más cara que una simple resistencia y no debemos perder de vista nuestro objetivo de minimizar el presupuesto. Este sensor es importante y no es demasiado complicado de implementar, sin embargo, por la escasez de tiempo disponible no se va a equipar en nuestra mano. A pesar del coste extra que supondría la implementación de este sensor (Sobre todo a causa de la célula Peltier) sería interesante en un futuro dotar a la mano de un sensor de temperatura, por lo que se dejará constancia de ello en el apartado de líneas de mejora de este TFG, en el cual se darán unas bases para que en el futuro se pueda desarrollar el sensor de temperatura sin la necesidad de partir desde cero.

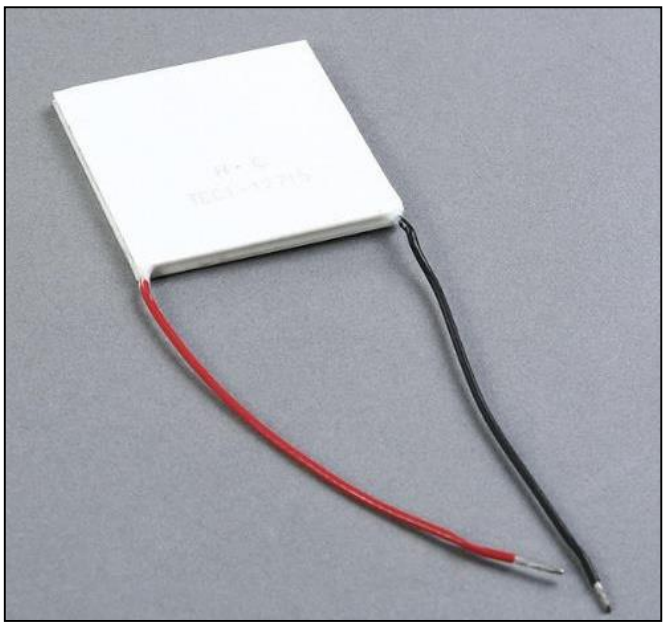


***Imagen 44**. Célula Peltier.*

Por otro lado, el sensor de contacto puede implementarse de distintas formas, con varios niveles de coste y complejidad. Por este motivo se probarán varias opciones de sensor de contacto sobre una placa de pruebas, y se evaluarán los pros y los contras de cada una de las opciones. Una vez escogida la mejor opción, si el tiempo del que se dispone lo permite, se tratará de implementar dicho sensor en nuestra prótesis.

2.2.1. SENSOR DE CONTACTO

Cómo ya se ha dicho la función principal del sensor de contacto es la de detectar cuándo se ha alcanzado el objeto con los dedos de la prótesis para que, de estar agarrando un objeto frágil, se detengan los motores, o se disminuya la fuerza que estos ejercen.

Este sensor no nos tiene que dar un feedback acerca de la magnitud de la fuerza que se está ejerciendo, sino que valdría con que detectase cuándo ha habido contacto para que enviase esta información al Arduino y este tomase las decisiones necesarias respecto al movimiento de la mano.

Hay una gran cantidad de opciones disponibles a la hora de implementar dicho sensor: Switch mecánicos, galgas extensiométricas, sensores bimorfos, e incluso se estudiarán las propiedades del grafeno con vistas a crear un sensor de contacto aprovechándonos de ellas.

A continuación, vamos a ver detalladamente cada una de las opciones de sensor de contacto que se han estudiado para este TFG.

2.2.1.1. MICROPULSADOR MECÁNICO

Es la opción más sencilla y más barata. Estos pulsadores simplemente cierran un circuito al ser pulsados, por lo que bastaría con establecer un pequeño circuito, que al cerrarse indicase al Arduino, a través de una de sus entradas, que se ha producido contacto para que esta actúe consecuentemente.

***Imagen 45**. Micropulsador mecánico.*

El montaje del circuito es muy simple, bastaría con conectar la salida de 5 voltios del Arduino a una de sus entradas a través del pulsador.

La principal ventaja de esta opción es que presenta un comportamiento aceptablemente bueno a pesar de su bajo coste y su gran sencillez. Por otro lado, presenta las desventajas del tamaño y que la superficie sobre la que debe haber contacto para que sea detectado es muy pequeña. Sin embargo, esto podría solucionarse colocando varios pulsadores en paralelo, cómo en la Imagen 46, de esta forma con que se activase un solo pulsador, ya se detectaría el contacto. Otra forma de superar este problema sería colocando una placa rígida sobre el pulsador aumentando así la superficie sobre la que se detectaría el contacto.

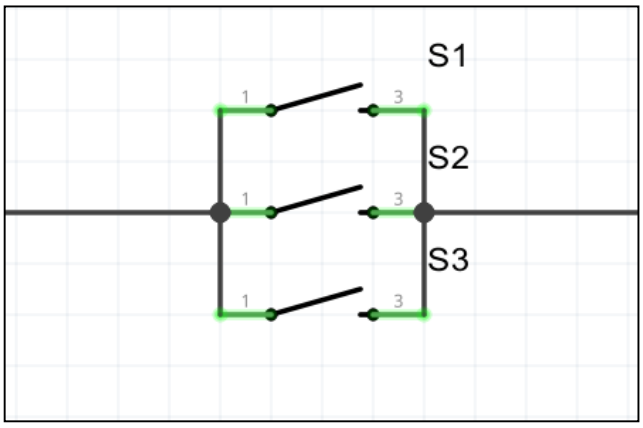


***Imagen 46**. Tres pulsadores colocados en paralelo.*

2.2.1.1.1. PRUEBAS

Tras un montaje simple con 2 micropulsadores en paralelo se comprobó lo que dichos pulsadores ofrecen unos resultados aceptablemente buenos. Queda patente que su principal desventaja es que son demasiado voluminosos y el proceso de integrarlos en la mano puede ser muy delicado, pero a pesar de esto, los micropulsadores siguen siendo una estupenda opción para crear un sensor de contacto.

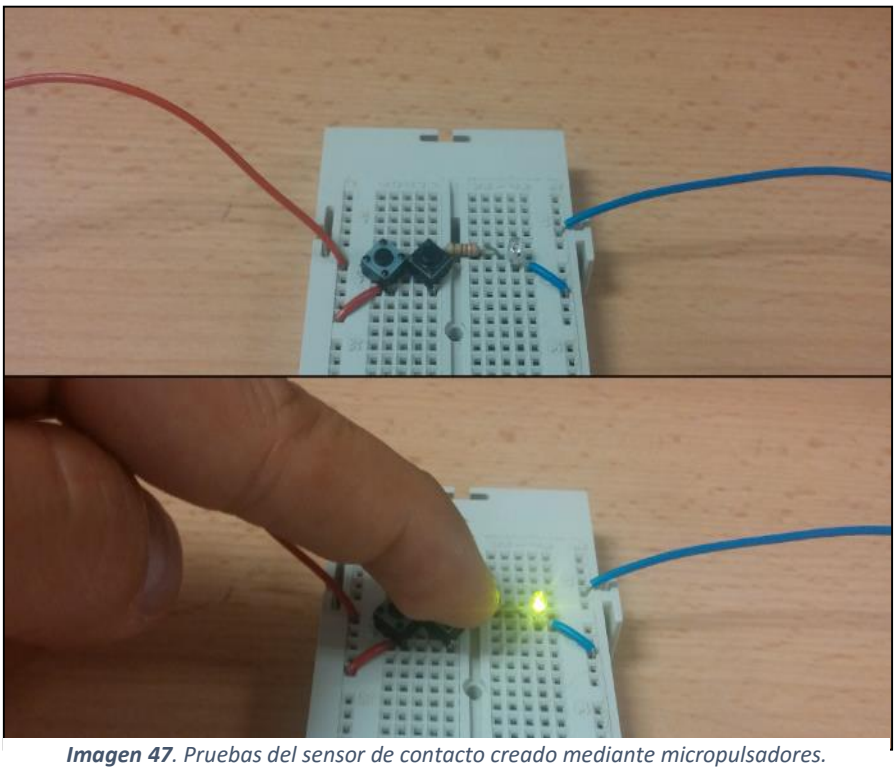


***Imagen 47**. Pruebas del sensor de contacto creado mediante micropulsadores.*

2.2.1.2. SENSOR BIMORFO

Estos sensores son piezas alargadas y estrechas hechas de un material con capacidades piezoeléctricas, normalmente de cerámica piezoeléctrica. Esto hace que al ser presionadas generen picos de tensión, los cuales se pueden usar como señal de entrada al Arduino para indicar cuándo se ha producido contacto.

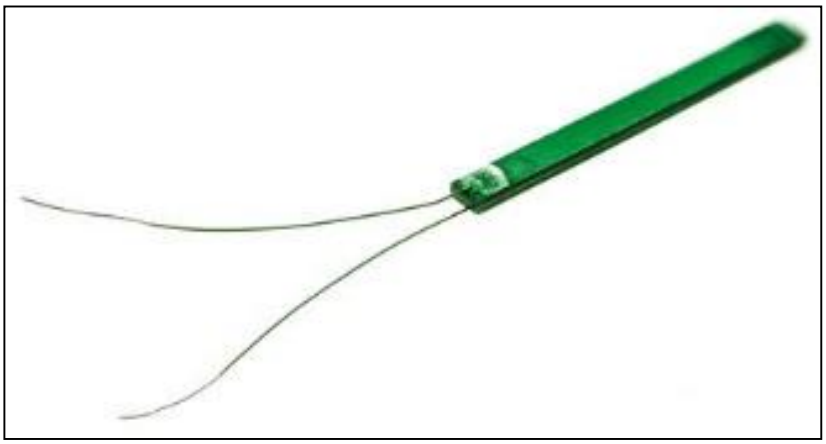


***Imagen 48**. Sensor bimorfo.*

El montaje necesario para este sensor será significativamente más complicado que el empleado en los micropulsadores, debido a la baja duración de los picos de voltaje generados por el sensor. Para aumentar la fiabilidad del sensor y que el Arduino pueda detectar más fácilmente cuando hay contacto, trataremos de que estos picos que genera el sensor bimorfo tengan mayor duración. Esto lo podremos conseguir con un circuito detector de envolvente. Hay varias formas de implementar esto, siendo la más sencilla construir el detector de envolvente con un diodo, una resistencia y un condensador (ver Imagen 49).

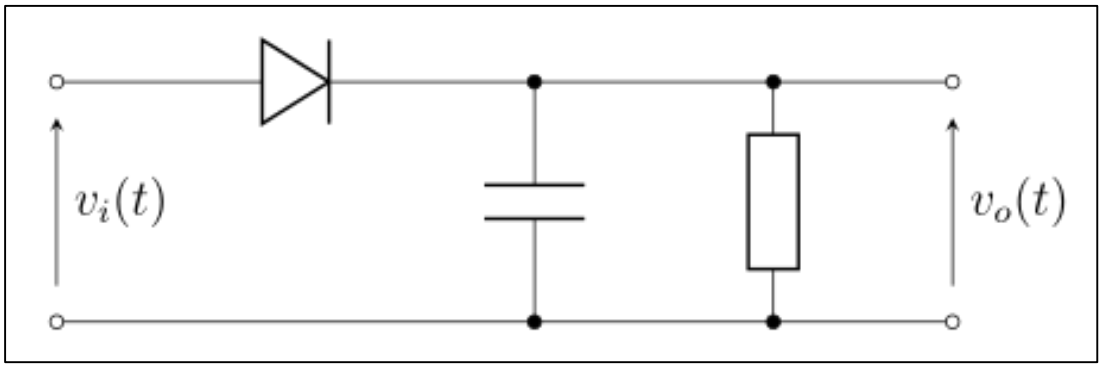


***Imagen 49**. Detector de envolvente sencillo.*

Como principal ventaja de este sensor, destacar su bajo precio y que al ser muy estrecho resulta fácil de integrar en la mano. Por otro también tiene importantes desventajas, las cuales no tienen una fácil solución como para el caso de los micropulsadores. Una de sus desventajas es el hecho de que requiera un circuito de adaptación lo cual nos incrementa la complejidad del cableado de la mano. Por último, la otra desventaja a tener en cuenta es que estos sensores son muy frágiles, por lo que podrían llegar a romperse si la fuerza que ejerce nuestra mano es muy elevada.

2.2.1.2.1. PRUEBAS

Cuando comenzaron las pruebas, se vio que la extrema fragilidad de los cables del sensor le hacían inviable para el uso que le queríamos dar. Al ser los cables sumamente estrechos, la más mínima tensión ejercida sobre ellos, provoca su ruptura, quedando de esta manera el sensor totalmente inutilizable.

Por este motivo se descartó este sensor al poco de comenzar las pruebas, las cuales han quedado reflejadas en el apartado 'Pruebas de funcionamiento'.

2.2.1.3. GALGA EXTENSIOMÉTRICA

Este sensor de contacto podría servirnos también de sensor de fuerza, pues es capaz de medir deformación, presión, carga, par, posición… y se basa en el efecto piezorresistivo, que es la propiedad de ciertos materiales de cambiar el valor nominal de su resistencia eléctrica cuando se le somete a ciertos esfuerzos y se deforman.

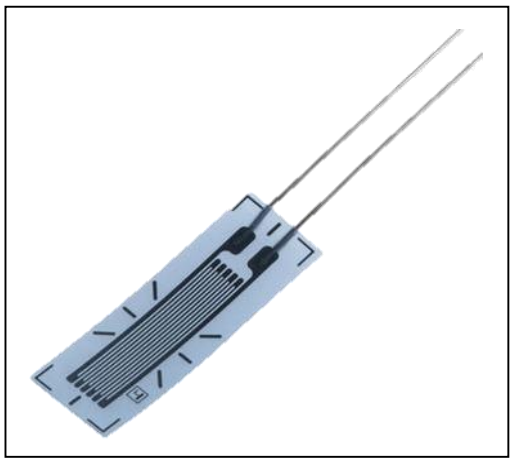


***Imagen 50**. Galga extensiométrica.*

Para poder detectar cuándo hay contacto bastaría con estar midiendo continuamente la resistencia de la galga, y cuando el valor de esta variase, se consideraría que ha habido contacto. Para ello lo primero que debemos hacer es implementar un óhmetro con nuestra placa Arduino. La manera más sencilla de hacer un divisor de tensión con Arduino la vemos en la Imagen 51. Este montaje no es otra cosa que un divisor de tensión, en el que se mide cuánto voltaje cae en la resistencia a medir, para así poder calcular su valor. Este circuito es muy sencillo y presenta un comportamiento considerablemente preciso dada su sencillez. Sin embargo, de ser necesario una mayor precisión podría implementarse uno de los muchos montajes disponibles en internet de óhmetros creados con Arduino con una precisión mayor al sugerido.

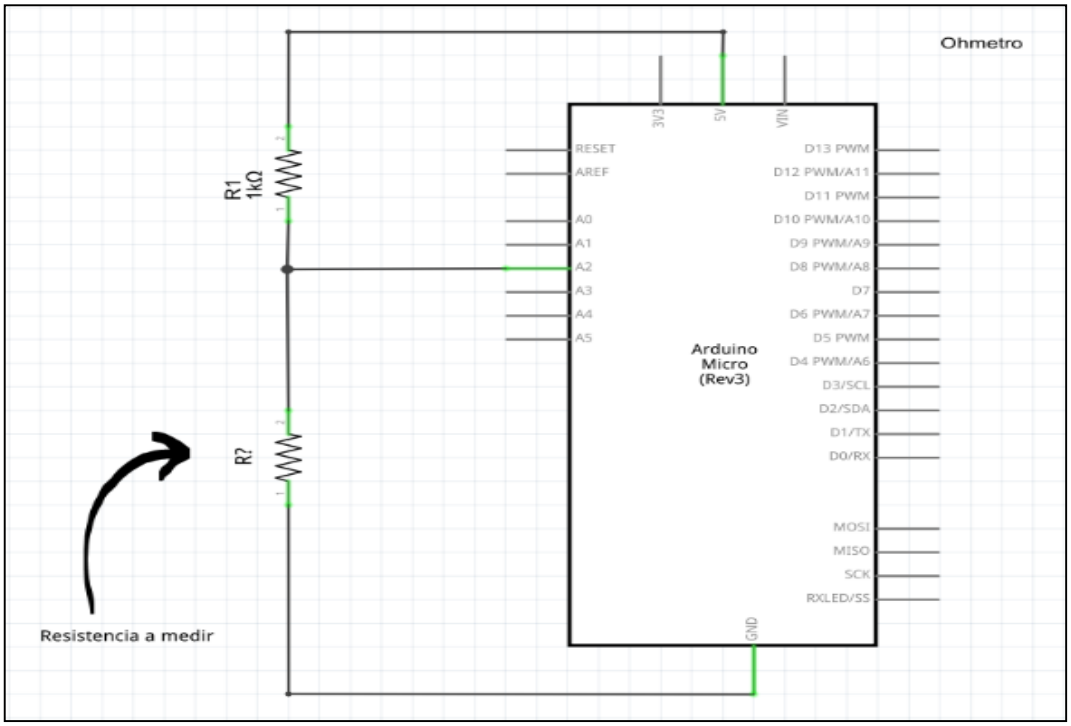


***Imagen 51**. Óhmetro sencillo realizado con Arduino.*

La mayor ventaja de este sensor es su gran versatilidad y fiabilidad, pues será capaz de detectarnos deformaciones muy pequeñas en cualquier punto de la galga. Otra ventaja importante es su pequeño tamaño, lo cual las hace ideales para poder ser implementadas en los dedos de nuestra mano. A pesar de sus importantes ventajas, no podemos perder de vista sus desventajas, las cuales son quizá más importantes. Destacan dos desventajas: Su alto precio y su gran fragilidad. Estas desventajas nos hacen descartar este sensor sin llegar a probarlo pues si este sensor, además de tener un precio elevado, es frágil y corre el riesgo de tener que ser sustituido por uno nuevo con cierta frecuencia, resulta totalmente inviable para ser implementado en nuestra prótesis, debido al requerimiento de un presupuesto lo más bajo posible.

2.2.1.4. SENSOR DE GRAFENO

Se ha decidido explotar las fantásticas propiedades del grafeno para desarrollar un sensor de contacto adecuado a las exigencias concretas de nuestro proyecto. Para ello se han realizado numerosas pruebas con las distintas muestras de grafeno que se han podido conseguir, hasta llegar a un sensor que aparentemente satisface todas nuestras necesidades.

El sensor desarrollado es muy simple y emplea las capacidades conductoras del grafeno. Consta de dos láminas elásticas de grafeno, las cuales están separadas por una esponja perforada en su parte central para que permita el contacto entre las dos láminas. Es importante elegir adecuadamente la esponja, eligiendo bien su densidad y grosor para que el contacto se realice de acuerdo a nuestros requerimientos.

En la Imagen 52 se pueden ver los materiales que componen el sensor creado, pudiéndose apreciar así la tremenda simplicidad y versatilidad que este nos ofrece en lo que a diseño se refiere.

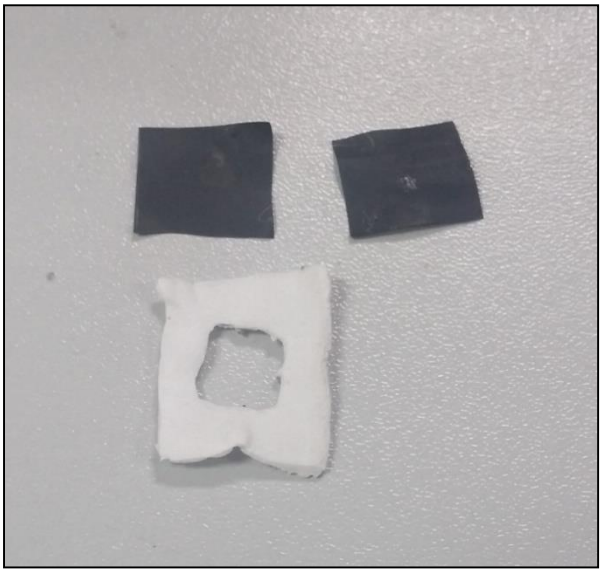


***Imagen 52**. Materiales empleados en el sensor de grafeno.*

Las láminas presentan conducción también entre sus dos caras, por lo que podremos fijar los cables a él por su parte exterior con cinta adhesiva, para así poder incorporar el sensor a distintos circuitos de una manera más cómoda. En la Imagen 53 podemos ver el aspecto final de nuestro sensor. Es importante destacar que a pesar de contar con un cable azul y uno rojo, el sensor no tiene una polaridad definida.

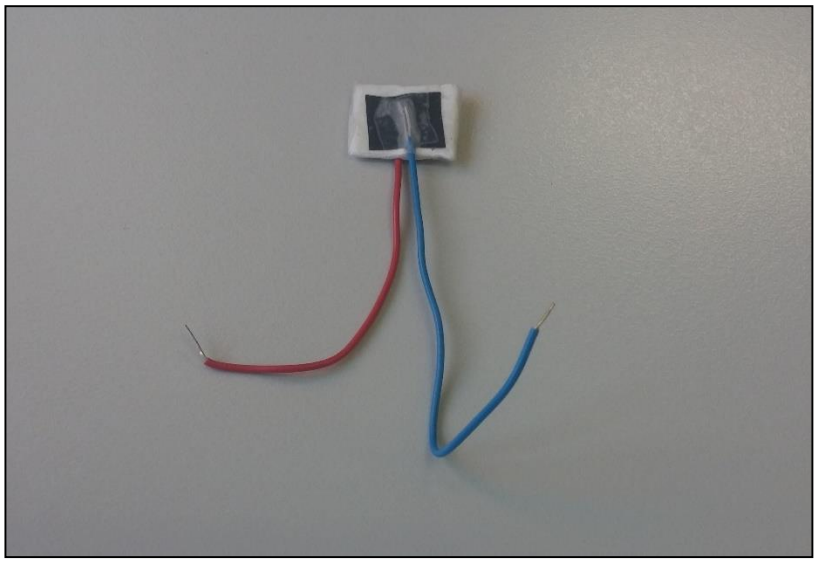


***Imagen 53**. Sensor de contacto creado.*

Este sensor está construido de una forma rápida y algo rudimentaria debido a la falta de tiempo. A pesar de ello, se han obtenido unos resultados muy buenos, obteniéndose un sensor muy delgado, flexible y fiable. Además, aunque el sensor desarrollado en este caso no es especialmente pequeño, podría emplearse el mismo principio para crear uno de un tamaño mucho menor.

Todo esto hace que este sensor resulte idóneo para nuestra prótesis, ya que no engordaría el tamaño de los dedos y es flexible, por lo que se puede adaptar a la forma de estos. Además,

presenta una gran versatilidad en su diseño, lo cual hace que se pueda construir a medida, para adaptarlo al lugar en el que se quiera situar.

2.2.1.4.1. PRUEBAS

En las pruebas de este sensor, se decidió emplear un LED, para así realizar las pruebas de una manera más visual. El montaje empleado para la prueba se presenta en la Imagen 54.

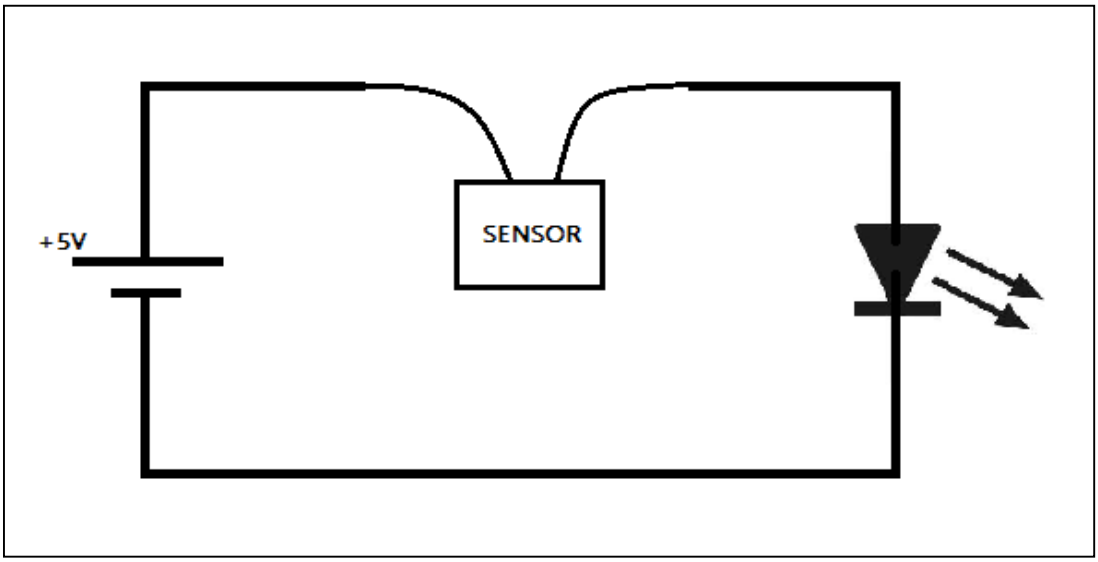


***Imagen 54**. Montaje empleado para probar el sensor de grafeno.*

Gracias a las pruebas se vio que el sensor presentaba un comportamiento óptimo, iluminándose el LED cuando se ejercía presión y apagándose cuando no, tal y como muestra la Imagen 55.

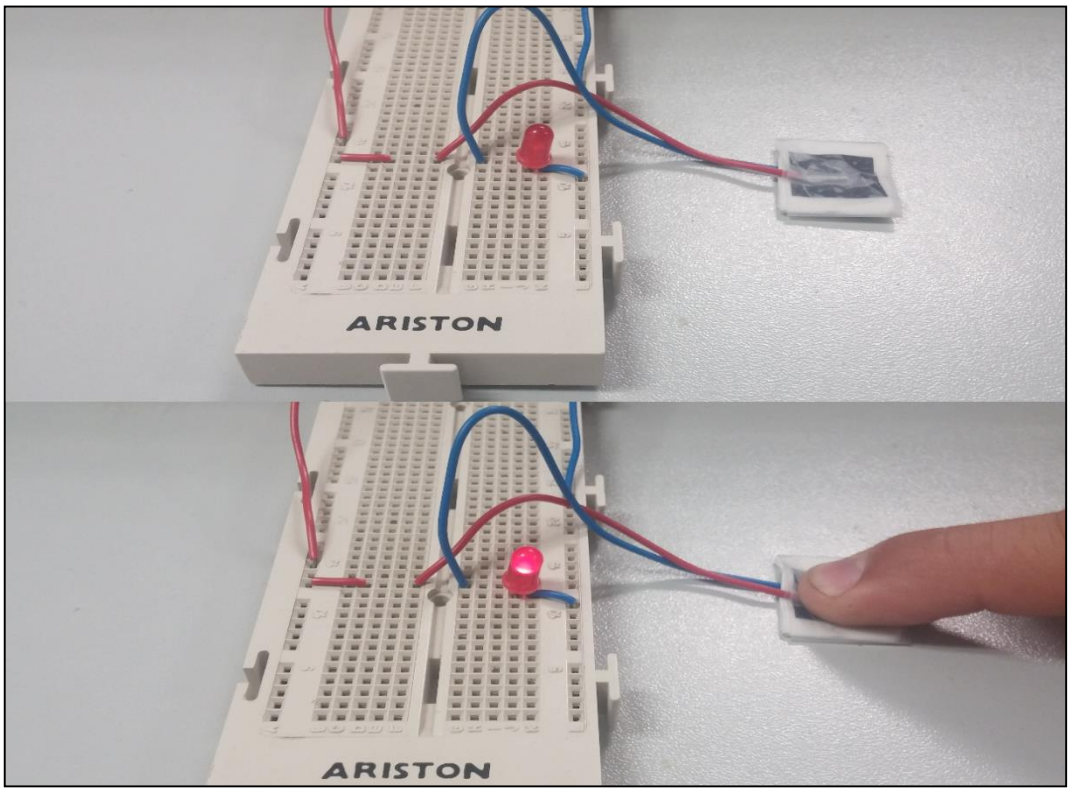


***Imagen 55**. Pruebas con el sensor de contacto.*

Durante las pruebas, también se comprobó que este sensor no nos ofrecía un resultado binario, es decir, encendido o apagado, sino que nos ofrece una activación progresiva en función de la fuerza que se ejerza, tal cómo podemos ver en el video del siguiente enlace:

https://www.youtube.com/watch?v=sB2yVks--3k

Esto nos abre todo un abanico de posibilidades, mejorándonos aún más las capacidades del sensor. De esta manera, ya no estemos ante un sensor de contacto, sino que estamos ante un sensor de fuerza, el cual nos puede medir la fuerza que estamos ejerciendo sobre él.

Por este motivo se decidió realizar una tabla en la que se medía la caída de voltaje en el sensor para diferentes presiones ejercidas, y así poder graficar la respuesta del sensor a la presión. Para realizar estas mediciones simplemente se alimentó el sensor con +5V y se midió la resistencia que ofrecía el sensor (R_a), tal y como se indica en la Imagen 56. Se escogió una alimentación de 5V dado que es la más empleada por la electrónica de nuestro sistema.

***Imagen 56**. Montaje empleado para medir la respuesta del sensor de presión.*

Por la falta de materiales para realizar unas medidas de la fuerza ejercida, se tuvo que hacer un montaje que permitiera hacerlo, aunque no fuera del todo preciso.

Para ello se pegó el sensor sobre una tapa de botella. A este se pegó una placa rígida y sobre esta última se situó una botella que se iba llenando de agua para simular el aumento de fuerza ejercido.

Con el fin de simular el tacto real, no se depositó el sensor sobre la mesa, sino que se realizaron las medidas apoyando el montaje sobre un dedo, tal como se muestra en la Imagen 57.

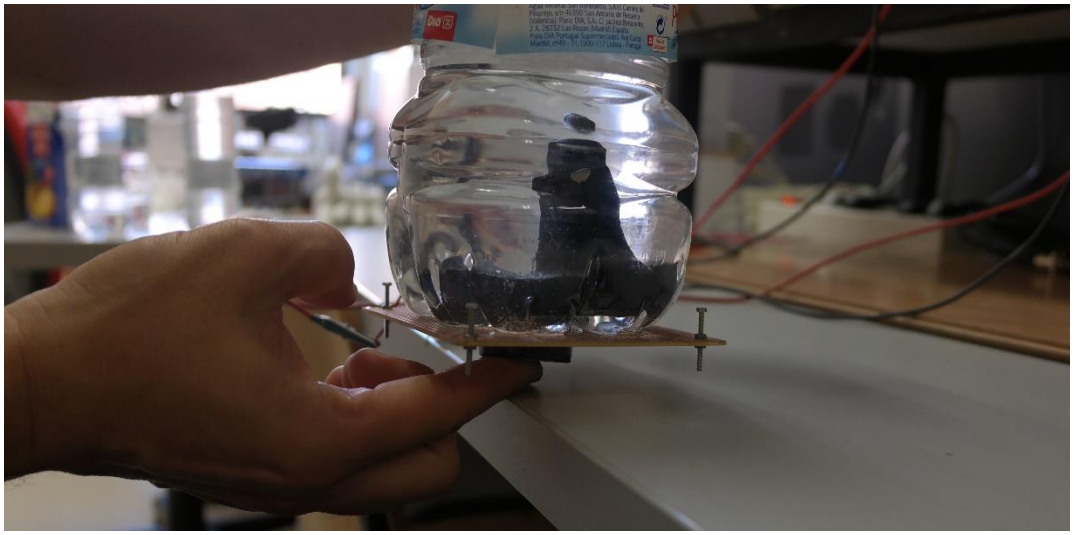


***Imagen 57**. Pruebas del sensor de grafeno (I).*

Para que la posición variase lo menos posible, se sujetó el montaje por su parte trasera para que este no se moviese.

***Imagen 58**. Pruebas con el sensor de grafeno (II).*

Para las pruebas se comenzó con un peso de 400 gramos y se fue aumentando el peso de 200 en 200 gramos, anotando la resistencia que ofrecía el sensor en cada punto para posteriormente dibujar la curva característica del sensor. Hay que tener en cuenta que cada sensor de este tipo puede presentar una curva diferente, y que esta no es 100% precisa debido a la manera en la que se han tomado las medidas.

Peso (Gramos)	*Resistencia (Kiloohmios)*
400	45
600	22
800	14
1000	8
1200	6.5
1400	5.4
1600	5

***Tabla 4**. Medidas del sensor de grafeno.*

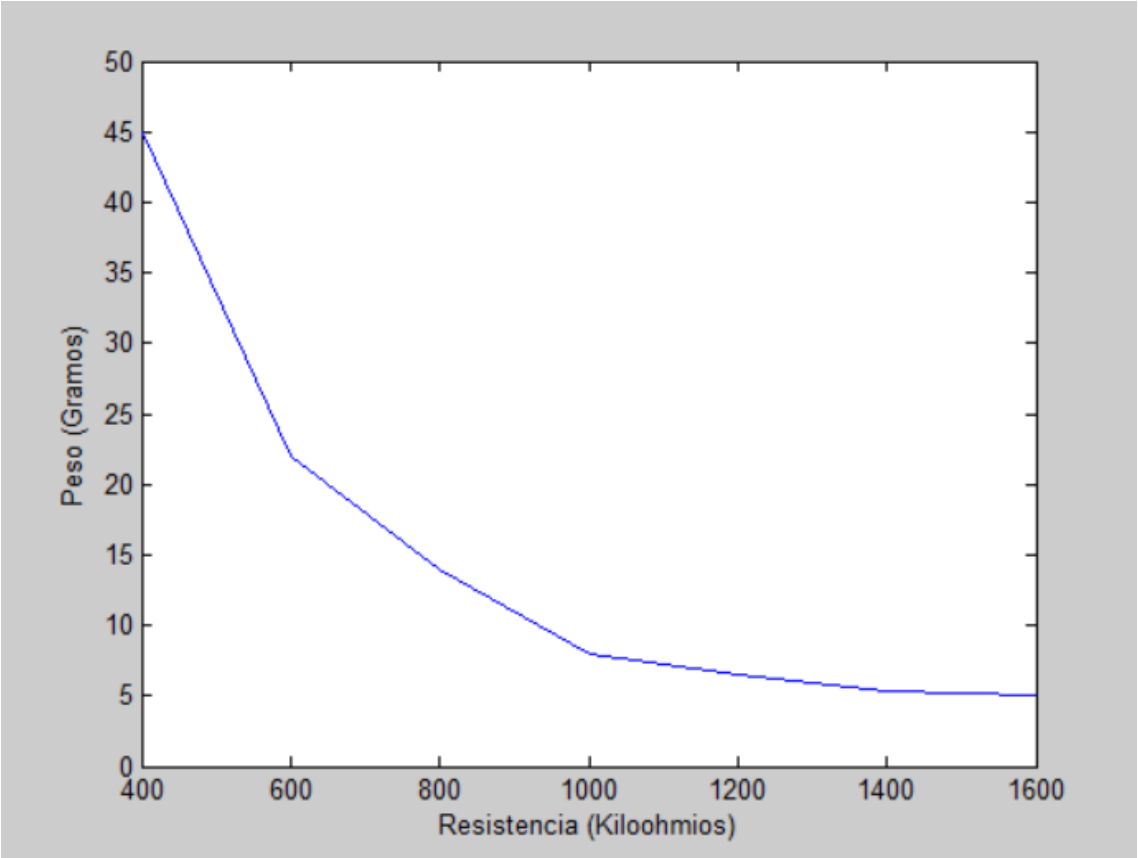


***Imagen 59**. Curva característica del sensor de grafeno.*

2.3. SISTEMA MIOELÉCTRICO DE CONTROL DE LA MANO

Cómo base de este sensor mioeléctrico se ha tomado el creado en el PFC "Sistema de control de dispositivos externos mediante electromiograma: Aplicación a una mano robótica" desarrollado por Javier García Rodríguez en octubre de 2014 ([8]). Sin embargo, no se ha tomado tal cual, sino que se le han implementado numerosas mejoras, siendo el montaje final muy diferente al de [8].

En esta área se expondrá con todo tipo de detalle cada una de las etapas de las placas de acondicionamiento y adquisición de las señales electromiográficas. Así mismo, se describirán los componentes empleados para la conformación de cada una de ellas. Cabe destacar que el diseño e implementación de la placa en [8] está basada en el desarrollo de [18], completándolo con los últimos avances del grupo de [19]. Posteriormente se implementarán mejoras sobre este diseño, basándonos en los montajes empleados en [20]. El objetivo de este sistema de control es detectar los cambios en la amplitud de las señales EMG que se producen cuándo se realiza un esfuerzo muscular.

Es importante mencionar que hubiera sido conveniente emplear integrados cuyos operacionales fuesen 'Rail-to-rail' (Operacionales que pueden ofrecer a su salida amplitudes similares a las de alimentación, sin apenas pérdidas), sin embargo, esto no ha sido posible debido al coste en tiempo que supondría pedirlos a una tienda y esperar a que llegaran, dado que en el laboratorio no disponíamos de ellos. A pesar de que nosotros no hemos podido emplear estos integrados, sí que sería interesante que en un futuro se empleasen, por lo que se dejará constancia de ello en el apartado 'Líneas de mejora'. En dicho apartado podremos ver una explicación más detallada de este problema.

A continuación, detallaremos cada una de las etapas mediante las cuales se llevará a cabo la adquisición y acondicionamiento de las señales, provenientes de los electrodos y que llevaremos al Arduino. Debemos remarcar que contaremos con dos placas idénticas, una para cada uno de los canales con los que controlaremos nuestro dispositivo.

2.3.1. AMPLIFICADOR DE INSTRUMENTACIÓN

Podemos definir el amplificador de instrumentación como un conjunto de amplificadores operacionales interconectados entre sí de una forma concreta, tal y como podemos apreciar en la Imagen 60 (obtenida de [21]). De esta manera, obtenemos la implementación de un nuevo dispositivo con un alto rechazo al modo común y una alta impedancia de entrada.

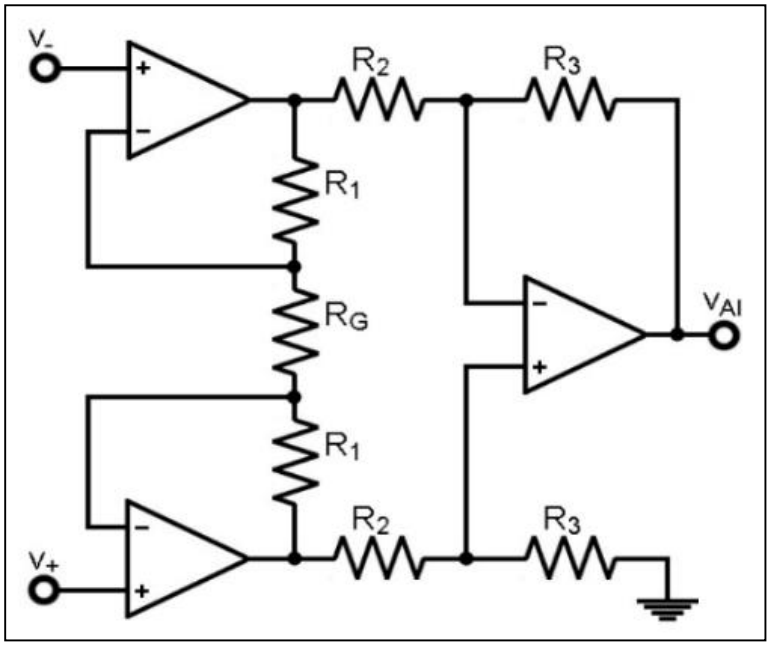


***Imagen 60**. Amplificador de instrumentación realizado a partir de tres amplificadores operacionales.*

En un amplificador de instrumentación común, el voltaje a la salida queda definido tal y como muestra la ecuación (1).

$$V_{AI} = (v_- - v_+)\left(1 + \frac{2R_1}{R_G}\right)\frac{R_3}{R_2} \qquad (1)$$

Donde podemos ver que la señal de salida depende de la diferencia entre las señales de entrada; de ahí que digamos que se trata de una señal diferencial.

Por otro lado, la ganancia viene definida por la ecuación (2), en la que son partícipes una resistencia discreta externa a la cual denominamos R_G, y una serie de componentes del amplificador de instrumentación, integradas en un mismo circuito.

$$G_{AI} = \left(1 + \frac{2R_1}{R_G}\right)\frac{R_3}{R_2} V_{AI} = (v_- - v_+)\left(1 + \frac{2R_1}{R_G}\right)\frac{R_3}{R_2} \qquad (1)$$

Tal y como se ha estado explicando, la primera etapa se corresponde con el amplificador de instrumentación. El objetivo fundamental de esta etapa es obtener un alto rechazo en modo común de la señal de entrada. Presenta una relación de rechazo en modo común (CMRR, Common Mode Rejection Ratio) muy alta, al mismo tiempo que se ve reducido notablemente el ruido [22] [23].

La señal de interés, es la correspondiente al modo diferencial a la entrada del sistema, que será lo que se obtenga, amplificada por un factor conveniente, a la salida de esta primera etapa. De esta forma, la salida queda definida por la ecuación (3) con G_{AI} definido previamente en la ecuación (2) a partir del resultado obtenido en (1).

$$V_{AI} = (v_- - v_+)\, G_{AI} \qquad (2)$$

Tras explicar el funcionamiento del amplificador de instrumentación, se procederá a mostrar el integrado que se empleará en el montaje final del sistema.

2.3.1.1. INTEGRADO INA114

Se decide emplear un circuito integrado INA 114, perteneciente a la casa Burr-Brown, cuya hoja de especificaciones se puede consultar en el anexo 13, con el objetivo de realizar las funciones de amplificador de instrumentación y, de esta forma, completar la primera etapa de cada una de las placas de adquisición de la señal.

Entre las características que hacen este circuito integrado óptimo para nuestros requerimientos, cabe destacar su bajo voltaje de continua a la salida, una baja corriente de polarización de entrada y un alto rechazo al modo común. Así mismo, en la hoja de características de dicho circuito integrado, se incluyen como posibles aplicaciones del mismo, su uso como instrumentación médica, que a fin de cuentas será el sector en el que desarrollemos el dispositivo que nos ocupa.

En la Imagen 61 podemos ver el diagrama de bloques completo del circuito integrado INA 114 sacado directamente de sus hojas de especificaciones. Tal y como podemos comprobar, la distribución interna del circuito es análoga a la mostrada en la Imagen 60 correspondiente a un amplificador de instrumentación convencional.

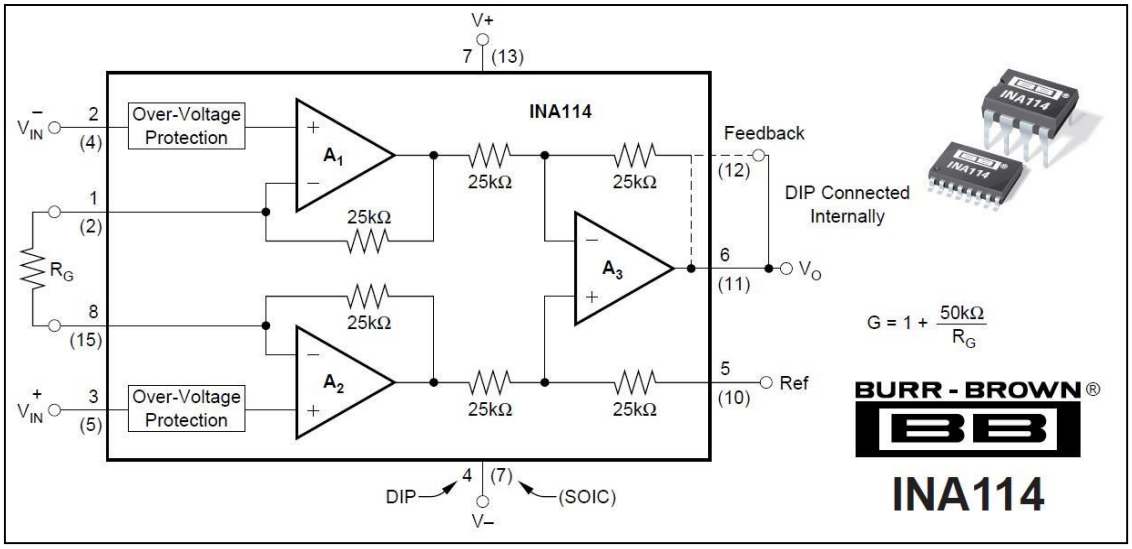


***Imagen 61**. Diagrama de bloques completo del circuito integrado INA 114.*

Como ya se comentó con anterioridad, en el diagrama aparece representada la resistencia externa RG, la cual fijaremos para obtener la ganancia requerida. Tal y como se nos indica en la hoja de especificaciones del circuito integrado INA114, la ganancia se puede calcular utilizando la ecuación (4). Esta ecuación se corresponde con la mostrada anteriormente en (2) particularizando para los valores de resistencia mostrados en la Imagen 61.

$$G = 1 + \frac{50k\Omega}{R_G} \qquad (3)$$

Necesitaremos emplear una resistencia discreta que se adapte a las necesidades de la etapa que queremos construir. Así mismo debemos encontrar un compromiso entre la ganancia de esta etapa y el resto de características de la etapa que nos ocupa.

Tal y como podemos ver observando las gráficas de la Imagen 62, se debe emplear una ganancia para la cual no se produzcan variaciones en frecuencia en la banda de trabajo. Por tanto, el requerimiento principal es que, en el rango de frecuencias de interés, es decir, desde los 50 Hz hasta los 500 Hz aproximadamente, no se produzcan variaciones en la ganancia.

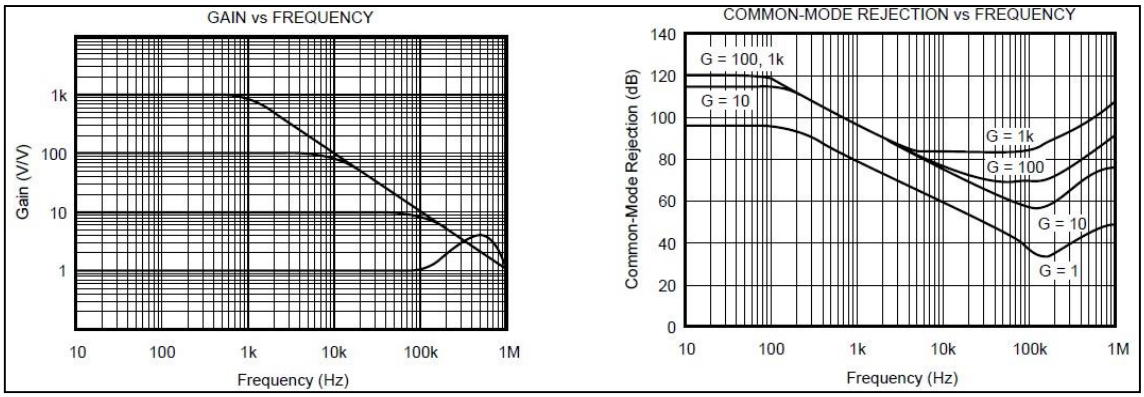


***Imagen 62**. Gráficas características del circuito integrado INA 114. Ganancia frente a la frecuencia (izquierda) y rechazo en modo común frente a frecuencia (derecha).*

Además, es importante conseguir un rechazo al modo común alto en el rango de frecuencias de trabajo. Teniendo presente los diferentes factores se eligió emplear una ganancia en torno a 20. Tal y como se puede concluir observando las gráficas de la Imagen 62, se obtiene una ganancia constante hasta los aproximadamente 10 kHz, punto en el cual comienza a decaer y un rechazo al modo común suficiente para nuestra aplicación, aunque si bien se percibirán variaciones en el mismo conforme aumenta la frecuencia. La clave está en buscar un equilibrio entre ganancia, relación de rechazo común y señal continua de entrada. Justificamos la elección de la ganancia de 20, pues en nuestro rango de frecuencias es constante, nos ofrece una relación de rechazo común aceptable y el offset que se introduce es aceptable.

Por tanto, rigiéndonos por la ecuación (4), para conseguir una ganancia aproximada de 20, utilizamos una resistencia de 2.7 KΩ. Si bien, debemos tener siempre en cuenta que dicho valor es un valor teórico.

$$G = 1 + \frac{50k\Omega}{2.7k\Omega} = 19.5185$$

El valor real tendrá que ver con la tolerancia de la resistencia discreta utilizada, en este caso, hemos empleado una resistencia con una tolerancia de un 1%. Podemos apreciar la disposición final de la etapa en la Imagen 63.

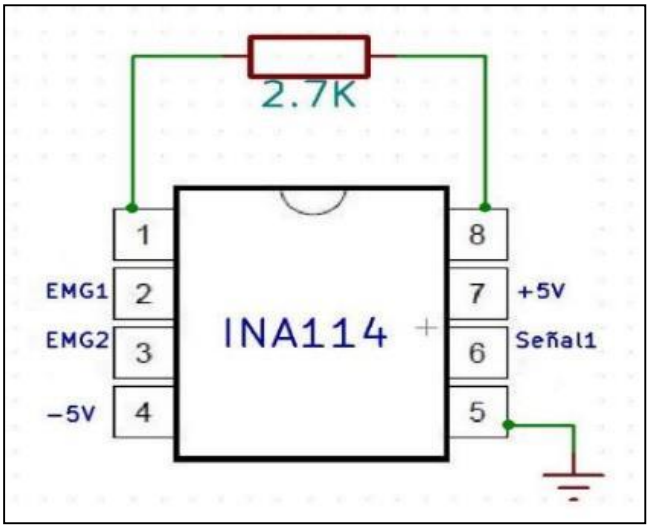


***Imagen 63**. Diagrama de la etapa correspondiente al amplificador de instrumentación.*

2.3.2. AMPLIFICADOR Y FILTRO PASO ALTO

En primer lugar, deberemos crear un amplificador para aumentar la amplitud de las señales y así poder trabajar con ellas de una manera más cómoda.

Idealmente deberíamos haber introducido la ganancia reduciendo la resistencia 'R_g' del amplificador de instrumentación. Sin embargo, como ya hemos visto aumentar la ganancia de esta forma podría ocasionarnos problemas, por lo que decidimos introducir la etapa amplificadora.

Nuestro amplificador consta de 2 fases: La primera de ellas es una fase inversora de ganancia 15, mientras que la segunda es también inversora pero esta vez de ganancia 1. Esto hace que a la salida de nuestro amplificador tengamos la señal de la entrada multiplicada por 15, dado que la inversión de las dos fases se anula entre sí.

Este amplificador incluye en la segunda fase un filtro paso alto con el que se pretende eliminar las componentes frecuenciales que estén por debajo de nuestra banda de interés. Principalmente trataremos de eliminar el ruido de red, pues es el que más nos estorba en debido a su amplitud, y la componente CC que saturaría los amplificadores siguientes.

Un filtro paso-alto (HPF, High Pass Filter) es un tipo de filtro electrónico capaz de realizar una selección de frecuencias, así como los niveles de continua, de manera que se atenuarán las componentes de baja frecuencia, pero no así las de alta frecuencia, las cuales, se pueden incluso amplificar si se trata de un filtro activo.

De este modo, la apariencia de la función de transferencia de un filtro paso alto con una frecuencia de corte a 3 dB (fo) y con ganancia unidad, es la que se refleja en la Imagen 64.

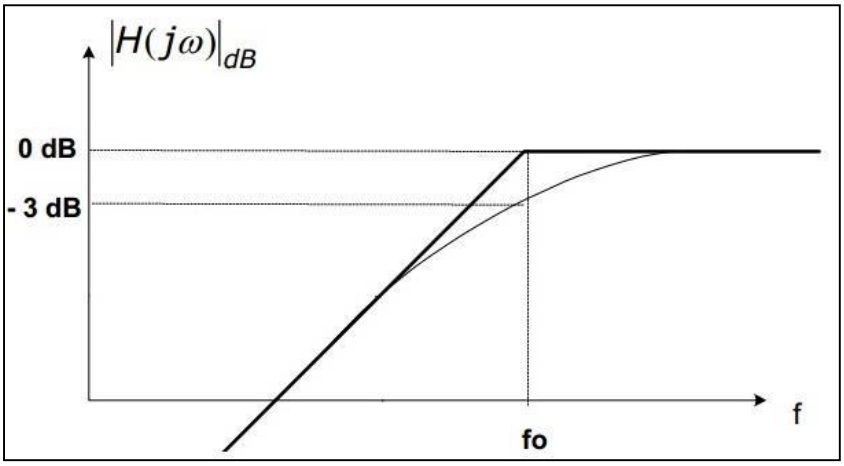


***Imagen 64**. Función de transferencia de un filtro paso-alto.*

La función de transferencia de un filtro paso-alto viene regida por la ecuación (5); en la cual, ω_c es la frecuencia de corte a 3dB, calculada a partir de (6).

$$H(\omega) = \frac{\frac{j\omega}{\omega_c}}{1+\frac{j\omega}{\omega_c}} = \frac{1}{1+\frac{\omega_c}{j\omega}} \quad (4)$$

$$|H(\omega_c)| = \frac{|H(\omega)|_{max}}{\sqrt{2}} \quad (5)$$

En la Imagen 65 podemos apreciar las principales características de las señales EMG en cuanto a amplitud y rango de frecuencias se refiere, al mismo tiempo que se compara con las señales de electrocardiograma (ECG) y las señales de potencial de acción de axón (AAP). Tal y como podemos apreciar, las señales EMG concentran su actividad en el rango de frecuencias comprendido entre los 50 y los 500 Hz.

Idealmente tomaríamos como frecuencia de corte 50Hz, pero de esta manera no conseguiríamos eliminar el ruido de red, por lo que tomaremos como frecuencia de corte 100Hz. El hecho de que este filtro sea muy suave, hace que las componentes del electromiograma entre 50Hz y 100Hz no sean eliminadas, sino que tan solo se verán atenuadas, en mayor o menor medida, según su posición en el espectro.

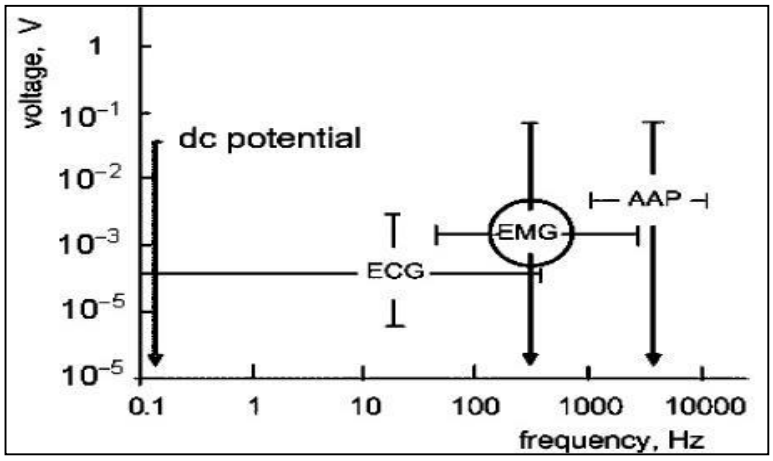


***Imagen 65**. Características señales EMG: amplitud y frecuencia.*

2.3.2.1. INTEGRADO TL084

Se decide emplear un circuito integrado TL084 perteneciente a la casa STMicroelectronics, para implementar esta etapa. Dicho circuito integrado consta de cuatro amplificadores operacionales con entrada JFET y con una alimentación común. En la Imagen 66 podemos apreciar el esquema general y distribución de las patillas, extraído de las hojas de características. En el anexo 14 podremos consultar sus hojas de especificaciones.

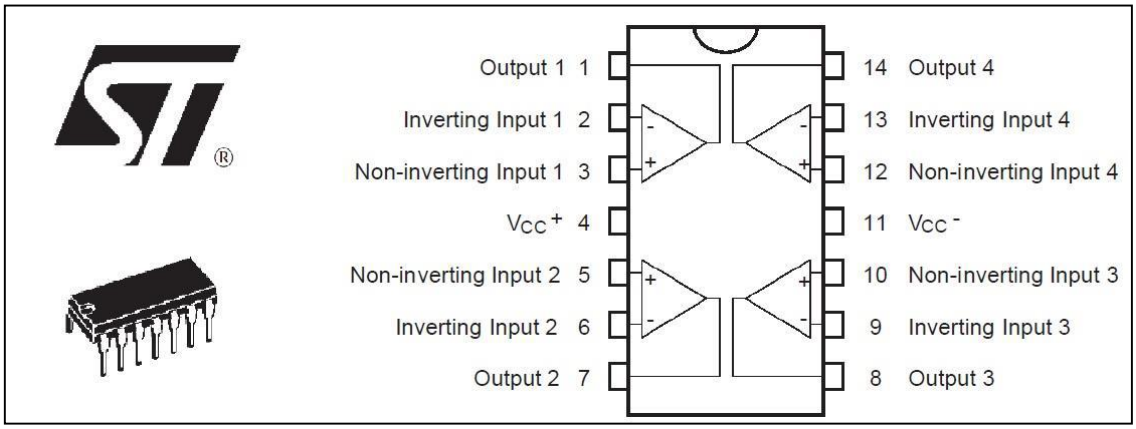


***Imagen 66**. Esquema de distribución del circuito integrado TL084.*

En el esquema mencionado podemos apreciar una visión general del circuito integrado junto con la distribución de cada una de sus patillas. Así mismo, se puede observar en el esquema la conexión interna con cada uno de los amplificadores operacionales, que incluye el circuito integrado del que estamos tratando.

Se decidió elegir el circuito integrado TL084 debido a la óptima adaptación al sistema diseñado, así como unas perfectas características que posee, entre las que cabe destacar, por ejemplo, su bajo consumo, su alta impedancia de entrada debido a que las entradas están basadas en tecnología de efecto de campo por su baja corriente de polarización en la entrada y su alta velocidad de cambio del voltaje de salida con respecto a las variaciones en el voltaje de entrada. Además, de poder utilizar una alimentación de ±5V, igual a la del resto del sistema.

Estas características, hacen que el circuito integrado TL084 se convierta en inmejorable para el sistema ya que nos conviene un consumo lo más bajo posible. Así mismo, se necesitan respuestas rápidas en la salida respecto a la entrada, pues hay que recordar que los períodos de las señales EMG son extremadamente cortos (3-15 ms).

Como el voltaje de alimentación es igual al de la etapa anterior correspondiente del amplificador de instrumentación, conseguimos que el rango dinámico sea constante, de manera que se reducen notablemente los problemas relacionados con las saturaciones, así como con los picos de consumo.

En la Imagen 67 podemos apreciar el diseño elegido para la implementación del amplificador con filtro paso alto de la etapa que nos ocupa. En la misma se pueden observar, además de los principales valores de interés de los componentes, los voltajes más significativos, así como la correspondencia de patillas del circuito integrado TL084.

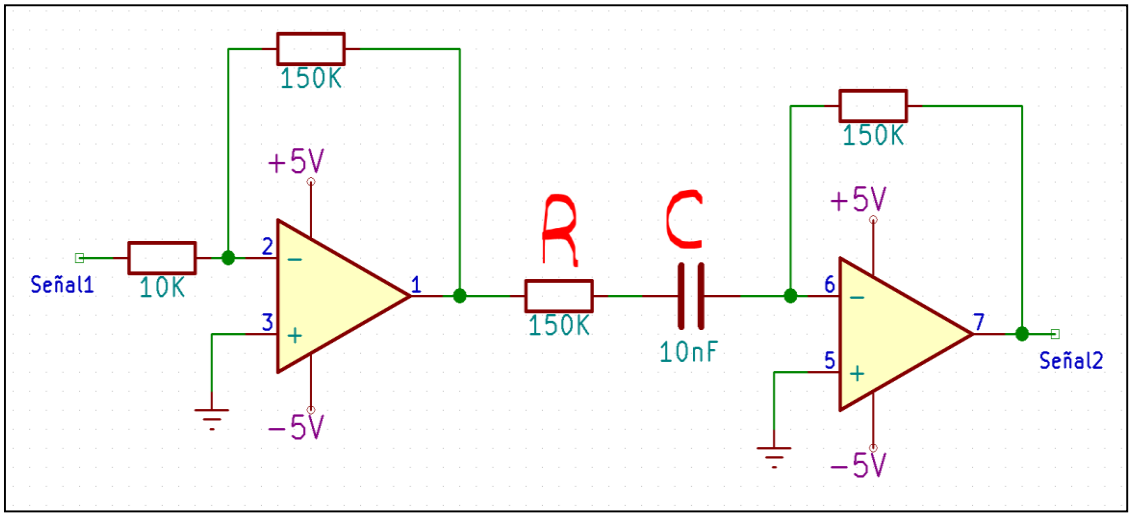


***Imagen 67**. Implementación del amplificador junto con el filtro paso alto.*

Sabemos que la fórmula que determina la frecuencia de corte en este tipo de filtros viene determinada por 1/(2π*RC). Como necesitamos que la frecuencia de corte sea de unos aproximadamente 100 Hz, elegimos dos valores arbitrarios que nos hagan cumplir esta premisa. Así pues, elegimos un valor de 10 nF para el condensador y 150 KΩ para la resistencia.

2.3.3. RECTIFICADOR DE ONDA COMPLETA

Un rectificador de onda completa es un circuito empleado para convertir una señal de corriente alterna de entrada en corriente continua de salida pulsante. A diferencia del rectificador de media onda, en este caso, la parte negativa de la señal se convierte en positiva o bien la parte positiva de la señal se convertirá en negativa, según se necesite una señal positiva o negativa de corriente continua. En nuestro caso nos interesa una señal positiva, dado que lo que queremos es tomar medidas sobre la amplitud.

Para crear un circuito rectificador de onda completa emplearemos dos diodos, los cuales irán variando entre corte y conducción para así convertir en positiva la parte negativa de la onda.

Para las resistencias decidimos utilizar unos valores de 10 KΩ. En esta etapa emplearemos un nuevo integrado TL084, el cual se compartirá con la etapa siguiente. En la Imagen 68 se indica el montaje escogido para implementar el rectificador de onda completa. En la misma se pueden observar, además de los principales valores de interés de los componentes, los voltajes más significativos, así como la correspondencia de patillas del circuito integrado TL084.

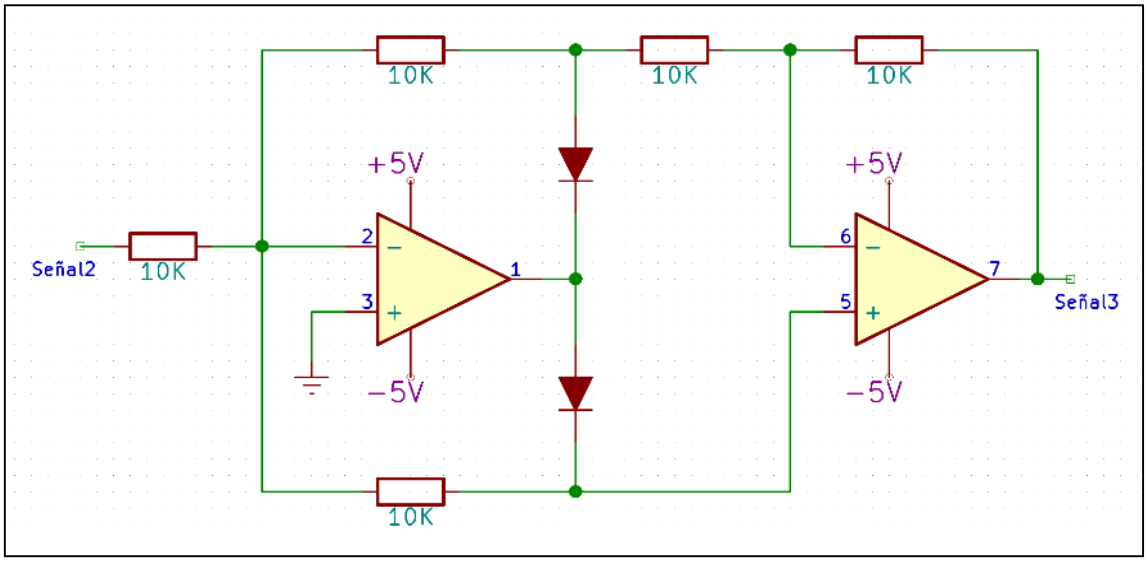


***Imagen 68**. Implementación de un rectificador de onda completa.*

En ocasiones, debido al voltaje umbral de los diodos, no se llegan a rectificar señales de unos pocos mV, sin embargo, como nosotros ya hemos amplificado nuestra señal, no nos preocuparemos por este hecho. El funcionamiento del circuito se resume en las formas de onda de la Imagen 69: Se ve que las señales positivas no se modifican, mientras que las negativas se convierten en positivas.

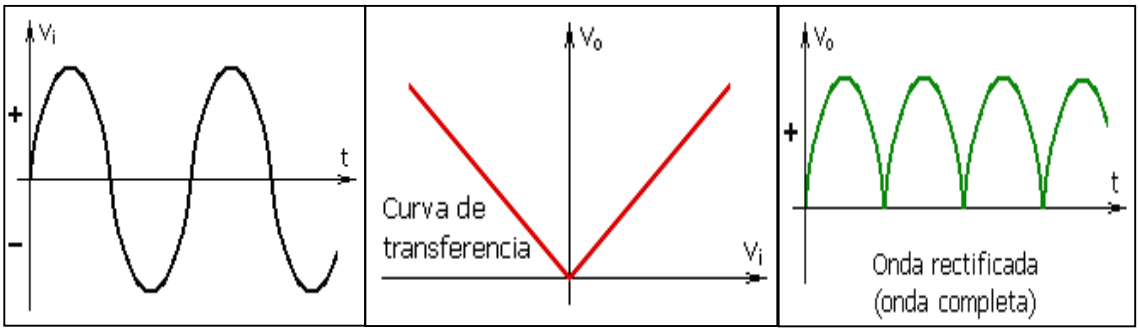


***Imagen 69**. Características de entrada, transferencia y salida de un rectificador de onda completa ideal con salida positiva.*

Por tanto, a la salida de esta etapa, en el punto llamado 'Señal3' tendremos la misma señal que en el punto 'Señal2' sólo que con las partes negativas de la señal convertidas en positivas. En Imagen 70 podemos apreciar el aspecto que presenta una señal EMG una vez rectificada.

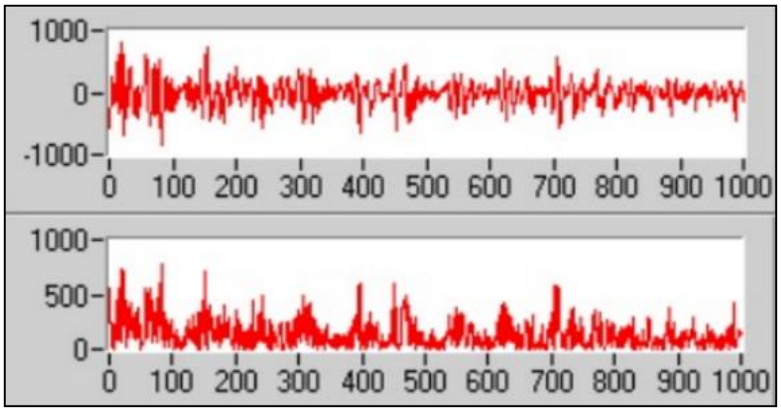


***Imagen 70**. Señal EMG a la salida del rectificador de onda completa.*

2.3.4. FILTRO PASO BAJO ACTIVO CON GANANCIA REGULABLE

Un filtro paso bajo (LPF, Low Pass Filter) es un tipo de filtro electrónico capaz de realizar una selección de frecuencias, de manera que se atenuarán las componentes de alta frecuencia, pero no así las de las frecuencias más bajas, las cuales, se pueden incluso amplificar si se trata de un filtro activo. En la Imagen 71, se puede ver la apariencia de la función de transferencia de un filtro paso bajo con una frecuencia de corte a 3 dB (fo).

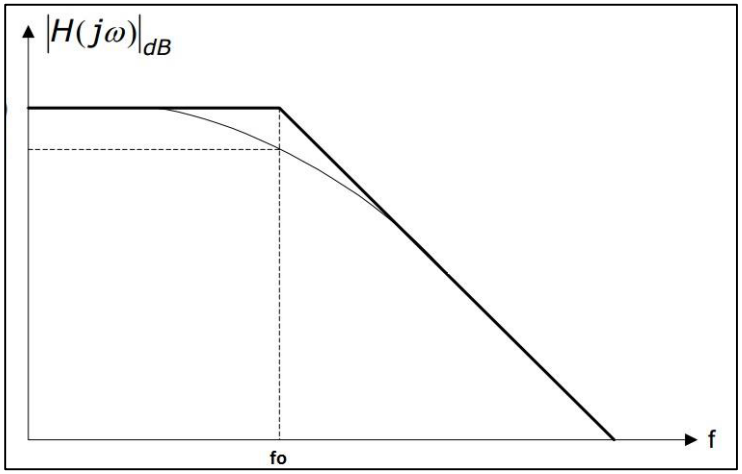


***Imagen 71**. Función de transferencia de un filtro paso-bajo.*

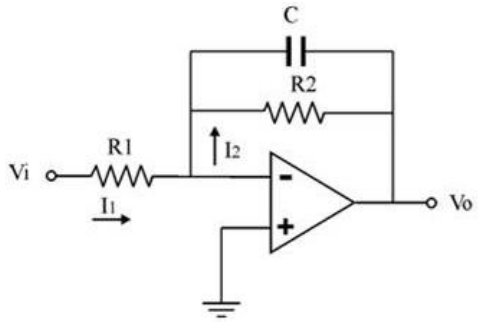
Un filtro paso-bajo, en analogía con el filtro paso-alto, es un dispositivo electrónico que permite el paso de las frecuencias bajas, pero produce una atenuación fuera de la banda de interés, es decir, que atenúa las frecuencias que vayan más allá de su región de funcionamiento.

La función de transferencia de un filtro paso-bajo viene regida por la ecuación (7); en la que ω_c es la frecuencia de corte a 3dB, calculada a partir de (8).

$$H(\omega) = \frac{1}{1 + \frac{j\omega}{\omega_c}} \qquad (6)$$

$$|H(\omega_c)| = \frac{|H(\omega)|_{max}}{\sqrt{2}} \qquad (7)$$

En este caso, no deseamos hacer un filtro paso bajo, que, combinado con el paso alto anterior, nos dé un paso banda que nos permita seleccionar las frecuencias de una banda de interés. Esta vez lo que queremos conseguir es un nivel de continua para ver la amplitud de nuestro EMG, y emplear dicha amplitud para distinguir los distintos niveles de esfuerzo muscular que estamos realizando, los cuales emplearemos para controlar nuestra prótesis. Es por esto por lo que ahora construiremos un filtro paso bajo con una frecuencia de corte extremadamente baja, consiguiendo así eliminar casi todas las componentes frecuenciales y quedándonos prácticamente con un nivel de continua.



$$H(\omega) = \frac{V_o}{V_i}$$

$$\left.\begin{aligned} I_1 &= \frac{V_i}{R_1} \\ I_2 &= \frac{-V_o}{Z_c // R_2} \end{aligned}\right\} \frac{V_i}{R_1} = \frac{-V_o}{Z_c // R_2}$$

$$H(\omega) = -\frac{Z_c // R_2}{R_1} = -\frac{Z_c R_2}{R_1(Z_c + R_2)} = \frac{-1}{R_1\left(\frac{1}{R_2} + \frac{1}{Z_c}\right)} = \frac{-1}{R_1\left(\frac{1}{R_2} + j\omega C\right)} = \frac{-1}{\frac{R_1}{R_2}(1 + j\omega C R_2)}$$

$$H(s) = -\frac{R_2}{R_1}\frac{1}{1 + \frac{s}{1/R_2 C}} \quad \Rightarrow \quad \begin{cases} H_o = H(s=0) = -\frac{R_2}{R_1} \\ \omega_c = \frac{1}{R_2 C} \end{cases}$$

Sabemos que la fórmula que determina la frecuencia de corte en este tipo de filtros viene determinada por $1/(2\pi * R_2 C)$. Buscaremos que la frecuencia de corte sea muy baja, de unos 0.5 Hz, y para ello, elegimos dos valores arbitrarios que nos hagan cumplir esta premisa. Así pues, elegimos un valor de 1 µF para el condensador y 82 KΩ para la resistencia.

El filtro aquí diseñado consta de dos fases. En la primera, la señal además de filtrarse, se amplifica y se invierte. Por otro lado, la segunda fase invierte de nuevo la señal, y además aporta a esta una ganancia variable, regulable gracias a un potenciómetro. Gracias a pruebas realizadas con el circuito se observó que una ganancia fija adecuada para la primera fase era 4. Para conseguir la amplificación mencionada se han elegido los valores de las resistencias de R1=22KΩ y R2=82KΩ. Se han elegido dichos valores ya que la ganancia en la banda de paso del filtro, en unidades naturales, viene regida por la ecuación (9)

$$|G| = \frac{R_2}{R_1} = \frac{82k\Omega}{22k\Omega} = 3.73 \qquad (8)$$

En la Imagen 72 se indica el montaje escogido para implementar el filtro paso bajo activo con amplitud variable. En la misma se pueden observar, además de los principales valores de interés de los componentes, los voltajes más significativos, así como la correspondencia de patillas del circuito integrado TL084, que será el mismo que el empleado en el rectificador de onda completa.

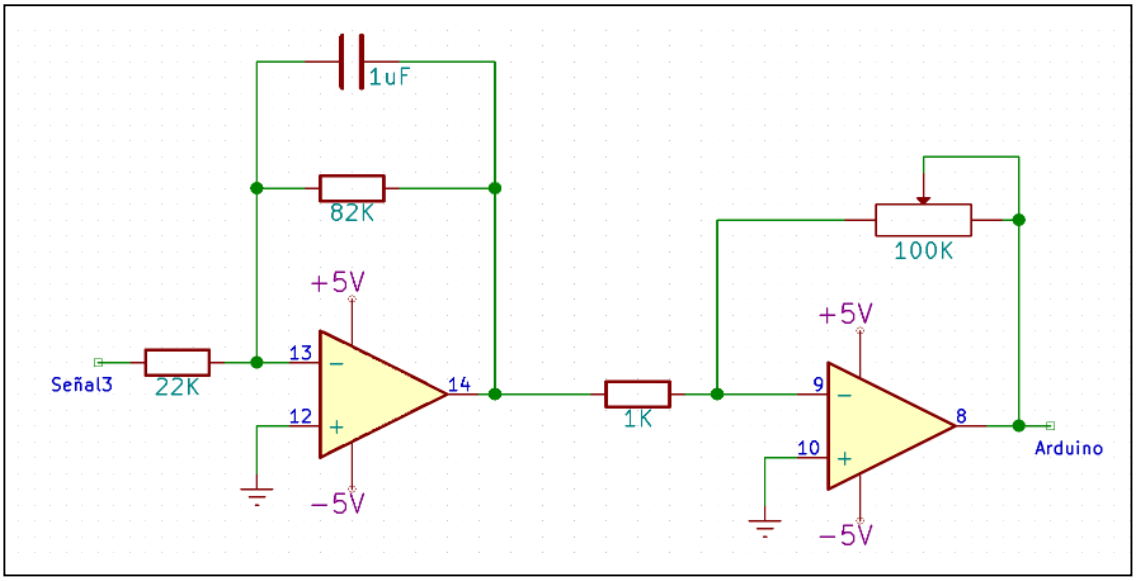


***Imagen 72**. Implementación del filtro paso bajo activo con amplitud variable.*

La ganancia variable se podrá modificar mediante un potenciómetro y su valor oscilará entre 0 y 100 (Debido a que el valor del potenciómetro oscila entre 0 y 100KΩ). Esta resistencia variable es de gran utilidad debido a que diferentes personas pueden necesitar diferentes valores de ganancia para el correcto procesamiento de sus señales mioeléctricas. Esto puede deberse a varios factores: Diferente cantidad de grasa en la piel, distinta conductividad de esta, desigual distribución de potenciales nerviosos en los músculos…

Para mis señales EMG concretas se obtenían unos resultados óptimos cuándo el potenciómetro ofrecía una resistencia de unos 8KΩ, o lo que es lo mismo, una ganancia de -8.

Es importante darse cuenta que, si inicialmente filtramos paso alto con una frecuencia de corte de 100Hz y posteriormente paso bajo con una frecuencia de corte de 0,5Hz nos saldría prácticamente 0. Sin embargo, esto no ocurre así gracias al rectificador de onda completa, que se encuentra situado entre los dos y que hace aparecer una considerable componente CC, que pasa por el filtro paso bajo, siendo amplificada además por el valor H_0.

2.3.5. MONTAJE FINAL DEL SENSOR EMG

En la Imagen 73 podemos ver el montaje final del sistema de captación y procesado de la señal EMG. Este sensor nos detectará los aumentos en la amplitud de la señal EMG que se producen al realizar esfuerzo muscular.

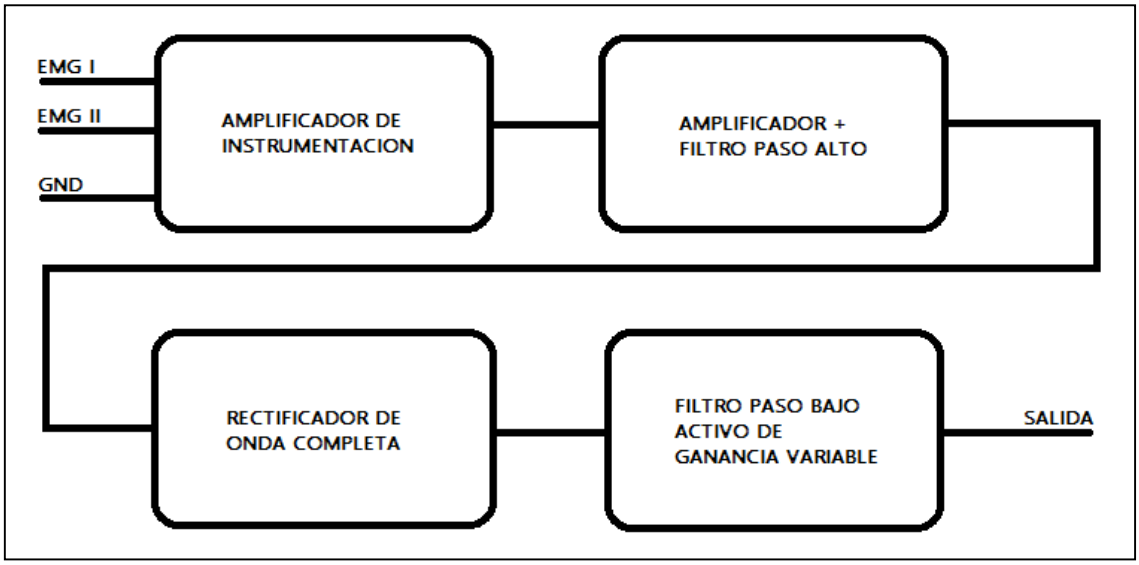


***Imagen 73**. Diagrama de bloques del sistema mioeléctrico de control.*

En resumen, nuestro sistema de captación y procesado de la señal realizará el siguiente proceso, para cada uno de los dos canales:

1. Tomará una señal diferencial de nuestro músculo y la pasará por un amplificador de instrumentación.
2. La señal será amplificada y filtrada paso alto.
3. La señal se pasará por un rectificador de onda completa.
4. Filtraremos la señal con un filtro paso bajo activo de ganancia variable.

En la

Imagen 74 podemos ver una foto del montaje final del sistema mioeléctrico.

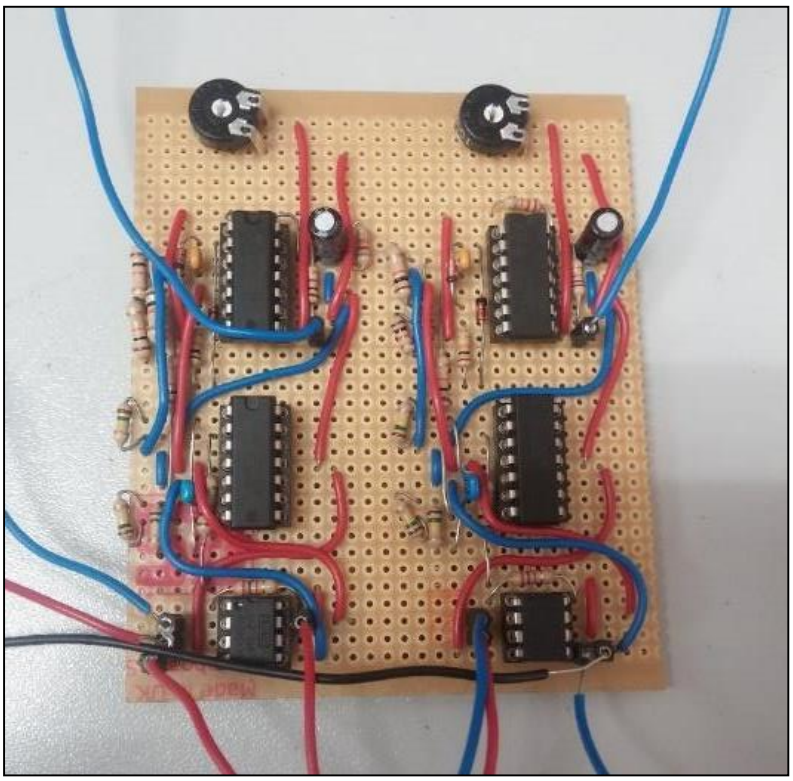


***Imagen 74**. Montaje final del mando mioeléctrico.*

2.3.6. CONEXIÓN DE LOS ELECTRODOS [8]

Una de las partes finales de la implementación del sistema será la conexión del periférico donde se colocarán los sensores que se utilizarán sobre la superficie del brazo o antebrazo, dependiendo de las necesidades del usuario.

En nuestro caso, se utiliza un tipo de electrodos pasivo convencional. Se han empleado unos parches superficiales los cuales podemos visualizar en la Imagen 75. Estos parches consisten en una superficie adhesiva y un conductor que entra en contacto con la piel (plata-cloruro de plata) con ayuda de un gel conductor que mejora el contacto entre el metal y la piel del usuario.

***Imagen 75**. Electrodos: parches superficiales.*

El diseño del sistema requiere de la conexión de cinco de estos electrodos. Todos ellos van conectados directamente a las placas de adquisición que conforman cada uno de los dos canales: dos de ellos conforman la entrada diferencial del canal uno, otros dos conforman la entrada diferencial del canal dos y por último, el quinto electrodo corresponde a la tierra o referencia de ambos canales, y se coloca en una zona bioeléctricamente nula. Dichos electrodos se disponen de la manera que se expone en la Imagen 76.

***Imagen 76**. Conexión de los electrodos.*

Se ha fabricado un periférico que nos permite conectar los electrodos con el sistema de una manera fácil y cómoda para el usuario, pero que a la vez sea efectiva y de precisión, pues cabe recordar que el rango de las señales oscila en torno a los micro-voltios. Como podemos ver en la Imagen 77, los parches encajarán perfectamente en el interior de botones automáticos metálicos, comúnmente conocidos como “corchetes”. De esta forma, se utilizarán los mismos “corchetes” como conectores entre los parches y los electrodos construidos.

***Imagen 77**. Conector de los electrodos superficiales al sistema.*

2.4. PLACA DE ALIMENTACIÓN

Esta placa se encargará de recibir la alimentación del exterior y distribuirla al resto de placas, entregando a cada una de ellas la tensión que necesite para funcionar.

Por un lado, a esta placa llegará una alimentación simple de +5V desde el Arduino, la cual emplearemos para conseguir una alimentación simétrica de ±5V. Por otro lado, también nos llegará a la placa una alimentación simple de +12V. Esto hace que dispongamos de 4 salidas en esta placa: Dos correspondientes a la alimentación simétrica de ±5V, una correspondiente a la tensión de +12V y una correspondiente a la tierra, que será común a todas las alimentaciones. Aunque la salida correspondiente a +12V es directamente la entrada, sin modificaciones, se ha querido que estas pasasen por la placa de alimentación para así tener más ordenado el sistema y que en un futuro sea más sencillo trabajar con él para realizar modificaciones.

Para obtener la alimentación de +5V proveniente del Arduino lo más conveniente sería alimentarlo por USB a través de un ordenador, ya que, si lo hiciéramos a través de un transformador conectado a la red, se nos filtraría ruido de red que podría impedir el correcto funcionamiento del sistema. Esto hace que la mejor forma de conseguir los +12V sea a través de una fuente de alimentación, dado que si empleáramos un transformador tendríamos de nuevo el problema del ruido de red mencionado antes. La alimentación sería idealmente obtenida de unas baterías, se usarían baterías de 12V y un conversor CC-CC de 12V a 5V, puesto que no podría estar cableada a un ordenador ni a una fuente de laboratorio, sin embargo, se ha decidido no emplearlas en este prototipo debido al desgaste que sufrirían por su bajo uso y al alto coste que conllevan.

Como ya hemos comentado antes, la entrada de alimentación simple de +5V proviene del Arduino (+5 voltios – tierra). Para conseguir la salida simétrica de ±5V, se utilizarán un circuito integrado de la serie TMA de la casa “Traco ElectronicAG”, las hojas de especificaciones para este integrado están disponibles en el anexo 15. Estos circuitos integrados son convertidores DC-DC de alta eficiencia que permiten conseguir diferentes voltajes a partir de uno de partida.

En la Imagen 78 se muestra el aspecto del circuito integrado, así como un esquema con la distribución de las patillas del mismo. [8]

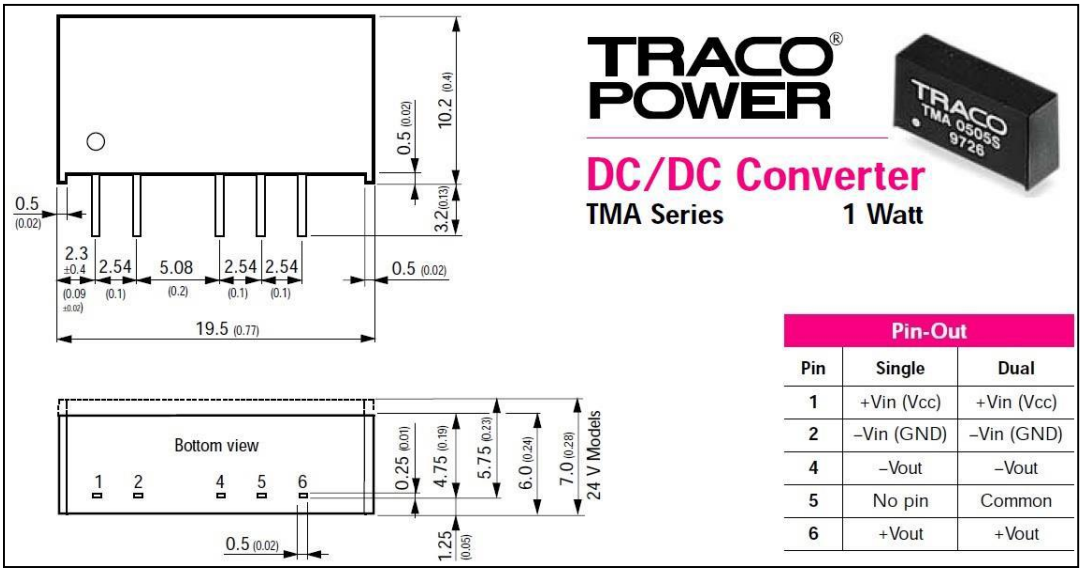


***Imagen 78**. Esquema, aspecto y entradas y salidas de los convertidores DC/DC de la serie TMA.*

Tras hacer una revisión de las necesidades de nuestro sistema se ha decidido emplear el convertidor DC-DC TMA0505D, dado que de todos los que disponemos en el laboratorio, es el que mejor se adapta a nuestras necesidades.

El acondicionamiento del dispositivo que nos ocupa es relativamente sencillo, pues tan sólo es necesario incluir un condensador en las patillas de entrada con el objetivo de eliminar el rizado de la señal de entrada. En el caso del circuito implementado se ha utilizado un condensador de 4,7µF, tal y como se indicaba en las hojas de especificación del integrado. [8]

El esquema de la placa de alimentación se puede ver en Imagen 79.

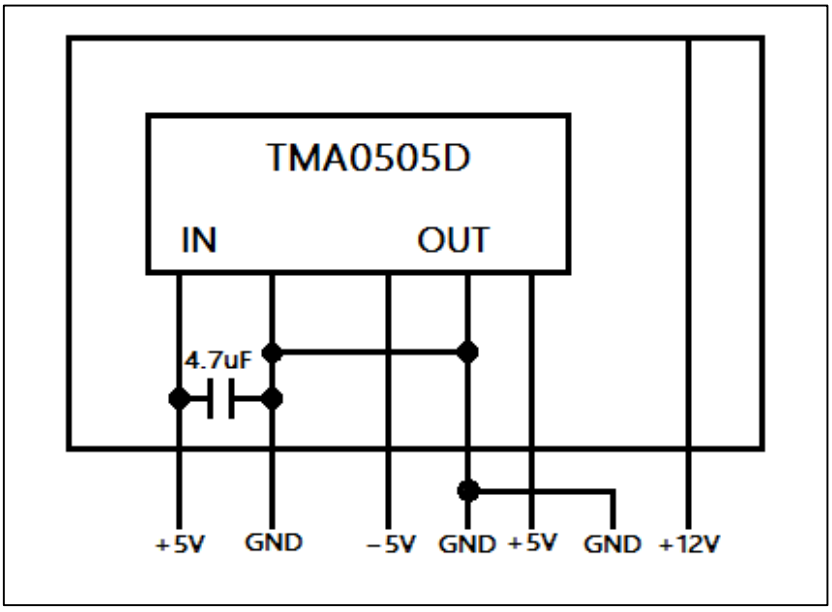


***Imagen 79**. Esquema de conexiones de la placa de alimentación.*

2.5. MONTAJE FINAL DEL SISTEMA

Una vez explicadas, en detalle y por separado, cada una de las partes que componen nuestro sistema, pasaremos a obtener una visión global del mismo. En la Imagen 80 podemos ver un diagrama de bloques de nuestro sistema, en el cual han quedado representadas todas sus partes.

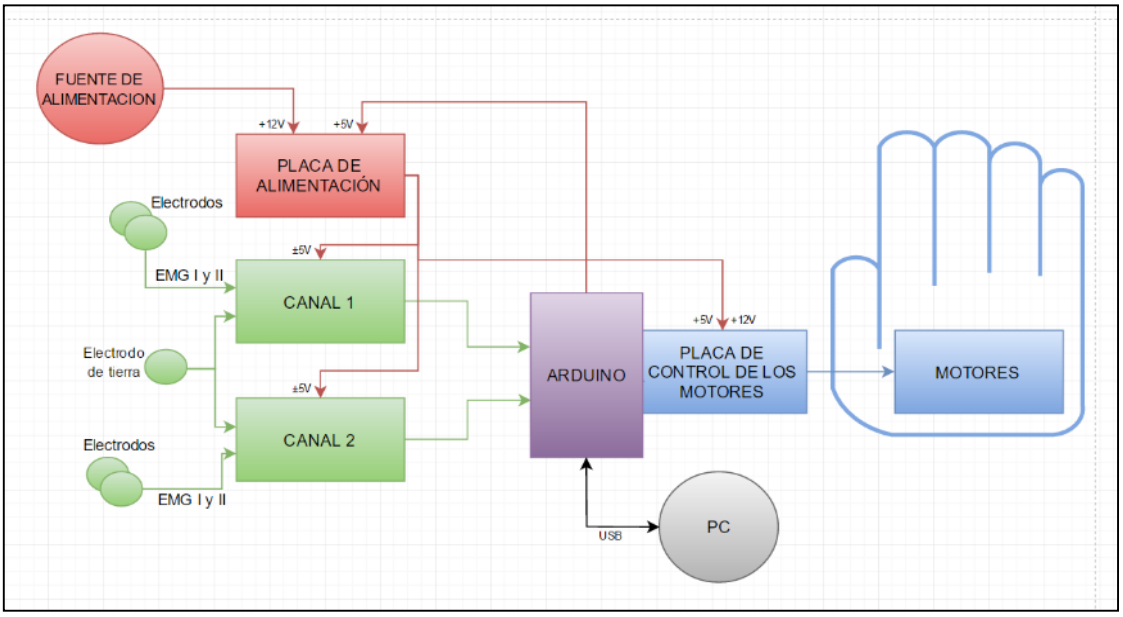


***Imagen 80**. Diagrama de bloques final del sistema.*

En primer lugar, se adquirirán señales musculares mediante 5 electrodos: Dos (Diferenciales) para cada canal y uno de tierra. Estas señales pasarán a las placas de adquisición y acondicionamiento, obteniéndose al final de cada una de ellas una señal de control limpia. Por lo tanto, contaremos con dos señales de control (Una de cada canal) que entrarán en nuestra placa Arduino para poder manejar la prótesis con ellas.

Por otro lado, el sistema cuenta, como ya hemos visto, con una placa de control de los motores. Del Arduino saldrán señales de control que llegarán a esta placa, la cual se encargará de activar unos motores u otros en función de las instrucciones que dé Arduino gracias a las mencionadas señales de control. Obviamente a esta placa están conectados todos los motores de los que dispone nuestra mano, incluido el servomotor.

Finalmente, tenemos la placa de alimentación, la cual se encarga de entregar la tensión necesaria a cada una de las placas. Por un lado, la placa de adquisición y acondicionamiento debe ser alimentada con ±5V. Mientras que, por otro lado, para alimentar la placa de control de los motores se emplearán dos voltajes simples diferentes, uno de +5V que procederá del Arduino directamente, para el servo y los integrados, y otro de +12V para los motores de DC. La placa de alimentación también se encarga de unir las tierras de las diferentes placas que conforman nuestro sistema.

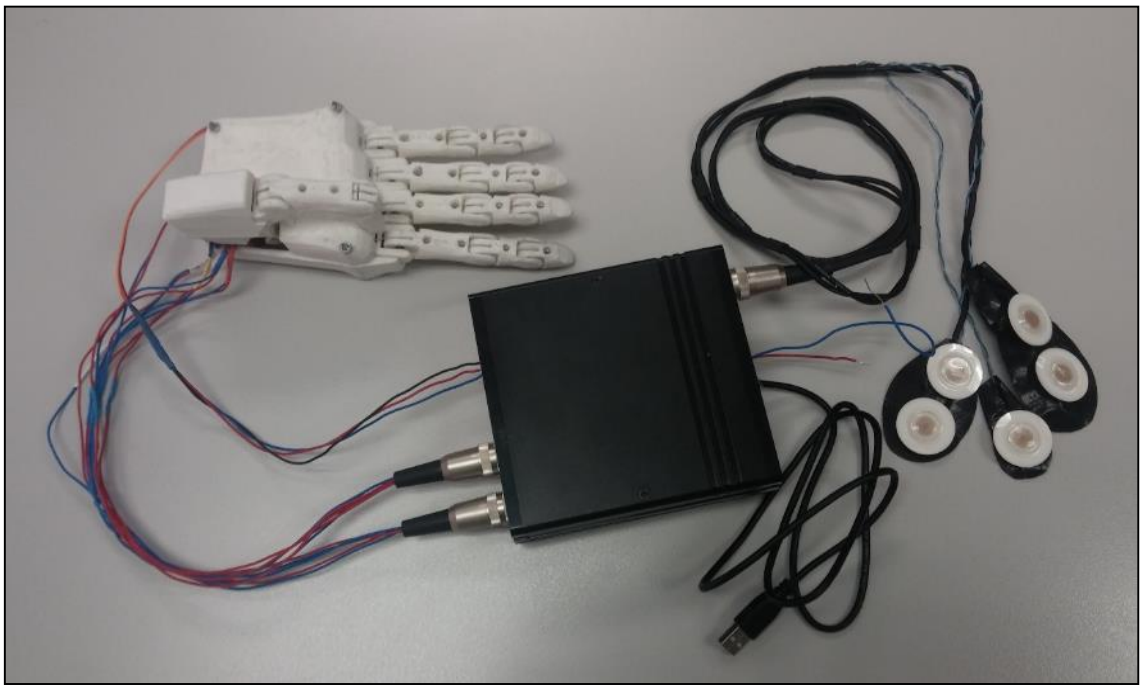


***Imagen 81**. Sistema final al completo.*

2.5.1. CONEXIONES DE LA CAJA QUE ALOJA NUESTRO SISTEMA

Toda la electrónica de nuestro sistema ha sido alojada en una caja de plástico para así protegerla de posibles averías que se pudieran producir durante su transporte y manipulación.

***Imagen 82**. Caja que contiene la electrónica.*

La caja cuenta con las siguientes conexiones con el exterior, que corresponden con las indicadas anteriormente:

- 3 conectores de 5 pines: Dos para los motores DC y uno para los electrodos.
- Un cable USB.
- 3 pines para el servo.
- 2 pines para la alimentación a 12V.

***Imagen 83**. Cara delantera de la caja.*

En la Imagen 83 podemos ver una de las caras de la caja, en la que se aprecian dos de los conectores de 5 pines y la conexión para el servo.

Es importante a la hora de conectar el servo prestar atención a la forma de conectarlo, siguiendo las indicaciones de la caja, pues de lo contrario se podría dañar el sistema. Por otro lado, los conectores de 5 pines, presentan la siguiente disposición de conexiones (Mirando los conectores de frente y con los pines en forma de U):

- El de la derecha (El más cercano al borde): Negativo del azul, positivo del azul, negativo del verde, positivo del verde y negativo del rojo
- El de la izquierda: Positivo del rojo, negativo del amarillo, positivo del amarillo, negativo del blanco y positivo del blanco.

Es importante tener en cuenta que la relación entre colores y dedos es la siguiente:

- Azul→ Índice
- Verde→ Pulgar
- Rojo→ Anular
- Amarillo→ Corazón
- Blanco→ Meñique

En el otro lado, vemos que tenemos los dos pines de alimentación, el cable USB y el conector de 5 pines de los electrodos.

***Imagen 84**.Cara trasera de la caja*

En este punto es de nuevo muy importante conectar los pines de alimentación tal como se indica en la caja, pues de lo contrario nuestro sistema se podría dañar. La distribución del conector de los electrodos es muy sencilla: Un canal diferencial a cada lado, y en el medio la tierra.

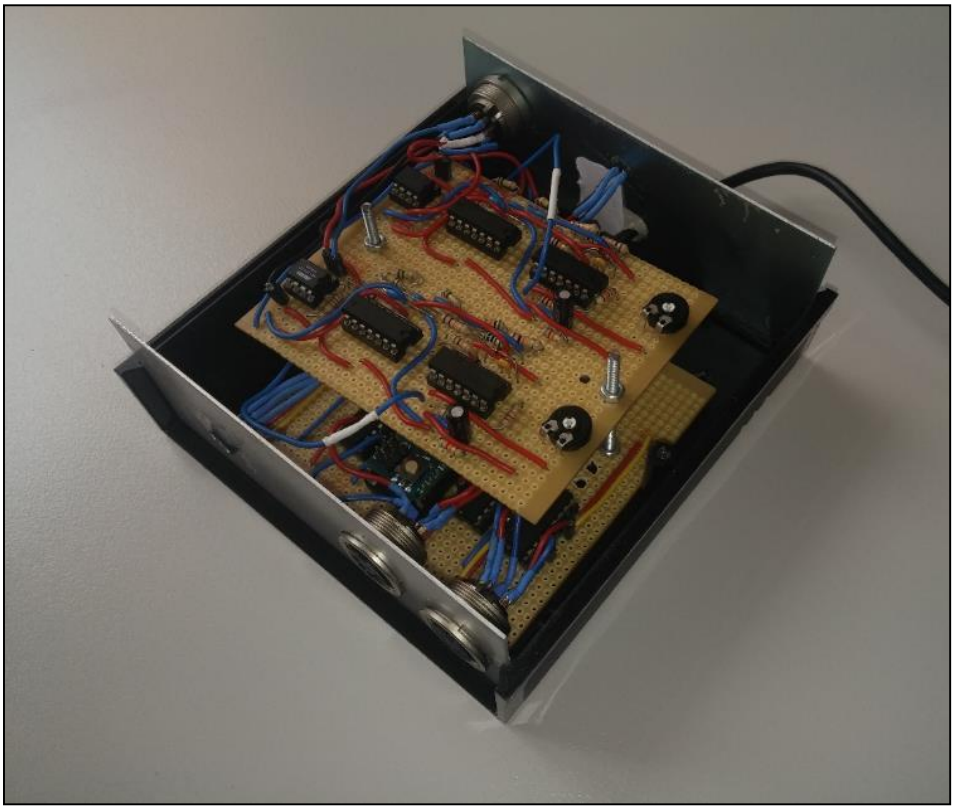


***Imagen 85**. Interior de la caja que contiene la electrónica.*

DESCRIPCIÓN DEL SOFTWARE

3.1. SISTEMA EXISTENTE E IMPLEMENTACIÓN DE ESTE EN LA NUEVA MANO [8]

El TFG del cual se está realizando la memoria en el presente documento, trata de crear una prótesis mioeléctrica a partir de una ya existente. Este es el motivo por el cual se intentará usar las aplicaciones de control que tenía la anterior mano, y adaptarlos a la nuestra. Las aplicaciones serán tomadas del PFC "Sistema de control de dispositivos externos mediante electromiograma: Aplicación a una mano robótica" desarrollado por Javier García Rodríguez en octubre de 2014 ([8]). Este apartado detalla las aplicaciones creadas en [8], por lo que con el fin de no repetir un trabajo que ya está hecho, tomaremos dichas explicaciones de [8].

3.1.1. SISTEMA DE CONTROL DE LA MANO EXISTENTE

La implementación y diseño del sistema de control que nos ocupa, implica el empleo de un determinado software. Esta parte relacionada con el software se implementará mediante la plataforma de hardware libre Arduino. Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing).

3.1.1.1. FUNDAMENTOS PREVIOS, ENTORNO Y LENGUAJE DE PROGRAMACIÓN

Las señales que tenemos a la entrada del Arduino son dos señales analógicas, correspondientes a cada uno de los dos canales. De esta forma, a la hora de confeccionar el código, debemos tener en cuenta que debemos trabajar con dos señales de entrada analógicas que han pasado previamente por un detector de envolvente, lo que implica que cada una de ellas está suavizada en el tiempo y su valor decae con relativa lentitud después de un impulso en el canal correspondiente.

Una vez somos conscientes de las limitaciones que nos imponen las señales de entrada, debemos tener en cuenta otros parámetros que nos influyen de igual forma a la hora de programar, como pueden ser las características que han de tener las señales de salida. Vamos a tener cinco señales de salida, correspondiéndose a cada uno de los cinco servomotores que accionarán los dedos de la mano robótica.

Por tanto, debemos crear, a través del código, cinco salidas que actúen sobre los servomotores, y de esta manera, han de tener una serie de características especiales. Los servomotores se controlan mediante impulsos de ancho variable que deben refrescarse periódicamente. Esto significa que, si dejamos de enviar la señal de control en el tiempo en el que el servomotor lo necesita, éste (a pesar de estar alimentado) dejará de mantenerse en la posición preestablecida y adoptará cualquier orientación regida por el esfuerzo mecánico al que esté sometido. Es decir, si no mantenemos la señal de control en forma efectiva todo el tiempo que sea necesario, el sistema quedará a merced de las fuerzas externas a la que sea sometido.

El refresco se realiza habitualmente con una frecuencia de 50 veces por segundo, pero es normal y efectivo trabajar entre los 10 y los 30 milisegundos, tal como muestra la Imagen 86. Por otro lado, el ancho del impulso, es decir, su tiempo de duración, dará la posición u orientación del actuador mecánico.

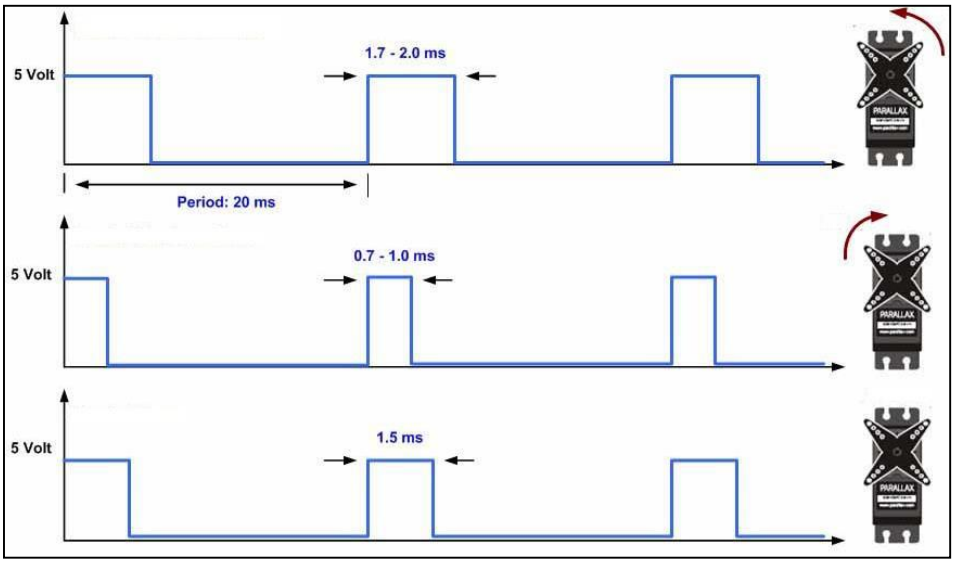


***Imagen 86**. Diagrama de tiempos de las señales de control de los servomotores. Eje voltaje - frecuencia.*

La plataforma libre Arduino posee una librería dedicada a las señales de los servomotores, proporcionándonos una serie de comandos realmente útiles para nuestra aplicación, que nos permitirán, entre otras cosas, seleccionar los grados exactos que ha de virar el servomotor o asociar con un pin concreto de salida.

3.1.1.2. ENTORNO DE PROGRAMACIÓN

Como siempre que necesitamos implementar un software en un determinado lenguaje de programación, necesitamos utilizar un entorno de dicho lenguaje. En nuestro caso empleamos el entorno propio de la plataforma de hardware y software libre Arduino. Dicho entorno lo podemos descargar de la página web de Arduino, en la sección de software (https://www.arduino.cc/en/Main/Software). En la Imagen 87 podemos observar el entorno de programación que empleamos, indicando cada una de las principales partes del mismo.

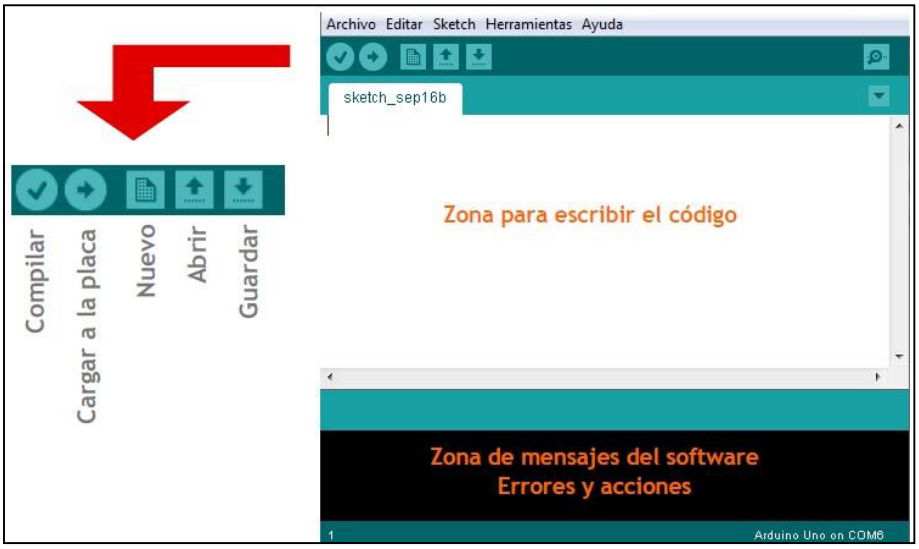


***Imagen 87**. Entorno de programación Arduino.*

3.1.1.3. LENGUAJE DE PROGRAMACIÓN

La plataforma Arduino se programa mediante el uso de un lenguaje propio basado en el lenguaje de programación de alto nivel Wiring. Aunque es posible utilizar otros lenguajes de programación y aplicaciones populares en Arduino, debido a que Arduino usa la transmisión serial de datos soportada por la mayoría de los lenguajes mencionados, en nuestro proyecto emplearemos el lenguaje propio de la plataforma. Para los que no soportan el formato serie de forma nativa, es posible utilizar software intermediario que traduzca los mensajes enviados por ambas partes para permitir una comunicación fluida.

3.1.2. DISEÑO DE LAS APLICACIONES

Se diseña e implementa el código necesario para crear dos aplicaciones independientes, con características y modos de funcionamiento diferentes. Una de ellas será una aplicación más funcional, cuyo perfeccionamiento del manejo del sistema requiere menos entrenamiento. La otra aplicación será una aplicación más intuitiva para los potenciales usuarios del sistema, que serán en su mayor parte, personas con algún tipo de distrofia muscular en el brazo o un grado de amputación del mismo.

3.1.2.1. CONTROL DE POSICIONES

La aplicación software “Control de posiciones” es la más versátil y eficaz de las dos diseñadas. El sistema está provisto por dos canales de carácter diferencial. Así pues, la aplicación software que nos ocupa se basará en elegir una posición por uno de los canales, y ejecutar la acción (o no) mediante el otro.

3.1.2.1.1. DIAGRAMA DE FLUJO

Se dotará a la aplicación software que nos ocupa de un conjunto de posiciones entre las que el usuario podrá seleccionar. Cada una de estas posiciones permitirá al sujeto ejecutar la acción asociada a la mencionada posición. Como se trata de un prototipo y la importancia del mismo recala en la comprobación del perfecto funcionamiento y manejabilidad del sistema, optamos por que sean tres las opciones que pueden ser elegidas por el usuario. Si bien cabe mencionar, tal y como podremos comprobar con posterioridad con el código presente, bastaría con copiar

unas líneas de código y modificar los parámetros pertinentes a la parte mecánica para incrementar el número de posiciones que se deseen.

Al Arduino llegan las señales analógicas de manera continua. Se adquirirán y adaptarán para poder procesarlas a tiempo real, se comprobará el valor de cada uno de los canales y se actuará en consecuencia, cambiando la posición en el caso del canal dos, y ejecutando la acción en el caso del canal 1. De esta manera, el potencial usuario podrá seleccionar una de las tres posiciones de las que está provisto el sistema mediante el canal 2, que preferiblemente irá conectado al músculo extensor o supinador. Así mismo, mediante el canal 1, el usuario será capaz de ejecutar, o no, la acción asociada a la posición actual que haya elegido.

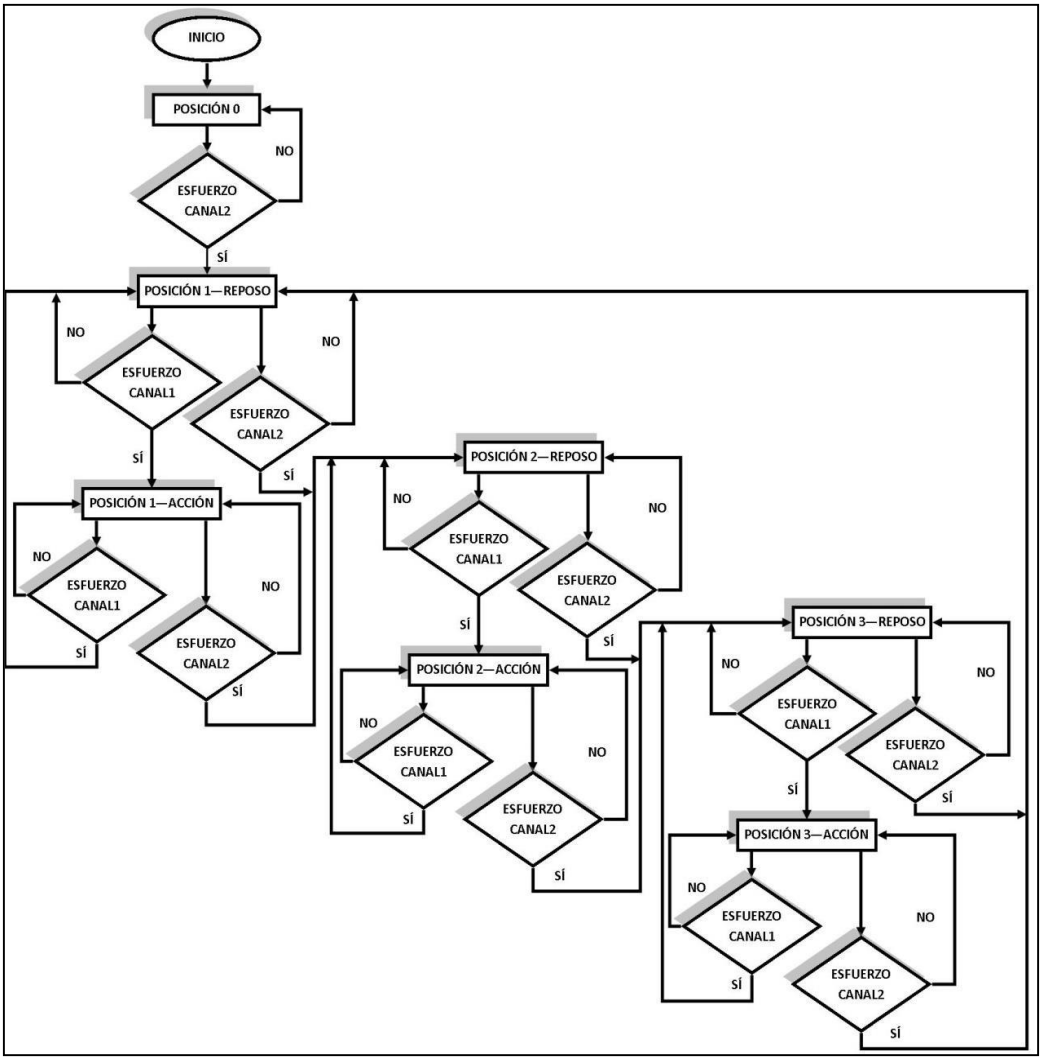


***Imagen 88**. Control de posiciones: diagrama de flujo.*

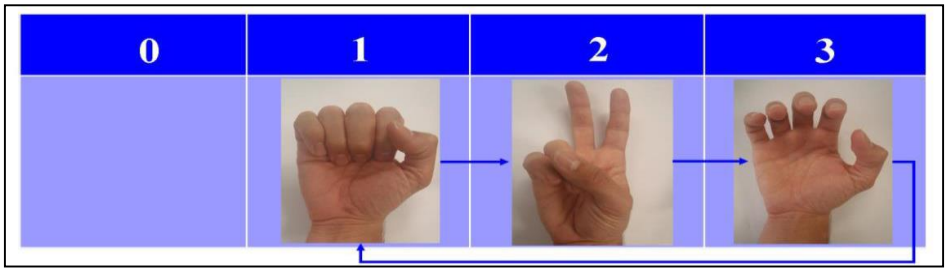


***Imagen 89**. Posiciones efectivas de la aplicación 'Control de posiciones'*

3.1.2.1.2. DESARROLLO DEL CÓDIGO

La dificultad de implementar la aplicación que nos ocupa no reside tanto en uso de comandos complejos sino más bien en tener clara la idea y realizar una buena estructura de la misma.

Si bien hemos comentado que el leguaje de Arduino se basa en el lenguaje de programación de alto nivel Wiring, a la hora de programar resulta similar a programar en C. Como en multitud de lenguajes de programación, necesitamos incluir, en primer lugar, las librerías que sean necesarias a la hora de desarrollar el código. En este caso particular, necesitaremos incluir la librería correspondiente a los servomotores, que nos facilitará la implementación del código, ya que nos ofrece unos comandos que nos permiten la completa manejabilidad de los servomotores.

```
#include <Servo.h>    // Incluimos la librería dedicada a Servo-motores
```

La librería anterior nos permite declarar variables del tipo “Servo”, tal y como se muestra en el siguiente trozo de código:

```
Servo myservo1;      // Declaramos 5 variables tipo 'Servo'
Servo myservo2;
Servo myservo3;
Servo myservo4;
Servo myservo5;
```

Seguidamente, debemos declarar el resto de variables que se utilizarán en el código, que son del tipo entero. Entre otras variables, las correspondientes al led indicador de encendido, las variables correspondientes a los datos entrantes de cada uno de los canales, la variable que indica la posición actual y una variable auxiliar que nos facilitará la ejecución de las acciones.

```
int led1=13;       // Declaramos variable para el led de encendido
int val1 = 0;      // Declaramos dos variables asociados a cada una de
                      los dos canales de entrada
int val2 = 0;
int flag1=0;       // Declaramos los flags que nos servirán a la hora
                      de cambiar de posición
int flag2=0;
int posicion1=0;   // Declaramos la variable correspondiente a posición
```

Llegados a este punto debemos destacar que la estructura de un código con el lenguaje Arduino es algo particular, ya que, tras la declaración de las librerías y de las variables pertinentes, corresponde el turno de la parte de “setup()”. Cada vez que se alimente el sistema, será la parte que primero se ejecute. También procederá su ejecución si se pulsa “Reset” en cualquier instante de tiempo. En dicha zona del código debemos asociar cada variable con el pin de salida pertinente en el caso de las variables “Servo” e indicar el modo y el valor en el caso del led de encendido. Así mismo estableceremos la velocidad de datos en bits por segundo (baudios) para la transmisión de datos en serie, en nuestro caso 9600, y también haremos que se muestre por la pantalla serial del entorno la posición y la palabra “posición”.

```
void setup() {    // Cada vez que pulsemos reset, se   inicializará la
                     rutina

 Serial.begin(9600);
 Serial.print(posicion1);
 Serial.print(" posicion  ");


   myservo1.attach(11);   // Liga el servo con el pin de salida
                             adecuado
   myservo2.attach(10);
   myservo3.attach(9);
   myservo4.attach(6);
   myservo5.attach(5);

   pinMode(led1,OUTPUT);  // Definimos el led de encendido como salida
   digitalWrite(led1,HIGH);  // Hacemos que el led este encendido al
                                iniciarse el programa
 }
```

Tras la zona de “setup()” o de inicio, aparece la parte de “loop()”. Se trata del grueso del programa, la parte que se repite cíclicamente en forma de bucle y donde escribimos las diferentes acciones que se deben suceder a lo largo de la ejecución de la aplicación.

Tras el código correspondiente a la adquisición y mapeo de las señales de entrada, abrimos un ‘switch’ con las tres posibles posiciones que hay disponibles (cuatro si contamos con la posición inicial de alimentación). Dentro de la zona de descripción de cada una de las mismas, hay una consecución de condiciones que habilitarán la acción, o no, ayudándonos de uno de las canales de entrada y una variable auxiliar o ‘flag’, que nos indicará de qué estado anterior venimos para activar o desactivar la acción según corresponda.

Para el desarrollo del código correspondiente a esta parte, empleamos la estructura que reflejamos en el siguiente fragmente de pseudo-código:

```
void loop() {

--Adquisición y mapeo de las señales de entrada

--Descripción monitor serial

if(val2<150){ }

else if(val2>150 && posicion1!=3){
   --Incrementamos posición;
 }

else if(val2>150 && posicion1==3){
   --Igualamos posición a 1;
}

  switch (posicion1){
        case 0:
                  --Acción que queremos que adquiera cada servo al
                  --alimentar el sistema
        break;

        case 1:            // Código pertinente a la posición 1
                  if(val1<150 && flag1==0) {
                    --Acciones para indicar el ángulo que queremos que
                      adquiera cada servo en la posición1 y acción off
                  }
```

```
            else if(val1<150 && flag1==1) {
              --Acciones para indicar el ángulo que queremos que
                adquiera cada servo en la posición1, cuando no se
                activa el valor1 y estamos con la acción ON. Se
                mantendrá la acción ON
            }

            else if(val1>150 && flag1==0) {
              --Acciones para indicar el ángulo que queremos que
                adquiera cada servo en la posición1, cuando se
                activa el valor1 y estamos con la acción OFF. Se
                ejecutará la acción ON
            }

            else if(val1>150 && flag1==1) {
              --Acciones para indicar el ángulo que queremos que
                adquiera cada servo en la posición1, cuando  se
                activa el valor1 y estamos con la acción ON. Se
                ejecutará la acción OFF
            }
        break;
        case 2:            // Código pertinente a la posición 2
        ---------------------
        break;

        case 3:            // Código pertinente a la posición 3
        ---------------------
        break;}}
```

A continuación, se muestra el código correspondiente a uno de los casos particulares del pseudo-código anterior, es decir, al código correspondiente a una de las posiciones, en el que podemos ver su estructura interna y sus correspondientes sentencias:

```
case 1:                    // Código pertinente a la posición 1
        if(val1<150 && flag1==0) { // Indicamos el ángulo que
        queremos que adquiera cada servo en la posición1 y
        acción OFF

        myservo1.write(160);
        delay(20);
        myservo2.write(180);
        delay(20);
        myservo3.write(0);
        delay(20);
        myservo4.write(0);
        delay(20);
        myservo5.write(135);
        delay(20);

        flag1=0;

        delay(500);  // Retraso necesario para que se ejecute
        la acción y no seguir adquiriendo valores
        }

        else if(val1<150 && flag1==1) {  // Indicamos el que
        ángulo queremos que adquiera cada servo en la
        posición1, cuando no se activa el valor1 y estamos
        con la acción ON. Se mantendrá la acción ON
```

```
                myservo1.write(20);
                delay(20);
                myservo2.write(0);
                delay(20);
                myservo3.write(120);
                delay(20);
                myservo4.write(150);
                delay(20);
                myservo5.write(73);
                delay(20);

                flag1=1;

                delay(500);
                }

                else if(val1>150 && flag1==0) {  // Indicamos el que
                ángulo queremos que adquiera cada servo en la
                posición1, cuando se activa el valor1 y estamos
                con la acción OFF. Se ejecutará la acción ON

                myservo1.write(20);
                delay(20);
                myservo2.write(0);
                delay(20);
                myservo3.write(120);
                delay(20);
                myservo4.write(150);
                delay(20);
                myservo5.write(73);
                delay(20);

                flag1=1;

                delay(500);
                }

                else if(val1>150 && flag1==1) {  // Indicamos el que
                ángulo queremos que adquiera cada servo en la
                posición1, cuando se activa el valor1 y estamos
                con la acción ON. Se ejecutará la acción OFF

                myservo1.write(160);
                delay(20);
                myservo2.write(180);
                delay(20);
                myservo3.write(0);
                delay(20);
                myservo4.write(0);
                delay(20);
                myservo5.write(135);
                delay(20);

                flag1=0;

                delay(500);
                }
        break;
```

Como podemos apreciar, dentro de cada una de las cuatro posibles condiciones, empleamos la sentencia 'nombre_variable_servo.write()' para indicar, en el interior de cada uno de los paréntesis, a cada uno de los cinco servos, el ángulo que ha de tomar. Tras cada una de las sentencias anteriores, incluimos un retraso (delay()) para proporcionar al Arduino el tiempo necesario para accionar cada uno de los servos de una manera fluida, pero teniendo siempre

presente que han de ser lo suficientemente pequeños como para que la sensación a tiempo real sea que los dedos se mueven de manera simultánea. Se incluye el retraso dentro del paréntesis y se mide en milisegundos.

A la hora de indicar el ángulo de un servomotor determinado en cada uno de las posiciones y acciones, debemos tener en cuenta el sentido físico del mismo. Cada servomotor tendrá una localización y estará orientado de manera diferente. Por tanto, si por ejemplo, queremos tener todos los dedos de la mano en reposo o mano abierta, los ángulos deberán ser distintos, tal y como se muestra a continuación:

```
myservo1.write(160);
delay(20);
myservo2.write(180);
delay(20);
myservo3.write(0);
delay(20);
myservo4.write(0);
delay(20);
myservo5.write(135);
delay(20);
```

El código completo de esta aplicación está disponible en el anexo 3.

3.1.2.2. CONTROL DE ESTADOS

La aplicación software “Control de estados”, o “Control de niveles”, es la más intuitiva y natural de las dos propuestas. El sistema posee dos canales de carácter diferencial, dependiendo el nivel de señal que se aplique por cada uno de los canales, se le asignará a cada uno de ellos un estado de tres posibles

Denominaremos a los tres niveles de cada canal, dependiendo de su nivel de intensidad, como: ‘bajo (L)’, ‘medio (M)’ y ‘alto (H)’. De esta manera, jugaremos con la combinación de estados de ambos canales, ofreciendo un abanico de nueve diferentes composiciones de estados.

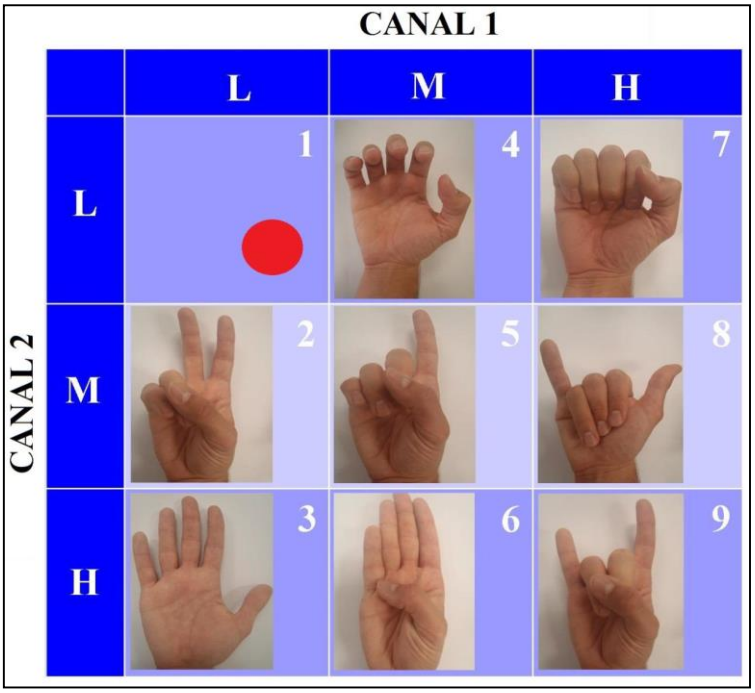


***Imagen 90**.. Estados y combinación de niveles para la aplicación 'Control de estados'.*

Podemos percibir la bola roja como la combinación del grado de intensidad que el usuario aplica sobre los canales. Para que la aplicación acceda a un nivel determinado, se establece un tiempo mínimo de estancia en el mismo. Una vez se accede a un nivel, de los nueve posibles, aunque el usuario permanezca en posición de reposo (o lo que es lo mismo, el nivel bajo-bajo (L-L)), se mantendrá la posición, o lo que es lo mismo, el estado (L-L) es la posición de 'no operación'.

Para impedir que pasemos por los niveles anteriores a uno determinado que queremos alcanzar, establecemos unos tiempos de permanencia en el nivel tal y como hemos mencionado. Se trata de una aplicación cerrada, difícilmente escalable. Si bien cabe destacar que la mayor ventaja de la misma, es que su uso, por parte de los potenciales usuarios, es intuitivo y natural, aunque por el contrario requiera un tiempo de entrenamiento para su perfecto control

3.1.2.2.1. DIAGRAMA DE FLUJO

Tras alimentar el dispositivo y acceder al nivel inicial, se comprueba en bucle el estado en el que estamos, si supera el tiempo de permanencia mínimo en dicho estado, se acciona la acción correspondiente al mismo. En caso contrario, de que no supere el tiempo mínimo establecido, se seguirá comprobando el estado actual de cada momento.

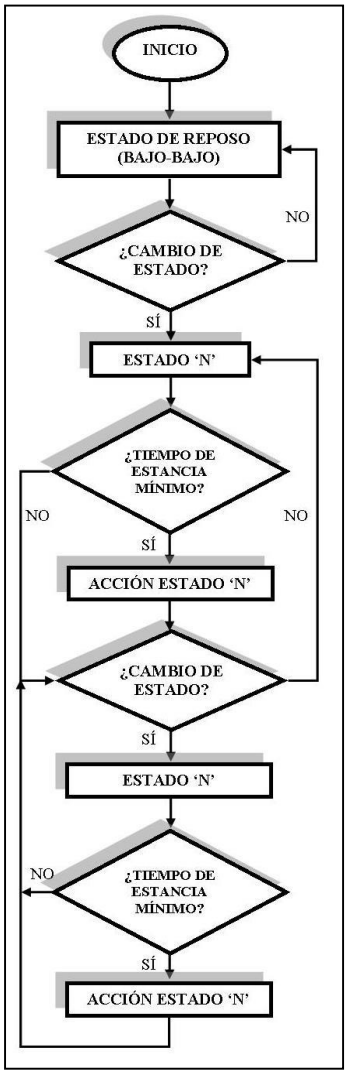


***Imagen 91**. Control de estados: diagrama de flujo.*

3.1.2.2.2. DESARROLLO DEL CÓDIGO

La dificultad de implementar la aplicación que nos ocupa no reside tanto en uso de comandos complejos, sino más bien en tener clara la idea de lo que se quiere hacer y de este modo poder estructurar correctamente la aplicación.

Si bien hemos comentado que el leguaje de Arduino se basa en el lenguaje de programación de alto nivel Wiring, a la hora de programar resulta similar a programar en C. Como en multitud de lenguajes de programación, necesitamos incluir en primer lugar, las librerías que sean necesarias a la hora de desarrollar el código. En este caso particular, necesitaremos incluir la librería correspondiente a servomotores, que nos facilitará la implementación del código, ya que nos ofrece unos comandos que nos permiten manejar el servomotor de una forma muy sencilla.

```
#include <Servo.h>   // Incluimos la librería dedicada a Servo-motores
```

La librería anterior nos permite declarar variables del tipo “Servo”, tal y como se muestra en el siguiente trozo de código:

```
Servo myservo1;     // Declaramos 5 variables tipo 'Servo'
Servo myservo2;
Servo myservo3;
Servo myservo4;
Servo myservo5;
```

Seguidamente, debemos declarar el resto de variables que se utilizarán en el código, que son del tipo entero. Entre otras variables, las correspondientes al led indicador de encendido, las variables correspondientes a los datos entrantes de cada uno de los canales, la variable que indica la posición actual y una variable auxiliar que nos facilitará la ejecución de las acciones.

En la sección dedicada a la aplicación anterior se explicó la declaración de las librerías pertinentes y las principales variables, como por ejemplo, las específicas de los servomotores. Procedemos a explicar las principales diferencias, en cuanto a sentencias, así como de su estructura global, en referencia a la aplicación software explicada anteriormente.

Tal y como podemos apreciar a continuación, las primeras diferencias vienen al declarar un nuevo tipo de variable, en este caso del tipo “unsigned long”, que nos servirán para declarar las variables temporales, que tendrán una vital importancia de cara al perfecto funcionamiento del sistema en cuanto a los tiempos de estancia. Dichas variables temporales nos permitirán establecer unos tiempos de permanencia mínimos en cada estado, a partir de los cuales, se accionará la acción asociada a dicho estado (exceptuando el estado de reposo, L-L).

```
int estadoActual=0;   // Declaramos las variables estadoActual y
                         estadoAnterior, pues lo implementaremos como
                         una máquina de estados
int estadoAnterior=0;
```

```
unsigned long tiempoEstado;  // Declaramos las variables tiempoActual
                                y tiempoAnterior para asociar el
                                tiempo a cada estado
unsigned long tiempoActual;

unsigned long mostrar=0;  // Variable auxiliar para controlar las
                             impresiones en el monitor serial
```

Para conseguir implementar los nueve diferentes estados, declararemos un 'switch' con tres diferentes opciones para emular los tres posibles estados del canal 1, y dentro de cada una de las mencionadas opciones, un 'switch' con otras tres diferentes opciones para emular los tres posibles estados del canal 2. De esta manera conseguimos la premisa de los nueve estados diferentes. A continuación, intentamos ofrecer una visión general de la aplicación mediante un sencillo pseudo-código:

```
#declaración de librerías

Declaración de variables

void setup() {

      --Asociación de pines
      --Posición inicial
}

void loop() {

      --Asociación de entradas y mapeo
      --Impresión monitor serial de las valores de entrada cada 200ms

  switch (val1){
  case 0 ... 70:                      // NIVEL BAJO CANAL1

    switch (val2){
    case 0 ... 70:                    // NIVEL BAJO-BAJO
    --
    break;

    case 71 ... 220:                  // NIVEL BAJO-MEDIO
    --
    break;

    case 221 ... 300:                    // NIVEL BAJO-ALTO
    --
    break;
    }
    break;


  case 71 ... 220:                    // NIVEL MEDIO CANAL1

    switch (val2){
    case 0 ... 70:                    // NIVEL MEDIO-BAJO
    --
    break;

    case 71 ... 220:                  // NIVEL MEDIO-MEDIO
    --
    break;
```

```
    case 221 ... 300:                    // NIVEL MEDIO-ALTO
    --
    break;
    }
    break;


  case 221 ... 300:                      // NIVEL ALTO CANAL1

    switch (val2){
    case 0 ... 70:                       // NIVEL ALTO-BAJO
    --
    break;

    case 71 ... 220:                     // NIVEL ALTO-MEDIO
    --
    break;

    case 221 ... 300:                    // NIVEL ALTO-ALTO
    --
    break;
    }
    break;

  break;

  default: break;
  }
  }
```

En la parte central o cíclica del código, la parte correspondiente al loop(), antes de comenzar con la parte correspondiente a la implementación de los estados, incluimos una condición para imprimir por pantalla los valores mapeados de cada uno de los canales, con una frecuencia de 200 milisegundos, para que, si el usuario lo desea, puede observarlos mediante el monitor serial. Se muestra a continuación:

```
if((millis()-mostrar)>200

    {
    Serial.println(val1);
    Serial.println(val2);

    mostrar=millis();
    }
```

Procede indicar que la anterior función no desborda con facilidad, de hecho, es muy complicado, pues la función “millis()” reinicia su contador al cabo de 50 días. Una vez descritas las principales variables y haber dado una visión global de la implementación del código, se procede a describir el código correspondiente a los estados cuatro, cinco y seis.

```
  case 71 ... 220:              // NIVEL MEDIO CANAL 1
    switch (val2){


    case 0 ... 70:              // NIVEL MEDIO-BAJO

                                // Supongo que el estado en el que
                                   entramos es el 4.Actualización de
                                   variables para todos los case.
      if(estadoActual != 4)
      {
        estadoAnterior = estadoActual;  // Actualizamos estadoAnterior
                                           y estadoActual
        estadoActual = 4;
        tiempoEstado = millis();
      }
```

```
  tiempoActual = millis();
  if((tiempoActual - tiempoEstado) > 200)// Comprobamos que si
                                         han pasado más de
                                         200ms y ejecutamos
                                         acción correspondiente
                                         al estado 4
  {
            myservo1.write(60);         // Instrucciones
                                         correspondientes a la
                                         acción del estado 4
            delay(20);
            myservo2.write(35);
            delay(20);
            myservo3.write(100);
            delay(20);
            myservo4.write(120);
            delay(20);

            myservo5.write(73);
            delay(20);

  Serial.print("MEDIO-BAJO");    // Imprimimos por el monitor
                                  serial, el estado que
                                  ejecutamos
  delay(1000);
  }
  break;

case 71 ... 220:          // NIVEL MEDIO-MEDIO

  if(estadoActual != 5)
  {
    estadoAnterior = estadoActual;
    estadoActual = 5;
    tiempoEstado = millis();
  }
  tiempoActual = millis();
  if((tiempoActual - tiempoEstado) > 200)
  {

            myservo1.write(20);
            delay(20);
            myservo2.write(20);
            delay(20);
            myservo3.write(120);
            delay(20);

            myservo4.write(0);
            delay(20);
            myservo5.write(73);
            delay(20);

  Serial.print("MEDIO-MEDIO");
  delay(1000);
  }
  break;

case 221 ... 300:         // NIVEL MEDIO-ALTO

  if(estadoActual != 6)
  {
    estadoAnterior = estadoActual;
    estadoActual = 6;
    tiempoEstado = millis();
  }
  tiempoActual = millis();
```

```
      if((tiempoActual - tiempoEstado) > 200)
      {

                  myservo1.write(160);
                  delay(20);
                  myservo2.write(160);
                  delay(20);
                  myservo3.write(0);
                  delay(20);
                  myservo4.write(0);
                  delay(20);
                  myservo5.write(73);
                  delay(20);
      Serial.print("MEDIO-ALTO");
      delay(1000);
      }
      break;
    }
    break;
```

Al final del código, para evitar cualquier tipo de error en las combinaciones, incluimos la sentencia "default: break"; que para cualquier combinación no descrita entre las anteriores, sale del 'switch' y seguimos tomando valores de entrada. En casa de no incluirlo dicha sentencia, si por ejemplo, por uno de los canales alcanzáramos el valor de 300, el programa quedaría bloqueado.

```
default: break;
```

El código completo de esta aplicación está disponible en el anexo 4.

3.1.3. IMPLEMENTACIÓN DEL ANTERIOR SISTEMA EN LA NUEVA MANO

Una vez descrito y entendido el sistema anterior, toca hacer una evaluación viendo los aspectos positivos y negativos que tendría el emplear este sistema en nuestra mano, en detrimento de crear nosotros uno desde cero.

En primer lugar, cómo ventaja principal, mencionar el tiempo que ahorraríamos al no necesitar crear el programa, ni realizar pruebas para depurarle etc. En conclusión, la principal ventaja de usar una de las aplicaciones ya existentes es el tiempo que ahorraríamos.

Por otro lado, es importante también mencionar las desventajas, pues también tienen bastante peso. En primer lugar, la mano anterior se accionaba mediante servomotores, mientras que con la que estamos trabajando ahora emplea motores DC, por lo que aquí encontramos un punto de divergencia, en el cual deberemos reemplazar parte del código existente por uno creado por nosotros. Además de esto, es importante mencionar que puede que los umbrales para detectar los diferentes niveles del electromiograma puede que varíen al variar el sujeto que maneja la prótesis. Esto implica realizar nuevas pruebas para detectar dichos umbrales y establecerlos correctamente. Otra desventaja que cabe destacar es que el enfoque que tiene esta nueva mano es distinto a la anterior. La nueva mano tiene más fuerza y pretende ser más funcional que la anterior, que tan solo pretendía ser un prototipo que posteriormente sería mejorado.

Con ambas, ventajas y desventajas sobre la mesa, vemos que emplear el código anterior para nuestra mano tiene más puntos desfavorables que favorables. La principal ventaja que nos ofrecía el empleo de las antiguas aplicaciones era un gran ahorro de tiempo, sin embargo, dicho ahorro de tiempo, se ve reducido de manera importante debido a dos factores: La necesidad de

implementar mejoras sobre el código ya existente y la obligación de crear unos códigos, por pequeños que sean, para realizar las distintas pruebas, pudiendo servir estos códigos como base para crear las aplicaciones definitivas.

Teniendo en cuenta todo lo visto anteriormente, se decidió no emplear el código existente y crear uno nuevo desde cero. Sin embargo, a pesar de no emplear el código antiguo para manejar nuestra mano, sí que se realizarán varias consultas en él durante la creación del nuevo código.

3.2. NUEVO SISTEMA

A diferencia de en [20], en el presente TFG tan sólo se va a desarrollar una aplicación para el manejo de la mano. Sin embargo, si en un futuro se decide desarrollar una nueva aplicación, en este documento está toda la información necesaria para ello.

Dicha aplicación está basada en la aplicación 'Control de estados' desarrollada en [20], aunque presenta las suficientes diferencias respecto a ella cómo para considerarlas aplicaciones diferentes.

3.2.1.1. APLICACIÓN DE CONTROL DE LA PRÓTESIS

La aplicación desarrollada para controlar nuestra prótesis es muy intuitiva y natural. El sistema posee dos canales de carácter diferencial, dependiendo el nivel de señal que se aplique por cada uno de los canales, se le asignará a cada uno de ellos un estado de tres posibles.

Denominaremos a los tres niveles de cada canal, dependiendo de su nivel de intensidad, como: 'Bajo' (L), 'Medio' (M) y 'Alto' (H). De esta manera, jugaremos con la combinación de estados de ambos canales, ofreciendo así un abanico de nueve diferentes composiciones de estados.

Para que la aplicación acceda a un nivel determinado, se establece un tiempo mínimo de estancia en el mismo. Una vez se accede a un nivel de los nueve posibles, aunque el usuario permanezca en posición de reposo (o lo que es lo mismo, el nivel Bajo-Bajo (L-L)), se mantendrá la posición. Es decir, el estado (L-L) es la posición de 'No operación'.

Se trata de una aplicación cerrada y difícilmente escalable. Si bien cabe destacar que la mayor ventaja de la misma, es que su uso por parte de los potenciales usuarios, es intuitivo y natural, aunque por el contrario requiera un cierto tiempo de entrenamiento para su perfecto control.

3.2.1.1.1. MOVIMIENTOS DISPONIBLES

A continuación, veremos en detalle que acciones se realizarán al acceder a cada uno de los 9 estados de los que dispone nuestra mano:

1. **ESTADO BAJO-BAJO**: Este estado es el de 'reposo' o 'no operación'. Cuando se está en él no se realizará ninguna acción. Tan solo se enviará por el puerto serie el mensaje "Estado de reposo".
2. **ESTADO BAJO-MEDIO**: Aquí se realizará una prueba de todos los motores, incluido el servo, para poder comprobar el correcto funcionamiento de estos. Se irán abriendo y cerrando cada uno de los dedos, individualmente, para finalmente probar el servomotor, haciéndole moverse entre las dos posiciones que emplearemos.
3. **ESTADO BAJO-ALTO**: En este estado se realizará un agarre suave, con dos dedos. Se cerrarán tan solo los dedos índice y pulgar (El cual estará situado enfrente del resto de dedos), y se mantendrá así la mano hasta que se ejecute una nueva orden. Este estado presenta un gran fallo y es que los dedos no se detienen al detectar contacto con el objeto a coger. Esto hace que el agarre no se realice satisfactoriamente en muchos casos, y nos lleva a necesitar un sensor de contacto que nos solucione este problema. Como ya hemos dicho, no se ha podido implementar dicho sensor por falta de tiempo, por ello se puede considerar que este estado ha quedado en cierto modo pendiente de finalizar.

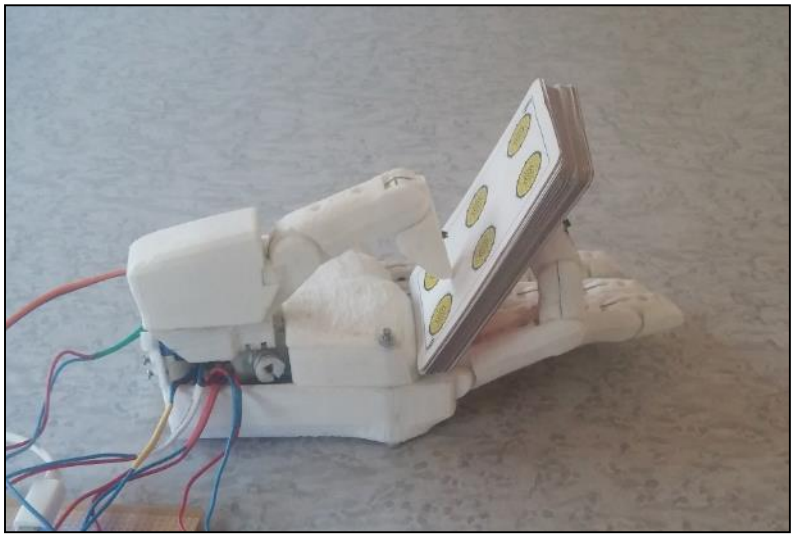


***Imagen 92**. Prótesis ejerciendo el agarre suave.*

4. **ESTADO MEDIO-BAJO**: Es muy posible que este estado sea el más empleado de todos, ya que es el más funcional. Consta simplemente de agarrar algo con toda la fuerza que los motores pueden desarrollar, manteniendo esa posición hasta que se ejecute otro estado. Se cerrarán todos los dedos simultáneamente, estando el pulgar situado enfrente del resto de dedos.

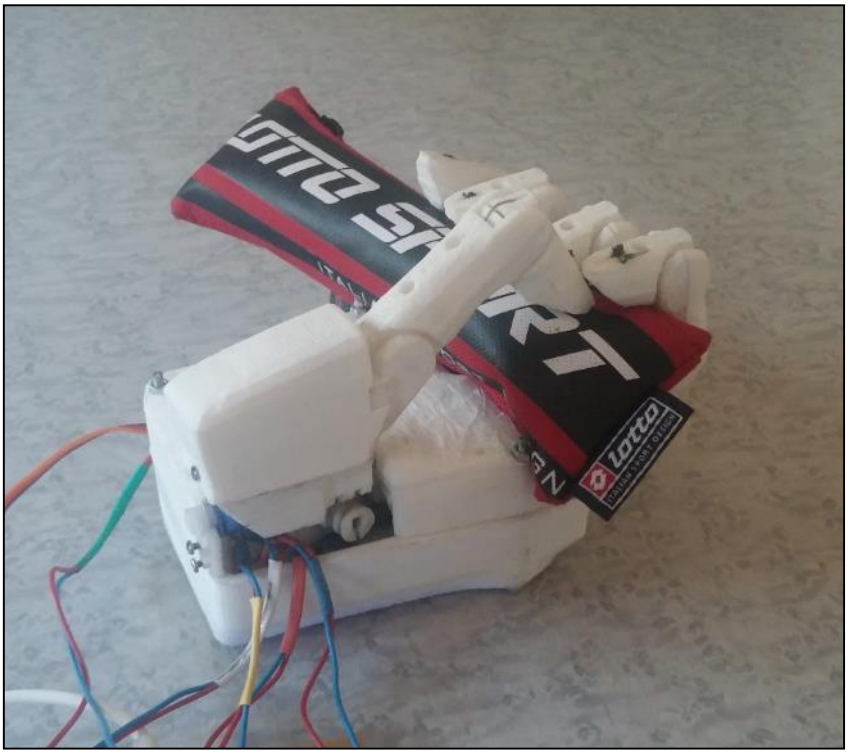


***Imagen 93**. Prótesis ejerciendo el agarre fuerte.*

5. **ESTADO MEDIO-MEDIO**: Este estado consta de realizar un gesto de 'vete a la mierda'. En este estado se cerrarán todos los dedos excepto el corazón, con el pulgar situado enfrente del resto de dedos. En este estado, tras realizar el gesto y esperar unos segundos, la mano se vuelve a abrir.

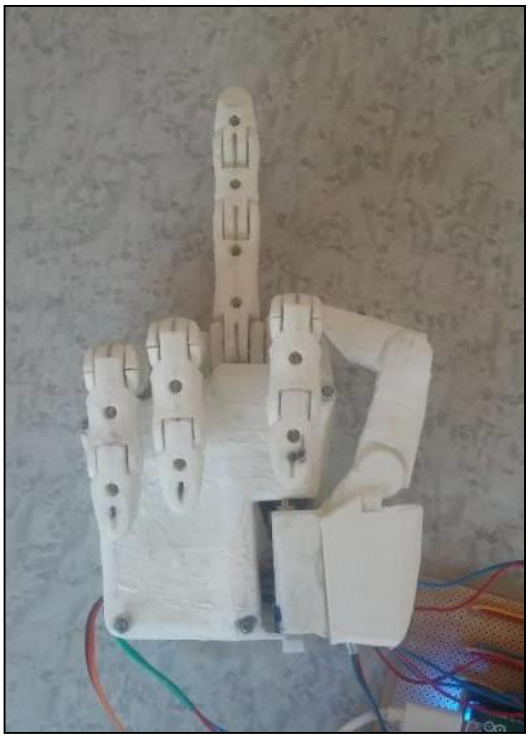


***Imagen 95**. Prótesis realizando el gesto de 'vete a la mierda'.*

6. **ESTADO MEDIO-ALTO**: En este estado se realizarán unos cuernos con la mano. En este estado se cerrarán todos los dedos excepto el meñique y el índice, con el pulgar situado perpendicular al resto de dedos. En este estado, tras realizar el gesto y esperar unos segundos, la mano se vuelve a abrir.

***Imagen 94**.Prótesis realizando unos cuernos.*

7. **ESTADO ALTO-BAJO**: Aquí realizaremos el gesto de ‘Ok’. Para ello cerraremos todos los dedos, a excepción del pulgar, el cual estará situado en perpendicular al resto de dedos. En este estado, tras realizar el gesto y esperar unos segundos, la mano se vuelve a abrir.

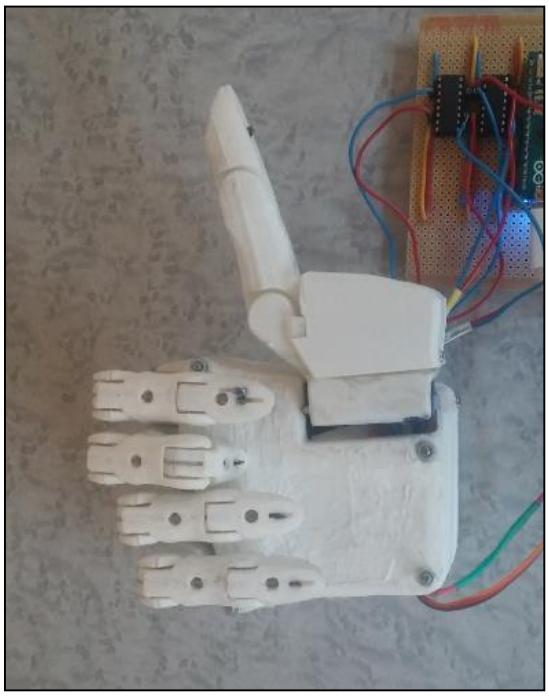


***Imagen 96**. Prótesis realizando el gesto de 'ok'.*

8. **ESTADO ALTO-MEDIO**: Este estado se ha incluido para probar el servomotor individualmente. Simplemente se probará el giro del servomotor haciéndole moverse entre las dos posiciones que se emplearán. Este estado es necesario debido a que como ya se ha comentado el servomotor presenta muchos fallos. Se partirá de la posición con el pulgar enfrentado al resto de dedos, se le hará moverse a la posición en perpendicular a los otros dedos para finalmente volver a la posición inicial.

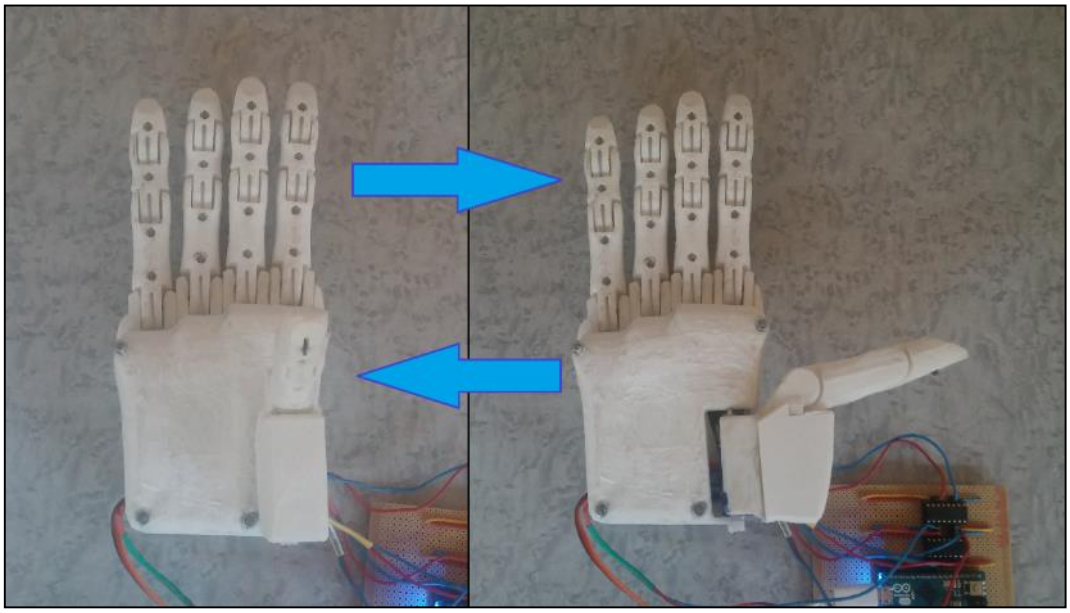


***Imagen 97**. Movimientos que ejecuta la prótesis al realizar la prueba del servomotor.*

9. **ESTADO ALTO-ALTO**: En este estado se abrirá la mano y se situará el pulgar en la posición que lo sitúa enfrente del resto de dedos. Para ello abriremos cada uno de los dedos simultáneamente y giraremos el pulgar en caso de que sea necesario. La utilidad de este estado está en poder abrir la mano y soltar lo que hemos cogido una vez se ha efectuado uno de los dos agarres disponibles (Niveles bajo-alto y medio-bajo).

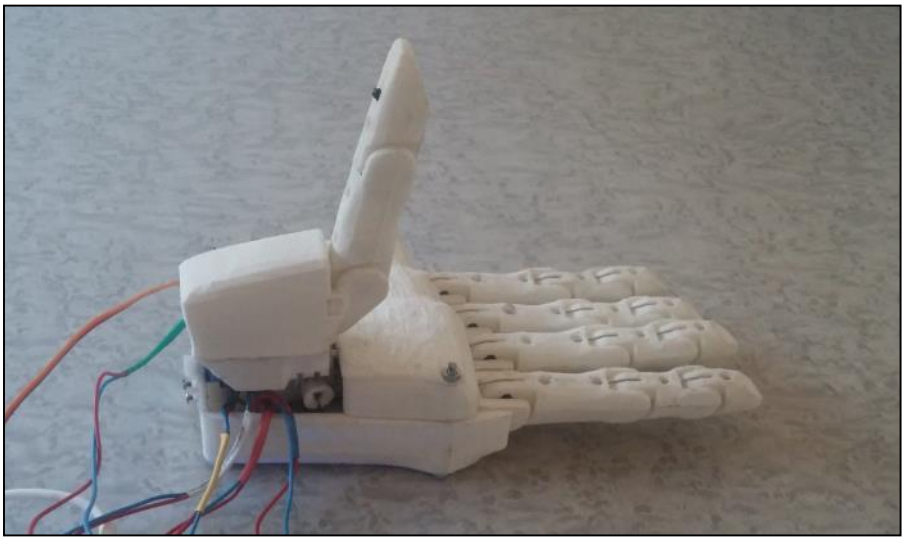


***Imagen 98**. Prótesis situada en la posición de mano abierta.*

3.2.1.1.2. DIAGRAMA DE FLUJO

Tras acceder al nivel inicial, se comprueba el estado en el que estamos. Si se supera el tiempo de permanencia mínimo en dicho estado, se realizará la acción correspondiente a dicho estado. En caso de que no supere el tiempo mínimo establecido, se seguirá comprobando el estado en cada momento.

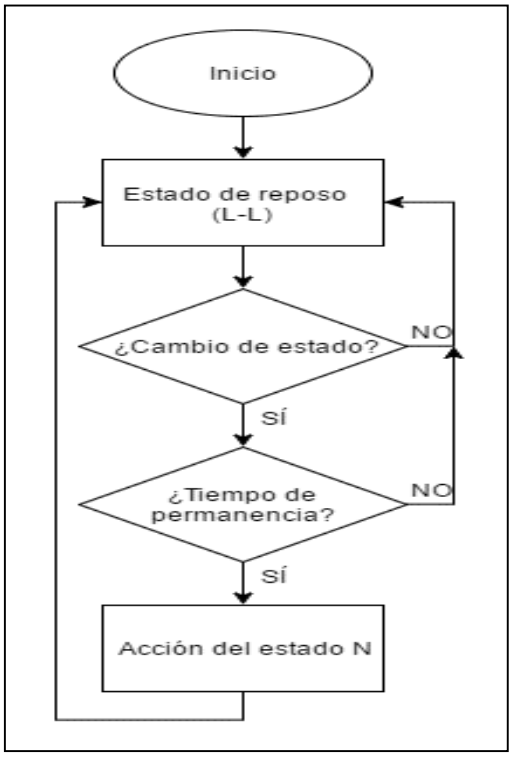


***Imagen 99**. Control de estados: diagrama de flujo.*

3.2.1.1.3. DESARROLLO DEL CÓDIGO

En la sección dedicada a la aplicación existente se explicaron varias cosas del código que también aparecen en el de esta aplicación, como por ejemplo la declaración de las librerías y de las principales variables, por lo que esas cosas no se volverán a explicar, centrándonos en explicar las partes nuevas del código que no hayamos visto hasta ahora.

Respecto a la declaración de librerías, variables e inicialización del sistema está todo explicado anteriormente por lo que no nos detendremos con ello. Las únicas sentencias nuevas que aparecerán son las siguientes:

```
#define E1 13  //Damos nombres a las variables de encendido de cada uno de los motores
#define E2 5
#define E3 9
#define E4 10
#define E5 11

#define I1 2  // Damos nombres a las variables de control de los motores
#define I2 4
```

La función de estas sentencias es muy simple, dar un nombre a una constante para poder referirnos a ella con ese nombre. De este modo podemos acceder a los pines de control de los motores con el nombre que les hemos dado, haciendo, de esta manera, que el código quede mucho más limpio.

Para conseguir implementar los nueve diferentes estados, declararemos un 'switch' idéntico al de la aplicación 'Control de estados' vista antes, cambiando únicamente los umbrales que se establecen para que se acceda a un estado o a otro.

A continuación, intentamos ofrecer una visión general de la aplicación mediante un sencillo pseudo-código:

```
# //Declaración de librerías
# //Asociación de nombres a contantes
//Declaración de variables

void setup() {

    //Conexión con el puerto serie
    //Posición inicial de la mano
    //Asociación de pines

}

void loop() {

    //Lectura de los valores de las variables de control y restricción de los valores de
estas


    //En el siguiente Switch se seleccionarán los distintos movimientos en función de
los distintos niveles de esfuerzo muscular realizado
    //Dentro de cada estado se ha establecido un if para que solo se ejecuten las
acciones asociadas a cada estado si se permanece en él un
    //mínimo de 700ms
    switch (canal1){
      case 0 ... 180:  //Nivel bajo canal1

        switch (canal2){
        case 0 ... 180:  //Nivel bajo-bajo

          delay(700);
          if(canal1>0 && canal1<70 && canal2>0 && canal2<70){
            //Acciones asociadas al estado bajo-bajo
          }

        break;
```

```
  case 181 ... 500:  //Nivel bajo-medio

    delay(700);
    if(canal1>0 && canal1<70 && canal2>71 && canal2<170){
      //Acciones asociadas al estado bajo-medio
    }

  break;

  case 501 ... 820:  //Nivel bajo-alto

    delay(700);
    if(canal1>0 && canal1<70 && canal2>171 && canal2<250){
      //Acciones asociadas al estado bajo-alto
    }

  break;
  }
break;


case 181 ... 500:  //Nivel medio canal1
  switch (canal2){
  case 0 ... 180:  //Nivel medio-bajo

    delay(700);
    if(canal1>71 && canal1<170 && canal2>0 && canal2<70){
      //Acciones asociadas al estado medio-bajo
    }

  break;

  case 181 ... 500:  //Nivel medio-medio

    delay(700);
    if(canal1>71 && canal1<170 && canal2>71 && canal2<170){
      //Acciones asociadas al estado medio-medio
    }

  break;

  case 501 ... 820:  //Nivel medio-alto

    delay(700);
    if(canal1>71 && canal1<170 && canal2>171 && canal2<250){
      //Acciones asociadas al estado medio-alto
      }

  break;
  }
break;


case 501 ... 820:  //Nivel alto canal1
  switch (canal2){
  case 0 ... 180:  //Nivel alto-bajo

  delay(700);
  if(canal1>171 && canal1<250 && canal2>0 && canal2<70){
    //Acciones asociadas al estado alto-bajo
  }

  break;

  case 181 ... 500:  //Nivel alto-medio

    delay(700);
    if(canal1>171 && canal1<250 && canal2>71 && canal2<170){
      //Acciones asociadas al estado alto-medio
    }

  break;

  case 501 ... 820:  //Nivel alto-alto
```

```
          delay(700);
          if(canal1>171 && canal1<250 && canal2>171 && canal2<250){
            //Acciones asociadas al estado alto-alto
          }

        break;
        }
      break;

      default: break;
      }
}
```

Dentro de cada uno de los 'case' del switch, se ha incluido un if para que no se realicen las acciones asociadas a cada estado, a no ser que se haya permanecido un mínimo de 700ms en él.

Al final del código, para evitar cualquier tipo de error en las combinaciones, incluimos la sentencia "`default: break`"; que para cualquier combinación no descrita entre las anteriores, sale del 'switch' y seguimos comprobando valores de entrada. Dicha sentencia no es necesaria, pues hemos restringido los valores de las entradas, pero se ha incluido para evitar posibles complicaciones.

Una vez se ha dado una visión global de la implementación del código, se procede a describir el código correspondiente a uno de los estados. Se describirá el estado alto-bajo (Signo de 'Ok'), pues es uno de los más completos, y una vez entendido este, comprender el resto resulta trivial.

```
        Serial.println("Okay seleccionado");

          //En este estado dejaremos sin cerrar solo el pulgar, el cual giraremos
          //para ponerlo en perpendicular al resto de dedos
          Serial.println("Cerrando");

          digitalWrite(I1, HIGH);
          digitalWrite(I2, LOW);
          analogWrite(E1,255);
          analogWrite(E2,255);
          analogWrite(E3,255);
          analogWrite(E4,255);
          srv1.write(110);

        delay(5000);

        Serial.println("Abriendo");

          digitalWrite(I1, LOW);
          digitalWrite(I2, HIGH);

        delay(2000);

          digitalWrite(I1, LOW);
          digitalWrite(I2, LOW);

          analogWrite(E1,0);
          analogWrite(E2,0);
          analogWrite(E3,0);
          analogWrite(E4,0);
          srv1.write(25);

      delay(2000);
```

- En primer lugar, se envía por el puerto serie una frase indicando a qué estado se ha accedido.
- Después se indica que se van a cerrar los dedos y se activarán los dedos E1, E2, E3 y E4, que son los que se desean cerrar, además se moverá el servo a la posición de 110º que es la que sitúa el pulgar en perpendicular al resto de dedos. Por otro lado, pondremos

en alta el pin I1 y en baja I2, de modo que al activarse los motores giren en la dirección que hará que se cierren los dedos.

- Después se esperarán 5 segundos y se indicará que se van a abrir los dedos, para ello simplemente se pondrá en pin I1 en baja y el I2 en alta, cambiando así la dirección de giro de los motores, ocasionando que los dedos se abran.
- Tras esto, se esperarán 2 segundos para que los dedos tengan tiempo de sobra para abrirse se pondrán todos los pines (I1, I2, E1, E2, E3 y E4) en baja y se moverá el servo a la posición inicial, que es la posición de 25º, la cual sitúa el pulgar enfrente del resto de dedos.
- Finalmente, el programa esperará 2 segundos antes de volver a buscar posibles cambios de estado, para que el servomotor tenga tiempo de sobra para volver a la posición de 25º.

El código completo de esta aplicación está disponible en el anexo 10.

PRUEBAS DE FUNCIONAMIENTO

4.1. PRUEBAS CON EL SISTEMA ARDUINO

A lo largo de todo el proceso de montaje de la mano, ha habido varios momentos en los que se ha hecho necesario probar el funcionamiento de algo antes de continuar con el proceso. Las pruebas son necesarias pues de no solucionar posibles errores en su momento, es muy posible que el remedio de estos fallos se complique conforme avanza el proceso de montaje.

Para simular los montajes electrónicos antes de llevarlos a cabo, y disponer así de un esquema con el que guiarnos, hemos empleado un programa llamado Fritzing. Este programa nos simplificará la tarea de crear los esquemas de nuestros montajes, tarea necesaria para posteriormente realizar estos sobre una placa real de una manera mucho más rápida y fiable que si lo hiciéramos sin guión alguno. Además, dicho programa cuenta con numerosas ventajas respecto de dibujar los esquemas sobre un folio: Se ahorra tiempo, los esquemas quedan más limpios, el programa tiene una gran facilidad de uso y posee una gran base de datos con muchísimos componentes electrónicos. Por mencionar alguna desventaja, decir que, si se tienen que colocar cables con muchos giros, el proceso se puede hacer un poco tedioso, ya que la manera de introducir un giro en un cable no está del todo optimizada en cuanto al tiempo requerido para ello. En el anexo 5 podremos encontrar más información acerca de este programa.

4.1.1. PRUEBA DE LOS SENSORES BIMORFOS

Se hicieron pruebas con sensores bimorfos, para comprobar la viabilidad de estos para ser implementados en nuestro sistema.

El programa creado era muy sencillo. Básicamente este leía constantemente el valor de una variable, apagando el motor cuando esta presentaba un voltaje de entrada por encima de un umbral. El umbral fue seleccionado en base a varios ensayos, probando diferentes valores y observando que el mejor comportamiento se daba para un umbral de unos 2,4V.

El código de Arduino de este programa está presente en el anexo 6.

Se vio que los sensores no poseían una eficacia del 100%, por lo que podría haber contactos que no fuesen detectados. Sin embargo, pese a no tener una eficacia del 100%, poseen una eficacia bastante buena como para ser empleados como sensores de contacto.

4.1.2. PRUEBA DE LOS DRIVER DE LOS MOTORES

Para realizar el control de los motores era necesario el empleo de drivers, para poder variar la dirección de giro de estos (Apertura y cierre de la mano), sin la necesidad de cambiar físicamente la posición de los pines. La prueba consistirá en seleccionar uno de los dos agarres desde el PC empleando el puerto serie del Arduino. De ahí Arduino activará unos motores u otros en función del agarre seleccionado.

Se realizó una prueba de funcionamiento empleando integrados LD293D, pues nos simplifican mucho el montaje del circuito requerido para este ensayo. Para realizar esta prueba sólo se contaba con un integrado, por lo que, para comprobar el funcionamiento de todos los dedos, y a la vez probar la configuración del integrado, se decidió unir dedos. Las hojas de especificaciones de este integrado están disponibles en el anexo 17.

Cada integrado puede controlar dos motores (O dos grupos de ellos), por lo que se decidió agrupar, por un lado, los dedos índice y pulgar, y por otro los dedos corazón, anular y meñique.

***Imagen 100**. Agarre fino.*

De esta manera se podría configurar el programa para escoger entre dos agarres: Agarre fuerte y agarre fino.

***Imagen 101**. Agarre fuerte.*

El esquema de conexión empleado es el de la Imagen 102, donde cada motor representa un grupo de motores.

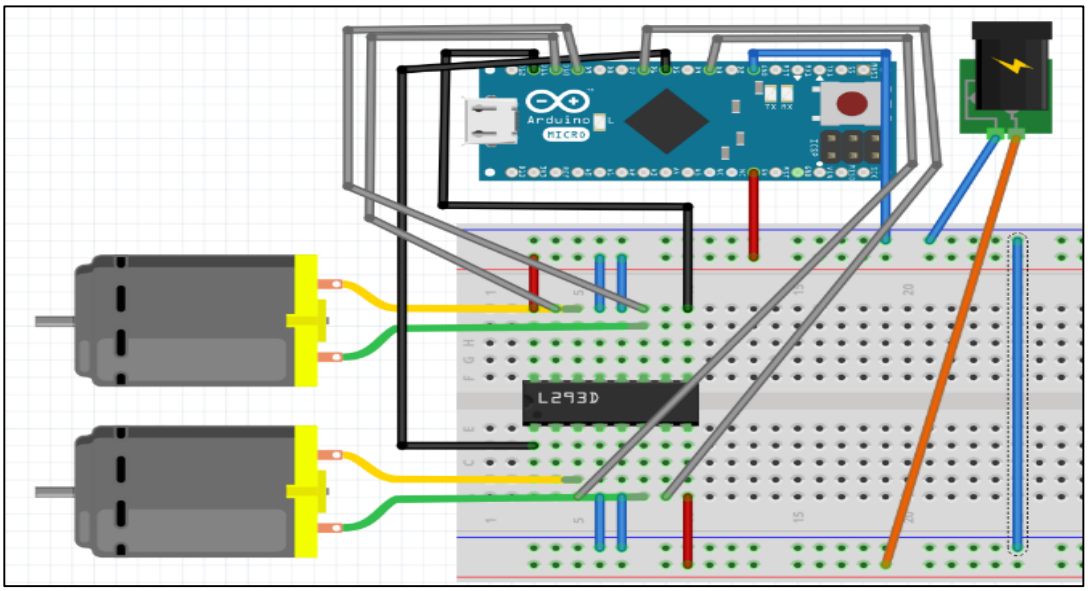


***Imagen 102**. Esquema de conexiones empleado en la prueba de los drivers de los motores.*

El código de Arduino de esta prueba está presente en el anexo 7.

Tras la prueba, comprobamos que el montaje realizado es correcto y que el integrado L293D puede controlar perfectamente nuestros motores, por lo que podremos avanzar un paso más y realizar el montaje con los 3 driver, controlando así cada motor individualmente.

4.1.3. PRUEBA INDIVIDUAL DE LOS MOTORES

Tras comprobar que el integrado L293D es adecuado para el control de nuestros motores se realizó el montaje con los 3 driver, empleando un esquema de conexiones similar al usado en el apartado anterior, pero extendido para poder controlar los 5 motores en vez de 2 (O dos grupos de ellos, que es lo que hacíamos). En la Imagen 103 se puede ver el esquema de montaje empleado.

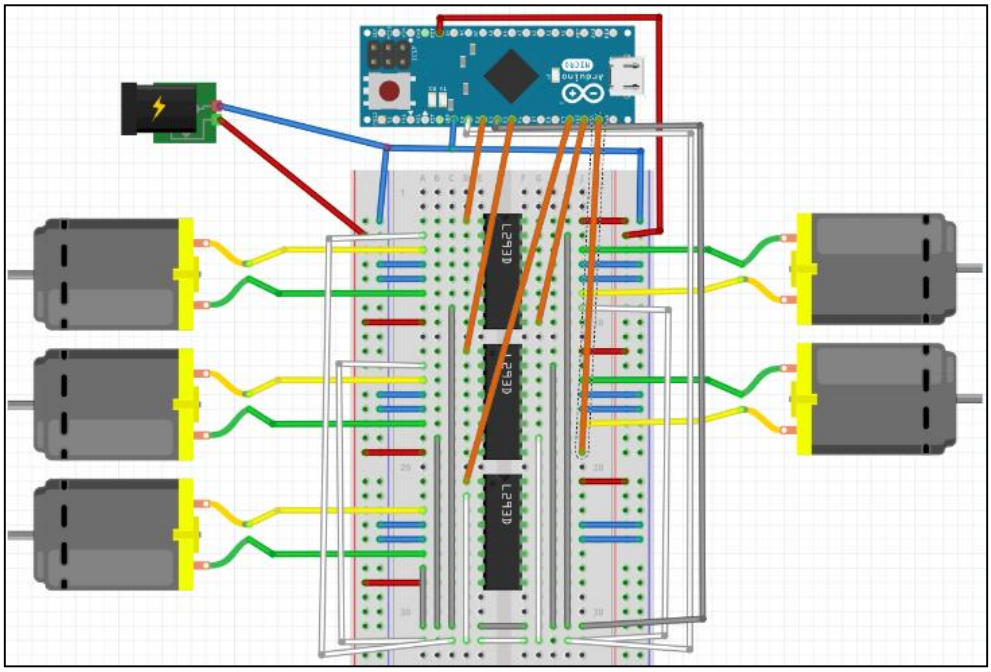


***Imagen 103**. Esquema de conexiones empleado en la prueba individual de los motores.*

Esta prueba consiste en ir abriendo y cerrando cada uno de los dedos de la mano individualmente. Se esperará unos segundos entre uno y otro, para poder observar donde se produce el error en caso de que se produjese alguno.

El código de Arduino de esta prueba está presente en el anexo 8.

Es importante remarcar el hecho de que, cómo en ningún momento se va a estar abriendo un dedo y cerrando otro al mismo tiempo, se han agrupado los pines de control de los motores para poder controlar todos los motores con tan solo 2 salidas del Arduino.

Esta prueba nos ha permitido comprobar que nuestro montaje es correcto, y que todos los dedos tienen un correcto funcionamiento

4.1.4. PRUEBA DE CONTROL PRECISO DE LOS MOTORES

Una vez comprobado que nuestro montaje es correcto, pasaremos a complicar un poco más los movimientos que nuestra mano realiza. Ahora seleccionaremos entre un movimiento u otro usando el puerto serie de Arduino para introducir datos desde el PC.

En esta prueba podremos seleccionar entre una gran variedad de movimientos (Agarre fuerte, agarre suave, signo de Okay…) mediante el puerto serie de Arduino. El montaje es prácticamente el mismo que en el apartado anterior, pero añadiendo las conexiones necesarias para el servomotor. Lo que sí que cambiará de forma importante es el código del programa, pues además de tener que incluir todas las órdenes para los distintos movimientos de la mano que queremos implementar, se empezará a trabajar con el servo, para así poder modificar la posición del pulgar. Trabajar con el servo es muy sencillo gracias a las sentencias que nos ofrece la librería '`Servo.h`', la cual, por ejemplo, nos permite mover el servo a una determinada posición sin más que emplear la función '`variableServo.write(argumento)`', introduciendo en el argumento la posición en grados en la que queremos que se sitúe el servo.

El código de Arduino para esta prueba está disponible en el anexo 9.

Tras realizar esta prueba hemos podido comprobar que podemos seleccionar los distintos movimientos que queremos que realice nuestra mano desde el PC sin ningún tipo de problema. Esto nos facilitará la tarea de incorporar un control mioeléctrico a la mano, pues sabremos que en esta parte no se produce ningún error. Por otro lado, el servo presenta ciertas complicaciones a la hora de girar, pues en ciertos momentos se atasca y tiene problemas para llegar a la posición que se le indica. Tras tratar de regularlo para mejorar su funcionamiento sin éxito, se vio que si se establecían los giros entre 25º y 110º y se forzaba un poco su posición tras cada giro, llegaba un punto en el que el servo alcanzaba una posición adecuada y podía realizar el giro correctamente.

4.1.4.1. PRUEBA DEL MONTAJE SOBRE LA STRIPBOARD.

Una vez comprobado el correcto funcionamiento del montaje anterior, y visto que el código creado nos permite realizar adecuadamente todos los movimientos que deseamos, pasamos a mover el montaje de la placa de pruebas a una placa de pistas en la cual el montaje quedará fijado definitivamente.

El gran problema de este montaje es que intenta reducir en gran medida el tamaño del circuito, por lo que se podrían presentar malas conexiones debido a la proximidad entre estas. Para evitar un mal funcionamiento del circuito debido a dicho motivo, se comprobó minuciosamente cada una de las conexiones con la ayuda de un multímetro.

En la Imagen 104 podemos ver el montaje que se ha realizado sobre la placa de pistas. Por simplicidad, y para poder visualizar mejor el montaje solo se ha incluido un motor DC, dado que el conexionado para los otros 4 motores se realizaría de forma análoga.

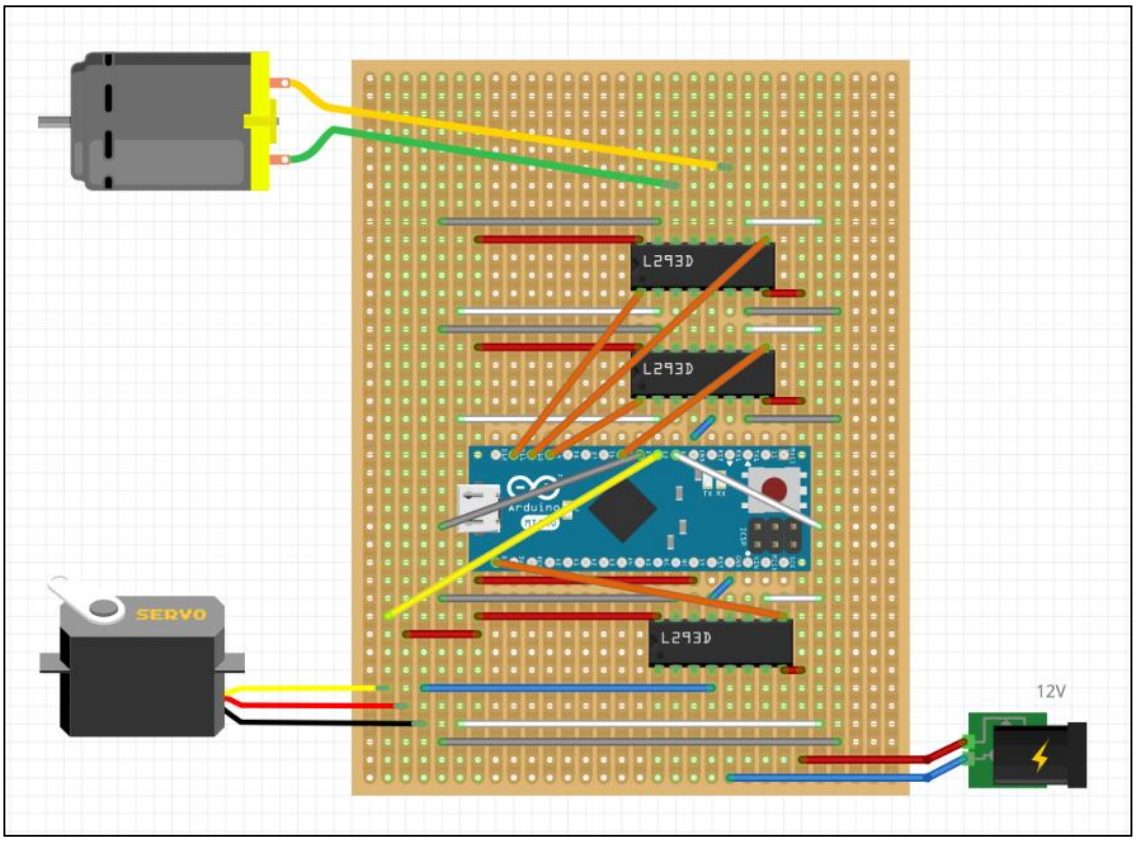


***Imagen 104**. Esquema de conexiones realizado para la prueba de control preciso de los motores.*

Tras colocar los integrados y el Arduino sobre sus zócalos y conectar los motores adecuadamente se probó el funcionamiento del montaje de la placa, viendo que no había ningún problema y que el funcionamiento era correcto.

4.1.5. PRUEBA DEL SISTEMA MIOELÉCTRICO DE CONTROL

Cómo ya se ha dicho, la mejor forma de construir el mando mioeléctrico es paso a paso sobre una placa de pruebas. En primer lugar, durante el montaje del primer canal, se iba montando esta etapa a etapa y probando cada una de ellas individualmente. Con esto, llegábamos al montaje final sabiendo que cada una de las etapas individualmente funcionaba y solo teníamos que comprobar el funcionamiento global del primer canal.

Para comprobarlo se empleó un programa muy simple, el cual, simplemente se mostrará por pantalla el nivel de fuerza que el Arduino considera que estamos desempeñando (Bajo, medio o fuerte) y veremos si se corresponde con el que realmente estamos haciendo. Esta prueba, además de ayudarnos a comprobar el correcto funcionamiento de nuestro canal de control, nos ayudará a fijar de manera óptima los umbrales que separan los distintos niveles de esfuerzo.

Una vez visto que nuestro canal funciona, pasaremos a replicarlo dado que para el control mioeléctrico necesitamos dos canales. El segundo canal también se deberá montar comprobado cada etapa individualmente, pues de esta manera es mucho más sencillo evitar malos contactos y demás fallos.

Tras comprobar el segundo canal, pasaremos a hacer pruebas con los dos canales simultáneamente para comprobar que somos capaces de acceder a los 9 estados de los que disponemos. Para realizar estas pruebas se empleó la aplicación de Arduino disponible en el anexo 11, el cual es similar al programa empleado para comprobar un solo canal, pero con pequeñas modificaciones para poder probar dos canales simultáneamente.

4.1.5.1. PRUEBA DEL MONTAJE SOBRE LA STRIPBOARD.

Una vez comprobado el correcto funcionamiento del montaje anterior, y visto que nos permite acceder adecuadamente a todos y cada uno de los estados de los que vamos a disponer, pasamos a mover el montaje de la placa de pruebas a una placa de pistas en la cual el montaje quedará fijado definitivamente.

El gran problema de este montaje es que intenta reducir en gran medida el tamaño del circuito, por lo que se podrían presentar malas conexiones debido a la proximidad entre estas. Para evitar un mal funcionamiento del circuito debido a dicho motivo, se comprobó minuciosamente cada una de las conexiones con la ayuda de un multímetro.

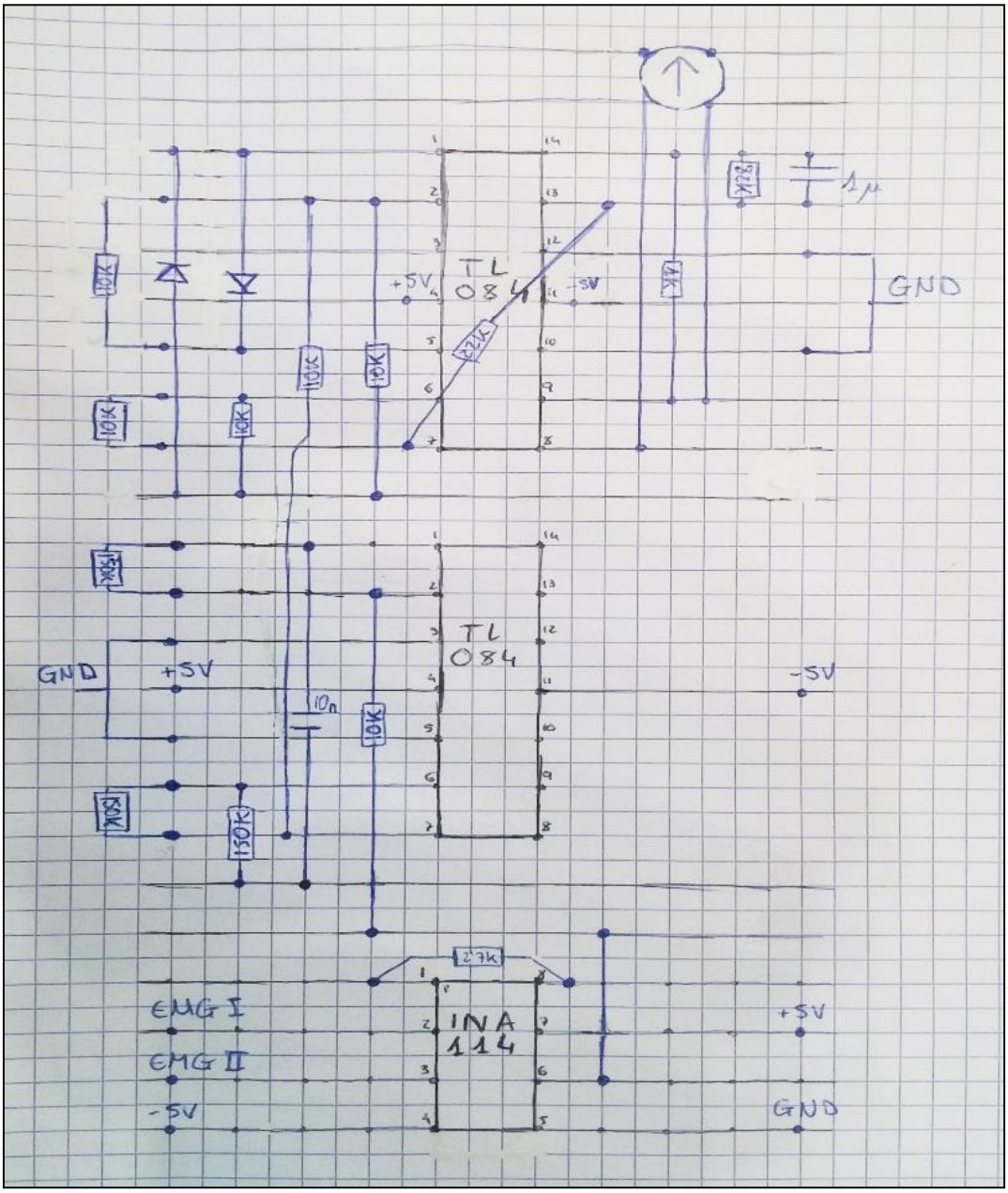


***Imagen 105**. Esquema de conexiones realizado para un canal del mando mioeléctrico.*

En la Imagen 105 podemos ver el montaje que se ha realizado sobre la placa de pistas. Una vez realizado este montaje, deberemos comprobar de nuevo los dos canales simultáneamente, empleando de nuevo la aplicación del anexo 11, para cerciorarnos de que el montaje es correcto y de que no hay ninguna mala conexión. El esquema para este montaje es demasiado complejo, por lo que no se ha podido utilizar 'Fritzing', y se ha realizado en una hoja de papel. Se ha realizado solo el esquema de un canal dado que el esquema para el segundo es idéntico. Aunque no se indique en la imagen, el potenciómetro empleado es de 100KΩ.

Después de esto, tenemos que conectar esta placa con la anteriormente creada para el control de los motores, llevando las dos salidas de los canales a dos entradas analógicas del Arduino y conectando la alimentación (sólo los +5V en este caso) y las tierras de ambas. Tras asegurarnos de que la conexión de las dos placas de pistas es correcta, podemos pasar a probar el sistema completo, empleando el software definitivo, disponible en el anexo 10.

En dicho software haremos uso de los umbrales fijados para que cada uno de los canales sea capaz de distinguir entre tres niveles de esfuerzo, de esta manera, podremos acceder a 9 movimientos distintos. Estos movimientos serán tomados del software en el que seleccionábamos distintos movimientos de la mano mediante el PC.

CONCLUSIONES Y LÍNEAS DE MEJORA

5.1. LÍNEAS DE MEJORA

5.1.1. IMPLANTACIÓN DE UN SISTEMA DE ACOPLAMIENTO PARA LA PRÓTESIS [24]

Los sistemas acopladores de próstesis existentes hasta hace unos pocos años, poseían numerosas funcionalidades, pero aun así contaban con algunas limitaciones importantes, como por ejemplo el uso de rotadores de muñeca potentes.

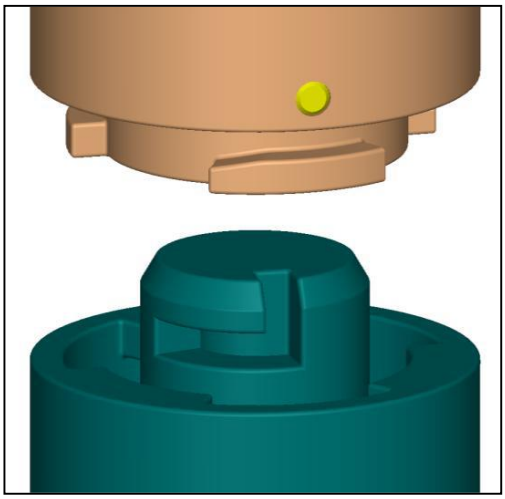


***Imagen 106**. Prototipo de acoplador de prótesis universal.*

En los últimos años ha estado trabajando en el desarrollo de sistemas de acoplamiento de muñeca universales, que cumplan todos los requerimientos que un amputado pueda tener. A continuación, se detallarán dichos requerimientos:

1. Restricciones de tamaño: Este requerimiento es importante, tanto por funcionalidad, como por apariencia. El acoplador no debería tener una largura mayor de 12mm para que los pacientes amputados a la altura de la muñeca puedan emplearlo cómodamente. Por otro lado, su diámetro no debería superar el de una muñeca pequeña (25mm) para poder ponerle un recubrimiento cosmético que simule una piel real.
2. Facilidad de conexión y desconexión: La conexión y desconexión debería constar de pocos movimientos, secuenciales en caso de ser más de uno. Estas acciones también deberían poder ser realizadas sin la necesidad de contacto visual directo con el sistema de acoplamiento, y sin la necesidad de quitar el recubrimiento cosmético, en caso de que lo hubiera.
3. Restricción de la rotación: El acoplador debe permitir bloquear su posición para que no rote indeseadamente. A la hora de bloquear la rotación hay que tener en cuenta que no se deben permitir contactos eléctricos incorrectos
4. Restricciones de fuerza: El acoplador debe resistir como mínimo 45 Nm de flexión y 15 Nm de torsión axial, aunque sería deseable algo más.
5. Interfaz con el segmento adyacente: El diámetro del acoplador debe permitir ser adaptado al tamaño requerido por cada paciente en particular.
6. Restricciones de fabricación: El acoplador debe estar construido de una manera que permita ser acoplado de alguna manera al muñón.

7. Localización: El acoplador debería poder ser situado en cualquier parte a lo largo del antebrazo.
8. Restricciones electrónicas: El dispositivo debe permitir pasar la corriente, y resistir a numerosas conexiones y desconexiones tanto mecánicas cómo eléctricas. El número mínimo de conexiones que debe permitir el acoplador es 6. El acoplador también debe evitar conexiones eléctricas erróneas y la penetración de la humedad dentro de ella.

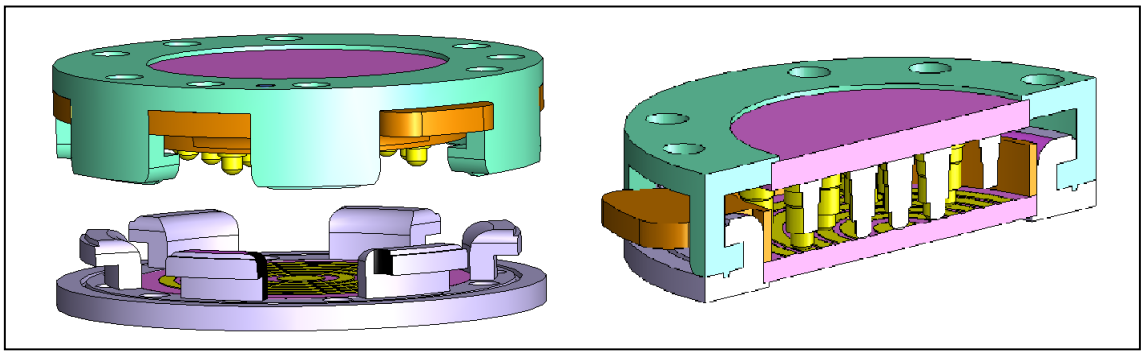


***Imagen 107**. Otro prototipo de acoplador de prótesis universal*

5.1.2. INCORPORACIÓN DE UN SENSOR DE TEMPERATURA

La incorporación de un sensor de temperatura en nuestra mano resultaría de gran utilidad, dado que es un elemento de seguridad muy importante. Por un lado, protegería al paciente de agarrar objetos demasiado calientes con los que se podría dañar, por ejemplo, podría avisarle de que está cogiendo una bebida demasiado caliente, para que no se la llevase a la boca y se quemase. Por otro lado, no debemos olvidar que nuestra mano está construida con plástico PLA, el cual tiene una temperatura de fusión muy baja, y puede empezar a sufrir daños a partir de 60º-70º, por lo que coger objetos demasiado calientes podrían provocar graves daños en nuestra prótesis.

Cómo sensor de temperatura se podría emplear un termistor. Los termistores están formados por un material semiconductor que varía su resistencia eléctrica con las variaciones de temperatura. Su fabricación se hace a base de óxidos semiconductores de los metales de transición del grupo del hierro como Cr, Mn, Fe Y Co. Inicialmente la resistividad de estos óxidos es muy elevada, pero al agregar ciertas impurezas (pequeñas cantidades de otros iones de distinta valencia) su respuesta eléctrica cambia de tal manera que son catalogados como semiconductores. [25] [26]

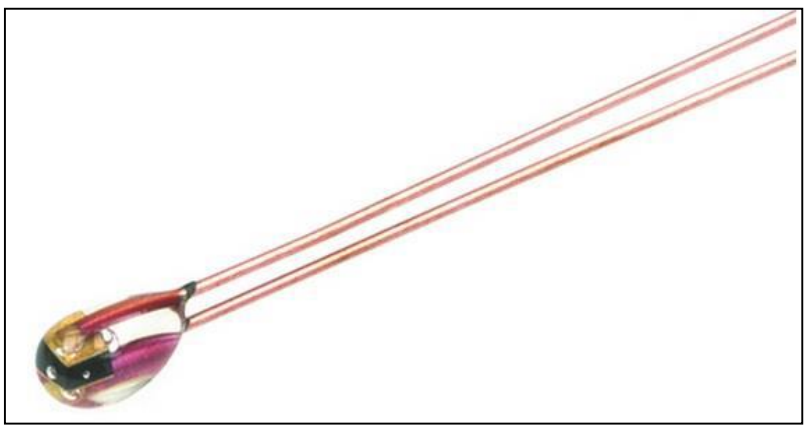


***Imagen 108**. Termistor tipo NTC.*

Los termistores tienen el inconveniente de ser poco lineales, la relación entre la resistencia y la temperatura de un termistor es exponencial. Sin embargo, al igual que con la mano natural no es necesario saber la temperatura exacta a la que se encuentra un objeto, sino que basta con saber si un objeto está caliente, frío, si quema… Por este motivo la falta de linealidad no supone un problema, dado que bastaría con establecer una cantidad discreta de sensaciones térmicas comprendiendo cada una de las sensaciones un rango más o menos amplio de temperaturas. Una posible implementación sería establecer 5 sensaciones térmicas (5 rangos de temperaturas): Congelado, frío, normal, caliente y quemando.

Como actuadores térmicos, podríamos emplear dos dispositivos: Una célula Peltier y una termorresistencia. Estos dispositivos serán los encargados de proporcionarle información al paciente acerca de la temperatura que se está detectando con el termistor.

La termorresistencia nos proporcionará la sensación de calor, dado que se calentará por el efecto Joule, al hacer circular una cierta por ella. Podremos regular la cantidad de calor que se genera alimentándola con más o menos voltaje, pudiendo así crear diferentes sensaciones térmicas con ella. Para nuestro ejemplo, con la resistencia podríamos crear las sensaciones térmicas de 'normal', 'caliente' y 'quemando'.

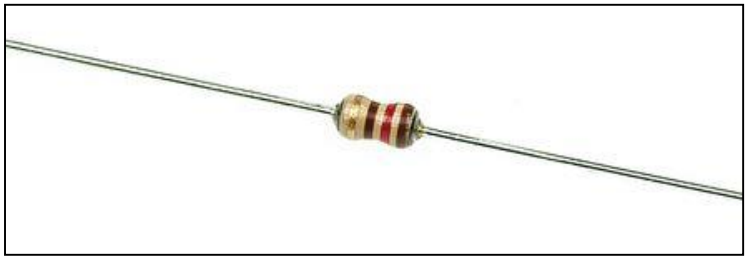


***Imagen 109**. Termorresistencia.*

La célula Peltier sería la encargada de proporcionarnos la sensación de frío. Una célula Peltier es un dispositivo electrónico basado en un principio físico-electrónico que descubrió Peltier. Está constituida por un conjunto de termopares, situados eléctricamente en serie y térmicamente en paralelo, de tal forma que todos los termopares que absorban calor están a un lado de la placa y los que desprenden calor están situados al otro lado. El resultado es una especie de chapita que al aplicarle una diferencia de potencial una de sus caras se enfría tanto como se calienta la otra.

El material semiconductor más utilizado en las Peltier es el Telurio de Bismuto, dopado por exceso (tipo N).

Este tipo de dispositivo puede bombear una potencia de calor comprendida entre mili vatios y los centenares de vatios.

La célula Peltier está compuesta por dos o varias placas de cerámica separadas por cubos de material semiconductor (Imagen 110). Estas placas de cerámica sirven como soporte mecánico de la estructura del dispositivo y como aislante eléctrico entre los elementos de la célula y la superficie de montaje externa. Al aplicar una señal de tensión continua a un módulo termoeléctrico, el calor se transmite a través del dispositivo desde una cara hasta la otra. El calor existente en una de las caras (cara fría) se extrae y se bombea a la otra cara (cara caliente). Por tanto, la temperatura de una de las caras disminuye mientras que la temperatura de la otra cara aumenta. [26]

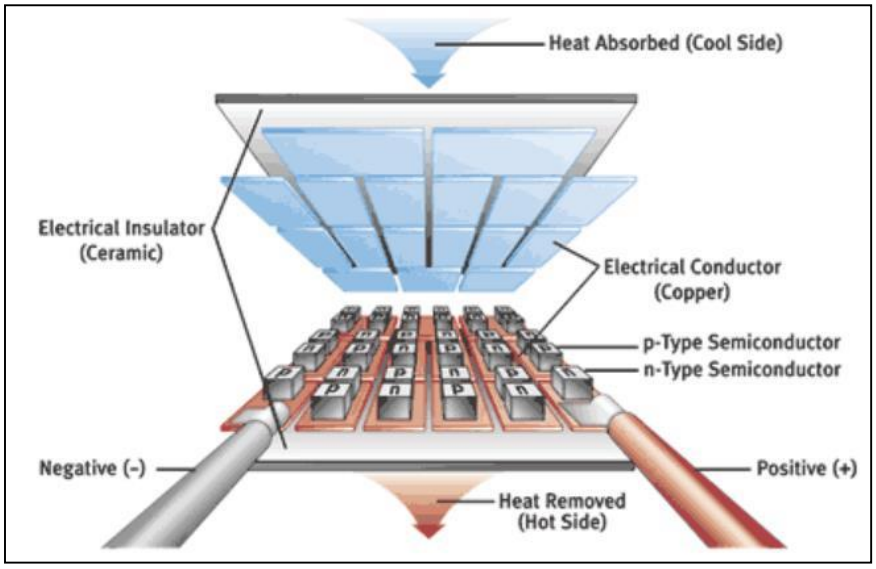


***Imagen 110**. Célula Peltier desglosada parte a parte. [27]*

Para poder alimentar la célula Peltier desde Arduino, se podría emplear un transistor que se encargase de amplificar la corriente que proporciona el Arduino, ya que esta no resulta suficientemente grande para emplearla en la célula Peltier. Cuanto mayor sea el voltaje con el que alimentamos la célula, mayor será la sensación de frío que nos genere, por lo que podremos crear diferentes sensaciones térmicas con ella. Para nuestro ejemplo, con la célula Peltier podríamos crear las sensaciones térmicas de 'frío' y 'congelado'.

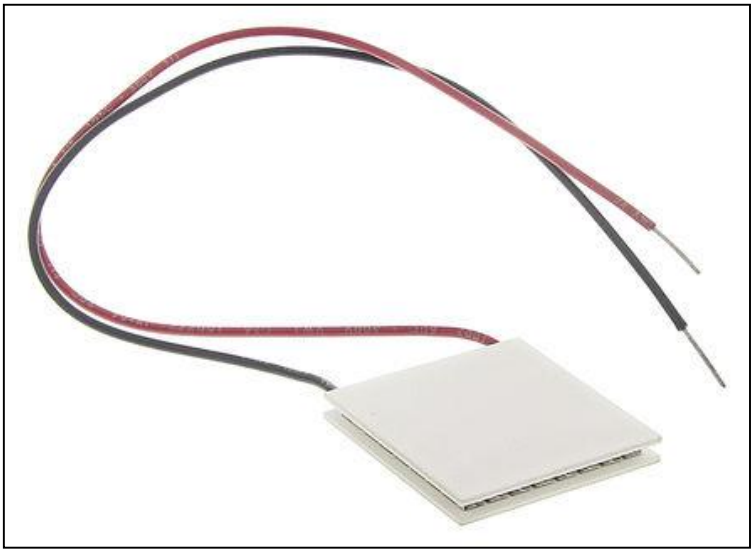


***Imagen 111**. Célula Peltier.*

5.1.3. EMPLEO DE OPERACIONALES 'RAIL-TO-RAIL'

Hemos visto que los amplificadores operacionales que hemos empleado en nuestra placa de adquisición y acondicionamiento, no nos ofrecen a su salida más de 4,2V, al ser alimentados con ±5V. Este hecho se debe a que cuándo un operacional se acerca a la tensión de alimentación, satura antes de llegar a ella. Esta saturación se produce en torno al 80% de la tensión de alimentación, aunque depende del integrado que se esté empleando.

El hecho de que nuestra salida no llegue a 5V hace que no se aproveche todo el rango dinámico del que disponemos, empeorando así la capacidad de nuestro sistema para detectar los 3 niveles (Bajo, medio y fuerte) de esfuerzo muscular que hemos establecido.

Una posible solución sería alimentar los integrados TL084 con ±6V. Sin embargo, esto no es una buena opción, dado que el resto del circuito se alimenta con 5V, y tener que usar dos alimentaciones distintas sólo por este hecho no es práctico.

La solución óptima para este problema son los amplificadores operacionales 'Rail-to-rail'. Estos amplificadores se desarrollaron para vencer el problema de la saturación de los operacionales, pudiendo llegar a alcanzar voltajes de salida pocos milivoltios menores que los voltajes de alimentación. Empleando este tipo de amplificadores conseguimos que nuestra señal se adapte al margen dinámico del que disponemos, optimizando así el funcionamiento de nuestro sistema.

La mejor forma para poder emplear estos operacionales en nuestro sistema sería buscar integrados de 14 pines que fueran 'Rail-to-rail' y que pudiesen ser alimentados con ±5V, para no tener que modificar nada más en nuestro montaje. Un ejemplo de integrados que podríamos utilizar serían los 'LMC660CN' o los 'TLV2374IN'. Por ello, para implementar esta mejora bastaría con sustituir los nuevos integrados por los TL084 que se estaban usando.

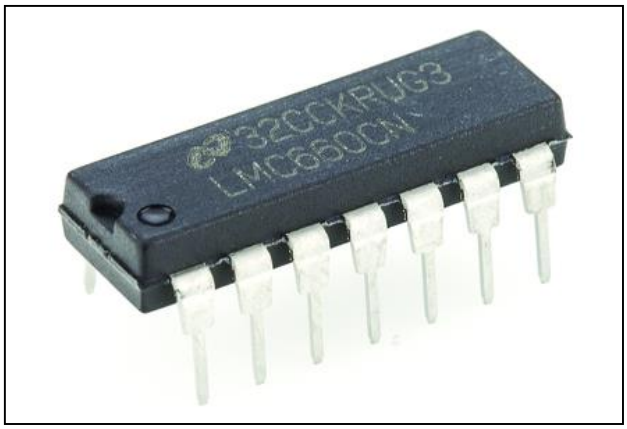


***Imagen 112**. Integrado LMC660CN.*

Como vemos, esta solución es muy simple, pero no se ha podido realizar por falta de tiempo, sin embargo, sería interesante que fuera implementada en un futuro, motivo por el cual se ha detallado en este apartado.

5.1.4. MEJORAS EN EL FILTRADO DEL SISTEMA DE CONTROL MIOELÉCTRICO

El sistema de control presenta un buen funcionamiento, sin embargo, tal vez unos cambios sencillos sobre él podrían mejorar su funcionamiento.

Se podría intentar reducir algo la frecuencia de corte del filtro paso alto, para no perder las componentes del electromiograma que se encuentran a baja frecuencia. Es necesario prestar atención al bajar la frecuencia de corte pues llegará un punto en el que si la seguimos bajando se nos empezará a colar por el filtro el ruido de red y ese es un hecho que debemos evitar a toda costa.

5.1.5. OTRAS POSIBLES MEJORAS

PIEL ARTIFICIAL: El aspecto rudo y poco humano de nuestra prótesis podría hacer que algunos usuarios no se terminaran de adaptar a ella, por lo que podría ser interesante recubrirla de algún tipo de material y pintarla, para que adquiriera un aspecto más humano. Sin embargo, esto no es algo sencillo de hacer por lo que habría que estudiar en profundidad cómo hacerlo sin perder de vista el hecho de que buscamos un presupuesto lo más bajo posible. Una posible manera de realizar una piel artificial sería recubriendo la mano con un guante que se adaptase a ella, adquiriendo así un aspecto mucho más uniforme. Si sobre este guante se pudiera aplicar algún tipo de pintura, al gusto del usuario, se conseguiría un importante avance en lo que al aspecto de la mano se refiere.

MEJORA DEL SERVOMOTOR: Cómo ya hemos dicho anteriormente, el servomotor que maneja el giro del pulgar tiene una importante carencia de fuerza que hace que no pueda funcionar correctamente. Sería interesante en un futuro conseguir un servo del mismo estándar (9g), pero con más de fuerza, y sustituirle por el que tenemos. Tal vez al aumentar la fuerza del servo perdamos algo de velocidad, sin embargo, este hecho no tiene demasiada relevancia dado que el servo actual tiene mucho margen de pérdida de velocidad hasta que esto se convierta en un problema. La sustitución del servo no debería ser demasiado complicada, requiere desmontar alguna pieza de la mano, pero remitiéndose al proceso de montaje no debería haber ningún problema para realizar la sustitución. Es importante que, a la hora de adquirir el servo, recordemos una vez más que queremos construir una prótesis de bajo coste, por lo que no deberemos escoger un servo demasiado caro.

SENSOR DE GRAFENO: Debido a la falta de tiempo, no se han podido realizar demasiadas pruebas con el grafeno que se ha conseguido. Sin embargo, una investigación más profunda realizada con el material disponible podría llevarnos a la creación de un sensor de contacto mucho mejor que el que se ha conseguido realizar, por lo que sería interesante, en un futuro, continuar realizando pruebas con las muestras de grafeno conseguidas.

5.2. CONCLUSIONES

En este apartado trataremos de evaluar, con el resultado final sobre la mesa, las funcionalidades de nuestra prótesis, así como sus ventajas e inconvenientes.

Por un lado, la principal ventaja de nuestra mano pretendía ser su bajo presupuesto, y así ha sido, pues el gasto total en ella ha sido inferior a los 300€. Sin embargo, además del presupuesto hay que evaluar la funcionalidad de esta.

Se ha visto que la mano tiene muchas posibilidades, pues es capaz de realizar gran cantidad de movimientos, e incluso agarrar y sujetar objetos de un cierto peso, sin embargo, antes de que sea completamente funcional necesitaría la implementación de ciertas mejoras.

La primera, y la más necesaria es la creación de un dispositivo de acoplo para unir la mano al muñón. Este dispositivo no resultaría caro de fabricar, sin embargo, sí que requeriría un tiempo del que no disponemos en este TFG. La otra mejora necesaria sería la incorporación de una batería que alimentara el sistema, para que de esta forma nos independizásemos del PC y de la fuente de alimentación.

Esto nos hace concluir que nuestra mano no resulta del todo funcional, pero que no sería demasiado difícil hacerla totalmente funcional empleando algo más de trabajo en ella.

Por otro lado, es importante también mencionar las desventajas de nuestra mano. Una de las desventajas más importantes es la falta de sensorización, lo cual le resta bastante funcionalidad a la mano. Otra desventaja a destacar es su peso y su tamaño, lo cual es meramente estético, pero tiene una gran importancia a la hora de adaptarse a una prótesis, pues ayuda a que el usuario se sienta cómodo con ella.

BIBLIOGRAFÍA

[1]. Maturana, J. (2014). *Xataka*. Obtenido de Xataka: http://www.xataka.com/perifericos/estas-son-las-tecnologias-de-impresion-3d-que-hay-sobre-la-mesa-y-lo-que-puedes-esperar-de-ellas
[2]. Burón, D. (Noviembre de 2013). *Silicon*. Obtenido de Silicon: http://www.silicon.es/impresion-tridimensional-llega-el-futuro-de-los-sistemas-de-produccion-49043
[3]. Dorador, J. M, y Rios, P., Robótica y Prótesis inteligentes. Revista Digital Universitaria, Vol. 6, pp. 1-15, 1067-6079. [ed.] UNAM, 2004.
[4]. Lopez, C, E., Una Introducción a la Aleaciones con Memoría de Forma. Universidad Nacional Atonoma de Mexico, 2002.
[5]. Lafont, M. P., Lantana, D. A. y Martinez, R.I., Polímeros con Memoria de Forma en el desarrollo de dispositivos médicos. Universidad Católica del Perú, 2007.
[6]. Arias, M.L.S. y Vanegas, U.L., Materiales compuestos inteligentes. Scientia et Technica Año X, No 25. pp. 143-148, 2004.
[7]. Alia, M. A., Estudio e implementación de sensores de fuerza 3D con aplicación a manos robóticas. Proyecto fin de carrera. Ingeniería Industrial. Universidad Carlos III de Madrid. Escuela Politécnica Superior, España, 2010.
[8]. García Rodríguez, J., "Sistema de control de dispositivos externos mediante el electromiograma: Aplicación a una mano robótica". Proyecto fin de carrera. Ingeniería de Telecomunicaciones. Universidad de Valladolid. Escuela Técnica Superior de Ingenieros de Telecomunicación, España, 2014.
[9]. Pérez Benavente, R. (septiembre de 2014). Obtenido de El Confidencial: http://www.elconfidencial.com/tecnologia/2014-09-17/dextrus-la-mano-robotica-de-bajo-coste-construida-con-el-plastico-de-los-lego_198529/
[10]. Histología UC, "Tejido muscular", Escuela de Medicina P. Universidad Católica de Chile.
[11]. Barea Navarro, R. "Instrumentación Biomédica", Departamento Electrónica. Universidad Alcalá.
[12]. Zecca M., Micera S., Caroza M., Dario P., "Control Of Multifunctional Prosthetic Hands By Processing The Electromyographic Signal", Critical Reviews in Biomedical Engineering. Vol.30, pp.459 485.2002.
[13]. Englehart K., Hudgins B., Parker P., Stevenson M., "Classification of the Myoelectric Signal Using Timefrecuency Based Representations". Institute of Biomedical Engineering, University of New Brunswick, Canada. 1999.
[14]. Propiedades y aplicaciones del grafeno, Claramaría Rodríguez González, Oxana Vasilievna Kharissova.
[15]. Ha nacido una estrella. El grafeno, Amadeo L. Vázquez de Parga.
[16]. http://graphene-flagship.eu/
[17]. http://www.instructables.com/id/Dextrus-v11-Robotic-Hand/
[18]. Ton-Tai Pan, Ping-Lin Fan, Huihua Kenny Chiang, Rong-Seng Chang, and Joe-Air Jiang, "Mechatronic Experiments Course Design: A Myoelectric Controlled Partial-Hand Prosthesis Project", IEEE transactions on education, vol. 47, no. 3, august 2004.
[19]. Laboratorio de electrónica y bioingeniería, Universidad de Valladolid, http://www.biolab.tel.uva.es/.
[20]. Villamizar Pinzón, J., Padilla Mayorga, R., Cabrera Hurtado, G., "Brazo robótico controlado por electromiografía", Scientia et Technica, 2012.
[21]. Carrera González, A "Innovaciones en sistemas e interfaces humano-máquina: aplicación a las tecnologías de rehabilitación", Tesis Doctoral, Universidad de Valladolid, 2013.

[22]. A. S. Sedra, K. C. Smith, “Microelectronic Circuits, 4th edition”, New York: Oxford Univ. Press, 1998.
[23]. S. Franco., “Design With Operational Amplifiers and Analog Integrated Circuits”, New York: McGraw-Hill, 1988.
[24]. Sutton, L. G., Clawson A., Walley Williams III, T., Lipsey, J. H., Sensinger, J. W., “Towards a universal coupler design for modern powered protheses”. University of New Brunswick, Canada. 2011.
[25]. Valencia, I. A. “Caracterización de un termistor NTC. Linealización y acondicionamiento de la señal”. Universidad del Quindío, Programa de Ing. Electrónica.
[26]. Fernández Calleja, S., “Estudio y diseño de un sistema de sustitución sensorial para prótesis de miembro superior”. Proyecto fin de carrera. Ingeniería de Telecomunicaciones. Universidad de Valladolid. Escuela Técnica Superior de Ingenieros de Telecomunicación, España, 2015.
[27]. Giner, J. J., “Medida de parámetros termoeléctricos en un sistema constituido por dispositivos Peltier Seebeck”. Trabajo fin de grado. Ingeniería electrónica. Universitat Politècnica de Catalunya. Escola Politècnica Superior d’Enginyeria de Vilanova i la Geltrú, España, 2013.
[28]. Krausz, N. E., Rorrer R. A. L., Weir R. F., “Design and fabricaton of a six degree-of-freedom open source hand”, IEEE, 2014.

OTRA BIBLIOGRAFÍA CONSULTADA GENÉRICAMENTE

- Puglisi, L. y Moreno, H., Prótesis Robóticas. Revista del Departamento de Automática e Informática Industrial, pp. 1-2, 2006.
- Ventimiglia, P., Design of a Human Hand Prosthesis. Faculty of the Worcester Polytechnic Institute, 2012.
- Loaiza, J. L. y Arzola, N., Evolución y tendencias en el desarrollo de prótesis de mano. Universidad Nacional de Colombia, 2011.
- Brito, J.L., Quinde, M. X., Cusco, D. y Calle J. I., Estudio del estado del arte de las prótesis de mano. Universidad Politécnica Salesiana, 2013.
- Vitali, R. y Andews, H., Amputaciones y Prótesis. JIMS, Lodres, 1996.
- Samuel, K., Weber, J. and Heff, H. Biomechanical Desing of and Powered. IEEE, Noordwijk, 2007.
- García Pinillos, J "Mejora de la autonomía del sistema de adquisición de señales bioeléctricas de la plataforma UVa-NTS", Proyecto fin de carrera, ETSI de Telecomunicación, Universidad de Valladolid, 2010.
- Driscoll, F. F. (1999). *Amplificadores operacionales y circuitos integrados lineales.* Pearson Education.

ANEXOS

ANEXO 1: PROCESO DE MONTAJE DE LA IMPRESORA 3D

Como ya hemos dicho anteriormente la impresora que vamos a emplear es una impresora del tipo FDM, en concreto una 'Prusa i3 Hephestos' del fabricante BQ. Esta impresora, al igual que la gran mayoría de las de su clase, se comercializan sin montar, dado que esto permite ahorrar costes para la empresa que lo fabrica.

Este ahorro de costes por parte de las empresas, se traduce en un precio de venta al público más asequible, lo cual en impresoras 3D de este tipo es un factor muy a tener en cuenta. El público al que están dirigidas estas impresoras busca tener una impresora 3D de calidad, pero con el menor coste posible, dado que, aunque en los últimos años estas han bajado su precio considerablemente, siguen siendo un gasto importante para la mayoría de los hogares.

Aparte del coste, el hecho de tener que montarnos nosotros mismos nuestra propia impresora nos ayuda a entender mejor su funcionamiento, y a poder repararla en caso de rotura o desgaste de alguna de sus piezas. En el caso de BQ, el propio fabricante proporciona los planos de las piezas que están impresas con impresora 3D, para que uno mismo pueda imprimirlas, facilitando aún más el proceso de reparación, en el caso de que sea una de estas piezas la que se ha deteriorado.

En las siguientes páginas web podemos encontrar, respectivamente, los manuales de montaje y dos vídeos en los que se describe el proceso de montaje. En el primer vídeo se describe el proceso de montaje de forma rápida, mientras que el segundo se describe de una forma más detallada:

http://www.imprimalia3d.com/recursosimpresion3d/manuales-prusa-i3-hephestos

http://diwo.bq.com/video/guia-de-montaje-de-la-prusa-i3-hephestos/

http://diwo.bq.com/video/montaje-prusa-i3-hephestos-ingles/

El proceso de montaje de la impresora es largo y puede variar de unas impresoras a otras, en concreto, para el modelo que he montado en este TFG se han empleado unas 15 horas. Este proceso cuenta con algunos pasos difíciles, en los que se ha empleado mucho tiempo por ser la primera vez que se realizaba un montaje similar, por lo que posiblemente, alguien con experiencia podría realizar el montaje en la mitad de tiempo.

El proceso de montaje se puede dividir en 6 partes, las cuales vamos a explicar detalladamente a continuación:

1. **PASOS PREVIOS**: Estos pasos no son más que una preparación de las piezas de la impresora, para que el montaje de la misma, sea más fluido, y así ahorrar tiempo. Esta parte tiene varios pasos:
    a. **Preparación de las poleas**: Consiste en introducir cada rodamiento en su pieza impresa correspondiente.
    b. **Embutido de tuercas**: Se deben introducir tuercas dentro de las piezas impresas, con la ayuda de un soldador de estaño, que las caliente y las ayude a entrar. Así se conseguirá que las tuercas queden alojadas dentro de las piezas impresas, logrando una mayor sujeción de estas.

c. **Preparación de los cables de los motores**: Aunque este paso es opcional, es altamente recomendable, ya que ayudará a que nuestra impresora no tenga cables demasiado largos que se puedan enredar y molestar. Es simplemente cortar y soldar de nuevos los cables de los motores, de acuerdo a unas medidas proporcionadas, para que estos tengan la medida exacta y a la hora de realizar el guiado y conexión de los cables, estos no sean demasiado largos.
d. **Preparación del cable de la fuente**: Este paso es muy simple, consiste en introducir los cables correspondientes en el conector Ramps 1.4 en el cual se conectará la fuente de alimentación.

2. **MONTAJE DEL EJE X**: Esta parte está compuesta por 10 pasos a lo largo de los cuales se montará el eje X de la impresora, es decir los ejes que permitirán al extrusor moverse a la izquierda y a la derecha, para poder imprimir las piezas.

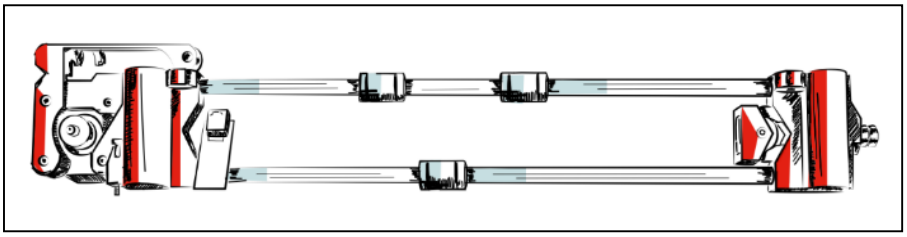


***Imagen 113**. Eje X de la impresora 3D.*

3. **MONTAJE DEL EJE Z**: Ahora se montará el eje Z de la impresora, esto es, el marco que permitirá al extrusor desplazarse hacia arriba y hacia abajo para darle altura a las piezas que se van a imprimir. En esta parte se incluye la unión del eje X con el eje Z. Consta de 9 pasos.

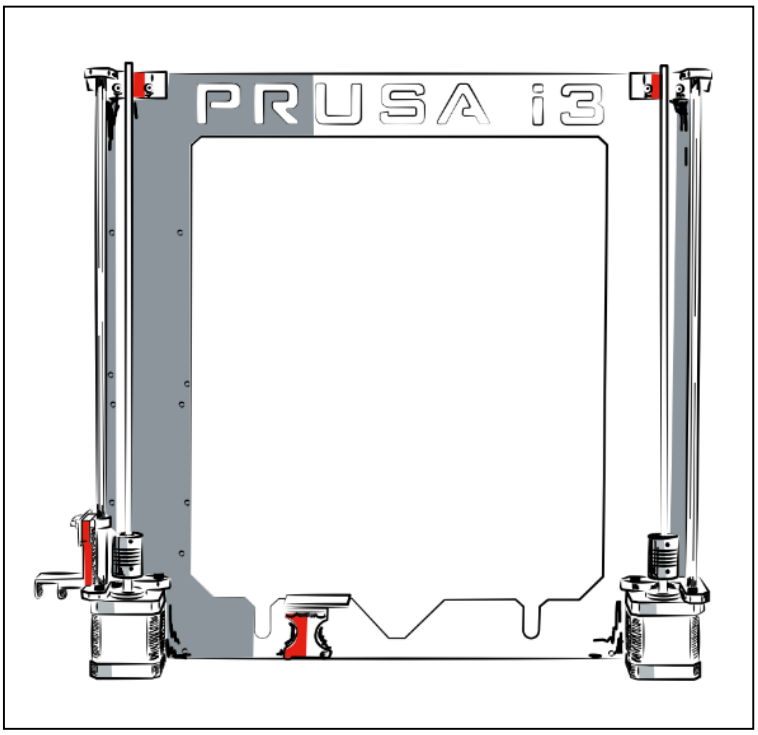


***Imagen 114**. Eje Z de la impresora 3D.*

4. **MONTAJE DEL EJE Y**: En esta parte se ensambla el eje Y de la impresora. Este, a diferencia de los dos ejes anteriores, no mueve el extrusor, sino que mueve la base en la que se sitúa la pieza, hacia adelante y hacia atrás. Este eje, junto con el eje X, permiten a la impresora dar largo y ancho a las piezas que se están imprimiendo. En esta parte también se incluye la unión de los dos ejes anteriores con el eje Y. Esta parte es sensiblemente más larga que las anteriores pues consta de 22 pasos.

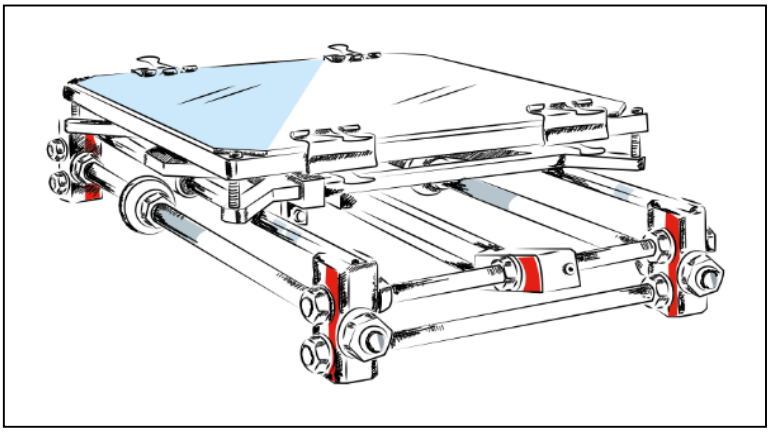


***Imagen 115**. Eje Y de la impresora 3D.*

5. **MONTAJE DEL EXTRUSOR**: Esta parte es muy sencilla, es simplemente montar el extrusor y colocarlo en su posición sobre el eje X. Tiene 8 pasos.

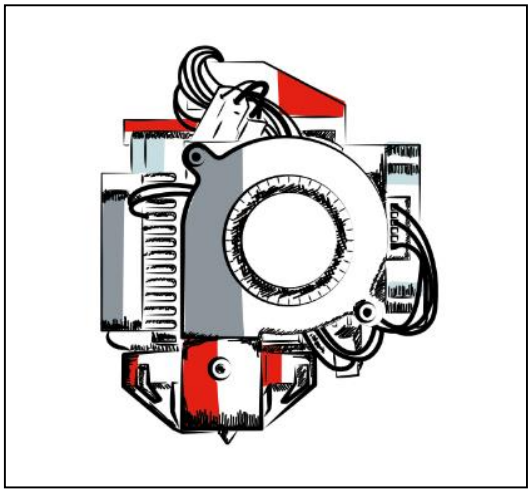


***Imagen 116**. Extrusor de la impresora 3D.*

6. **MONTAJE DE LA ELECTRÓNICA**: En esta parte hay que poner especial atención a las indicaciones, y realizar los pasos con sumo cuidado, pues en función de la manera en la que realicemos estos pasos, los cables de la impresora nos quedarán más o menos enmarañados. Es importante que los cables queden ordenados, pues de lo contrario, alguna de las partes móviles de la impresora podría engancharse en un cable provocando la rotura de alguna pieza. Además, un correcto orden en los cables de la impresora facilitaría mucho el proceso de sustitución de alguna pieza de la parte electrónica en caso de deterioro de esta. Tras esta parte, nuestra impresora quedaría completamente montada, y podríamos pasar a la calibración de esta para poder realizar nuestra primera impresión.

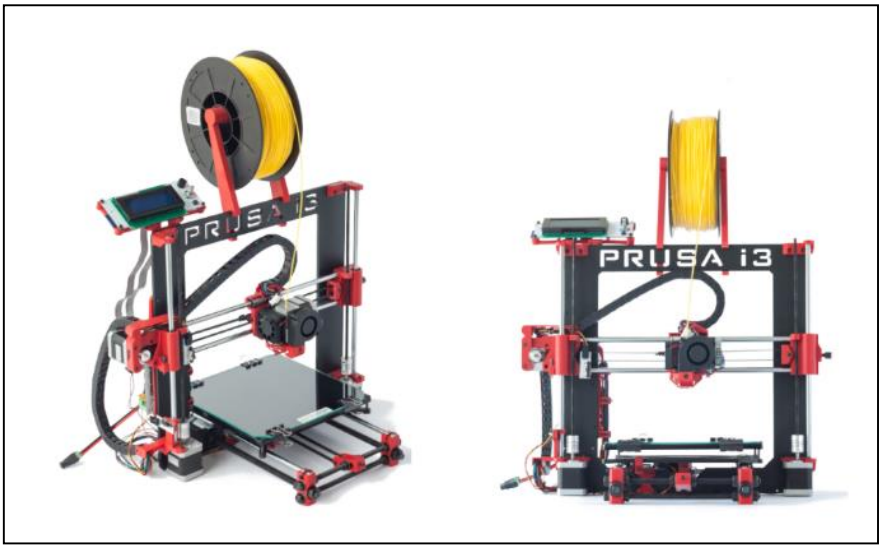


***Imagen 117**. Impresora 3D completamente montada.*

Aquí no se ha hecho más que un resumen del proceso completo de montaje, ya que este viene lo suficientemente claro en los manuales proporcionados por el fabricante junto con la impresora. Estos, unidos a los vídeos explicativos del montaje, disponibles en la web de BQ son una perfecta guía de montaje de la impresora. Sin embargo, en el caso de presentarse algún problema en el proceso, siempre se podrá acudir, o bien al propio soporte de BQ, o bien a los foros RepRap, donde hay un gran número de usuarios con una Prusa i3 Hephestos, de los cuales seguro que alguno puede resolver nuestras dudas.

Con nuestra impresora montada, debemos pasar a la calibración. Este proceso consta de dos fases: La primera en la que ajustaremos el eje Z para que la punta del extrusor quede a una distancia adecuada de la base donde se realizará la impresión, y la segunda en la que se ajustará la base para que quede totalmente nivelada. Este proceso es algo tedioso, y hay que hacerlo con cierta frecuencia, sin embargo, no debería tardar en realizarse más de media hora. El proceso completo de calibración de la impresora, junto con otros datos de interés, cómo mantenimiento de esta, software de impresión a emplear… se encuentra detallado en el anexo 2 de este documento.

ANEXO 2: PROCESO DE IMPRESIÓN 3D

IMPRESORA UTILIZADA

Hemos empleado una impresora de Fused Filament Fabrication (FFF), en concreto la Prusa i3 Hephestos del fabricante BQ. Esta impresora tiene un precio de 499,90€ en su web oficial, lo cual la hace una impresora 3D asequible, pero sin perder las garantías y servicio postventa que nos ofrece una gran empresa cómo BQ.

El material empleado es PLA, un plástico de origen orgánico, cuyos componentes básicos son plantas cómo el maíz. Entre sus ventajas está que no emite gases nocivos y que tiene un mayor rango de colores. Por otro lado, sus desventajas son que no admite altas temperaturas (Se empieza a descomponer a 50-60 ºC) y que el postprocesado de este es más complicado (Mecanizado, pintado y sobre todo pegado).

SOFTWARE UTILIZADO

Hay una gran cantidad de software que nos permiten realizar impresiones con nuestra impresora, todos ellos podrían servirnos, teniendo cada uno sus ventajas e inconvenientes, y ante la imposibilidad de probar todos ellos por motivos de tiempo, nos hemos decidido por uno recomendado por BQ en su página web, el Cura.

Lo primero de todo es descargarnos la versión más reciente del software desde su web oficial (https://ultimaker.com/en/products/cura-software). Posteriormente realizaremos la instalación de este seleccionando las preferencias de instalación que deseemos (Ruta de instalación, accesos directos al programa…). Esta instalación es muy sencilla e intuitiva y no necesita indicaciones concretas para poder realizarse adecuadamente.

Una vez hecho esto, pasaremos a configurar el programa de acuerdo a la impresora y al carrete de PLA que estamos usando. Empezaremos iniciando el programa, y conectando la impresora por USB para que instale los drivers (En Windows se instalan automaticamente). Seguidamente, en la barra superior, en el menú desplegable 'Machine' seleccionaremos 'Add new machine'. Se nos abrirá una nueva ventana en la que indicaremos que nuestra impresora es del tipo 'Other (Ex: RepRap, MakerBot, Witbox)' y en el siguiente paso, cuando se nos muestren los perfiles preconfigurados disponibles indicaremos 'Hephestos'.

Con esto habremos cargado la configuración básica concreta para nuestra impresora. Sin embargo, aún podemos configurar varios parámetros, de los cuales nos interesan principalmente los siguientes:

- 'Layer height (mm)': Nos da la anchura de cada capa, parámetro inversamente relacionado con la resolución de la impresión. A más altura menos resolución. Esta puede variar entre 0,06mm y 0,25mm.
- 'Printing Temperature (C)': Nos indicará la temperatura a la que se calienta el material que usa la impresora. Para el caso del PLA, es suficiente con una temperatura de 210 ºC.
- 'Diameter (mm)': Nos permite indicarle al programa el diámetro del filamento que vamos a emplear. Para este TFG se ha empleado un filamento de 1,75mm de grosor.

Se pueden hacer múltiples cambios sobre la pieza a imprimir directamente con Cura, de los cuales destacamos:

- Se puede variar la posición de la pieza sobre la base sin más que pinchar con el botón izquierdo y arrastrar.
- Se puede variar el tamaño, para ello pincharemos sobre la pieza, y posteriormente, sobre el icono central ('Scale') de los tres que habrán aparecido en la parte inferior. Pulsando y arrastrando sobre los diferentes ejes que habrán aparecido sobre la figura podremos variar el tamaño de esta.
- Se puede rotar la pieza, para realizar la impresión sobre la cara con mayor superficie. Para ello pulsaremos sobre la pieza y posteriormente sobre el primero ('Rotate') de los tres iconos que habrán aparecido abajo. Pinchando y arrastrando sobre cada uno de los 3 ejes circulares que habrán aparecido sobre la pieza podremos girar esta. Cada uno de los ejes circulares nos girará la pieza sobre uno de los ejes cartesianos. Es importante la función 'Lay Flat', cuyo icono aparece encima de 'Rotate' al pulsar sobre este. Dicha función nos tumbará la pieza sobre la superficie plana más cercana, aunque a veces no trabaja todo lo bien que debería.

DONDE CONSEGUIR MODELOS

Hay muchas webs en la que se pueden conseguir modelos, tanto gratuitos, como de pago, creados tanto por particulares, como por empresas. Se han revisado varias webs (123D Gallery, CubeHero, Thingiverse…) siendo Thingiverse la más empleada al realizar este TFG, debido a su interfaz amigable y a la enorme cantidad de modelos disponibles en ella.

- Thingiverse (https://www.thingiverse.com/): Es una web que ofrece mucha descripción para los modelos, comentarios, votaciones de usuarios, manuales en el caso de tener que ensamblar las piezas.... La mayoría de los modelos, son bajo licencia Creative Commons - Attribution - Non-Commercial. Es una de las webs de referencia a la hora de buscar modelos 3d sin coste. En esta web en concreto es donde se ha encontrado el modelo de la mano empleada en el TFG (Dextrus v1.1), así como la gran mayoría de modelos empleados en la calibración y ajuste inicial de la impresora.

Si se tienen conocimientos de modelado 3D, también se podrían crear piezas uno mismo, siempre teniendo en cuenta las dimensiones máximas de impresión de la impresora 3D (XxYxZ 215x210 x180 mm) y que los archivos con los modelos 3D deben estar en formato .stl.

Es muy importante que el modelo a imprimir tenga una base plana suficientemente grande para que sirva de base en la impresión de toda la pieza, pues de lo contrario la base se podría despegar, siendo necesario comenzar de nuevo la impresión desde cero.

Puede haber ciertos casos en los que sea necesario partir la pieza para tener una base sólida con la que imprimir. Por ejemplo, en el caso de necesitar imprimir una esfera lo más conveniente sería partir esta con un programa 3D, e imprimir las dos semiesferas, para posteriormente pegarlas obteniendo así la esfera que queríamos obtener inicialmente.

CALIBRADO DE LA IMPRESORA

Antes de comenzar a imprimir hay que calibrar los 3 ejes de la impresora. Comenzaremos calibrando el eje Z, el cual nos regulará cuanto baja el extrusor, y para acabar calibraremos el plano XY, para que la base de cristal quede plana y se realicen las impresiones correctamente.

Para realizar las calibraciones, basta con seguir las instrucciones de los siguientes vídeos. Con el primero calibraremos el eje Z y con el segundo el plano XY:

https://www.youtube.com/watch?v=gDcAXN6jsOc

https://www.youtube.com/watch?v=UyZDEy34tNY

COLOCACIÓN DEL CARRETE

Tras calibrar el eje Z, y el plano XY, hay que colocar el filamento de PLA dentro del extrusor, para lo cual basta con seguir las sencillas instrucciones de este vídeo (Hasta el minuto 1:30):

https://www.youtube.com/watch?v=s-7whacfyI4

Para retirar un carrete de filamento, cortaremos este a una altura cercana al extrusor, y navegando por la pantalla de la Prusa, iremos a 'Control', luego a 'Filament' y finalmente a 'Unload'. Tras seleccionar 'Unload' solo tendremos que esperar y la impresora soltará automáticamente todo el filamento que quede dentro de ella.

IMPRESIÓN DESDE USB

Una vez configurado el software, realizar una impresión vía USB es muy sencillo, tan solo tenemos que abrir el Cura y cargar el modelo a imprimir, pulsando el icono 'Load'. Situaremos el modelo en la parte de la base que deseemos, le daremos el tamaño que queramos y una vez esté todo listo pulsaremos el botón 'Print from USB'. Se nos abrirá una pequeña ventana, y el programa conectará con la impresora, una vez conectado se nos habilitará el botón 'Print' en esa misma pantalla, el cual pulsaremos. Una vez realizado esto la impresora se calentará y comenzará a imprimir, ya solo tenemos que esperar a que acabe para tener nuestra pieza.

Es importante antes de empezar la impresión rociar la base de cristal con una generosa capa de laca de pelo convencional, para que la base de la pieza se pegue al plato de cristal y no se despegue, arruinándonos la impresión.

IMPRESIÓN DESDE SD

Para realizar una impresión desde una tarjeta SD, deberemos cargar la pieza en el Cura, darla una posición y tamaño adecuados y posteriormente en la barra superior, seleccionaremos el menú desplegable 'File', y dentro de este 'Save GCode'. Esta opción nos creará un archivo .gcode a partir de nuestro .stl, pues a la impresora necesita ficheros .gcode para trabajar.

Una vez tengamos este fichero, lo meteremos en una tarjeta SD (U otro tipo de tarjeta con adaptador SD) formateada en FAT32 e introduciremos esta en la ranura SD de la impresora, que se encuentra en la parte superior trasera de la placa donde se encuentra la pantalla.

***Imagen 118**. Ranura para tarjetas SD de la Prusa i3 Hephestos.*

Con la SD metida en la ranura, encenderemos la impresora y seguiremos los pasos que se nos indican en este vídeo (A partir del minuto 2:13):

https://www.youtube.com/watch?v=s-7whacfyI4

En este punto, tendremos que hacer uso de la laca de nuevo para que la impresión se realice correctamente.

PRECAUCIONES A TENER EN CUENTA

- Es de vital importancia, comprobar antes de cada impresión que en el eje Z, el tornillo que presiona el interruptor final de carrera está alineado con dicho interruptor.

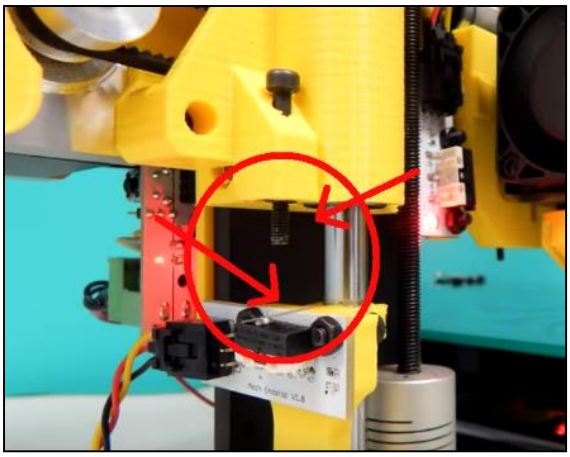


***Imagen 119**. Interruptor final de carrera del eje Z.*

- Existen otros dos interruptores de final de carrera (Para los ejes X e Y), los cuales al estar resguardados es más difícil que se descoloquen, pero sí que resulta conveniente revisarlos cada cierto tiempo para comprobar que los tornillos y los interruptores de final de carrera continúan alineados. Para el caso del eje X, no es un tornillo el encargado de presionar el interruptor final de carrera, sino un pico de plástico que se encuentra en el soporte del extrusor.

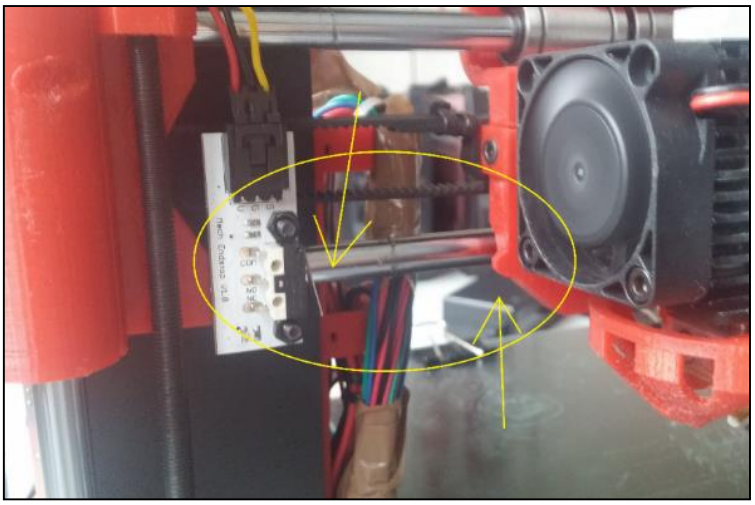


***Imagen 120**. Interruptor final de carrera del eje X.*

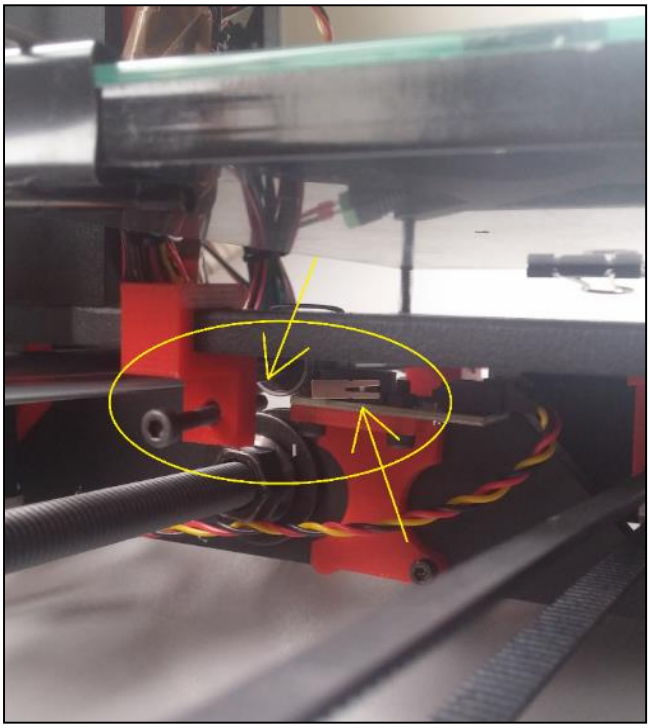


***Imagen 121**. Interruptor final de carrera del eje Y.*

- No es necesario calibrar la impresora con cada impresión. Sin embargo, si la impresora se ha movido, lleva mucho tiempo sin utilizarse, o no se ha calibrado en mucho tiempo, es conveniente realizar de nuevo la calibración.
- No es necesario limpiar el plato de cristal tras cada impresión. No obstante, sí que es conveniente hacerlo cada día o dos días de uso, para que no se junten los residuos con las nuevas piezas y para que estos no se queden demasiado pegados a la base.
- Es conveniente revisar periódicamente la impresión en curso, dado que, si la base de la pieza se desprende del plato de cristal, hay que detener la impresión, retirar los restos del plato de cristal y de la punta del extrusor, y volver a realizar la impresión desde cero, esta vez echando más laca sobre la base para que la pieza se pegue mejor. De no detener la impresión y dejarla finalizar, existiría un alto riesgo de que se formase una masa de plástico sobre el extrusor, pudiendo resultar muy complicado retirarla y limpiar la zona posteriormente.
- A la hora de imprimir una pieza tratar de situarla con Cura en una parte de la base en la que no haya residuos de otras impresiones anteriores, para evitar posibles problemas derivados de la superposición de la nueva impresión con restos de impresiones anteriores.
- Para retirar las piezas que estén muy pegadas de la base de cristal, es necesario darlas un golpe seco. Es conveniente retirar el plato de cristal de la impresora, para así despegar las piezas más cómodamente, para poder retirar el plato de la impresora basta con quitar las cuatro pinzas que lo sujetan. Para piezas que son muy bajas, en vez de un golpe seco, las separamos introduciendo algo fino y duro (Como la cuchilla de un cúter) entre la pieza y la base de cristal, para que esta se empiece a despegar. Posteriormente haremos palanca con lo que hemos introducido entre la base y la pieza hasta que esta se termine de despegar.

ANEXO 3: CÓDIGO DE ARDUINO PARA LA APLICACIÓN DE CONTROL DE POSICIONES

```
#include <Servo.h>        // Incluimos la librería dedicada a Servo-
motores

Servo myservo1;           // Declaramos 5 variables tipo 'servo' que
posteriormente asociaremos a cada uno de ellos
Servo myservo2;
Servo myservo3;
Servo myservo4;
Servo myservo5;

int led1=13;

int val1 = 0;             // Declaramos dos variables asociados a cada
una de los dos canales de entrada
int val2 = 0;

int flag1=0;              // Declaramos los flags que nos servirán a
la hora de cambiar de posición
int flag2=0;

int posicion1=0;

void setup() {           // Cada vez que pulsemos reset, se
inicializará la rutina:

 Serial.begin(9600);
 Serial.print(posicion1);  // Se muestra en pantalla el mensaje
 Serial.print(" Posición y valor:  ");


   myservo1.attach(11);    // Liga el servo con el pin de salida
indicado
   myservo2.attach(10);
   myservo3.attach(9);
   myservo4.attach(6);
   myservo5.attach(5);

   pinMode(led1,OUTPUT);
   digitalWrite(led1,HIGH);  // Activamos el LED indicador de
encendido el conectar el dispositivo


}


                                                  // El resto de la
rutina se ajecutará posteriormente:
void loop() {

 val1=analogRead(A0);                            // Asociamos las
entradas analógicas A0 y A1 provinientes de cada uno de los canales,
con las variables val1 y val2
```

```
 val2=analogRead(A1);

 //Serial.println(val1);
// Serial.println(val2);

 val1=constrain(val1,0,650);                    // Restringimos el
rango de valores de entrada de 0-650 en ambas variables
 val2=constrain(val2,0,650);

 val1 = map(val1,0,650,0,300);                  // Remapeamos los
valores para facilitar la calibración
 val2 = map(val2,0,650,0,300);

 Serial.println(posicion1);                     // Mostramos por la
pantalla serial del entorno Arduino el valor del canal1 y la posición
 Serial.println(val1);

 if(val2<150){                                  // Si por el canal 2 no
se recibe el nivel de impulso necesario, el tipo de posición no cambia
 }

 else if(val2>150 && posicion1!=3){             // Si por el canal 2 se
recibe el nivel de impulso necesario y no es la última posición, se
incrementa la posición
   posicion1=posicion1+1;
   delay(50);
 }

else if(val2>150 && posicion1==3){              // Si por el canal 2 se
recibe el nivel de impulso necesario y es la última posición, se pasa
a la primera posición
   posicion1=1;
   delay(50);
}
  switch (posicion1){                           // Código pertinente a
la posición inicial
       case 0:
                 myservo1.write(160);           // Indicamos el ángulo
que queremos que adquiera cada servo al alimentar el sistema
                 delay(20);                     // debemos tener en
cuenta el sentido físico del servomotor
                 myservo2.write(180);
                 delay(20);
                 myservo3.write(0);
                 delay(20);
                 myservo4.write(0);
                 delay(20);
                 myservo5.write(135);
                 delay(20);

                 flag1=0;
                 flag2=0;

                 delay(500);
        break;

        case 1:                                         // Código
pertinente a la posición 1
                 if(val1<150 && flag1==0) {             // Indicamos el
ángulo que queremos que adquiera cada servo en la posición1 y acción
off
```

```
                myservo1.write(160);
                delay(20);
                myservo2.write(180);
                delay(20);
                myservo3.write(0);
                delay(20);
                myservo4.write(0);
                delay(20);
                myservo5.write(135);
                delay(20);

                flag1=0;
                flag2=0;

                delay(500);                     // Retraso necesario
para que se ejecute la acción y no seguir adquiriendo valores
                }

                else if(val1<150 && flag1==1) {  // Indicamos el
ángulo que queremos que adquiera cada servo en la posición1, cuando no
se activa el valor1
                myservo1.write(20);             // y estamos con la
acción ON. Se mantendrá la acción ON
                delay(20);
                myservo2.write(0);
                delay(20);
                myservo3.write(120);
                delay(20);
                myservo4.write(150);
                delay(20);
                myservo5.write(73);
                delay(20);

                flag1=1;
                flag2=0;

                delay(500);
                }

                else if(val1>150 && flag1==0) {  // Indicamos el
ángulo que queremos que adquiera cada servo en la posición1, cuando se
activa el valor1
                                                // y estamos con la
acción OFF. Se ejecutará la acción ON
                myservo1.write(20);
                delay(20);
                myservo2.write(0);
                delay(20);
                myservo3.write(120);
                delay(20);
                myservo4.write(150);
                delay(20);
                myservo5.write(73);
                delay(20);

                flag1=1;
                flag2=0;

                delay(500);
                }
```

```
                else if(val1>150 && flag1==1) {  // Indicamos el
ángulo que queremos que adquiera cada servo en la posición1, cuando se
activa el valor1
                                                 // y estamos con la
acción ON. Se ejecutará la acción OFF
                myservo1.write(160);
                delay(20);
                myservo2.write(180);
                delay(20);
                myservo3.write(0);
                delay(20);
                myservo4.write(0);
                delay(20);
                myservo5.write(135);
                delay(20);

                flag1=0;
                flag2=0;

                delay(500);
                }
      break;

      case 2:                                    // Código pertinente
a la posición 2

                if(val1<150 && flag1==0) {
                myservo1.write(160);
                delay(20);
                myservo2.write(180);
                delay(20);
                myservo3.write(0);
                delay(20);
                myservo4.write(0);
                delay(20);
                myservo5.write(135);
                delay(20);

                flag1=0;
                flag2=0;

                delay(500);
                }

                else if(val1<150 && flag1==1) {
                myservo1.write(0);
                delay(20);
                myservo2.write(0);
                delay(20);
                myservo3.write(0);
                delay(20);
                myservo4.write(0);
                delay(20);
                myservo5.write(20);
                delay(20);

                flag1=1;
                flag2=0;

                delay(500);
```

```
                }

                else if(val1>150 && flag1==0) {

                myservo1.write(0);
                delay(20);
                myservo2.write(0);
                delay(20);
                myservo3.write(0);
                delay(20);
                myservo4.write(0);
                delay(20);
                myservo5.write(20);
                delay(20);

                flag1=1;
                flag2=0;

                delay(500);
                }

                else if(val1>150 && flag1==1) {

                myservo1.write(160);
                delay(20);
                myservo2.write(180);
                delay(20);
                myservo3.write(0);
                delay(20);
                myservo4.write(0);
                delay(20);
                myservo5.write(135);
                delay(20);

                flag1=0;
                flag2=0;

                delay(500);

                }
      break;

      case 3:                                   // Código pertinente a
la posición 3
                if(val1<150 && flag1==0) {

                myservo1.write(160);
                delay(20);
                myservo2.write(180);
                delay(20);
                myservo3.write(0);
                delay(20);
                myservo4.write(0);
                delay(20);
                myservo5.write(135);
                delay(20);

                flag1=0;
                flag2=0;

                delay(500);
```

```
                    }

                    else if(val1<150 && flag1==1) {
                    myservo1.write(60);
                    delay(20);
                    myservo2.write(35);
                    delay(20);
                    myservo3.write(100);
                    delay(20);
                    myservo4.write(120);
                    delay(20);
                    myservo5.write(70);
                    delay(20);

                    flag1=1;
                    flag2=0;

                    delay(500);
                    }

                    else if(val1>150 && flag1==0) {

                    myservo1.write(60);
                    delay(20);
                    myservo2.write(35);
                    delay(20);
                    myservo3.write(100);
                    delay(20);
                    myservo4.write(120);
                    delay(20);
                    myservo5.write(70);
                    delay(20);

                    flag1=1;
                    flag2=0;

                    delay(500);
                    }

                    else if(val1>150 && flag1==1) {

                    myservo1.write(160);
                    delay(20);
                    myservo2.write(180);
                    delay(20);
                    myservo3.write(0);
                    delay(20);
                    myservo4.write(0);
                    delay(20);
                    myservo5.write(135);
                    delay(20);

                    flag1=0;
                    flag2=0;

                    delay(500);
                    }
      break;
           }
}
```

ANEXO 4: CÓDIGO DE ARDUINO PARA LA APLICACIÓN DE CONTROL DE ESTADOS

```
#include <Servo.h>         // Incluimos la librería dedicada a Servo-
motores

Servo myservo1;            // Declaramos 5 variables tipo 'servo' que
posteriormente asociaremos a cada uno de ellos
Servo myservo2;
Servo myservo3;
Servo myservo4;
Servo myservo5;

int led1=13;

int val1 = 0;              // Declaramos dos variables asociados a cada
una de los dos canales de entrada
int val2 = 0;

int estadoActual=0;        // Declaramos las variables estadoActual y
estadoAnterior, pues lo implementaremos como una máquina de estados
int estadoAnterior=0;

unsigned long tiempoEstado;  // Declaramos las variables tiempoActual
y tiempoAnterior para asociar el tiempo a cada estado
unsigned long tiempoActual;

unsigned long mostrar=0;  // Variable auxiliar para controlar las
impresiones en el monitor serial



void setup() {             // Cada vez que pulsemos reset, se
inicializará la rutina:

  Serial.begin(9600);

   myservo1.attach(11);    // Ligamos cada servo con el pin de salida
indicado
   myservo2.attach(10);
   myservo3.attach(9);
   myservo4.attach(6);
   myservo5.attach(5);

   pinMode(led1,OUTPUT);
   digitalWrite(led1,HIGH);  // Avtivamoe el LED indicador de
encendido

   myservo1.write(160);    // Posición inicial al encender el
dispositivo
   delay(20);
   myservo2.write(180);
   delay(20);
   myservo3.write(0);
   delay(20);
```

```
   myservo4.write(0);
   delay(20);
   myservo5.write(135);
   delay(20);

}


void loop() {

  val1=analogRead(A0);               // Asociamos las entradas
analógicas A0 y A1 provinientes de cada uno de los canales, con las
variables val1 y val2
  val2=analogRead(A1);

  val1=constrain(val1,0,650);        // Restringimos el rango de
valores de entrada de 0-650 en ambas variables
  val2=constrain(val2,0,650);

  val1 = map(val1,0,650,0,300);      // Remapeamos los valores para
facilitar la calibración
  val2 = map(val2,0,650,0,300);

   if((millis()-mostrar)>200)        // Imprimimos por pantalla cada
200 msg os valores de entrada ya en el rango deseado
  {
    //Serial.println('Canal 1:');
    Serial.println(val1);
    //Serial.println('Canal 2:');
    Serial.println(val2);
    mostrar=millis();               // La variable mostrar se
actualiza con el instante de tiempo cada 200 milisegundos
  }


  switch (val1){
  case 0 ... 70:                                    // NIVEL BAJO CANAL1

    switch (val2){
    case 0 ... 70:                                  // NIVEL BAJO-BAJO

      if(estadoActual != 1)
      {
        estadoAnterior = estadoActual;
        estadoActual = 1;
        tiempoEstado = millis();
      }
      tiempoActual = millis();

      break;


    case 71 ... 220:                        // NIVEL BAJO-MEDIO

      //supongo que el estado en el que entramos es el 2.
      //actualización de variables para todos los case.
      if(estadoActual != 2)
      {
        estadoAnterior = estadoActual;
        estadoActual = 2;
```

```
        tiempoEstado = millis();            // Una vez estamos en un
determinado estado, se actualiza la variable tiempoEstado con el
instante actual en milisegundos
      }
      tiempoActual = millis();
      if((tiempoActual - tiempoEstado) > 200)    // Si se permanece al
menos 200 msg en el estado, se ejecutará la acción asociada al mismo.
      {

                  myservo1.write(0);
                  delay(20);
                  myservo2.write(0);
                  delay(20);
                  myservo3.write(0);
                  delay(20);
                  myservo4.write(0);
                  delay(20);
                  myservo5.write(20);
                  delay(20);

      Serial.print("BAJO-MEDIO");
      delay(1000);
      }
      break;

    case 221 ... 300:                       // NIVEL BAJO-ALTO

      //supongo que el estado en el que entramos es el 3.
      //actualización de variables para todos los case.
      if(estadoActual != 3)
      {
        estadoAnterior = estadoActual;
        estadoActual = 3;
        tiempoEstado = millis();
      }
      tiempoActual = millis();
      if((tiempoActual - tiempoEstado) > 200)
      {

                  myservo1.write(160);
                  delay(20);
                  myservo2.write(180);
                  delay(20);
                  myservo3.write(0);
                  delay(20);
                  myservo4.write(0);
                  delay(20);
                  myservo5.write(135);
                  delay(20);

       Serial.print("BAJO-ALTO");

      delay(1000);
      }
      break;
    }
    break;


  case 71 ... 220:
    switch (val2){
```

```
case 0 ... 70:                        // NIVEL MEDIO-BAJO

  //supongo que el estado en el que entramos es el 4.
  //actualización de variables para todos los case.
  if(estadoActual != 4)
  {
    estadoAnterior = estadoActual;
    estadoActual = 4;
    tiempoEstado = millis();
  }
  tiempoActual = millis();
  if((tiempoActual - tiempoEstado) > 200)
  {

            myservo1.write(60);
            delay(20);
            myservo2.write(35);
            delay(20);
            myservo3.write(100);
            delay(20);
            myservo4.write(120);
            delay(20);
            myservo5.write(73);
            delay(20);

  Serial.print("MEDIO-BAJO");
  delay(1000);
  }
  break;


case 71 ... 220:                        // NIVEL MEDIO-MEDIO

  //supongo que el estado en el que entramos es el 5.
  //actualización de variables para todos los case.
  if(estadoActual != 5)
  {
    estadoAnterior = estadoActual;
    estadoActual = 5;
    tiempoEstado = millis();
  }
  tiempoActual = millis();
  if((tiempoActual - tiempoEstado) > 200)
  {
            myservo1.write(20);
            delay(20);
            myservo2.write(0);
            delay(20);
            myservo3.write(120);
            delay(20);
            myservo4.write(0);
            delay(20);
            myservo5.write(20);
            delay(20);

  Serial.print("MEDIO-MEDIO");
  delay(1000);
  }
  break;

case 221 ... 300:                       // NIVEL MEDIO-ALTO
```

```
    //supongo que el estado en el que entramos es el 6.
    //actualización de variables para todos los case.
    if(estadoActual != 6)
    {
      estadoAnterior = estadoActual;
      estadoActual = 6;
      tiempoEstado = millis();
    }
    tiempoActual = millis();
    if((tiempoActual - tiempoEstado) > 200)
    {
              myservo1.write(160);
              delay(20);
              myservo2.write(180);
              delay(20);
              myservo3.write(0);
              delay(20);
              myservo4.write(0);
              delay(20);
              myservo5.write(20);
              delay(20);
    Serial.print("MEDIO-ALTO");
    delay(1000);
    }
    break;
  }
  break;

case 221 ... 300:
  switch (val2){
  case 0 ... 70:                    // NIVEL ALTO-BAJO

    //supongo que el estado en el que entramos es el 7.
    //actualización de variables para todos los case.
    if(estadoActual != 7)
    {
      estadoAnterior = estadoActual;
      estadoActual = 7;
      tiempoEstado = millis();
    }
    tiempoActual = millis();
    if((tiempoActual - tiempoEstado) > 200)
    {
              myservo1.write(20);
              delay(20);
              myservo2.write(0);
              delay(20);
              myservo3.write(120);
              delay(20);
              myservo4.write(150);
              delay(20);
              myservo5.write(73);
              delay(20);

    Serial.print("ALTO-BAJO");
    delay(1000);
    }
    break;

  case 71 ... 220:                  // NIVEL ALTO-MEDIO
```

```
      //supongo que el estado en el que entramos es el 8.
      //actualización de variables para todos los case.
      if(estadoActual != 8)
      {
        estadoAnterior = estadoActual;
        estadoActual = 8;
        tiempoEstado = millis();
      }
      tiempoActual = millis();
      if((tiempoActual - tiempoEstado) > 200)
      {
                myservo1.write(160);
                delay(20);
                myservo2.write(0);
                delay(20);
                myservo3.write(120);
                delay(20);
                myservo4.write(150);
                delay(20);
                myservo5.write(135);
                delay(20);

      Serial.print("ALTO-MEDIO");
      delay(1000);
      }
      break;

    case 221 ... 300:                       // NIVEL ALTO-ALTO

      //supongo que el estado en el que entramos es el 9.
      //actualización de variables para todos los case.
      if(estadoActual != 9)
      {
        estadoAnterior = estadoActual;
        estadoActual = 9;
        tiempoEstado = millis();
      }
      tiempoActual = millis();
      if((tiempoActual - tiempoEstado) > 200)
      {
                myservo1.write(160);
                delay(20);
                myservo2.write(0);
                delay(20);
                myservo3.write(120);
                delay(20);
                myservo4.write(0);
                delay(20);
                myservo5.write(73);
                delay(20);

      Serial.print("ALTO-ALTO");
      delay(1000);
      }
      break;
    }
    break;
    default: break;
  }
}
```

ANEXO 5: SOFTWARE FRITZING

Fritzing es un software para la creación de esquemas de circuitos electrónicos. Tiene una enorme base de datos con componentes de todo tipo, desde varios tipos de placas Arduino, hasta gran variedad de circuitos integrados, pasando por resistencias, condensadores…

Fritzing es gratuito, y se puede descargar desde la propia web del desarrollador en el siguiente link, donde podremos escoger, tanto la plataforma donde lo queremos usar, como su versión (32 o 64 bits):

http://fritzing.org/download/?donation=0

Para instalarlo, es conveniente leernos previamente, las notas para la instalación para la plataforma con la que estemos trabajando. Podremos acceder a dichas notas a través del siguiente enlace:

http://fritzing.org/download/?donation=0#install

Una vez instalado, antes de empezar a usarlo deberíamos leernos una guía básica de uso para hacernos con la interfaz del programa. La propia web del desarrollador nos ofrece varios manuales para aprender a usar este programa. El manual de iniciación le podemos encontrar en el siguiente link:

http://fritzing.org/learning/get-started/

Para acceder al resto de tutoriales, deberemos dirigirnos al apartado 'Learning' de la propia web del programa. En dicha página encontraremos gran cantidad de tutoriales, aparte del tutorial ´básico' con los cuales podremos profundizar en el uso del programa, adquiriendo así amplios conocimientos de este. Los tutoriales están disponibles en la siguiente página:

http://fritzing.org/learning/

Finalmente, decir que, en el caso de buscar tutoriales en un idioma distinto al inglés, tendremos que acudir a la sección 'Translations' dentro del apartado 'Learning'. En dicha sección podemos encontrar tutoriales en varios idiomas, aunque, ni son los mismos, ni hay tanta variedad de tutoriales como en inglés. Los idiomas para los que existen tutoriales son: español, francés, indonesio, alemán, italiano, polaco y portugués. El link donde se encuentran los tutoriales para los distintos idiomas es el siguiente:

http://fritzing.org/learning/translations

ANEXO 6: CÓDIGO DE ARDUINO PARA LA PRUEBA DE LOS SENSORES BIMORFOS.

```
int motor=12; // Motor en el pin 12
int i=0;
int val1 = 0; // Declaramos una variable asociada a la entrada, valor
// de 0 a 1024. 0 son 0V, 1024 5V, 512 2,5V etc.

void setup() { // Cada vez que pulsemos reset, se inicializará la
rutina:

  Serial.begin(9600); // Baudios puerto serie

   pinMode(motor,OUTPUT); // Motor salida
   pinMode(A1,INPUT); // A1 entrada

}


void loop() {

   val1=analogRead(A1); // Asociamos la entrada A0 con val1
   Serial.println(val1); // Sacamos val1 por pantalla

   if (val1 < 500) { // Si la salida esta a casi 0.1V, activo el motor
     if (i==0){
      i=1;
      digitalWrite(motor,HIGH);  // Activamos el motor
      Serial.println("Mayor que 4,9V aprox");
     }
     //delay(1000);
  }
  else { // Si la salida en mas de 0.1, apago el motor
    if (i==1){
      i=0;
      digitalWrite(motor,LOW);  // Desactivamos el motor
      Serial.println("Menor que 4,9V aprox");
    }
    delay(100);
  }
}
```

ANEXO 7: CÓDIGO DE ARDUINO PARA LA PRUEBA DE LOS DRIVER DE LOS MOTORES.

```
#define E1 11 // Enable pin for motor 1
#define E2 5 //Enable pin for motor 2

#define I1 10 // Control pin 1 for motor 1
#define I2 9 // Control pin 2 for motor 1
#define I3 3 // Control pin 1 for motor 2
#define I4 6 // Control pin 2 for motor 2
int modo;
int i = 0;

void setup() {
    Serial.begin(9600);

    Serial.print(i);

    pinMode(E1, OUTPUT);

    pinMode(I1, OUTPUT);
    pinMode(I2, OUTPUT);
}

void loop() {
    if (i==0){
      Serial.println("Seleccione modo de agarre");
      Serial.println("1.-Agarre fuerte");
      Serial.println("2.-Agarre suave");
      Serial.println("Si selecciona un numero no valido debera esperar
10 segundos antes de volver a escoger");
      i=1;
    }

    if (Serial.available()) {
      modo = Serial.parseInt();

      switch(modo){

        case 1:
          Serial.println("Agarre fuerte seleccionado");

          Serial.println("Cerrando");

          digitalWrite(I1, HIGH);
          digitalWrite(I2, LOW);
          digitalWrite(I3, HIGH);
          digitalWrite(I4, LOW);

          analogWrite(E1,255);
          analogWrite(E2,255);
          delay(10000);
```

```
          Serial.println("Abriendo");

          digitalWrite(I1, LOW);
          digitalWrite(I2, HIGH);
          digitalWrite(I3, LOW);
          digitalWrite(I4, HIGH);

          delay(5000);

          digitalWrite(I1, LOW);
          digitalWrite(I2, LOW);
          digitalWrite(I3, LOW);
          digitalWrite(I4, LOW);
          analogWrite(E1,0);
          analogWrite(E2,0);

          i=0;
        break;

        case 2:
          Serial.println("Agarre suave seleccionado");

          Serial.println("Cerrando");

          digitalWrite(I1, HIGH);
          digitalWrite(I2, LOW);

          analogWrite(E1,255);

          delay(10000);

          Serial.println("Abriendo");

          digitalWrite(I1, LOW);
          digitalWrite(I2, HIGH);

          delay(5000);

          digitalWrite(I1, LOW);
          digitalWrite(I2, LOW);
          analogWrite(E1,0);

          i=0;
        break;

        case 99:
          i=0;
        break;

        default:
         Serial.println("Numero no valido");
         delay(10000);
         i=0;
        break;
      }
    }
}
```

ANEXO 8: CÓDIGO DE ARDUINO PARA LA PRUEBA INDIVIDUAL DE LOS MOTORES.

```
#define E1 3 // Enable Pin for motor 1
#define E2 5 //Enable motor 2
#define E3 9 // Enable Pin for motor 3
#define E4 10 //Enable motor 4
#define E5 11 //Enable motor 4

#define I1 2 // Control pin 1
#define I2 4 // Control pin 2

void setup() {
    Serial.begin(9600);

    pinMode(E1, OUTPUT);
    pinMode(E2, OUTPUT);
    pinMode(E3, OUTPUT);
    pinMode(E4, OUTPUT);
    pinMode(E5, OUTPUT);

    pinMode(I1, OUTPUT);
    pinMode(I2, OUTPUT);
}

void loop() {

    delay(5000);
    Serial.println("Cerrando 1");

          digitalWrite(I1, HIGH);
          digitalWrite(I2, LOW);
          analogWrite(E1,255);

    delay(2000);
    Serial.println("Abriendo 1");

          digitalWrite(I1, LOW);
          digitalWrite(I2, HIGH);

    delay(5000);

          analogWrite(E1,0);

    Serial.println("Cerrando 2");

          digitalWrite(I1, HIGH);
          digitalWrite(I2, LOW);
          analogWrite(E2,255);

    delay(2000);
    Serial.println("Abriendo 2");
```

```
        digitalWrite(I1, LOW);
        digitalWrite(I2, HIGH);

    delay(5000);

        analogWrite(E2,0);

    Serial.println("Cerrando 3");

        digitalWrite(I1, HIGH);
        digitalWrite(I2, LOW);
        analogWrite(E3,255);

    delay(2000);
    Serial.println("Abriendo 3");

        digitalWrite(I1, LOW);
        digitalWrite(I2, HIGH);

     delay(5000);

        analogWrite(E3,0);

    Serial.println("Cerrando 4");

        digitalWrite(I1, HIGH);
        digitalWrite(I2, LOW);
        analogWrite(E4,255);

    delay(2000);
    Serial.println("Abriendo 4");

        digitalWrite(I1, LOW);
        digitalWrite(I2, HIGH);

     delay(7000);

        analogWrite(E4,0);

    Serial.println("Cerrando 5");

        digitalWrite(I1, HIGH);
        digitalWrite(I2, LOW);
        analogWrite(E5,255);

    delay(2000);
    Serial.println("Abriendo 5");

        digitalWrite(I1, LOW);
        digitalWrite(I2, HIGH);

    delay(2000);

        analogWrite(E5,0);

    delay(15000);
}
```

ANEXO 9: CÓDIGO DE ARDUINO PARA LA PRUEBA DE CONTROL PRECISO DE LOS MOTORES.

```
#include <Servo.h>        // Incluimos la librería dedicada a Servo-
motores

Servo srv1;

#define E1 3 // Enable Pin for motor 1
#define E2 5 // Enable Pin for motor 2
#define E3 9 // Enable Pin for motor 3
#define E4 10 // Enable Pin for motor 4
#define E5 11 // Enable Pin for motor 5

#define I1 2  // Control pin 1
#define I2 4  // Control pin 2

int modo;
int i = 0;

void setup() {
    Serial.begin(9600);
    srv1.attach(13);
    srv1.write(110);

    pinMode(E1, OUTPUT);
    pinMode(E2, OUTPUT);
    pinMode(E3, OUTPUT);
    pinMode(E4, OUTPUT);
    pinMode(E5, OUTPUT);

    pinMode(I1, OUTPUT);
    pinMode(I2, OUTPUT);

}

void loop() {
    if (i==0){
      Serial.println("Seleccione modo de agarre");
      Serial.println("1.-Prueba de todos los dedos");
      Serial.println("2.-Agarre suave");
      Serial.println("3.-Agarre fuerte");
      Serial.println("4.-Vete a la mierda");
      Serial.println("5.-Cuernos");
      Serial.println("6.-Dedo pulgar");
      Serial.println("7.-Dedo indice");
      Serial.println("8.-Dedo corazon");
      Serial.println("9.-Dedo anular");
      Serial.println("10.-Dedo meñique");
      Serial.println("11.-Okay");
      Serial.println("12.-Giro del servo");
      Serial.println("13.-Guardar");
```

```
      Serial.println("Seleccione 99 al iniciar para poner la mano en
posicion");
      Serial.println("Seleccione 999 al iniciar el programa para
desplegar este menu");
      Serial.println("Si selecciona un numero no valido debera esperar
5 segundos antes de volver a escoger");
      Serial.println("Para el servo solo debera esperar 2 segundos");
      Serial.println("Tras cada movimiento debe esperar 5 segundos
para seleccionar uno de nuevo");
      i=1;
    }

    if (Serial.available()) {
      modo = Serial.parseInt();

      switch(modo){

        case 1:
          Serial.println("prueba de todos los dedos seleccionado");

          delay(2000);
          Serial.println("Cerrando 1");

            digitalWrite(I1, HIGH);
            digitalWrite(I2, LOW);

            analogWrite(E1,255);

          delay(2000);
          Serial.println("Abriendo 1");

          digitalWrite(I1, LOW);
          digitalWrite(I2, HIGH);

          delay(2000);

            analogWrite(E1,0);

          Serial.println("Cerrando 2");

            digitalWrite(I1, HIGH);
            digitalWrite(I2, LOW);

            analogWrite(E2,255);

          delay(2000);
          Serial.println("Abriendo 2");

            digitalWrite(I1, LOW);
            digitalWrite(I2, HIGH);

          delay(2000);

            analogWrite(E2,0);

          Serial.println("Cerrando 3");

            digitalWrite(I1, HIGH);
            digitalWrite(I2, LOW);

            analogWrite(E3,255);
```

```
  delay(2000);
  Serial.println("Abriendo 3");

    digitalWrite(I1, LOW);
    digitalWrite(I2, HIGH);

  delay(2000);

    analogWrite(E3,0);

  Serial.println("Cerrando 4");

    digitalWrite(I1, HIGH);
    digitalWrite(I2, LOW);

    analogWrite(E4,255);

  delay(2000);
  Serial.println("Abriendo 4");

    digitalWrite(I1, LOW);
    digitalWrite(I2, HIGH);

  delay(2000);

    analogWrite(E4,0);

  Serial.println("Cerrando 5");

    digitalWrite(I1, HIGH);
    digitalWrite(I2, LOW);

    analogWrite(E5,255);

  delay(2000);
  Serial.println("Abriendo 5");

    digitalWrite(I1, LOW);
    digitalWrite(I2, HIGH);

  delay(2000);

    digitalWrite(I1, LOW);
    digitalWrite(I2, LOW);
    analogWrite(E5,0);

  Serial.println("Servo");

  delay(2000);

    srv1.write(110);

  delay(1500);

    srv1.write(25);

  delay(5000);

  i=0;
break;
```

```
case 2:
  Serial.println("Agarre suave seleccionado");

  Serial.println("Cerrando");

    digitalWrite(I1, HIGH);
    digitalWrite(I2, LOW);

    analogWrite(E4,255);
    analogWrite(E5,255);

  delay(5000);

  Serial.println("Abriendo");

    digitalWrite(I1, LOW);
    digitalWrite(I2, HIGH);

  delay(2000);

    digitalWrite(I1, LOW);
    digitalWrite(I2, LOW);

    analogWrite(E4,0);
    analogWrite(E5,0);

  delay(5000);

  i=0;
break;

case 3:
  Serial.println("Agarre fuerte seleccionado");

  Serial.println("Cerrando");

    digitalWrite(I1, HIGH);
    digitalWrite(I2, LOW);

    analogWrite(E1,255);
    analogWrite(E2,255);
    analogWrite(E3,255);
    analogWrite(E4,255);
    analogWrite(E5,255);

  delay(5000);

  Serial.println("Abriendo");

    digitalWrite(I1, LOW);
    digitalWrite(I2, HIGH);

  delay(2000);

    digitalWrite(I1, LOW);
    digitalWrite(I2, LOW);

    analogWrite(E1,0);
    analogWrite(E2,0);
    analogWrite(E3,0);
```

```
    analogWrite(E4,0);
    analogWrite(E5,0);

  delay(5000);

  i=0;
break;

case 4:
  Serial.println("Vete a la mierda seleccionado");

  Serial.println("Cerrando");

    digitalWrite(I1, HIGH);
    digitalWrite(I2, LOW);

    analogWrite(E1,255);
    analogWrite(E2,255);
    analogWrite(E4,255);
    analogWrite(E5,255);
    srv1.write(110);

  delay(5000);

  Serial.println("Abriendo");

    digitalWrite(I1, LOW);
    digitalWrite(I2, HIGH);

  delay(2000);

    digitalWrite(I1, LOW);
    digitalWrite(I2, LOW);

    analogWrite(E1,0);
    analogWrite(E2,0);
    analogWrite(E4,0);
    analogWrite(E5,0);
    srv1.write(25);

  delay(5000);

  i=0;
break;

case 5:
  Serial.println("Cuernos seleccionado");

  Serial.println("Cerrando");

    digitalWrite(I1, HIGH);
    digitalWrite(I2, LOW);

    analogWrite(E2,255);
    analogWrite(E3,255);
    analogWrite(E5,255);

  delay(5000);

  Serial.println("Abriendo");
```

```
    digitalWrite(I1, LOW);
    digitalWrite(I2, HIGH);

  delay(2000);

    digitalWrite(I1, LOW);
    digitalWrite(I2, LOW);

    analogWrite(E2,0);
    analogWrite(E3,0);
    analogWrite(E5,0);

  delay(5000);

  i=0;
break;

case 6:
  Serial.println("Pulgar seleccionado");

  Serial.println("Cerrando");

    digitalWrite(I1, HIGH);
    digitalWrite(I2, LOW);

    analogWrite(E5,255);

  delay(5000);

  Serial.println("Abriendo");

    digitalWrite(I1, LOW);
    digitalWrite(I2, HIGH);

  delay(2000);

    digitalWrite(I1, LOW);
    digitalWrite(I2, LOW);

    analogWrite(E5,0);

  delay(5000);

  i=0;
break;

case 7:
  Serial.println("Indice seleccionado");

  Serial.println("Cerrando");

    digitalWrite(I1, HIGH);
    digitalWrite(I2, LOW);

    analogWrite(E4,255);

  delay(5000);

  Serial.println("Abriendo");

    digitalWrite(I1, LOW);
```

```
    digitalWrite(I2, HIGH);

  delay(2000);

    digitalWrite(I1, LOW);
    digitalWrite(I2, LOW);

    analogWrite(E4,0);

  delay(5000);

  i=0;
break;

case 8:
  Serial.println("Corazon seleccionado");

  Serial.println("Cerrando");

    digitalWrite(I1, HIGH);
    digitalWrite(I2, LOW);

    analogWrite(E3,255);

  delay(5000);

  Serial.println("Abriendo");

    digitalWrite(I1, LOW);
    digitalWrite(I2, HIGH);

  delay(2000);

    digitalWrite(I1, LOW);
    digitalWrite(I2, LOW);

    analogWrite(E3,0);

  delay(5000);

  i=0;
break;

case 9:
  Serial.println("Anular seleccionado");

  Serial.println("Cerrando");

    digitalWrite(I1, HIGH);
    digitalWrite(I2, LOW);

    analogWrite(E2,255);

  delay(5000);

  Serial.println("Abriendo");

    digitalWrite(I1, LOW);
    digitalWrite(I2, HIGH);

  delay(2000);
```

```
    digitalWrite(I1, LOW);
    digitalWrite(I2, LOW);

    analogWrite(E2,0);

  delay(5000);

  i=0;
break;

case 10:
  Serial.println("Menique seleccionado");

  Serial.println("Cerrando");

    digitalWrite(I1, HIGH);
    digitalWrite(I2, LOW);

    analogWrite(E1,255);

  delay(5000);

  Serial.println("Abriendo");

    digitalWrite(I1, LOW);
    digitalWrite(I2, HIGH);

  delay(2000);

    digitalWrite(I1, LOW);
    digitalWrite(I2, LOW);

    analogWrite(E1,0);

  delay(5000);

  i=0;
break;

case 11:
  Serial.println("Okay seleccionado");

  Serial.println("Cerrando");

    digitalWrite(I1, HIGH);
    digitalWrite(I2, LOW);

    analogWrite(E1,255);
    analogWrite(E2,255);
    analogWrite(E3,255);
    analogWrite(E4,255);
    srv1.write(110);

  delay(5000);

  Serial.println("Abriendo");

    digitalWrite(I1, LOW);
    digitalWrite(I2, HIGH);
```

```
    delay(2000);

      digitalWrite(I1, LOW);
      digitalWrite(I2, LOW);

      analogWrite(E1,0);
      analogWrite(E2,0);
      analogWrite(E3,0);
      analogWrite(E4,0);
      srv1.write(25);

    delay(5000);

    i=0;
  break;

  case 12:
    Serial.println("Servo");
    delay(1000);

      srv1.write(110);

    delay(1500);

      srv1.write(25);

    delay(2000);

    i=0;
  break;

case 13:
    Serial.println("Guardar");

    delay(2000);

      srv1.write(110);

    delay(3000);

      digitalWrite(I1, HIGH);
      digitalWrite(I2, LOW);

      analogWrite(E5,255);

    delay(3000);

      analogWrite(E1,0);

      digitalWrite(I1, LOW);
      digitalWrite(I2, LOW);

    delay(5000);
    i=0;
  break;

  case 99:
    i=0;
    srv1.write(25);

    digitalWrite(I1, LOW);
```

```
            digitalWrite(I2, HIGH);

            analogWrite(E5,255);

            delay(3000);

            analogWrite(E1,0);

            digitalWrite(I1, LOW);
            digitalWrite(I2, LOW);
          break;

         case 999:
            i=0;
          break;

          default:
           Serial.println("Numero no valido");
           delay(5000);
           i=0;
          break;
       }
    }
}
```

ANEXO 10: CÓDIGO DE ARDUINO DE LA APLICACIÓN FINAL DE CONTROL DE LA PRÓTESIS.

```
#include <Servo.h>  // Incluimos la librería dedicada a Servo-motores

Servo srv1;  //Declaramos la variable del servomotor

#define E1 13  //Declaramos las variables de encendido de cada uno de
los motores
#define E2 5
#define E3 9
#define E4 10
#define E5 11

#define I1 2  //Declaramos las variables de control de los motores
#define I2 4

int modo;

int canal1 = 0;  // Declaramos dos variables asociadas a cada uno de
los dos canales de entrada
int canal2 = 0;

void setup() {
    Serial.begin(9600);  //Establecemos conexion con el puerto serie

    srv1.attach(3);  //Establecemos el servo en su posición inicial
    srv1.write(110);

    pinMode(E1, OUTPUT);  //Declaramos todas las variables de los
motores DC como salidas
    pinMode(E2, OUTPUT);
    pinMode(E3, OUTPUT);
    pinMode(E4, OUTPUT);
    pinMode(E5, OUTPUT);
    pinMode(I1, OUTPUT);
    pinMode(I2, OUTPUT);
}

void loop() {
    canal1=analogRead(A0);  //Asociamos las entradas analógicas A0 y
A1 provinientes de cada uno de los canales, con las variables canal1 y
canal2
    canal2=analogRead(A1);

    canal1=constrain(canal1,0,820);  //Restringimos el rango de
valores de entrada a 0-820 en ambas variables
    canal2=constrain(canal2,0,820);

    //En el siguiente Switch se seleccionarán los distintos
movimientos en función de los distintos niveles de esfuerzo muscular
realizado
    //Dentro de cada estado se ha establecido un if para que solo se
ejecuten las acciones asociadas a cada estado si se permanece en él un
```

```
    //mínimo de 700ms
    switch (canal1){
      case 0 ... 180:  //Nivel bajo canal1

        switch (canal2){
        case 0 ... 180:  //Nivel bajo-bajo

          delay(700);  //En el estado de reposo no se hace nada
          if(canal1>0 && canal1<70 && canal2>0 && canal2<70){
            Serial.print("Estado de reposo");
            delay(1000);
          }

        break;

        case 181 ... 500:  //Nivel bajo-medio

          delay(700);  //En este estado iremos probando todos los
motores, incluido el servo
          if(canal1>0 && canal1<70 && canal2>71 && canal2<170){
            Serial.println("Prueba de todos los dedos seleccionado");

            delay(2000); //En primer lugar iremos abriendo y cerrando
todos los dedos
            Serial.println("Cerrando 1");

              //Establecemos los controles para que se cierre el dedo
              digitalWrite(I1, HIGH);
              digitalWrite(I2, LOW);

              //Activamos el motor que queramos
              analogWrite(E1,255);

            delay(2000);
            Serial.println("Abriendo 1");

              //Invertimos los controles para que así el dedo se abra
              digitalWrite(I1, LOW);
              digitalWrite(I2, HIGH);

            delay(2000);

              //Tras acabar de abrir el dedo, se quita la corriente
del motor
              analogWrite(E1,0);

            Serial.println("Cerrando 2");

              digitalWrite(I1, HIGH);
              digitalWrite(I2, LOW);

              analogWrite(E2,255);

            delay(2000);
            Serial.println("Abriendo 2");

              digitalWrite(I1, LOW);
              digitalWrite(I2, HIGH);

            delay(2000);
```

```
  analogWrite(E2,0);

Serial.println("Cerrando 3");

  digitalWrite(I1, HIGH);
  digitalWrite(I2, LOW);

  analogWrite(E3,255);

delay(2000);
Serial.println("Abriendo 3");

  digitalWrite(I1, LOW);
  digitalWrite(I2, HIGH);

delay(2000);

  analogWrite(E3,0);

Serial.println("Cerrando 4");

  digitalWrite(I1, HIGH);
  digitalWrite(I2, LOW);

  analogWrite(E4,255);

delay(2000);
Serial.println("Abriendo 4");

  digitalWrite(I1, LOW);
  digitalWrite(I2, HIGH);

delay(2000);

  analogWrite(E4,0);

Serial.println("Cerrando 5");

  digitalWrite(I1, HIGH);
  digitalWrite(I2, LOW);

  analogWrite(E5,255);

delay(2000);
Serial.println("Abriendo 5");

  digitalWrite(I1, LOW);
  digitalWrite(I2, HIGH);

delay(2000);

  digitalWrite(I1, LOW);
  digitalWrite(I2, LOW);
  analogWrite(E5,0);

Serial.println("Servo");

        //Para manejar el servo es aún más facil pues solo tenemos
que introducir
        //los grados en los que queremos que se sitúe en la
sentencia srv1.write()
```

```
            delay(2000);

              srv1.write(110);

            delay(1500);

              srv1.write(25);

            delay(2000);

            }

        break;

        case 501 ... 820:  //Nivel bajo-alto

          delay(700);  //Agarre suave
          if(canal1>0 && canal1<70 && canal2>171 && canal2<250){
            Serial.println("Agarre suave seleccionado");

            //Para este estado cerraremos solo los dedos índice y
pulgar, simultáneamente
            Serial.println("Cerrando");

              digitalWrite(I1, HIGH);
              digitalWrite(I2, LOW);

              analogWrite(E4,255);
              analogWrite(E5,255);

            delay(2000);

              digitalWrite(I1, LOW);
              digitalWrite(I2, LOW);

              analogWrite(E4,0);
              analogWrite(E5,0);

            delay(2000);
          }

        break;
        }
      break;


      case 181 ... 500:  //Nivel medio canal1
        switch (canal2){
        case 0 ... 180:  //Nivel medio-bajo

          delay(700);  //Agarre fuerte
          if(canal1>71 && canal1<170 && canal2>0 && canal2<70){
            Serial.println("Agarre fuerte seleccionado");

            //En este estado cerraremos todos los dedos
            Serial.println("Cerrando");

              digitalWrite(I1, HIGH);
              digitalWrite(I2, LOW);

              analogWrite(E1,255);
```

```
            analogWrite(E2,255);
            analogWrite(E3,255);
            analogWrite(E4,255);
            analogWrite(E5,255);

          delay(2000);

            digitalWrite(I1, LOW);
            digitalWrite(I2, LOW);

            analogWrite(E1,0);
            analogWrite(E2,0);
            analogWrite(E3,0);
            analogWrite(E4,0);
            analogWrite(E5,0);

          delay(2000);
        }

      break;

      case 181 ... 500:  //Nivel medio-medio

        delay(700);
        if(canal1>71 && canal1<170 && canal2>71 && canal2<170){
          Serial.println("Vete a la mierda seleccionado");

          //En este estado cerraremos todos los dedos excepto el
corazón y situaremos el
          //pulgar en perpendicular al resto de dedos en vez de
enfrente de ellos
          Serial.println("Cerrando");

            digitalWrite(I1, HIGH);
            digitalWrite(I2, LOW);

            analogWrite(E1,255);
            analogWrite(E2,255);
            analogWrite(E4,255);
            analogWrite(E5,255);
            srv1.write(110);

          delay(5000);

          Serial.println("Abriendo");

            digitalWrite(I1, LOW);
            digitalWrite(I2, HIGH);

          delay(2000);

            digitalWrite(I1, LOW);
            digitalWrite(I2, LOW);

            analogWrite(E1,0);
            analogWrite(E2,0);
            analogWrite(E4,0);
            analogWrite(E5,0);
            srv1.write(25);

          delay(2000);
```

```
          }

        break;

        case 501 ... 820:  //Nivel medio-alto

          delay(700);//Hacer los cuernos
          if(canal1>71 && canal1<170 && canal2>171 && canal2<250){
            Serial.println("Cuernos seleccionado");

            //Para este estado cerraremos solo los dedos pulgar,
anular y corazon
            Serial.println("Cerrando");

              digitalWrite(I1, HIGH);
              digitalWrite(I2, LOW);

              analogWrite(E2,255);
              analogWrite(E3,255);
              analogWrite(E5,255);

            delay(5000);

            Serial.println("Abriendo");

              digitalWrite(I1, LOW);
              digitalWrite(I2, HIGH);

            delay(2000);

              digitalWrite(I1, LOW);
              digitalWrite(I2, LOW);

              analogWrite(E2,0);
              analogWrite(E3,0);
              analogWrite(E5,0);

            delay(2000);
            }

        break;
        }
      break;


      case 501 ... 820:  //Nivel alto canal1
        switch (canal2){
        case 0 ... 180:  //Nivel alto-bajo

        delay(700); //Signo de Ok
        if(canal1>171 && canal1<250 && canal2>0 && canal2<70){
          Serial.println("Okay seleccionado");

            //En este estado dejaremos sin cerrar solo el pulgar, el
cual giraremos
            //para ponerlo en peroendicular al resto de dedos
            Serial.println("Cerrando");

            digitalWrite(I1, HIGH);
            digitalWrite(I2, LOW);
```

```
        analogWrite(E1,255);
        analogWrite(E2,255);
        analogWrite(E3,255);
        analogWrite(E4,255);
        srv1.write(110);

      delay(5000);

      Serial.println("Abriendo");

        digitalWrite(I1, LOW);
        digitalWrite(I2, HIGH);

      delay(2000);

        digitalWrite(I1, LOW);
        digitalWrite(I2, LOW);

        analogWrite(E1,0);
        analogWrite(E2,0);
        analogWrite(E3,0);
        analogWrite(E4,0);
        srv1.write(25);

      delay(2000);
    }

    break;

    case 181 ... 500:  //Nivel alto-medio

      delay(700);  //Prueba del servo
      if(canal1>171 && canal1<250 && canal2>71 && canal2<170){
        Serial.println("Prueba del servo seleccionada");
        delay(1000);

          //En este estado probaremos el giro del servomotor
          srv1.write(110);

        delay(1500);

          srv1.write(25);

        delay(2000);
      }

    break;

    case 501 ... 820:  //Nivel alto-alto

      delay(700); //Abrir la mano
      if(canal1>171 && canal1<250 && canal2>171 && canal2<250){
        Serial.println("Abrir la mano seleccionado");

        //En este estado cerraremos todos los dedos de la mano a
la vez
        Serial.println("Abriendo");

          digitalWrite(I1, LOW);
          digitalWrite(I2, HIGH);
```

```
            analogWrite(E1,255);
            analogWrite(E2,255);
            analogWrite(E3,255);
            analogWrite(E4,255);
            analogWrite(E5,255);

          delay(2000);

            digitalWrite(I1, LOW);
            digitalWrite(I2, LOW);

            analogWrite(E1,0);
            analogWrite(E2,0);
            analogWrite(E3,0);
            analogWrite(E4,0);
            analogWrite(E5,0);

          delay(2000);
        }

      break;
      }
    break;

    default: break;
    }

}
```

ANEXO 11: CÓDIGO DE ARDUINO PARA LA PRUEBA DEL MANDO MIOELÉCTRICO.

```
int canal1 = 0;             // Declaramos dos variables asociados a
cada una de los dos canales de entrada
int canal2 = 0;

void setup() {            // Cada vez que pulsemos reset, se
inicializará la rutina:

  Serial.begin(9600);

}


void loop() {

  canal1=analogRead(A0);               // Asociamos las entradas
analógicas A0 y A1 provinientes de cada uno de los canales, con las
variables val1 y val2
  canal2=analogRead(A1);

  canal1=constrain(canal1,0,815);        // Restringimos el rango de
valores de entrada de 0-650 en ambas variables
  canal2=constrain(canal2,0,815);

  switch (canal1){
  case 0 ... 180:                                    // NIVEL BAJO
CANAL1

    switch (canal2){
    case 0 ... 180:                                  // NIVEL BAJO-BAJO

      delay(300);
      if(canal1>0 && canal1<180 && canal2>0 && canal2<180){
        Serial.print("Nivel bajo-bajo: ");
        Serial.print(canal1);
        Serial.print(" y ");
        Serial.println(canal2);
        delay(2000);
      }

    break;

    case 181 ... 500:                        // NIVEL BAJO-MEDIO

      delay(300);
      if(canal1>0 && canal1<180 && canal2>181 && canal2<500){
        Serial.print("Nivel bajo-medio: ");
        Serial.print(canal1);
        Serial.print(" y ");
        Serial.println(canal2);
        delay(2000);
      }
```

```
  break;

  case 501 ... 820:                        // NIVEL BAJO-ALTO

    delay(300);
    if(canal1>0 && canal1<180 && canal2>501 && canal2<820){
      Serial.print("Nivel bajo-alto: ");
      Serial.print(canal1);
      Serial.print(" y ");
      Serial.println(canal2);
      delay(2000);
    }

  break;
  }
break;


case 181 ... 500:                         // NIVEL MEDIO CANAL1
  switch (canal2){
  case 0 ... 180:                        // NIVEL MEDIO-BAJO

    delay(300);
    if(canal1>181 && canal1<500 && canal2>0 && canal2<180){
      Serial.print("Nivel medio-bajo: ");
      Serial.print(canal1);
      Serial.print(" y ");
      Serial.println(canal2);
      delay(2000);
    }

  break;

  case 181 ... 500:                        // NIVEL MEDIO-MEDIO

    delay(300);
    if(canal1>181 && canal1<500 && canal2>181 && canal2<500){
      Serial.print("Nivel medio-medio: ");
      Serial.print(canal1);
      Serial.print(" y ");
      Serial.println(canal2);
      delay(2000);
    }

  break;

  case 501 ... 820:                        // NIVEL MEDIO-ALTO

    delay(300);
    if(canal1>181 && canal1<500 && canal2>501 && canal2<820){
      Serial.print("Nivel medio-alto: ");
      Serial.print(canal1);
      Serial.print(" y ");
      Serial.println(canal2);
      delay(2000);
    }

  break;
  }
break;
```

```
  case 501 ... 820:                       // NIVEL ALTO CANAL1
    switch (canal2){
    case 0 ... 180:                       // NIVEL ALTO-BAJO

      delay(300);
      if(canal1>501 && canal1<820 && canal2>0 && canal2<180){
        Serial.print("Nivel alto-bajo: ");
        Serial.print(canal1);
        Serial.print(" y ");
        Serial.println(canal2);
        delay(2000);
      }

    break;

    case 181 ... 500:                       // NIVEL ALTO-MEDIO

      delay(300);
      if(canal1>501 && canal1<820 && canal2>181 && canal2<500){
        Serial.print("Nivel alto-medio: ");
        Serial.print(canal1);
        Serial.print(" y ");
        Serial.println(canal2);
        delay(2000);
      }

    break;

    case 501 ... 820:                       // NIVEL ALTO-ALTO

      delay(300);
      if(canal1>501 && canal1<820 && canal2>501 && canal2<820){
        Serial.print("Nivel alto-alto: ");
        Serial.print(canal1);
        Serial.print(" y ");
        Serial.println(canal2);
        delay(2000);
      }

    break;
    }
  break;

  default: break;
  }
}
```

ANEXO 12: ESQUEMA Y FOTO DE LA PLACA DE CONTROL DE LOS MOTORES.

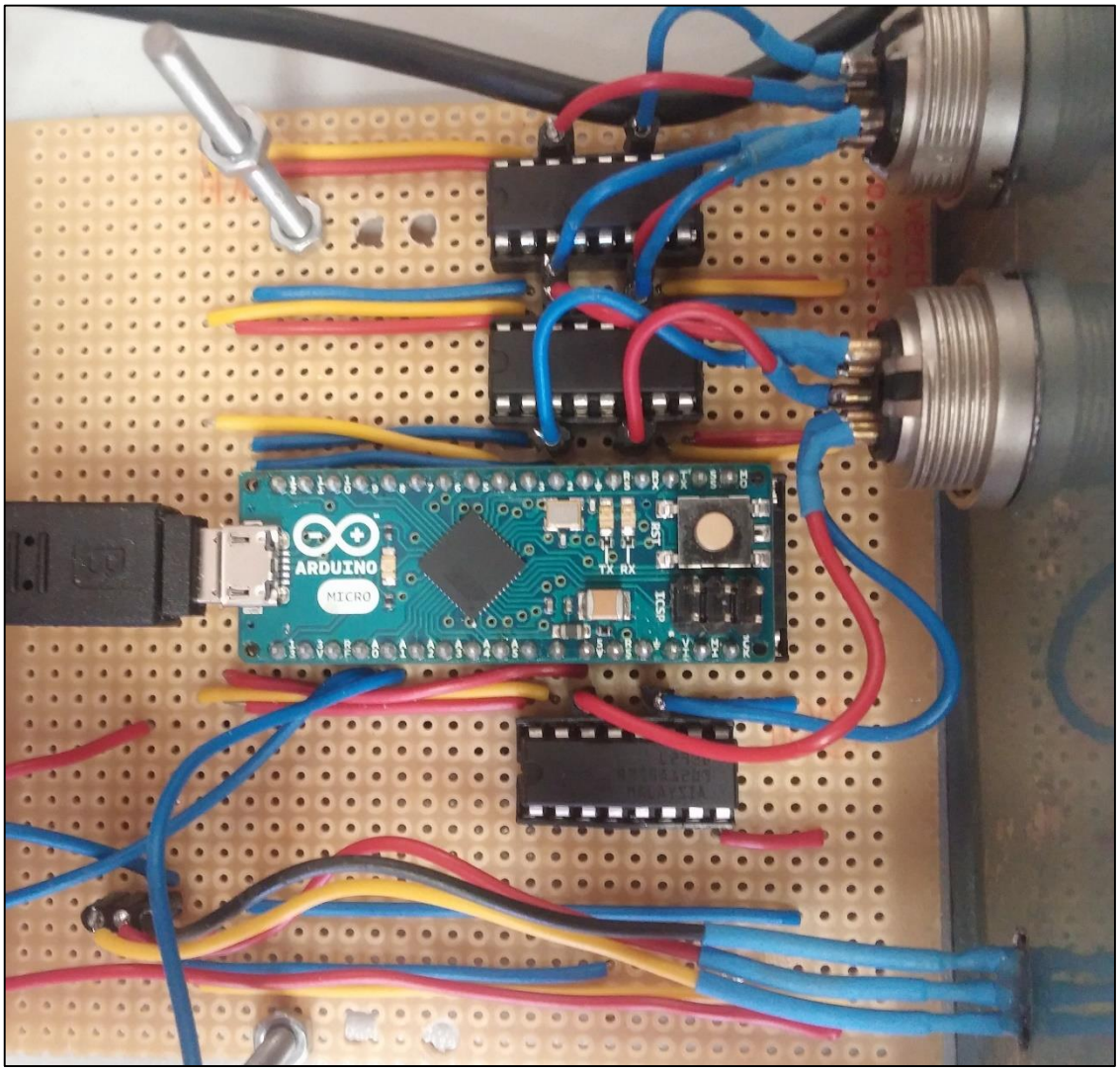


***Imagen 122**. Foto de la placa de control de los motores.*

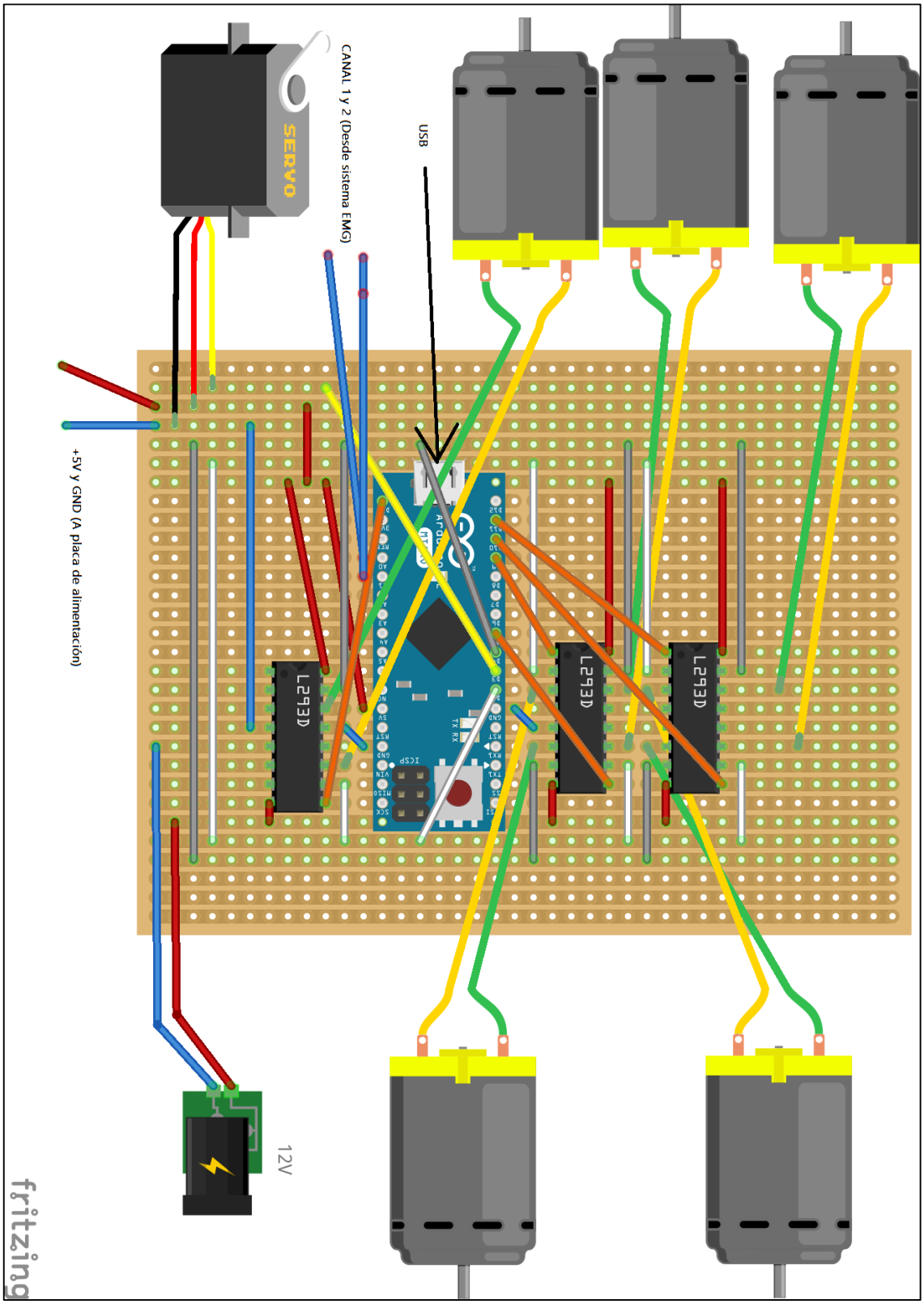
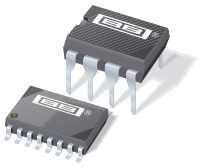


***Imagen 123**. Esquema de conexiones de la placa de control de los motores.*

ANEXO 13: HOJAS DE ESPECIFICACIONES DEL INTEGRADO INA114.

BURR - BROWN®



INA114

Precision INSTRUMENTATION AMPLIFIER

FEATURES

- LOW OFFSET VOLTAGE: 50 V max
- LOW DRIFT: 0.25 V/ C max
- LOW INPUT BIAS CURRENT: 2nA max
- HIGH COMMON-MODE REJECTION: 115dB min
- INPUT OVER-VOLTAGE PROTECTION: 40V
- WIDE SUPPLY RANGE: 2.25 to 18V
- LOW QUIESCENT CURRENT: 3mA max
- 8-PIN PLASTIC AND SOL-16

APPLICATIONS

- BRIDGE AMPLIFIER
- THERMOCOUPLE AMPLIFIER
- RTD SENSOR AMPLIFIER
- MEDICAL INSTRUMENTATION
- DATA ACQUISITION

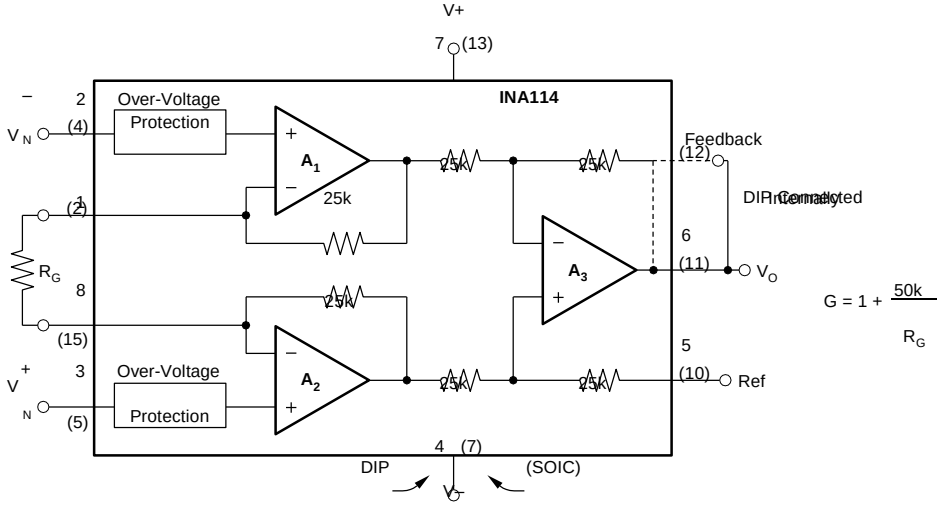
DESCRIPTION

The INA114 is a low cost, general purpose instrumentation amplifier offering excellent accuracy. Its versatile 3-op amp design and small size make it ideal for a wide range of applications.

A single external resistor sets any gain from 1 to 10,000. Internal input protection can withstand up to 40V without damage.

The INA114 is laser trimmed for very low offset voltage (50 V), drift (0.25 V/ C) and high common-mode rejection (115dB at G = 1000). It operates with power supplies as low as 2.25V, allowing use in battery operated and single 5V supply systems. Quiescent current is 3mA maximum.

The INA114 is available in 8-pin plastic and SOL-16 surface-mount packages. Both are specified for the –40 C to +85 C temperature range.



International Airport Industrial Park • Mailing Address: PO Box 11400, Tucson, AZ 85734 • Street Address: 6730 S. Tucson Blvd., Tucson, AZ 85706 • Tel: (520) 746-1111 • Twx: 910-952-1111
Internet: http://www.burr-brown.com/ • FAXLine: (800) 548-6133 (US/Canada Only) • Cable: BBRCORP • Telex: 066-6491 • FAX: (520) 889-1510 • Immediate Product Info: (800) 548-6132

PDS-1142D

Printed in U.S.A. March, 1998

SBOS014

SPECIFICATIONS

ELECTRICAL

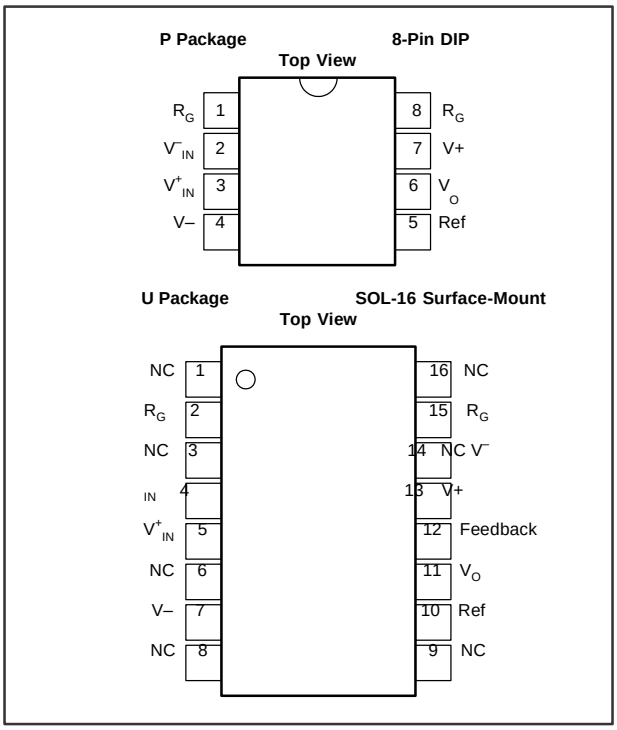
At T_A = +25 C, V_S = 15V, R_L = 2k unless otherwise noted.

PARAMETER	CONDITIONS	INA114BP, BU			INA114AP, AU			UNITS
		MIN	TYP	MAX	MIN	TYP	MAX	
INPUT								
Offset Voltage, RTI								
Initial	T_A = +25 C		10 + 20/G	50 + 100/G		25 + 30/G	125 + 500/G	V
vs Temperature	T_A = T_{MIN} to T_{MAX}		0.1 + 0.5/G	0.25 + 5/G		0.25 + 5/G	1 + 10/G	V/ C
vs Power Supply	V_S = 2.25V to 18V		0.5 + 2/G	3 + 10/G		*	*	V/V
Long-Term Stability			0.2 + 0.5/G			*		V/mo
Impedance, Differential			10^{10} || 6			*		|| pF
Common-Mode			10^{10} || 6			*		|| pF
Input Common-Mode Range		11	13.5		*	*		V
Safe Input Voltage				40			*	V
Common-Mode Rejection	V_{CM} = 10V, R_S = 1k							
	G = 1	80	96		75	90		dB
	G = 10	96	115		90	106		dB
	G = 100	110	120		106	110		dB
	G = 1000	115	120		106	110		dB
BIAS CURRENT			0.5	2		*	5	nA
vs Temperature			8			*		pA/ C
OFFSET CURRENT			0.5	2		*	5	nA
vs Temperature			8			*		pA/ C
NOISE VOLTAGE, RTI	G = 1000, R_S = 0							
f = 10Hz			15			*		nV/ $\sqrt{Hz}$
f = 100Hz			11			*		nV/ $\sqrt{Hz}$
f = 1kHz			11			*		nV/ Hz
f_B = 0.1Hz to 10Hz			0.4			*		Vp-p
Noise Current								
f=10Hz			0.4			*		pA/ $\sqrt{Hz}$
f=1kHz			0.2			*		pA/ Hz
f_B = 0.1Hz to 10Hz			18			*		pAp-p
GAIN								
Gain Equation			1 + (50k /R_G)			*		V/V
Range of Gain		1		10000	*		*	V/V
Gain Error			0.01	0.05		*	*	%
	G = 1		0.02	0.4		*	0.5	%
	G = 10		0.05	0.5		*	0.7	%
	G = 100		0.5	1		*	2	%
	G = 1000		2	10		*	10	ppm/ C
Gain vs Temperature	G = 1		25	100		*	*	ppm/ C
50k Resistance[1]			0.0001	0.001		*	0.002	% of FSR
Nonlinearity	G = 1		0.0005	0.002		*	0.004	% of FSR
	G = 10		0.0005	0.002		*	0.004	% of FSR
	G = 100		0.002	0.01		*	0.02	% of FSR
	G = 1000							
OUTPUT								
Voltage	I_O = 5mA, T_{MIN} to T_{MAX}	13.5	13.7		*	*		V V
	V_S = 11.4V, R_L = 2k	10	10.5		*	*		V
	V_S = 2.25V, R_L = 2k	1	1.5		*	*		mA
Load Capacitance Stability			1000			*		pF
Short Circuit Current			+20/–15			*		
FREQUENCY RESPONSE								
Bandwidth, –3dB	G = 1		1			*		MHz
	G = 10		100			*		kHz
	G = 100		10			*		kHz
	G = 1000		1			*		kHz
Slew Rate	V_O = 10V, G = 10	0.3	0.6		*	*		V/ s
Settling Time, 0.01%	G = 1		18			*		s
	G = 10		20			*		s
	G = 100		120			*		s
	G = 1000		1100			*		s
Overload Recovery	50% Overdrive		20			*		s
POWER SUPPLY								
Voltage Range		2.25	15	18	*	*	*	V
Current	V_{IN} = 0V		2.2	3		*	*	mA
TEMPERATURE RANGE								
Specification		–40		85	*		*	C C
Operating		–40		125	*		*	C/W
$_{JA}$			80			*		

* Specification same as INA114BP/BU.

NOTE: (1) Temperature coefficient of the "50k " term in the gain equation.

PIN CONFIGURATIONS



ABSOLUTE MAXIMUM RATINGS(1)

Supply Voltage	18V
Input Voltage Range	40V
Output Short-Circuit (to ground)	Continuous
Operating Temperature	–40 C to +125 C
Storage Temperature	–40 C to +125 C
Junction Temperature	+150 C
Lead Temperature (soldering, 10s)	+300 C

NOTE: (1) Stresses above these ratings may cause permanent damage.

ELECTROSTATIC DISCHARGE SENSITIVITY

This integrated circuit can be damaged by ESD. Burr-Brown recommends that all integrated circuits be handled with appropriate precautions. Failure to observe proper handling and installation procedures can cause damage.

ESD damage can range from subtle performance degradation to complete device failure. Precision integrated circuits may be more susceptible to damage because very small parametric changes could cause the device not to meet its published specifications.

PACKAGE/ORDERING INFORMATION

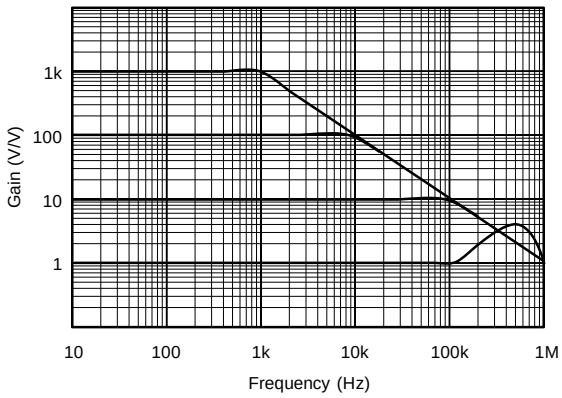
PRODUCT	PACKAGE	PACKAGE DRAWING NUMBER(1)	TEMPERATURE RANGE
INA114AP	8-Pin Plastic DIP	006	–40 C to +85 C
INA114BP	8-Pin Plastic DIP SOL-	006	–40 C to +85 C
INA114AU	16 Surface-Mount	211	–40 C to +85 C
INA114BU	SOL-16 Surface-Mount	211	–40 C to +85 C

NOTE: (1) For detailed drawing and dimension table, please see end of data sheet, or Appendix C of Burr-Brown IC Data Book.

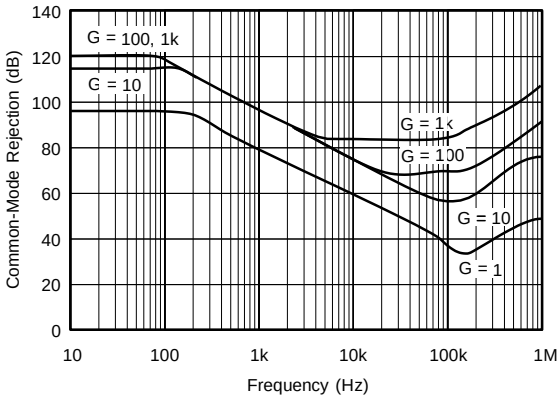
TYPICAL PERFORMANCE CURVES

At T_A = +25 C, V_S = 15V, unless otherwise noted.

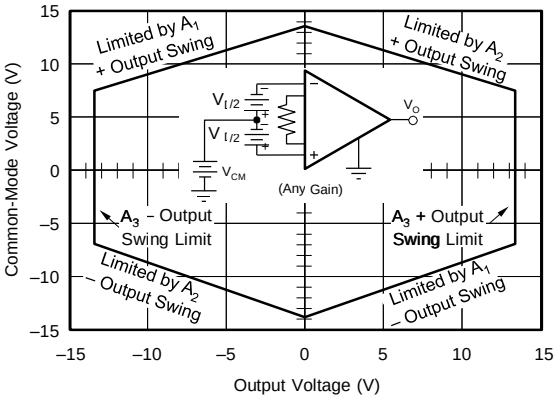
GAIN vs FREQUENCY



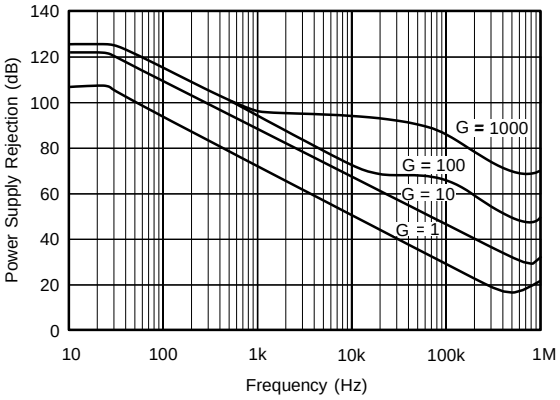
COMMON-MODE REJECTION vs FREQUENCY



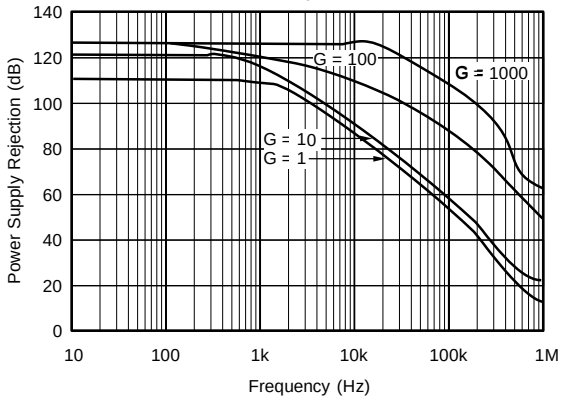
INPUT COMMON-MODE VOLTAGE RANGE vs OUTPUT VOLTAGE



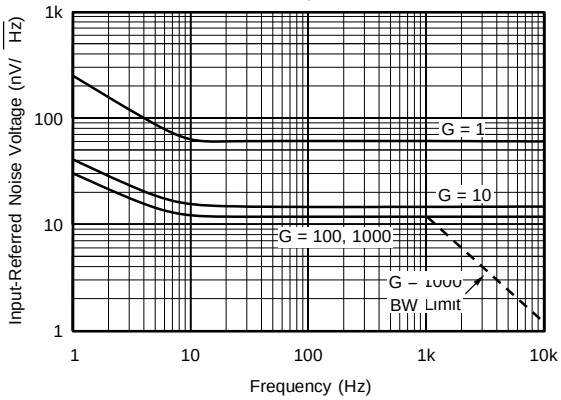
POSITIVE POWER SUPPLY REJECTION vs FREQUENCY



NEGATIVE POWER SUPPLY REJECTION vs FREQUENCY



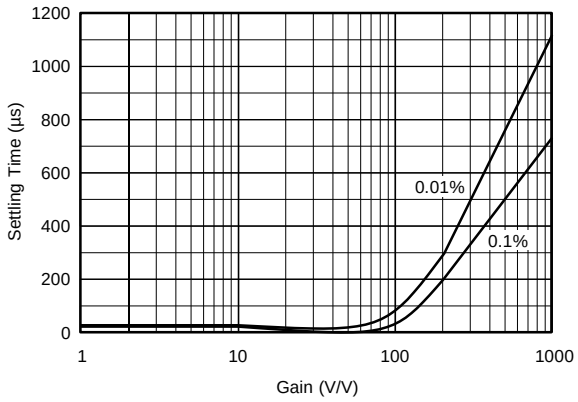
INPUT-REFERRED NOISE VOLTAGE vs FREQUENCY



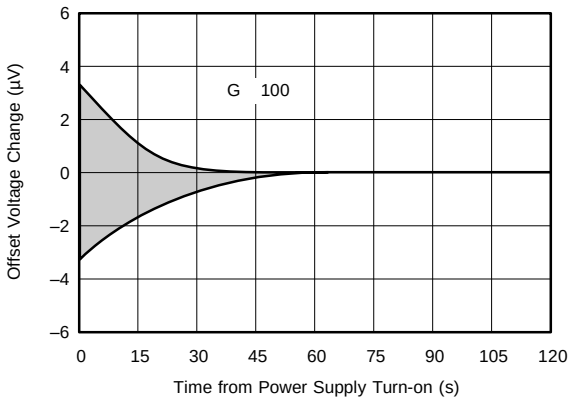
TYPICAL PERFORMANCE CURVES (CONT)

At T_A = +25 C, V_S = 15V, unless otherwise noted.

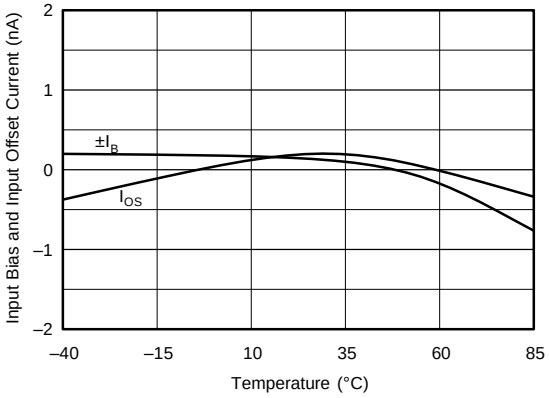
SETTLING TIME vs GAIN



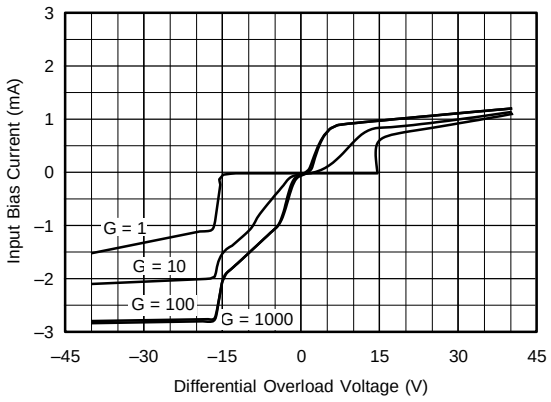
OFFSET VOLTAGE WARM-UP vs TIME



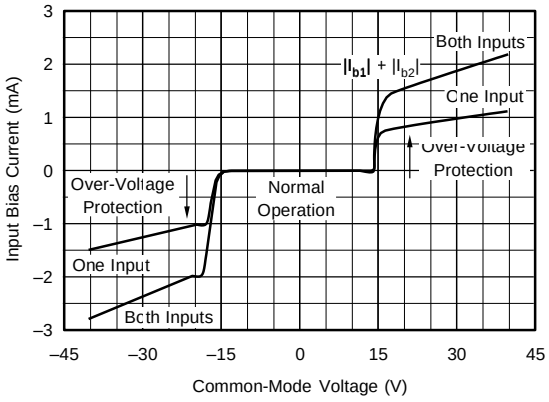
INPUT BIAS AND INPUT OFFSET CURRENT vs TEMPERATURE



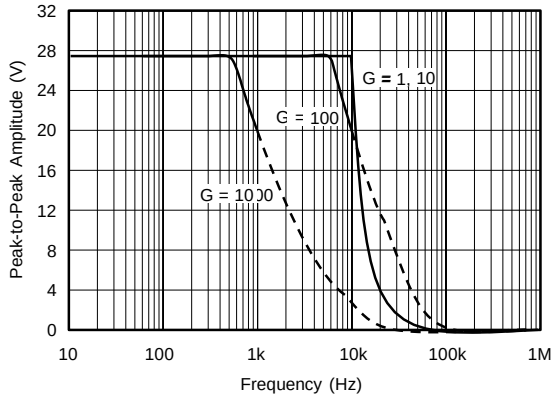
INPUT BIAS CURRENT vs DIFFERENTIAL INPUT VOLTAGE



INPUT BIAS CURRENT vs COMMON-MODE INPUT VOLTAGE



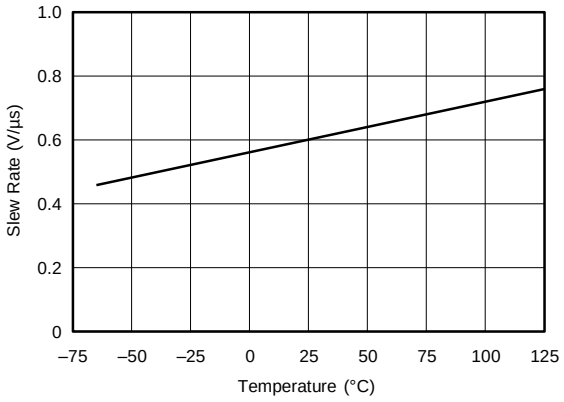
MAXIMUM OUTPUT SWING vs FREQUENCY



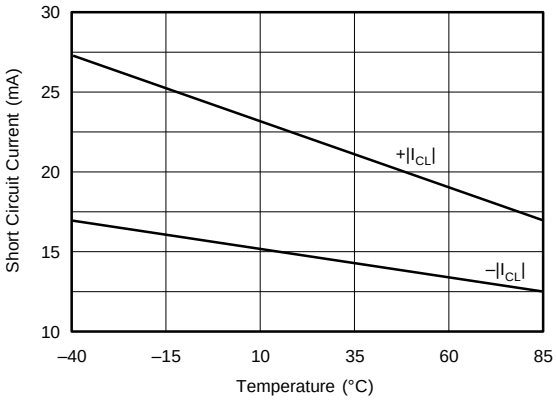
TYPICAL PERFORMANCE CURVES (CONT)

At T_A = +25 C, V_S = 15V, unless otherwise noted.

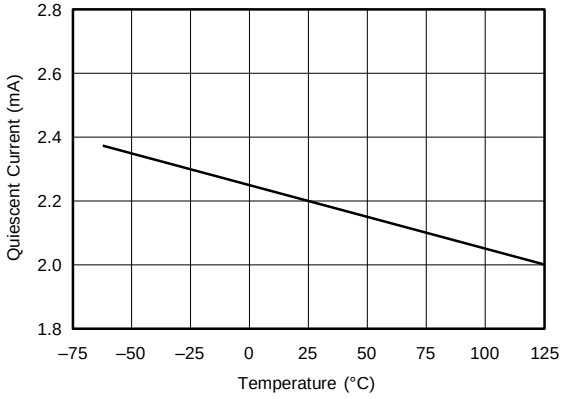
SLEW RATE vs TEMPERATURE



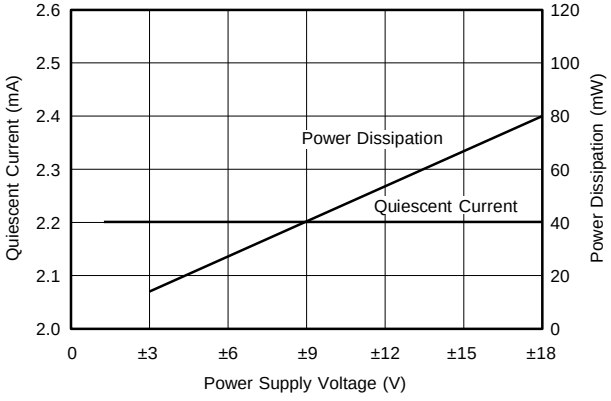
OUTPUT CURRENT LIMIT vs TEMPERATURE



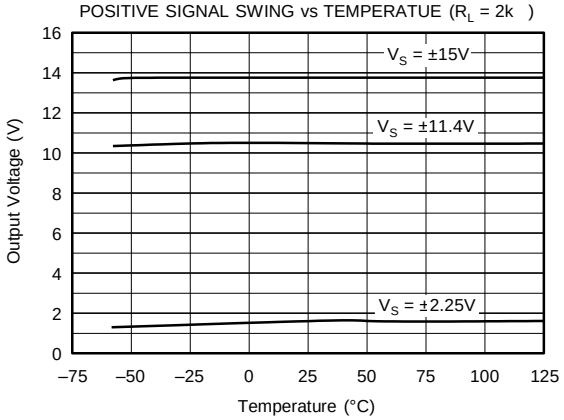
QUIESCENT CURRENT vs TEMPERATURE



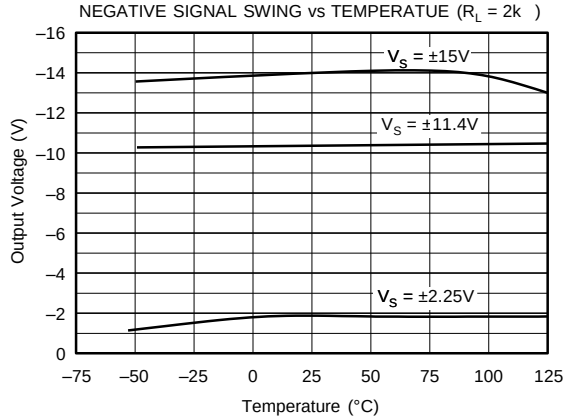
QUIESCENT CURRENT AND POWER DISSIPATION vs POWER SUPPLY VOLTAGE



POSITIVE SIGNAL SWING vs TEMPERATUE (R_L = 2k)



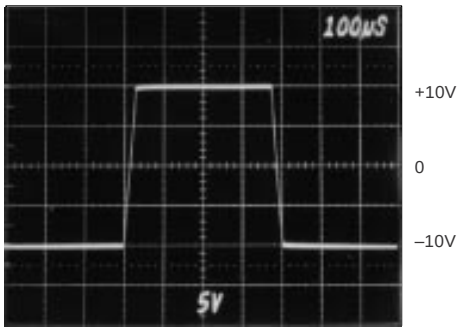
NEGATIVE SIGNAL SWING vs TEMPERATUE (R_L = 2k)



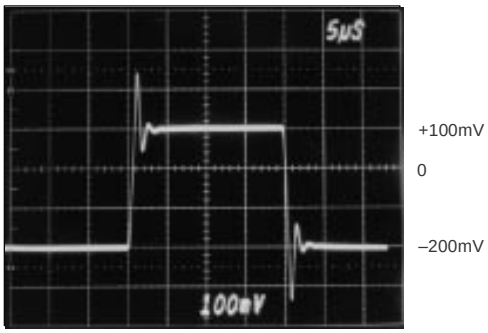
TYPICAL PERFORMANCE CURVES (CONT)

At T_A = +25 C, V_S = 15V, unless otherwise noted.

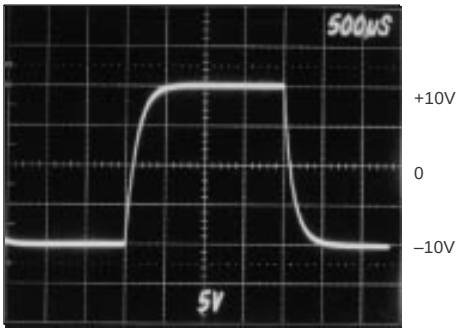
LARGE SIGNAL RESPONSE, G = 1



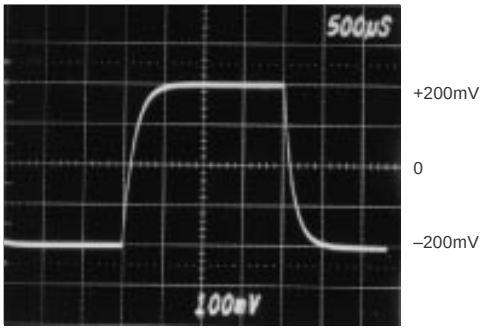
SMALL SIGNAL RESPONSE, G = 1



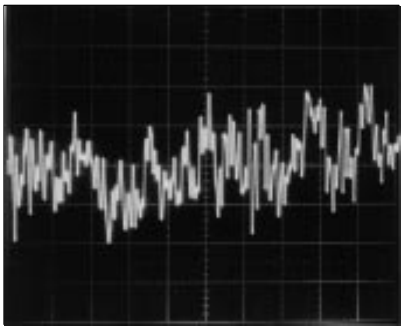
LARGE SIGNAL RESPONSE, G = 1000



SMALL SIGNAL RESPONSE, G = 1000



INPUT-REFERRED NOISE, 0.1 to 10Hz



APPLICATION INFORMATION

Figure 1 shows the basic connections required for operation of the INA114. Applications with noisy or high impedance power supplies may require decoupling capacitors close to the device pins as shown.

The output is referred to the output reference (Ref) terminal which is normally grounded. This must be a low-impedance connection to assure good common-mode rejection. A resistance of 5 in series with the Ref pin will cause a typical device to degrade to approximately 80dB CMR (G = 1).

SETTING THE GAIN

Gain of the INA114 is set by connecting a single external resistor, R_G:

$$G = 1 + \frac{50\ k\Omega}{R_G} \qquad (1)$$

Commonly used gains and resistor values are shown in Figure 1.

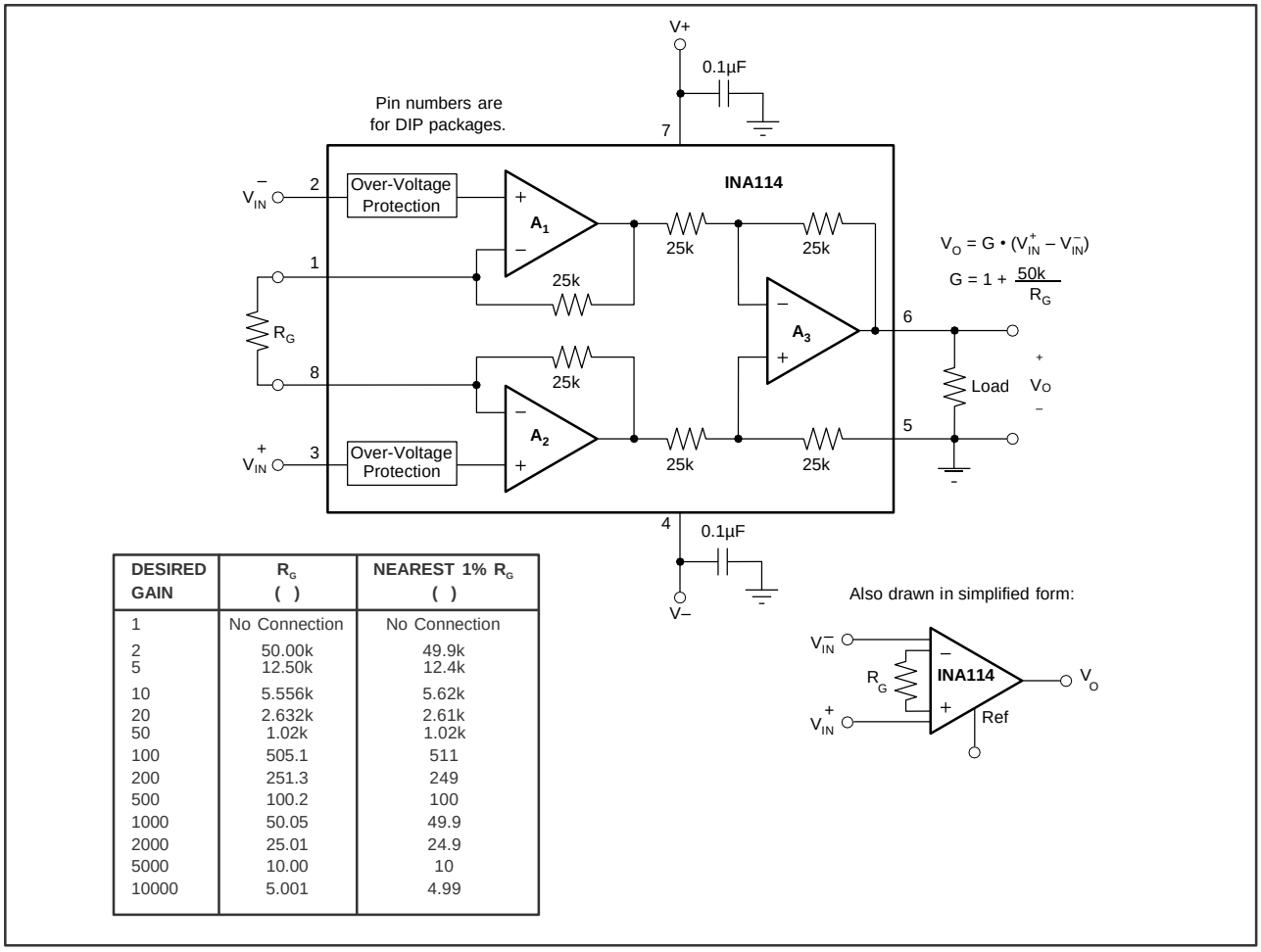
The 50k term in equation (1) comes from the sum of the two internal feedback resistors. These are on-chip metal film resistors which are laser trimmed to accurate absolute values. The accuracy and temperature coefficient of these resistors are included in the gain accuracy and drift specifications of the INA114.

The stability and temperature drift of the external gain setting resistor, R_G, also affects gain. R_G's contribution to gain accuracy and drift can be directly inferred from the gain equation (1). Low resistor values required for high gain can make wiring resistance important. Sockets add to the wiring resistance which will contribute additional gain error (possibly an unstable gain error) in gains of approximately 100 or greater.

NOISE PERFORMANCE

The INA114 provides very low noise in most applications. For differential source impedances less than 1k , the INA103 may provide lower noise. For source impedances greater than 50k , the INA111 FET-input instrumentation ampli- fier may provide lower noise.

Low frequency noise of the INA114 is approximately 0.4 Vp-p measured from 0.1 to 10Hz. This is approximately one-tenth the noise of "low noise" chopper-stabilized amplifiers.



DESIRED GAIN	R_G ( )	NEAREST 1% R_G ( )
1	No Connection	No Connection
2	50.00k	49.9k
5	12.50k	12.4k
10	5.556k	5.62k
20	2.632k	2.61k
50	1.02k	1.02k
100	505.1	511
200	251.3	249
500	100.2	100
1000	50.05	49.9
2000	25.01	24.9
5000	10.00	10
10000	5.001	4.99

FIGURE 1. Basic Connections.

OFFSET TRIMMING

The INA114 is laser trimmed for very low offset voltage and drift. Most applications require no external offset adjustment. Figure 2 shows an optional circuit for trimming the output offset voltage. The voltage applied to Ref terminal is summed at the output. Low impedance must be maintained at this node to assure good common-mode rejection. This is achieved by buffering trim voltage with an op amp as shown.

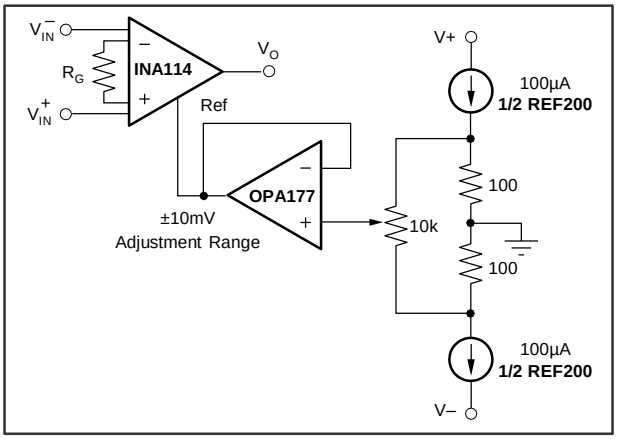


FIGURE 2. Optional Trimming of Output Offset Voltage.

INPUT BIAS CURRENT RETURN PATH

The input impedance of the INA114 is extremely high—approximately 10^{10} . However, a path must be provided for the input bias current of both inputs. This input bias current is typically less than 1nA (it can be either polarity due to cancellation circuitry). High input impedance means that this input bias current changes very little with varying input voltage.

Input circuitry must provide a path for this input bias current if the INA114 is to operate properly. Figure 3 shows various provisions for an input bias current path. Without a bias current return path, the inputs will float to a potential which exceeds the common-mode range of the INA114 and the input amplifiers will saturate. If the differential source resistance is low, bias current return path can be connected to one input (see thermocouple example in Figure 3). With higher source impedance, using two resistors provides a balanced input with possible advantages of lower input offset voltage due to bias current and better common-mode rejection.

INPUT COMMON-MODE RANGE

The linear common-mode range of the input op amps of the INA114 is approximately 13.75V (or 1.25V from the power supplies). As the output voltage increases, however, the linear input range will be limited by the output voltage swing of the input amplifiers, A_1 and A_2. The common-mode range is related to the output voltage of the complete amplifier—see performance curve "Input Common-Mode Range vs Output Voltage."

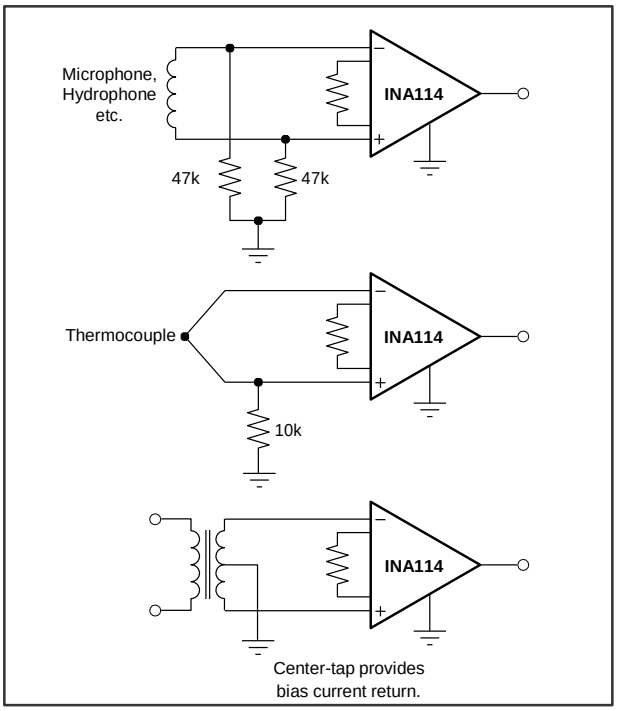


FIGURE 3. Providing an Input Common-Mode Current Path.

A combination of common-mode and differential input signals can cause the output of A_1 or A_2 to saturate. Figure 4 shows the output voltage swing of A_1 and A_2 expressed in terms of a common-mode and differential input voltages. Output swing capability of these internal amplifiers is the same as the output amplifier, A_3. For applications where input common-mode range must be maximized, limit the output voltage swing by connecting the INA114 in a lower gain (see performance curve "Input Common-Mode Voltage Range vs Output Voltage"). If necessary, add gain after the INA114 to increase the voltage swing.

Input-overload often produces an output voltage that appears normal. For example, an input voltage of +20V on one input and +40V on the other input will obviously exceed the linear common-mode range of both input amplifiers. Since both input amplifiers are saturated to nearly the same output voltage limit, the difference voltage measured by the output amplifier will be near zero. The output of the INA114 will be near 0V even though both inputs are overloaded.

INPUT PROTECTION

The inputs of the INA114 are individually protected for voltages up to 40V. For example, a condition of –40V on one input and +40V on the other input will not cause damage. Internal circuitry on each input provides low series impedance under normal signal conditions. To provide equivalent protection, series input resistors would contribute excessive noise. If the input is overloaded, the protection circuitry limits the input current to a safe value (approximately 1.5mA). The typical performance curve "Input Bias Current vs Common-Mode Input Voltage" shows this input

current limit behavior. The inputs are protected even if no power supply voltage is present.

OUTPUT VOLTAGE SENSE (SOL-16 package only)

The surface-mount version of the INA114 has a separate output sense feedback connection (pin 12). Pin 12 must be connected to the output terminal (pin 11) for proper operation. (This connection is made internally on the DIP version of the INA114.)

The output sense connection can be used to sense the output voltage directly at the load for best accuracy. Figure 5 shows how to drive a load through series interconnection resistance. Remotely located feedback paths may cause instability. This can be generally be eliminated with a high frequency feedback path through C_1. Heavy loads or long lines can be driven by connecting a buffer inside the feedback path (Figure 6).

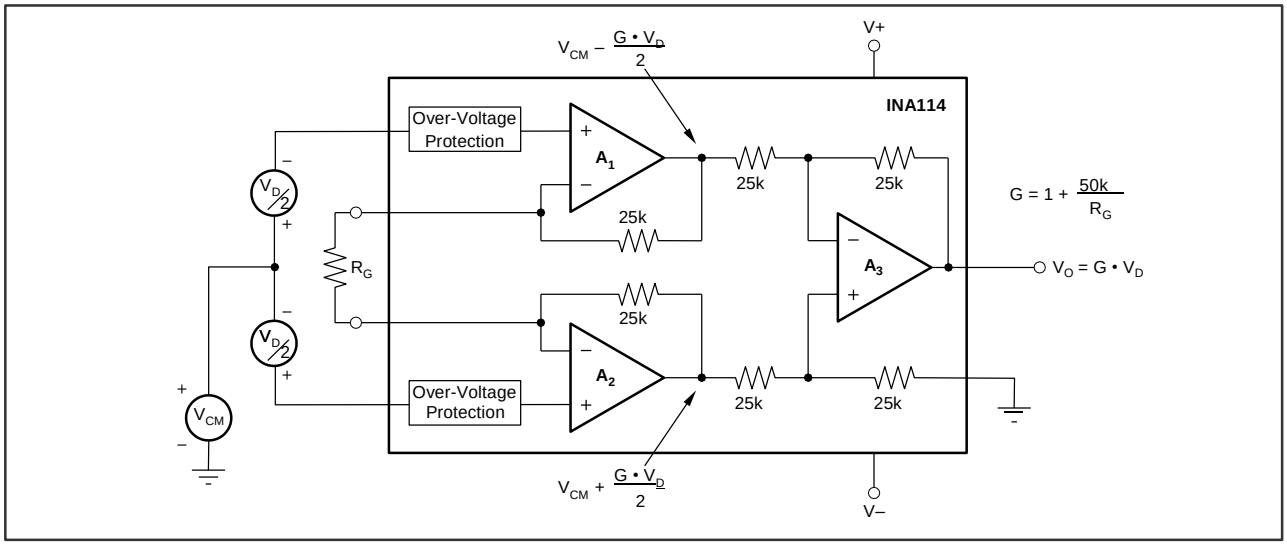


FIGURE 4. Voltage Swing of A_1 and A_2.

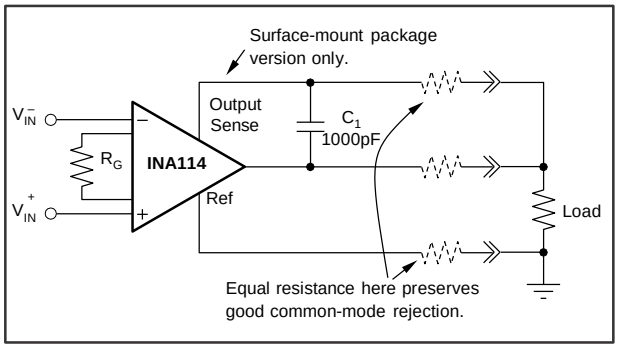


FIGURE 5. Remote Load and Ground Sensing.

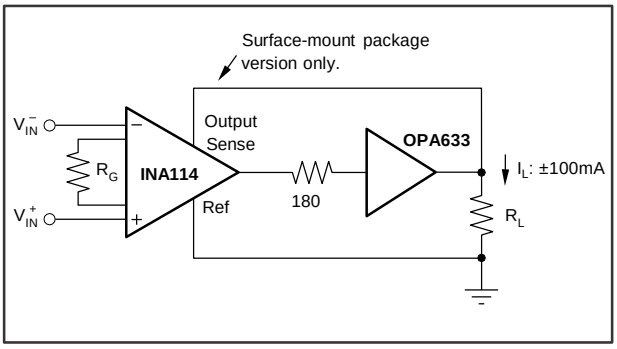


FIGURE 6. Buffered Output for Heavy Loads.

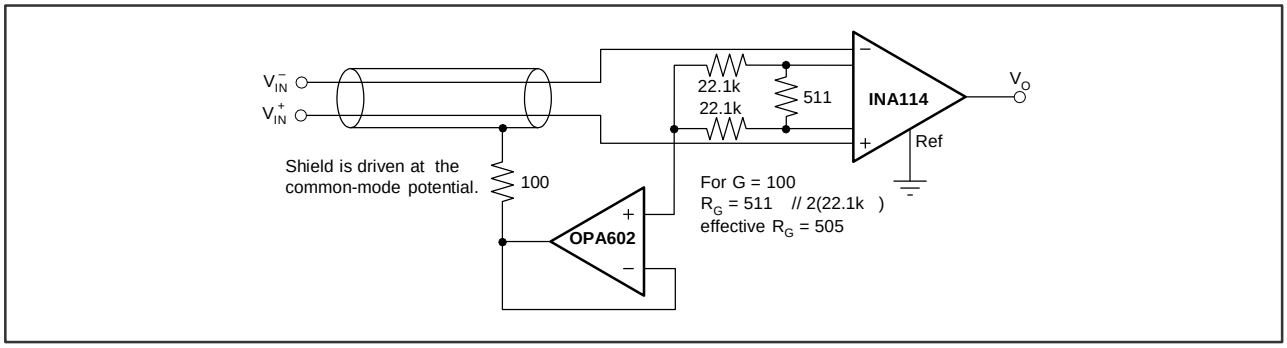


FIGURE 7. Shield Driver Circuit.

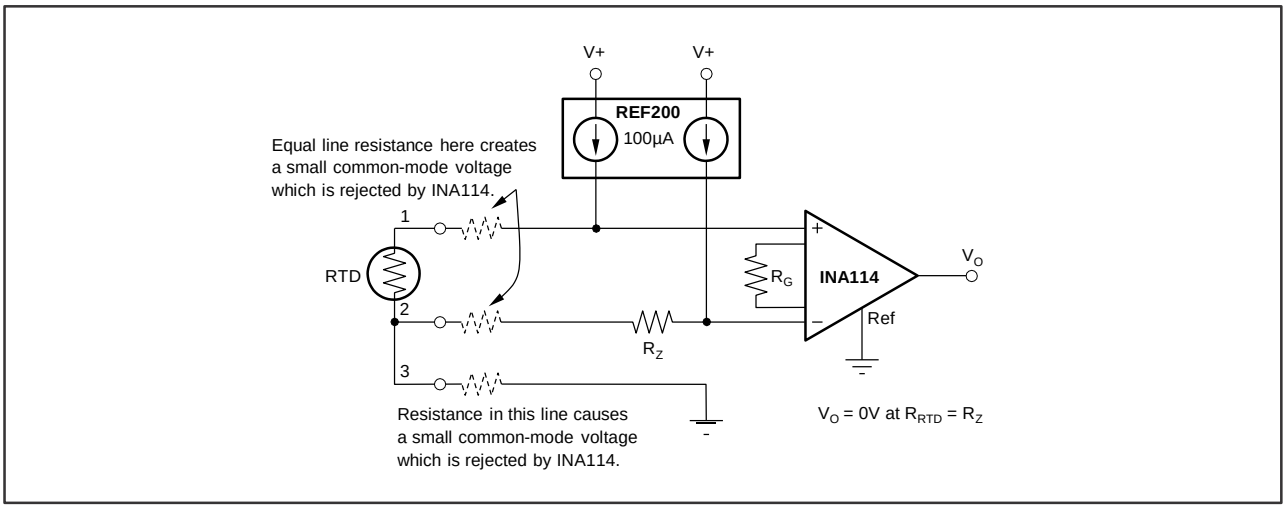
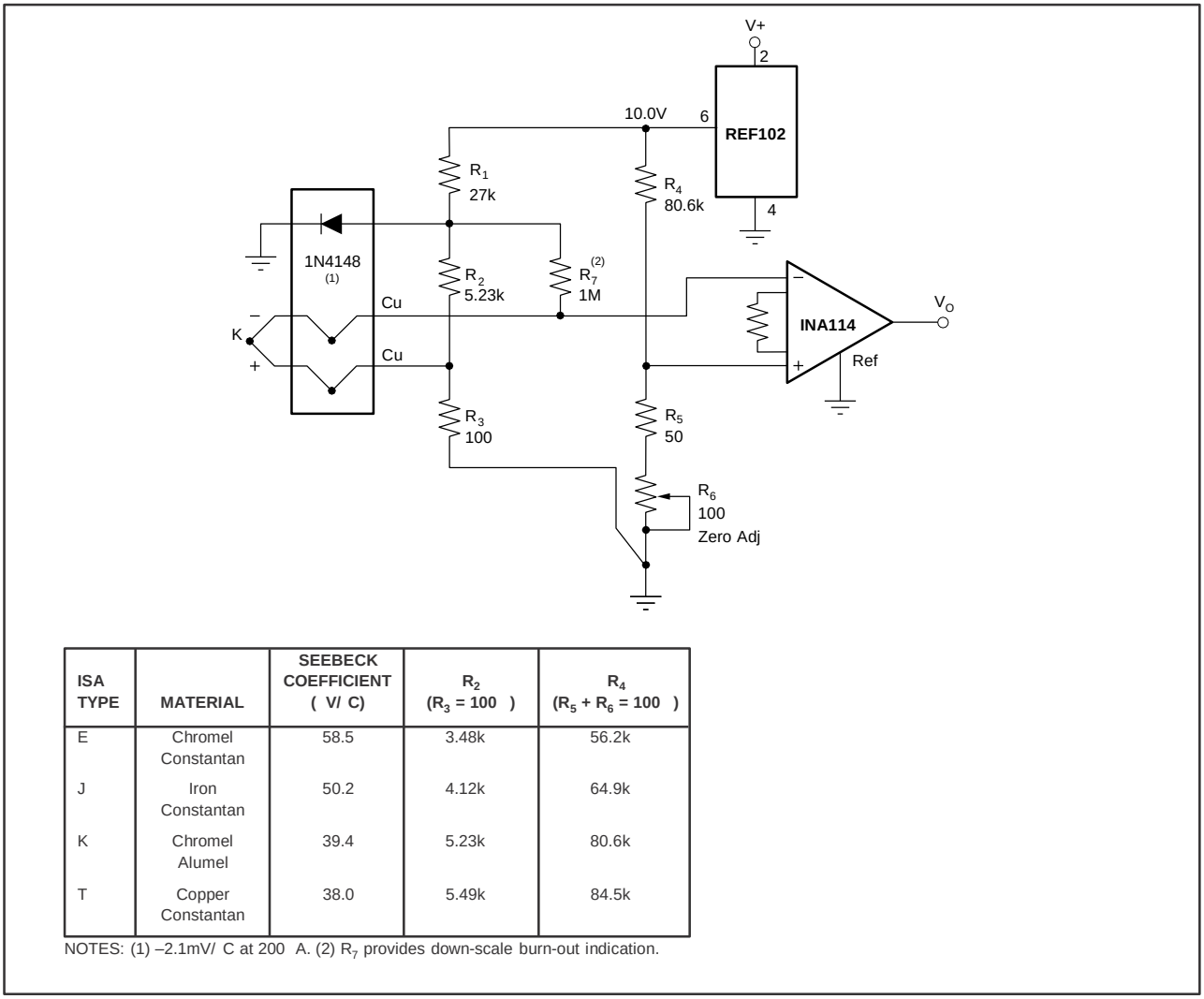


FIGURE 8. RTD Temperature Measurement Circuit.



ISA TYPE	MATERIAL	SEEBECK COEFFICIENT (V/ C)	R_2 (R_3 = 100)	R_4 (R_5 + R_6 = 100)
E	Chromel Constantan	58.5	3.48k	56.2k
J	Iron Constantan	50.2	4.12k	64.9k
K	Chromel Alumel	39.4	5.23k	80.6k
T	Copper Constantan	38.0	5.49k	84.5k

NOTES: (1) –2.1mV/ C at 200 A. (2) R_7 provides down-scale burn-out indication.

FIGURE 9. Thermocouple Amplifier With Cold Junction Compensation.

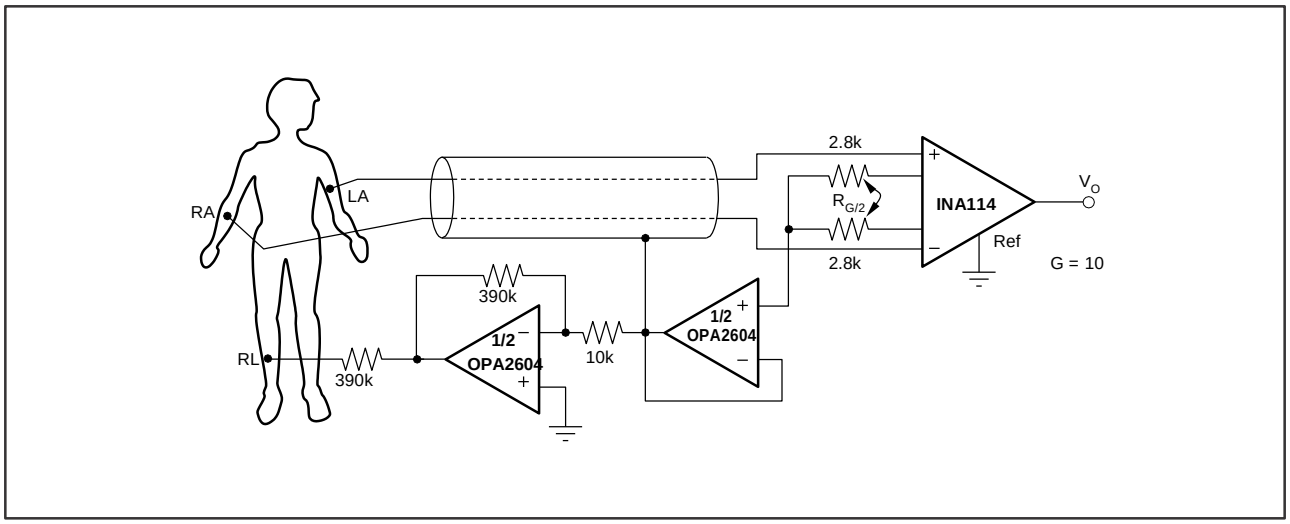


FIGURE 10. ECG Amplifier With Right-Leg Drive.

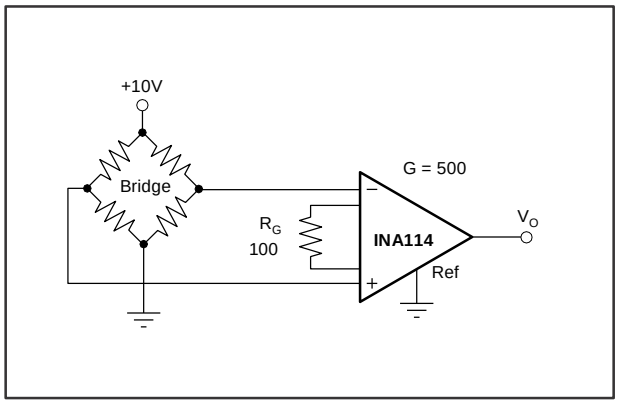


FIGURE 11. Bridge Transducer Amplifier.

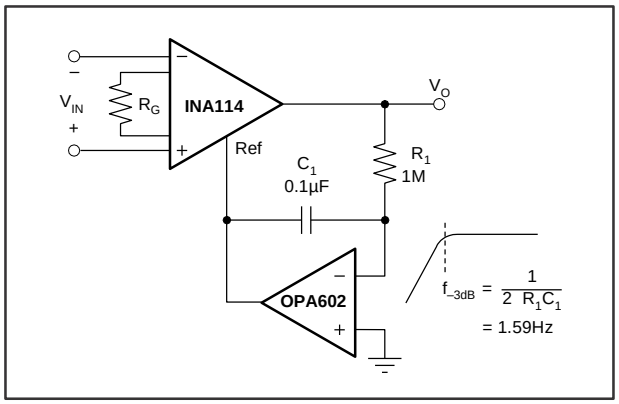
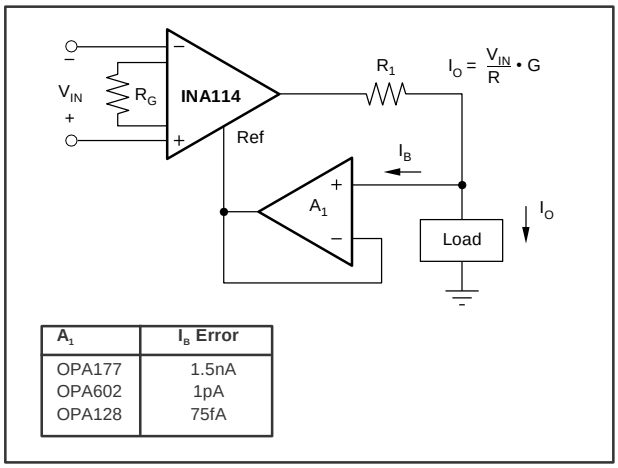


FIGURE 12. AC-Coupled Instrumentation Amplifier.



A_1	I_B Error
OPA177	1.5nA
OPA602	1pA
OPA128	75fA

FIGURE 13. Differential Voltage-to-Current Converter.

PACKAGING INFORMATION

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish	MSL Peak Temp (3)
INA114AP	ACTIVE	PDIP	P	8	50	TBD	Call TI	Level-NA-NA-NA
INA114AU	ACTIVE	SOIC	DW	16	48	Pb-Free (RoHS)	CU NIPDAU	Level-3-260C-168 HR
INA114AU/1K	ACTIVE	SOIC	DW	16	1000	Pb-Free (RoHS)	CU NIPDAU	Level-3-260C-168 HR
INA114AU/1KE4	ACTIVE	SOIC	DW	16	1000	Pb-Free (RoHS)	CU NIPDAU	Level-3-260C-168 HR
INA114AUE4	ACTIVE	SOIC	DW	16	48	Pb-Free (RoHS)	CU NIPDAU	Level-3-260C-168 HR
INA114BP	ACTIVE	PDIP	P	8	50	TBD	Call TI	Level-NA-NA-NA
INA114BU	ACTIVE	SOIC	DW	16	48	Pb-Free (RoHS)	CU NIPDAU	Level-3-260C-168 HR
INA114BU/1K	ACTIVE	SOIC	DW	16	1000	Pb-Free (RoHS)	CU NIPDAU	Level-3-260C-168 HR

(1) The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

(2) Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS) or Green (RoHS & no Sb/Br) - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.
TBD: The Pb-Free/Green conversion plan has not been defined.
Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.
Green (RoHS & no Sb/Br): TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

(3) MSL, Peak Temp. -- The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

ANEXO 14: HOJAS DE ESPECIFICACIONES DEL INTEGRADO TL084.

TL084, TL084A, TL084B

General purpose JFET quad operational amplifiers

Datasheet — production data

Features

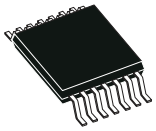
- Wide common-mode (up to V_{CC}^{+}) and differential voltage range
- Low input bias and offset current
- Output short-circuit protection
- High input impedance JFET input stage
- Internal frequency compensation
- Latch up free operation
- High slew rate: 16 V/µs (typical)

Description

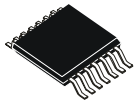
The TL084, TL084A, and TL084B are high-speed, JFET input, quad operational amplifiers incorporating well matched, high voltage JFET and bipolar transistors in a monolithic integrated circuit.

The devices feature high slew rates, low input bias and offset currents, and low offset voltage temperature coefficient.

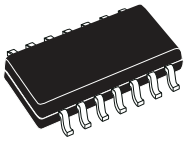
N
DIP14
(Plastic package)



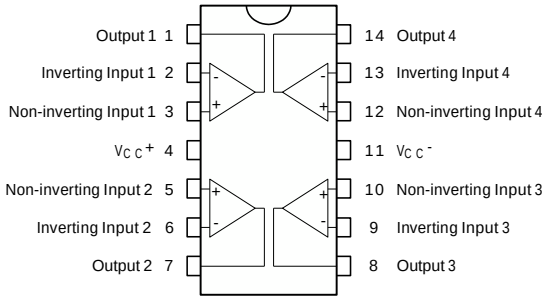
D
TSSOP14
(Thin shrink small outline package)



D SO-14
(Plastic micropackage)



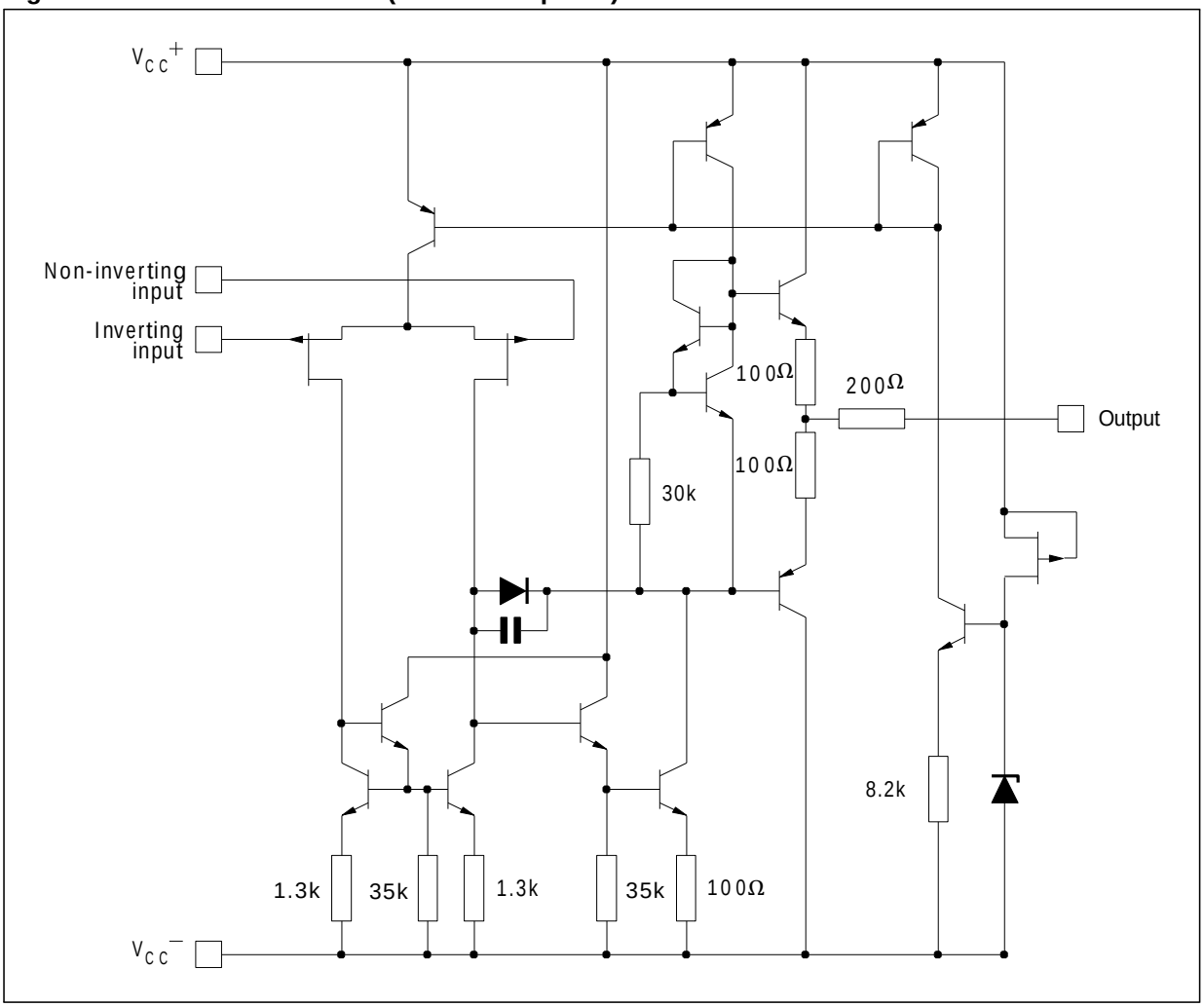
Pin connections
(Top view)



Contents

1 Schematic diagram

Figure 1. Circuit schematics (for each amplifier)



2 Absolute maximum ratings and operating conditions

Table 1. Absolute maximum ratings

Symbol	Parameter	Value	Unit
V_{CC}	Supply voltage(1)	±18	V
V_{in}	Input voltage(2)	±15	
V_{id}	Differential input voltage(3)	±30	
R_{thja}	Thermal resistance junction to ambient(4)(5) DIP14 TSSOP14 SO-14	 80 100 105	°C/W
R_{thjc}	Thermal resistance junction to case(4)(5) DIP14 TSSOP14 SO-14	 33 32 31	
P_{tot}	Power dissipation	680	mW
	Output short-circuit duration(6)	Infinite	
T_{oper}	Operating free-air temperature range: for TL084I/TL084AI/TL084BI	-40 to +105	°C
	Operating free-air temperature range: for TL084C/TL084AC/TL084BC	0 to +70	
T_{stg}	Storage temperature range	-65 to +150	
ESD	HBM: human body model(7)	1000	V
	MM: machine model(8)	150	
	CDM: charged device model(9)	1500	

1. All voltage values, except differential voltage, are with respect to the zero reference level (ground) of the supply voltages where the zero reference level is the midpoint between V_{CC}^+ and V_{CC}^-.
2. The magnitude of the input voltage must never exceed the magnitude of the supply voltage or 15 volts, whichever is less.
3. Differential voltages are the non-inverting input terminal with respect to the inverting input terminal.
4. Short-circuits can cause excessive heating and destructive dissipation.
5. R_{th} are typical values.
6. The output may be shorted to ground or to either supply. Temperature and/or supply voltages must be limited to ensure that the dissipation rating is not exceeded.
7. Human body model: 100 pF discharged through a 1.5 kΩ resistor between two pins of the device, done for all couples of pin combinations with other pins floating.
8. Machine model: a 200 pF cap is charged to the specified voltage, then discharged directly between two pins of the device with no external series resistor (internal resistor < 5 Ω), done for all couples of pin combinations with other pins floating.
9. Charged device model: all pins plus package are charged together to the specified voltage and then discharged directly to the ground.

Table 2. Operating conditions

Symbol	Parameter	TL084I/AI/BI	TL084C/AC/BC	Unit
V_{CC}	Supply voltage range	6 to 36		V
T_{oper}	Operating free-air temperature range	-40 to +105	0 to +70	°C

3 Electrical characteristics

Table 3. V_{CC} = ±15 V, T_{amb} = +25 °C (unless otherwise specified)

Symbol	Parameter	TL084I/AI/AC/BI/BC			TL084C			Unit
		Min.	Typ.	Max.	Min.	Typ.	Max.	
V_{io}	Input offset voltage (R_S = 50 Ω) T_{amb} = +25 °C TL084 T_{amb} = +25 °C TL084A T_{amb} = +25 °C TL084B $T_{min} \le T_{amb} \le T_{max}$ TL084 $T_{min} \le T_{amb} \le T_{max}$ TL084A $T_{min} \le T_{amb} \le T_{max}$ TL084B		 3 3 1	 10 6 3 13 7 5		3	 10 13	mV
$\Delta V_{io}/\Delta T$	Input offset voltage drift		10			10		µV/°C
I_{io}	Input offset current T_{amb} = +25 °C $T_{min} \le T_{amb} \le T_{max}$		5	 100 4		5	 100 4	 pA nA
I_{ib}	Input bias current(1) T_{amb} = +25 °C $T_{min} \le T_{amb} \le T_{max}$		20	 200 20		30	 200 20	 pA nA
A_{vd}	Large signal voltage gain (R_L = 2 kΩ, V_o = ±10 V) T_{amb} = +25 °C $T_{min} \le T_{amb} \le T_{max}$	 50 25	200		 25 15	200		V/mV
SVR	Supply voltage rejection ratio (R_S = 50 Ω) T_{amb} = +25 °C $T_{min} \le T_{amb} \le T_{max}$	 80 80	86		 70 70	86		dB
I_{CC}	Supply current, no load T_{amb} = +25 °C $T_{min} \le T_{amb} \le T_{max}$		1.4	 2.5 2.5		1.4	 2.5 2.5	mA
V_{icm}	Input common mode voltage range	±11	+15 -12		±11	+15 -12		V
CMR	Common mode rejection ratio (R_S = 50 Ω) T_{amb} = +25 °C $T_{min} \le T_{amb} \le T_{max}$	 80 80	86		 70 70	86		dB
I_{os}	Output short-circuit current T_{amb} = +25 °C $T_{min} \le T_{amb} \le T_{max}$	 10 10	40	 60 60	 10 10	40	 60 60	mA
$\pm V_{opp}$	Output voltage swing T_{amb} = +25 °C R_L = 2 kΩ R_L = 10 kΩ $T_{min} \le T_{amb} \le T_{max}$ R_L = 2 kΩ R_L = 10 kΩ	 10 12 10 12	 12 13.5		 10 12 10 12	 12 13.5		V
SR	Slew rate V_{in} = 10 V, R_L = 2 kΩ, C_L = 100 pF, unity gain	8	16		8	16		V/µs

Table 3. V_{CC} = ±15 V, T_{amb} = +25 °C (unless otherwise specified) (continued)

Symbol	Parameter	TL084I/AI/AC/BI/BC			TL084C			Unit
		Min.	Typ.	Max.	Min.	Typ.	Max.	
t_r	Rise time V_{in} = 20 mV, R_L = 2 kΩ, C_L = 100 pF, unity gain		0.1			0.1		µs
K_{ov}	Overshoot V_{in} = 20 mV, R_L = 2 kΩ, C_L = 100 pF, unity gain		10			10		%
GBP	Gain bandwidth product V_{in} = 10 mV, R_L = 2 kΩ, C_L = 100 pF, F= 100 kHz	2.5	4		2.5	4		MHz
R_i	Input resistance		10^{12}			10^{12}		Ω
THD	Total harmonic distortion F= 1 kHz, R_L = 2 kΩ,C_L = 100 pF, A_v = 20 dB, V_o = 2 V_{pp})		0.01			0.01		%
e_n	Equivalent input noise voltage R_S = 100 Ω, F= 1 kHz		15			15		$\frac{nV}{\sqrt{Hz}}$
Øm	Phase margin		45			45		degrees
V_{o1}/V_{o2}	Channel separation A_v = 100		120			120		dB

1. The input bias currents are junction leakage currents which approximately double for every 10°C increase in the junction temperature.

Figure 2. Maximum peak-to-peak output voltage vs. frequency (R_L = 2 kΩ)

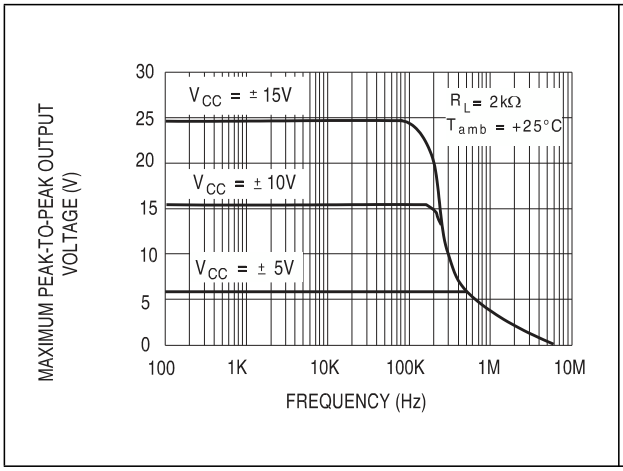


Figure 3. Maximum peak-to-peak output voltage vs. frequency (R_L = 10 kΩ)

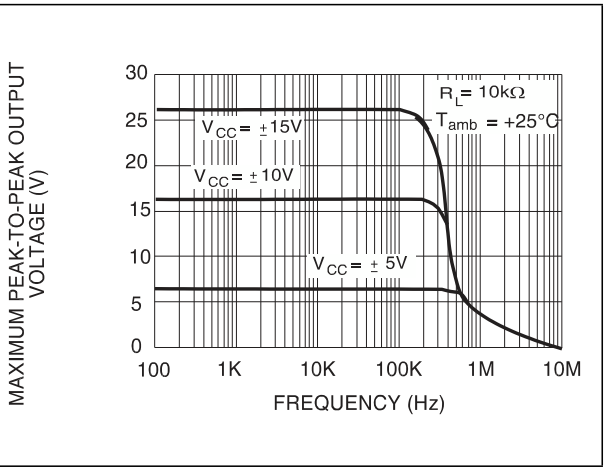


Figure 4. Maximum peak-to-peak output voltage vs. frequency and temp.

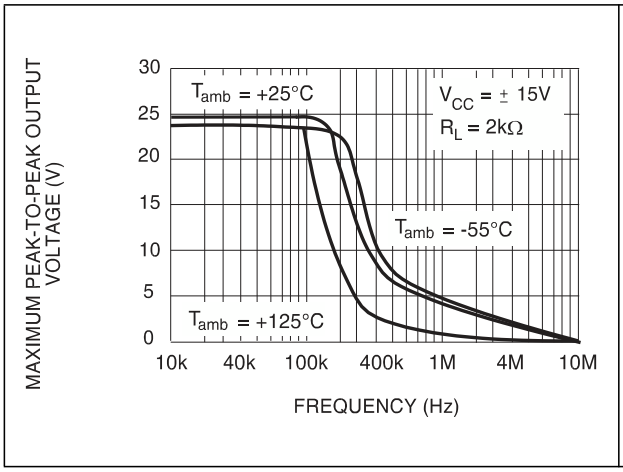


Figure 5. Maximum peak-to-peak output voltage vs. free air temp.

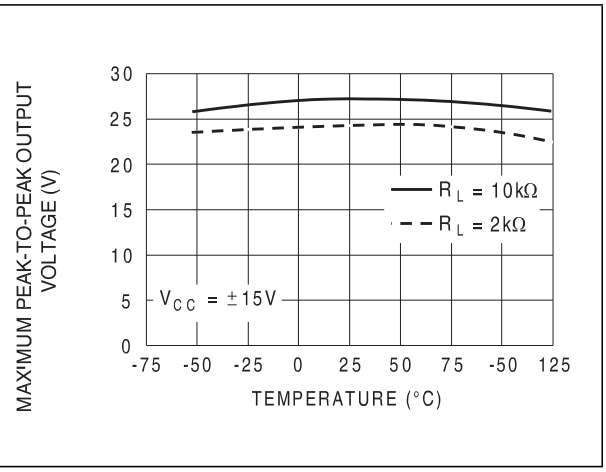


Figure 6. Maximum peak-to-peak output voltage vs. load resistance

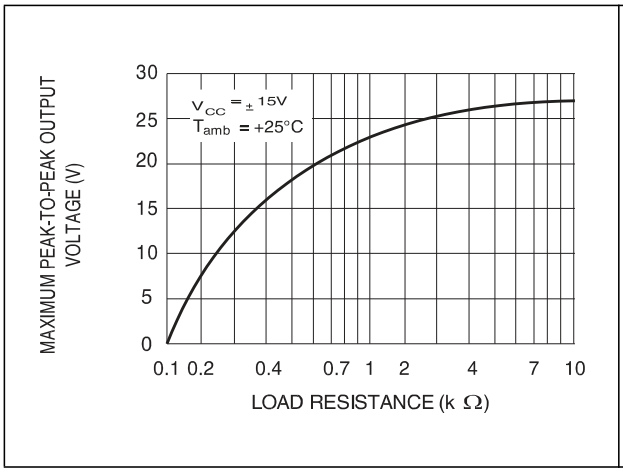


Figure 7. Maximum peak-to-peak output voltage vs. supply voltage

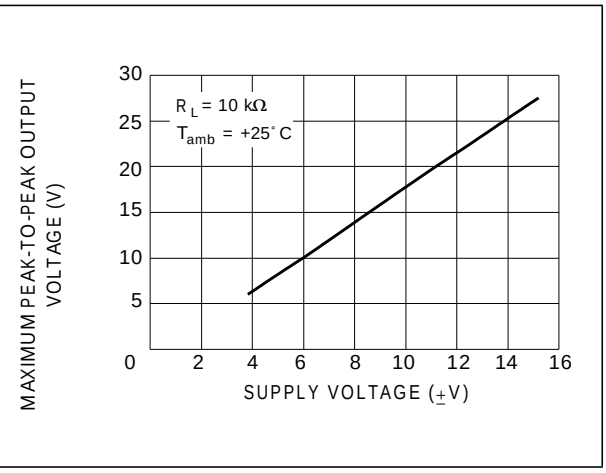


Figure 8. Input bias current vs. free air temp.

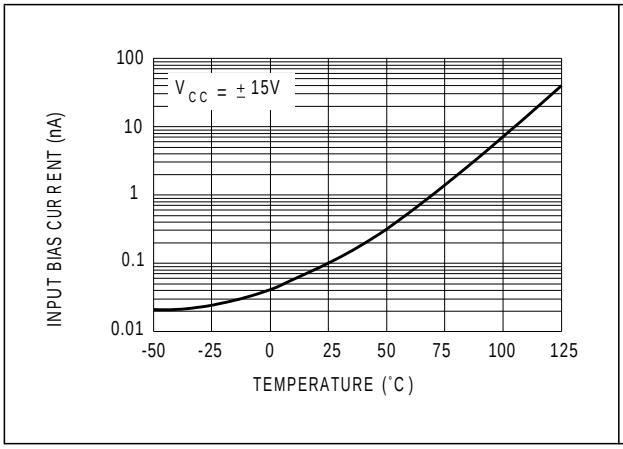


Figure 9. Large signal differential voltage amplification vs. free air temp.

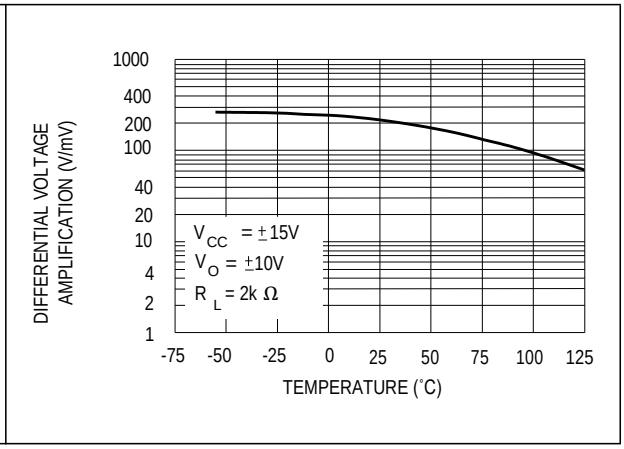


Figure 10. Large signal differential voltage amplification and phase shift vs. frequency

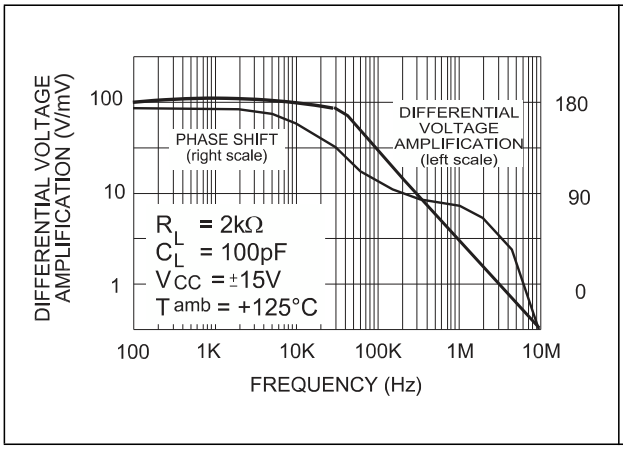


Figure 11. Total power dissipation vs. free air temp.

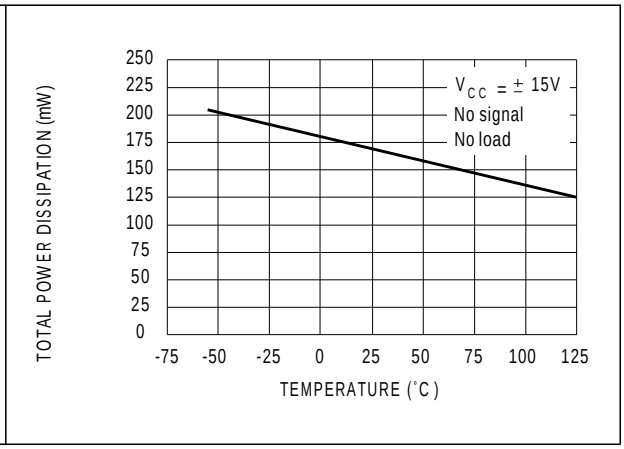


Figure 12. Supply current per amplifier vs. free air temp.

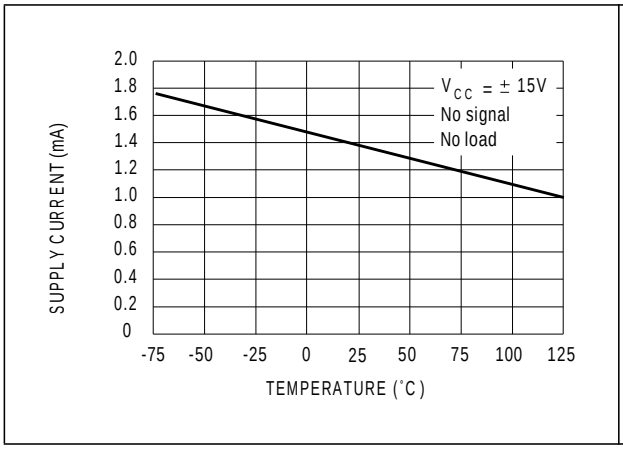


Figure 13. Supply current per amplifier vs. supply voltage

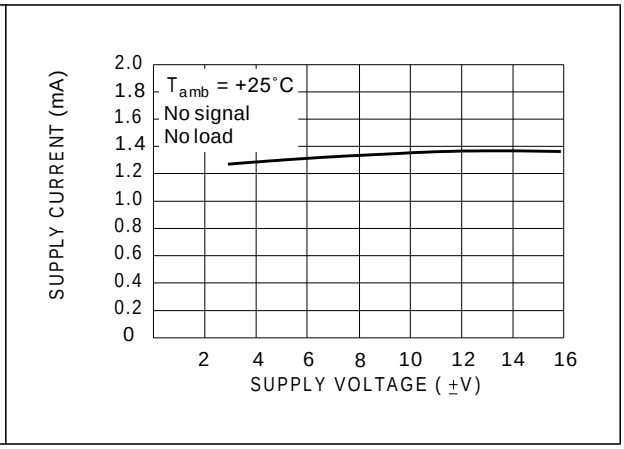


Figure 14. Common mode rejection ratio vs. free air temp.

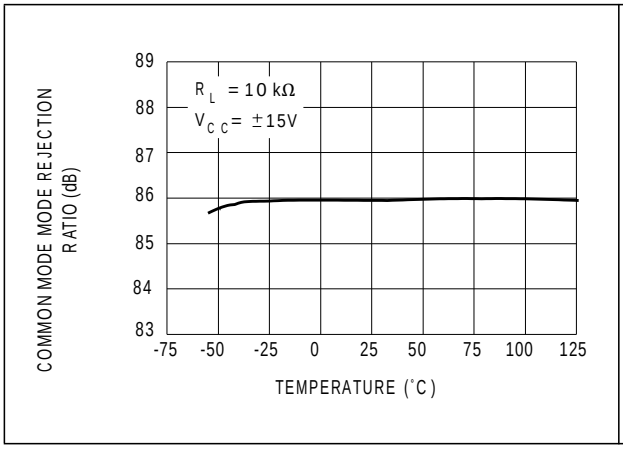


Figure 15. Voltage follower large signal pulse response

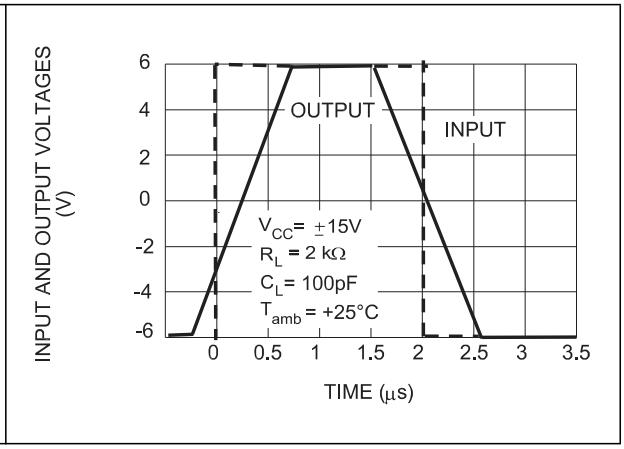


Figure 16. Output voltage vs. elapsed time

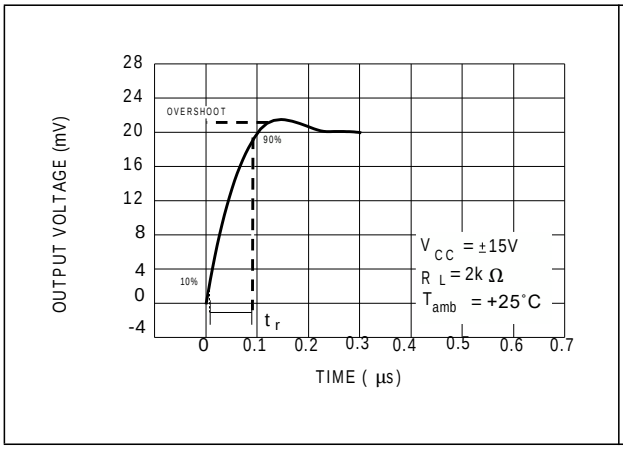


Figure 17. Equivalent input noise voltage vs. frequency

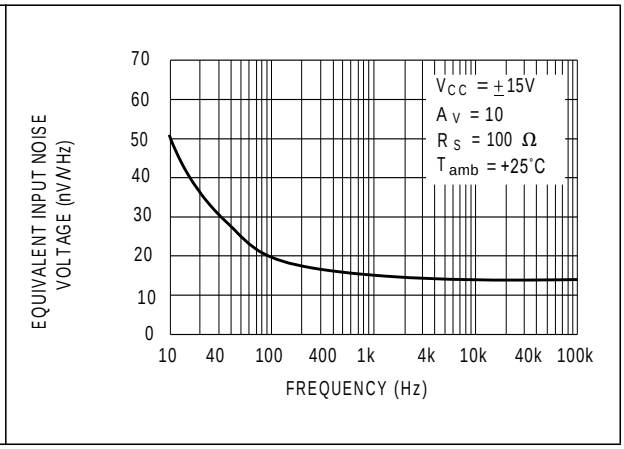
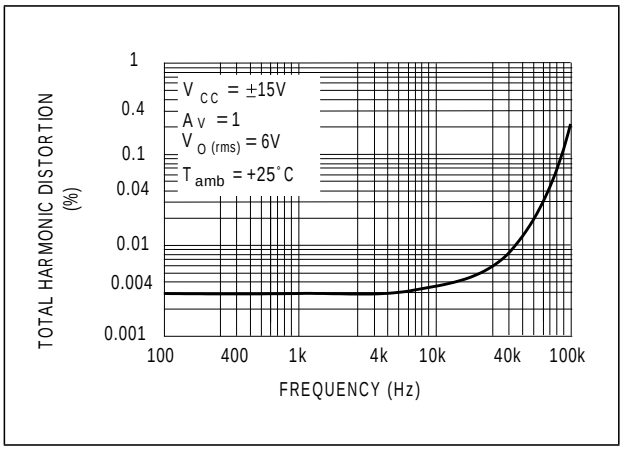


Figure 18. Total harmonic distortion vs. frequency



4 Parameter measurement information

Figure 19. Voltage follower

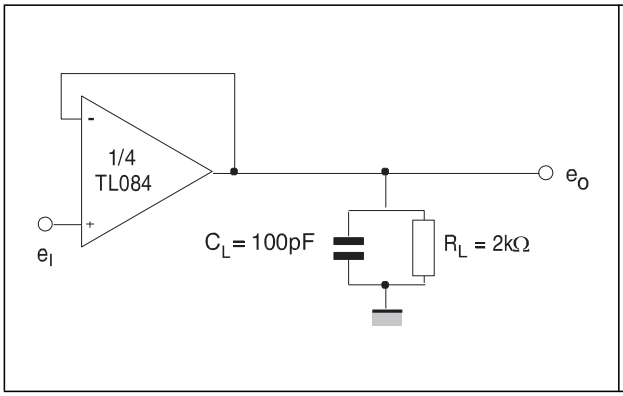
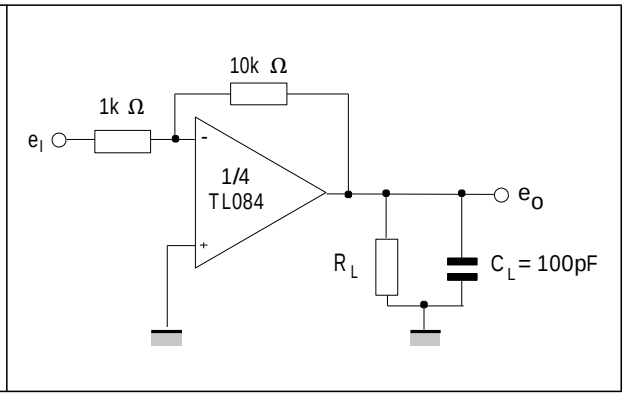


Figure 20. Gain-of-10 inverting amplifier



5 Typical applications

Figure 21. Audio distribution amplifier

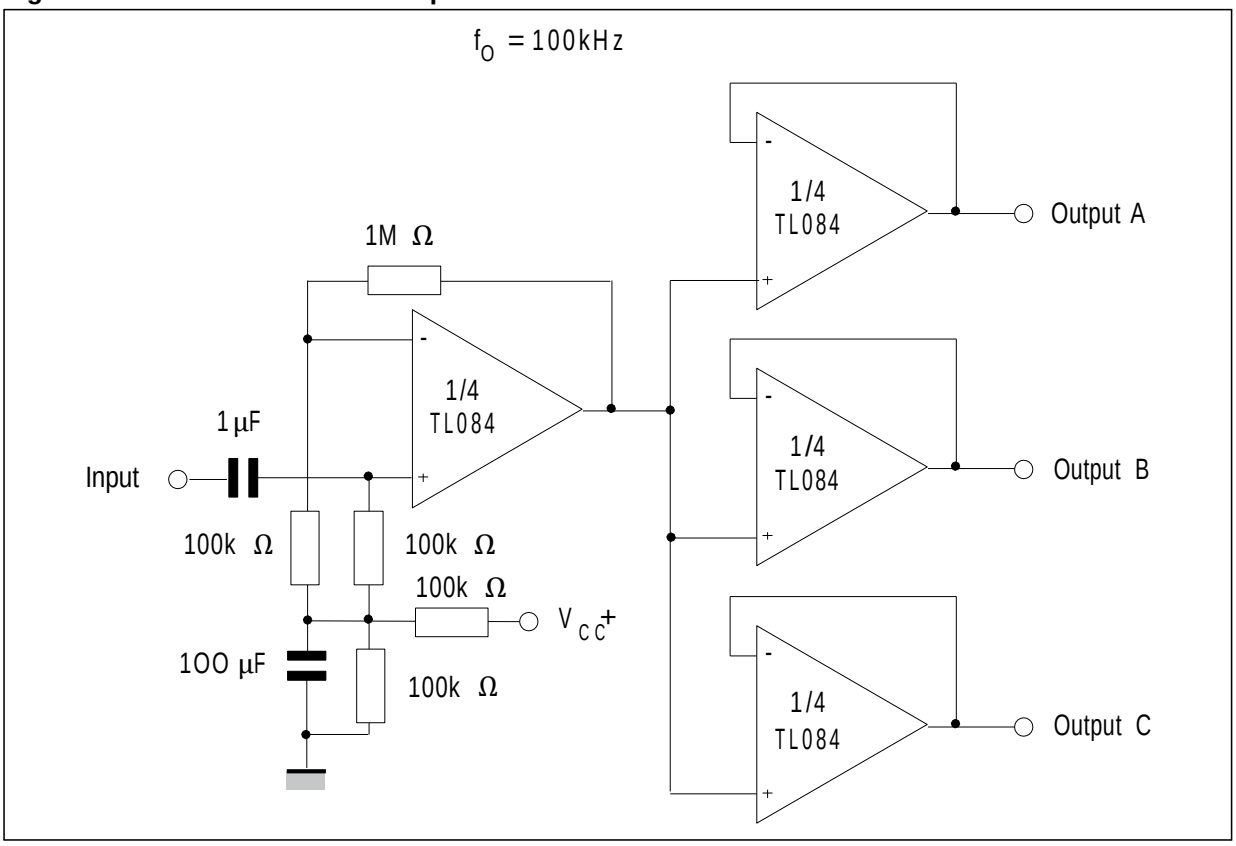


Figure 22. Positive feeback bandpass filter

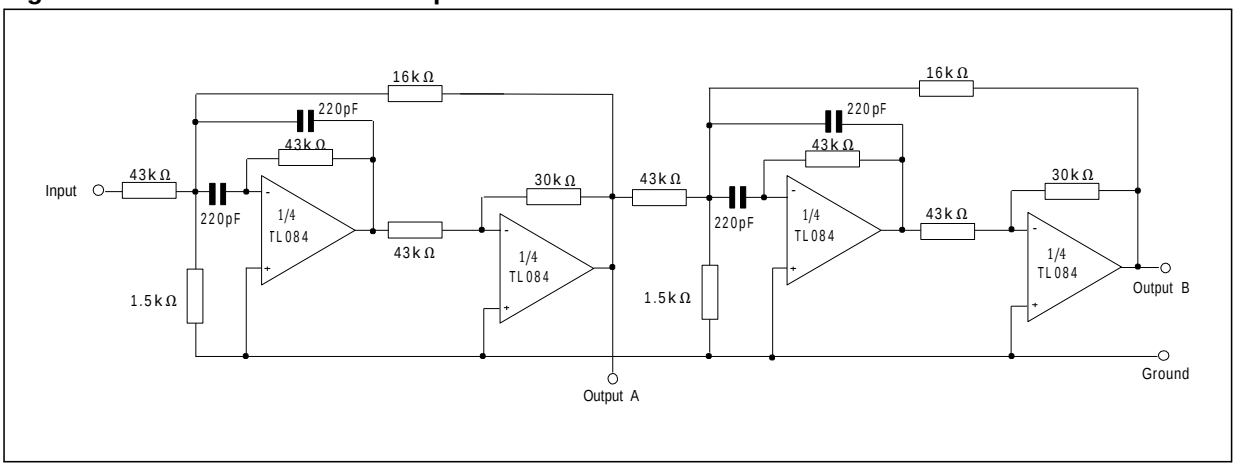
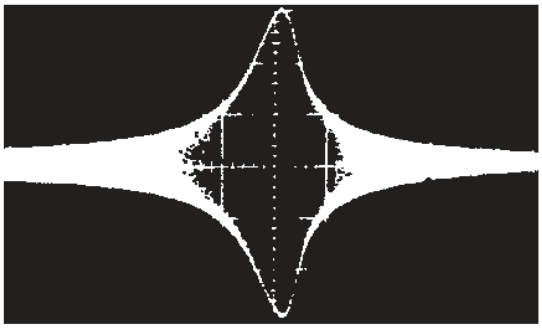
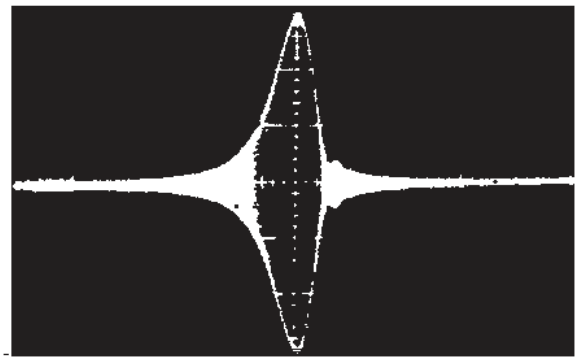


Figure 23. Output A



Second order bandpass filter
fo = 100 kHz; Q = 30; Gain = 4

Figure 24. Output B



Cascaded bandpass filter
fo = 100 kHz; Q = 69; Gain = 16

6 Package information

In order to meet environmental requirements, ST offers these devices in different grades of ECOPACK® packages, depending on their level of environmental compliance. ECOPACK® specifications, grade definitions and product status are available at: *www.st.com*. ECOPACK® is an ST trademark.

6.1 DIP14 package information

Figure 25. DIP14 package mechanical drawing

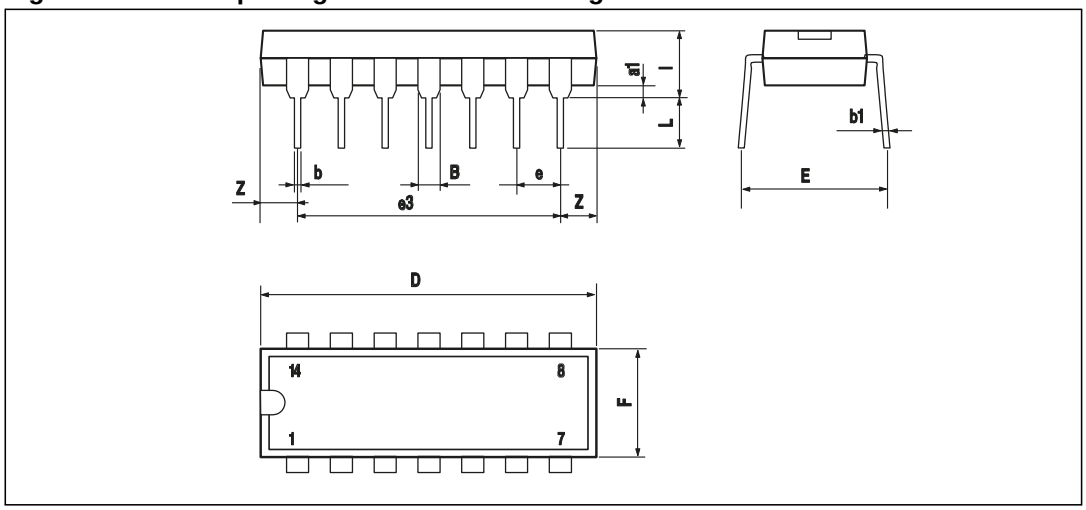


Table 4. DIP14 package mechanical data

Ref.	Dimensions					
	Millimeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
a1	0.51			0.020		
B	1.39		1.65	0.055		0.065
b		0.5			0.020	
b1		0.25			0.010	
D			20			0.787
E		8.5			0.335	
e		2.54			0.100	
e3		15.24			0.600	
F			7.1			0.280
I			5.1			0.201
L		3.3			0.130	
Z	1.27		2.54	0.050		0.100

6.2 TSSOP14 package information

Figure 26. TSSOP14 package mechanical drawing

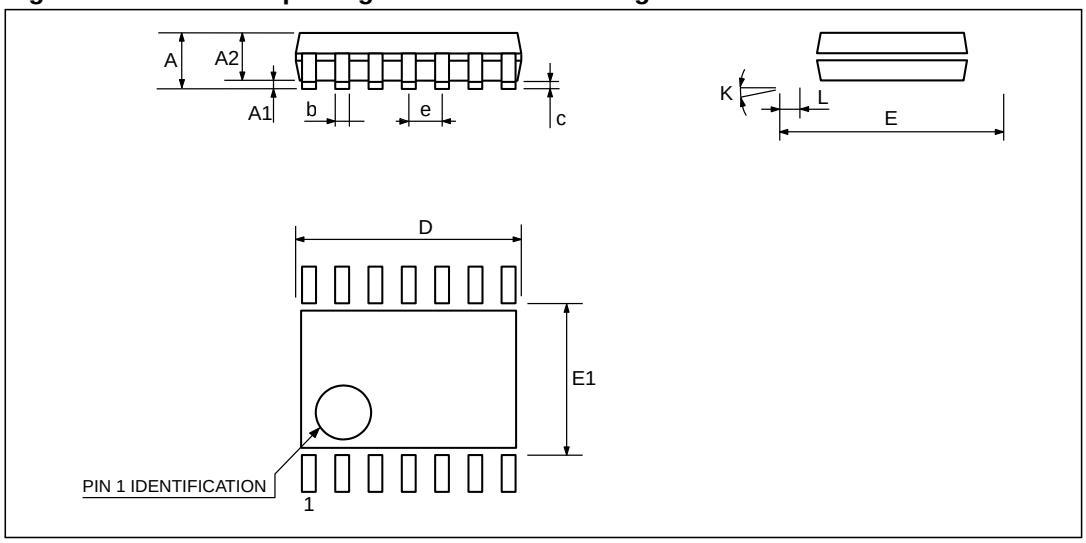


Figure 27. TSSOP14 package mechanical data

Ref.	Millimeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A			1.2			0.047
A1	0.05		0.15	0.002	0.004	0.006
A2	0.8	1	1.05	0.031	0.039	0.041
b	0.19		0.30	0.007		0.012
c	0.09		0.20	0.004		0.0089
D	4.9	5	5.1	0.193	0.197	0.201
E	6.2	6.4	6.6	0.244	0.252	0.260
E1	4.3	4.4	4.48	0.169	0.173	0.176
e		0.65 BSC			0.0256 BSC	
K	0°		8°	0°		8°
L1	0.45	0.60	0.75	0.018	0.024	0.030

6.3 SO-14 package information

Figure 28. SO-14 package mechanical drawing

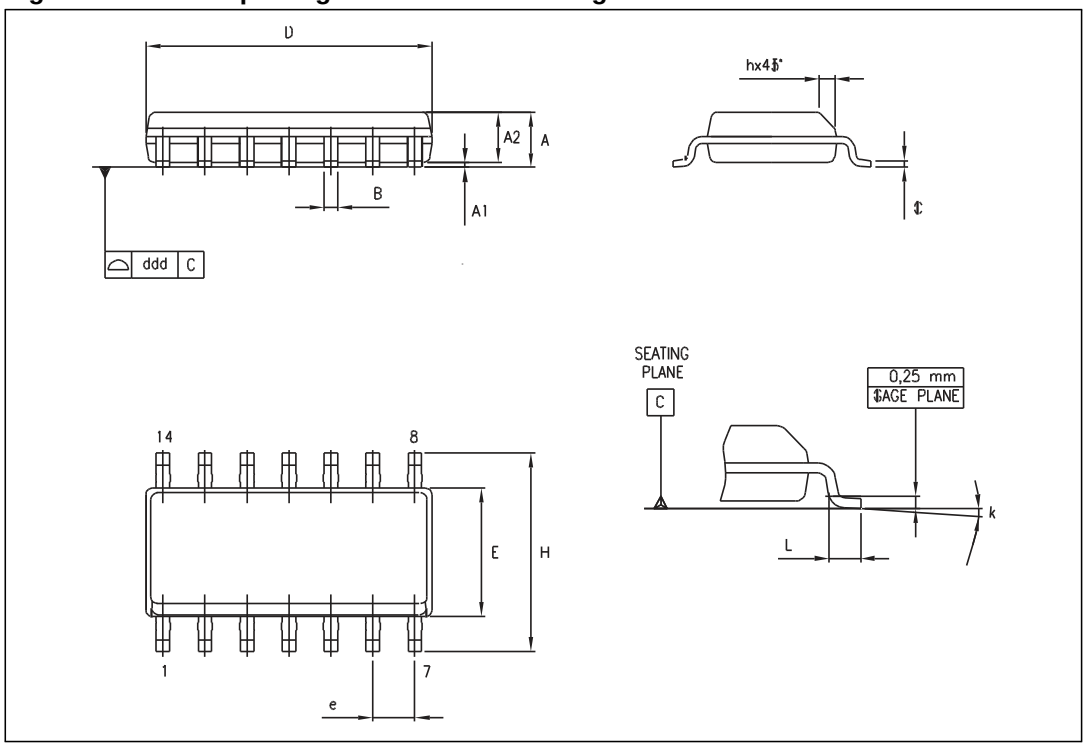


Table 5. SO-14 package mechanical data

Dimensions						
Ref.	Millimeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A	1.35		1.75	0.05		0.068
A1	0.10		0.25	0.004		0.009
A2	1.10		1.65	0.04		0.06
B	0.33		0.51	0.01		0.02
C	0.19		0.25	0.007		0.009
D	8.55		8.75	0.33		0.34
E	3.80		4.0	0.15		0.15
e		1.27			0.05	
H	5.80		6.20	0.22		0.24
h	0.25		0.50	0.009		0.02
L	0.40		1.27	0.015		0.05
k	8° (max.)					
ddd			0.10			0.004

7 Ordering information

Table 6. Order codes

Order code	Temperature range	Package	Packing	Marking
TL084IN TL084AIN TL084BIN	-40°C, +105°C	DIP14	Tube	TL084IN TL084AIN TL084BIN
TL084ID/IDT TL084AID/AIDT TL084BID/BIDT		SO-14	Tube or tape & reel	084I 084AI 084BI
TL084IYDT(1) TL084AIYDT(1) TL084BIYDT(1)		SO-14 (Automotive grade)	Tube or tape & reel	084IY 084AIY 084BIY
TL084IP/IPT TL084AIP/AIPT TL084BIP/BIPT		TSSOP14	Tube or tape & reel	084I 084AI 084BI
TL084CN TL084ACN TL084BCN	0°C, +70°C	DIP14	Tube	TL084CN TL084ACN TL084BCN
TL084CD/CDT TL084ACD/ACDT TL084BCD/BCDT		SO-14	Tube or tape & reel	084C 084AC 084BC
TL084CP/CPT TL084ACP/ACPT TL084BCP/BCPT		TSSOP14	Tube or tape & reel	084C 084AC 084BC

1. Qualification and characterization according to AEC Q100 and Q003 or equivalent, advanced screening according to AEC Q001 & Q 002 or equivalent.

8 Revision history

Table 7. Document revision history

Date	Revision	Changes
28-Mar-2001	1	Initial release.
30-Jul-2007	2	Added values for R_{thja}, R_{thjc} and ESD in *Table 1: Absolute maximum ratings*. Added *Table 2: Operating conditions*. Expanded *Table 6: Order codes*. Template update.
15-Jul-2008	3	Removed information concerning military temperature ranges (TL084Mx, TL084AMx, TL084BMx). Added automotive grade order codes in *Table 6: Order codes*.
05-Jul-2012	4	Removed commercial types TL084IYD, TL084AIYD and TL084BIYD. Updated *Table 6: Order codes*.
29-Jan-2013	5	Added part numbers TL084A and TL084B. Added SO-14 package silhouette. Updated layout of *Table 1: Absolute maximum ratings*. Updated of *Table 3: V_{CC} = ±15 V, T_{amb} = +25 °C (unless otherwise specified)*. Replaced SO-14 package mechanical drawing (*Figure 28: SO-14 package mechanical drawing*). Replaced SO-14 package mechanical data (*Table 5: SO-14 package mechanical data*).

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY TWO AUTHORIZED ST REPRESENTATIVES, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com

ANEXO 15: HOJAS DE ESPECIFICACIONES DEL INTEGRADO TMA0505D.

DC/DC Converters

TMA Series, 1 Watt

Features

- Single-in-line (SIP) package
- Single and dual output models
- I/O isolation 1'000 VDC
- High efficiency up to 81%
- Operating temp. range –40°C to +85°C
- Industry standard pinout
- 100% Burn-in (8 h)
- Lead free design, RoHS compliant
- 3-year product warranty

The TMA series are miniature, isolated 1 W DC/DC-converters in a Single-in-Line package (SIP). Requiring only 1.2 cm2 board space they offer the ideal solution in many space critical applications for board level power distribution. The use of SMD-technology makes it possible to offer a product with high performance at low cost.

Models

Ordercode	Input voltage	Output voltage	Output current max.	Efficiency typ.
TMA 0505S	5 VDC ± 10%	5 VDC	200 mA	71 %
TMA 0512S		12 VDC	80 mA	78 %
TMA 0515S		15 VDC	65 mA	78 %
TMA 0505D		± 5 VDC	±100 mA	72 %
TMA 0512D		±12 VDC	± 40 mA	78 %
TMA 0515D		±15 VDC	±35 mA	79 %
TMA 1205S	12 VDC ± 10%	5 VDC	200 mA	73 %
TMA 1212S		12 VDC	80 mA	80 %
TMA 1215S		15 VDC	65 mA	80 %
TMA 1205D		±5 VDC	±100 mA	74 %
TMA 1212D		±12 VDC	±40 mA	81 %
TMA 1215D		±15 VDC	±35 mA	81 %
TMA 1505S	15 VDC ± 10%	5 VDC	200 mA	73 %
TMA 1512S		12 VDC	80 mA	80 %
TMA 1515S		15 VDC	65 mA	80 %
TMA 1505D		±5 VDC	±100 mA	74 %
TMA 1512D		±12 VDC	±40 mA	81 %
TMA 1515D		±15 VDC	±35 mA	81 %
TMA 2405S	24 VDC ± 10%	5 VDC	200 mA	71 %
TMA 2412S		12 VDC	80 mA	78 %
TMA 2415S		15 VDC	65 mA	79 %
TMA 2405D		±5 VDC	±100 mA	72 %
TMA 2412D		±12 VDC	±40 mA	79 %
TMA 2415D		±15 VDC	±35 mA	80 %

Input Specifications

Input current no load / full load	5 Vin models:	30 mA / 260 mA typ.
	12 Vin models:	12 mA / 110 mA typ.
	15 Vin models:	12 mA / 100 mA typ.
	24 Vin models:	7 mA / 55 mA typ.
Surge voltage (1 sec. max.)	5 Vin models:	9 V max.
	12 Vin models:	18 V max.
	15 Vin models:	21 V max.
	24 Vin models:	30 V max.
Reverse voltage protection		0.3 A max.
Reflected input ripple current		can be reduced by ext. 1–3.3 µF polyester film capacitor
Input filter		internal capacitors

Output Specifications

Voltage set accuracy		±3 %
Voltage balance (dual output models)		±1 % max.
Regulation	– Input variation	±1.2 % / 1 % change Vin
	– Load variation 20 – 100 %	±10 % max.
Ripple and noise (20 MHz Bandwidth)		100 mVp-p typ.
Temperature coefficient		±0.02 %/K
Short circuit protection		limited 1 sec. max.
Capacitive load	– Single output models	220 µF max.
	– Dual output models	100 µF max.

General Specifications

Temperature ranges	– Operating	–40°C to +85°C
	– Case temperature	+95°C max.
	– Storage	–40°C to +105°C
Humidity (non condensing)		95 % rel H max.
Reliability, calculated MTBF (Mil-HDBK-217F, at +25°C, ground benign)		>2'000'000 h
Isolation voltage (input/output)		1'000 VDC
Isolation capacitance (input/output)		60 pF typ.
Isolation resistance (input/output)		>1'000 Mohm
Switching frequency		100 kHz typ. (frequency modulation)
Frequency change over line and load		±30 % max.
Environmental compliance	– Reach	www.tracopower.com/products/reach-declaration.pdf
	– RoHS	RoHS directive 2011/65/EU

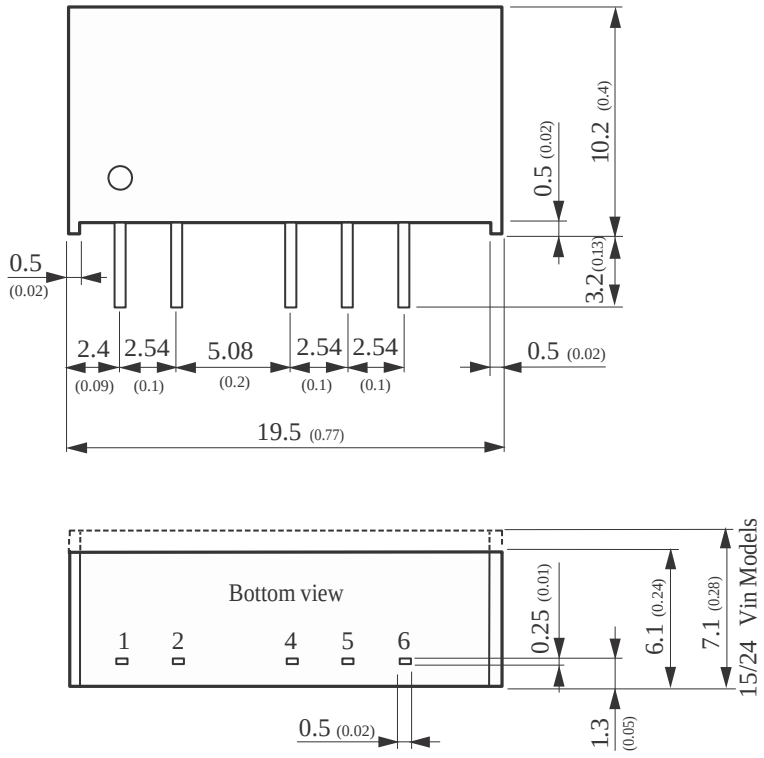
Application note: www.tracopower.com/products/tma-application.pdf

All specifications valid at nominal input voltage, full load and +25°C after warm-up time unless otherwise stated.

Physical Specifications

Casing material		non conductive black plastic (UL 94V-0 rated)
Package weight	Single output models: Dual output models:	2.1 g (0.07 oz) 2.6 g (0.09 oz)
Potting material		Epoxy
Soldering temperature		max. 265°C / 10 sec

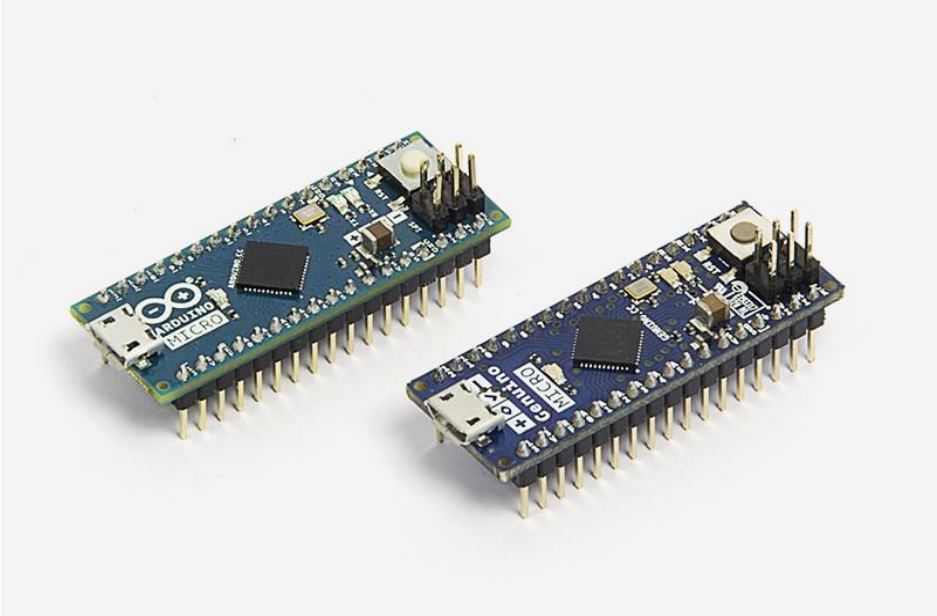
Outline Dimensions mm (inches)



Pin-Out		
Pin	**Single**	**Dual**
1	+Vin (Vcc)	+Vin (Vcc)
2	–Vin (GND)	–Vin (GND)
4	–Vout	–Vout
5	No pin	Common
6	+Vout	+Vout

Tolerances ±0.25 (±0.01)
Pin pitch tolerances ±0.13 (±0.005)
pins ±0.05 (±0.002)

ANEXO 16: HOJAS DE ESPECIFICACIONES DE ARDUINO



Arduino MICRO (USA ONLY)
& Genuino MICRO (OUTSIDE USA)

The Micro is the smallest board of the family, easy to integrate it in everyday objects to make them interactive. The Micro is based on the ATmega32U4 microcontroller featuring a built-in USB which makes the Micro recognisable as a mouse or keyboard.

Overview

5V 8-bit 16 MHz AVR

The Micro is a microcontroller board based on the ATmega32U4 (datasheet (http://www.atmel.com/Images/Atmel-7766-8-bit-AVR-ATmega16U4-32U4_Datasheet.pdf)), developed in conjunction with Adafruit (http://adafruit.com/). It has 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a micro USB cable to get started. It has a form factor that enables it to be easily placed on a breadboard.

The Micro board is similar to the Arduino Leonardo in that the ATmega32U4 has built-in USB communication, eliminating the need for a secondary processor. This allows the Micro to appear to a connected computer as a mouse and keyboard, in addition to a virtual (CDC) serial / COM port. It also has other implications for the behavior of the board; these are detailed on the getting started page (/en/Guide/ArduinoLeonardoMicro).

You can find here (/en/Main/warranty) your board warranty informations.

Getting Started
You can find in the Getting Started section (/en/Guide/HomePage) all the information you need to configure your board, use the Arduino Software (IDE), and start tinker with coding and electronics.

Need Help?

- On the Software on the Arduino Forum (https://forum.arduino.cc/index.php?board=63.0)
- On Projects on the Arduino Forum (https://forum.arduino.cc/index.php?board=3.0)
- On the Product itself through our Customer Support (https://store.arduino.cc/index.php?main_page=contact_us&language=en)

Technical specs

Microcontroller	ATmega32U4 (http://www.atmel.com/Images/Atmel-7766-8-bit-AVR-ATmega16U4-32U4_Datasheet.pdf)
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	20
PWM Channels	7
Analog Input Channels	12
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega32U4) of which 4 KB used by bootloader
SRAM	2.5 KB (ATmega32U4)
EEPROM	1 KB (ATmega32U4)
Clock Speed	16 MHz
Length	48
mm Width	18
mm Weight	13

Documentation

OSH: Schematics, Reference Design, Board size
Arduino / Genuino Micro is open-source hardware! You can build your own board using the follwing files:

EAGLE FILES
IN .ZIP

(/en/uploads/Main/arduino-micro-reference-design.zip)

SCHEMATICS
IN .PDF

(/en/uploads/Main/arduino-micro-schematic.pdf)

BOARD SIZE
IN .DXF

(http://arduino.cc/documents/dimensioni%20Micro.dxf)

Programming
The Micro board can be programmed with the Arduino Software (IDE) (/en/Main/Software). Select "Arduino/Genuino Micro from the Tools > Board menu. For details, see the reference (/en/Reference/HomePage) and tutorials (/en/Tutorial/HomePage).

The ATmega32U4 on the Micro comes preprogrammed with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the AVR109 protocol.

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP (/en/Main/ArduinoISP) or similar; see these instructions for details (/en/Hacking/Programmer).

Warnings

The Micro has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Power

The Micro can be powered via the micro USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from a DC power supply or battery. Leads from a battery or DC power supply can be connected to the Gnd and Vin pins.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- VI. The input voltage to the MICRO board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin.
- 5V. The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- 3V. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND. Ground pins.

Memory

The ATmega32U4 has 32 KB (with 4 KB used for the bootloader). It also has 2.5 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library (en/Reference/EEPROM)).

Input and Output

See the mapping between Arduino pins and ATmega 32U4 ports, and the Pin Mapping of the Arduino Micro:

PIN MAPPING	PIN MAPPING
ATmega 32U4	MICRO

(/en/Hacking/PinMapping32u4)

(/en/uploads/Main/ArduinoMicro_Pinout3.png)

Each of the 20 digital i/o pins on the Micro can be used as an input or output, using pinMode() (/en/Reference/PinMode), digitalWrite() (/en/Reference/DigitalWrite), and digitalRead() (/en/Reference/DigitalRead) functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50 k ohm. A maximum of 40mA is the value that must not be exceeded to avoid permanent damage to the microcontroller.

In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data using the ATmega32U4 hardware serial capability. Note that on the Micro, the Serial class refers to USB (CDC) communication; for TTL serial on pins 0 and 1, use the Serial1 class.
- TWI: 2 (SDA) and 3 (SCL). Support TWI communication using the Wire library (/en/Reference/Wire).
- External Interrupts: 0(RX), 1(TX), 2, 3 and 7. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() (/en/Reference/AttachInterrupt) function for

details.

- PWM: 3, 5, 6, 9, 10, 11 and 13. Provide 8-bit PWM output with the analogWrite() (/en/Reference/AnalogWrite) function.
- SPI: on the ICSP header. These pins support SPI communication using the SPI library (/en/Reference/SPI). Note that the SPI pins are not connected to any of the digital I/O pins as they are on the Uno, they are only available on the ICSP connector and on the nearby pins labelled MISO, MOSI and SCK.
- RX_LED/SS This is an additional pin compared to the Leonardo. It is connected to the RX_LED that indicates the activity of transmission during USB communication, but is can also used as slave select pin (SS) in SPI communication.
- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- Analog Inputs: A0-A5, A6 - A11 (on digital pins 4, 6, 8, 9, 10, and 12). The Micro has a total of 12 analog inputs, pins from A0 to A5 are labelled directly on the pins and the other ones that you can access in code using the constants from A6 trough A11 are shared respectively on digital pins 4, 6, 8, 9, 10, and 12. All of which can also be used as digital I/O. Each analog input provide 10 bits of resolution (i.e. 1024 different values). By default the analog inputs measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() (/en/Reference/AnalogReference) function.

There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analogReference() (/en/Reference/AnalogReference).
- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Micro has a number of facilities for communicating with a computer, another board of the Arduino & Genuino family, or other microcontrollers. The 32U4 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). The ATmega32U4 also allows for serial (CDC) communication over USB and appears as a virtual com port to software on the computer. The chip also acts as a full speed USB 2.0 device, using standard USB COM drivers. On Windows, a .inf file is required (/en/Guide/Windows#toc4) . The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library (/en/Reference/SoftwareSerial) allows for serial communication on other Micro's digital pins.

The ATmega32U4 also supports I2C (TWI) and SPI communication. The Arduino Software (IDE) includes a Wire library to simplify use of the I2C bus; see the documentation (/en/Reference/Wire) for details. For SPI communication, use the SPI library (/en/Reference/SPI).

The Micro appears as a generic keyboard and mouse (/en/Reference/MouseKeyboard), and can be programmed to control these input devices using the Keyboard and Mouse classes.

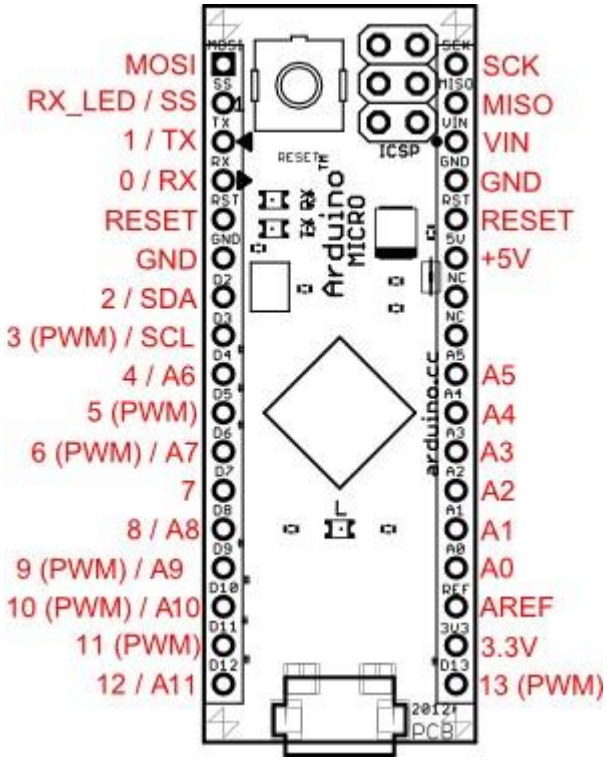
Physical Characteristics

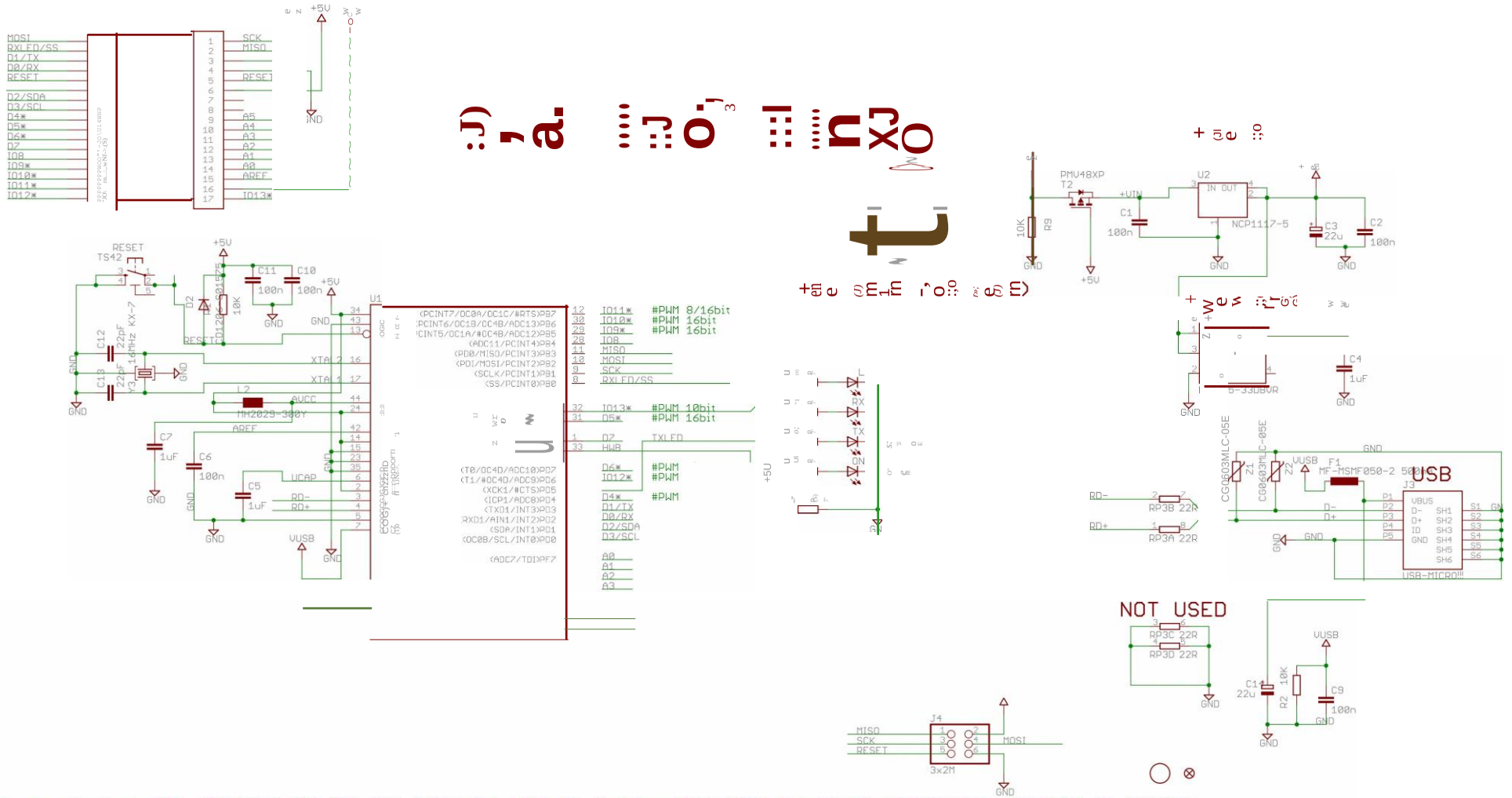
The maximum length and width of the Micro PCB are 4.8cm and 1.77cm respectively, with the USB connector extending beyond the former dimension. The layout allows for easy placement on a solderless breadboard..

Automatic (Software) Reset and Bootloader Initiation

Rather than requiring a physical press of the reset button before an upload, the Micro board is designed in a way that allows it to be reset by software running on a connected computer. The reset is triggered when the Micro's virtual (CDC) serial / COM port is opened at 1200 baud and then closed. When this happens, the processor will reset, breaking the USB connection to the computer (meaning that the virtual serial / COM port will disappear). After the processor resets, the bootloader starts, remaining active for about 8 seconds. The bootloader can also be initiated by pressing the reset button on the Micro. Note that when the board first powers up, it will jump straight to the user sketch, if present, rather than initiating the bootloader.

Because of the way the Micro handles reset it's best to let the Arduino Software (IDE) try to initiate the reset before uploading, especially if you are in the habit of pressing the reset button before uploading on other boards. If the software can't reset the board, you can always start the bootloader by pressing the reset button on the board.





Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS. Arduino DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR Arduino may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this

ARDUINO . trademark.

ANEXO 17: HOJAS DE ESPECIFICACIONES DEL INTEGRADO L293D

L293x Quadruple Half-H Drivers

1 Features

- Wide Supply-Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- Internal ESD Protection
- High-Noise-Immunity Inputs
- Output Current 1 A Per Channel (600 mA for L293D)
- Peak Output Current 2 A Per Channel (1.2 A for L293D)
- Output Clamp Diodes for Inductive Transient Suppression (L293D)

2 Applications

- Stepper Motor Drivers
- DC Motor Drivers
- Latching Relay Drivers

3 Description

The L293 and L293D devices are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, DC and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN.

The L293 and L293D are characterized for operation from 0°C to 70°C.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
L293NE	PDIP (16)	19.80 mm × 6.35 mm
L293DNE	PDIP (16)	19.80 mm × 6.35 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Logic Diagram

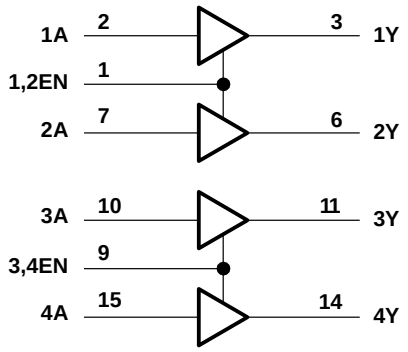


Table of Contents

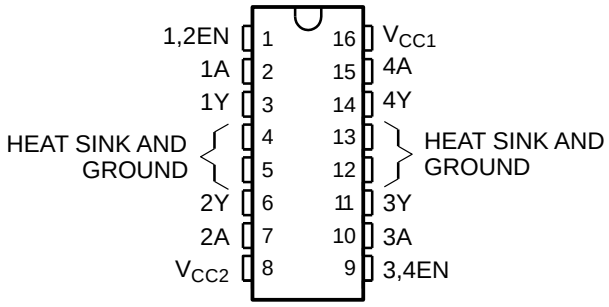
4 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision C (November 2004) to Revision D	Page
• Removed *Ordering Information* table	1
• Added *ESD Ratings* and *Thermal Information* tables, *Feature Description* section, *Device Functional Modes*, *Application and Implementation* section, *Power Supply Recommendations* section, *Layout* section, *Device and Documentation Support* section, and *Mechanical, Packaging, and Orderable Information* section.	1

5 Pin Configuration and Functions

NE Package
16-Pin PDIP
Top View



Pin Functions

PIN		TYPE	DESCRIPTION
NAME	NO.		
1,2EN	1	I	Enable driver channels 1 and 2 (active high input)
<1:4>A	2, 7, 10, 15	I	Driver inputs, noninverting
<1:4>Y	3, 6, 11, 14	O	Driver outputs
3,4EN	9	I	Enable driver channels 3 and 4 (active high input)
GROUND	4, 5, 12, 13	—	Device ground and heat sink pin. Connect to printed-circuit-board ground plane with multiple solid vias
V_{CC1}	16	—	5-V supply for internal logic translation
V_{CC2}	8	—	Power VCC for drivers 4.5 V to 36 V

6 Specifications

6.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)(1)

	MIN	MAX	UNIT
Supply voltage, V_{CC1}(2)		36	V
Output supply voltage, V_{CC2}		36	V
Input voltage, V_I		7	V
Output voltage, V_O	–3	V_{CC2} + 3	V
Peak output current, I_O (nonrepetitive, t ≤ 5 ms): L293	–2	2	A
Peak output current, I_O (nonrepetitive, t ≤ 100 µs): L293D	–1.2	1.2	A
Continuous output current, I_O: L293	–1	1	A
Continuous output current, I_O: L293D	–600	600	mA
Maximum junction temperature, T_J		150	°C
Storage temperature, T_{stg}	–65	150	°C

(1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions*. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) All voltage values are with respect to the network ground terminal.

6.2 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Electrostatic discharge	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001(1)	±2000	V
		Charged-device model (CDM), per JEDEC specification JESD22-C101(2)	±1000	

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.
(2) JEDEC document JEP157 states that 250-V CDM allows safe manufacturing with a standard ESD control process.

6.3 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

			MIN	NOM	MAX	UNIT
	Supply voltage	V_{CC1}	4.5		7	V
		V_{CC2}	V_{CC1}		36	
V_{IH}	High-level input voltage	V_{CC1} ≤ 7 V	2.3		V_{CC1}	V
		V_{CC1} ≥ 7 V	2.3		7	V
V_{IL}	Low-level output voltage		–0.3(1)		1.5	V
T_A	Operating free-air temperature		0		70	°C

(1) The algebraic convention, in which the least positive (most negative) designated minimum, is used in this data sheet for logic voltage levels.

6.4 Thermal Information

THERMAL METRIC(1)		L293, L293D NE (PDIP) 16 PINS	UNIT
$R_{\theta JA}$	Junction-to-ambient thermal resistance (2)	36.4	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance	22.5	°C/W
$R_{\theta JB}$	Junction-to-board thermal resistance	16.5	°C/W
ψ_{JT}	Junction-to-top characterization parameter	7.1	°C/W
ψ_{JB}	Junction-to-board characterization parameter	16.3	°C/W

(1) For more information about traditional and new thermal metrics, see the *Semiconductor and IC Package Thermal Metrics* application report, SPRA953.
(2) The package thermal impedance is calculated in accordance with JESD 51-7.

6.5 Electrical Characteristics

over operating free-air temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage		L293: I_{OH} = −1 A		V_{CC2} – 1.8	V_{CC2} – 1.4		V
			L293D: I_{OH} = − 0.6 A					
V_{OL}	Low-level output voltage		L293: I_{OL} = 1 A			1.2	1.8	V
			L293D: I_{OL} = 0.6 A					
V_{OKH}	High-level output clamp voltage		L293D: I_{OK} = –0.6 A			V_{CC2} + 1.3		V
V_{OKL}	Low-level output clamp voltage		L293D: I_{OK} = 0.6 A			1.3		V
I_{IH}	High-level input current	A	V_I = 7 V			0.2	100	µA
		EN				0.2	10	
I_{IL}	Low-level input current	A	V_I = 0			–3	–10	µA
		EN				–2	–100	
I_{CC1}	Logic supply current		I_O = 0	All outputs at high level		13	22	mA
				All outputs at low level		35	60	
				All outputs at high impedance		8	24	
I_{CC2}	Output supply current		I_O = 0	All outputs at high level		14	24	mA
				All outputs at low level		2	6	
				All outputs at high impedance		2	4	

6.6 Switching Characteristics

over operating free-air temperature range (unless otherwise noted) V_{CC1} = 5 V, V_{CC2} = 24 V, T_A = 25°C

PARAMETER			TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	Propagation delay time, low-to-high-level output from A input	L293NE, L293DNE	C_L = 30 pF, See Figure 2		800		ns
		L293DWP, L293N L293DN			750		
t_{PHL}	Propagation delay time, high-to-low-level output from A input	L293NE, L293DNE			400		ns
		L293DWP, L293N L293DN			200		
t_{TLH}	Transition time, low-to-high-level output	L293NE, L293DNE			300		ns
		L293DWP, L293N L293DN			100		
t_{THL}	Transition time, high-to-low-level output	L293NE, L293DNE			300		ns
		L293DWP, L293N L293DN			350		

6.7 Typical Characteristics

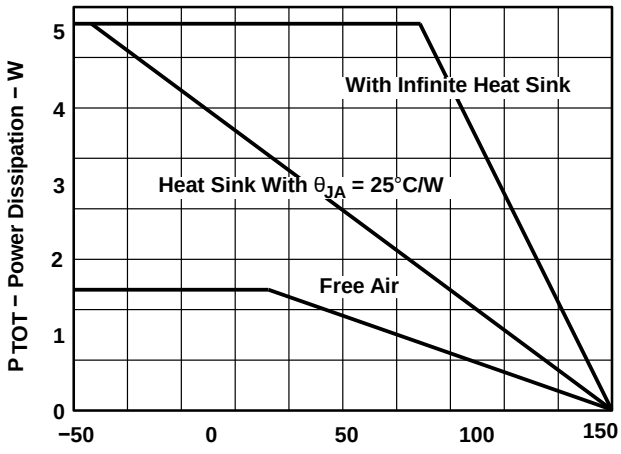
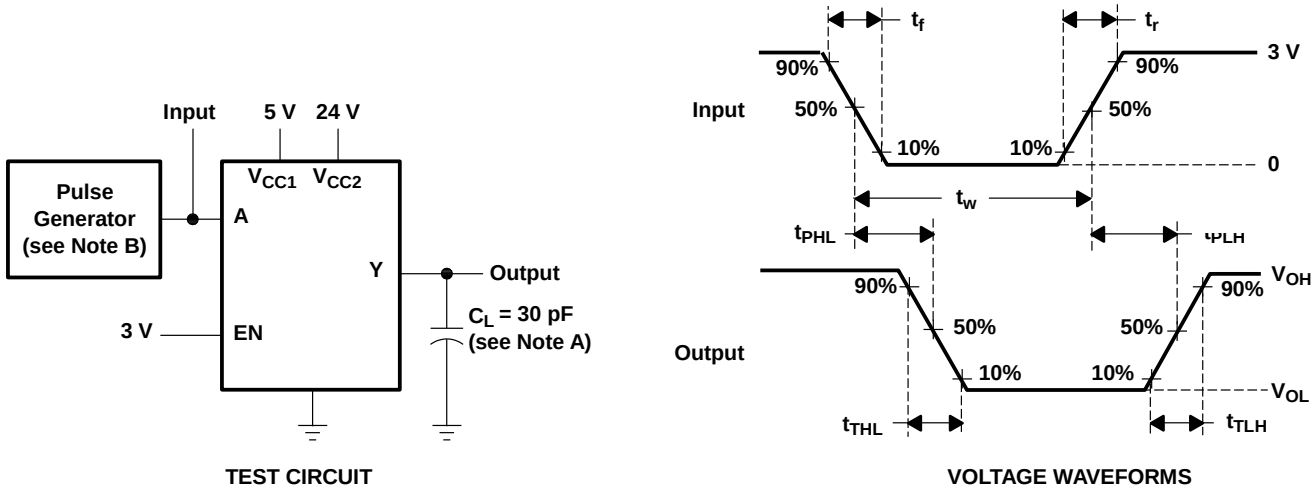


Figure 1. Maximum Power Dissipation vs Ambient Temperature

7 Parameter Measurement Information



TEST CIRCUIT **VOLTAGE WAVEFORMS**

NOTES: A. C_L includes probe and jig capacitance.
B. The pulse generator has the following characteristics: $t_r \leq 10$ ns, $t_f \leq 10$ ns, $t_w = 10$ µs, PRR = 5 kHz, $Z_O = 50$ Ω.

Figure 2. Test Circuit and Voltage Waveforms

8 Detailed Description

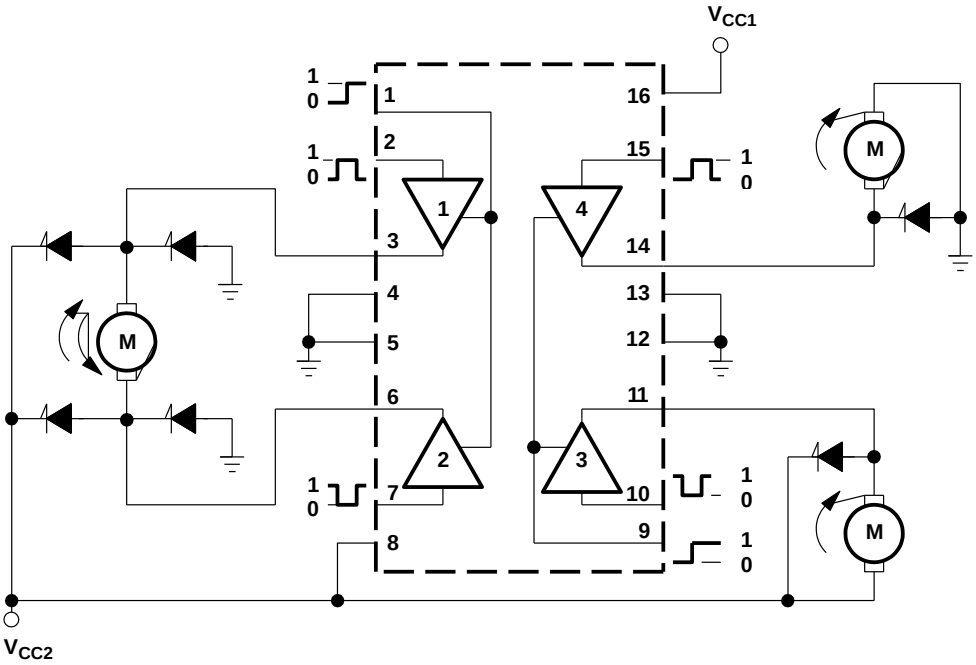
8.1 Overview

The L293 and L293D are quadruple high-current half-H drivers. These devices are designed to drive a wide array of inductive loads such as relays, solenoids, DC and bipolar stepping motors, as well as other high-current and high-voltage loads. All inputs are TTL compatible and tolerant up to 7 V.

Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled, and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

On the L293, external high-speed output clamp diodes should be used for inductive transient suppression. On the L293D, these diodes are integrated to reduce system complexity and overall system size. A V_{CC1} terminal, separate from V_{CC2}, is provided for the logic inputs to minimize device power dissipation. The L293 and L293D are characterized for operation from 0°C to 70°C.

8.2 Functional Block Diagram



Output diodes are internal in L293D.

8.3 Feature Description

The L293x has TTL-compatible inputs and high voltage outputs for inductive load driving. Current outputs can get up to 2 A using the L293.

8.4 Device Functional Modes

Table 1 lists the fuctional modes of the L293x.

Table 1. Function Table (Each Driver)(1)

INPUTS (2)		OUTPUT (Y)
A	EN	
H	H	H
L	H	L
X	L	Z

(1) H = high level, L = low level, X = irrelevant, Z = high impedance (off)
(2) In the thermal shutdown mode, the output is in the high-impedance state, regardless of the input levels.

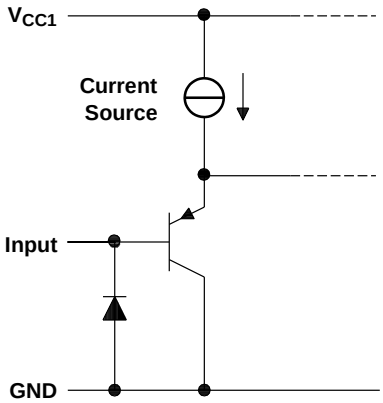


Figure 3. Schematic of Inputs for the L293x

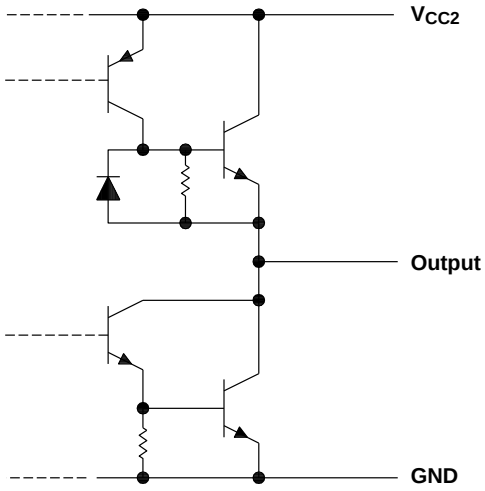


Figure 4. Schematic of Outputs for the L293

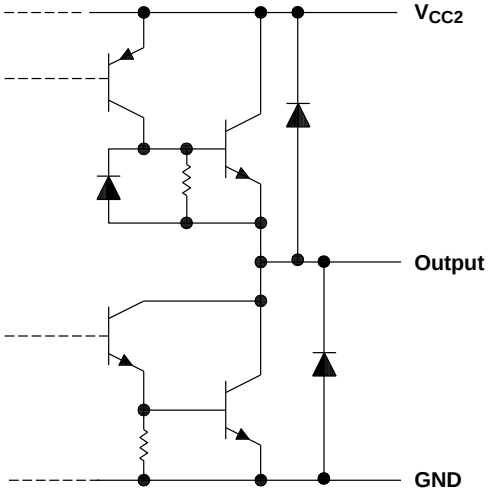


Figure 5. Schematic of Outputs for the L293D

9 Application and Implementation

NOTE

Information in the following applications sections is not part of the TI component specification, and TI does not warrant its accuracy or completeness. TI's customers are responsible for determining suitability of components for their purposes. Customers should validate and test their design implementation to confirm system functionality.

9.1 Application Information

A typical application for the L293 device is driving a two-phase motor. Below is an example schematic displaying how to properly connect a two-phase motor to the L293 device.

Provide a 5-V supply to V_{CC1} and valid logic input levels to data and enable inputs. V_{CC2} must be connected to a power supply capable of supplying the needed current and voltage demand for the loads connected to the outputs.

9.2 Typical Application

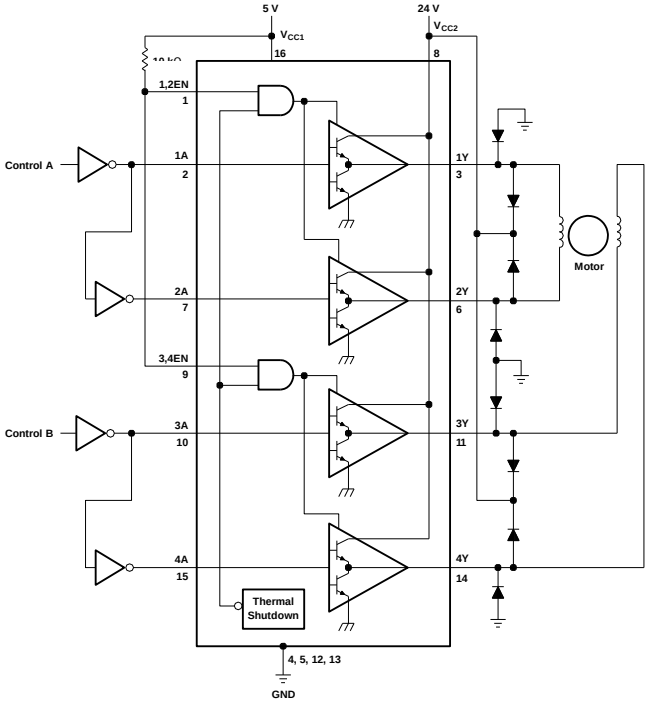


Figure 6. Two-Phase Motor Driver (L293)

9.2.1 Design Requirements

The design techniques in the application above as well as the applications below should fall within the following design requirements.

1. V_{CC1} should fall within the limits described in the *Recommended Operating Conditions*.
2. V_{CC2} should fall within the limits described in the *Recommended Operating Conditions*.
3. The current per channel should not exceed 1 A for the L293 (600mA for the L293D).

9.2.2 Detailed Design Procedure

When designing with the L293 or L293D, careful consideration should be made to ensure the device does not exceed the operating temperature of the device. Proper heatsinking will allow for operation over a larger range of current per channel. Refer to the *Power Supply Recommendations* as well as the *Layout Example*.

Typical Application (continued)

9.2.3 Application Curve

Refer to *Power Supply Recommendations* for additional information with regards to appropriate power dissipation. Figure 7 describes thermal dissipation based on Figure 14.

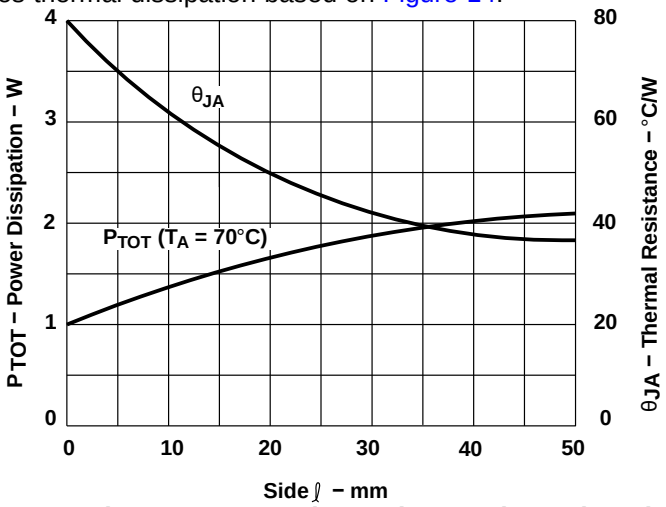


Figure 7. Maximum Power and Junction vs Thermal Resistance

9.3 System Examples

9.3.1 L293D as a Two-Phase Motor Driver

Figure 8 below depicts a typical setup for using the L293D as a two-phase motor driver. Refer to the *Recommended Operating Conditions* when considering the appropriate input high and input low voltage levels to enable each channel of the device.

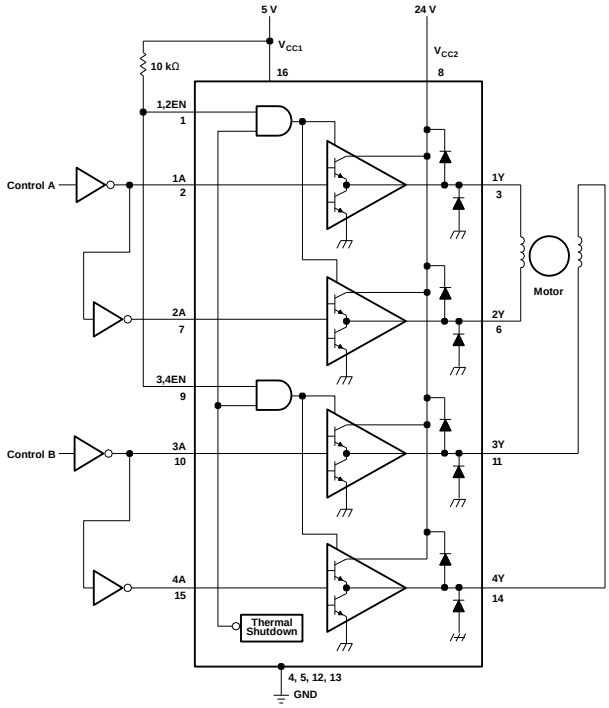
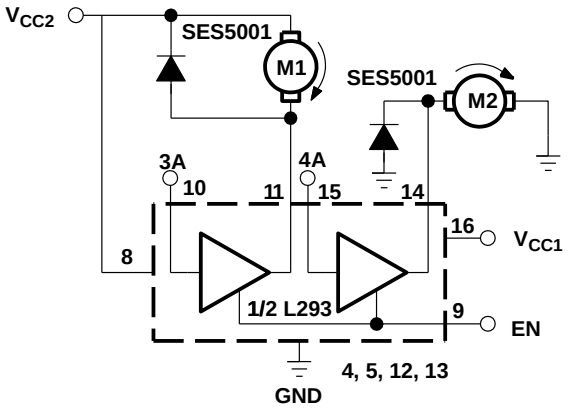


Figure 8. Two-Phase Motor Driver (L293D)

System Examples (continued)

9.3.2 DC Motor Controls

Figure 9 and Figure 10 below depict a typical setup for using the L293 device as a controller for DC motors. Note that the L293 device can be used as a simple driver for a motor to turn on and off in one direction, and can also be used to drive a motor in both directions. Refer to the function tables below to understand unidirectional vs bidirectional motor control. Refer to the *Recommended Operating Conditions* when considering the appropriate input high and input low voltage levels to enable each channel of the device.



Connections to ground and to supply voltage

Figure 9. DC Motor Controls

Table 2. Unidirectional DC Motor Control

EN	3A	M1 (1)	4A	M2
H	H	Fast motor stop	H	Run
H	L	run	L	Fast motor stop
L	X	Free-running motor stop	X	Free-running motor stop

(1) L = low, H = high, X = don't care

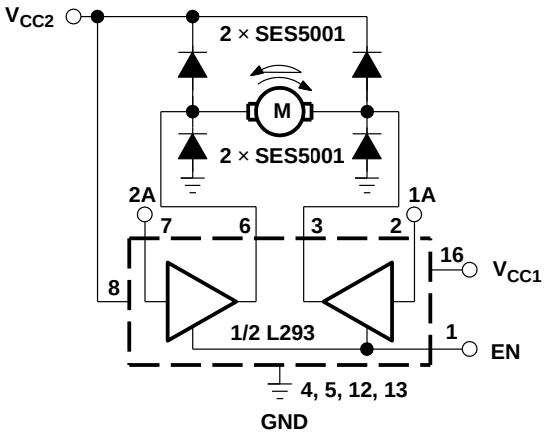


Figure 10. Bidirectional DC Motor Control

Table 3. Bidirectional DC Motor Control

EN	1A	2A	FUNCTION (1)
H	L	H	Turn right
H	H	L	Turn left

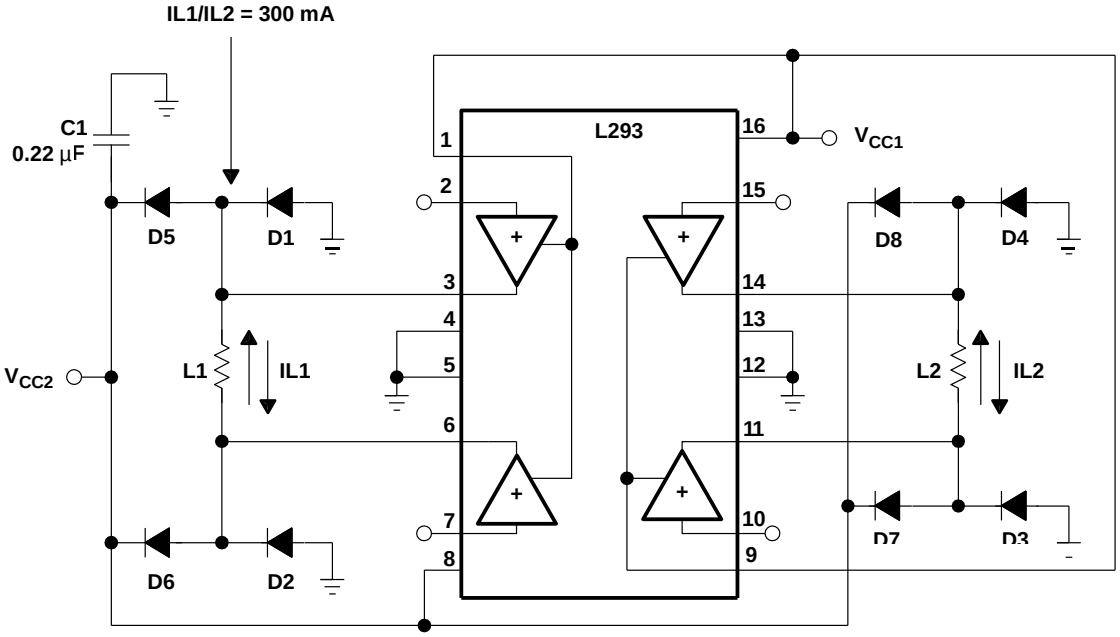
(1) L = low, H = high, X = don't care

Table 3. Bidirectional DC Motor Control (continued)

EN	1A	2A	FUNCTION(1)
H	L	L	Fast motor stop
H	H	H	Fast motor stop
L	X	X	Free-running motor stop

9.3.3 Bipolar Stepping-Motor Control

Figure 11 below depicts a typical setup for using the L293D as a two-phase motor driver. Refer to the *Recommended Operating Conditions* when considering the appropriate input high and input low voltage levels to enable each channel of the device.



D1−D8 = SES5001

Figure 11. Bipolar Stepping-Motor Control

10 Power Supply Recommendations

V_{CC1} is 5 V ± 0.5 V and V_{CC2} can be same supply as V_{CC1} or a higher voltage supply with peak voltage up to 36 V. Bypass capacitors of 0.1 uF or greater should be used at V_{CC1} and V_{CC2} pins. There are no power up or power down supply sequence order requirements.

Properly heatsinking the L293 when driving high-current is critical to design. The Rthj-amp of the L293 can be reduced by soldering the GND pins to a suitable copper area of the printed circuit board or to an external heat sink.

Figure 14 shows the maximum package power PTOT and the θJA as a function of the side of two equal square copper areas having a thickness of 35 µm (see Figure 14). In addition, an external heat sink can be used (see Figure 12).

During soldering, the pin temperature must not exceed 260°C, and the soldering time must not exceed 12 seconds.

The external heatsink or printed circuit copper area must be connected to electrical ground.

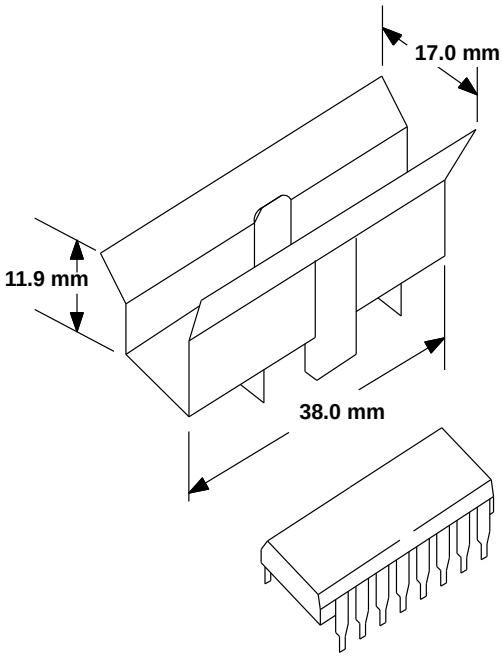


Figure 12. External Heat Sink Mounting Example (θ_{JA} = 25°C/W)

11 Layout

11.1 Layout Guidelines

Place the device near the load to keep output traces short to reduce EMI. Use solid vias to transfer heat from ground pins to ground plane of the printed-circuit-board.

11.2 Layout Example

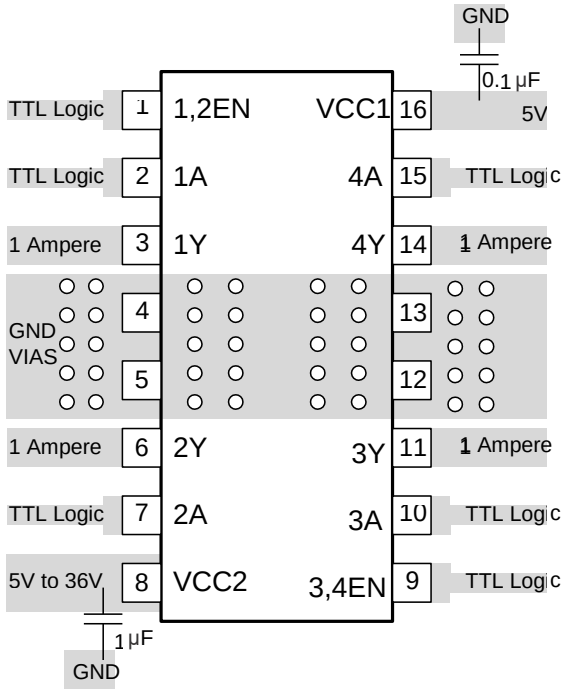


Figure 13. Layout Diagram

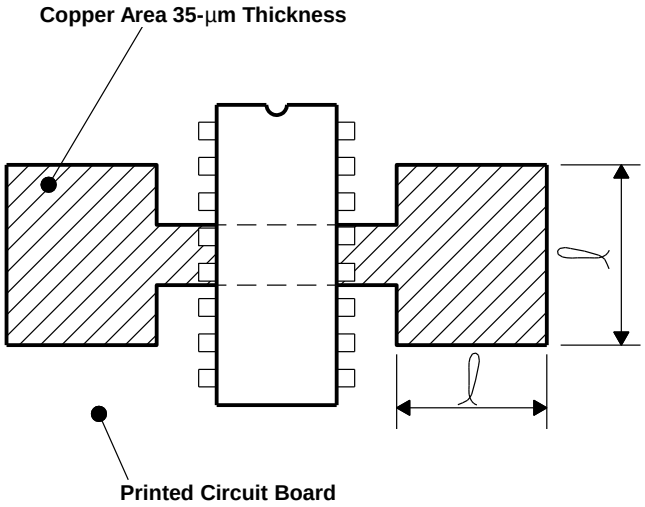


Figure 14. Example of Printed-Circuit-Board Copper Area (Used as Heat Sink)

12 Device and Documentation Support

12.1 Related Links

The table below lists quick access links. Categories include technical documents, support and community resources, tools and software, and quick access to sample or buy.

Table 4. Related Links

PARTS	PRODUCT FOLDER	SAMPLE & BUY	TECHNICAL DOCUMENTS	TOOLS & SOFTWARE	SUPPORT & COMMUNITY
L293	Click here	Click here	Click here	Click here	Click here
L293D	Click here	Click here	Click here	Click here	Click here

12.2 Community Resources

The following links connect to TI community resources. Linked contents are provided "AS IS" by the respective contributors. They do not constitute TI specifications and do not necessarily reflect TI's views; see TI's Terms of Use.

TI E2E™ Online Community ***TI's Engineer-to-Engineer (E2E) Community.*** Created to foster collaboration among engineers. At e2e.ti.com, you can ask questions, share knowledge, explore ideas and help solve problems with fellow engineers.

Design Support ***TI's Design Support*** Quickly find helpful E2E forums along with design support tools and contact information for technical support.

12.3 Trademarks

E2E is a trademark of Texas Instruments.
All other trademarks are the property of their respective owners.

12.4 Electrostatic Discharge Caution

These devices have limited built-in ESD protection. The leads should be shorted together or the device placed in conductive foam during storage or handling to prevent electrostatic damage to the MOS gates.

12.5 Glossary

SLYZ022 — *TI Glossary*.

This glossary lists and explains terms, acronyms, and definitions.

13 Mechanical, Packaging, and Orderable Information

The following pages include mechanical, packaging, and orderable information. This information is the most current data available for the designated devices. This data is subject to change without notice and revision of this document. For browser-based versions of this data sheet, refer to the left-hand navigation.

PACKAGING INFORMATION

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish (6)	MSL Peak Temp (3)	Op Temp (°C)	Device Marking (4/5)	Samples
L293DNE	ACTIVE	PDIP	NE	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type	0 to 70	L293DNE	Samples
L293DNEE4	ACTIVE	PDIP	NE	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type	0 to 70	L293DNE	Samples
L293DWP	OBSOLETE	SOIC	DW	28		TBD	Call TI	Call TI	0 to 70	L293DWP	
L293DWPG4	OBSOLETE	SOIC	DW	28		TBD	Call TI	Call TI	0 to 70		
L293DWPTR	OBSOLETE	SO PowerPAD	DWP	28		TBD	Call TI	Call TI	0 to 70		
L293N	OBSOLETE	PDIP	N	16		TBD	Call TI	Call TI	0 to 70	L293N	
L293NE	ACTIVE	PDIP	NE	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type	0 to 70	L293NE	Samples
L293NEE4	ACTIVE	PDIP	NE	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type	0 to 70	L293NE	Samples
L293NG4	OBSOLETE	PDIP	N	16		TBD	Call TI	Call TI	0 to 70		

(1) The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

(2) Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS), Pb-Free (RoHS Exempt), or Green (RoHS & no Sb/Br) - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.
TBD: The Pb-Free/Green conversion plan has not been defined.
Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.
Pb-Free (RoHS Exempt): This component has a RoHS exemption for either 1) lead-based flip-chip solder bumps used between the die and package, or 2) lead-based die adhesive used between the die and leadframe. The component is otherwise considered Pb-Free (RoHS compatible) as defined above.
Green (RoHS & no Sb/Br): TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

(3) MSL, Peak Temp. - The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

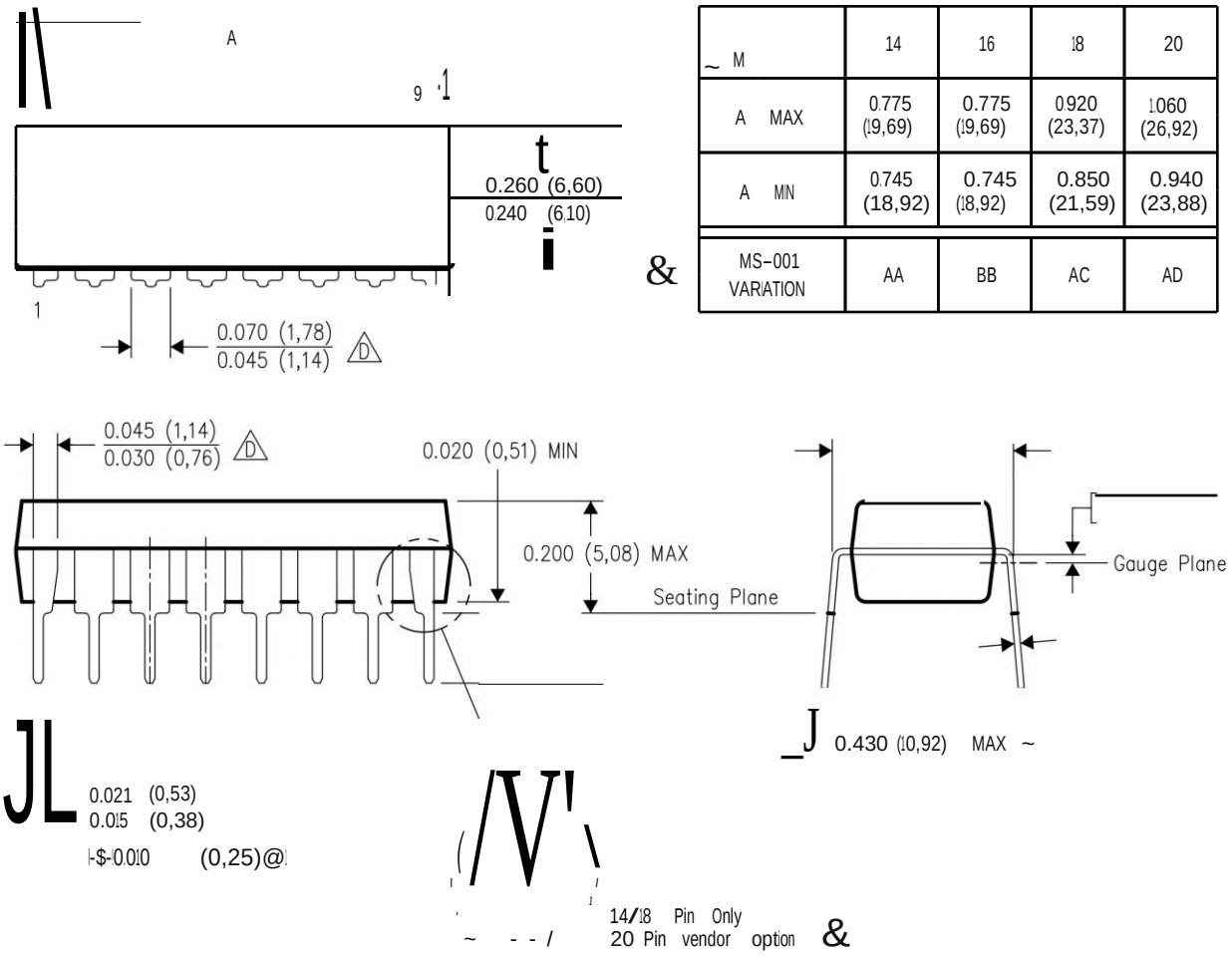
(4) There may be additional marking, which relates to the logo, the lot trace code information, or the environmental category on the device.

(5) Multiple Device Markings will be inside parentheses. Only one Device Marking contained in parentheses and separated by a "~" will appear on a device. If a line is indented then it is a continuation of the previous line and the two combined represent the entire Device Marking for that device.

(6) Lead/Ball Finish - Orderable Devices may have multiple material finish options. Finish options are separated by a vertical ruled line. Lead/Ball Finish values may wrap to two lines if the finish value exceeds the maximum column width.

N (R-PDIP-T**) PLASTIC DUAL-IN-UNE PACKAGE

!6 PINS SHOWN



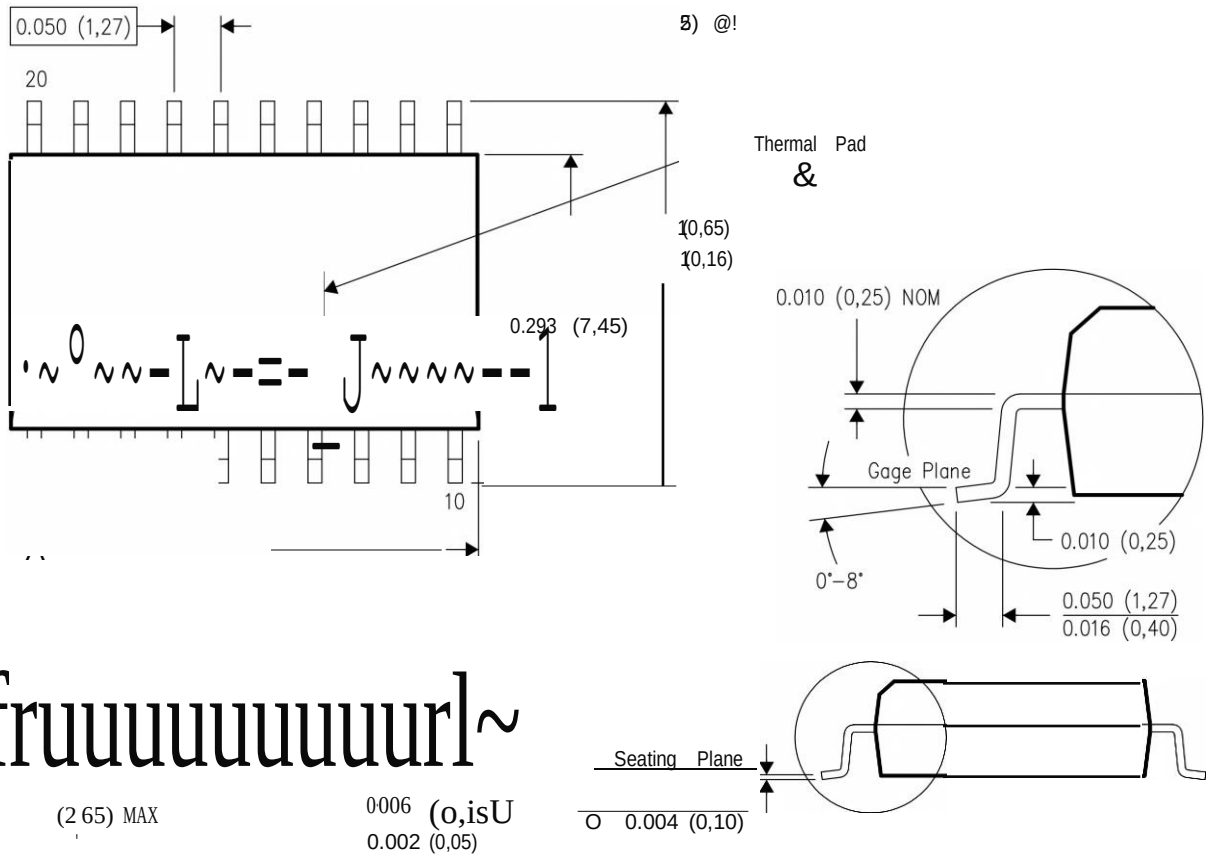
~ M	14	16	!8	20
A MAX	0.775 (!9,69)	0.775 (!9,69)	0.920 (23,37)	!.060 (26,92)
A MIN	0.745 (18,92)	0.745 (!8,92)	0.850 (21,59)	0.940 (23,88)
MS-001 VARIATION	AA	BB	AC	AD

4040049/E !2/2002

NOTES:
A. All linear dimensions are in inches (millimeters).
B. This drawing is sub ject to change without notice.
& Falls within JEDEC MS-001, except !8 and 20 pin minimum body length (Dim A).
& The 20 pin end lead shoulder width is a vendor option, either half or full width.

DWP (R-PDSO-G**) PowerPAD™ PLASTIC SMALL-OUTLINE PACKAGE

20 PINS SHOWN



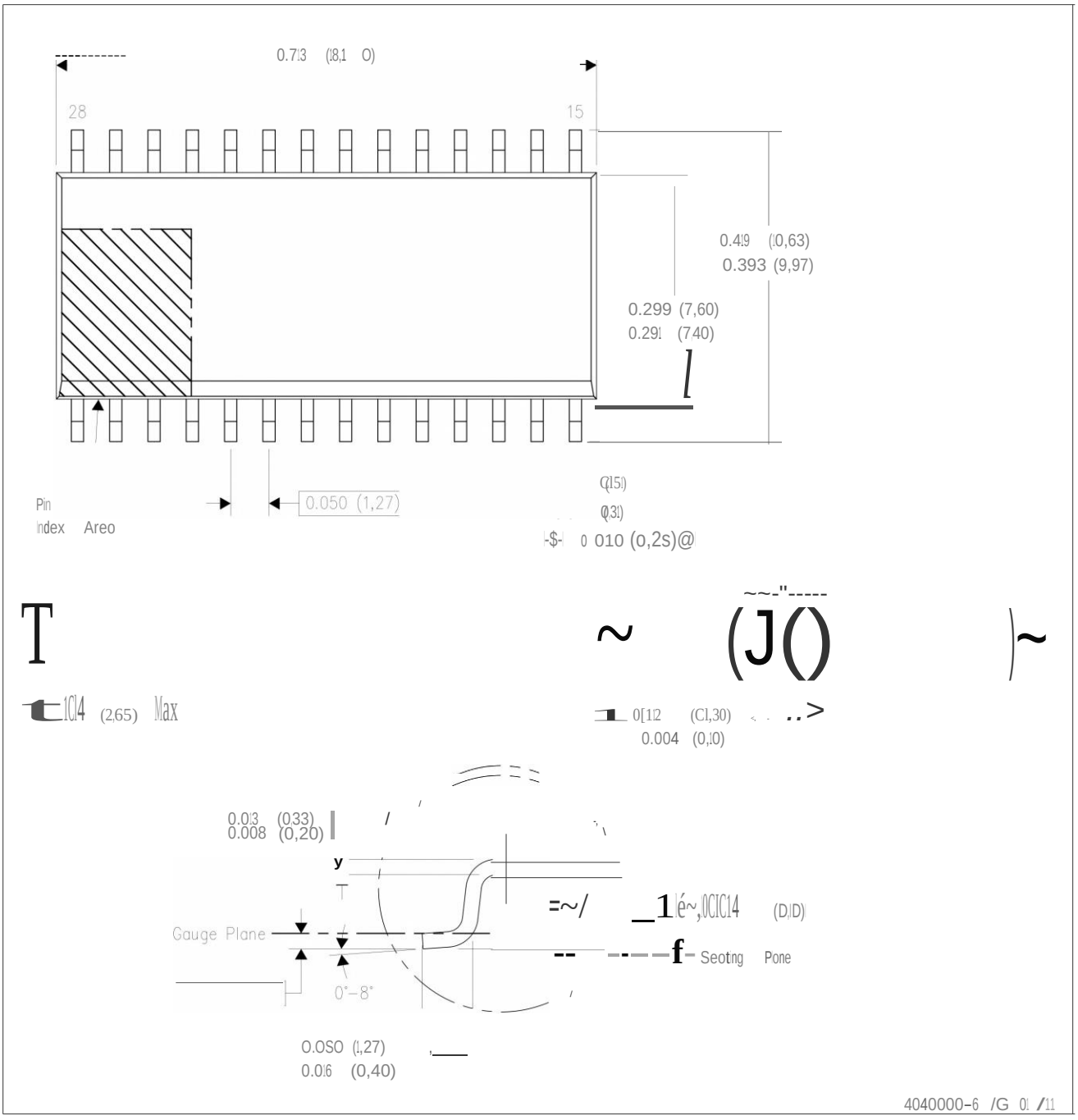
~ M	16	20	24	28
A MAX	0.410 (10,41)	0.510 (12,95)	0.610 (15,49)	0.710 (18,03)
A MIN	0.400 (10,16)	0.500 (12,70)	0.600 (15,24)	0.700 (17,78)

4147575/C 02/05

NOTES:
- A. All linear dimensions are in inches (millimeters).
- B. This drawing is subject to change without notice.
- C. Body dimensions do not include mold flash or protrusion not to exceed 0.006 (0,15).
- & This package is designed to be soldered to a thermal pad on the board. Refer to Technical Brief, PowerPad Thermally Enhanced Package, Texas Instruments Literature No. SLMA002 for information regarding recommended board layout. This document is available at www.ti.com <http://www.ti.com>. See the product data sheet for details regarding the exposed thermal pad dimensions.

PowerPAD is a trademark of Texas Instruments.

0W (R-POSO-G28) PLASTIC SMALL OUTLINE



NOTES:
A. All linear dimensions are in inches (millimeters). Dimensioning and tolerancing per ASME Y14.5M-1994.
B. This drawing is subject to change without notice.
C. Body dimensions do not include mold flash or protrusion not to exceed 0.006 (0,15).
D. Falls within JEDEC MS-013 variation AE.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265