UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECIFICAS DE TELECOMUNICACIÓN

Prototipado y programación de la electrónica de una prótesis de mano

Autor:

D. Pablo Benito González

Tutor/es:

Dr. D. Alberto Mansilla Gallo

Dr. D. Mario Martínez Zarzuela

Valladolid, Junio de 2024

TÍTULO:	**Prototipado y programación de la electrónica de una prótesis de mano**
AUTOR:	**D. Pablo Benito González**
TUTOR:	**Dr. D. Alberto Mansilla Gallo**
	Dr. D. Mario Martínez Zarzuela
DEPARTAMENTO:	**Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**Dr. D. Javier Manuel Aguiar Pérez**
VOCAL:	**Dra. Dª. Míriam Antón Rodríguez**
SECRETARIO:	**Dr. D. Mario Martínez Zarzuela**
SUPLENTE 1:	**Dr. D. David González Ortega**
SUPLENTE 2:	**Dr. D. Carlos Gómez Peña**

FECHA:	**Junio 2024**
CALIFICACIÓN:	

Agradecimientos

Me gustaría agradecer a Alberto Mansilla Gallo y a Mario Martínez Zarzuela por proporcionarme la oportunidad de desarrollar y presentar este proyecto.

También me gustaría agradecer a mis compañeros, amigos y familia por la ayuda y el apoyo que me han proporcionado durante estos años de carrera.

RESUMEN

Las prótesis pediátricas plantean un delicado problema a la hora de la adquisición y uso por sus principales receptores, los niños, pues la combinación de su alto coste económico junto a su corta vida útil hace que sean difíciles de adquirir y sustituir. El precio de una prótesis mioeléctrica para un niño puede variar entre los 8.000 y los 15.000 €, con una vida útil que rara vez supera los dos o tres años ya que, pasado este tiempo, el niño ha crecido lo suficiente como para no poder utilizar o continuar con su uso. Si a estos factores se une el coste del mantenimiento periódico que suele necesitar la prótesis y lo propensos que son los niños a actividades tendentes a generar accidentes que pueden comprometer la integridad de esta, hace que las familias de niños con algún tipo de amputación se ven en la necesidad de desembolsar grandes cantidades de dinero, de manera periódica, para permitir que sus hijos tengan acceso a herramientas que les permitan tener una infancia plena.

La aparición de nuevas tecnologías, como la impresión 3D, ha permitido producir piezas complejas a nivel local por precios asequibles. Por otro lado, el creciente mercado de microcontroladores de propósito general con precios asequibles, como pueden ser los de marcas como Arduino y Raspberry, ha permitido y motivado el desarrollo de este proyecto, encaminado a la creación de una prótesis de mano, por impresión 3D, que permita dar una solución más económica que las opciones disponibles actualmente en el mercado.

Se va a trabajar a partir de un primer prototipo desarrollado en otro TFG El propósito de este trabajo es diseñar una circuitería y sistema de control que permita hacer la prótesis portátil y facilite manejarla de manera satisfactoria, todo ello manteniendo el espíritu del proyecto original, cuyo objetivo es el de crear una alternativa más ventajosa económicamente para los potenciales usuarios, frente a otras opciones de mercado, las cuales cuentan con un precio de entrada elevado y mantenimientos caros.

Palabras clave

Prótesis, Prótesis de mano, Prótesis mioeléctrica, EMG, EMG kinesiológico, Impresión 3D, Circuitos impresos.

ABSTRACT

Pediatric prothesis pose a delicate problem when we talk about their acquisition and use by their main receptors, the children, the high economic cost, and the short useful life this kind of products have make for a product that is hard to acquire and maintain. The price of a kid's prothesis is usually between the 8.000 € and 15.000 €, with a span of usually 2 or 3 years for its use, after this time the children have usually grown enough and the prothesis does not fit anymore. Combine these factors with the usual maintenance this kind of product needs and the tendency kids have to partake in activities that can be damaging to delicate electronic and mechanic components and you will come to the conclusion that the families of children with some kind of amputation have to pay out a considerable amount of money in a regular basis, just so their kids have access to a tool that lets them have a good childhood.

The rise of new technologies, like 3d printing, allows for complex pieces to be made locally and for a low price. Also, the growing microcontroller market now offers cheap all-purpose microcontrollers like the ones from Arduino and Raspberry. These factors have allowed and motivated this project, where a 3D printed prothesis is being developed to be a cheap option when compared to the options of the actual market.

All work will be done using as base a prototype developed in a previous TFG. In this project, I will be designing the circuits and control systems necessary for the prothesis, the objective is to make the prothesis portable and easy to use, all this will be made while considering the vision of the original project, making an alternative to the prothesis in the actual market, which have a high initial cost and expensive upkeep.

Keywords

Prothesis, Hand prothesis, Myoelectric prothesis, EMG, Kinesiologic EMG, 3D printing, PCB.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

1.1 MOTIVACIÓN

Este Trabajo de Fin de Grado (TFG) viene a continuar con un proyecto que empezó a desarrollar Carlos Calle del Río[3], en su TFG defendido públicamente en 2023 en el Grado en Ingeniería en Electrónica Industrial y Automática, y como tal, la motivación que ha llevado a la continuación de este está fuertemente condicionada con la que inspiró el proyecto de origen.

Las últimas décadas han sido la edad dorada de la miniaturización en la electrónica. La capacidad de computación de los procesadores lleva aumentando de manera exponencial décadas y el tamaño que ocupan se ha reducido a un ritmo similar.

Este rápido desarrollo en el mundo de la electrónica ha permitido invenciones sin precedentes como el smartphone y ha facilitado que industrias, como la automovilística, agilicen su cadena de producción y mejoren la calidad general de sus productos.

Una ciencia que se ha visto claramente beneficiada por esta miniaturización de la electrónica ha sido la medicina. Al ser una ciencia donde, en muchos casos, se necesita de medidas precisas y trabajo en espacios reducidos, no es de extrañar que el avance de la electrónica haya permitido mejorar la calidad de ciertos procedimientos, así como permitir otros que antes se hubiesen considerado imposibles.

La ortopedia, y más en concreto la creación de prótesis, se ha visto fuertemente beneficiada por estos avances. El objetivo final de una prótesis es sustituir una parte del cuerpo que ha sido perdida por una razón u otra. El problema, a la hora de desarrollar estas herramientas, es que el ser humano es increíblemente complejo, por lo que crear una réplica adecuada de una de sus partes suele ser un gran desafío. La electrónica moderna ha permitido crear prótesis con movimientos más complejos y precisos, al ser capaz de integrar en un espacio reducido, como puede ser el equivalente a un brazo o una pierna, un componente lógico complejo que permite identificar múltiples señales y responder de distintas formas, en función de la información recibida.

Al hablar de estos reemplazos, es fundamental tener en cuenta el coste de su creación y desarrollo. Una prótesis de mano puede costar, en función de su gama, entre 15.000 y 100.000 € [18]. Este rango de precios varía en función del grado de complejidad de la ingeniería que hay detrás de la prótesis. Su baja demanda, así como la naturaleza del cuerpo humano, hace necesaria la personalización de cada producto, lo que suele terminar aumentando todavía más el precio.

Debido a su precio, una prótesis suele ser una gran inversión, que se puede ver como un mal necesario, como la adquisición de un coche o una casa, una inversión esencial para la vida diaria que se prevé que se va a amortizar en un largo periodo de tiempo.

El problema del coste de la prótesis se agrava aún más cuando se entra en el ámbito pediátrico, no solo porque la oferta disponible es muy reducida, sino porque además existe

el problema del rápido crecimiento de los niños. Si bien las prótesis pediátricas tienden a ser menos costosas que las de los adultos (alrededor de los 8.000 o 15.000 €), el crecimiento del niño implica que la prótesis deba ser renovada cada pocos años.

La motivación principal de este proyecto es crear una prótesis de bajo coste que, sin ser necesariamente mejor que otros productos en el mercado, permita realizar movimientos de la mano sin que sea necesaria una inversión tan grande como la que requieren hoy en día la mayoría de las opciones que existen a la venta.

1.2 OBJETIVOS Y FASES

1.2.1 Objetivos globales

El objetivo global de este proyecto es crear un segundo prototipo de la prótesis de bajo coste, que, a diferencia del primero, debe poderse separar del banco de trabajo, es decir, debe ser portátil, y además permitir un control fino del movimiento en pinza que permite la mano. El primer prototipo solo permitía realizar rotaciones completas, abriendo y cerrando la mano del todo. Este segundo prototipo debe dejar mover la pinza con más libertad.

Estos objetivos deben de ser cumplidos sin comprometer los del prototipo original, es decir, mantener su fuerza de agarre teniendo en cuenta que lo que se está diseñando es una opción más económica que el resto de la oferta de mercado.

1.2.2 Fases del desarrollo

Crear una placa de circuito impreso (PCB) para conectar de manera cómoda los componentes.

En el primer prototipo los componentes del circuito se encuentran sobre un par de placas de pruebas, lo que no es adecuado para una aplicación que se va a transportar y mover de manera habitual. Para solucionar esto, todos los componentes del circuito se van a integrar en una PCB. El objetivo no es diseñar la PCB final, sino un prototipo que sea cómodo de transportar y que no sea tan compacto que dificulte las diferentes pruebas que se van a realizar sobre los componentes del circuito.

Programación del controlador del circuito.

Igualmente, en el primer prototipo, el código original del controlador del circuito es muy limitado, y solo indica al controlador del motor que abra o cierre la mano cuando el estado de los sensores cambia. Se espera mejorar este código para que procese la señal de manera más acertada. El objetivo es tener un código capaz de interpretar la señal procedente de los sensores, mandando las ordenes de movimiento hacia el motor con cierta intencionalidad. Este segundo prototipo, deberá filtrar errores y bloquear el movimiento cuando se detecten señales de valores no esperados.

Programación del controlador del motor.

El motor del primer prototipo cuenta con un controlador programable que necesita utilizar un software particular del fabricante. El pseudocódigo que se utiliza para programar este controlador debe actualizarse, ya que en su versión inicial solo permite que la mano se abra o se cierre completamente.

Pruebas con nuevos sensores EMG.
Los sensores que se utilizaban en el primer prototipo de la prótesis chocaban con la premisa de que la prótesis deba ser una opción barata y asequible. Por esta razón, se van a realizar pruebas con unos sensores de gama más baja, más acertados para la dirección de este proyecto. Estos sensores no van a poder ofrecer el nivel de calidad de los sensores originales, por lo que habrá que adaptar el resto de los componentes a estos sensores menos precisos.

Realizar pruebas con una batería externa.
Para simular una situación realista, donde esta prótesis es utilizada por un paciente, se van a realizar pruebas con una fuente de alimentación independiente que se pueda transportar con el resto del circuito. Esto va a permitir comprobar el espacio aproximado que va a ocupar una batería y el tipo de problemas que este tipo de fuente de alimentación puede acarrear.

1.3 ESTRUCTURA DE LA MEMORIA
Esta memoria se ha estructurado en una serie de capítulos que pretenden exponer los distintos pasos que se han dado para la realización de este proyecto.

- El primer capítulo expone las motivaciones y objetivos del proyecto.
- El segundo afianza la base teórica de algunos de los conceptos que se van a tratar en esta memoria. Estos conocimientos hacen que resulte más sencillo comprender la motivación que existe detrás de las decisiones que se han tomado, y que se exponen en capítulos posteriores.
- El tercer capítulo cubre el estado inicial del proyecto, esto permite diferenciar adecuadamente el realizado en este TFG, de versiones anteriores y poder así explicar el porqué de algunas de las decisiones tomadas en su desarrollo.
- El cuarto capitulo cubre la exposición del proyecto, se van a explicar en detalle las acciones que se han realizado para cumplir los objetivos marcados en el primer capítulo de esta memoria.
- El quinto capítulo expone los resultados obtenidos, junto con algunas pruebas realizadas para demostrar que los objetivos planteados han sido cumplidos.
- El sexto y último capítulo expone las conclusiones obtenidas y plantea futuras líneas de trabajo por las que podría continuar este proyecto.

2 PRÓTESIS, SUS MÉTODOS DE CONTROL Y EMG

En este capítulo se van a contextualizar algunos de los conceptos más importantes de los que se va a hablar en este trabajo. Se va a exponer un resumen de la historia de las prótesis y de sus métodos de control, también se va a hablar de la electromiografía, que es la técnica más moderna utilizada para el control de prótesis, y la que se ha utilizado en este trabajo.

2.1 HISTORIA DE LAS PRÓTESIS

A lo largo de la historia el desarrollo de la protésica ha ido fuertemente ligado al de la medicina y a las grandes guerras. Para comprender este proceso, se estudiará el papel y complejidad de las prótesis en distintos periodos históricos. Es importante tener en cuenta que la protésica no ha sido considerada una disciplina técnica propia hasta después de las grandes guerras del siglo XX, cuando el número de amputados aumentó de forma considerable. Antes de esta época las prótesis eran producidas por artesanos de todo tipo cuando la necesidad se presentaba. [13]

2.1.1 Prehistoria

La falta de historia escrita de esta época y del limitado desarrollo tecnológico hace que en este periodo sea más relevante hablar de la amputación que de la protésica como tal.

Las primeras evidencias de amputados conviviendo en sociedades humanas tienen alrededor de 40.000 años de antigüedad. Un esqueleto actualmente exhibido en el Instituto Smithsoniano (USA) datado alrededor del 40.000 AC cuenta con indicios en la forma de sus dientes que, según los antropólogos, indican una amputación de miembro superior. Por otro lado, pinturas rupestres que se estima fueron creadas alrededor del 36.000 AC encontradas en Francia y España (figura 1), representan con claridad manos de personas con amputaciones en sus dedos.

Figura 1. Pinturas rupestres de manos a las que le faltan dedos [14](McCauley et al., 2018).

El primer registro histórico existente en el que se hable sobre amputaciones y prótesis es un poema sánscrito de la India llamado RIG-VEDA, del guerrero Queen Vispla aprox. 3800-1600 AC. El poema habla de un guerrero que pierde su pierna en batalla y se fabrica una prótesis metálica para volver a batallar [13][14].

2.1.2 Edad Antigua

En la época de las grandes civilizaciones antiguas de occidente empiezan a aparecer registros escritos de las primeras prótesis. Solían ser necesarias como consecuencia de gangrenas, la lepra o infecciones, aunque también existen casos donde se realizaban amputaciones por razones religiosas.

Se han encontrado momias de esta época que fueron enterradas con prótesis con señales de haber sido utilizadas en vida [13].

2.1.3 Edad Media

La Edad Media es conocida por su mediocre desarrollo científico. La mayoría de las prótesis que se encuentran de esta época son las viejas "peg legs" o patas de palo y garfios simples para los miembros superiores. Las más destacables de eran las realizadas por herreros. Cuando un caballero que había perdido alguna parte del cuerpo solicitaba una armadura, los herreros hacían piezas modificadas para los miembros amputados que les facilitaba cabalgar y sujetar espadas o escudos. Si bien estas prótesis no tenían un uso fuera del campo de batalla, permitían a los soldados esconder la debilidad que suponía la falta de un miembro [13].

2.1.4 Renacimiento

El Renacimiento se caracterizó por el resurgir de la ciencia, esto supuso la creación de algunas prótesis complejas. Existen varios casos aislados de prótesis muy complejas desarrolladas en esta época, la más prominente es la que utilizó Götz Berlichingen, considerada la primera prótesis funcional de la historia (figura 2), que era lo suficientemente compleja como para permitirle cabalgar, empuñar una espada e incluso escribir. Esta mano, que fue creada alrededor del 1504, era tan elaborada que se necesitaron cuatro siglos para que volviese a aparecer una prótesis con un nivel de funcionalidad similar. No sería hasta el siglo XX cuando el doctor Sauerbruch desarrolló una prótesis del mismo estilo, basándose en los registros que existían de esa vieja mano [7][13].

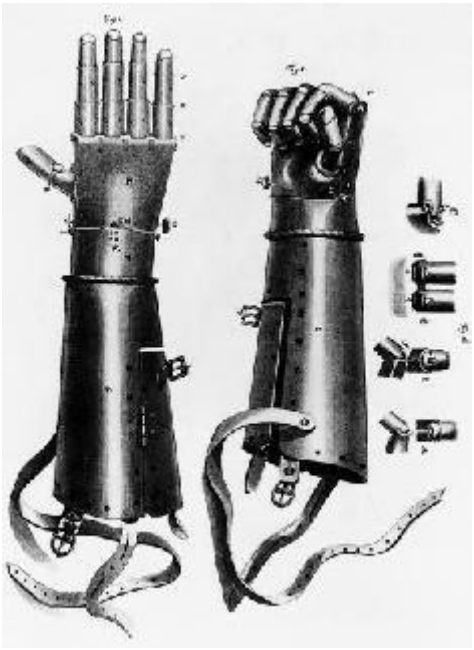


*Figura 2. Prótesis utilizada por Götz von Berlichiengen [7](*Historias con Historia » Blog Archive » Götz 'Mano de Hierro' Berlichingen*, 2010).*

2.1.5 Edad contemporánea

La protésica se establece como una disciplina propia en esta época. Después de la revolución industrial se suceden una serie de conflictos que dejan una gran cantidad de amputados en muchos países de occidente. La guerra civil americana y las dos guerras mundiales suponen la aparición de una cantidad de amputados en los países partícipes, sin precedentes hasta la fecha. La necesidad de crear prótesis para todas estas personas supone la apertura de clínicas especializadas, normalmente regentadas por amputados que, en la búsqueda de darle una solución a su problema dan con una oportunidad de negocio a largo plazo.

Estas nuevas clínicas impulsan el desarrollo de la protésica como ciencia, lo que capta el interés de varias instituciones académicas que comienzan a invertir en proyectos relacionados con el mundo de las prótesis. Esta serie de eventos culmina a mediados del siglo XX, cuando aparecen los primeros cursos especializados en protésica en universidades americanas y europeas y se comercializan las primeras prótesis mioeléctricas por la firma Ottobock [13].

2.2 TIPOS DE PRÓTESIS

Las prótesis se pueden clasificar en distintos grupos en función de la tecnología utilizada para crearlas y de la cantidad de movilidad que permiten recuperar al usuario. En este apartado se muestran los principales tipos de prótesis que hay en el mercado. Algunos de estos tipos no son mutuamente exclusivos.

2.2.1 Prótesis estéticas

Las prótesis estéticas, también llamadas prótesis pasivas, no tienen movimiento y solo pretenden cubrir el aspecto estético del miembro faltante. Se suelen hacer de materiales

muy ligeros para que sean cómodas de utilizar. La ausencia de partes móviles hace que estas prótesis apenas necesiten mantenimiento [2].

2.2.2 Prótesis mecánicas

Las prótesis mecánicas están pensadas para cumplir con funciones muy básicas, normalmente controlando el movimiento a través de otra articulación. En el caso de las prótesis de mano, suelen ser garfios o pinzas que permiten agarrar objetos grandes usando movimientos del codo o del hombro [2].

2.2.3 Prótesis eléctricas

Las prótesis eléctricas cuentan con un motor eléctrico que puede ser controlado con algún tipo de servo-control o interruptor. Tienden a ser costosas y pesadas, pero a cambio permiten movimientos rápidos y que proporcionan mucha fuerza [2].

2.2.4 Prótesis neumáticas

Las prótesis neumáticas utilizan aire a presión para lograr su movimiento, cuentan con gran fuerza y rapidez de movimiento, pero los dispositivos necesarios para su movimiento y control suelen ser voluminosos y pesados con un mantenimiento costoso y difícil de realizar. Apenas existen prótesis de mano neumáticas debido a las características de estas, por otro lado, estas prótesis se usan a menudo para sustituir la articulación de la rodilla [2].

2.2.5 Prótesis mioeléctricas

Las prótesis mioeléctricas son de las más utilizadas en el mundo, especialmente cuando hablamos de prótesis de mano. Este tipo de prótesis viene definido por el tipo de sensores que utiliza para detectar las señales de los músculos, son los llamados sensores mioeléctricos o sensores EMG. Las prótesis mioeléctricas pueden ser eléctricas o neumáticas [2].

2.2.6 Prótesis híbridas

Las prótesis hibridas combinan métodos anteriores para lograr el movimiento, la más común, lo que justifica su inclusión de este apartado, es la prótesis que combina la forma de funcionamiento de las prótesis mecánicas y mioeléctricas, en la figura 3 se puede ver un esquema sencillo de cómo se estructuran estas prótesis.

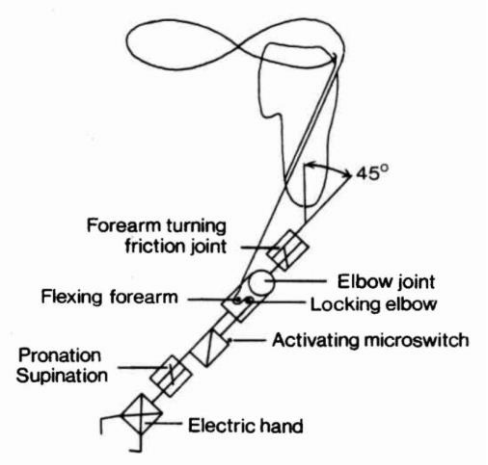


Figura 3. Esquema de las partes de una prótesis hibrida [15](Ober, 1982).

Estas prótesis se usan para amputaciones por encima del codo. El movimiento mecánico permite mover el codo, mientras que algunos sensores EMG permiten algún movimiento simple de la mano, como el de pinza, en la figura 4 se puede ver una persona usando este tipo de prótesis [2][15][19].

Figura 4. Persona utilizando una prótesis hibrida [19](Trent et al., 2019).

2.3 SENSORES EMG

La electromiografía o EMG es la técnica experimental en la que se tratan y analizan las señales mioeléctricas. Estas señales están formadas por variaciones fisiológicas en el estado de las fibras de los músculos [10].

En este apartado se va a hablar de las señales mioeléctricas en lo que a las prótesis mioeléctricas se refiere. Esto significa que cuando hablemos de EMG no vamos a hablar de EMG neurológico, proceso electromiográfico más conocido, en el que se provoca un estímulo eléctrico artificial en el músculo mediante un electrodo. En su lugar hablaremos de EMG kinesiológico, que estudia la activación muscular voluntaria [10].

Para entender adecuadamente el funcionamiento de los sensores EMG y las señales mioeléctricas, lo primero que hay que preguntarse es qué se está midiendo.

Si se estudian las membranas musculares humanas, en su funcionamiento se puede observar un fenómeno curioso. En los músculos existe una diferencia de potencial eléctrico entre las membranas exteriores e interiores del músculo, esta diferencia suele ser de unos -80 mV tomando la zona interna del músculo como referencia. La excitación del músculo desencadena un proceso llamado despolarización, que invierte la polaridad de la diferencia de potencial eléctrico a unos 30 mV [10].

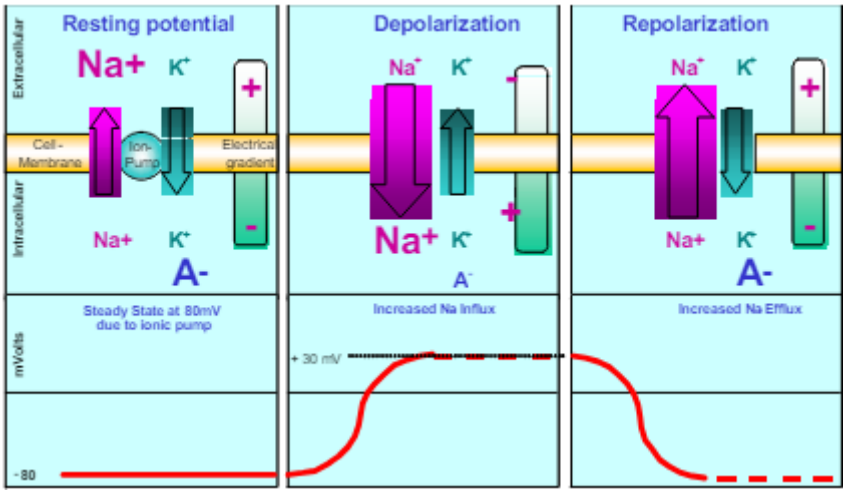


Figura 5. Esquema del proceso que invierte la polaridad del potencial eléctrico en las fibras de un músculo [10](Konrad, 2006).

El fenómeno de despolarización no ocurre al mismo tiempo en todo el músculo, en la literatura es descrito como una zona de entre 1 y 3 mm^2, después de que el músculo sea excitado, en esta zona viaja a lo largo de la fibra muscular a una velocidad de 2-6 m/s [10].

Este movimiento en forma de "ola" a lo largo del músculo, hace que una pareja de electrodos pueda detectar esta señal sin necesidad de una referencia, nótese que la distancia entre electrodos deberá ser de 3 mm como mínimo para que la zona despolarizada no sea detectada por los dos sensores al mismo tiempo [10].

Esta explicación nos da una idea aproximada de cómo se está detectando una señal mioeléctrica con un par de electrodos, pero no justifica completamente la forma de la señal. Una señal mioeléctrica tiene la forma que se puede apreciar en la figura 6.

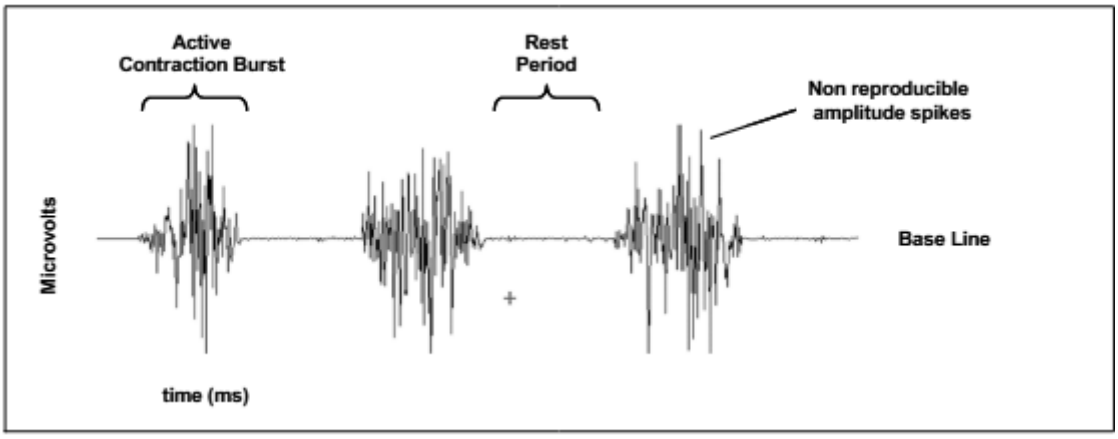


Figura 6. Señal detectada por un sensor mioeléctrico explicada en detalle [10] (Konrad, 2006).

Con la explicación que se ha dado, la señal detectada debería tener una forma más similar a un perfil sinusoidal o cuadrado, pero en la señal se aprecia una gran cantidad de picos de voltaje. La razón detrás de esta forma tan característica es que un electrodo no está detectando la señal en una única fibra muscular, sino que detecta varias fibras siendo excitadas al mismo tiempo. Estas fibras no tienen el mismo tamaño, ni se sitúan a la misma distancia, ni el electrodo se encuentra en el mismo punto de la fibra; por lo que la señal

viaja desfasada. Esta señal tan característica que se puede ver en la imagen anterior es la superposición de las señales detectadas en todas las fibras musculares. A este patrón se le suele llamar patrón de interferencia[10].

El estudio de cómo se forma la señal y de los métodos que se han utilizado para detectarla, permite sacar varias conclusiones. Las más importantes tienen que ver con cómo se va a afrontar la detección y el procesado de estas señales [10].

La primera conclusión importante es que estas señales no son replicables, el patrón de interferencia depende de varias fibras musculares sobre las que no se tiene el control, por lo que no se puede esperar que dos contracciones den el mismo perfil [10].

La segunda conclusión es que el estudio de la señal no puede tener en cuenta la amplitud de esta. Varios factores como la cantidad de fuerza ejercida al flexionar el músculo, la resistividad de la piel del paciente, el patrón de interferencia o la posición del sensor, pueden hacer variar tanto la amplitud de la señal. A existir tantas variables que afectan al valor de la amplitud, tomar decisiones o llegar a conclusiones en función de valores nominales de esta señal no es acertado [10].

3 ESTADO INICIAL DEL PROYECTO

Para entender la manera en la que se ha desarrollado este trabajo hay que partir de que este TFG no es un proyecto que se haya iniciado desde cero, sino que es la continuación del de otro alumno [3]. Esto significa que este documento no va a comprender todas las etapas de desarrollo de la prótesis, sino que solo se van a tratar aquellos aspectos en los que se han realizado mejoras o cambios significativos. Si se está interesado en otros aspectos, que no se tratan en profundidad en este documento, como el modelado de las distintas piezas de la mano o las razones por las que la prótesis utiliza un motor en concreto, y no otro, se puede consultar el TFG de Carlos Calle del Río de la Escuela de ingenierías industriales de Valladolid [3].

Como ya se manifestó en el apartado de "objetivos del proyecto", lo que se ha buscado en el presente TFG, es hacer la prótesis portátil y más responsiva. Dado que este proyecto es la continuación de otro, parece importante considerar el estado del proyecto en fases previas, antes de empezar a hablar de las mejoras y cambios que se han realizado.

3.1 ELEMENTOS QUE FORMAN LA PRÓTESIS

Cuando se inició este proyecto la prótesis contaba con los siguientes elementos:

3.1.1 La mano

Esta es la pieza (figura 8) en la que más tiempo se invirtió en el TFG que precede a éste. Esta mano consta con una combinación de motor + cabezal + encoder DCX19S EB+GPX19HP+ENX16EASY. Para conocer el razonamiento por el que este motor fue elegido consultar el TFG que precede a éste [3].

La carcasa que rodea el motor y forma de la mano están creados por impresión 3D y su diseño está directamente inspirado por la mano ElectroHand 2000 de Ottobock (figura 7) [16].

Figura 7. Protesis infantil Electrohand de Ottobock [16](Otto Bock Electrohand 2000 | Ottobock US Shop*, s. f.)*.

El conjunto de la mano cuenta con dos grupos de cables, la alimentación del motor y los cables necesarios para interaccionar con el encoder del motor.

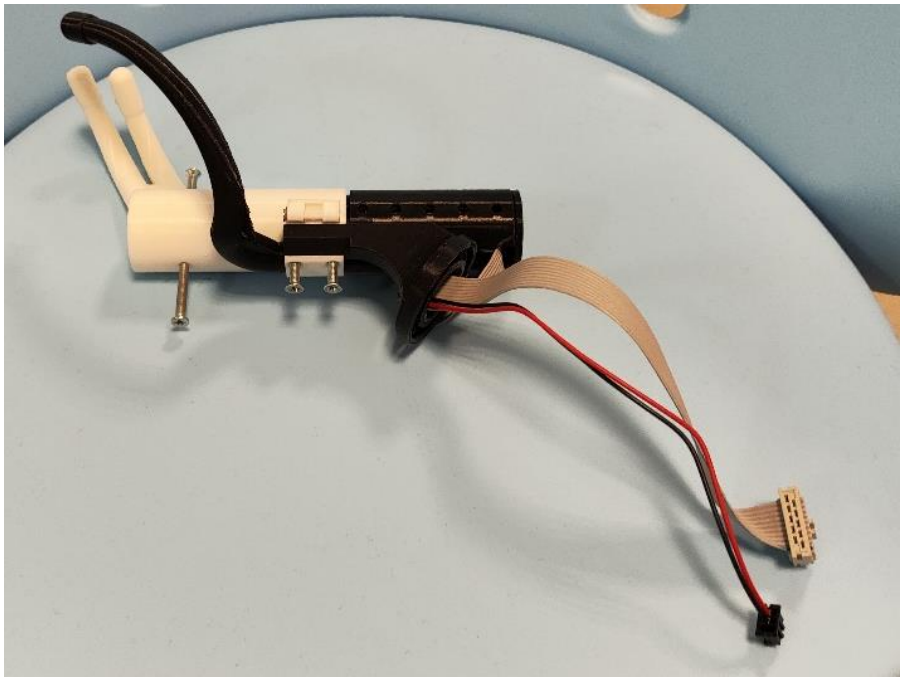


Figura 8. Mano protésica desarrollada para este proyecto.

3.1.2 Módulo IPOS-3602

El módulo IPOS-3602 (figura 9) es un controlador para motores desarrollado por la compañía Technosoft [4]. Este módulo es programable y lo suficientemente complejo como para regular la corriente que es suministrada al motor según parámetros que se pueden

guardar en su memoria. También puede ser programado con un pseudocódigo para que el motor realice una serie de movimientos, los cuales serán función de distintos valores.

Figura 9. Módulo IPOS-3602 [4](Dumitru, s. f.).

3.1.3 Raspberry Pico

A diferencia de la mayoría de los productos de Raspberry, la Raspberry Pico (figura 10) es un microcontrolador que no cuenta con un sistema operativo, esto hace de esta placa concreta, la más apta de entre las que ofrece la marca para proyectos que requieren de una programación sencilla y de un consumo de energía reducido [11].

Figura 10. Raspberry Pi Pico [11](Ltd, 2024).

3.1.4 Sensores de Ottobock

Los sensores utilizados en esta primera iteración de la prótesis son los sensores de Ottobock (figura 11) utilizados para la mano ElectroHand 2000 [5]. Estos sensores son de una calidad excelente, pero su precio va en contra de la filosofía que se está siguiendo en este proyecto, debido a su elevado precio.

*Figura 11. Sensores EMG de la marca Ottobock [5](*Electrode | Ottobock US Shop*, 2024).*

3.1.5 Placas para pines

Todos los componentes anteriormente nombrados estaban conectados utilizando un par de placas para pines de las que se utilizan típicamente en laboratorios.

En la figura 12, se muestra un esquema de cómo se encontraban conectados todos estos componentes:

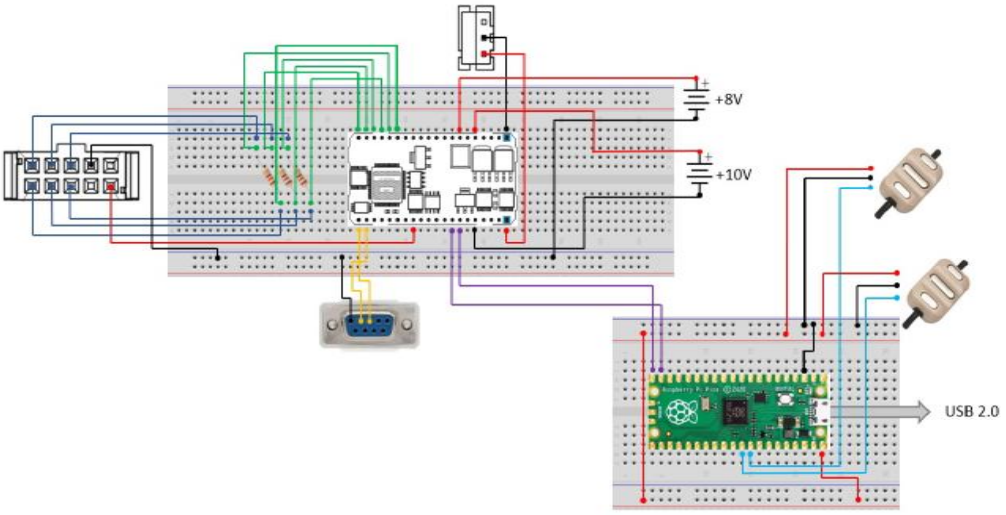


Figura 12. Esquema de las conexiones eléctricas del primer prototipo de la prótesis [3](Calle del Río, 2023).

3.2 ESTADO DEL HARDWARE Y SUS CONEXIONES

Como se acaba de ver todos los componentes que forman la prótesis se encontraban conectados mediante un par de placas de pines de laboratorio. A continuación, se muestra en la figura 13 todo el montaje en el laboratorio.

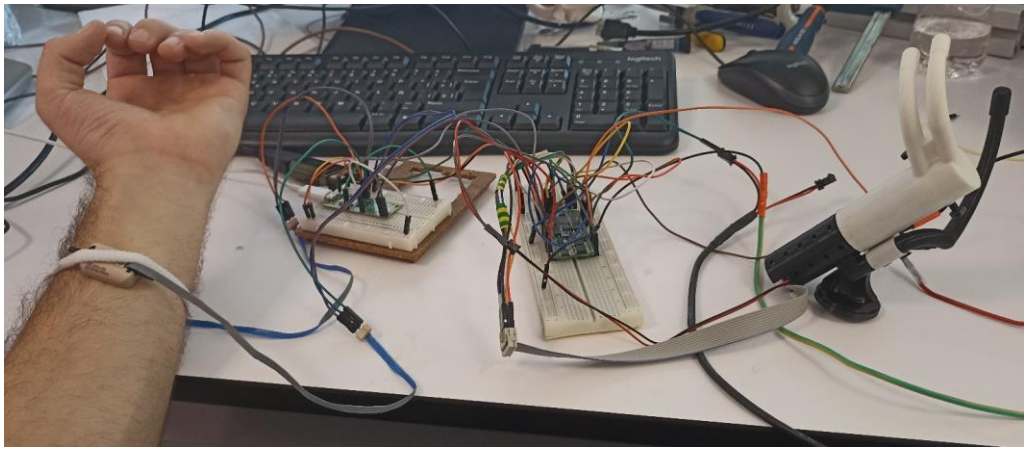


Figura 13. Primer prototipo de la prótesis completamente montado [3](Calle del Río, 2023).

Esta imagen y el esquema mostrado al final del apartado anterior pueden dar una idea general de cómo se encuentran conectados los distintos elementos del circuito. En cualquier caso, hay algunas conexiones con elementos externos que deben ser detalladas con más profundidad:

El primer elemento externo con el que este circuito está siendo conectado, es una fuente de alimentación DC variable que hay en el laboratorio. En el esquema se puede apreciar que distintas partes del circuito están siendo conectadas a valores de alimentación distintos, en este caso 10 y 8 V.

Otra conexión con un elemento externo es la que se está realizando a través del puerto Serie que está conectado al controlador del motor. Este puerto se utiliza para conectar el controlador del motor a un ordenador. A través de esta conexión el controlador puede ser programado utilizando un software distribuido por el fabricante.

Por último, hay que hablar del puerto USB de la Raspberry Pico que se está utilizando para conectar esta placa con un ordenador, esta conexión permite programar la placa y caso alimentarla.

Uno de los objetivos de este trabajo es separar la prótesis del banco de trabajo, esto significa que estas tres conexiones deben desaparecer y la prótesis debe seguir funcionando.

3.3 ESTADO DEL CÓDIGO DEL CONTROLADOR

El código de la Raspberry Pico que se utilizó en el primer prototipo de esta prótesis es bastante limitado. Esto no es de extrañar ya que la programación del controlador no era el objetivo principal del TFG previo.

El código (figura 14) estaba escrito en micro Python y es bastante sencillo, consiste en un bucle en el que se leen las señales recibidas de los dos sensores, y si estas señales

superan cierto umbral, se cambia el valor de un par de pines de la placa que están conectados al controlador del motor.

Debido a lo reducido que es este código, en este proyecto se va a escribir un nuevo código para acomodar las necesidades de este segundo prototipo de la prótesis.

```
import machine
import utime

from machine import Pin
from machine import ADC

Sensor_1 = ADC(26)
Sensor_2 = ADC(27)

Motor_der = Pin(15,Pin.OUT)
Motor_izq = Pin(14,Pin.OUT)

factor_16 = 3.3/(65535)

def main():

    while(True):

        senal_1 = Sensor_1.read_u16() * factor_16
        print (senal_1)

        if senal_1 > 2.5:
            Motor_der.on()
            Motor_izq.off()
        else:
            Motor_der.off()

        senal_2 = Sensor_2.read_u16() * factor_16
        print (senal_2)

        if senal_2 > 2.5:
            Motor_izq.on
            Motor_der.off()
        else:
            Motor_izq.off()

        utime.sleep(0.1)

if __name__ == "__main__":
    main()
```

Figura 14. Código de la Raspberry Pi Pico en el primer prototipo [3](Calle del Río, 2023).

3.4 ESTADO DEL PROGRAMA DEL CONTROLADOR DEL MOTOR

Como ya se ha comentado anteriormente, el controlador que se está utilizando puede ser programado y cuenta con un software, *EasyMotion Studio*, distribuido por la empresa que lo produce, *Technosoft*.

Este programa permite un par de funcionalidades. La primera es calibrar distintos parámetros para adaptar el controlador a las necesidades del motor que va a ser conectado. Esto permite calibrar la corriente que puede consumir, la velocidad de giro posible, y otros valores pertinentes. Esta parte del programa no se va a tocar en este trabajo, por lo que si se quiere conocer en detalle se recomienda consultar el TFG previo.

La segunda funcionalidad que se puede programar en *EasyMotion* Studio es el comportamiento del motor. La forma en la que se va a mover el motor puede ser programada utilizando una especie de pseudocódigo donde se crean distintas sentencias para determinar el movimiento del motor.

En la figura 15 vemos el pseudocódigo que se escribió para esta versión de la prótesis:

```
int Sensor_flex; // Define integer variable Sensor_flex
int Sensor_ext; // Define integer variable Sensor_ext
SetAsInput(0); //Set IO line 0 as input
Sensor_flex = IN(0); //Read I/O line 0 data into variable Sensor_flex ( 0 -> low, 1 -> high )
SetAsInput(1); //Set IO line 1 as input
Sensor_ext = IN(1); //Read I/O line 1 data into variable Sensor_ext ( 0 -> low, 1 -> high )
Deteccion: //Define label Deteccion
//Define event: When the digital input IN0 is high
!IN#0 1;
MODE VC;//Set Voltage Contouring
REF0 = 0;//Initial reference set to 0[V]
SEG 399U, 73.37750;
UPD!; //Execute on event
SEG 1U, 73.37653;
SEG 1001U, 0.00000;
SEG 400U, -73.19452;
SEG 1U, -73.19238;
SEG 0, 0.0; //End of contouring
//Define event: When the digital input IN1 is high
!IN#1 1;
MODE VC;//Set Voltage Contouring
REF0 = 0;//Initial reference set to 0[V]
SEG 399U, -73.37750;
UPD!; //Execute on event
SEG 1U, -73.37653;
SEG 1001U, 0.00000;
SEG 400U, 73.19452;
SEG 1U, 73.19238;
SEG 0, 0.0; //End of contouring
GOTO Deteccion; //Branch to Deteccion
```

Figura 15. Pseudocódigo que controlaba el comportamiento del motor en el programa EasyMotion Studio [3](Calle del Río, 2023).

Se pueden ver estas sentencias utilizadas para definir el comportamiento del motor. En este apartado no se va a explicar en profundidad como se crean estos bloques y su significado, solo se va a explicar por encima el funcionamiento que se logra con este código.

El código escrito para esta versión de la prótesis consiste en un bucle que espera a que las entradas IN0 o IN1 del controlador se pongan a uno. Estas entradas están conectadas a la Raspberry y cambian cuando la Raspberry lee el nivel de señal adecuado en los sensores. Cuando una de estas entradas cambia, el controlador entra en un proceso que dura varios segundos en el que la mano realizan una abertura o un cierre completo en función de cuál de las entradas ha cambiado. Una vez acaba el movimiento, se continúa escuchando en el bucle hasta que se detecte otro uno.

Este código es muy limitado y solo permite aberturas y cierres completos. Esto da un control limitado donde, en caso de detectar una señal, el controlador entra en un proceso de varios segundos donde el input por parte del usuario es ignorado.

4 DESARROLLO DEL PROYECTO

En este apartado se va a hablar en profundidad de todos los cambios e inclusiones de nuevos elementos que se han realizado en el proyecto.

4.1 DISEÑO DE LA PLACA Y EL CIRCUITO

El primer problema que se ha intentado solucionar ha sido el sistema de conexiones de los distintos componentes que forman la prótesis. Si bien es cierto que las placas de pines son herramientas muy útiles que permiten montar circuitos sencillos de manera rápida para probarlos, en cuanto el circuito en el que se está trabajando empieza a ser medianamente complejo, el mantener correctamente conectados todos los elementos, se convierte en una tarea tediosa, que impide realizar medidas y pruebas de manera cómoda. Además, estas conexiones poco fiables no podrían transportarse cómodamente sin estropear el circuito.

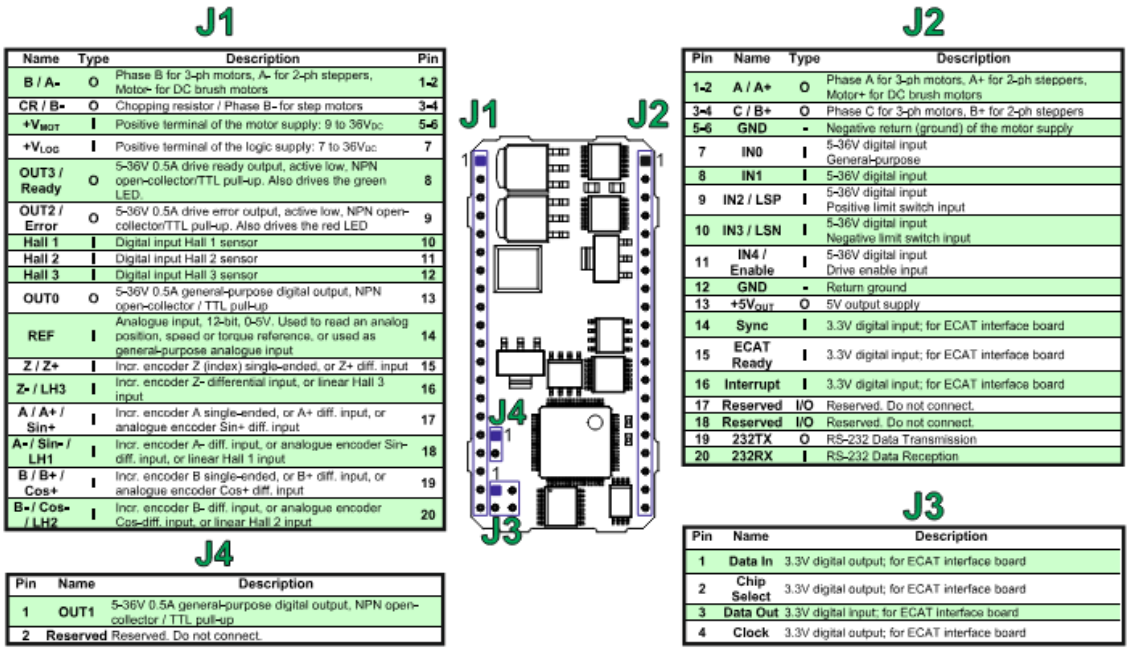
La solución que se ha encontrado a este problema ha sido diseñar una PCB, o circuito impreso, que contenga todas estas conexiones en un formato fácil de manipular y transportar.

4.1.1 Mapa de conexiones

Para diseñar un circuito impreso primero hay que tener claro las conexiones del circuito que se está intentando diseñar, a continuación, se van a mostrar todas las conexiones con las que cuenta el circuito y se va a justificar su existencia. En caso de realizar un cambio significativo en las conexiones, en comparación con la primera versión de la prótesis se explicará el porqué del cambio.

Conexiones del controlador del motor

Como ya se ha comentado antes, el controlador que se está utilizando es el IPOS3602 de *Technosoft*. Este componente cuenta con un manual de usuario que indica la manera segura de conectar esta placa a distintos componentes. Antes de comentar las distintas conexiones de manera particular, se le va a echar un vistazo al mapa de conexiones de la placa:



J1

Name	Type	Description	Pin
B / A-	O	Phase B for 3-ph motors, A- for 2-ph steppers, Motor- for DC brush motors	1-2
CR / B-	O	Chopping resistor / Phase B- for step motors	3-4
+V_{MOT}	I	Positive terminal of the motor supply: 9 to 36V_{DC}	5-6
+V_{LOG}	I	Positive terminal of the logic supply: 7 to 36V_{DC}	7
OUT3 / Ready	O	5-36V 0.5A drive ready output, active low, NPN open-collector/TTL pull-up. Also drives the green LED.	8
OUT2 / Error	O	5-36V 0.5A drive error output, active low, NPN open-collector/TTL pull-up. Also drives the red LED	9
Hall 1	I	Digital input Hall 1 sensor	10
Hall 2	I	Digital input Hall 2 sensor	11
Hall 3	I	Digital input Hall 3 sensor	12
OUT0	O	5-36V 0.5A general-purpose digital output, NPN open-collector / TTL pull-up	13
REF	I	Analogue input, 12-bit, 0-5V. Used to read an analog position, speed or torque reference, or used as general-purpose analogue input	14
Z / Z+	I	Incr. encoder Z (index) single-ended, or Z+ diff. input	15
Z- / LH3	I	Incr. encoder Z- differential input, or linear Hall 3 input	16
A / A+ / Sin+	I	Incr. encoder A single-ended, or A+ diff. input, or analogue encoder Sin+ diff. input	17
A- / Sin- / LH1	I	Incr. encoder A- diff. input, or analogue encoder Sin- diff. input, or linear Hall 1 input	18
B / B+ / Cos+	I	Incr. encoder B single-ended, or B+ diff. input, or analogue encoder Cos+ diff. input	19
B- / Cos- / LH2	I	Incr. encoder B- diff. input, or analogue encoder Cos- diff. input, or linear Hall 2 input	20

J2

Pin	Name	Type	Description
1-2	A / A+	O	Phase A for 3-ph motors, A+ for 2-ph steppers, Motor+ for DC brush motors
3-4	C / B+	O	Phase C for 3-ph motors, B+ for 2-ph steppers
5-6	GND	-	Negative return (ground) of the motor supply
7	IN0	I	5-36V digital input General-purpose
8	IN1	I	5-36V digital input
9	IN2 / LSP	I	5-36V digital input Positive limit switch input
10	IN3 / LSN	I	5-36V digital input Negative limit switch input
11	IN4 / Enable	I	5-36V digital input Drive enable input
12	GND	-	Return ground
13	+$5V_{OUT}$	O	5V output supply
14	Sync	I	3.3V digital input; for ECAT interface board
15	ECAT Ready	I	3.3V digital input; for ECAT interface board
16	Interrupt	I	3.3V digital input; for ECAT interface board
17	Reserved	I/O	Reserved. Do not connect.
18	Reserved	I/O	Reserved. Do not connect.
19	232TX	O	RS-232 Data Transmission
20	232RX	I	RS-232 Data Reception

J3

Pin	Name	Description
1	Data In	3.3V digital output; for ECAT interface board
2	Chip Select	3.3V digital output; for ECAT interface board
3	Data Out	3.3V digital input; for ECAT interface board
4	Clock	3.3V digital output; for ECAT interface board

J4

Pin	Name	Description
1	OUT1	5-36V 0.5A general-purpose digital output, NPN open-collector / TTL pull-up
2	Reserved	Reserved. Do not connect.

Figura 16. Pinout del IPOS-3602 [8](iPOS360x MX-CAN - User Manual*, 2024)*.

En la figura 16 se puede ver el esquema de pines de la placa. Nótese que los pines están divididos en 4 grupos distintos, el modelo de la placa que se ha utilizado solo cuenta con los pines correspondientes a los grupos J1 y J2.

Conexión con el motor

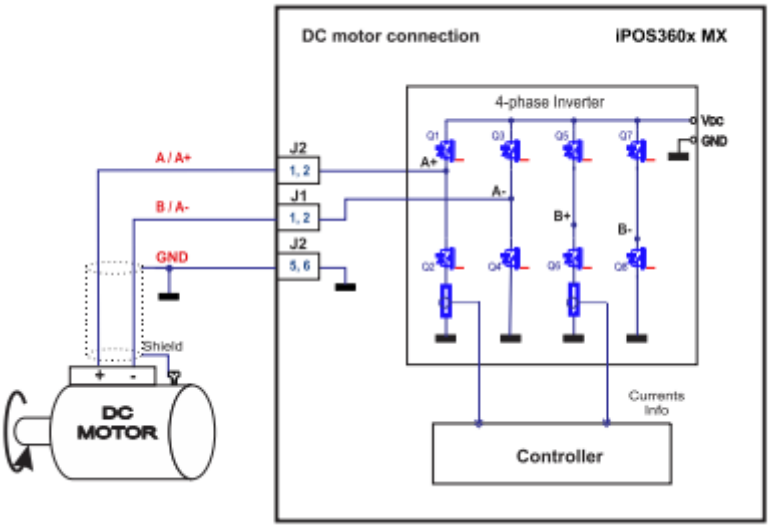


Figura 17. Esquema de las conexiones al motor [8](iPOS360x MX-CAN - User Manual*, 2024)*.

El motor ha sido conectado como un motor DC normal, en el caso de este motor en concreto, el cable de tierra es omitido por lo que solo se han conectado los cables llamados A y B en la figura 17.

Conexión con el *encoder*

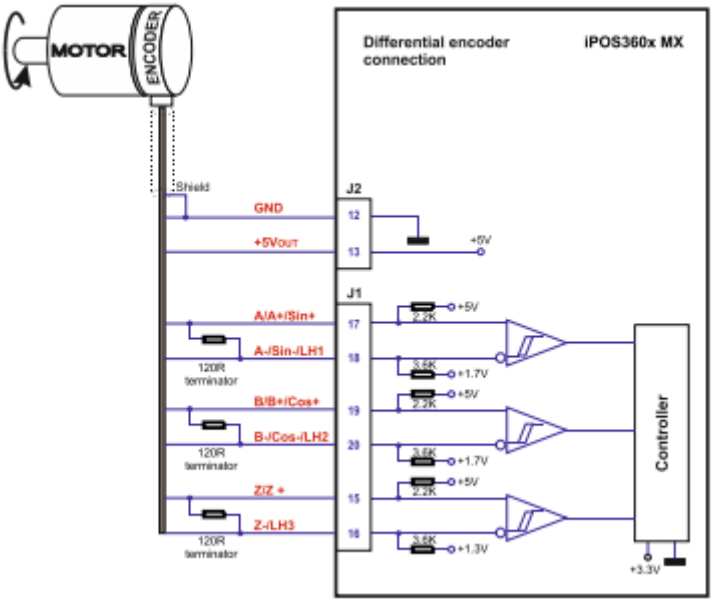


*Figura 18. Esquema de las conexiones con el encoder [8](*iPOS360x MX-CAN - User Manual*, 2024).*

El *encoder* que se está utilizando es un *encoder* diferencial, las conexiones se han realizado como se ve en la figura 18. Las resistencias que recomienda el manual de usuario son de 120 ohm, debido a que no se tenían a mano resistencias de este valor concreto se han utilizado resistencias de 100 ohm en su lugar. Un cambio de esta magnitud no debería representar un problema en el funcionamiento del *encoder*.

Conexión de la alimentación

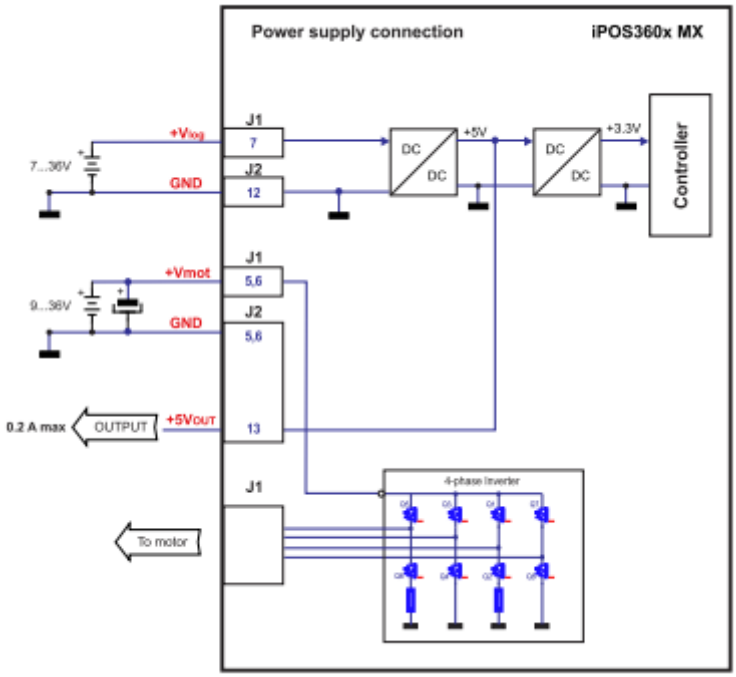


*Figura 19. Esquema de las conexiones a la alimentación [8](*iPOS360x MX-CAN - User Manual*, 2024).*

Debido a la naturaleza portátil de una prótesis, uno de los objetivos a cumplir en este proyecto ha sido poder alimentar el circuito entero con una única batería. Se ha incluido en la entrada de la alimentación un condensador de 220 µF para evitar potenciales daños en los componentes del circuito.

Para alimentar la Raspberry y los sensores que funcionan a 5V, se ha utilizado la salida de voltaje que está representada en la figura 19. Sobre esta forma de alimentar los componentes de 5V se deben resaltar algunas cuestiones. La primera es que se han realizado pruebas para comprobar que el consumo de estos elementos no supere los 0.2 A que puede suministrar el controlador y se ha comprobado que son más que suficientes. La segunda cuestión que hay que comentar es que con pruebas posteriores se ha comprobado que en caso de conectar la Raspberry a un ordenador usando el puerto USB, la corriente puede entrar hacia el controlador del motor por esta salida. Para evitar que esto ocurra hay que incluir un diodo a la salida del pin J2/13 del controlador del motor. Este diodo no está incluido en el prototipo que se ha utilizado, pero debería ser incluido en todas las iteraciones posteriores de esta placa.

El último punto que comentar sobre esta conexión es que en la versión del anterior TFG se utilizó una salida de la fuente de alimentación a 8V para alimentar la lógica de la placa y una salida de 10V para alimentar la circuitería de potencia. Revisando el esquema de la conexión se ha llegado a la conclusión de que ambos elementos pueden ser conectados a la misma alimentación, que será una batería de 12 V.

Conexión al PC

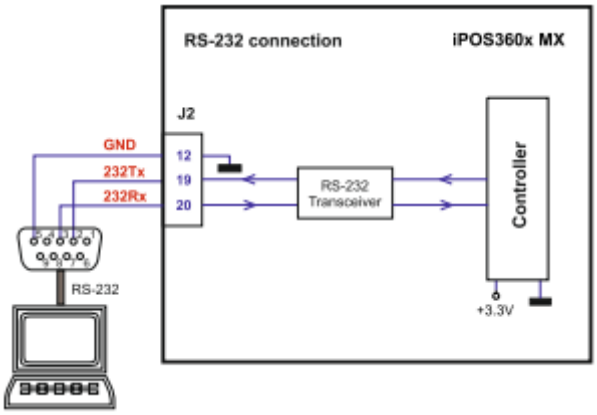


*Figura 20. Esquema de las conexiones con el PC [8](*iPOS360x MX-CAN - User Manual*, 2024).*

La última conexión relevante (figura 20) que se puede encontrar en el manual de usuario de la placa, es la que permite conectar la placa a un ordenador usando un puerto RS-232. Si bien esta utilidad no será necesaria en el diseño final, para poder desarrollar cómodamente este proyecto se debe dar una salida a estos pines. Como ya se ha comentado el controlador IPOS-3602 es programable y debe ser programado usando el software distribuido por la empresa fabricante. Para que este software pueda interactuar con la placa hay que conectarlo con el ordenador de alguna manera y ahí es donde entra esta conexión “Serial”.

El puerto Serie cuenta con 9 pines, pero solo se han incluido los tres necesarios: tierra, lectura (Rx) y escritura (Tx).

Conexión a la Raspberry

Ya se ha hablado en el apartado de la alimentación de como la Raspberry es alimentada a través del controlador del motor, pero estas no son las únicas conexiones que tienen en común. Dos de los pines digitales de la Raspberry se encentran conectados con las entradas digitales del controlador llamada IN0 e IN1. Estas dos conexiones se van a utilizar para que la Raspberry pueda informar al controlador del motor cuando debe mover el motor y en qué dirección.

Conexiones de la Raspberry Pi Pico

Ya se ha hablado en el apartado anterior de las conexiones entre la Raspberry y el motor, a continuación se van a tratar las otras conexiones con las que cuenta la Raspberry.

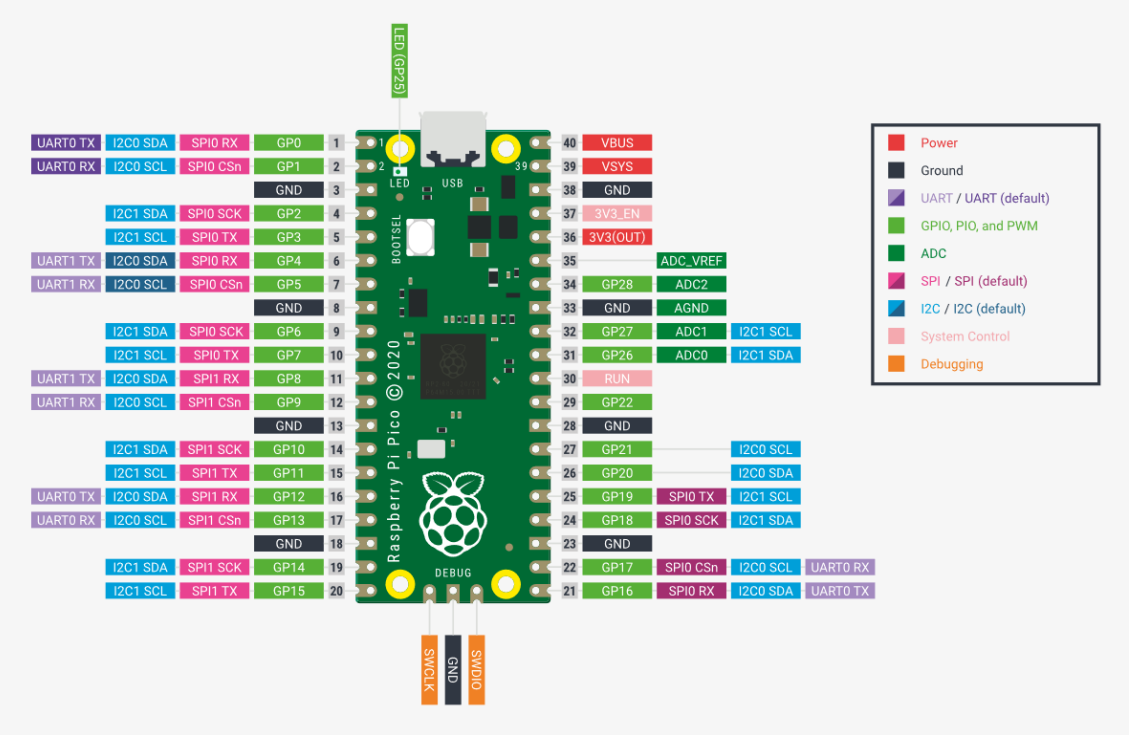


Figura 21. Pinout de la Raspberry Pi Pico [17](Raspberry Pi Pico Datasheet, *s. f.*).

En la imagen 21 se puede ver el *pinout* de la Raspberry Pico, casi todas las conexiones de las que se va a hablar pueden ser realizadas utilizando múltiples pines. Por ejemplo, una salida digital podría utilizar cualquier pin que en la figura 21 tenga la etiqueta “GP”. En el diseño se han tomado los pines que han resultado más convenientes en cada caso, por lo que a continuación no se va a denominar a los pines por su número, sino que se van a clasificar según el papel que cumplen.

Conexión con los sensores

La información necesaria para determinar el movimiento de la prótesis es recogida utilizando una pareja de sensores mioeléctricos (EMG). Ya se ha comentado que estos sensores son alimentados con la misma metodología con la que se alimenta la Raspberry, por otro lado, las señales producidas en estos sensores van conectada a una pareja de pines analógicos de la Raspberry para posteriormente poder ser procesadas. En el *pinout* se pueden encontrar los pines que pueden cumplir esta función buscando la etiqueta verde oscura, que indica los pines con acceso a un ADC.

Conexión con el ordenador

La Raspberry cuenta con un puerto micro-USB que puede utilizar para conectarse a un ordenador. Esta conexión no es permanente y no debe usarse cuando la prótesis vaya a ser utilizada. Este puerto es necesario para poder programar la Raspberry, además hasta que se encuentre un método mejor para hacerlo este puerto también es utilizado para poder calibrar los sensores.

Por último, es importante comentar que todos los elementos que deben ser conectados a tierra están siendo conectados a una tierra común.

4.1.2 Diseño de la placa

La PCB ha sido diseñada utilizando el editor de placas de *KiCad 6.0*, el diseño final que se ha usado en este trabajo es el que se puede ver en la figura 22:

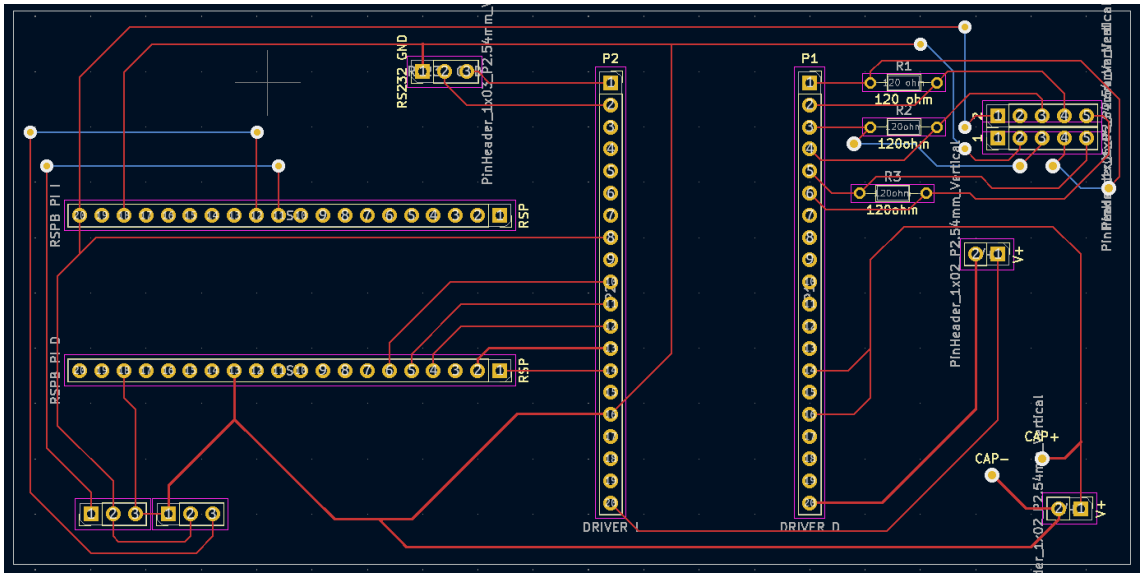


Figura 22. Diseño final de la placa en el editor de PCBs de KiCad.

A continuación, se van a comentar distintas decisiones de diseño y limitaciones que han llevado a la creación de este prototipo concreto.

Empezando por las limitaciones, con el fin de ahorrar costes y tiempo, se ha utilizado una placa con un único substrato de cobre. Hoy en día se suelen utilizar más substratos y materiales que permiten pistas más finas y mayor precisión, pero como el objetivo de este proyecto era el de crear un prototipo, se ha sacrificado el reducir el tamaño de la placa para lograr inmediatez.

Este método algo arcaico de fabricación ha supuesto un tamaño mayor y la necesidad de pasar algunos cables por la parte superior de la placa. Se pueden apreciar en la imagen del diseño unos cables que son de color azul, en lugar de rojos. Estos son cables que no podían ser incluidos en la pista de cobre debido a la topología del circuito. Para poder incluirlos, se sueldan los cables de forma que pasen por la cara donde no se encuentra el cobre.

Además de la limitación técnica ya expuesta, también se han tomado otras decisiones que han limitado la manera en la que la placa ha sido diseñada:

- Los tres pines que se pueden ver en la parte superior izquierda del esquema son los que se utilizan para conectar el controlador del motor con un PC, en un diseño final se podrían omitir, pero en este prototipo son necesarios.
- La Raspberry pico debe ser posicionada de manera que su puerto USB apuntase al exterior de la placa, como se ha comentado en el apartado de las conexiones, el puerto USB de la Rasberry va a ser utilizado de manera habitual, por lo que debe ser fácilmente accesible.
- Las conexiones a la mano y las conexiones a los sensores deben, si es posible estar en lados opuestos de la placa, debido a la posición prevista de este elemento cuando se use la prótesis, los cables que salen a estos componentes van a ir en direcciones opuestas por lo que poner sus conectores en lados opuestos hace acomodar la PCB de modo más sencillo.

Tras algunas iteraciones de esta placa se ha llegado al diseño que se puede ver en la figura 22 al inicio de este apartado. Después de lograr este diseño se han creado los Gerbers, que han sido utilizados para que una fresadora elabore las separaciones en la placa de cobre de la PCB y perfore los agujeros donde van a ir conectados los pines. El diseño final con todos los componentes ya soldados se puede ver a continuación en las figuras 23 y 24.

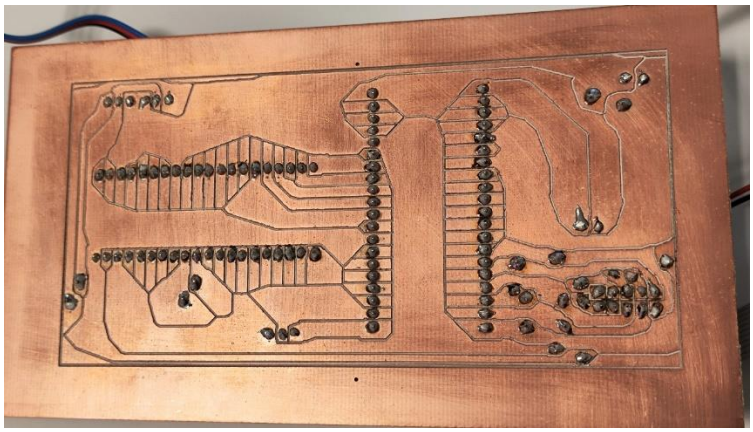


Figura 23. Parte trasera de la PCB.

Figura 24. Parte delantera de la PCB.

4.1.3 Prototipo fallido

El diseño que se ha visto a lo largo de este apartado no fue el primero, se hizo una primera placa que se tuvo que de desechar debido a algunos fallos irreversibles.

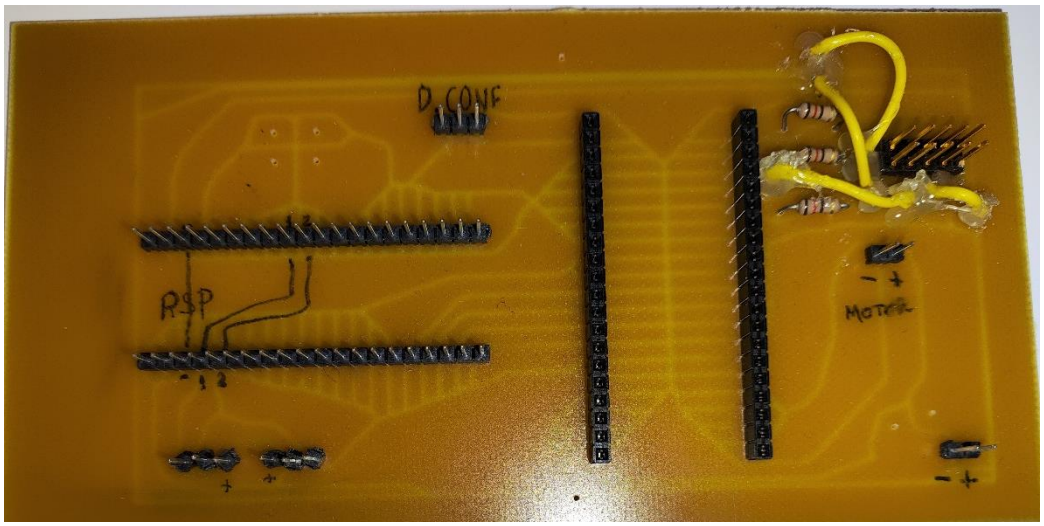


Figura 25. Parte frontal del prototipo de placa fallido.

El diseño de la placa era muy similar, como se puede apreciar en la figura 25, pero se cometieron algunos errores al diseñarla que permitían utilizar el circuito, pero lo hacían dificultoso y tedioso. Además, uno de los errores no permitía que el diseño funcionase para el propósito que se le tenía pensado.

4.1.3.1 Error 1. Separación accidental de la tierra.

En este primer diseño se separó la tierra del circuito en dos por accidente, en la imagen se puede observar unas líneas dibujadas a través de los pines de la Raspberry pico. El que va en línea recta es la conexión de tierra que se omitió. La tierra se encontraba conectada por completo cuando la Raspberry se encontraba en su lugar, pero como se verá a continuación eso no era siempre posible.

4.1.3.2 Error 2. Pines digitales sobre analógicos

En primera instancia se pensaba que los pines digitales de la Rasberry eran suficiente para recibir la señal procedente del sensor, en cuanto se hicieron los primeros intentos se comprobó que no iba a ser posible utilizar estos pines. Las líneas dibujadas en los pines de la Raspberry con un 1 y un 2 muestran los pines originales y los que se utilizaron en la segunda placa.

4.1.3.3 Error 3. Colocación de los pines de los sensores

La colocación de los pines para la conexión de los sensores de esta placa está pensada para los sensores de *Ottobock*. Estos eran los únicos sensores EMG que se tenían cuando se diseñó esta placa y la idea de usar unos sensores EMG más baratos apareció después de que esta fuera impresa, otro problema que se encontró con estos sensores es que, debido a la particular forma de los *sockets* de estos sensores, resultaba muy difícil conectarlos a la placa de manera cómoda.

En la figura 26 se ha dibujado una line que va entre el centro del primer y el tercer *socket*, se puede apreciar que el segundo no se encuentra en la misma recta, creando una forma difícil de replicar.

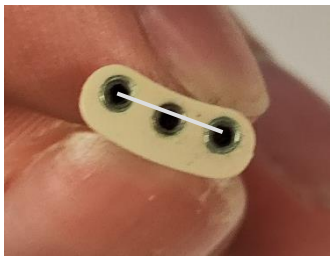


Figura 26 Conexión peculiar de los sensores de Ottobock

4.1.3.4 Error 4. Fijación de la Raspberry

La Raspberry pico es una placa que se suele vender sin pines y éste era el estado en el que se encontraba cuando se comenzó el presente TFG.

Figura 27. Raspberry Pi Pico [17](Raspberry Pi Pico Datasheet*, s. f.)*.

Si se intenta mantener esta placa conectada colocándola sobre pines estándar de 0.76 mm rápidamente puede verse que mantener todas las conexiones de manera adecuada resulta imposible con la cantidad de juego que queda en los orificios para los pines. Este problema causaba que algunas de las entradas de la Raspberry dejasen de hacer contacto e hiciesen fallar el circuito. Para solucionar este problema se le han soldado unos pines a la Raspberry como se puede ver en la figura 28, lo que va a hacer más fácil trabajar con ella.

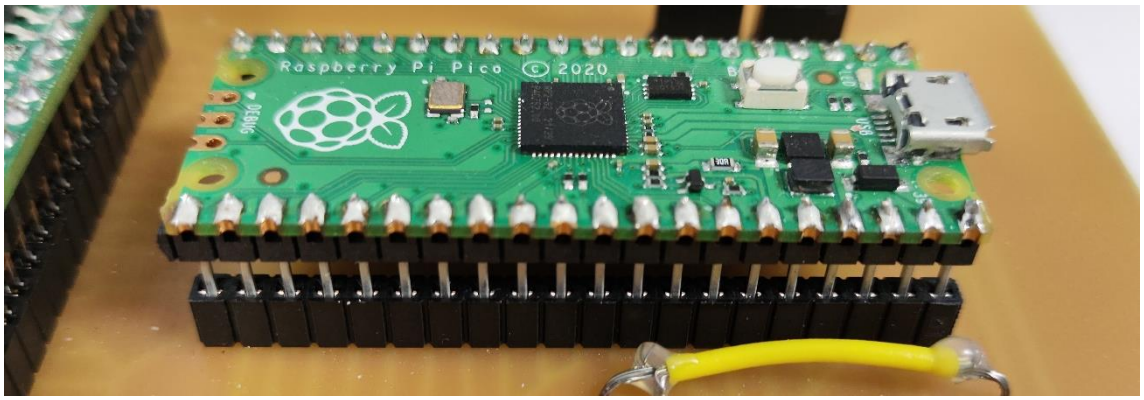


Figura 28. Raspberry Pi Pico con pines soldados.

Otra diferencia que se puede notar entre esta primera versión de la placa y la versión final es la presencia del espacio para un botón. Si se mira con detalle la placa se puede ver hueco para cuatro pines, ahí es donde iría el botón. El botón no se terminó de instalar en esta primera placa porque no se había dejado espacio para introducir una resistencia necesaria para su correcto funcionamiento. En el segundo diseño la idea de incluirlo fue descartada.

4.2 PROGRAMACIÓN DEL CONTROLADOR

Este proyecto está utilizando una Raspberry pi pico como controlador del circuito. Este controlador puede ser programado en varios lenguajes de programación, en el TFG previo se programó utilizando micro Python, pero como el código va a ser reescrito de cero, en este caso se va a programar utilizando Arduino IDE, la razón detrás de este cambio no es para mejorar ningún aspecto técnico en particular, se hace porque estoy más familiarizado con el entorno de trabajo de Arduino. Otra razón es que la naturaleza *open source* de los productos de Arduino hace muy sencillo encontrar los recursos necesarios para programar esta placa usando en internet.

El trabajo del controlador en la prótesis consiste en recibir las señales procedentes de los sensores, interpretarlas y producir las señales necesarias para comunicarle al controlador del motor que se debe mover. Además de esta tarea principal, el controlador también puede mostrar por pantalla las distintas señales con las que trabaja cuando se conecta a un ordenador, lo que facilita calibrar los sensores para que se adapten a cada usuario.

Se va a comentar el código escrito, no se va a comentar línea por línea, sino que se van a destacar algunas partes importantes o difíciles de interpretar.

4.2.1 EMGFilters.h

La biblioteca "EMGFilter.h" es la única biblioteca externa que se ha utilizado para el proyecto, es una biblioteca que incluye algunos filtros para mejorar la calidad de la señal recibida de los sensores. Esta biblioteca ha sido utilizada ya que es recomendada por el fabricante para este tipo de proyectos. El filtro que se ha utilizada está pensado para eliminar la posible interferencia a 50 Hz producida por la alimentación [9]. Nótese que en el diseño final de la prótesis este ruido no debería existir, ya que la fuente de alimentación del circuito es una batería DC. A pesar de esto se ha conservado el filtro para casos en los que se realicen pruebas con otras alimentaciones como puede ser una fuente de alimentación variable de laboratorio.

Un detalle importante sobre esta biblioteca es que la versión que se está usando no es la que se puede encontrar en GitHub si la buscas en internet. La versión en internet de esta biblioteca solo permite tener una instancia del filtro. En el caso de este proyecto, necesitaríamos al menos dos filtros, uno para cada sensor. Las modificaciones realizadas a la biblioteca están basadas en una propuesta no implementada que se puede encontrar en las *pull request* del repositorio de la biblioteca en GitHub [12].

4.2.2 Throhold

En el código se pueden encontrar un par de constantes que han sido llamadas "Throhold1" y "Throhold2" Estas señales son utilizadas para eliminar el ruido de base de la señal. Si se le echa un vistazo a una medición de un sensor EMG cualquiera:

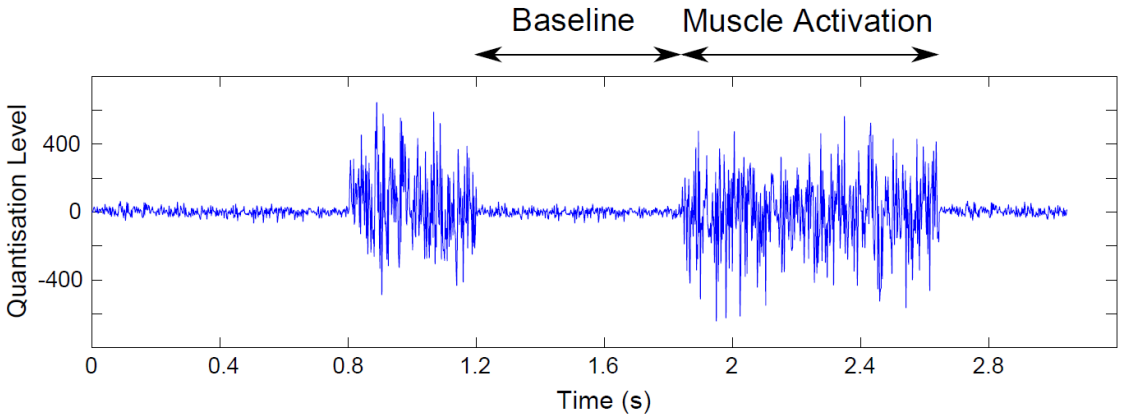


Figura 29. Señal a la salida de un sensor mioeléctrico [10](Konrad, 2006).

En la figura 29 podemos ver que cuando no hay activación del músculo, la señal sigue fluctuando. Esto se llama ruido de base o *baseline noise*. Las constantes Throhold se utilizan en el código para que estas pequeñas fluctuaciones cerca del cero sean ignoradas y no provoquen falsas activaciones.

```
if((envlope1 > Throhold1) && (envlope1 < Failureth1)){
  if((conectAxis == 1) && (keepAlive1 > keepAliveLowTH))
    digitalWrite(controllerCom5, LOW);
  if(keepAlive1 < keepAliveHighTH){
      keepAlive1 = keepAlive1 + 10;
  }
```

Figura 30. Fragmento del código con la condición que depende de Throhold.

En este fragmento de código (figura 30) se puede ver como para poder llegar a la línea *"digitalWrite(controllerCom5, LOW);"*, encargada de activar el motor en una de sus direcciones, la variable "envlope1" que guarda el valor actual de la señal procedente del sensor se compara con "Throhold1" que como se ha explicado tiene un valor lo suficientemente grande como para que la señal no pueda ser ruido. Si la señal en este instante es más grande que el valor fijo "Throhold1" se considera que se está detectando señal.

En la condición se puede ver que la señal guardada en "envlope1" también es comparado con otro valor que se ha llamado "Failureth1", y en este caso "envlope1" debe ser de un valor menor al guardado en "Failureth1", esta es otra constante que guarda un valor tan grande, que si se alcanza se puede considerar que la medida que se está tomando es errónea. Las señales EMG tienen un rango de valores típicos, si se detecta un valor mucho más grande que los presentes en este rango es muy probable que se trate de una interferencia, o que los sensores no estén funcionando como deberían. Una situación típica donde se dan estos valores es cuando los sensores no están en contacto con la piel.

4.2.3 KeepAlive

En el código (figura 31) se pueden encontrar un par de variables que han sido llamadas "keepAlive1" y "keepAlive2".

```
int keepAlive1 = 0;
int keepAlive2 = 0;
```

Figura 31. Variables keepAlive en el código.

Esta pareja de variables se utiliza para guardar un valor que permita tener una noción de como se ha comportado la señal en los bucles actuales al anterior. El valor de estas señales aumenta cada vez que "envlope" tiene un valor que cumple la condición vista en el apartado anterior, y disminuye cuando no cumple la condición. Las variaciones de "keepAlive" permiten conocer como se ha estado comportando la señal en los bucles anteriores, y así reaccionar de acuerdo con múltiples instancias de la señal en lugar de solamente al instante actual. Así por ejemplo si el valor de "envlope" es válido, pero "keepAlive" es cero, sabemos que, de las últimas instancias de la señal, esta es la primera con un valor válido. Con este conocimiento, se puede esperar a los siguientes bucles para asegurarse de que el valor detectado no es un error. En otro caso donde "envlope" es válido, pero "keepAlive" tiene un valor alto, sabemos que los instantes anteriores estaban detectando señal. Por lo que podemos suponer, que el musculo está efectivamente siendo flexionando y la señal que estamos detectando no se trata de un error.

Este sistema también funciona en casos donde el valor de "envlope" no es válido. Si el valor no es válido, pero "keepAlive" es muy grande, podemos suponer que se ha medido un valor anormalmente bajo y que el músculo sigue flexionado. En cambio un valor "envlope" no válido y un valor de "keepAlive" bajo nos indica que la señal no se encuentra activa.

Decidir para que valores de "keepAlive" consideramos que lo que se está detectando es un error y como de rápido aumenta y disminuye el valor de "keepAlive" es un balance delicado. Si los valores son muy exigentes y la variación de la variable muy lenta, la prótesis se sentirá poco responsiva y el usuario notará un retardo en el funcionamiento de la mano, si por otro lado se utilizan valores laxos y variaciones rápidas de la variable existe la posibilidad de que se cuelen errores que provoquen movimientos erróneos o inintencionados.

Los valores finales que se han decidido han sido obtenidos a base de prueba y error, al igual que el valor de "Throhold" que debe ser ajustado en función de la persona que esté usando la prótesis estos valores pueden ser modificados para lograr una mejor experiencia.

4.2.4 Lectura por pantalla

En la parte final del código se pueden observar unas pocas líneas que utilizan las funciones "Serial" de Arduino (figura 32). Estas funciones se utilizan para comunicarse con otros dispositivos utilizando el puerto micro USB con el que suelen contar este tipo de placas.

```
timeStamp = micros() - timeStamp;
if (timerDebug){
  Serial.print("Squared Data: ");
  Serial.println(envlope1);
}
```

Figura 32. Fragmento del código que permite mostrar variables por pantalla en función del tiempo.

La razón por la que estas líneas han sido incluidas es porque permiten observar el estado de distintas variables del programa en función del tiempo. Esto ayuda a comprobar el correcto funcionamiento del código y permite calibrar los sensores cuando surge la necesidad.

4.3 PROGRAMACIÓN DEL CONTROLADOR DEL MOTOR

El controlador del motor que se está utilizando es programable y permite condicionar el movimiento del motor en función de algunas variables de entrada. Se van a utilizar estas entadas para darle al controlador la información necesaria para que el motor se mueva de acuerdo con las señales detectadas en los sensores.

EasyMotion Studio es un software desarrollado por Technosoft que se utiliza para la programación de sus productos. El controlador IPOS-3602 es un producto de Technosoft por lo que este software es la herramienta más adecuada para programarlo.

Si se abre un proyecto en este programa nos encontramos con la siguiente pantalla (figura 33):

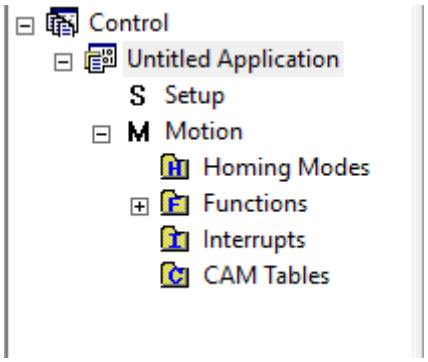


Figura 33. Pestaña para navegar en EasyMotion Studio.

"Control" es el nombre que se le ha dado al proyecto y como se puede ver cuenta con dos apartados principales que el programa llama "Stepu" y "Motion".

4.3.1 Setup

Al hacer clic en la ventana de "Setup" podemos observar lo que se ve en la figura 34:

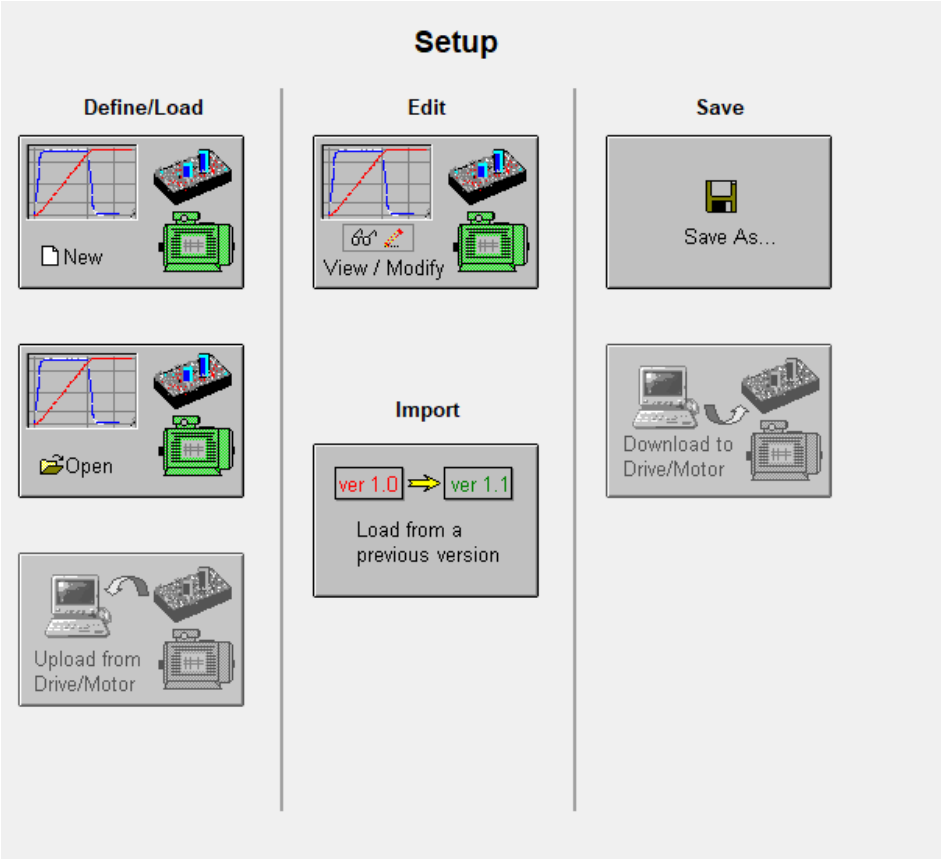


Figura 34. Pestaña "Setup" de EasyMotion Studio cuando no tiene una placa conectada.

Esta ventana permite ver y modificar varios valores relacionados con el controlador y con el motor que se va a conectar a este, la configuración que se lleva a cabo en esta pantalla es necesaria para lograr un correcto funcionamiento del motor, por suerte este proceso fue realizado en el TFG anterior a este. Si se quieren conocer los detalles de esta configuración y como se obtuvieron los valores necesarios para realizarla se puede consultar el TFG del primer prototipo.

4.3.2 Motion

Si se hace clic en "Motion" el programa abre otra ventana que va a tener una apariencia similar a la vista en la figura 35:

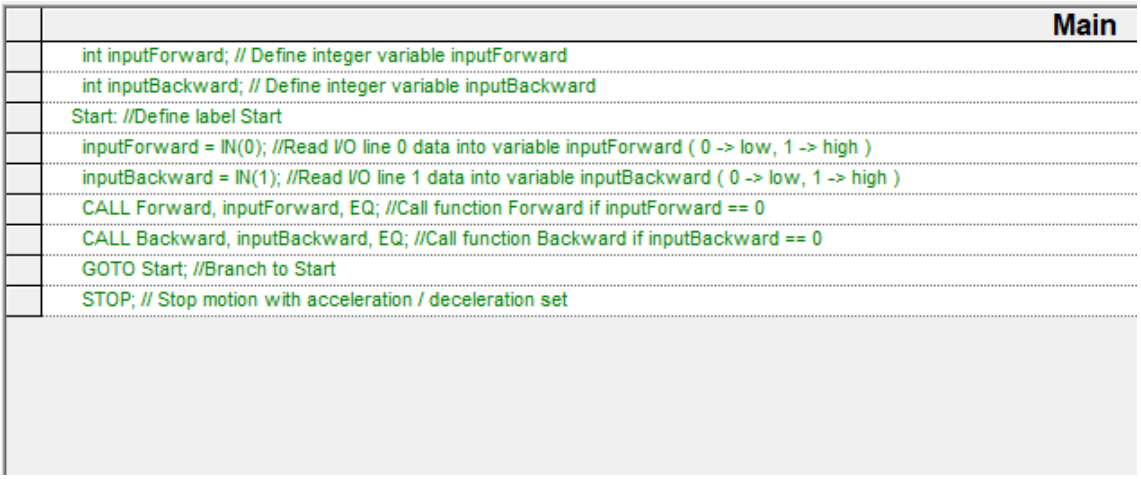


Figura 35. Pestaña "Motion", donde se puede ver el código "Main" del controlador.

Esta serie de líneas forma un pseudocódigo que permite programar como se va a comportar el motor conectado a este controlador. Esta herramienta ha sido utilizada para crear un programa que mueve el motor de la mano de acuerdo con la pareja de señales que se envían al controlador desde la Raspberry pico.

A continuación, se van a revisar las distintas líneas y se va a explicar cuál es su propósito y funcionamiento.

4.3.2.1 Declaración de variables

```
int inputForward; // Define integer variable inputForward
int inputBackward; // Define integer variable inputBackward
```

Figura 36. Pseudocódigo para declarar dos variables.

Empezando desde arriba, las dos primeras líneas que se pueden ver (figura 36) declaran dos variables que se han llamado “inputForward” e “inputBackward”. En estas dos variables se va a guardar el estado de las señales procedentes desde la Raspberry pico.

Si se hace doble clic en estas líneas se puede ver la pantalla que ha sido utilizada para declararlas (figura 37).

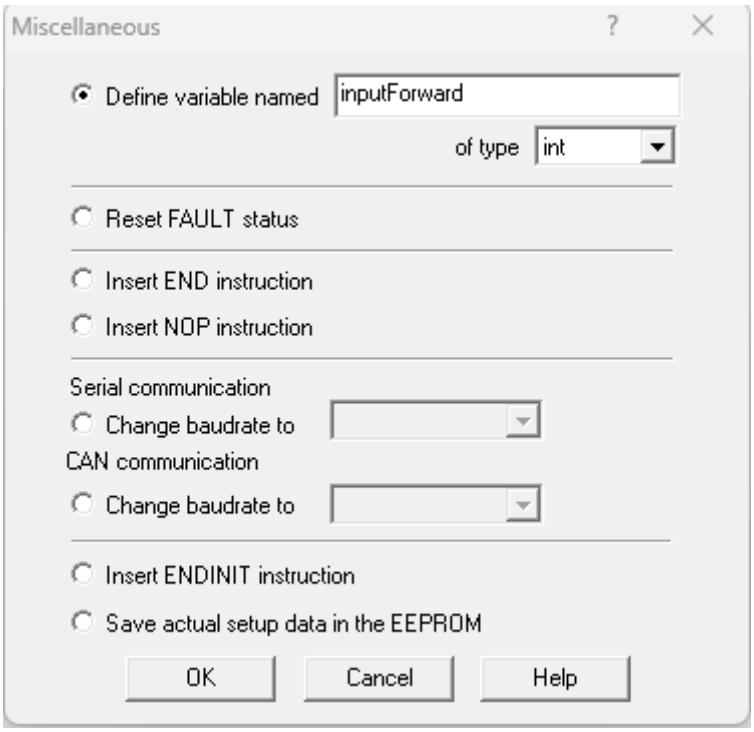


Figura 37. Pestaña utilizada para declarar variables.

4.3.2.2 Etiqueta "Start"

```
Start: //Define label Start
```

Figura 38. Pseudocódigo que crea una etiqueta.

En la línea siguiente (figura 38) se declara lo que el programa apoda "etiqueta" con el nombre "Start". Estas líneas se pueden utilizar como puntos de referencia para que el programa se puedan mover hasta ellas. En este caso esta etiqueta va a ser utilizada para definir el inicio del bucle principal del programa de ahí su nombre.

Haciendo clic en esta línea se puede ver la pantalla que se muestra a continuación (figura 39). Esta pantalla también va a ser utilizada para declarar la línea que permite saltar de vuelta a esta etiqueta, así como cualquier otra línea que tenga que ver con saltos y llamadas a funciones.

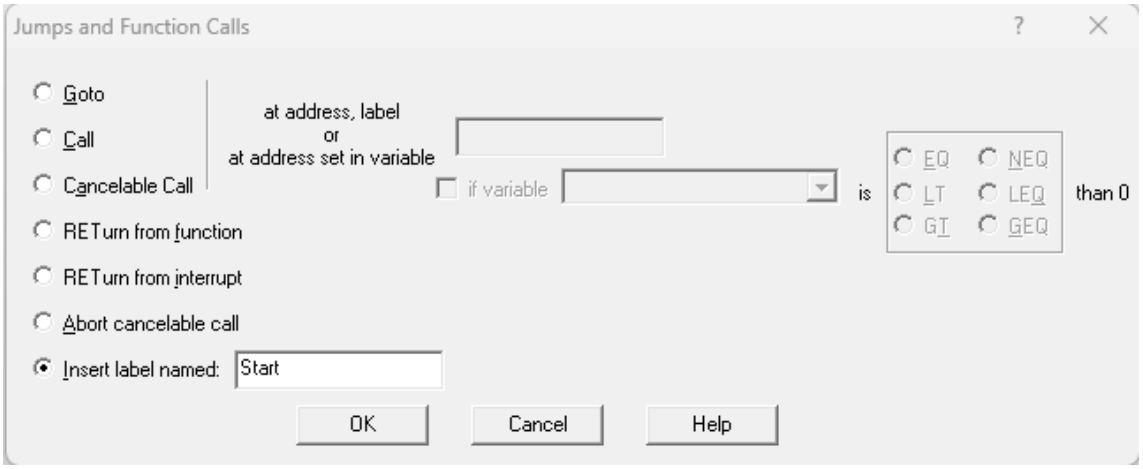


Figura 39. Pestaña utilizada para la creación de etiquetas.

4.3.2.3 *Lectura de las señales de entrada*

```
inputForward = IN(0); //Read I/O line 0 data into variable inputForward ( 0 -> low, 1 -> high )
inputBackward = IN(1); //Read I/O line 1 data into variable inputBackward ( 0 -> low, 1 -> high )
```

Figura 40. Pseudocódigo para leer valores de entrada.

En estas líneas (figura 40) lo que se hace es leer dos de los pines de entrada del controlador llamados IN0 e IN1. Estos pines se encuentran conectados con dos de los pines I/O digitales de la Raspberry pico a través de estos pines la Rasberry envía señales de acuerdo con lo que se está detectando en los sensores como se ha podido ver en el apartado 4.2.

Solo se va a mostrar la parte relevante para la lectura de los pines de la ventana correspondiente (figura 41) debido a que la ventana "I/O" de este programa es bastante extensa. Un detalle interesante sobre los pines de entrada de esta placa es que, a pesar de contar con cinco pines de entrada, que son llamados IN0-5 solamente IN0 e IN1 son para entradas de datos general, con los otros pines teniendo tareas más concretas. En caso de querer enviar más de dos señales al controlador del motor, habría que diseñar otro sistema más complejo para hacerlo.

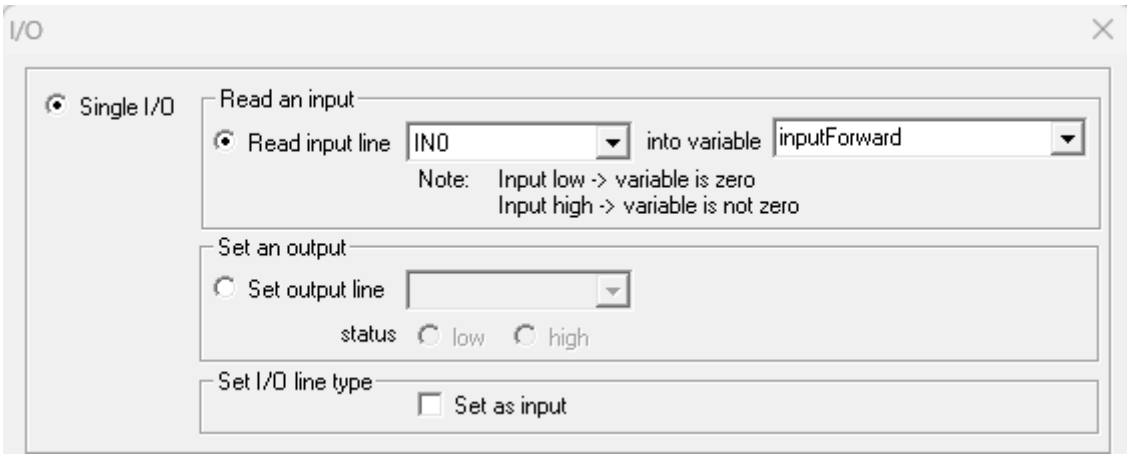


Figura 41. Parte de la pestaña utilizada para declarar entradas y salidas.

4.3.2.4 *Funciones de movimiento*

```
CALL Forward, inputForward, EQ; //Call function Forward if inputForward == 0
CALL Backward, inputBackward, EQ; //Call function Backward if inputBackward == 0
```

Figura 42. Pseudocódigo para acceder a funciones.

Una vez se han obtenido los valores de las señales de entrada se hace una llamada condicional a un par de funciones (figura 42). Estas llamadas hacen referencia a un par de funciones que han sido llamadas "Forward" y "Backward". Los pines de entrada se están utilizando de manera que son activos en baja, esto significa que cuando el valor leído en el pin es 0 la llamada funciona y si el valor leído es 1 la llamada es ignorada.

El sistema de condiciones con el que cuenta este programa es bastante limitado por lo que se ha intentado mantener las condiciones lógicas a las mínimas necesarias. En la figura 43 se puede ver un ejemplo de cómo se crean estas condiciones en este programa:

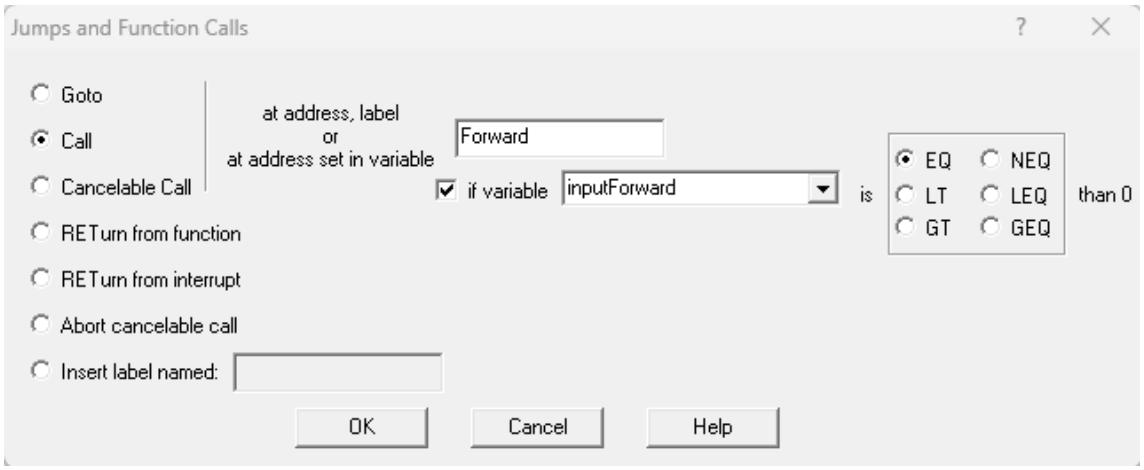


Figura 43. Pestaña en la que se crean llamadas a las funciones.

Ya se ha dicho que estas llamadas hacen referencia a un par de funciones, “Forward” y “Backward” para poder analizar cómo están estructuradas primero hay que encontrarlas en el programa. En la pestaña del programa donde anteriormente se han visto los apartados “Setup” y “Motion” se pueden encontrar otros subapartados pertenecientes a “Motion”, uno de estos subapartados se llama “Functions”, y en su interior es donde se han declarado estas dos funciones. Al igual que en un lenguaje de programación clásico, son partes de código que por una razón u otra han sido escritas por separado. En la figura 44 se puede ver el estado del programa tras hacer clic en una de estas funciones:

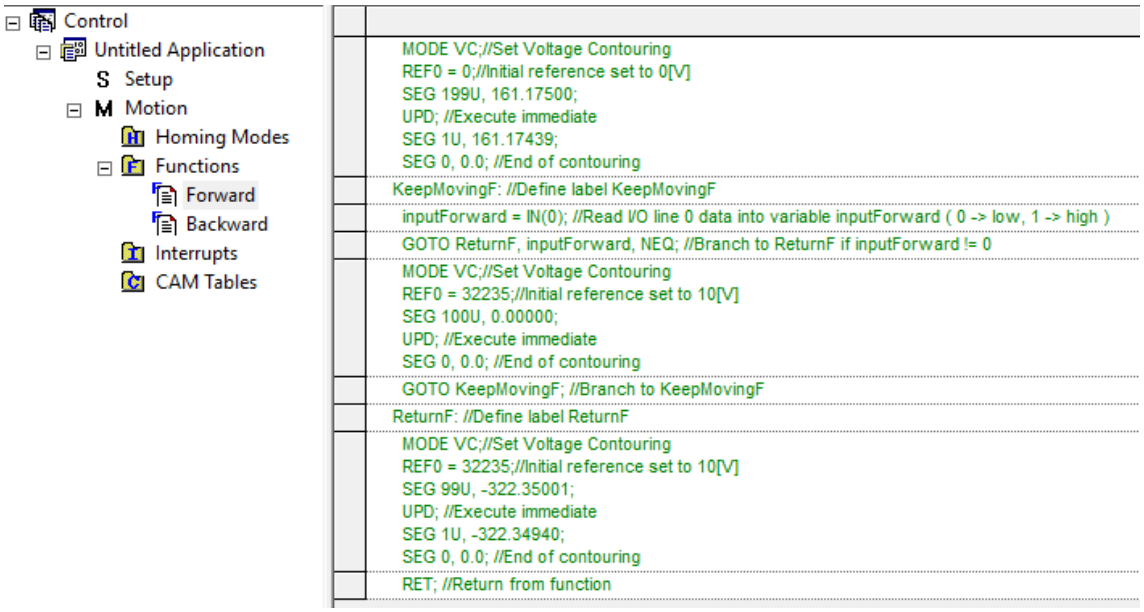


Figura 44. Función "Forward".

En estas funciones se está perfilando la forma de la señal que va a ser enviada al motor.

Si se hace clic en los distintos apartados se pueden ver ventanas que muestran con mayor detalle la forma de las señales que están siendo enviadas y la lógica con la que están siendo enviadas.

Si se hace clic en el primer apartado de todos entre otros elementos se puede ver la figura 45:

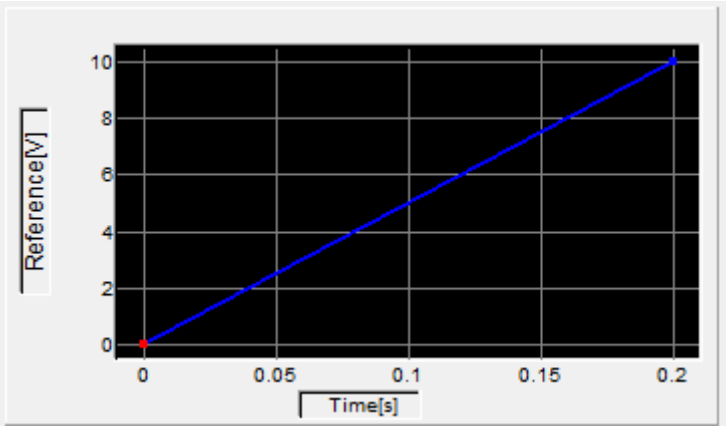


Figura 45. Perfil del primer tramo de la señal enviada al motor.

En ella se muestra la señal que se envía cuando se llega a este apartado a lo largo del tiempo. En este caso es una señal que va aumentando su voltaje de 0 a 10V a lo largo de 0.2 segundos. Esta señal provoca que el motor comience a moverse, el aumento gradual de voltaje se realiza para evitar daños al motor y a la circuitería. Si en lugar de "Forward" estuviéramos viendo "Backward" comprobaríamos que en lugar de ir de 0 a 10V la señal iría de 0 a -10V para lograr el movimiento en el sentido contrario.

Una vez el voltaje ya está en el nivel deseado se continúa a las siguientes partes del código (figura 46).

```
KeepMovingF: //Define label KeepMovingF
inputForward = IN(0); //Read I/O line 0 data into variable inputForward ( 0 -> low, 1 -> high )
GOTO ReturnF, inputForward, NEQ; //Branch to ReturnF if inputForward != 0
MODE VC;//Set Voltage Contouring
REF0 = 32235;//Initial reference set to 10[V]
SEG 100U, 0.00000;
UPD; //Execute immediate
SEG 0, 0.0; //End of contouring
GOTO KeepMovingF; //Branch to KeepMovingF
ReturnF: //Define label ReturnF
```

Figura 46. Parte de la función "Forward".

Este fragmento de código forma un bucle que funciona de mamera similar al bucle principal de este programa. En él se lee el estado actual del pin de entrada relevante para el movimiento que se está realizando y si la señal sigue siendo 0 el movimiento se extiende en el tiempo (figura 47) mientras que si la señal es 1 se sale del bucle y se regresa al bucle principal tras detener el motor (figura 48).

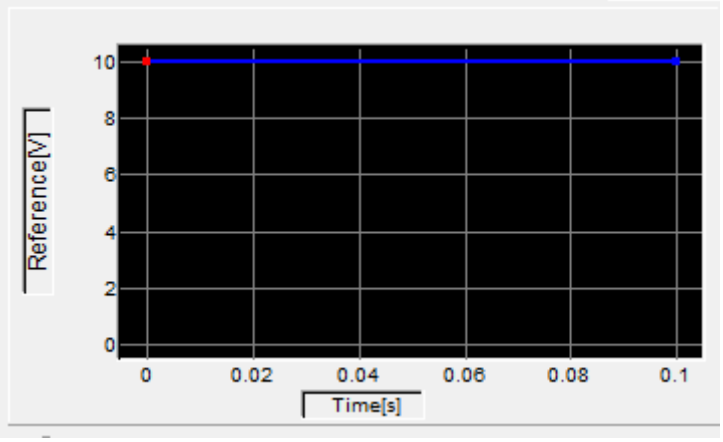


Figura 47. Perfil del tramo extensible de la señal enviada al motor.

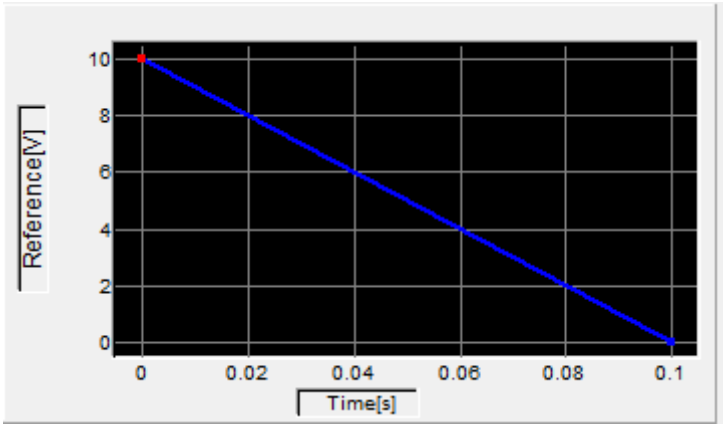


Figura 48. Perfil del tramo final de la señal enviada al motor.

4.3.2.5 GOTO

```
GOTO Start; //Branch to Start
```

Figura 49. Pseudocódigo con un salto GOTO.

Después de las llamadas a las funciones no queda nada que hacer en el bucle, por lo que se utiliza la línea GOTO (figura 48) para regresar a la etiqueta que se había declarado al inicio del programa.

En la figura 50 se puede ver como se crea la línea GOTO.

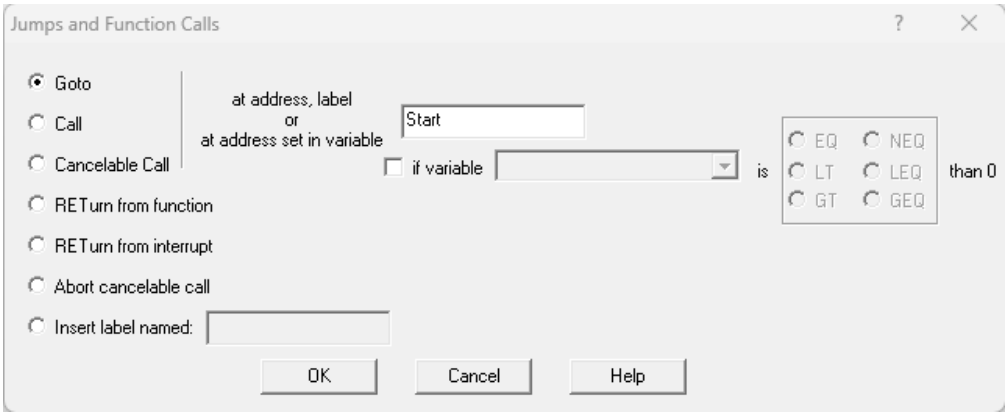


Figura 50. Pestaña de saltos y llamadas creando una línea con un GOTO.

4.4 CAMBIO DE LOS SENSORES

Los sensores que se estaban utilizando inicialmente en este proyecto no van de acuerdo con la dirección y objetivos que se han marcado. Uno de los principales objetivos de este proyecto es crear una alternativa asequible pero funcional a otras prótesis. Los sensores que se estaban utilizando originalmente en este proyecto eran los sensores EMG estándar de la empresa Ottobock (figura 51).

Figura 51. Electrodos de la marca Ottobock [5](Electrode | Ottobock US Shop*, 2024)*.

Estos sensores son de una calidad excelente y son considerados uno de los mejores sensores EMG existentes para uso práctico en prótesis. Por desgracia, una pareja de estos sensores alcanza valores cercanos a los 3.000 €. Esto significaría que en caso de usar estos sensores se estaría gastando solo en ellos más del doble de lo invertido en producir el resto de la prótesis. Por esta razón se ha buscado una alternativa asequible que, si bien no va a alcanzar el nivel de calidad de la marca Ottobock, pueda servir para las dimensiones de este proyecto.

Para este prototipo se han utilizado los sensores “Gravity” de OYMotion, que cuestan alrededor de 50 € la unidad (figura 52). Como es razonable, estos sensores no tienen la misma calidad que los sensores de Ottobock, por lo que en este apartado se va a habla de los problemas que ha supuesto esta bajada en la calidad del producto y que soluciones se plantean.

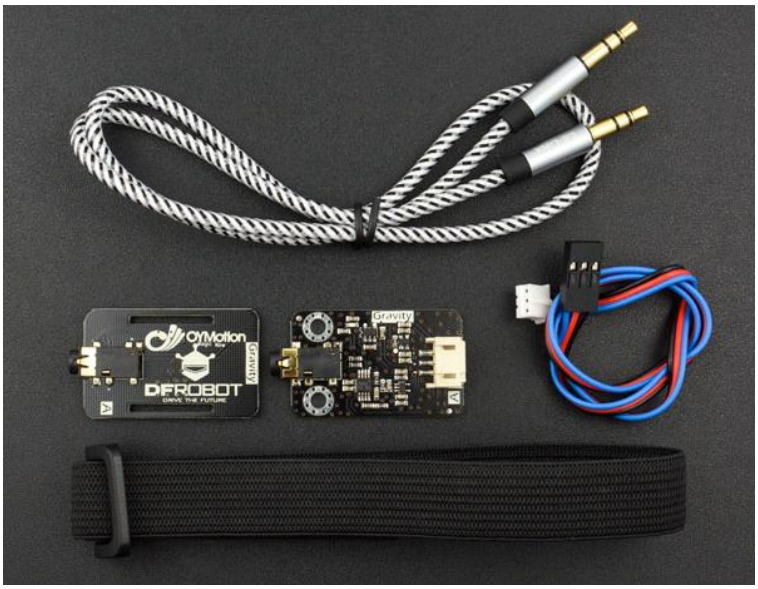


Figura 52. Sensor "Gravity" de OYMotion [1](Analog_EMG_Sensor_by_OYMotion_SKU_SEN0240-DFRobot*, s. f.)*.

El principal defecto que se observa en estos sensores con respecto a los de Ottobock es la separación del sensor y de la circuitería de amplificación. Si se utiliza el cable que está incluido con el sensor uno se puede dar cuenta que medir señales delante de una mesa de trabajo es sencillo y el sensor no da problemas, pero si nos intentamos desplazar y se introduce movimientos innecesarios en el sistema, el movimiento del cable y la circuitería provoca interferencias que pueden provocar movimientos de la prótesis no deseados. Se han realizado pruebas con cables más cortos para comprobar el efecto que este cambio tendría es estas interferencias y se ha observado una reducción importante en la aparición de estas. En pruebas futuras sería interesante encontrar una manera de conectar directamente el sensor y su circuitería para estudiar los resultados.

Hay otro defecto digno de mención que, si bien también afectaba en menor medida al sensor de Ottobock, se ha visto más acentuado en este sensor. Las condiciones en las que se debe utilizar un sensor EMG para lograr resultados adecuados son engorrosas de replicar cuando se realizan pruebas en un laboratorio. Para que el sensor estuviera en condiciones ideales se debería limpiar adecuadamente la zona en la que va a ir el sensor y se debería eliminar el pelo en caso de haberlo, esto se hace para reducir todo lo posible la impedancia de la piel. Además, el sensor debería de ser fijado adecuadamente utilizando parches adhesivos en lugar de una goma como se ha utilizado en el laboratorio. Esto evita que uno de los electrodos se pueda separar de la piel, razón de un importante porcentaje de los errores que se han dado cuando se han realizado pruebas con la prótesis.

4.5 INTRODUCCIÓN DE UNA ALIMENTACIÓN PORTATIL.

Con la introducción de las mejoras planteadas en los apartados anteriores se ha logrado una mano que responde como esperamos a los datos detectados por los sensores. Esto significa que la prótesis ya funciona, pero no es suficiente con que funcione, una prótesis sirve de poco si no puedes llevarla contigo. Para lograr que la prótesis se pueda transportar se ha buscado sustituir la fuente de alimentación DC variable que se estaba utilizando por una batería.

La batería que se está buscando debe adecuarse a las necesidades de nuestro circuito, en este caso queremos una batería con una diferencia de potencial de unos 12V y con la capacidad de suministrar corrientes de hasta 2 A, las corrientes van a rondar típicamente los 0.2 A, pero 2 A son suficientes para asegurar que no ocurran problemas.

Figura 53. Batería DC utilizada en el prototipo de la prótesis [6](Eleoption - DC-168, CD 12 V, 1800 mAh*, s. f.)*.

En la figura 53 se puede ver la batería que se ha utilizado finalmente, no está pensada como una solución final, pero es suficiente para realizar pruebas con una batería portátil.

5 PRUEBAS Y RESULTADOS

A lo largo del desarrollo se han realizado varias pruebas de funcionamiento, pero las más relevantes son las que han sido realizadas en última instancia.

Durante las pruebas de este diseño se han observado los siguientes problemas y resultados:

- La prótesis responde muy bien a las intenciones del usuario, los intentos de mover la mano por parte del usuario nunca se suelen ver ignorados.
- La mano puede agarrar objetos sin problema, el material utilizado no tiene un agarre demasiado bueno, pero cuando se han realizado pruebas poniendo un guante a la prótesis se podían coger una gran variedad de objetos sin problema.
- La mano es bastante fuerte, ha podido levantar objetos de hasta 2 Kg de múltiples maneras sin demasiados problemas. Si la posición y forma del objeto es la adecuada puede levantar objetos de hasta 4 Kg por si sola.
- La electrónica de los sensores es muy sensible al movimiento, si se siguen utilizando los sensores que se está usando ahora mismo hay que encontrar una manera de asegurarlos para que no provoquen ruido.
- Los sensores provocan señales erróneas cuando no están en contacto con la piel, habría que encontrar una manera de evitar el movimiento del motor en estas circunstancias. Con, por ejemplo, un parche adhesivo.
- La mano puede realizar movimientos destructivos, habría que encontrar la manera de evitar esta posibilidad por el bien de la integridad del producto.
- Los sensores deben de ser calibrados cuando los va a utilizar una nueva persona, la Raspberry debe estar conectada a un ordenador para realizar esta calibración.

Estas pruebas esclarecen algunos problemas del prototipo que deberían ser resueltos, pero estos serán tratados en más profundidad en un apartado posterior. Los resultados de las pruebas realizadas han sido generalmente positivos, la mano no ha perdido capacidades con respecto al prototipo anterior, ahora cuenta con la capacidad para mover la articulación de la pinza con precisión y está haciendo todo esto sin depender de ningún elemento de la mesa de trabajo.

5.1 COSTE ECONÓMICO

A continuación, se va a dar un resumen del coste de producción del prototipo realizado, esto se hace para comprobar si el objetivo de crear una opción más asequible que otras opciones del mercado ha sido realizado. Se recuerda que el precio de una prótesis de mano pediátrica ronda los 8.000 € / 15.000 €.

- Coste del controlador del motor. 265 €
- Coste del motor. 480 €
- Coste Raspberry pico. 5 €
- Pines y componentes de la placa. 15 €
- Batería. 20 €
- Sensores EMG. 100 €

- Costes de producción no considerados (impresión 3D, Creación PCB). 50 €

Con esta estimación el precio de producción actual de la prótesis es de 935 €. Si bien se espera que en futuras etapas del proyecto este precio pueda aumentar, aunque este es un precio de producción y no de venta, en comparación con el precio de la prótesis pediátrica promedia se ha conseguido una reducción de precio considerable.

6 CONCLUSIONES Y LINEAS FUTURAS DE TRABAJO

6.1 CONCLUSIONES

Cuando se ha comenzado este proyecto se han planteado varios objetivos a cumplir. Se ha logrado que la prótesis sea completamente portátil y ya no depende de ninguno de los elementos del banco de trabajo, por otro lado, el control es decente pero los sensores siguen resultando un problema y su funcionamiento no es siempre el esperado. El proyecto sigue estando relativamente verde, todavía lejos de una versión que se pueda probar con pacientes, por no hablar de una versión comercial.

A nivel personal este proyecto me ha ayudado a lograr una mejor disciplina a la hora de enfrentar grandes proyectos y me ha enseñado como separarlos en pequeños apartados manejables. También me ha permitido ganar algo de experiencia en el campo de la robótica, en el cual estoy muy interesado.

6.2 LÍNEAS FUTURAS DE TRABAJO

Aún quedan muchos aspectos de la prótesis por pulir y desarrollar por esa razón se van a presentar aquí varios aspectos que quedan pendientes sobre los que se puede trabajar.

- Revisar el diseño de la mano. Las pruebas realizadas a lo largo de este trabajo han demostrado que el diseño actual de algunas piezas que forman la mano debe ser revisadas, en especial la pieza superior de la pinza, que es desplazada por el motor debe ser mejorada ya que el motor las desgasta hasta dejarlas inservibles en un periodo muy corto de tiempo.
- Arreglar el encoder. El encoder del motor no funciona como debería, gracias al programa de Technosoft se ha podido comprobar que el encoder no está funcionando como uno esperaría, por desgracia no cuento con los conocimientos necesarios para enfrentar este problema, arreglar el encoder permitiría poner medidas para evitar movimientos destructivos por parte de la mano.
- Miniaturización de la electrónica. Las conexiones y diseño de la electrónica ya están planteadas y se ha demostrado que funcionan. El diseño actual fue creado con la inmediatez en mente y ha terminado siendo uy voluminoso. El uso de tecnologías más modernas, como una PCB con más de un sustrato, pueden reducir considerablemente el espacio necesario para guardar la circuitería de la prótesis.
- Diseño de una interfaz de usuario. Hay variables en el código de la prótesis que debe ser manipuladas de manera frecuente, en particular se está hablando de las variables que permiten calibrar los sensores. En la versión actual estos valores se modifican en el código y luego se guardan en la memoria de la Raspberry. Crear una interfaz que permita hacer esto sin interactuar directamente con el código no es una prioridad, pero es algo que habría que hacer en algún momento.

7 REFERENCIAS

[1] Analog_EMG_Sensor_by_OYMotion_SKU_SEN0240-DFRobot. (s. f.). Recuperado 2 de mayo de 2024, https://wiki.dfrobot.com/Analog_EMG_Sensor_by_OYMotion_SKU_SEN0240

[2] Brito, J., Quinde, M., Cusco, D., & Calle, J. (2013). Estudio del estado del arte de las prótesis de mano. Ingenius, 9, Article 9. https://doi.org/10.17163/ings.n9.2013.08

[3] Calle del Río, C. (2023). "Diseño, prototipado y programación de una prótesis de mano mediante impresión 3D" [info:eu-repo/semantics/bachelorThesis]. https://uvadoc.uva.es/handle/10324/60362

[4] Dumitru, V. (s. f.). Intelligent Drives IPOS3604. Technosoft Motion Control. Recuperado 2 de mayo de 2024, https://technosoftmotion.com/en/intelligent-drives/?&SingleProduct=10

[5] Electrode | Ottobock US Shop. (2024, abril 29). https://shop.ottobock.us/c/Electrode/p/13E200~550

[6] Eleoption—DC-168, CD 12 V, 1800 mAh—Batería para cámara CCTV inalámbrica, monitor para bebé: Amazon.es: Electrónica. (s. f.). Recuperado 2 de mayo de 2024, https://www.amazon.es/dp/B07546GMCT/ref=sspa_dk_detail_1?psc=1&pd_rd_i=B07546GMCT&pd_rd_w=96bhJ&content-id=amzn1.sym.b83bda31-8587-4f3d-8cd8-9f9ef93265d3&pf_rd_p=b83bda31-8587-4f3d-8cd8-9f9ef93265d3&pf_rd_r=TM0RPNVZ311RJPTDWSWM&pd_rd_wg=BCJoK&pd_rd_r=72272f7d-404c-4717-a49d-85af3d65e883&sp_csd=d2lkZ2V0TmFtZT1zcF9kZXRhaWxfdGhlbWF0aWM

[7] Historias con Historia » Blog Archive » Götz 'Mano de Hierro' Berlichingen. (2010, junio 30).

https://web.archive.org/web/20100630134842/http:/historiasconhistoria.es/2008/07/25/gotz-mano-de-hierro-berlichingen.php

[8] iPOS360x MX-CAN - User Manual. (2024, abril 29). https://technosoftmotion.com/wp-content/uploads/P091.028.iPOS360x.MX_.CAN_.CAT_.UM_.pdf

[9] Issues · oymotion/EMGFilters. (2024, abril 29). GitHub. https://github.com/oymotion/EMGFilters

[10] Konrad, P. (s. f.). A Practical Introduction to Kinesiological Electromyography.

[11] Ltd, R. P. (2024, abril 29). Buy a Raspberry Pi Pico. Raspberry Pi. https://www.raspberrypi.com/products/raspberry-pi-pico/

[12] Make the filter state instance data by edgar-bonet · Pull Request #4 · oymotion/EMGFilters · GitHub. (2024, abril 29). https://github.com/oymotion/EMGFilters/pull/4

[13] Montane, F., & Muñoz, J. (2018). EL DESARROLLO DE LA PROTÉSICA A LO LARGO DE LA HISTORIA HUMANA 1 2018. Ortesis, Protesis y Movilidad.

[14] McCauley, B., Maxwell, D., & Collard, M. (2018). A Cross-cultural Perspective on Upper Palaeolithic Hand Images with Missing Phalanges. Journal of Paleolithic Archaeology, 1(4), Article 4. https://doi.org/10.1007/s41982-018-0016-8

[15] Ober, J. (1982). Upper limb prosthetics for high level arm amputation. Prosthetics and orthotics international, 6, 17-20. https://doi.org/10.3109/03093648209167733

[16] Otto Bock Electrohand 2000 | Myo Terminal Devices | Myo Hands and Components | Upper Limb Prosthetics | Prosthetics | Ottobock US Shop. (s. f.). Recuperado 2 de mayo de 2024, https://shop.ottobock.us/Prosthetics/Upper-

Limb-Prosthetics/Myo-Hands-and-Components/Myo-Terminal-Devices/Otto-Bock-Electrohand-2000/p/8E51

[17] Raspberry Pi Pico Datasheet. (s. f.).

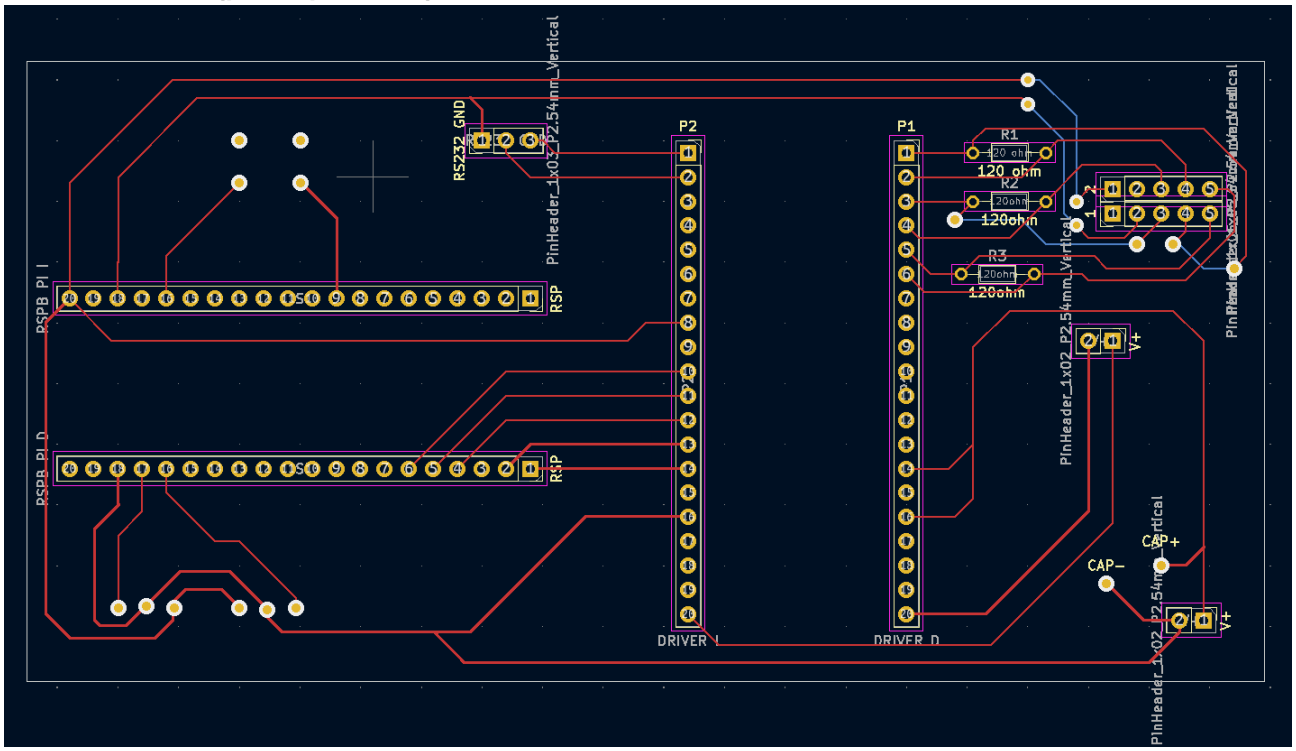
[18] The Complete Guide To Arm & Hand Amputations and Prosthetics | MCOP. (s. f.). MCOP Prosthetics. Recuperado 2 de mayo de 2024, https://mcopro.com/blog/resources/arm-hand-prosthetics/

[19] Trent, L., Intintoli, M., Prigge, P., Bollinger, C., Walters, L., Conyers, D., Miguelez, J., & Ryan, T. (2019). A narrative review: Current upper limb prosthetic options and design. Disability and Rehabilitation: Assistive Technology, 15, 1-10. https://doi.org/10.1080/17483107.2019.1594403

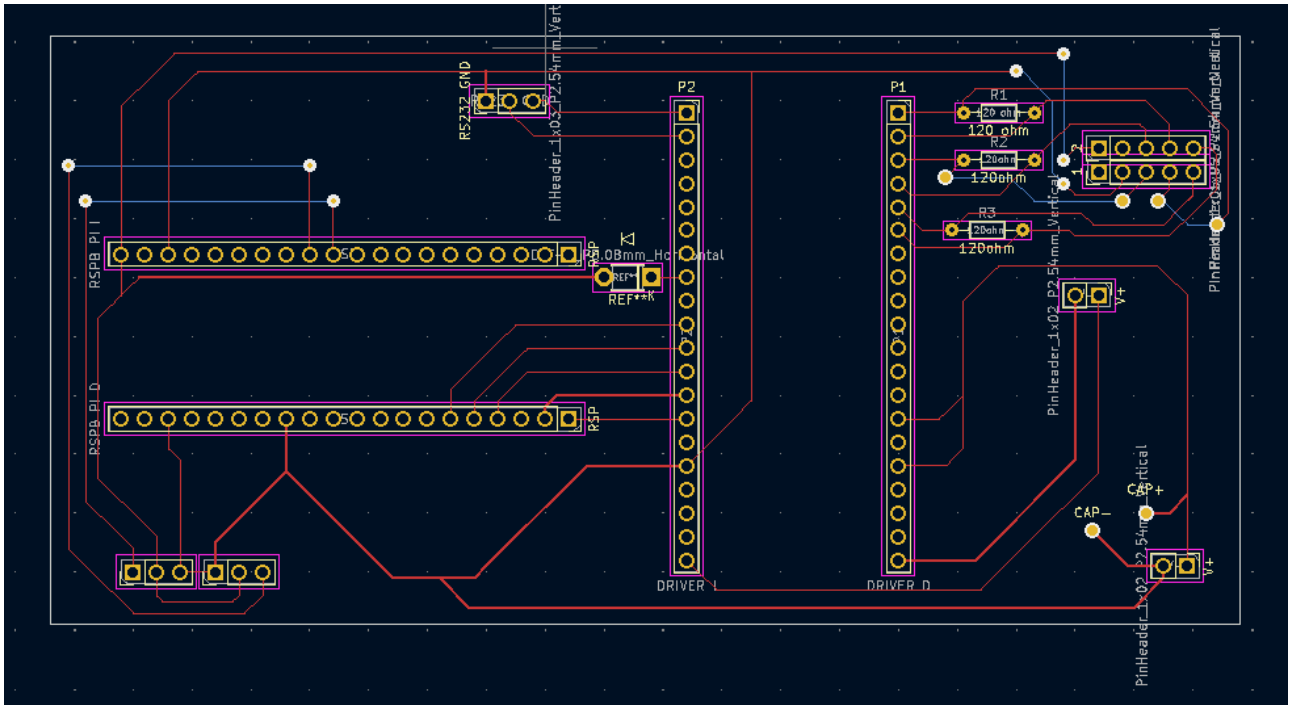
8 ANEXOS

8.1 DISEÑOS DE PCB

8.1.1 Diseño 1 (prototipo fallido)



8.1.2 Diseño 2



8.2 CÓDIGO RASPBERRY PI PICO

```
#include <EMGFilters.h>

#define timerDebug  1

#define sensorIn1  A1 //Se da un nombre a las entradas analógicas de la placa que van a ser utilizadas
#define sensorIn2  A0

const int controllerCom1 = 11;  //Se da un nombre a las salidas digitales
const int controllerCom2 = 12;
const int controllerCom3 = 13;
const int controllerCom4 = 14;
const int controllerCom5 = 15;

EMGFilters filtroSen1;  //Se crea una pareja de instancias de la biblioteca EMGFilters para poder utilizar los filtros de esta
EMGFilters filtroSen2;

SAMPLE_FREQUENCY sampleRate = SAMPLE_FREQ_1000HZ; //Estos términos van a ser utilizador para la creacción de los filtros, determinan la velocidad de muestreo y la de filtrado
NOTCH_FREQUENCY humFreq = NOTCH_FREQ_50HZ;

static int Throhold1 = 1500;  //Esta pareja de valores determinan como de grande debe ser una señal detectada por los sensores para que se considere una detección, se deben calibrar con cada uso de la prótesis
static int Throhold2 = 3000;

static int Failureth1 = 15000;
static int Failureth2 = 15000;

unsigned long timeStamp;  //Estas variables se usan en la interfaz que se está utilizadon para visualizar la entrada de los sensores
unsigned long timeBudget;

int keepAlive1 = 0; //Estas variables se utilizan para convertir la erratica salida de los sensores en una señal más fácil de interpretar para el controlador del motor
int keepAlive2 = 0;

int keepAliveHighTH = 100;
int keepAliveLowTH = 50;

int conectAxis = 1; //Flag utilizada para determinar si los valores detectados se envían o no al motor, utilzada para la calibración de los sensores

void setup() {
  Serial.begin(115200);

  filtroSen1.init(sampleRate,humFreq,true,true,true); //Se inician los filtros que se van a usar sobre la entrada de los sensores
  filtroSen2.init(sampleRate,humFreq,true,true,true);

  timeBudget = 1e6 / sampleRate;

  pinMode(sensorIn1, INPUT);  //Se definen los pines de los sensores como entradas
  pinMode(sensorIn2, INPUT);

  pinMode(controllerCom1, OUTPUT); //Se definen los pines conectados al controlador del motor como salidas
  pinMode(controllerCom2, OUTPUT);
  pinMode(controllerCom3, OUTPUT);
  pinMode(controllerCom4, OUTPUT);
  pinMode(controllerCom5, OUTPUT);

  digitalWrite(controllerCom1, HIGH); //Se ponen todas las salidas en alto ya que las entradas del driver del motor son activas en baja
  digitalWrite(controllerCom2, HIGH);
  digitalWrite(controllerCom3, HIGH);
  digitalWrite(controllerCom4, HIGH);
  digitalWrite(controllerCom5, HIGH);
}

void loop() {

  timeStamp = micros();

  //En este primer bloque del bucle se leen las entradas del sensor y se filtran sus valores

  int valorSen1 = analogRead(sensorIn1);
  int valorFiltrado1 = filtroSen1.update(valorSen1);
  int envlope1 = sq(valorFiltrado1);

  int valorSen2 = analogRead(sensorIn2);
  int valorFiltrado2 = filtroSen2.update(valorSen2);
  int envlope2 = sq(valorFiltrado2);

  // En este segundo bloque se toman varias decisiones lógicas, si se considera que alguno de los valores detectados en los sensores es válido (envelope > Throhld)
  // se envía una señal al controlador del motor para que abra o cierre la mano y se aumenta el valor de la variable keep alive, cuando se deja de detectar una señal válida,
  // se esperan los bucles necesarios para que la variable keepAlive llegue a 0 y se desactiva la señal que mueve el motor

  if((envlope1 > Throhold1) && (envlope1 < Failureth1)){
    if((conectAxis == 1) && (keepAlive1 > keepAliveLowTH))
      digitalWrite(controllerCom5, LOW);
    if(keepAlive1 < keepAliveHighTH){
      keepAlive1 = keepAlive1 + 10;
    }
  }
  else{
    if(keepAlive1 == 0){
      digitalWrite(controllerCom5, HIGH);
      envlope1 = 0;
    }
```

```
    else{
      keepAlive1 = keepAlive1 - 1;
    }
  }

  if((envlope2 > Throhold2) && (envlope2 < Failureth2)){
    if((conectAxis == 1) && (keepAlive2 > keepAliveLowTH))
      digitalWrite(controllerCom4, LOW);
    if(keepAlive2 < keepAliveHighTH){
        keepAlive2 = keepAlive2 + 10;
    }
  }
  else{
    if(keepAlive2 == 0){
      digitalWrite(controllerCom4, HIGH);
      envlope2 = 0;
    }
    else{
      keepAlive2 = keepAlive2 - 1;
    }
  }

  //Cambiar el valor impreso entre envlope1 y envlope2 para calibrar ambos sensores, este fragmento del código permite ver envelope1 y 2 cuando se está conectado a un ordenador
  timeStamp = micros() - timeStamp;
  if (timerDebug){
    Serial.print("Squared Data: ");
    Serial.println(envlope2);
  }

  delayMicroseconds(1000);
}
```

8.3 CÓDIGO CONTROLADOR IPOS-3602

Main

```
int inputForward; // Define integer variable inputForward
int inputBackward; // Define integer variable inputBackward
Start: //Define label Start
inputForward = IN(0); //Read I/O line 0 data into variable inputForward ( 0 -> low, 1 -> high )
inputBackward = IN(1); //Read I/O line 1 data into variable inputBackward ( 0 -> low, 1 -> high )
CALL Forward, inputForward, EQ; //Call function Forward if inputForward == 0
CALL Backward, inputBackward, EQ; //Call function Backward if inputBackward == 0
GOTO Start; //Branch to Start
STOP; // Stop motion with acceleration / deceleration set
```

Function Forward

```
MODE VC;//Set Voltage Contouring
REF0 = 0;//Initial reference set to 0[V]
SEG 199U, 161.17500;
UPD; //Execute immediate
SEG 1U, 161.17439;
SEG 0, 0.0; //End of contouring
KeepMovingF: //Define label KeepMovingF
inputForward = IN(0); //Read I/O line 0 data into variable inputForward ( 0 -> low, 1 -> high )
GOTO ReturnF, inputForward, NEQ; //Branch to ReturnF if inputForward != 0
MODE VC;//Set Voltage Contouring
REF0 = 32235;//Initial reference set to 10[V]
SEG 100U, 0.00000;
UPD; //Execute immediate
SEG 0, 0.0; //End of contouring
GOTO KeepMovingF; //Branch to KeepMovingF
ReturnF: //Define label ReturnF
MODE VC;//Set Voltage Contouring
REF0 = 32235;//Initial reference set to 10[V]
SEG 99U, -322.35001;
UPD; //Execute immediate
SEG 1U, -322.34940;
SEG 0, 0.0; //End of contouring
RET; //Return from function
```

Function Backward

```
MODE VC;//Set Voltage Contouring
REF0 = 0;//Initial reference set to 0[V]
SEG 199U, -161.17500;
UPD; //Execute immediate
SEG 1U, -161.17439;
SEG 0, 0.0; //End of contouring
KeepMovingB: //Define label KeepMovingB
inputBackward = IN(1); //Read I/O line 1 data into variable inputBackward ( 0 -> low, 1 -> high )
GOTO ReturnB, inputBackward, NEQ; //Branch to ReturnB if inputBackward != 0
MODE VC;//Set Voltage Contouring
REF0 = -32235;//Initial reference set to -10[V]
SEG 100U, 0.00000;
UPD; //Execute immediate
SEG 0, 0.0; //End of contouring
GOTO KeepMovingB; //Branch to KeepMovingB
ReturnB: //Define label ReturnB
MODE VC;//Set Voltage Contouring
REF0 = -32235;//Initial reference set to -10[V]
SEG 199U, 161.17500;
UPD; //Execute immediate
SEG 1U, 161.17439;
SEG 0, 0.0; //End of contouring
RET; //Return from function
```