Universidad Zaragoza
1542

Trabajo Fin de Grado

Modelado y Control de una prótesis robótica mediante sensores EMG y la plataforma a Raspberry-Pi.

Modeling and Control of a robotic prothesis using EMG sensors and the Raspberry-Pi platform

Autor

Eduardo Salcedo Puyo

Directores

Jesús Sergio Artal Sevil
José Antonio Domínguez Navarro

Titulación del autor

Grado Universitario en Ingeniería Electrónica y Automática

Universidad de Zaragoza Escuela de Ingeniería y Arquitectura
Departamento de Ingeniería Eléctrica

2022

AGRADECIMIENTOS

Me gustaría aprovechar este espacio para agradecer a Sergio Artal y a Carlos Millán toda la disposición y dedicación en la dirección de este Trabajo de Fin de Grado. Sin olvidar toda la ayuda, recibida, la paciencia y el buen trato recibido durante todos estos meses.

Agradecer a mi familia todo el cariño el apoyo y la confianza en mi depositada y a mis amigos por todos los momentos pasados juntos.

INDICE

1. Introducción

1.1 Motivación y contexto

Según el "Centro Nacional de la Información sobre la Pérdida de la Extremidad", anualmente, en Estados Unidos se realizan alrededor de 185 mil amputaciones de alguna extremidad. De estas, un 82% se realizan por causas patológicas, siendo las más frecuentes enfermedades vasculares y la diabetes, y un 18% a causa de un traumatismo como podría ser un accidente laboral. En definitiva, la amputación ocurre cuando no hay otra alternativa y el paciente debe decidir entre la pérdida de una extremidad o el riesgo de perder la vida. [1].

Como respuesta, a lo largo de la historia el ser humano ha desarrollado multitud de dispositivos y artilugios con el fin de reemplazar los miembros faltantes. Desde las rudimentarias piezas protésicas utilizadas por los antiguos egipcios, que únicamente aportaban una sensación de "completitud", pasando por la búsqueda funcional del presente, hasta el prometedor futuro inminente que se nutre de las emergentes tecnologías de la "Industria 4.0". [2]

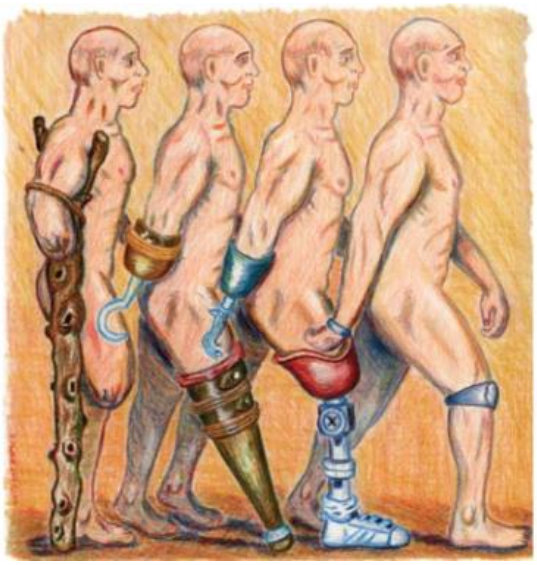


Figura 1.1 *Ilustración de Scott McNutt (Kim Norton. "A Brief History of Prosthetics". 2007)*

Hasta ahora, la gran mayoría de las prótesis utilizadas por los pacientes son pasivas, es decir, sus accionamientos son puramente mecánicos. Se han alcanzado grandes resultados con este tipo de prótesis para amputaciones en extremidades inferiores, sin embargo, este tipo de prótesis ha llegado a su techo con unos resultados mucho menos satisfactorios en el caso de las amputaciones superiores. Dada la complejidad biomecánica de las extremidades superiores humanas y la gran variedad de movimientos que realizan, las investigaciones están siguiendo la tendencia de implementar elementos activos en este tipo de prótesis. En consecuencia, estos elementos activos pueden ser controlados de una manera preprogramada o mediante sensores que capturan algún tipo de señal bioeléctrica (EEG, EMG) del usuario. Esta segunda opción, dota al usuario de un control y una experiencia más realista, sin embargo, para un correcto funcionamiento, tanto el desarrollo del control, como el modelo físico de cada prótesis, necesita ser totalmente personalizado para cada paciente. Dado que esto imposibilita la fabricación a

gran escala, los costes se elevan hasta imposibilitar el acceso a la gran mayoría de la población. [3]

Sin embargo, la aparición de la impresión 3D ha revolucionado súbitamente la manera de fabricar prótesis. Aunque es cierto que esta tecnología existe desde los años 80, no ha sido hasta la actualidad cuando esta tecnología se ha asentado en el mercado e incluso ha llegado a los hogares particulares. La revolución que supone esta tecnología radica en su capacidad de creación de geometrías complejas antes imposibles con otros métodos de fabricación, al mismo tiempo que su rapidez y bajo costo. Del mismo modo, la inteligencia artificial está en un periodo de crecimiento repentino que promete un cambio de paradigma en la industria. Cada vez es mayor la tendencia a dotar a las máquinas del llamado autoaprendizaje para realizar acciones personalizadas en las que antes se utilizaban softwares complejos y rígidos con un alto coste computacional.

Figura 1.2 *Proyecto "Astro Dog" fabricado en 3D y controlado con AI*

1.2 Objetivos

Por todo lo mencionado anteriormente, en el presente proyecto se trata de analizar el potencial y viabilidad de la combinación de estas y otras tecnologías para su aplicación en el diseño y fabricación de una prótesis de bajo costo. Por un lado, la impresión 3D proporciona al producto la personalización física tan necesaria para esta aplicación, mientras que, con la inteligencia artificial, la prótesis podría "aprender" de los movimientos del propio paciente para así desarrollar un sistema de control individualizado.

Los objetivos principales de este proyecto son entonces, el rediseño, impresión y montaje de una prótesis 3D OpenSource, el estudio e implementación de diferentes sensores para dotar a la prótesis de mayor capacidad de interacción con el entorno, el diseño e implementación del control de diferentes tipos de agarre y el diseño del control de la prótesis mediante redes neuronales.

Cabe destacar que, con objeto del satisfactorio desarrollo del proyecto, se han diseñado y construido diferentes prototipos y circuitos impresos que permiten realizar las pruebas necesarias para el correcto análisis y comparación de todas las soluciones planteadas.

2. Electromiografía - EMG

La electromiografía consiste en la captación, registro y análisis del potencial eléctrico producido por la contracción de las fibras musculares. Desde mediados de los 80 se lleva utilizando esta técnica para el diagnóstico de enfermedades neuromusculares y la monitorización de la activación muscular de un paciente. Sin embargo, en la última década, cada vez es más común su uso para el control de prótesis, dando lugar a las llamadas prótesis mioeléctricas.

Existen dos principales métodos para la obtención de estas señales, el intramuscular (invasivo) y el superficial (no invasivo) (Figura.2.1). El primer método consiste en la inserción de una aguja-electrodo entre el tejido muscular y el segundo consiste en la colocación de electrodos de un material altamente conductivo que se adhiere a la piel del paciente. Dado que el método intramuscular puede ser demasiado invasivo y se necesita de un profesional para su implantación, en este trabajo se utiliza el método no invasivo. [4]

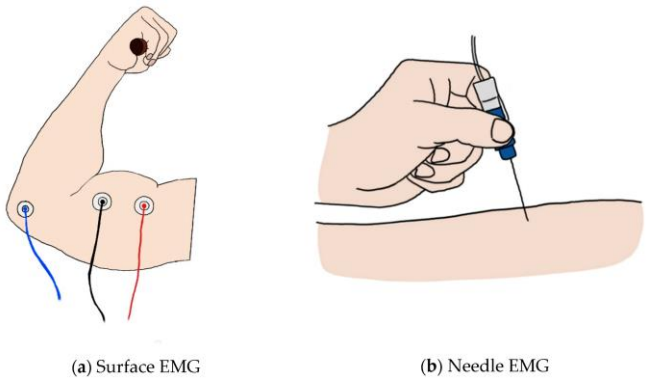


Figura.2.1 *EMG superficial / EMG intramuscular*

2.1. Fundamentos de Adquisición de señales EMG

Como se ha comentado anteriormente, estos sensores, captan la diferencia de potencial que generan las fibras musculares cuando se contraen, de esta manera, analizando la presencia, tamaño y forma de onda, se obtiene información acerca de la respuesta muscular ante determinados estímulos nerviosos. Dada la naturaleza de las señales producidas por las fibras, se necesita cumplir una serie de requisitos en cuanto a la calidad, uso y ubicación de los electrodos. Del mismo modo y por el mismo motivo, se necesita un procesado de señal específico para poder extraer información relevante de las señales captadas. A lo largo de este capítulo se tratarán en profundidad tanto la preparación de los electrodos como el tratado de señal mencionado. [4]

2.1.1. Los electrodos

El electrodo es el componente que está en contacto con la capa superficial de la piel del paciente y debe ser de un material con una muy baja impedancia (normalmente AgCl) (Figura.2.2). Dada la alta impedancia ofrecida por la piel se recomienda eliminar el vello de la dermis y aplicar un gel conductor.

Figura.2.2 *Electrodos ECG*

En nuestro caso utilizaremos un tríodo de electrodos, dos para la medición de la diferencia de potencial del músculo, y otro alejado de ellos en una ubicación libre de fibras musculares (como puede ser el codo o el dorso de la mano) que actuará como referencia.

Según las recomendaciones de la "European Concerted Action Surface EMG for noninvasive assessment of muscle", la colocación de los electrodos positivo y negativo se debe llevar a cabo a una distancia entre sí de no más de 20 mm . Como podemos observar en la Figura.2.3, los electrodos miden el potencial de los dos puntos separados por una distancia d, dando lugar a la ecuación 2.1, siendo n el ruido registrado. Si derivamos la emanación de la señal a lo largo de la piel, v1 y v2 no tienen el mismo valor ya que la señal EMG viaja a unos 3 m/s. Sin embargo, el ruido adicional captado por ambos electrodos es constante y común en ambos puntos. [5]

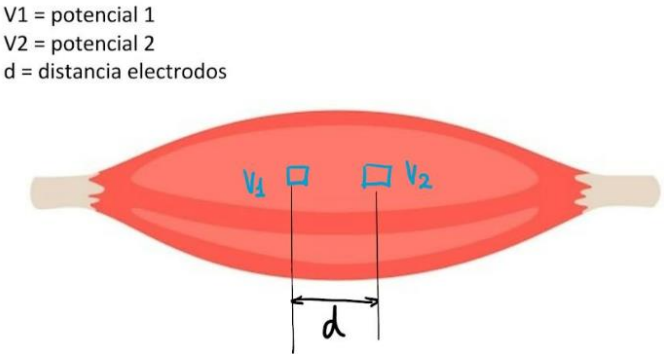


Figura.2.3 *Detalle colocación electrodos*

$$\square\square\square\square = (\square 1 + \square) - (\square 2 + n) \quad (2.1) \qquad \square\square\square\square = \square 1 - \square 2 \quad (2.2)$$

De esta manera, a pesar de que en la práctica siempre existe un ruido permanente, con esta configuración teóricamente el ruido 'n' queda eliminado de la ecuación 2.2, lo cual ayuda en gran medida a disminuir la interferencia producida por la diafonía procedente de otras señales bioeléctricas.

2.1.2 Características de la señal

La señal producida por la contracción de los músculos ronda una amplitud de unos 250μV, aunque en algunos casos puede llegar hasta picos de 6mV y, aunque su canal de información va de los 20 Hz a los 500 Hz, la mayor concentración mayor de energía se encuentra en la franja de entre 50 Hz y 150 Hz [6].

Además de su débil nivel de tensión, la señal está altamente contaminada tanto en altas frecuencias como por el ruido proveniente de fuentes del entorno (típicamente la frecuencia de 60 Hz introducida por la red de distribución eléctrica). Por ello, como hemos mencionado anteriormente, será necesario un preprocesamiento y un procesamiento de la señal captada por los electrodos.

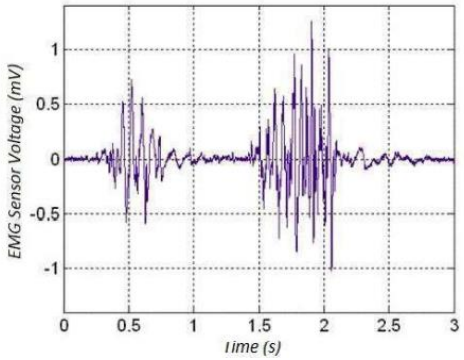


Figura.2.4. *Señal EMG del bíceps contracciones breves (Moritani T. 1978).*

2.1.3 Preprocesamiento de la señal

Para este cometido se ha utilizado el módulo "Grove-EMG Sensor v1.1" [Anexo.C.1] , el cual contiene todas las etapas necesarias para el preprocesamiento de la señal.

Como podemos apreciar en el diagrama que muestra la Figura.2.5, la adecuación de la señal consta de diferentes etapas, las cuales serán analizadas individualmente a continuación. [7]

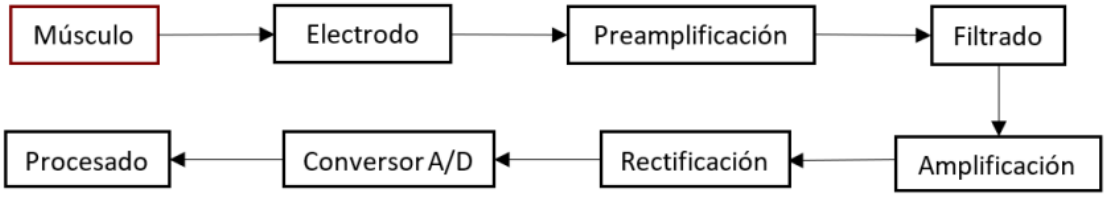


Figura.2.5 *Diagrama preprocesamiento señal*

- **Preamplificación:** Dada la baja amplitud del potencial bioeléctrico captado por los electrodos, es necesaria una preamplificación inicial. A su vez, el ruido acompaña la señal obtenida, superándola hasta varios órdenes de magnitud. Dado

que la aplicación necesita ser lo suficientemente sensible para detectar y amplificar las pequeñas señales detectadas por los electrodos, al mismo tiempo que rechace los ruidos que la acompañan, un amplificador diferencial sería una buena opción ya que nos permite medir la diferencia de potencial entre los 2 electrodos activos respecto al tercero considerado como referencia. En nuestro el amplificador integrado en el módulo para este propósito es el amplificador de instrumentación INA331IDGKT (Texas Instruments)

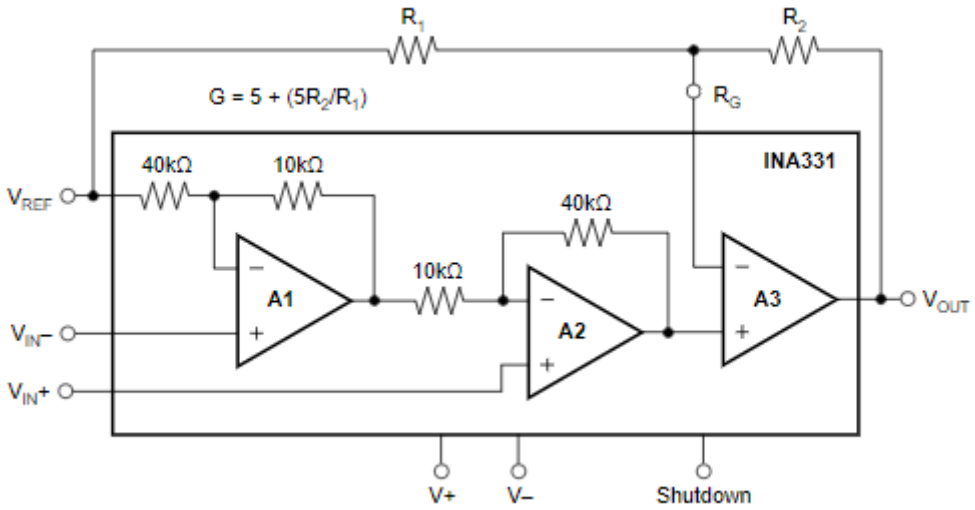


Figura.2.6. *INA331IDGKT*

- **Filtrado:** Con objeto de obtener una señal más "limpia", tras la preamplificación se hará un filtrado de señal que contará con tres partes diferenciadas, filtro pasa-baja, filtro "notch" y filtro pasa-alta.

  Según el datasheet del módulo empleado y de acuerdo con las pruebas experimentales, las etapas de filtrado pasa-baja y pasa-alta dan como resultado un ancho de banda de 20Hz a 500Hz. De esta manera, conseguimos cubrir la totalidad del rango del canal de información de la señal EMG.

  Por otro lado, como ya hemos mencionado al inicio del capítulo, los electrodos son muy sensibles al ruido generado por otros dispositivos electrónicos y en concreto, al ruido generado por la línea de potencia (60Hz). Por ello, es conveniente la introducción de un filtro "notch" que rechace exclusivamente esta banda frecuencial.

- **Amplificación:** Tras el filtrado, para conseguir los niveles de amplitud requeridos, el módulo incorpora una nueva etapa de amplificación con una ganancia regulada (G=5.71) para maximizar el rango dinámico a la entrada del conversor Analógico/Digital.

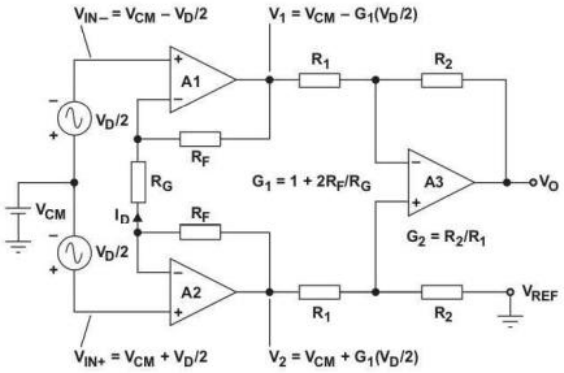


Figura.2.7 *Etapa amplificación*

- **Conversión A/D**

Para terminar la adecuación de la señal, antes de comenzar con el procesado de esta, se llevará a cabo una discretización de la señal analógica.

Para ello se ha utilizado el conversor analógico/digital ya integrado en la tarjeta Arduino Uno. Este conversor tiene una resolución de 10 bits (valores de 0 a 1023) y por tanto teniendo una incertidumbre aproximada en la medición de 4.88 mV.

2.2. Adquisición de señales EMG

Debido a su fácil uso, durante todo el proyecto se ha utilizado la plataforma Arduino para la adquisición de señales. En el caso de las señales EMG la placa utilizada ha sido Arduino Uno. Para esta primera prueba, se ha realizado un pequeño programa de muestreo de la tensión de salida del módulo EMG, por medio de la función *"analogRead(pin)"*. Los valores devueltos tras el paso por el conversor ADC, son valores comprendidos entre 0 y 1023.

Para el correcto funcionamiento del módulo EMG el único requerimiento es alimentarlo a 5V y conectarlo a tierra. El módulo emite una señal de salida analógica que se conecta al pin 23 (A0) para su discretización.

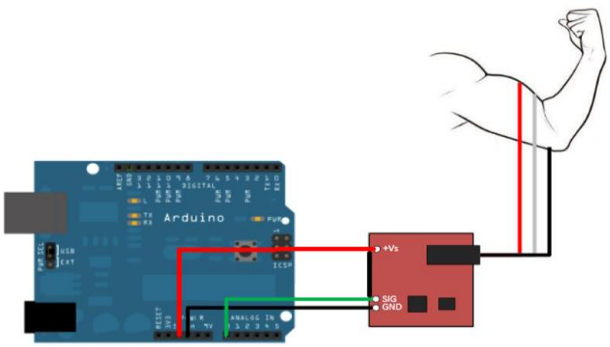


Figura.2.8 *Esquema electrónico captación señales EMG*

2.2.1 Ubicación electrodos

Normalmente, para músculos grandes, la ubicación del par de electrodos positivo y negativo puede colocarse sin una precisión extraordinaria siempre y cuando se respeten las recomendaciones citadas en el apartado 2.1.1. Sin embargo, en nuestro caso, la fisiología de la mano y el antebrazo humano es excepcionalmente compleja. La gran variedad de movimientos simultáneos, complejos y precisos que este órgano es capaz de realizar, ocurre gracias a la disposición de diferentes grupos musculares destinados a movimientos concretos.

Por este motivo, se hace imposible la distinción independiente del movimiento de cada dedo mediante el uso de un único tríodo de electrodos. Sin embargo, se pueden elegir diferentes grupos musculares para colocar varios pares de electrodos, tratando de aislar las señales que activan la flexión de cada uno de los dedos. Aunque, el aislamiento total de estas señales no es posible, existen posiciones concretas en las que la señal de cada dedo llega a su máxima amplitud (Figura.2.9). De este modo, es posible descifrar las diferentes señales implicadas en el movimiento individual de cada dedo. [8]

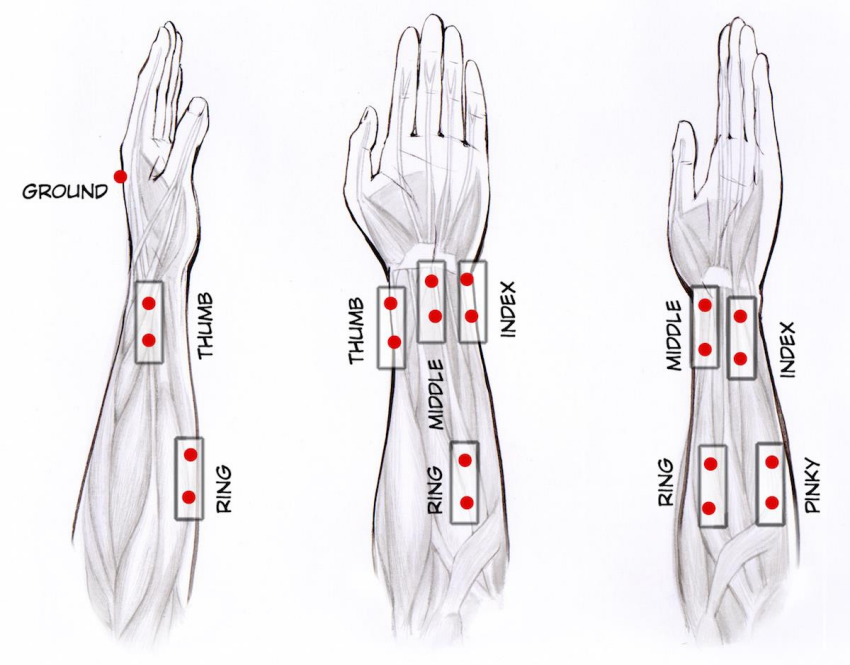


Figura.2.9 *Posicionamiento máxima amplitud cada dedo*

2.2.2 Señales obtenidas

La gráfica representada en la Figura.2.10, corresponde a la señal procedente de los electrodos situados en la posición del pulgar, en este caso concreto, se ha realizado una serie de contracciones y relajaciones del dedo pulgar.

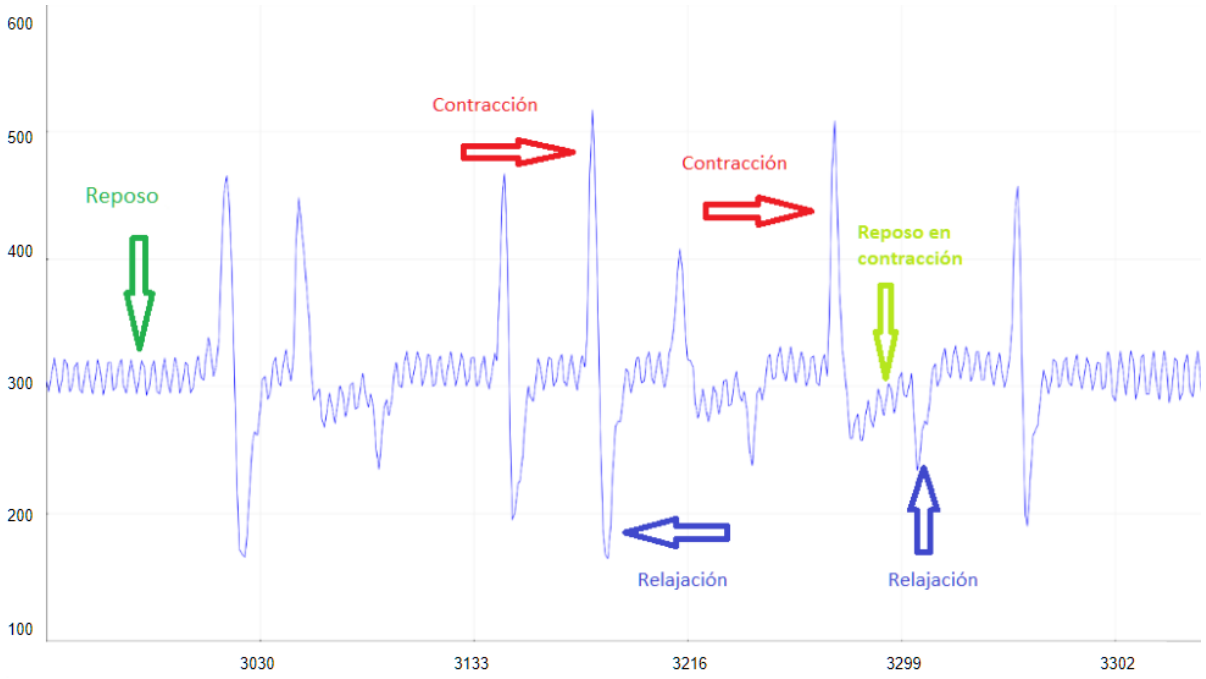


Figura.2.10 *Señal EMG posición pulgar flexión consecutivas pulgar*

Durante su estudio, se ha observado que la amplitud experimenta un repentino aumento (hasta alrededor de 1V) en el momento de la contracción proporcional a la fuerza del movimiento. Del mismo modo, en el momento de la relajación muscular, se observa un pico negativo y de menor amplitud (hasta los 600mV). Sin embargo, la señal solo experimenta cambios instantáneos que coinciden con la contracción y relajación, ya que esta vuelve al nivel de reposo (media de 1.5V con fluctuaciones de 10mV de amplitud) independientemente de si la contracción se mantiene en el tiempo o no.

Por otro lado, no se ha observado ninguna diferencia significativa de la que se pueda extraer información relacionada con los tiempos de subida y bajada, por lo que podemos tratarla como una señal unidimensional. Por ello, podemos concluir que la información relevante de estas señales se corresponde con el valor máximo y mínimo de cada contracción y relajación.

Como ya hemos comentado, el aislamiento total de los grupos musculares correspondientes con el movimiento de cada dedo no es posible. En el caso mostrado en la Figura.2.11, se han detectado de manera simultánea las señales EMG de las posiciones 1 (ubicación para el pulgar) y 2 (ubicación para el índice), durante el movimiento de contracción del dedo pulgar. La señal de máxima amplitud se corresponde con la posición que concuerda con el movimiento (posición del pulgar y movimiento del pulgar), sin embargo, como podemos observar, existe una señal parásita en la posición del índice (rojo) que corrobora la imposibilidad de aislar las señales de cada grupo muscular.

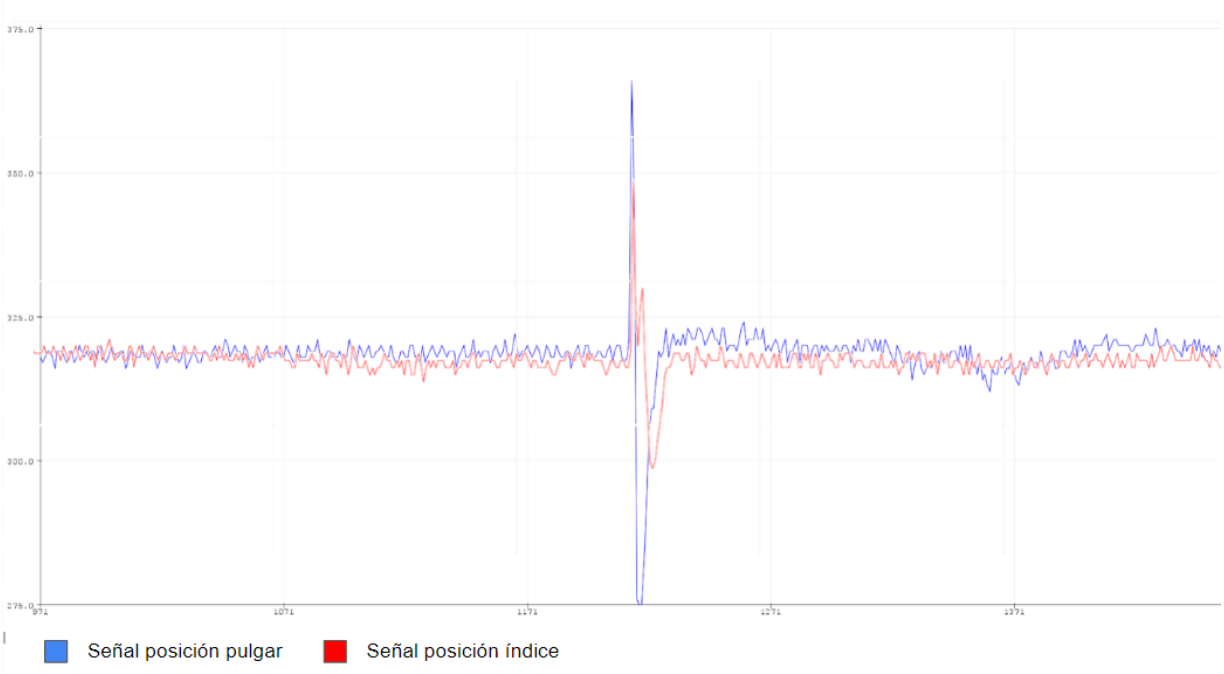


Figura.2.11 *Señales de las posiciones 1 (pulgar) y 2 (índice) capturadas durante la flexión del dedo pulgar*

Por tal motivo, se propone la implementación de una red neuronal que nos permita clasificar el movimiento individual de cada dedo mediante la información capturada por los 5 pares de electrodos ubicados en los distintos grupos musculares. La metodología concreta utilizada para adquisición de la señal de los 5 dedos, así como el desarrollo de la red, serán analizados en detalle en el Capítulo.6

3. Sensorización y diseño de hardware

El objetivo de este capítulo es dotar a la prótesis de mayor capacidad de interacción con el entorno por medio de una amplia sensorización. Del mismo modo, parte de esta sensorización nos da la capacidad de realimentar el control de la prótesis y de esta manera aumentar la calidad de este.

Se ha analizado y comparado la respuesta de diferentes soluciones para varios cometidos como la parametrización de distancia, fuerza, posición y temperatura. Para ello, se han diseñado y fabricado diferentes circuitos impresos que nos permiten realizar un estudio más cómodo y fiable. Del mismo modo, se han diseñado y fabricado varios prototipos de dedos protésicos para probar el funcionamiento de diferentes sensores y accionamientos motrices. Por último, se ha rediseñado, fabricado e implementado una nueva prótesis completa en la cual se han implementado diferentes mejoras mecánicas y los sensores escogidos tras el estudio de la sensorización.

3.1. Sensores analizados

A continuación, se describen los diferentes sensores probados:

- **Sensor Flex:**

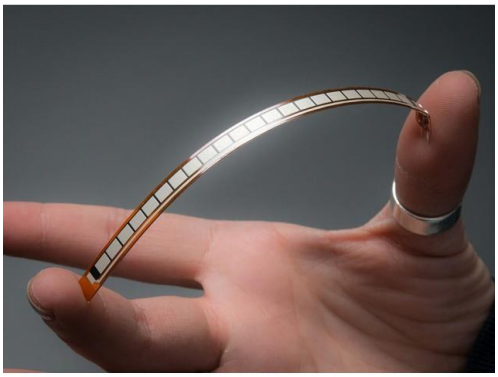


Figura.3.1 *Sensor Flex*

El sensor Flex produce una variación en su resistencia interna dependiendo del grado en el que esté doblada. Son resistencias analógicas que trabajan como divisores de tensión analógica variable. Están compuestos de por elementos resistivos de carbono en el interior de un sustrato altamente flexible y delgado. Estos elementos de carbono varían su resistividad dependiendo de la densidad de carbono que tengan (a más carbono menor impedancia). [9]

Se propone implementar este sensor ubicándolo a lo largo del dedo para, midiendo la variación de la resistividad de este, poder definir la posición del dedo. De esta manera podríamos cerrar el lazo abierto y dotar de realimentación a nuestro control.

- **Sensor fuerza**

Con intención de parametrizar la fuerza del agarre se han analizado dos soluciones diferentes:

- La primera solución planteada consiste en la medición de la corriente que demanda el servomotor. El servomotor tiene un consumo de corriente uniforme mientras realiza un movimiento, sin embargo, si en ese proceso se produce algún tipo de resistencia ajena, como puede ser el agarre de un objeto, la demanda de corriente aumentará para intentar superar el obstáculo. Si medimos esta corriente, podemos discernir si se está produciendo un agarre y con qué fuerza.

- El **sensor FSR** se compone por dos membranas separadas por un fino espacio de aire. El espacio de aire se mantiene gracias a un espaciador alrededor de los bordes y a la rigidez de las dos membranas. Una de las membranas tiene dos conjuntos de pistas intercaladas aisladas eléctricamente y cada conjunto se conecta a un terminal de salida. La otra membrana está revestida con tinta FSR. Cuando se presiona, la tinta FSR hace un corto circuito entre los dos trazos y la resistencia depende de la fuerza aplicada. [10]

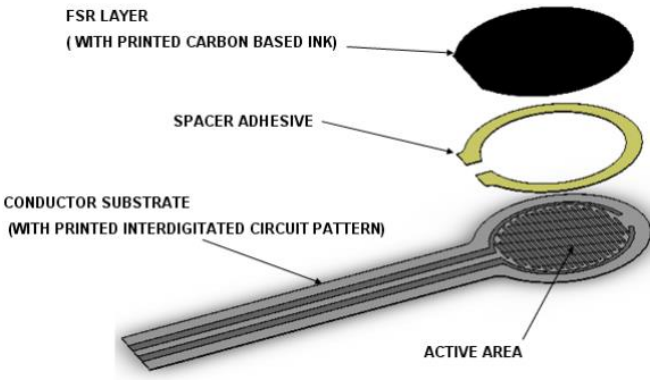


Figura.3.2 *Estructura básica FSR*

Se propone implementar este sensor ubicándolo en diferentes posiciones que serán analizadas posteriormente para, midiendo el nivel de impedancia, determinar si está teniendo lugar el agarre y la cantidad de fuerza utilizada. De esta manera podemos modelar diferentes controles en los que se apliquen distintas fuerzas dependiendo de la tarea a realizar.

- **Sensor de proximidad**

En este caso, tratamos de implementar un sensor de proximidad que nos permita saber la distancia entre la mano y el objetivo. Para ello se han analizado dos sensores de proximidad de distintas aptitudes. VL6180X-SATEL

- El sensor ***SHARP GP2Y0A21YK0F*** es un sensor óptico capaz de medir la distancia entre él y un objeto, para esto el sensor con la ayuda de un emisor infrarrojo y un receptor miden la distancia usando triangulación. El método de triangulación consiste en medir uno de los ángulos que forma el triángulo emisor-objeto-receptor, el Receptor es un PSD (Position Sensitive Detector) que detecta el punto de incidencia el cual depende del ángulo y a su vez de la distancia del objeto. Su rango de trabajo varía desde los 10 cm a los 80 cm. [11]

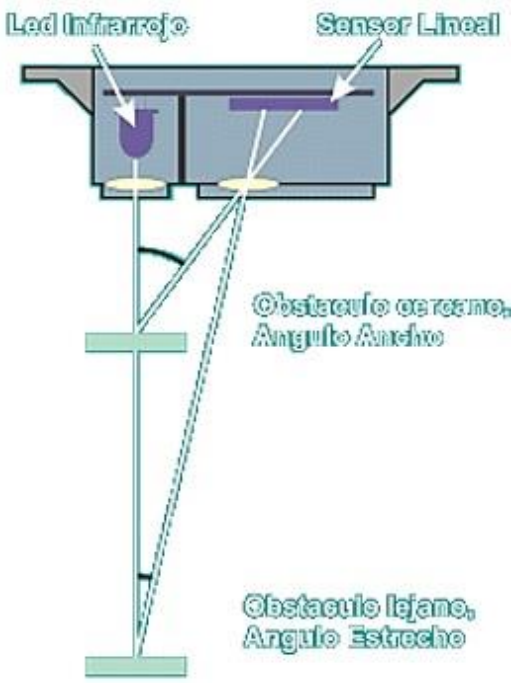


Figura.3.3 *Método triangulación sensor SHARP*

Figura.3.4 *Sensor SHARP*

- El sensor ***VL6180X-SATEL***, en lugar de estimar la distancia midiendo la cantidad de luz reflejada por el objeto, mide el tiempo que la luz tarda en viajar hasta el objeto más cercano y reflejarse en el sensor (tiempo de vuelo). Esta tecnología permite medir la distancia absoluta independientemente de la reflectancia del objetivo. Su rango de trabajo "garantizado" varía de 1 cm a 15 cm. [12]

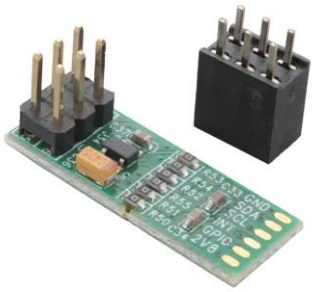


Figura.3.5 *Sensor VL6180X-SATEL*

- **Sensor de temperatura**

Se propone aplicar un sensor de temperatura en la prótesis que nos permita medir la temperatura del objetivo de la prótesis. De esta manera podríamos aplicar un control que evitara el agarre de objetos demasiado calientes o simplemente obtener información sobre la temperatura del objeto en cuestión.

Se ha abordado este objetivo mediante el análisis de dos sensores de diferentes prestaciones:

- El ***LMT85*** es un sensor de temperatura CMOS de precisión con una precisión típica de ±0,4°C (±2,7°C máxima). El sensor devuelve una tensión analógica de salida inversamente proporcional a la temperatura. [13]

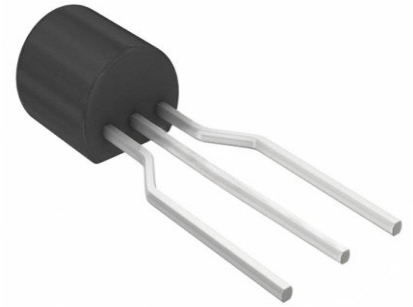


Figura.3.6 *Sensor LMT85LPG*

- La cámara térmica basada en el sensor de Panasonic ***AMG883*** de Adafruit, es un array de sensores térmicos infrarrojos IR de 8x8. Permite medir temperaturas entre 0ºC y 80ºC y mostrar la huella térmica captada mediante interpolación.
  Puede detectar a una persona a una distancia de hasta 7 metros y tiene una tasa de refresco de 10Hz.

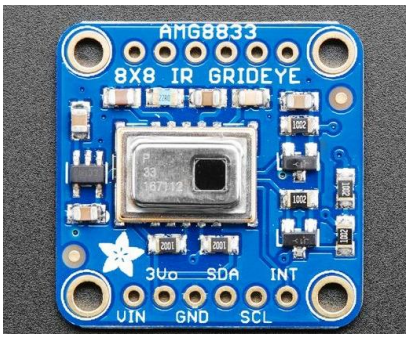


Figura.3.7 *Cámara térmica AMG883*

- **Sensor táctil**

Por último, se busca implementar un sensor táctil que detecte el momento de contacto de la yema de los dedos de la prótesis con el objetivo determinado. Este cometido se llevaba a cabo en la prótesis anterior por medio de unos pulsadores integrados en la yema (Figura.3.8). Sin embargo, se necesita una dirección concreta de contacto y una gran fuerza para conseguir cambiarlo de estado, por lo que se ha descartado.

Figura.3.8 *Pulsador implementado prótesis anterior*

En este sentido, hemos tenido en cuenta dos sensores táctiles diferentes:

- El ***Touch TTP223*** es un sensor capacitivo que hace las veces de un pulsador. La detección capacitiva es una tecnología basada en el acoplamiento capacitivo que puede detectar cualquier cosa que sea conductiva o que tenga una diferencia dieléctrica con el aire. De esta manera, cuando un objeto entra contacto con el sensor cambia la capacitancia y da un valor en alto equivalente a la alimentación usada.

Figura.3.9 *Sensor TTP223*

- El ***AT42QT1070*** es un controlador de sensor capacitivo digital de modo de ráfaga. El dispositivo puede detectar de una a siete teclas, dependiendo del modo seleccionado. Utiliza un método de adquisición de doble pulso. Esto proporciona una mayor inmunidad al ruido y elimina la necesidad de condensadores de muestreo externos, lo que permite la detección táctil utilizando un solo pin. [14]

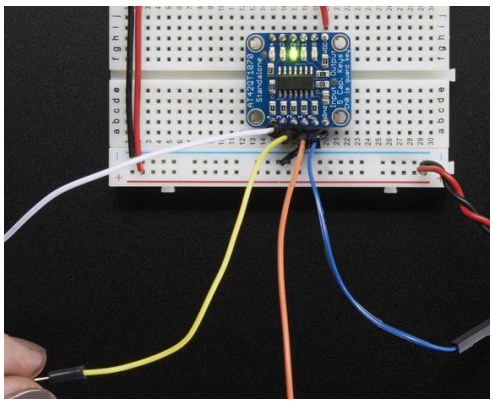


Figura.3.10 *Sensor AT42QT1070*

3.2. Prototipos

Al inicio de este proyecto se contaba con el trabajo anterior realizado por Andrea Pascual Acón y Jesús Sergio Artal-Sevil en el cual se realizó la impresión de una prótesis 3D para su control mediante la plataforma Arduino. En tal proyecto, se realizó una comparación de varios modelos 3D OpenSource de manos biónicas, con el fin de encontrar el modelo que mejor maximizara el compromiso entre simplicidad de diseño y efectividad (Tabla.3.1). Tras el estudio mencionado, se escogió el modelo InMoov, un ambicioso proyecto francés basado en el desarrollo de un humanoide impreso en tecnología 3D [15].

					
Nombre	Brunel	Ada hand v1.2	Dextrus	Dextra	Inmoov
Grados de libertad	9-DOF	5-DOF	6-DOF	6-DOF	6-DOF
Tipo de proyecto	Open source	Open source	Open Source	Open Source Myoelectric	Open Source
Compatibilidad	Arduino IDE	Arduino IDE	Arduino IDE	Arduino IDE	Arduino IDE
Programación	Programable por USB	Programable por USB	Programable por USB y SD	PC Interface	Programable por USB
Peso	371 g	380 g	393 g	405 g	475 g
Dimensiones	198 x 127 x 66 mm	215 x 118 x 58 mm	210 x 159 x 59 mm	196 x 125 x 45 mm	
Materiales	PLA plastic, TPU	Impresión 3D	ABS	Impresión 3D, PLA o ABS	Impresión 3D ABS
Fuente de alimentación	+6/12V	+12V	+9V	+9V	+6V
Actuadores	IMU integrado de 9 ejes	5 actuadores lineales PQ12-30-12-P	5 motores Adafruit Shield	5 pololu Micro Metal Gearmotor 1000:1	5 servos HK1598B y 1 servo MG996r
Microcontrolador	SAMD21G18	ATMEGA2560	ATMEGA2560	Teensy 3.1	
PCB	Chestnut v1.0	Almond v1.2	Chestnut v1.2	Simple PCB	Nervo Board InMoov

Tabla.3.1 *Comparación de diferentes proyectos OpenSource correspondiente a manos Biónicas (Diseño de un Brazo Robótico de bajo coste controlado por sensores EMG superficiales, J.S Artal-Sevil 2018) [24]*

El mecanismo para ejercer el movimiento de los diferentes dedos está basado en cables y muelles colocados a modo de tendones. Estos cables son activados a su vez por una polea incorporada sobre un servo Futaba S3003. El desplazamiento angular de los mismos es controlado por una señal PWM. Como se ha mencionado en el apartado anterior, en esta prótesis, ha sido incorporado un pequeño pulsador en el extremo de cada dedo para obtener una señal de feedback en el agarre [Figura.3.8].

Los dedos son accionados mediante poleas de modo que ceden o recogen cable en función del ángulo de giro recorrido por los servos. A su vez, existe un sexto servomotor alojado en el interior de la muñeca de la mano biónica que proporciona un giro máximo de 180º, al igual que en el brazo humano. En la imagen se observa la prótesis en cuestión y un detalle de los cables y poleas que accionan su movimiento.

Figura.3.11 *Prótesis anterior*

Dado que esta prótesis ya era un proyecto completo en el que no se había contemplado su sensorización, se han diseñado y fabricado los prototipos de dos dedos protésicos diferentes con el objetivo de analizar los sensores escogidos, así como estudiar diferentes tipos de accionamiento mecánico. A su vez, se han desarrollado distintos circuitos impresos que permiten llevar a cabo los estudios de una manera más sencilla y fiable.

3.2.1 Dedo accionado por servomotor

A continuación, se detallarán las características del prototipo y el circuito impreso fabricado para el estudio.

Figura.3.12 PCB *y prototipo "single finger 1" accionado por servomotor*

- **Circuito impreso:**

Se ha diseñado una PCB mediante el software EAGLE para la adquisición de señales y datos. La plataforma elegida para su control ha sido Arduino Uno debido a la sencillez que aporta para la adquisición de datos. Dado que se ha contemplado la posibilidad de ser el módulo de control de una prótesis completa, se han implementado 5 sensores Touch TTP223, un módulo de 5 salidas AT42QT1070, 5 salidas para servos, un módulo para la lectura de corriente INA 219, un módulo de conversión A-D de 12 bit ADS1015, un Joystick para el control del movimiento del dedo y se han habilitado diferentes salidas y entradas analógicas capturar la salida de sensores de fuerza. La información técnica sobre componentes, programas implementados en Arduino y los detalles del proceso de diseño y fabricación de la placa impresa se encuentran en el ANEXO XX.

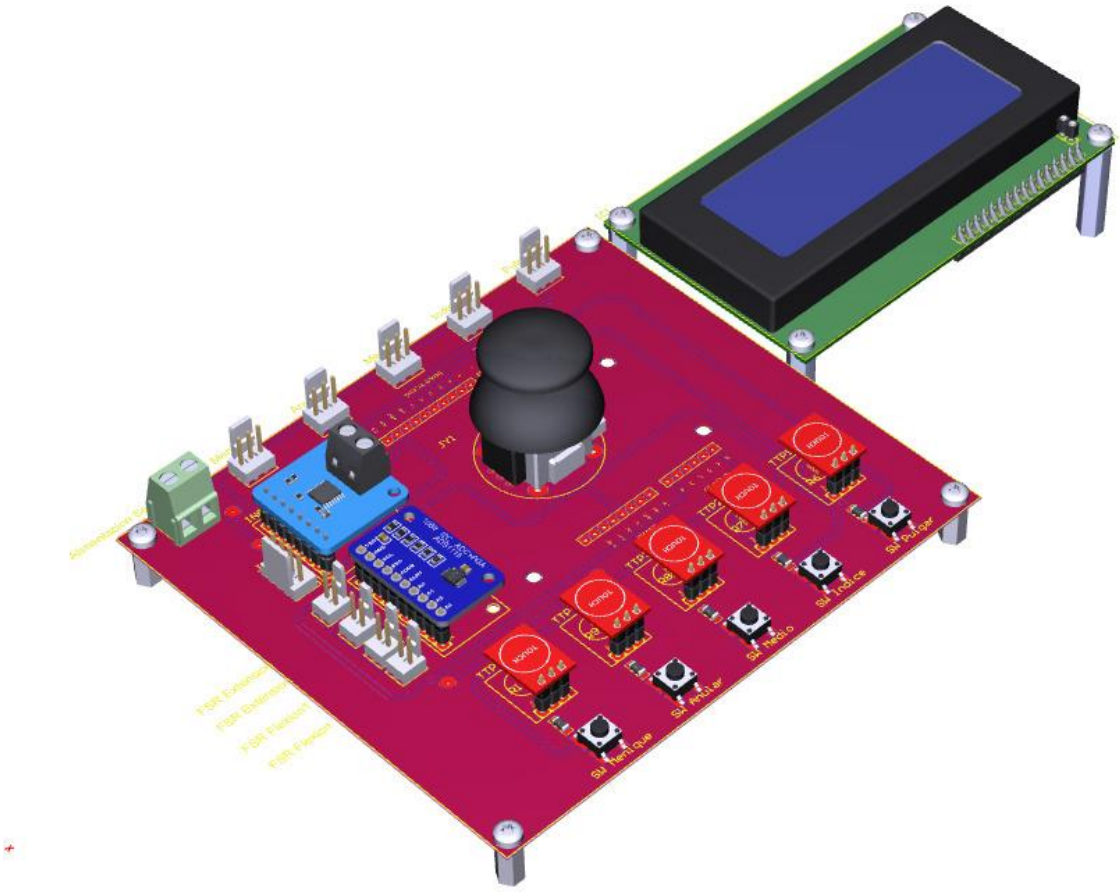


Figura.3.13 Modelo *3D PCB prototipo "single finger 1"*

- **Accionamiento mecánico:**

Este primer prototipo, al igual que la primera prótesis, tiene como mecanismo de accionamiento un servomotor. Sin embargo, esta vez, el servomotor no actúa como una polea que cede y recoge cable en ambas direcciones. En este caso, el servo es el encargado de realizar el movimiento de contracción mientras que el movimiento de relajación lo produce un muelle instalado a lo largo de la parte externa del dedo. A medida que el servomotor recoge el dedo, el muelle acumula energía potencial elástica que más tarde será utilizada para realizar el movimiento de relajación. De esta manera el movimiento se asemeja más al del dedo humano.

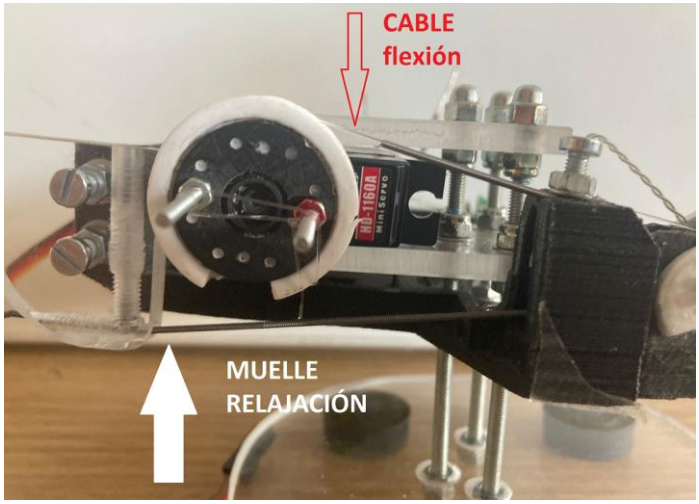


Figura.3.14 Detalle *accionamiento mecánico*

- **<u>Control movimiento mediante Joystick:</u>**

Para el control del movimiento del dedo se ha incorporado un Joystick y se ha realizado un programa en Arduino mediante el cual se puede controlar tanto la dirección como la velocidad del movimiento.

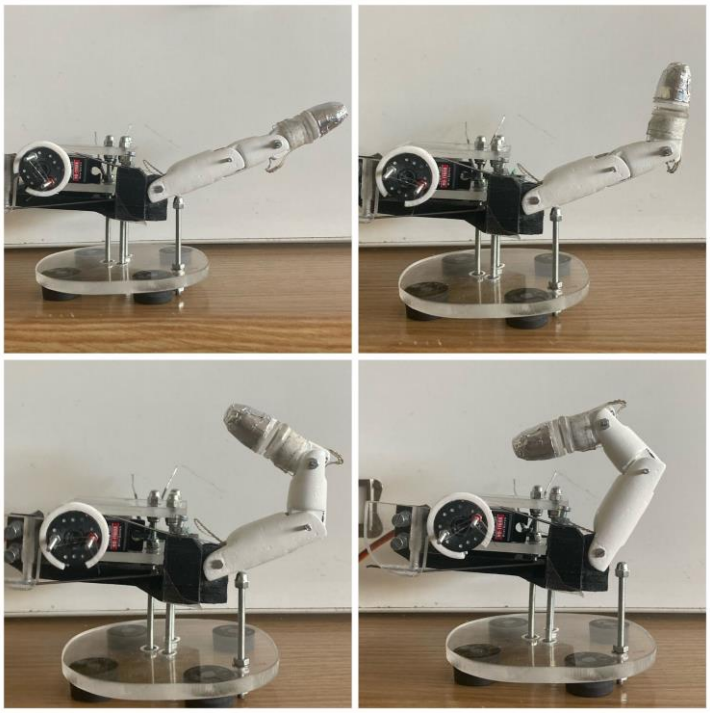


Figura.3.15 Relación *posición angular servo / posición falanges*

Dado que el dedo está compuesto de tres falanges consecutivas de diferente tamaño, si se desea un movimiento de velocidad constante se necesitan 3 velocidades angulares diferentes. En esta línea, tras el estudio de la relación entre la posición angular del servo y la posición de cada falange, se ha realizado un programa que realiza una corrección de la velocidad del angular del servo en el momento en el que el movimiento cambia de falange. De esta manera conseguimos un movimiento constante del dedo completo.

Del mismo modo, a medida que el usuario desplaza el Joystick hacia una de las posiciones Norte o Sur, el dedo incrementa su velocidad de flexión o relajación de una manera gradual. El código de esta aplicación se muestra en el ANEXO A.1.

- **Detección contacto:**

Para la detección del contacto de la yema con el objetivo se han analizado el sensor Touch TTP223 y el sensor AT42QT1070. Se ha programado un pequeño código que detiene el movimiento del dedo en el momento que se detecta un contacto. En el caso del sensor TTP223 se ha simulado el contacto de la yema con el objetivo tocándolo en la PCB durante el movimiento del dedo ya que no ha sido integrado en el prototipo.

Por otro lado, el AT42QT1070 se ha implementado directamente en el prototipo. Para ello, se ha recubierto la yema del dedo de papel de aluminio. La salida del módulo ha sido directamente conectada al muelle y el fin de este con el papel de la yema. De esta manera, dado que el sensor cambia la señal de salida en el momento en que la capacitancia del pin varía, se ha aprovechado la conductividad del muelle y del papel de aluminio. En la Figura.3.16 se puede apreciar como en el momento del contacto con la yema del dedo se enciende el led del pin correspondiente.

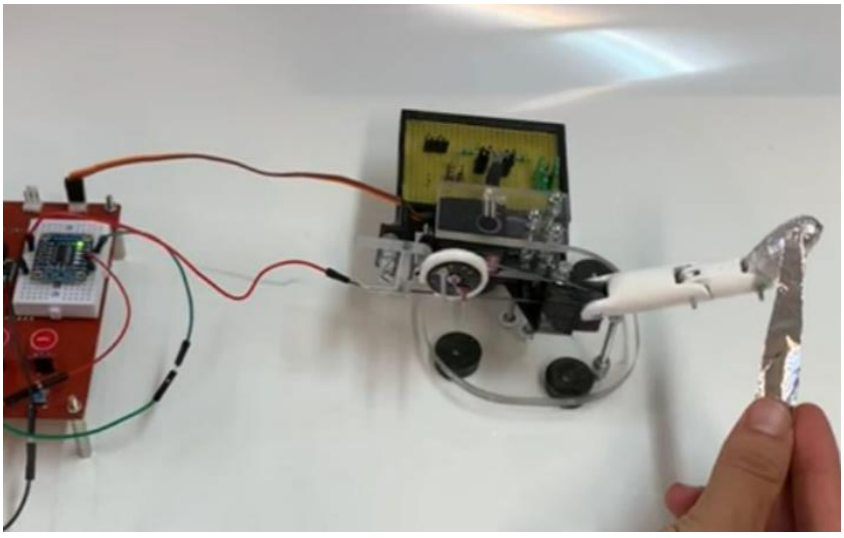


Figura.3.16 Detalle *conexionado AT42QT1070 y momento de contacto*

- **Parametrización fuerza:**

Como hemos mencionado en el apartado anterior, para este objetivo, se plantea la medición de la corriente demandada por el servomotor y la medición de la impedancia del sensor FSR.

- Para la medición de la corriente que consume el servomotor se ha colocado en la entrada PWM del servomotor un módulo **INA219** [16] que por medio de una resistencia de SHUNT nos permite realizar una medición de corriente de precisión. Esta medición se envía vía I2C al Arduino y nos da una resolución de 10 bit.

Según los datos recogidos, mientras el servomotor está realizando un giro sin obstáculos la corriente varía desde unos 8.5 mA a los 14 mA [Figura.3.15]. Posteriormente se han simulado agarres y bloqueos del dedo para determinar la variación en la corriente demandada por el servo. En este caso la demanda de corriente aumenta drásticamente hasta un máximo de unos 43 mA en el caso de bloqueo total. Se ha definido un valor Threshold de 25 mA para el cual, si la corriente consumida supera dicho valor, podemos determinar que se está realizando un agarre o el servomotor está bloqueado por otro motivo [Figura.3.16].

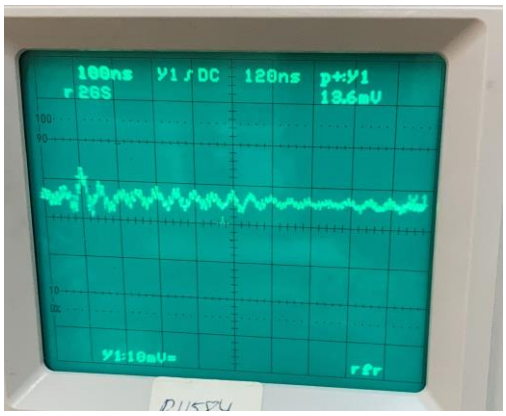


Figura.3.17
Visualización corriente en osciloscopio servo libre

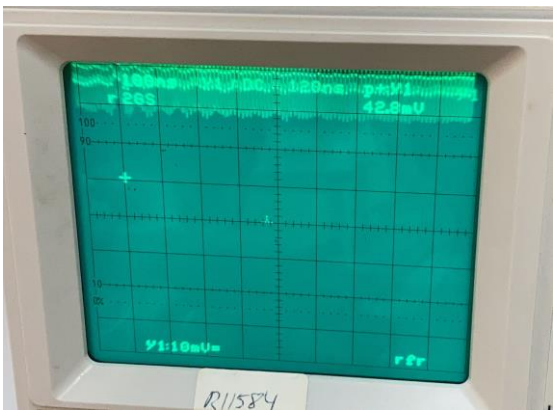
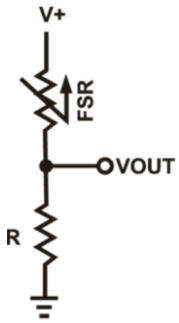


Figura.3.18
Visualización corriente en osciloscopio con obstáculo

- Por otro lado, dado que en el caso del sensor FSR la variación derivada de la fuerza aplicada se traduce variación de impedancia, se ha colocado una resistencia de 10K en serie para hacer un divisor de tensión. De esta manera la tensión Vout [Figura.3.19], tiene una dependencia lineal con la fuerza aplicada en el sensor y, por tanto, midiéndose podemos conseguir la información de la fuerza aplicada.



$$Vout = \frac{R}{R + FSR} \cdot Vin$$

Figura.3.19 Circuito *medición FSR*

Del mismo modo, se han probado dos ubicaciones diferentes para la colocación del sensor FSR.

En primer lugar, se ha colocado una pareja de sensores en las caras dorsales del servo. Como se puede apreciar en la imagen, las FSR están en contacto con dos placas de metacrilato y a su vez, el servo se sujeta únicamente mediante un tornillo que le permite pivotar una pequeña holgura. De esta manera, cuando el dedo se encuentra con un obstáculo, el servo presiona uno de los FSR dependiendo de si está realizando el movimiento de flexión o relajación (superior → relajación, inferior → flexión).

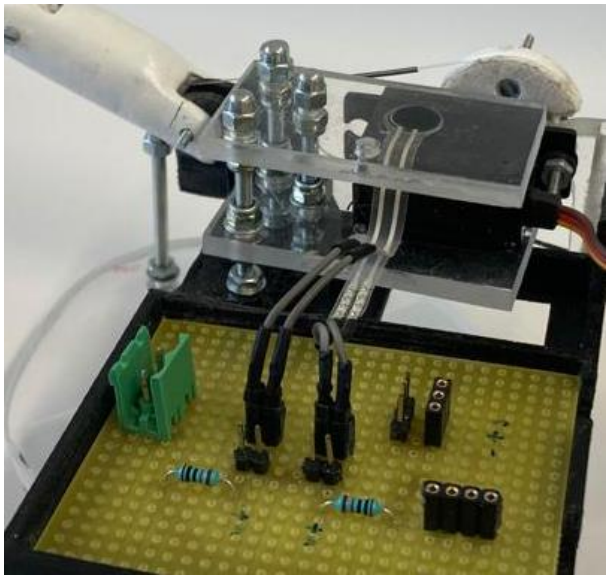


Figura.3.20 Detalle *configuración FSR 1*

En segundo lugar, se ha colocado una FSR en el interior de la yema con un sistema de bisagra y un disco de silicona que presiona el FSR en el momento en el que la yema entra en contacto con un obstáculo en su trayectoria u objetivo (Figura.3.20). Esta configuración, no responde si el obstáculo se contacta con el dedo en otro lugar diferente a la yema, sin embargo, nos devuelve una información más precisa sobre la fuerza del agarre cuando este se da correctamente.

Figura.3.21 Detalle *configuración FSR 2*

Como podemos observar en la Figura.3.22, los valores de salida de la configuración realizada para el FSR están comprendidos aproximadamente entre 1020 y 300 (4.8V y 1.45V) dependiendo de la fuerza aplicada. El nivel de tensión de salida disminuye proporcionalmente a la fuerza realizada siendo el valor mínimo correspondiente a una fuerza aproximada de 2kp.
Se han aplicado diferentes fuerzas al sensor en la yema y se han determinado los siguientes intervalos utilizados en el Capítulo 5 para el control de la prótesis. Agarre de precisión [1023,610], agarre normal de oposición (610,315] y agarre fuerte de oposición (315,0).

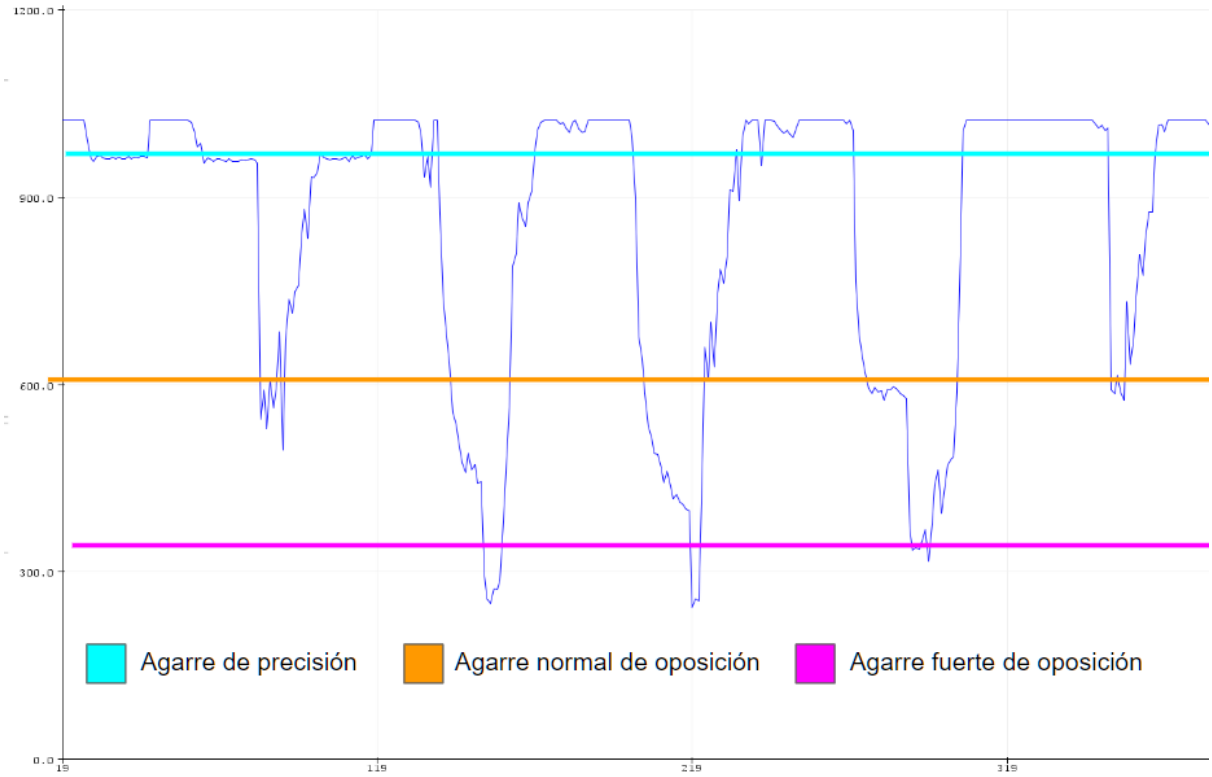


Figura.3.22 Gráfica *Vout sensor FSR ubicación yema*

3.2.1 Dedo accionado por motor lineal

A continuación, se detallan las características del prototipo y el circuito impreso fabricado para el estudio.

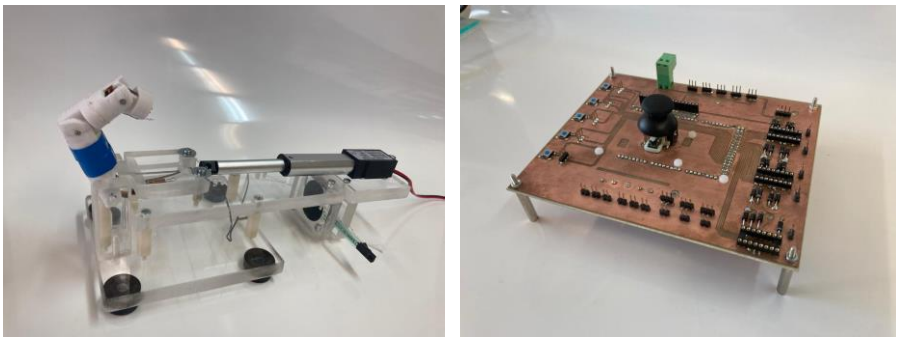


Figura.3.23 Prototipo *"single finger" accionado por motor lineal y PCB*

- **Circuito impreso:**

Al igual que en caso anterior, se ha diseñado una PCB mediante el software EAGLE para la adquisición de señales y datos. La plataforma elegida para su control ha sido Arduino Due a la necesidad de mayor capacidad de computación y número de pines. Dado que se ha contemplado la posibilidad de ser el módulo de control de una prótesis completa, se han implementado 5 sensores Touch TTP223, un módulo de 5 salidas AT42QT1070, 5 salidas para 5 motores lineales con sus respectivos puentes en H, 2 módulos para la lectura de corriente INA 219, un Joystick para el control del movimiento del dedo y se han habilitado diferentes salidas y entradas analógicas capturar la salida de sensores de fuerza y FLEX. La información técnica sobre componentes, programas implementados en Arduino y los detalles del proceso de diseño y fabricación de la placa impresa se encuentran en el ANEXO XX.

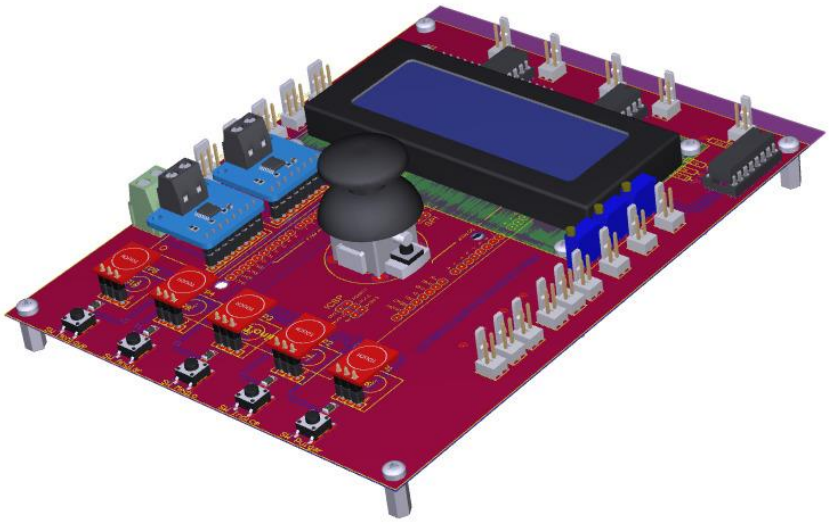


Figura.3.24 Modelo *3D PCB prototipo "single finger 2" motor lineal*

- **Accionamiento mecánico:**

Este segundo prototipo utiliza como accionamiento mecánico un motor lineal de corriente continua en vez de un servomotor. El objetivo es analizar las posibles mejoras en el movimiento mecánico que puede aportar que el movimiento del accionador tenga la misma dirección que el dedo. Es decir, sin necesidad de transformar el movimiento angular del servo en lineal. Por otro lado, con intención de estudiar el mejor posicionamiento del muelle encargado de realizar el movimiento opuesto al realizado por el motor, esta vez se ha colocado en la cara interior del dedo. De esta manera, cuando el vástago se recoge, el dedo realiza el movimiento de relajación (cuando el motor tira del cable) y por contra, el dedo realiza el movimiento de flexión cuando el vástago sale, gracias al potencial elástico del muelle. A su vez, el sensor FLEX instalado en la parte exterior del dedo, dada la flexibilidad que le proporciona el sustrato que recubre las galgas de su interior, elimina las holguras que pueda tener el movimiento de relajación.

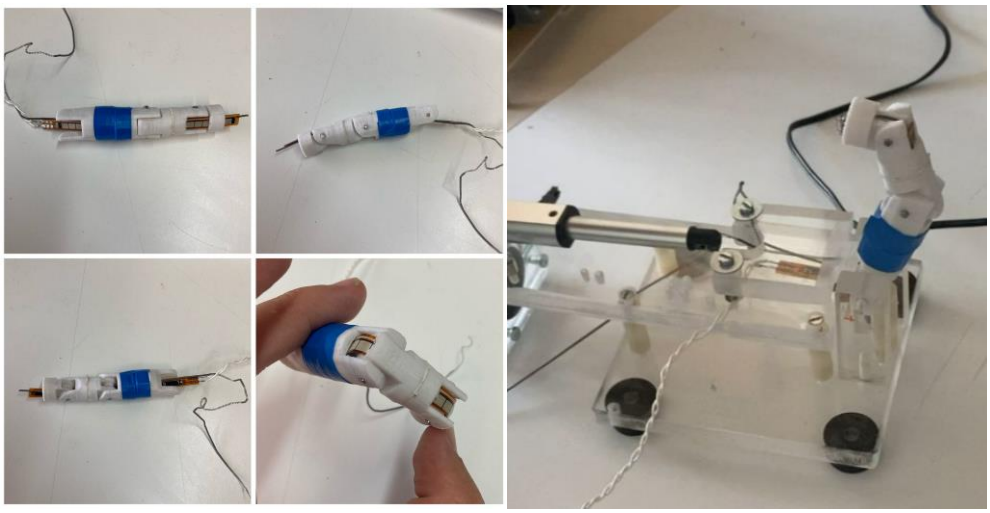


Figura.3.25 Detalle *accionamiento mecánico*

- **Control posicionamiento:**

Para este cometido se han analizado 2 soluciones diferentes.

- En primer lugar, se ha instalado un sensor **FLEX** a lo largo de la cara externa del dedo. Dado que el diseño original carecía de las aperturas necesarias para instalarlo en el interior del dedo, se han realizado unas modificaciones en el diseño 3D por medio del software FUSION 360. Tras ello, se ha imprimido el modelo con las nuevas cavidades y se ha colocado como muestra la Figura.3.25. Los detalles del rediseño 3D y los archivos 3D se encuentran en el Anexo.B.1.

  Del mismo modo que en el caso de las FSR, se ha realizado un divisor de tensión [Figura.3.19] pero con una resistencia de 47 KΩ, para poder medir directamente un voltaje proporcional a los grados de flexión del dedo.

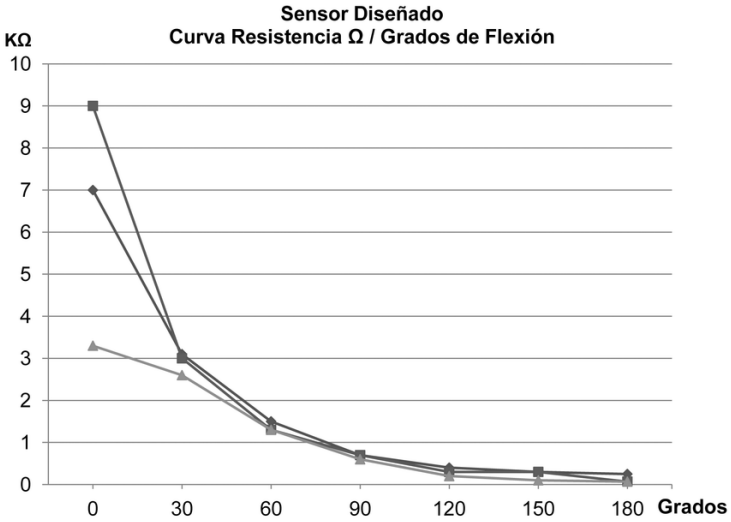


Figura.3.26 Curva *Resistencia/Grados flexión [17]*

Como muestra la gráfica de la Figura.3.25, si el pliegue se realiza en torno a un punto, el comportamiento de la impedancia tiene un carácter que se acerca a la linealidad.

Dado que el movimiento del dedo tiene varios ejes de giro, no se obtiene una curva tan lineal pero podemos observar una gran repetibilidad que nos permite realizar la parametrización de la posición. Si nos fijamos en la Figura.3.27 podemos observar que la salida Vout varía entre 200 para la flexión completa y 400 para la posición de relajación (1V a 2V).

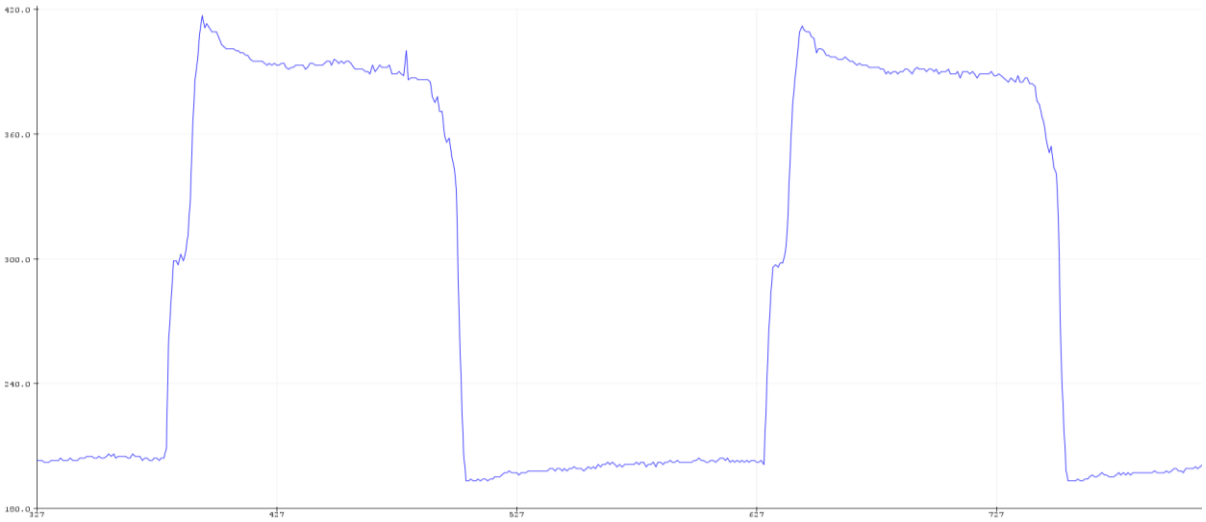


Figura.3.27 Curva *Vout/Grados flexión cada falange*

- En segundo lugar, se ha instalado un potenciómetro lineal en paralelo con el motor lineal. La pestaña del potenciómetro se ha fijado al vástago del motor y de esta manera realiza el movimiento de este de una manera solidaria.

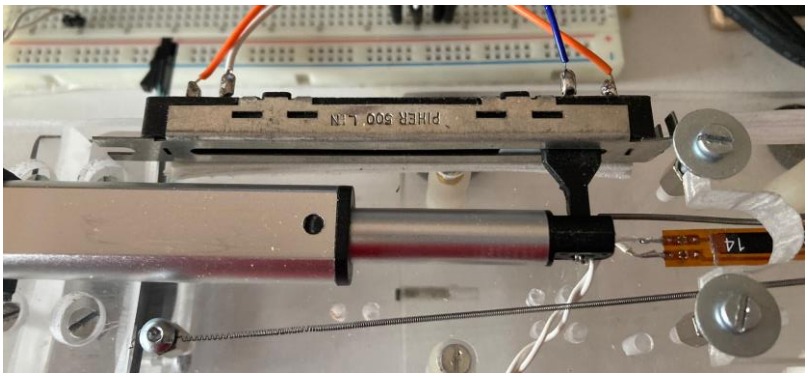


Figura.3.28 Potenciómetro *lineal solidario al vástago*

Se ha seguido el mismo procedimiento que en el caso del sensor flex con un divisor de tensión. La respuesta del potenciómetro va de 292K en estado de flexión total, hasta los 150K en estado de relajación total de una manera lineal. De esta manera, la salida Vout está comprendida entre los 250 para la flexión completa y 625 para la relajación completa (1.22V a 3.71).
Atendiendo a la Figura 3.29 podemos observar que la linealidad en la respuesta de este método es total. Dada la configuración mecánica del prototipo, el vástago sigue avanzando, y por tanto el potenciómetro, más allá de la posición de relajación total. Por ello, se ha marcado en verde el valor de la salida a la que el dedo está totalmente relajado (625).

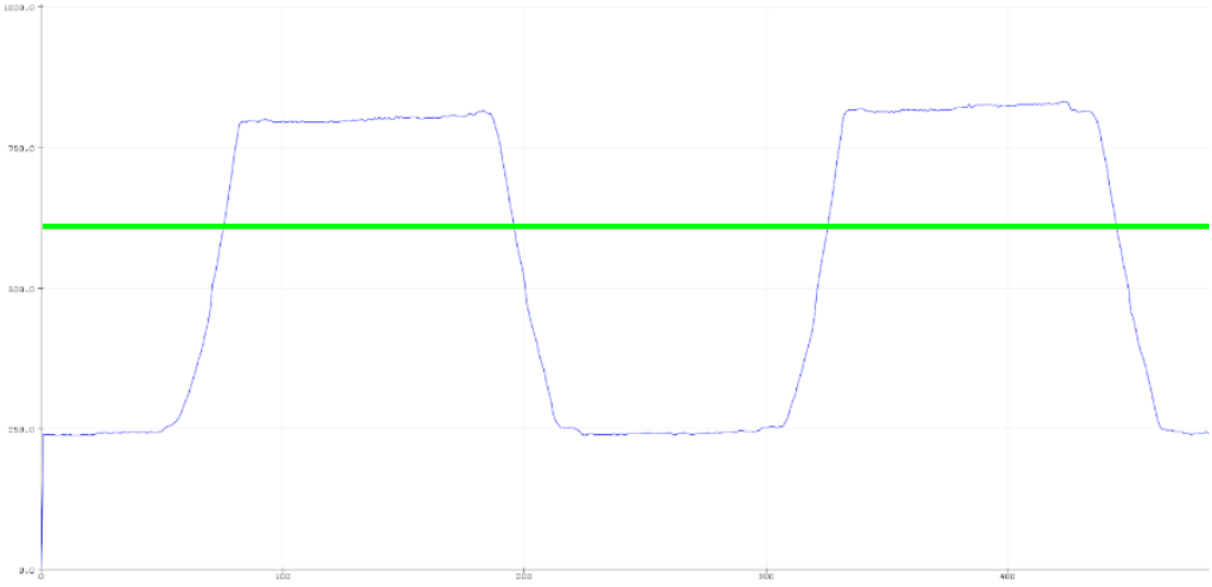


Figura.3.29 Potenciómetro *lineal solidario al vástago*

- **Parametrización fuerza:**

En el caso de esta prótesis se ha probado un FSR de mayor tamaño, en una ubicación diferente a la del primer prototipo.

Dado que el accionamiento mecánico esta vez viene dado por un motor lineal, se han colocado dos placas de metacrilato paralelas entre sí y perpendiculares al movimiento del vástago. El sensor FSR está ubicado entre ellas con un disco de silicona a cada lado para conseguir una presión uniforme en él. De esta manera, utilizando la misma configuración electrónica que en los casos anteriores, conseguimos apreciar si el dedo ha encontrado un objeto en su trayectoria o no.

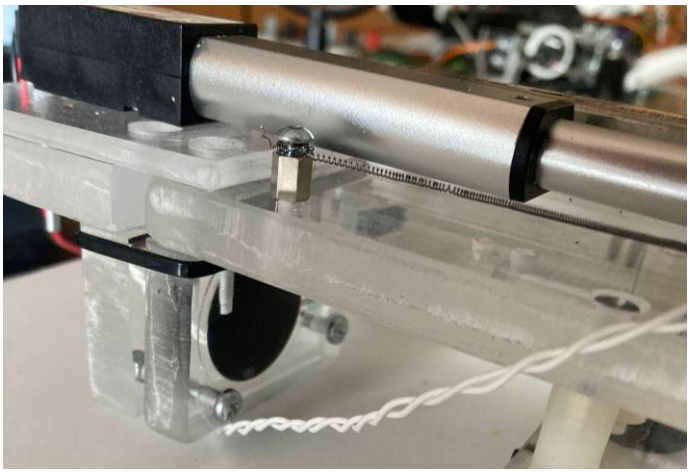


Figura.3.30 Detalle *configuración FSR 3*

3.2.3 Estudio sensores de distancia y temperatura

En este apartado se han analizado los sensores de distancia SHARP y VL6180X, y los sensores de temperatura LMT85 y AMG883 anteriormente descritos. Sin embargo, los primeros de cada categoría han sido descartados rápidamente debido a sus características. Concretamente, por un lado, el sensor SHARP tiene un rango de trabajo de al menos un orden superior al que se busca en esta aplicación y por otro lado, el sensor LMT85 necesita contacto con el objeto, o al menos una distancia muy escasa para medir temperatura, además de tener una lenta respuesta.

Por tanto, para este apartado se ha diseñado un pequeño circuito impreso en el cual acoplar los sensores VL6180X y AMG883 al mismo tiempo que un display LCD para la visualización del registro de la cámara termográfica y un Arduino Due para la toma de datos.

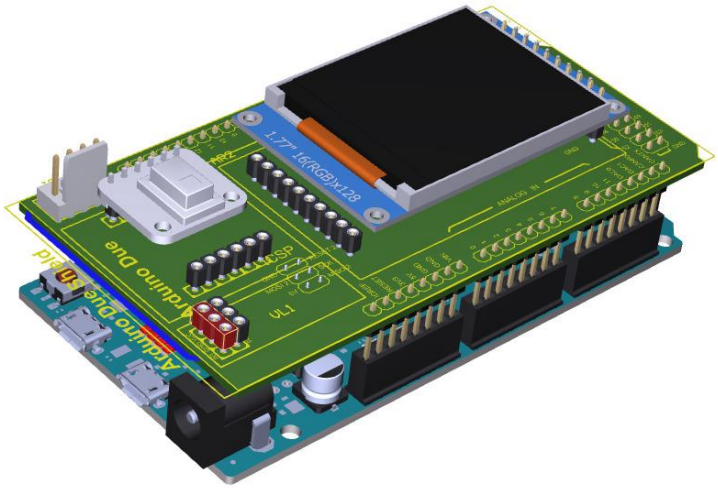


Figura.3.31 Modelo *3D PCB temperatura y distancia*

- **Parametrización temperatura**

Para este objetivo se ha programado un código en Arduino que calcula la media de temperatura captada por la cámara termográfica y muestra en el display la huella térmica a tiempo real. De esta manera, si la media de temperatura calculada supera los 55ºC o si alguno de los 64 (8x8) sensores IR infrarrojos detecta una temperatura mayor a 80 ºC se activa una salida digital que más adelante será utilizada para el control de la prótesis. [Anexo A.2]

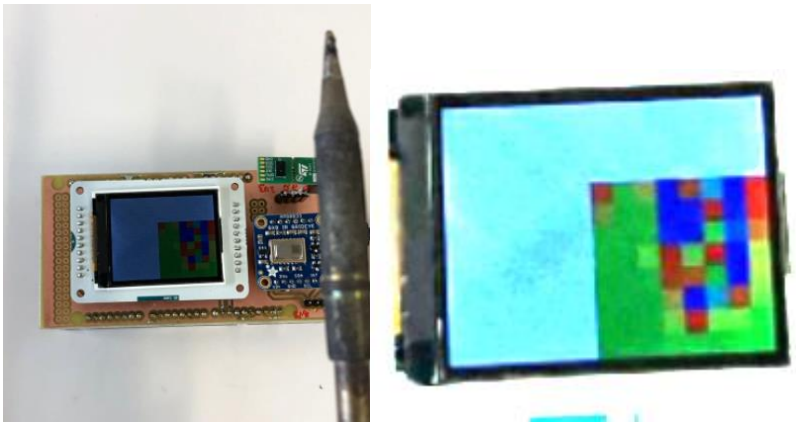


Figura.3.32 Detección *de temperatura sensor AMG883*

3.3. Conclusiones

Tras el estudio y análisis de los diferentes sensores, métodos y mecanismos motrices considerados se han escogido las siguientes opciones:

- **Mecanismo motriz:** Tras el estudio de ambos mecanismos se ha observado que la utilización de un motor lineal no aporta tanta robustez y repetibilidad al movimiento como se había supuesto. Por ello, dado que su instalación y control son más complejos que en el caso de un servomotor, se ha decidido mantener el servo como mecanismo motriz.
- **Posición:** Dado que el potenciómetro lineal es una solución para la parametrización de la posición que se utiliza únicamente en el caso del motor lineal y ha sido descartado, escogemos la solución con sensor FLEX. Asimismo, dado que está ubicado en el propio dedo, la configuración con sensor flex es menos sensible a posibles errores debidos a desajustes mecánicos ocasionados por el uso de la prótesis.
- **Táctil:** En ambos casos el análisis de la respuesta de los sensores ha sido igualmente satisfactorio, sin embargo, el AT42QT1070 conlleva una implementación física mucho más práctica, sencilla y realista.
- **Fuerza:** Tanto la solución de medición de corriente (INA219) y la primera configuración con FSR (en el servo) nos devuelven información independientemente de si el agarre se da correctamente, pero con una menor precisión que la segunda configuración de FSR (en la yema). Por tanto, es interesante combinar ambos métodos para abarcar todas las posibilidades. Dado que la primera configuración de FSR es muy compleja y no aporta más información de relevancia que la solución de medición de corriente, se descarta la configuración con 2 FSR (en el servo), y se escoge una combinación del método de medición de corriente y el FSR en la yema.
- **Proximidad:** Aunque han sido probados, como ya se ha mencionado anteriormente, el sensor de proximidad SHARP ha sido rápidamente descartado debido a que su rango de funcionamiento empieza a los 10 cm y para esta aplicación se necesita al menos un orden menor. Por ello se ha escogido el VL6180 que nos ofrece una respuesta con una precisión de menos de 1cm.
- **Temperatura:** Para esta aplicación buscamos un sensor que nos permita saber la temperatura de un objeto en concreto y no de la temperatura ambiente. El sensor es demasiado lento y para realizar la medición de temperatura de un objeto debería estar en contacto con él por tanto queda descartado.

VARIABLE	OPCIÓN	IMPLEMENTACIÓN
MECANISMO MOTRIZ	SERVOMOTOR	✓
	MOTOR LINEAL	✗
POSICIÓN	FLEX	✓
	POTENCIÓMETRO	✗
TÁCTIL	TTP223	✗
	AT42Q1070	✓
FUERZA	FSR “yema”	✓
	FSR “servo”	✗
	INA219	✓
PROXIMIDAD	SHARP	✗
	VL6180	✓
TEMPERATURA	LMT85	✗
	AMG883	✓

Tabla.3.2 *Elección de componentes a implementar prótesis final*

Por último, se ha decidido implementar 2 led RGB en el interior de las yemas de los dedos índice y pulgar (por ser los más significativos) con el fin de informar al usuario de diferentes situaciones como puede ser la detección de un contacto, la finalización de un agarre o indicar el tipo de agarre seleccionado.

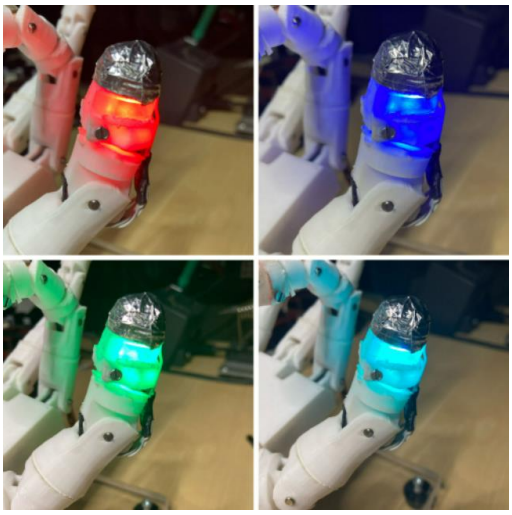


Figura.3.33 Led *RGB instalado en el interior de la yema del dedo pulgar*

3.4. Prótesis final

Tras la elección de los nuevos componentes a implementar se ha llevado a cabo la impresión 3D y el montaje de la prótesis definitiva. Para ello se han realizado diferentes modificaciones en los modelos 3D originales con el fin de habilitar las ubicaciones necesarias para albergar los sensores.

Una vez realizado el rediseño se ha imprimido el modelo completo y se ha montado siguiendo los pasos mostrados en el Anexo B. La configuración de los sensores implementados es la mostrada en la siguiente tabla.

Ubicación	Componente	Número Total
FLEX	Pulgar	X3
	Índice	
	Medio	
Táctil	Pulgar	X5
	Índice	
	Medio	
	Anular	
	Meñique	
FSR	Pulgar	X5
	Índice	
	Medio	
	Anular	
	Meñique	
LED	Pulgar	X2
	Índice	

Proximidad	Palma	X1
Temperatura	Palma	X2

Tabla.3.3 *Componentes implementados y ubicación*

.

Figura.3.34 *Prótesis final*

4. Reproducción del alfabeto de signos español

El objetivo de este apartado consiste en la reproducción del dactilológico de la Lengua de Signos Española por medio de la prótesis robótica, a modo de demostración de las capacidades mecánicas de la misma.

4.1 Antecedentes y contexto

Según los datos recogidos por el Instituto Nacional de Estadística (INE), en España hay un total de 1.064.000 de discapacitados auditivos (lo cual supone un 2,2% de la población total). Como respuesta natural a esta limitación sensorial, a lo largo de la historia las personas sordas han desarrollado diferentes lenguas de signos. [18]

En España, desde 2007, la lengua gestual oficial es la Lengua de Signos española (LSE) y cuenta con más de 100.000 usuarios signantes, para los que alrededor del 25% esta es su segunda lengua. Asimismo, en España existen otras variantes y lenguas gestuales entre las que cabe destacar la Lengua de Signos Catalana (LSC) y la Lengua de signos Valenciana (LSCV). Sin embargo, la inteligibilidad mutua de todas las variantes en la península ibérica es muy alta (incluyendo la Lengua de Señas Portuguesa) debido a que incluso en los casos más peculiares, su léxico no difiere en más de un 30%. [19]

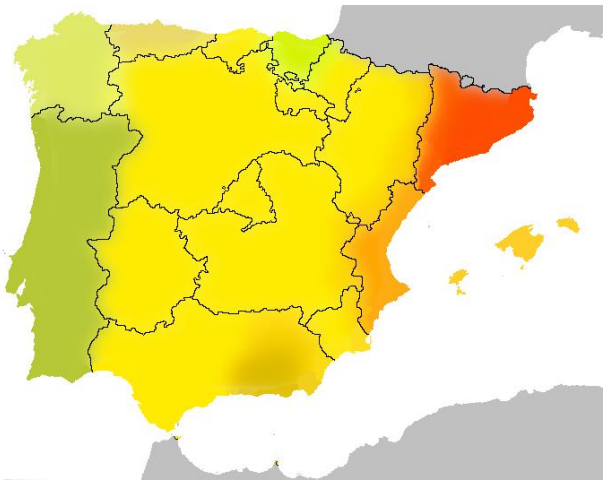


Figura 4.1. Mapa lengua signos Península Ibérica: *En amarillo LSE, rojo LSC. Los colores se alejan del espectro amarillo en función de su grado de diferenciación de la LSE*

El alfabeto manual o dactilológico es un sistema de representación de las letras de los alfabetos orales-escritos por medio de las manos. Existen los bimanuales y los unimanuales, en concreto el español pertenece a este último tipo y data de 1620. [20]

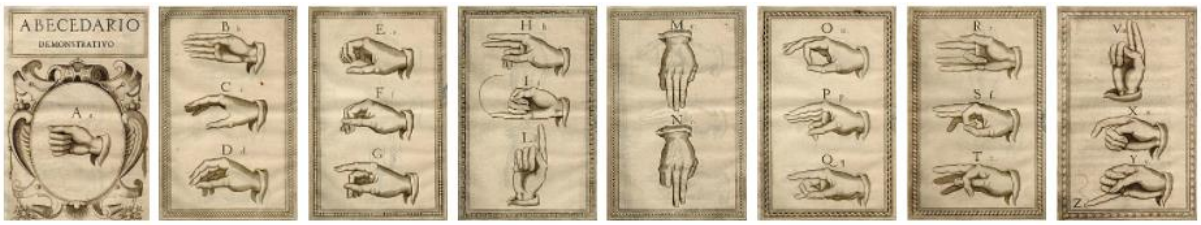


Figura 4.2. *Grabados calcográficos de Diego de Astor en la obra Reducción de las letras y Arte para enseñar á hablar los Mudos.*

4.2 Tecnología y metodología

Para la realización de esta tarea se han utilizado la combinación de la placa *Raspberry Pi Model 3B* con *Matlab/Simulink* y la placa *Arduino Nano* (microcontrolador *ATmega328p*) y el entorno de desarrollo de *Arduino (IDE)*.

- **Matlab/Simulink**: Dentro del paquete MATLAB utilizamos la plataforma de simulación multidominio *Simulink*, que a su vez, integra la herramienta *StateFlow* con la que conseguimos modelar máquinas de estados y diagramas de flujo.

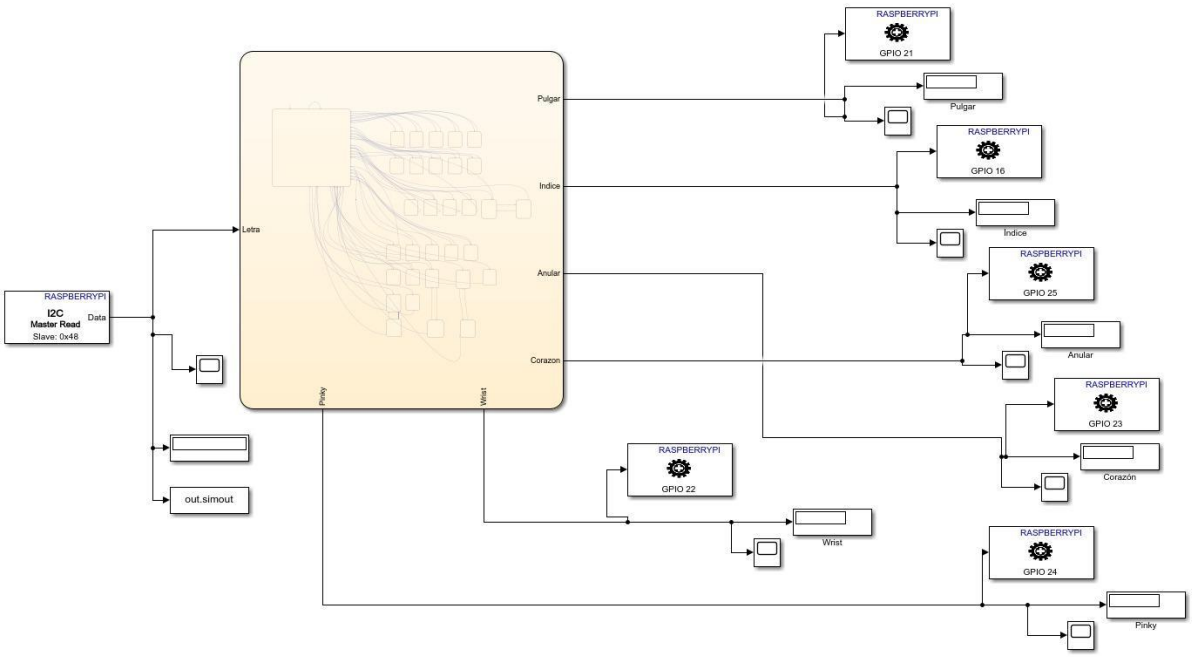


Figura 4.3. *Captura del modelo completo en el que se aprecian bloques Simulink y el bloque "chart" de la máquina de estados de Stateflow.*

El modelo es programado en un equipo externo y volcado en la placa Raspberry Pi como un sistema embebido. De esta manera, como vemos en la Fig.4.3, el modelo tiene como entrada un bloque “I2C Master Read” correspondiente a la dirección 0x48 de la Raspberry, y un total de seis bloques GPIO correspondientes a cada una de las salidas lógicas de la placa. Del mismo modo, existe otro gran bloque “chart” (color anaranjado) que se corresponde con la máquina de estados de *StateFlow*, el cual consta, a su vez, del mismo número de entradas y salidas. Este bloque en cuestión es el encargado de, dependiendo de la letra de entrada, configurar las salidas de manera que la “mano” imite el gesto correspondiente siguiendo el dactilológico español.

El bloque StateFlow está compuesto por un total de 29 bloques que constan de 6 parámetros que se corresponden con las salidas de los servos de cada dedo y la muñeca (Figura.4.4). En esta línea, el diagrama se estructura en torno a un bloque llamado “EstiradoInicial” del cual derivan los 28 restantes. Cuando el sistema entra en el bloque *StateFlow* lo hace por este estado, quedando a la espera de la entrada de las letras a reproducir. De esta manera, si por ejemplo se diera la señal de entrada de la letra “E”, el estado cambiaría al correspondiente y por tanto la configuración de las salidas. El sistema mantendrá estas salidas hasta la entrada de otra letra, por ejemplo, la “X”, en este momento, la máquina de estados cambiará al estado de la letra “X” pasando siempre antes por el estado de reposo “Estirado Inicial” (Figura.4.5).

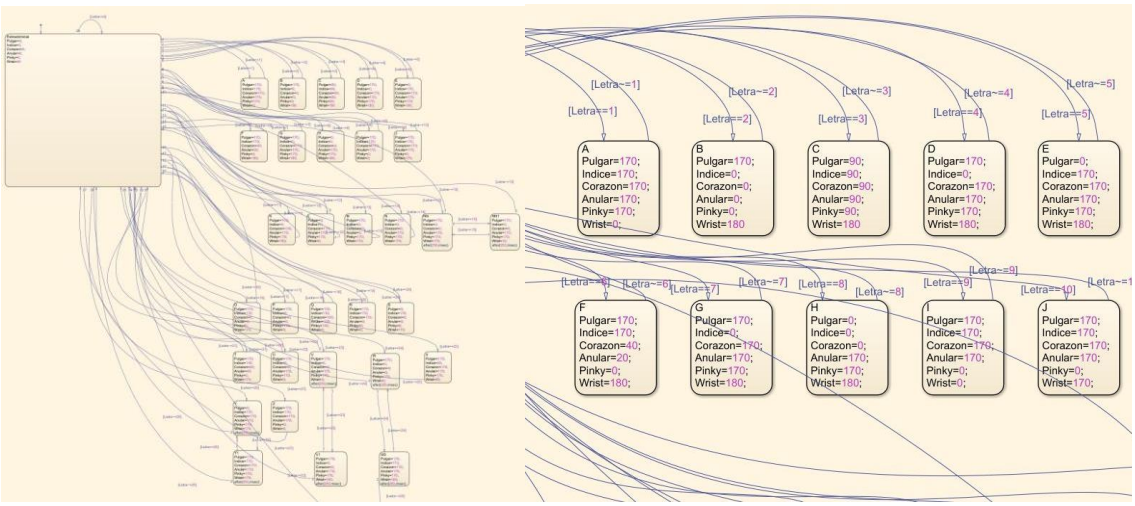


Figura 4.4. *Detalle del bloque “chart” de la máquina de estados de Stateflow.*

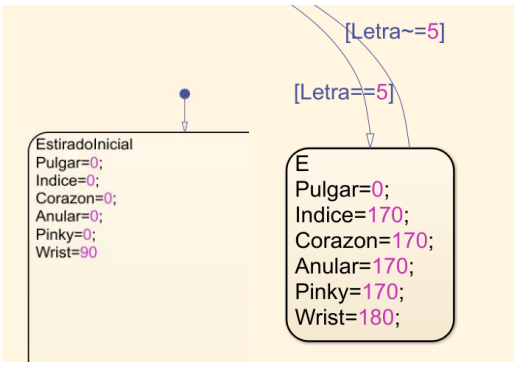


Figura 4.5. *Detalle bloques de estados de StateFlow*

- **Arduino Nano:** Dado que el modelo está embebido en la Raspberry, no podemos acceder a él para modificar la letra de entrada en tiempo real. Por ello, se ha decidido realizar un pequeño programa independiente, mediante la plataforma Arduino, que lee las letras indicadas por el usuario desde el teclado y envía el "integer" respectivo para cada caso. De esta manera, nuestro modelo será capaz de interpretar la entrada y actuar en consecuencia. Para consultar el código Anexo.A.3.

  Para la comunicación entre las dos placas se ha elegido el protocolo I2C ya que ambas cuentan con esta tecnología. Para que esta comunicación se dé correctamente, el dispositivo esclavo debe realizar una solicitud al maestro para poder transmitir los datos. En nuestro caso, dado que el Arduino es el dispositivo esclavo, se ha incluido en el programa una función que realiza el "request" necesario.

  Por otro lado, existe una discrepancia en el voltaje utilizado por cada plataforma para este protocolo, siendo que Arduino utiliza 5V mientras que Raspberry 3.3V. Por ello, se ha añadido un convertidor lógico bidireccional I2C entre ellos, que soluciona la disparidad. Tras estos ajustes, la comunicación se da correctamente y el modelo funciona con éxito.

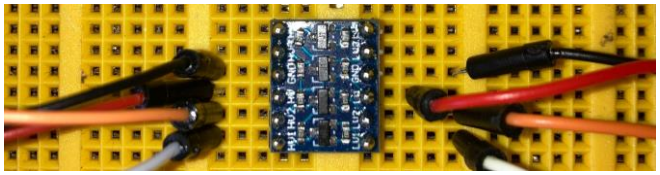


Figura 4.6. *Convertidor lógico bidireccional para comunicación I2C*

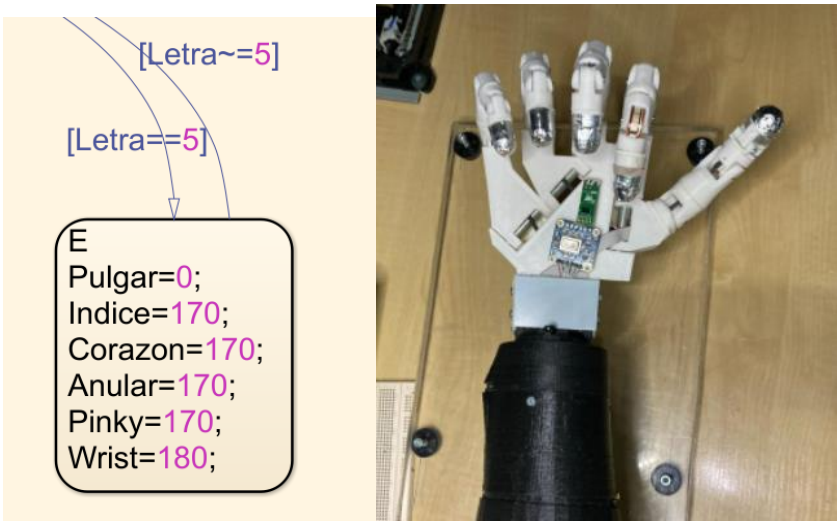


Figura 4.7. *Ejemplo ejecución de la letra E por la prótesis*

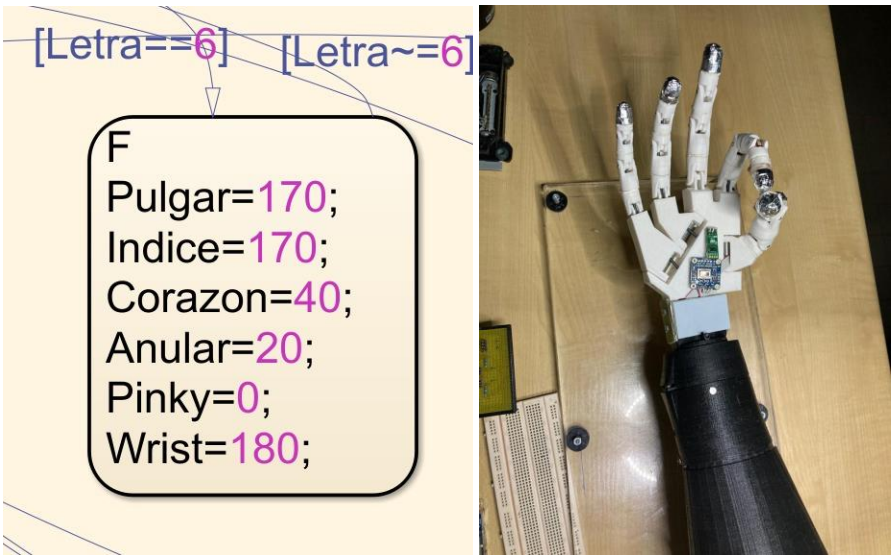


Figura 4.8. *Ejemplo ejecución de la letra F por la prótesis.*

5. Control agarres

El objetivo de este apartado es realizar un control capaz de realizar diferentes tipos de agarre dependiendo de las necesidades del usuario en un momento determinado. Para ello se han utilizado los mecanismos y sensores elegidos tras el estudio llevado a cabo en el Capítulo.3.

5.1. Control agarre general

El primer modelo de control a realizar se trata de un agarre cilíndrico general que permite agarrar de forma intuitiva y precisa una gran variedad de objetos. En líneas generales, el control debe ser capaz de detectar cuando el usuario ordena realizar un agarre, comenzar el movimiento cuando el objetivo se encuentre a la distancia correcta que permita el agarre, detectar el contacto con el objetivo y modular la fuerza con la que ejecutarlo. Asimismo, el agarre deberá ser abortado si el objeto en cuestión supera una temperatura peligrosa.

Para este, y para el resto de los controles se ha utilizado una combinación de Matlab/Simulink y Arduino IDE, así como de la plataforma Raspberry Pi y de los sensores elegidos.

- **Programa Arduino:** Como se viene haciendo en todo el proyecto, la toma de señales

  se lleva a cabo por medio de Arduino. En este caso se ha realizado un programa que capta todas las señales provenientes de los sensores. Dada la escasa capacidad de cómputo del Arduino, esta toma de señales no se hará de manera simultánea, sino que se llevará a cabo de manera individual cuando el modelo de Simulink así lo demande. Del mismo modo, dependiendo de las necesidades del control, cada sensor tiene un nivel predefinido que, al ser superado, el programa de Arduino envía una señal digital a la Raspberry indicando la satisfacción del parámetro en cuestión lo que normalmente desencadenará un cambio de estado en el modelo de control de Simulink (ver Figura 5.1). Para más información sobre el programa realizado visitar el Anexo.A.5

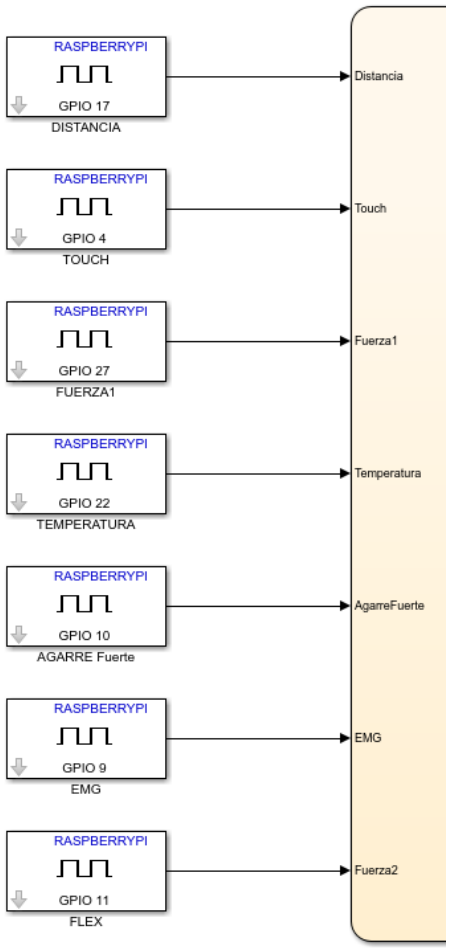


Figura.5.1 *Detalle entradas digitales al modelo Simulink provenientes de Arduino*

- **Señal EMG:** Se ha colocado un tríodo de electrodos en la zona central del antebrazo del usuario para la toma de señales EMG con intención de detectar la contracción correspondiente a un cierre total de la mano. Tras los análisis anteriormente realizados, se ha definido un valor de 400 (2V) como valor Threshold, el cual, si es superado, activa la señal digital "EMG" que se conecta con la Raspberry mediante el GPIO 9 para su uso en el control Simulink.
- **Señal Distancia:** Se ha instalado el sensor de distancia VL6180X en la palma de la mano con intención de medir la distancia entre la zona central de agarre y el objetivo en cuestión. Para ello, se ha definido el valor de salida 200 (1V) que se corresponde con una distancia de 5 cm, que una vez alcanzado activa la señal "DISTANCIA" conectada al GPIO 17 indicando que el objetivo se encuentra a una distancia óptima para su agarre.
- **Señal Touch:** Cada dedo lleva instalado una toma de las 5 que contiene el sensor AT42QT1070. Así pues, cuando cualquiera de los dedos contacte con el objetivo una el programa de Arduino activará la salida digital "Touch" que es conectada con el GPIO 4 de la Raspberry Pi

- **Señal Agarre Fuerte:** Un botón conectado al GPIO 10 nos permite seleccionar entre dos fuerzas de agarre diferente: normal y fuerte. De esta manera, si esta señal está en activo en el momento del inicio del agarre se realizará el agarre fuerte y de lo contrario el agarre normal.
- **Señal Fuerza:** En la yema de cada dedo se encuentra un sensor FSR con el que podemos medir la fuerza de agarre de cada dedo. Dependiendo de la elección tomada por el usuario al inicio del agarre (agarre normal o agarre fuerte), la señal "Fuerza", conectada al GPIO 27, se activará cuando se supere el valor predeterminado. Estos valores son 610 para el agarre normal y 315 para el fuerte (Ver Figura.3.22).
- **Señal Temperatura:** Se ha instalado una cámara termográfica AMG883 con el que podemos medir la temperatura del objeto a agarrar. De esta manera, si la temperatura media de los 64 sensores IR que componen la cámara, es superior a 80ºC se activa la señal digital "Temperatura" que va conectada al GPIO 22.

- **Modelo de control Simulink:** Atendiendo a la Figura 5.2 podemos observar la estructura del control. El sistema recibe la información relevante de los sensores mediante las entradas digitales que podemos encontrar en el margen izquierdo, con esta información realiza la configuración de sus salidas para activar los movimientos de los dedos mediante la máquina de estados que se encuentra en el bloque *StateFlow* y comunicarse de vuelta con el programa Arduino.

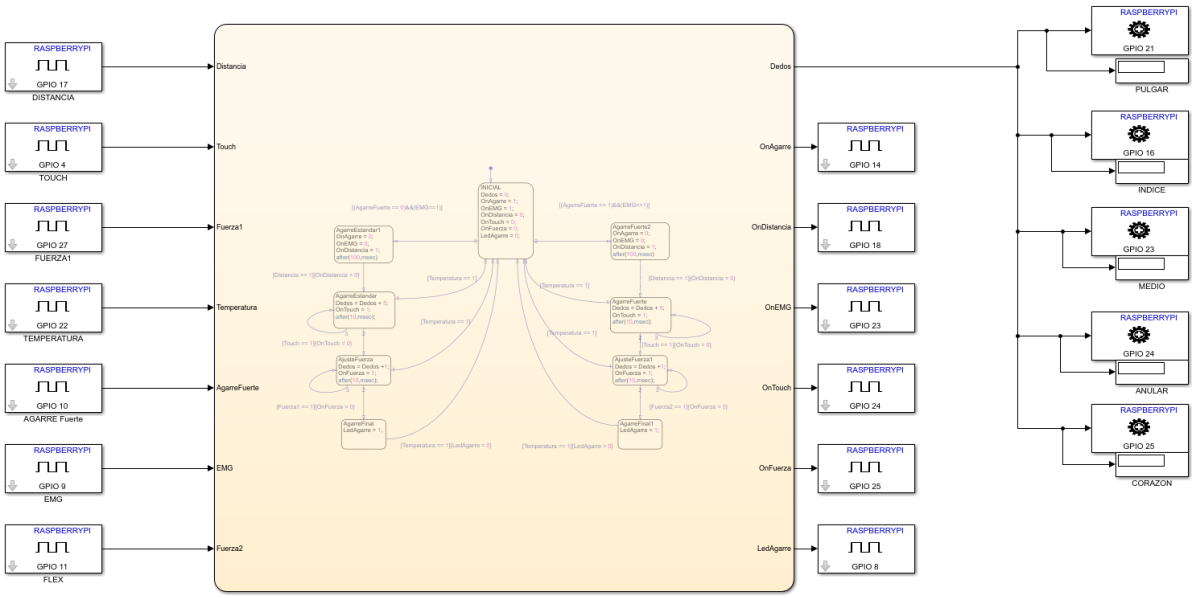


Figura.5.2 *Modelo Simulink para el control del Agarre*

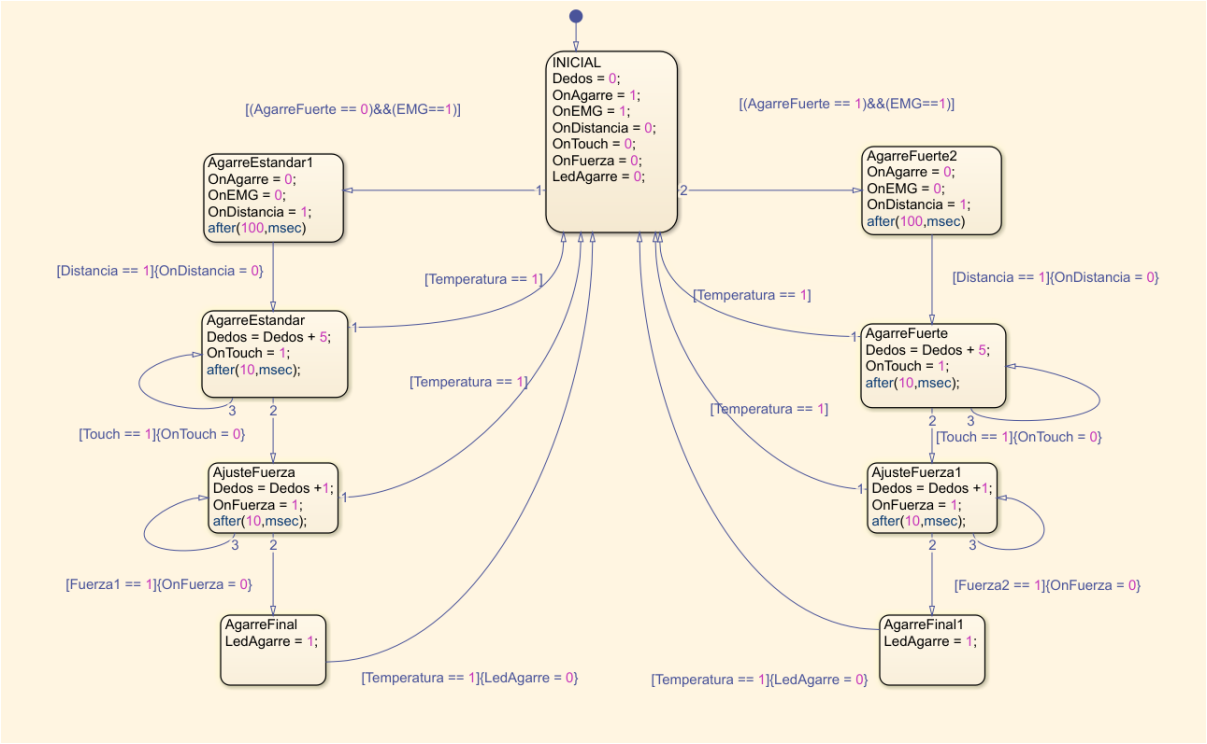


Figura.5.3 *Detalle del desempeño de la máquina de estados.*

- **Máquina de estados:** Atendiendo a la Figura 5.3 podemos observar el funcionamiento de la máquina de estado de *StateFlow.*

  En primer lugar, el estado inicial realiza una configuración de espera en reposo aguardando a la señal de inicio de agarre. Para ello inicializa todas las variables "On" a 0 excepto la señal "EMG" ya que será la que marque el inicio del agarre. Estas variables "On" son las encargadas de demandar al Arduino la toma de datos de cada sensor y se desactivan o activan dependiendo de si son menester o no. Una vez el usuario active la señal "EMG" con una contracción muscular, el modelo comenzará el agarre normal o el agarre fuerte si así lo ha indicado pulsando el botón "AgarreFuerte".

  El siguiente estado (por ejemplo "AgarreEstandar1") activa la toma de datos del sensor de distancia con "OnDistancia = 1" y desactiva la toma de datos para el tipo de Agarre y el inicio del mismo poniendo "OnAgarre" y "OnEMG" a 0.

  Una vez el Arduino detecta que el objetivo se encuentra a una distancia óptima para su agarre se desactiva la toma de datos de distancia y el sistema pasa al siguiente estado "AgarreEstandar". En él se activa la toma de datos
  "Touch" y comienza el movimiento de los servos a una velocidad de 5 grados cada 10 milisegundos hasta el momento que detecte un contacto. En ese momento se deshabilita la toma de datos del sensor Touch y pasamos al siguiente estado donde se activa la toma de datos de "Fuerza" y se reduce la velocidad de los servos un 80%. De esta manera, cuando el nivel de fuerza alcance el nivel marcado por el tipo de agarre, el sistema desactiva la toma de datos de fuerza y enciende un los Led indicando que

el agarre se ha completado. Asimismo, durante todo el proceso, el análisis de temperatura se mantiene activo para abortar el agarre en el caso de que el sensor detecte una temperatura de más de 80ºC.

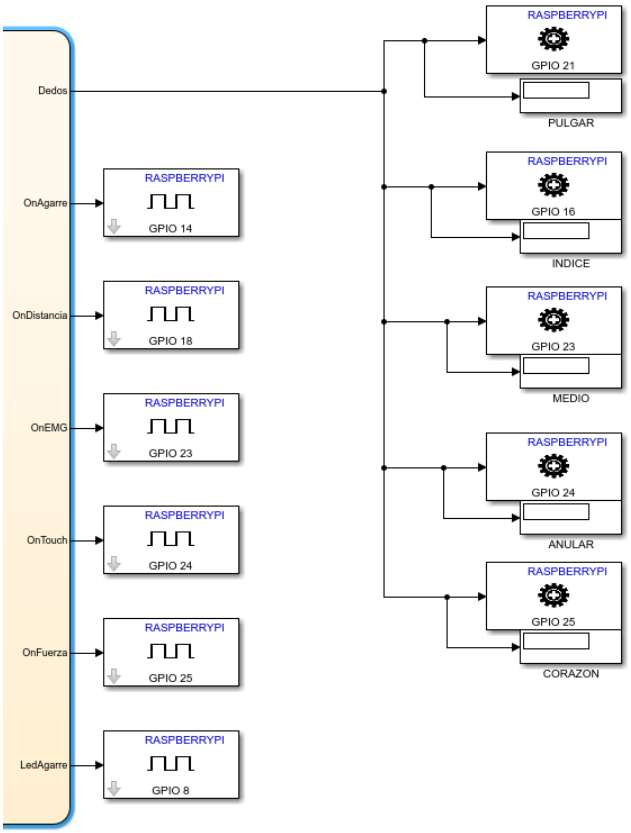


Figura.5.4 *Detalle salidas. En el lado izquierdo podemos observar las salidas digitales tipo "On" encargadas de indicar al Arduino que datos capturar en cada momento. En el lado derecho se encuentran las salidas de los servos que se corresponden con el movimiento de los dedos.*

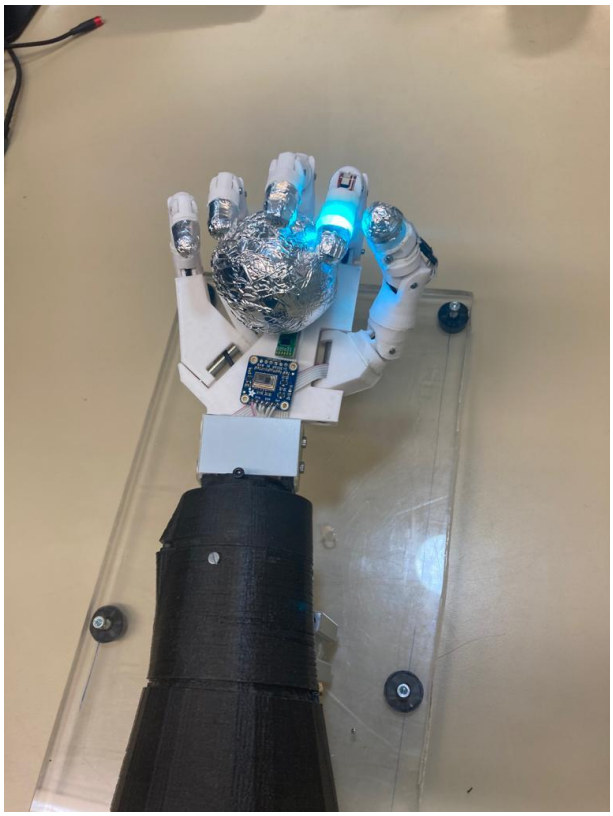


Figura.5.5 *Agarre general en el momento de contacto*

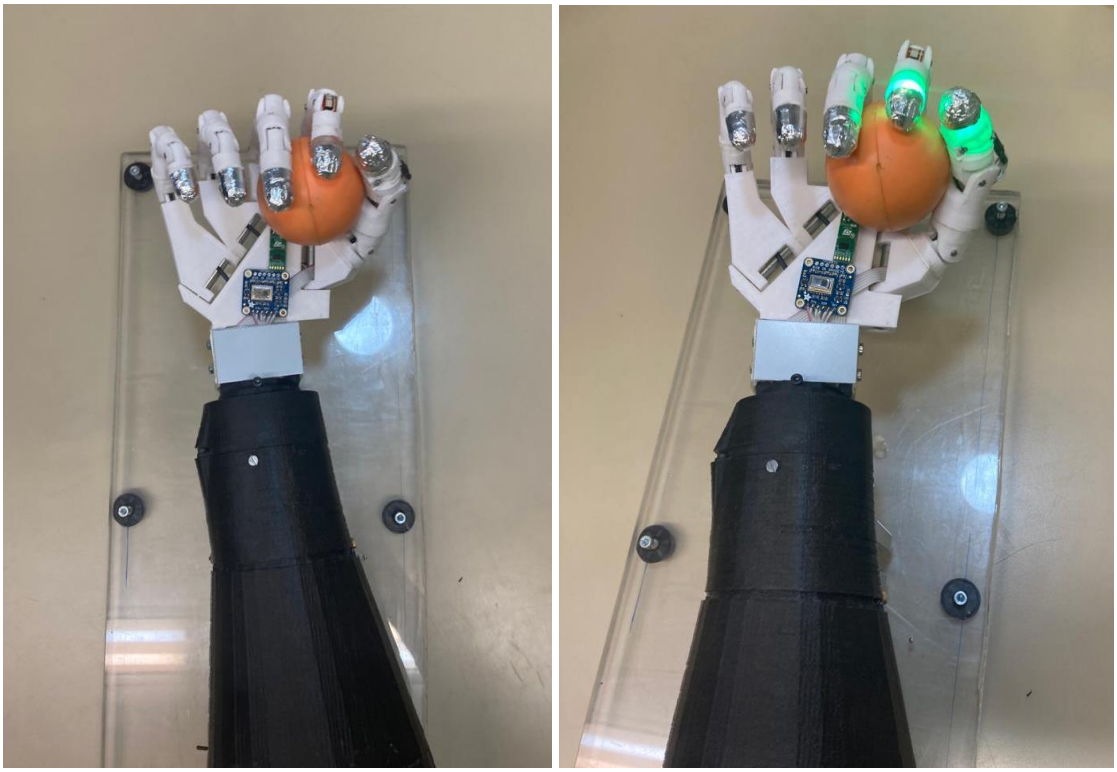


Figura.5.6 *Agarre general fuerte realizado por la prótesis durante y tras el ajuste de fuerza final*

5.1. Control agarre de precisión

En este caso se ha realizado el control para realizar un agarre de precisión que cuenta con dos modalidades diferentes: cerrado de precisión y apertura de precisión. Estos agarres son un método rápido y fiable de coger o manipular objetos pequeños y complejos. El cerrado de precisión presiona el dedo índice contra el pulgar mientras el resto de los dedos se cierran sobre la palma. Por otro lado, la apertura de precisión también presiona el índice contra el pulgar, pero esta vez el resto de los dedos permanecen abiertos.

Figura.5.7 *Ejemplo de cerrado y apertura de precisión de la prótesis "Bebionic"*

- **Máquina de estados:** Dada la similitud al control anterior, con intención de sintetizar

  y evitar redundancias, se muestra únicamente la parte fundamental del control. Atendiendo a la Figura.5.8 podemos apreciar la máquina de estados que regula la demanda de información al programa Arduino explicado en el apartado anterior y detallado en el Anexo.A.4. Como novedad frente al caso anterior, este control regula la posición de los dedos por medio de los sensores Flex.

  En primer lugar, una vez el usuario a demandado el inicio del agarre y su tipo, el sistema cierra o abre los dedos meñique, anular y medio dependiendo del agarre seleccionado. Una vez el sensor Flex alojado en el dedo medio llega a la posición definida (abierto totalmente o cerrado), comienza el movimiento simultáneo del dedo pulgar e índice a modo de pinza. Este movimiento se detiene momento en el que se produce el contacto por oposición entre ellos o con el objeto a agarrar. Tras ello, se activa la captura de datos de los sensores FSR del pulgar e índice y los servos vuelven a avanzar a un 20% de la velocidad anterior hasta llegar al nivel definido para el agarre de precisión (610).

  En ese momento el agarre se da por finalizado y la prótesis devuelve una señal lumínica mediante la activación de los diodos Led con objeto de notificar de ello al usuario. [Figura.5.9]

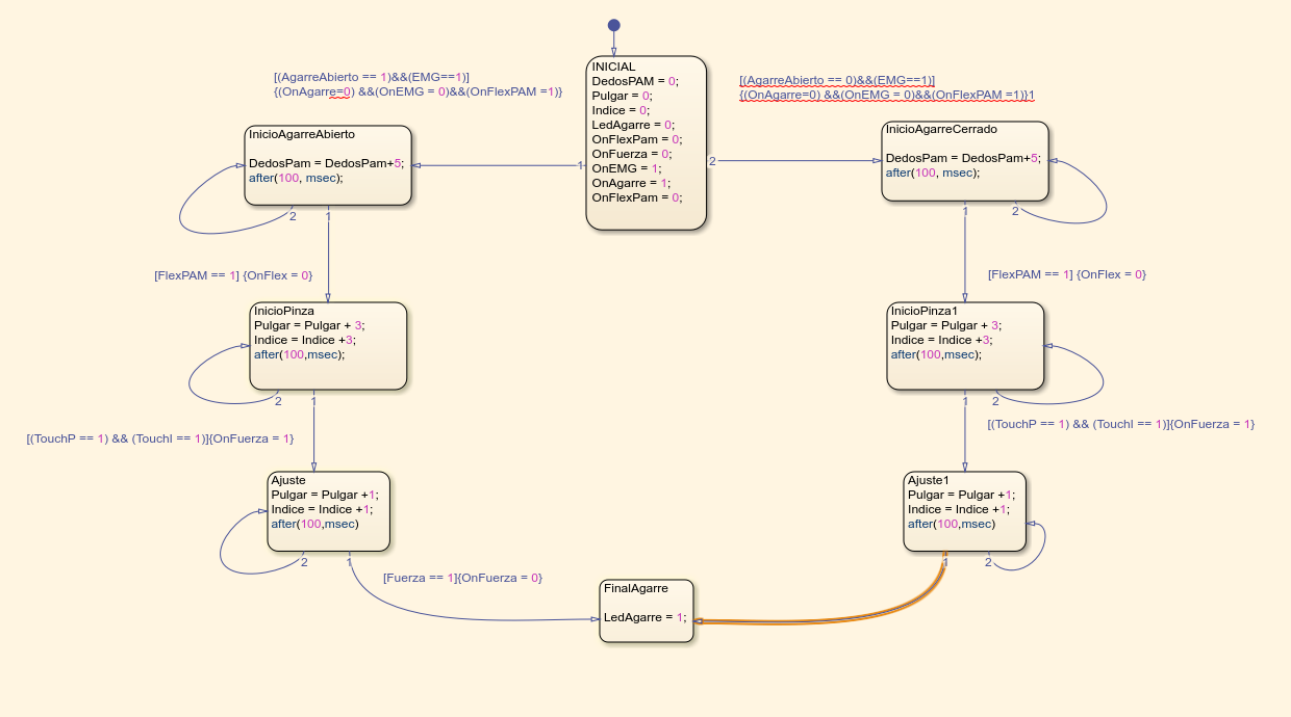


Figura.5.8 *Máquina de estados control agarre preciso*

Figura.5.9 *Momento fin de agarre de precisión abierto*

6. Control Redes Neuronales

El objetivo de este apartado consiste en la clasificación singular del movimiento de cada uno de los dedos a raíz de 5 señales EMG y el consiguiente control de la prótesis.

Como se ha explicado en el Capítulo 2, las señales EMG son muy sensibles a multitud de variables como pueden ser, la ubicación de los electrodos, las características de la piel del usuario o la fisiología y fuerza muscular del mismo. Del mismo modo, aún para un mismo usuario y en condiciones idénticas, la repetibilidad es baja, dificultando en gran medida la clasificación singular de cada dedo por métodos tradicionales. Así pues, se ha optado por una solución basada en aprendizaje automático y más concretamente una red neuronal para abordar el objetivo de identificar y clasificar los movimientos individuales de cada dedo.

6.1. Fundamentos de las redes neuronales

Antes de comenzar con el detalle del procedimiento realizado para este objetivo es preciso exponer un breve resumen sobre la red neuronal y su funcionamiento.

- **Redes neuronales:** Son un tipo de algoritmos característicos del llamado "machine learning" que permiten el aprendizaje jerárquico posibilitando una gran capacidad de abstracción del conocimiento adquirido. Está compuesta por un conjunto de neuronas artificiales que se conectan masivamente entre sí y que trabajan conjuntamente con objeto de resolver problemas del mismo modo que el cerebro humano. Estos sistemas se forman a sí mismos por medio del entrenamiento en lugar de ser programados explícitamente.

  La neurona es la unidad mínima de procesamiento de estas redes y recibe su nombre su analogía a las neuronas biológicas ya que responde a estímulos externos de manera similar. No obstante, a nivel computacional, una neurona es una simple función que recibe una o varias señales de entrada (input), realiza una serie de cálculos internos y generan un valor único de salida (output). [22]

$$y = f\,(x^T w + b) = f\,(\sum w_n x_n + b)$$

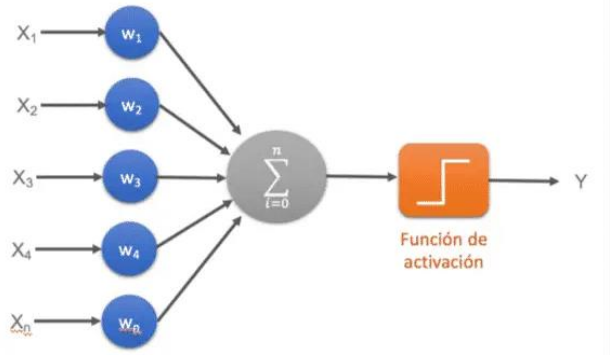
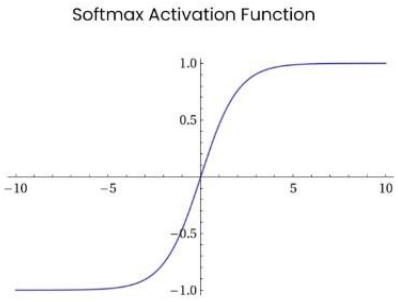


Figura.6.1.1 *Matemática de la neurona*

En referencia a la Figura.6.1.1:

- x_n : Cada una de los inputs x_n
- w_n: peso que multiplica cada input
- $f()$: es la función de activación
- b: sesgo de la función de activación
- y: output de la neurona

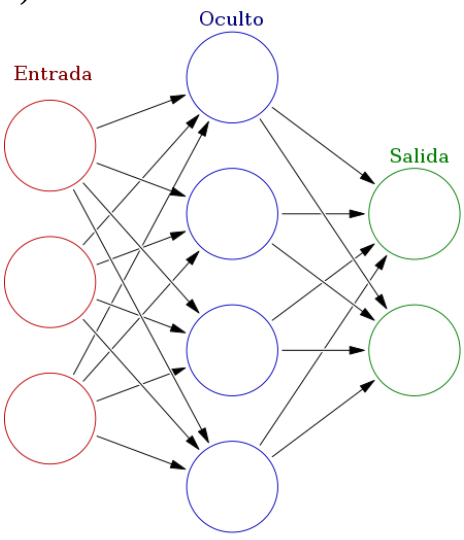
Por tanto, la neurona realiza una suma ponderada de todas las entradas a la misma y, mediante la función de activación elegida, genera una salida. Esta función de activación suele tener una derivada simple para minimizar el coste computacional. En nuestro caso la función de activación utilizada será la función *Softmax (normalized exponencial)* (Figura.6.1.2)



$$f_i(\vec{a}) = \frac{e^{a_i}}{\sum_k e^{a_k}}$$

Figura.6.1.2 *Función de activación Softmax*

Para que una red neuronal funcione correctamente al menos necesita una capa de entrada/s y una de salida/s, aunque normalmente, y donde se encuentra la esencia de esta tecnología, tiene una o varias capas intermedias llamadas capas ocultas (ver Figura.6.12).



-

Figura.6.1.3 *Arquitectura red neuronal simple*

- **Paradigmas del aprendizaje:** Tras exponer los principales elementos que componen la neurona, debemos elegir cómo emplearlos para llevar a cabo el objetivo encomendado. Existen 3 paradigmas del aprendizaje que abarcan las técnicas existentes de machine learning: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje reforzado. En nuestro caso, el paradigma utilizado es el primero.

  La base del **aprendizaje supervisado** consiste en que el programador suministra a la red una base de datos que incluye tanto los datos a clasificar como la etiqueta asociada a cada dato (la solución de cada caso). Por ejemplo, si queremos desarrollar una aplicación que clasifique fotos en las que aparece un perro o un gato, el dataset suministrado estaría formado por fotos en las que aparecieran perros o gatos con su correspondiente solución. De esta manera, podríamos entrenar a nuestra red mostrándole cada foto junto a su etiqueta, con intención de que, tras el proceso de aprendizaje, la red sea capaz de distinguir entre fotos de perros y gatos sin haberle proporcionado la etiqueta. [21]

  En este tipo de redes, una base de datos de calidad es crucial para el buen desempeño de la aplicación desarrollada.

- **Retropropagación:** Para realizar el aprendizaje supervisado, la red neuronal se vale del algoritmo conocido como algoritmo de *Backpropagation* que emplea un ciclo de propagación en dos fases. En primer lugar, cuando los estímulos de entrada entran por primera vez en la red neuronal, las neuronas están configuradas con unos pesos aleatorios. Los estímulos se propagan por las capas y neuronas hasta llegar a la generación de una salida, esta salida se compara con la salida deseada y se parametriza un error. Una vez tenemos el error este se propaga desde la capa de salida hasta la entrada pasando por todas las capas que contribuyen al error de salida final. Calculando el gradiente de la función de pérdidas para cada parámetro conseguimos encontrar la dirección en la magnitud varía de manera máxima y positiva, en nuestro caso, la dirección en la que nuestra función de pérdidas aumenta si variamos los pesos $\boldsymbol{w_{n\,y}}$ sesgos $\boldsymbol{b}$. Tras ello se modifica la dirección a la opuesta y así conseguimos encontrar la dirección (valores $\boldsymbol{w_n}$ y $\boldsymbol{b}$) en la que nuestra función de pérdidas (loss function) disminuye de manera máxima, y tras varias iteraciones o épocas la función de error va disminuyendo hasta llegar a un mínimo. [21]

6.2. Creación de la base de datos

En nuestro caso particular, para llegar al objetivo de la clasificación del movimiento de cada dedo, contamos con 5 entradas de información que se corresponden con las señales EMG de cada uno de los 5 pares de electrodos colocados en el antebrazo del usuario. La ubicación de estos electrodos viene dada como se muestra en la Figura.2.11 ya que, con esta configuración, conseguimos el menor solapamiento de señales posible.

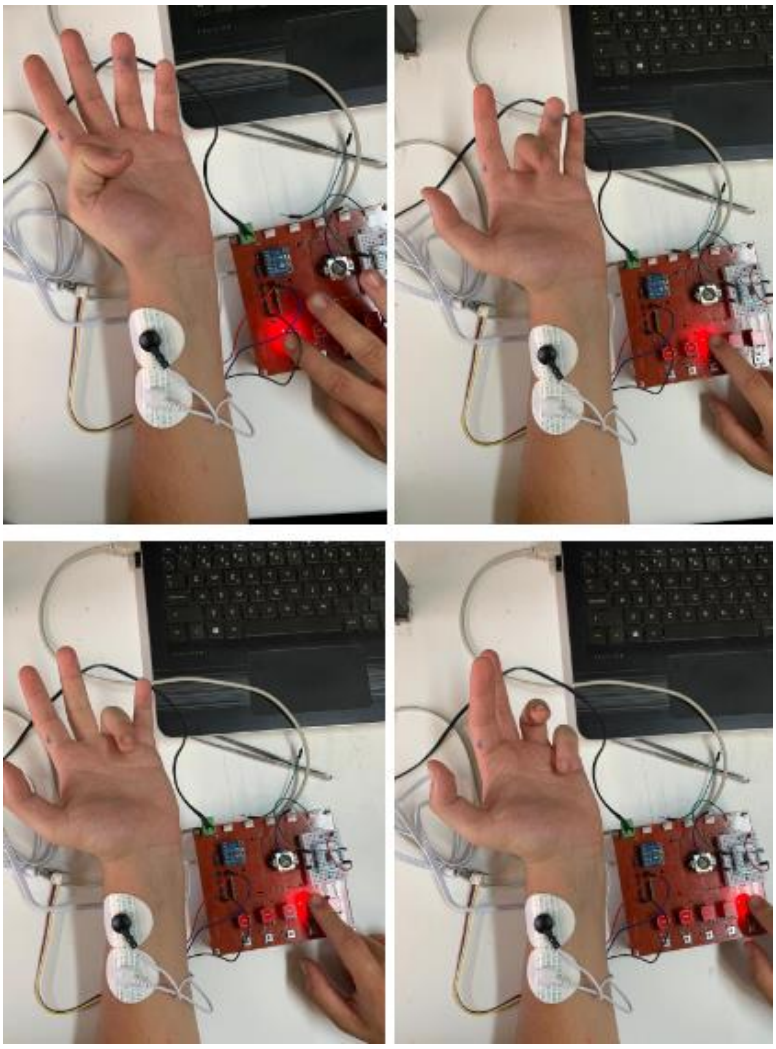


Figura.6.1.4 *Momento de adquisición de datos de los diferentes dedos para la Posición 1 ("MaxPulgar")*

Para la adquisición de los datos del dataset, se ha realizado un programa en Arduino debido a su sencillez en la toma de datos. El programa está diseñado para capturar cómo interfiere el movimiento de cada dedo en cada una de las ubicaciones de los electrodos. Por ejemplo, en el caso ilustrado en la Figura.6.2.1, se está llevando a cabo la recogida de datos de la posición 1 (ubicación en la que se maximiza la señal del movimiento del pulgar). El procedimiento consiste en realizar la contracción de cada dedo mientras se está pulsando el botón correspondiente (de izquierda a derecha: pulgar, índice, corazón, anular y meñique). De esta manera, el programa detecta el "peak" de la señal mientras está siendo pulsado un botón, lo almacena y lo etiqueta. En la Tabla 6.2.1 se puede apreciar el extracto de una toma de la señal detectada por cada posición para cada dedo. En el ANEXO XX se encuentra el detalle de este programa.

Posición 1	MaxPulgar: 499	MaxIndice: 384	MaxMedio: 366	MaxAnular: 375	MaxPinky: 385
Posición 2	MaxPulgar: 370	MaxIndice: 381	MaxMedio: 367	MaxAnular: 389	MaxPinky: 388
Posición 3	MaxPulgar: 510	MaxIndice: 451	MaxMedio: 368	MaxAnular: 427	MaxPinky: 412
Posición 4	MaxPulgar: 398	MaxIndice: 389	MaxMedio: 369	MaxAnular: 389	MaxPinky: 394
Posición 5	MaxPulgar: 415	MaxIndice: 392	MaxMedio: 370	MaxAnular: 402	MaxPinky: 410

Tabla.6.2.1 *Extracto de una toma de la señal detectada por cada posición para cada dedo*

Dado que la repetibilidad de los valores de las señales EMG es baja, es necesario un buen número de datos para que su rendimiento sea el deseado. Por ello, se ha decidido tomar 20 muestras del por cada dedo en cada posición, haciendo un total de 200x5 muestras.

Una vez recopilado todos los datos, estos se organizan para hacer nuestro dataset. Para ello se han invertido filas y columnas de manera que tenemos el valor que captura cada par de electrodos (entradas) cuando movemos un dedo en concreto (salida). De esta manera conseguimos un vector de entrada de 5 datos y se etiqueta la salida correspondiente para las configuraciones tomadas.

P1 PULGAR	P2 INDICE	P3 MEDIO	P4 ANULAR	P5 PINKY	SALIDA
MaxPulgar: 496	MaxPulgar: 370	MaxPulgar: 510	MaxPulgar: 398	MaxPulgar: 415	1
MaxPulgar: 499	MaxPulgar: 405	MaxPulgar: 536	MaxPulgar: 403	MaxPulgar: 405	1
MaxIndice: 384	MaxIndice: 381	MaxIndice: 451	MaxIndice: 389	MaxIndice: 392	2
MaxIndice: 395	MaxIndice: 393	MaxIndice: 429	MaxIndice: 387	MaxIndice: 388	2
MaxMedio: 366	MaxMedio: 385	MaxMedio: 421	MaxMedio: 392	MaxMedio: 395	3
MaxMedio: 369	MaxMedio: 387	MaxMedio: 409	MaxMedio: 403	MaxMedio: 405	3
MaxAnular: 375	MaxAnular: 389	MaxAnular: 427	MaxAnular: 389	MaxMedio: 402	4
MaxAnular: 377	MaxAnular: 390	MaxAnular: 446	MaxAnular: 395	MaxAnular: 408	4
MaxPinky: 385	MaxPinky: 388	MaxPinky: 412	MaxPinky: 394	MaxPinky: 410	5
MaxPinky: 382	MaxPinky: 393	MaxPinky: 415	MaxPinky: 390	MaxPinky: 413	5

Tabla.6.2.2 *Extracto de vectores de entrada y su etiqueta de salida*

6.3. Red neuronal Matlab

Para la creación de la red neuronal se ha usado la Deep Learning Toolbox de Matlab, la cual incluye diferentes herramientas que permiten la implementación de redes neuronales de una manera sencilla y rápida. Esta ToolBox, entre otras cosas, contiene una herramienta llamada Neural Network que contiene a su vez cuatro *wizards* que nos ayudan a solucionar 4 tipos típicos de problemas diferentes. En concreto nuestro problema es del tipo de *clasificación* por lo que usaremos la herramienta *Neural Pattern Recogniton*. Esta herramienta nos permite seleccionar los datos, crear y entrenar una red y evaluar su rendimiento mediante la entropía cruzada y matrices de confusión.

6.3.1 Configuración red neuronal

- **Selección de datos:** En primer lugar, se debe cargar la base de datos separándose entre *inputs* y *targets*. El entorno permite el formato *.xlsx* por lo que se ha cargado la base de datos anterior como *input* eliminando de ella las salidas y los caracteres no numéricos. Del mismo modo se ha preparado otro archivo *.xlsx* únicamente con los Targets correspondientes y se ha cargado. La herramienta de *Pattern Recognition* únicamente acepta salidas de tipo booleano, es decir, en lugar de tener una salida con 5 posibles modos, tenemos 5 salidas de las cuales solo puede estar activa una al mismo tiempo.

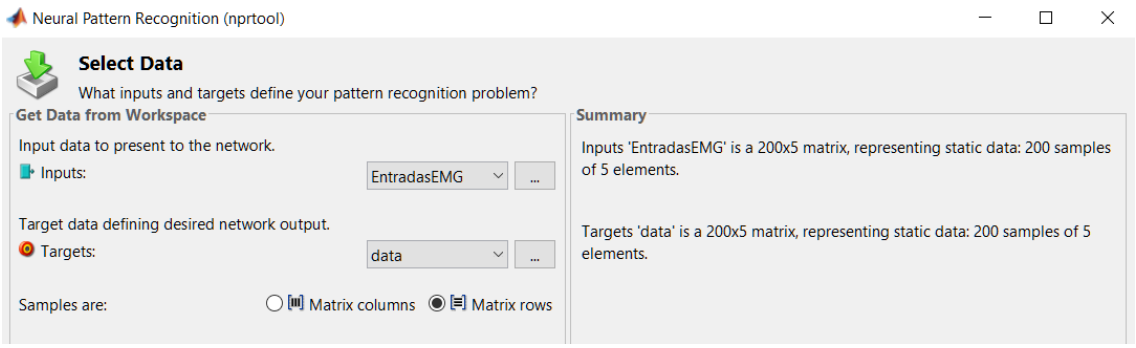


Figura.6.3.5 *Selección de datos para el data set*

- **Configuración samples:** Uno de los pasos más importantes al establecer nuestro dataset, es configurar el porcentaje de muestras que irán destinados a entrenar la red, a realizar la validación y el *testing*. Para tener un buen comportamiento de la red deberemos establecer entre el 70 y 80% de los samples para el entrenamiento de la misma, el restante repartido entre la validación de la red y su testeo. Del mismo modo, los samples de la base de datos deben mostrarse en un orden aleatorio para evitar falsas interpretaciones de la red. En la etapa de entrenamiento se le muestran los datos y en cada iteración el sistema va ajustando los parámetros de las neuronas en función de su error. Los samples destinados a la validación se utilizan para medir la generalización de la red y para detener el entrenamiento cuando la generalización deja de mejorar (para evitar el sobreentrenamiento). Por último, los samples destinados al *testing* no tienen ningún efecto sobre el entrenamiento, se utilizan para realizar una medida independiente del rendimiento de la red durante y después del entrenamiento, esto nos permite realizar un análisis del funcionamiento y rendimiento.

  En nuestro caso se ha establecido un 70% (140 samples) para el entrenamiento, 15% (30 samples) para la validación y el 15% (30 samples) restante para el *testing*.

- **Configuración número de neuronas:** En esta herramienta, la red que podemos programar puede tener únicamente una capa oculta, sin embargo, podemos configurar el número de neuronas que la componen. El último parámetro que configurar antes de proceder al entrenamiento es establecer el número de neuronas en la capa oculta de la red de reconocimiento de patrones. No obstante, la elección de este número no tiene otra metodología aparte de la prueba y error dentro de unos márgenes razonables. La red viene preconfigurada con 10 neuronas en la capa y se han probado diferentes configuraciones desde 5 a 300 que pueden ser consultadas en el ANEXO XX. Tras estas pruebas, se ha encontrado que el mejor rendimiento de la red se da con un número de 25 neuronas en la capa oculta.
-

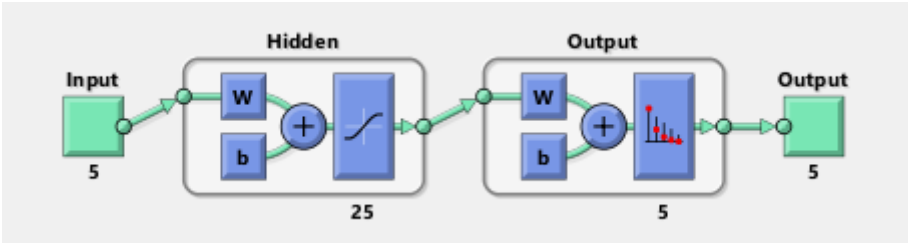


Figura.6.3.6 *Diagrama de la red neuronal*

En la Tabla.6.3.1 se muestra el resumen de la configuración elegida para la red neuronal completa.

Porcentajes Data Samples	**Training:** 70%	**Validation:** 15%	**Testing:** 15%
Data division	Random (dividerand)		
Training	Scaled Conjugate Gradient (trainscg)		
Performance	Cross-Entropy (crossentropy)		
Número neuronas hidden layer	25		

Tabla.6.3.1 *Resumen configuración elegida*

6.3.2 Análisis rendimiento obtenido

Como se ha mencionado en el apartado anterior han sido probadas diversas configuraciones y de entre todas ellas se ha elegido por su desempeño la configuración mostrada en la Taabla.6.3.1.

A continuación, se procede a analizar el rendimiento de la red:

- **Matriz de confusión:** Esta herramienta nos permite valorar cómo de bueno es un sistema de clasificación. En concreto, en el eje y se colocan las salidas predichas y en el x las soluciones correctas, de esta manera, cuanta más concentración haya en la diagonal se corresponderá con un mejor rendimiento. Por tanto, los cuadrados verdes se corresponden con predicciones acertadas y por contra los rojos con predicciones falsas. Asimismo, los cuadrados blancos muestran la precisión en la predicción y en gris la exactitud.

Figura.6.3.7 *Matriz de confusión*

Como podemos observar en la figura, encontramos una gran calidad de rendimiento tanto en el entrenamiento como en la validación y testing. Cabe destacar que la etapa de testing mantiene una precisión y exactitud del 100%.

Si nos fijamos en la matriz total de confusión podemos analizar el rendimiento de la red para cada uno de los dedos. Por encima de todos, destaca la precisión de los dedos pulgar e índice siendo del 100% y el 97.1% respectivamente. Esto se debe a que el movimiento de estos dedos, y en especial el pulgar, viene dado por la contracción de grupos musculares que producen más fuerza que el resto y se encuentran en ubicaciones que dotan a sus señales de una mayor robustez ante las interferencias causadas por el movimiento de otros dedos.

Por otro lado, si observamos las predicciones del resto de dedos, podemos observar que la precisión desciende levemente debido a que, como se muestra en el apartado 6.2, la diferencia de potencial generada por estos dedos registra unos valores parecidos, al mismo tiempo que las interferencias en las señales de estos grupos musculares son mayores.

- **Función de pérdidas (loss function):** Esta función evalúa la desviación entre las predicciones realizadas por la red y las soluciones reales a lo largo de las épocas de entrenamiento. Cuanto menor sea su valor más eficiente y precisa deberá ser la red evaluada. De esta manera, como se ha explicado (apdo. 6.1), se van ajustando los parámetros neuronales época tras época hasta llegar al mínimo error.

  En nuestro caso concreto, atendiendo a la Figura.6.3.4, podemos apreciar que la curva de pérdidas tiene una forma adecuada y lógica. Todas las líneas dibujan curvas paralelas y mantienen una tendencia negativa indicando el correcto transcurso del entrenamiento. Asimismo, la red neuronal ha encontrado su mejor rendimiento de validación en la época número 41 y por tanto los parámetros internos de la red deberán ser los correspondientes a esta iteración para evitar el sobreentrenamiento.

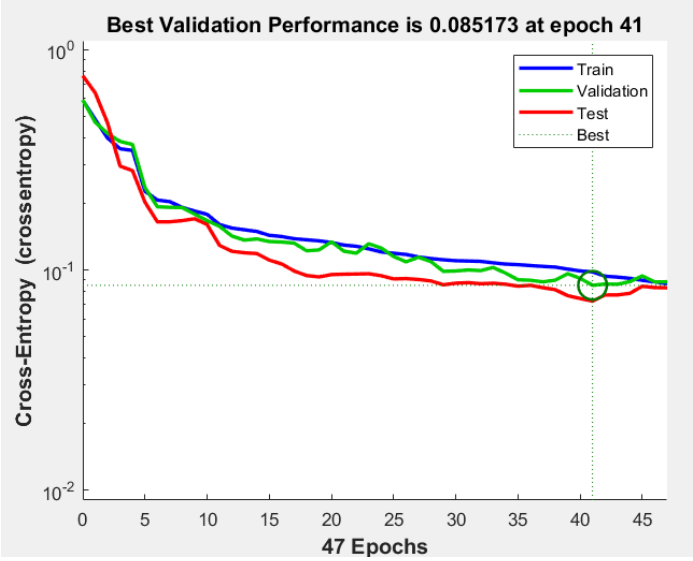


Figura.6.3.8 *Loss-function*

- **Receiver Operating Characteristic (ROC):** Esta herramienta nos muestra la relación entre las predicciones positivas acertadas y las falsas separadas por cada clase. De esta manera podemos analizar el rendimiento de clasificación del movimiento de cada uno de los dedos.

  Atendiendo a la Figura.6.3.5 podemos observar (y confirmar lo analizado con la matriz de confusión) que aunque todos los dedos mantienen una precisión en la predicción, los dedos pulgar e índice destacan por encima del resto.

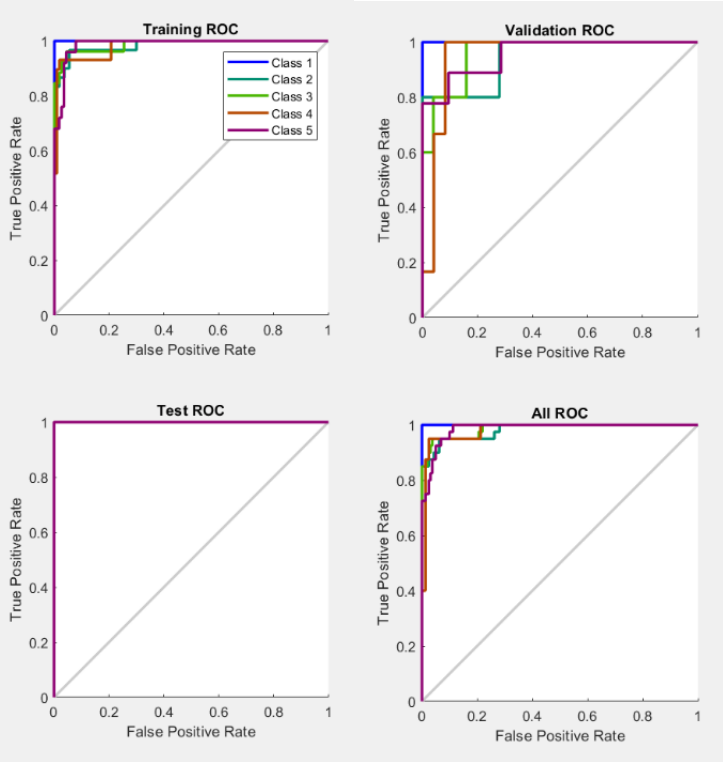


Figura.6.3.9 *ROC*

- **Histograma de Error:**

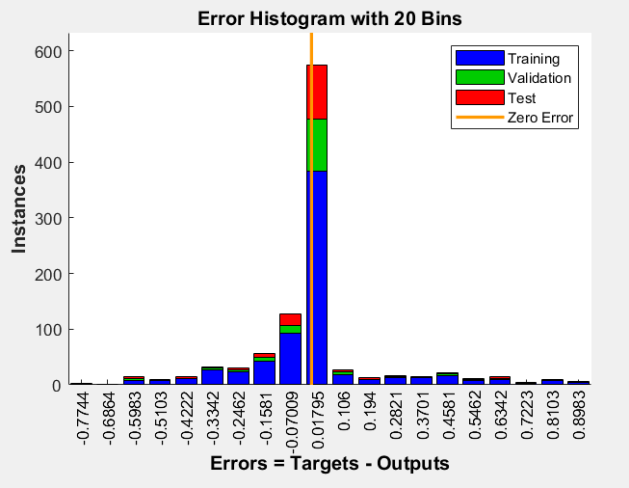


Figura.6.3.10 *Histogram Error*

En definitiva, tras el análisis del rendimiento de la red podemos destacar una precisión global del 92% y de un 86% en el caso del dedo más restrictivo (el meñique), y un valor del error en la función de pérdida en torno a 0.017. Estos valores denotan una gran fiabilidad en la clasificación de las señales por lo que la implementación de la red en el control de la prótesis se considera viable y adecuada para el objetivo del proyecto.

6.4. Implementación en modelo de control

Una vez comprobado el correcto funcionamiento de la red y su viabilidad para el objetivo procedemos a realizar un modelo en Matlab Simulink para el control de la prótesis.

Al igual que en el resto de controles, nos hemos valido de diferentes *ToolBox* y herramientas de Matlab. En concreto para este caso, hemos utilizado las *ToolBox DeepLearning* y *Simulink Support Package for Raspberry Pi Hardware* y la herramienta StateFlow.

Atendiendo a la Figura.6.4.1 podemos observar la estructura del control de una manera clara. El sistema recoge las entradas de las señales EMG como un vector de entrada de 5 integers por medio del bloque *I2C Master Read*, este vector entra directamente en el bloque *Pattern Recognition Neural Network* que hemos desarrollado en el apartado anterior para la clasificación del movimiento. De este bloque sale una señal que, pasando por un demultiplexor, conseguimos separar en las 5 señales que nos marcan el estado de cada dedo según la predicción realizada por la red. Estas señales no son booleanas si no que devuelven un valor decimal entre 0 y 1 que nos marca la probabilidad que tiene cada movimiento de ser el realizado. Estas señales entran en el bloque de *StateFlow* y dependiendo de sus valores, si alguna supera el 70% de probabilidad, la máquina de estados configura las salidas correspondientes a los servomotores para realizar el movimiento predicho por la red (Ver Figura.6.4.2). Una vez se realiza el movimiento, el sistema mantiene la posición durante 1500 milisegundos y vuelve a la posición de reposo a la espera de otra señal de entrada.

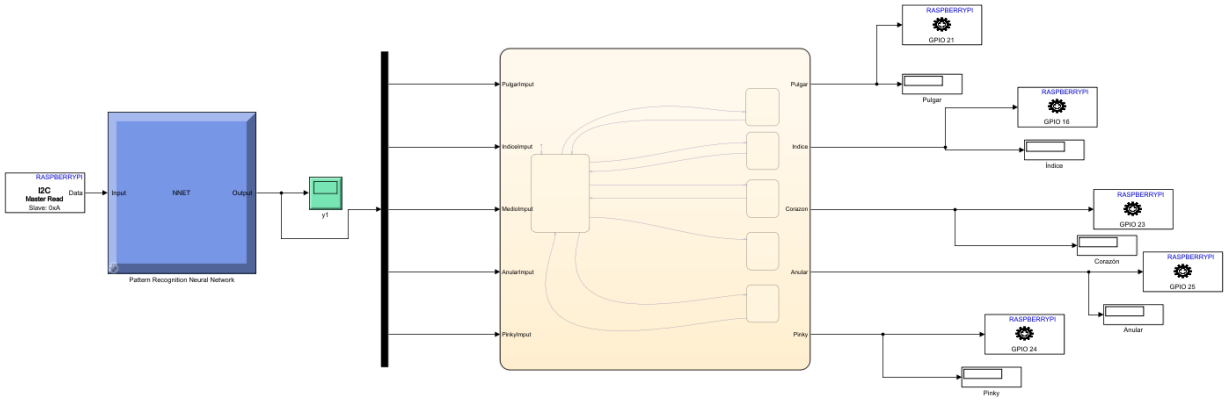


Figura.6.4.11 *Diagrama de control con redes Neuronales*

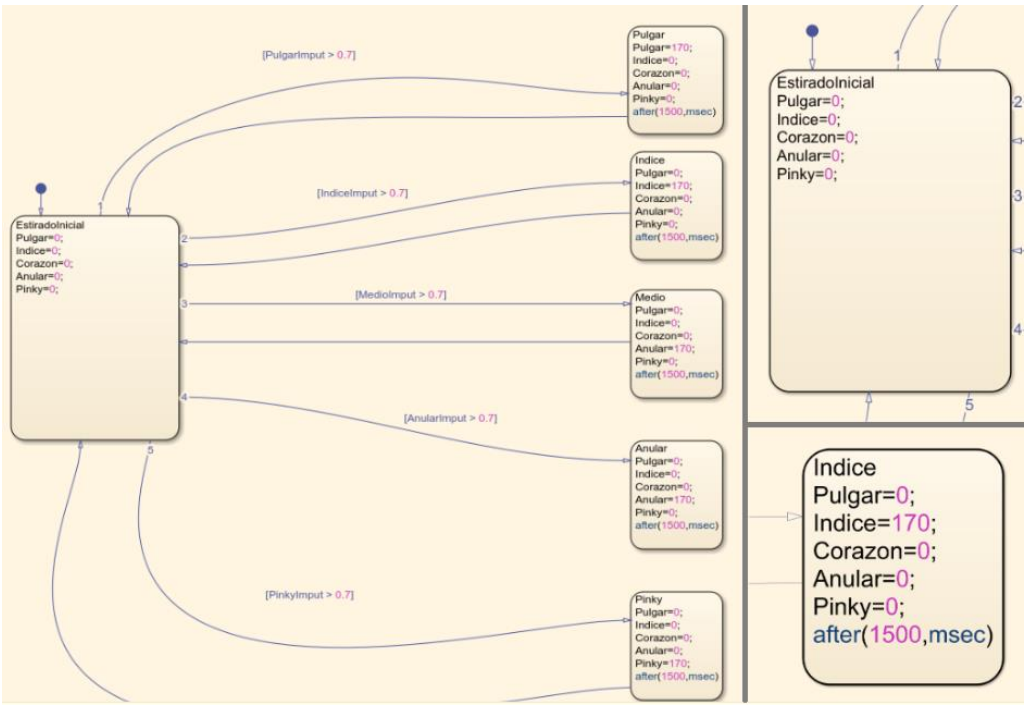


Figura.6.4.12 *Detalle máquina de estados de StateFlow para configuración salida*

Las siguientes figuras corresponden al funcionamiento del control modelado en el caso concreto del movimiento del dedo pulgar:

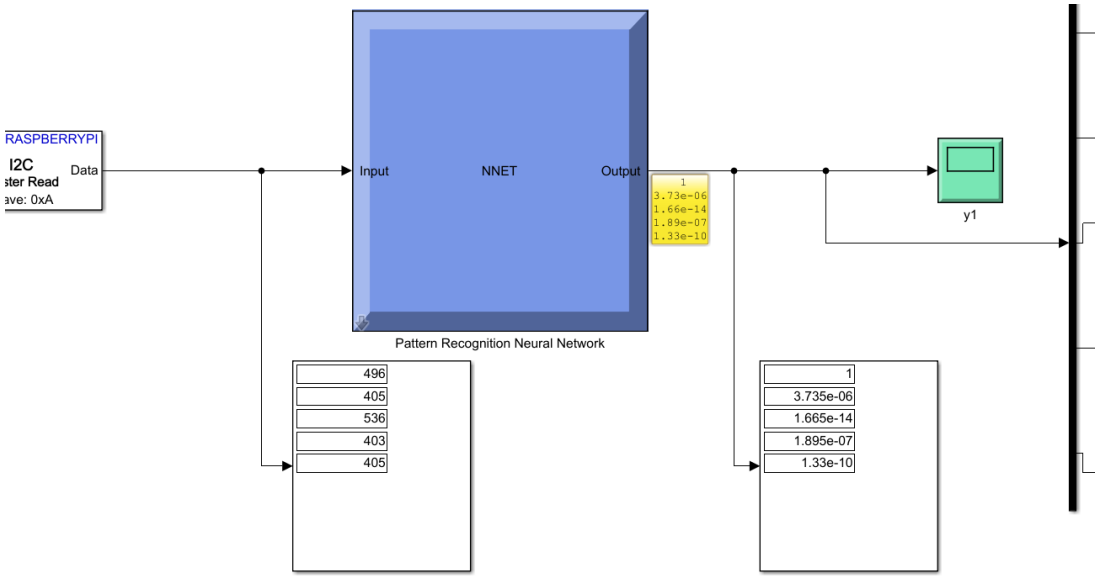


Figura.6.4.13 ***Detalle del desempeño de la red neuronal***. *Como podemos observar (display izquierdo) las señales leídas entran en la red neuronal y tras el paso por el bloque* ***PRNN*** *generan 1 vector de salida tipo double (display derecho) con valores correspondientes a la probabilidad de cada salida. En este caso, el sistema predice con un 99,99% de probabilidad que se trata de un movimiento de pulgar. Esta salida se separa en 5 mediante un demultiplexor y se les asigna el nombre correspondiente para la subsiguiente configuración de salidas.*

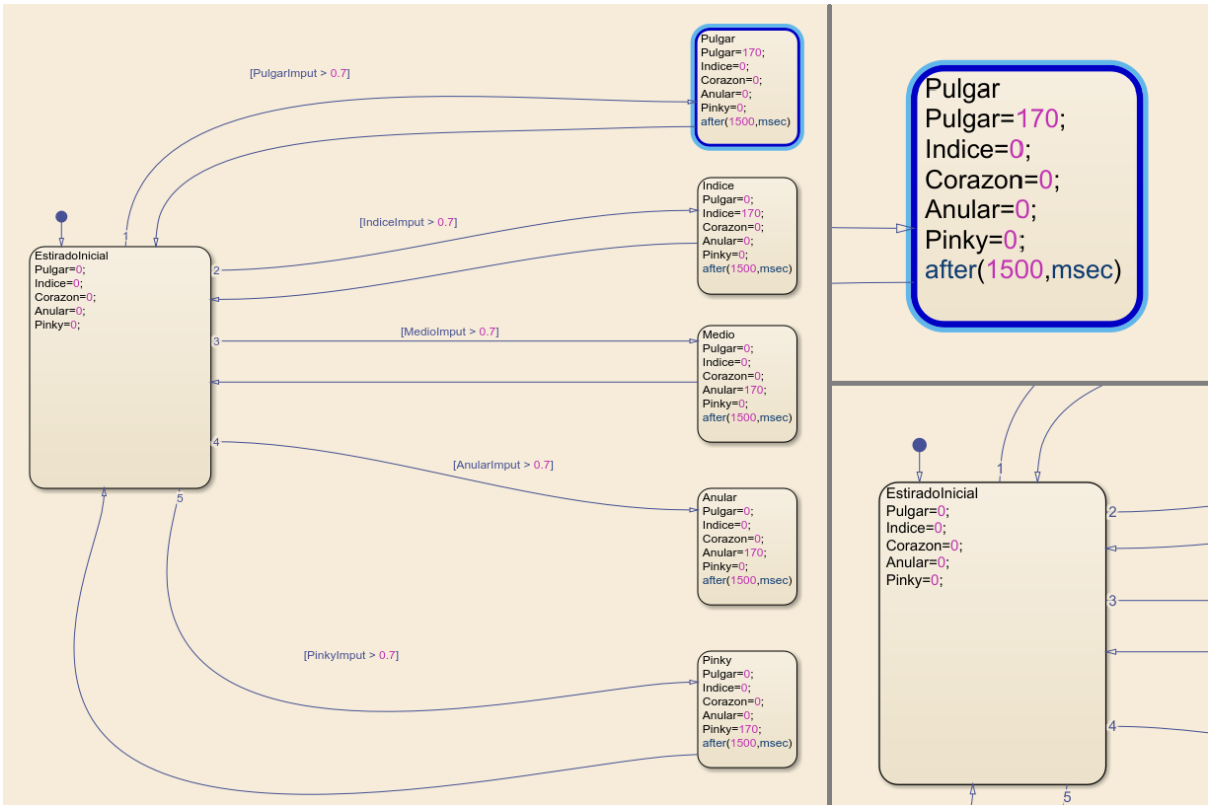


Figura.6.4.14 ***Detalle del desempeño de la máquina de estados.*** *Dado que existe una señal con un valor mayor a 0.7 (PulgarImput), se cumple la condición nº1 y el diagrama cambia de estado al estado "Pulgar" el cual configura las salidas con un valor de 170º para el pulgar y de 0º para el resto. Tras 1500 milisegundos el diagrama cambiará de estado a "EstiradoInicial" que contiene la configuración de salidas correspondiente al estado de reposo de la mano.*

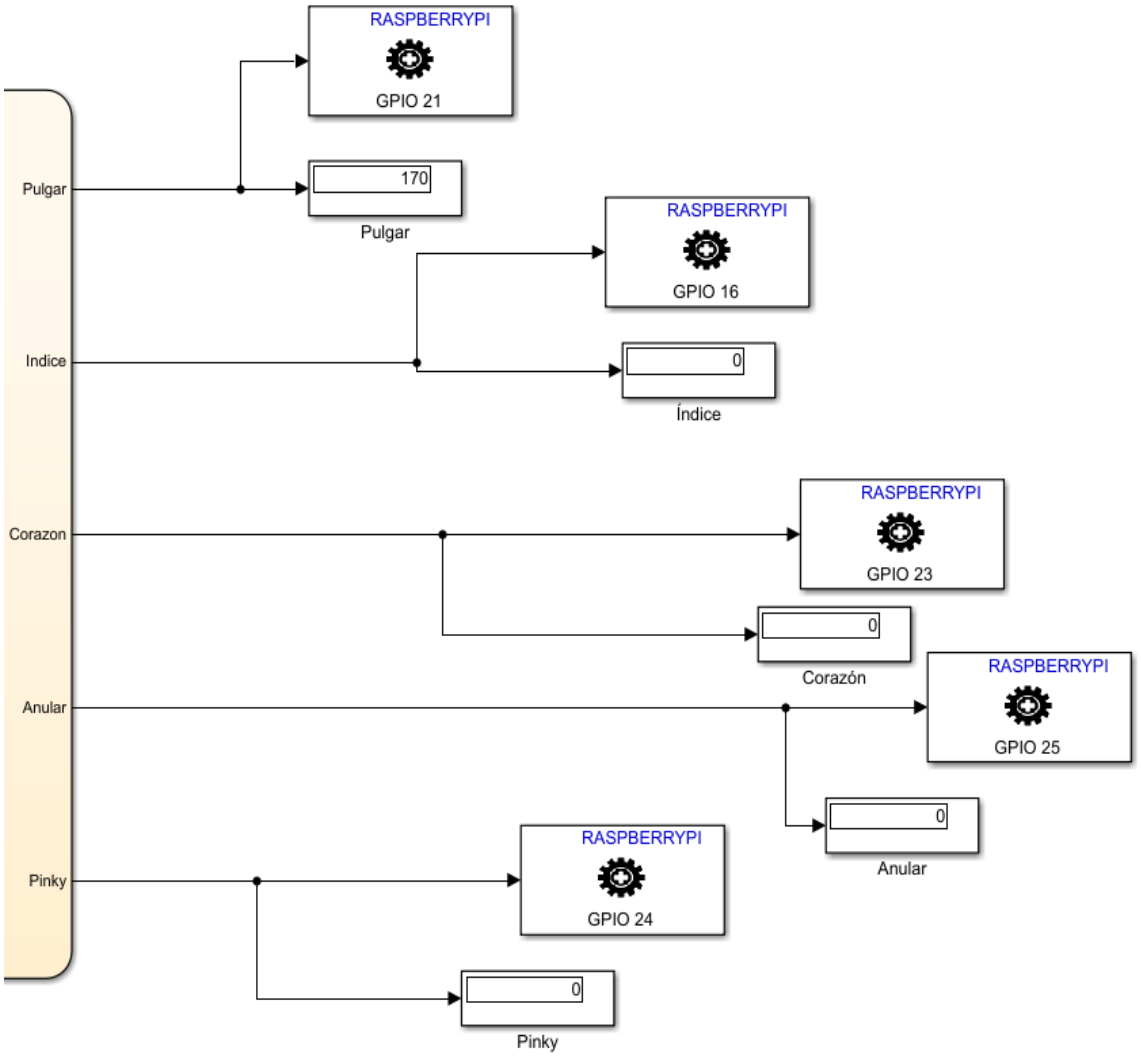


Figura.6.4.15 ***Detalle salidas servomotores.*** *Del bloque StateFlow salen las señales de cada dedo. Estas señales se conectan a 5 bloques "Standard Servo Write" que tienen una dirección diferente para cada dedo y modulan una salida PWM que se corresponde con los grados indicados. Como podemos observar, las salidas finales que modulan el movimiento de los servos son 170º para el servo del pulgar y 0 para el resto.*

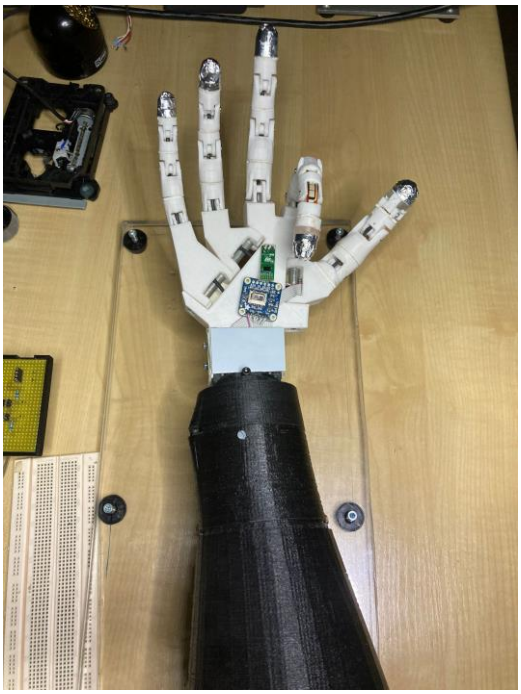


Figura.6.4.16 *Movimiento resultante del Índice*

7. Conclusiones y trabajo futuro

Tras el desarrollo del proyecto se consideran satisfechos todos los objetivos planteados al inicio de este. Se ha logrado construir una prótesis robótica impresa en 3D y se han conseguido desarrollar diferentes controles de esta de manera satisfactoria.

Tras el estudio y análisis de una gran variedad de elementos, se ha logrado implementar una amplia sensorización en la prótesis, consiguiendo así, dotarla de una mayor capacidad de interacción con el entorno. Asimismo, esta sensorización nos ha permitido realimentar el control de la prótesis abriéndonos un abanico de posibilidades inalcanzables en la prótesis anterior.

Gracias a esta sensorización, se ha llevado a cabo el control de dos tipos de agarre diferentes que permiten, a un supuesto usuario, la realización de agarres básicos de diferente fuerza y precisión. Debido a los plazos de entrega y para no alargar la memoria con modelos redundantes, se han realizado los dos tipos de agarre más relevantes, sin embargo, dado el satisfactorio desarrollo de estos modelos de control, queda demostrada la capacidad de la prótesis y de la metodología realizada. En esta línea, se propone desarrollar nuevos agarres y aplicaciones de uso cotidiano que puedan ser seleccionados por el usuario de una manera sencilla y rápida.

También se ha logrado, mediante la implementación de una red neuronal, la clasificación singular del movimiento de cada uno de los dedos a raíz de 5 señales EMG y el consiguiente control de la prótesis. Para alcanzar este objetivo, se han desarrollado métodos para la adquisición y etiquetado de las señales EMG que han resultado eficientes, funcionales y fiables. Una vez creada nuestra base de datos, se ha desarrollado la red neuronal la cual ha resultado ser altamente precisa y fiable. Quizás, esta precisión se debe a un posible sobreentrenamiento de la red neuronal debido al escaso tamaño de la base de datos y sobre todo a la total diferencia de señales que existe entre usuarios. Sin embargo, podemos concluir que, para un mismo usuario y condiciones, nuestra red es muy eficaz en la clasificación de movimientos. Por ello, se propone como trabajo futuro el desarrollo de una aplicación que automatice la toma de datos y el desarrollo de una base de datos. De esta manera, en unos 15 minutos, el usuario podría tener una red neuronal entrenada y funcional, capaz de clasificar los movimientos, y a su vez, seguir prosiguiendo con el entrenamiento y mejora del control mientras el usuario la usa de manera cotidiana.

Por otro lado, aunque no es un objetivo concreto del proyecto, es preciso señalar el diseño y montaje de los prototipos y circuitos integrados realizados. Estos han permitido estudiar multitud de comportamientos de sensores y accionamientos, al mismo tiempo que simular diferentes situaciones que han dotado de una gran fiabilidad a los resultados y decisiones finales. Del mismo modo, subrayar y agradecer la inestimable ayuda de Carlos Millán por su gran implicación en el montaje y supervisión de estos prototipos.

Asimismo, cabe destacar que, debido al carácter multidisciplinar de este proyecto, ha tenido lugar una gran etapa de investigación y estudio en diferentes campos anteriormente inexplorados por mí, como son la electromiografía, la tecnología de

impresión 3D o el uso de las redes neuronales. Por ello, a nivel personal, enfatizar una grata sensación de satisfacción ya que se ha abordado un proyecto de una extraordinaria amplitud y que pertenece a campos de estudio todavía en desarrollo con un futuro prometedor. Del mismo modo, el objetivo de desarrollar una solución de bajo coste que busca mejorar la calidad de vida de personas que han sufrido una amputación, ha sido y es, una gran fuente de motivación para el trabajo realizado y futuro.

Índice de Figuras

Índice de las

Referencias

[1] Dan Conyers, CPO, FAAOP; and Pat Prigge, CP, LP, FAAOP(D). “The First 12 Months After Upper-Limb Amputation”. *inMotion, Volume 21, Issue 1 January/February 2011*

[2] Kim Norton. “A Brief History of Prosthetics”. *inMotion, Volumen 17 · Número 7 · Noviembre/Diciembre 2007*

[3] Martin J, Pollock A, Hettinger J. Microprocessor Lower-Limb Prosthetics: Review of Current State of the Art. *JPO: Journal of Prosthetics and Orthotics 2010;22(3):183-93.*

[4] X. Li, L. Tian, Y.Zheng, O.W. Samuel, P. Fang, L. Wang, and G. Li. "A new strategy based on feature filtering technique for improving the real-time control performance of myoelectric prostheses". *Journal of Neuroscience Methods. Elsevier ScienceDirect. Vol.: 70, September 2021; pp.: 1-11*

[5] https://cordis.europa.eu/project/id/BMH4960424/es

[6] Harold A. Romo, Esp., Judy C. Realpe, Ing., Pablo E. Jojoa, PhD. “Surface EMG Signals Analysis and Its Applications in Hand Prosthesis Control ”. Universidad del Cauca

[7] https://www.dalcame.com/emg.html#.Ybi91ºPdDMJEY

[8]https://backyardbrains.com/experiments/RobotHand

[9] https://rambal.com/presion-peso-nivel-flex/250-sensor-flex.html

[10] Interlink Electronics FSR Force Sensing Resistors. FSR Integration Guide (http://www.electronicoscaldas.com/datasheet/FSR-Integration_Guide_Interlink.pdf)

[11]https://www.alldatasheet.es/datasheet-pdf/pdf/412635/SHARP/GP2Y0A21YK0F.html

[12]https://www.mouser.es/datasheet/2/389/vl6180x-satel-1799840.pdf

[13]https://www.ti.com/lit/ds/symlink/lmt85.pdf?ts=1654510311108&ref_url=https%253A%252F%252Fwww.google.com%252F

[14] https://www.adafruit.com/product/1362

[15] https://inmoov.fr/

[16] https://www.ti.com/lit/ds/symlink/ina219.pdf

[17]https://www.researchgate.net/publication/263353011_DISENO_Y_CONSTRUCCION_DE_UN_GUANTE_DE_DATOS_MEDIANTE_SENSORES_DE_FLEXIBILIDAD_Y_ACELEROMETRO

[18]CNSE
https://www.cnse.es/inmigracion/index.php?option=com_content&view=category&id=19&Itemid=236&lang=es

[19] Reyes Tejedor, M. (2007), "Sobre el estatuto lingüístico de las lenguas de señas", en *Philologia Hispalensis, Sevilla: Universidad de Sevilla, 21, pp. 1-19.*

[20] Pablo Bonet, J. de (1620) "Reduction de las letras y Arte para enseñar á ablar los Mudos." *Ed. Abarca de Angulo, Madrid, ejemplar facsímil accesible en la Biblioteca Histórica de la Universidad de Sevilla*

[21] Carlos Santana Vega. ¿Qué es el Machine Learning?¿Y Deep Learning? Un mapa conceptual | *DotCSV. Youtube. 2017. url:*
https://www.youtube.com/watch?v=KytW151dpqU& ab_channel=DotCSV.

[22]Vicente Rodríguez - Conceptos básicos sobre redes neuronales
https://vincentblog.xyz/posts/conceptos-basicos-sobre-redes-neuronales

[23] https://empresas.blogthinkbig.com/como-interpretar-la-matriz-de-confusion-ejemplo-practico

[24] J.S Artal-Sevil. "Diseño de un Brazo Robótico de bajo coste controlado por sensores EMG superficiales", *Congreso TAEE, Zaragoza, 2018*

ANEXOS

Anexo A. Algorítmica

Tras la explicación detallada del grueso del proyecto, es necesario mostrar y profundizar en los programas desarrollados, así como el procedimiento seguido para el desarrollo y uso de los mismos.

Como se ha ido recordando durante la memoria, el proyecto ha utilizado la plataforma Arduino para la toma de señales y algún sencillo control de los prototipos utilizados para los estudios.

A.1. Control movimiento mediante Joystick

Dada la necesidad de poder controlar de una manera sencilla y rápida la posición y velocidad del dedo para hacer posible la parametrización de los sensores, se ha desarrollado un pequeño algoritmo que nos permite modular la posición y velocidad de avance del servomotor de una manera intuitiva.

El programa consiste en, dependiendo de la posición del joystick, determinar la dirección y velocidad del movimiento. Para ello se lee la salida analógica del eje X del joystick y se varía el retardo entre señal y señal con el cual conseguimos variar la velocidad. De esta manera, a más distancia del punto de reposo 0 del joystick, más velocidad de giro para el servomotor. Por otro lado, para solucionar el cambio de velocidad que experimenta el dedo en el cambio de falange, se multiplica el retardo por un valor de ajuste que reduce en un porcentaje determinado la velocidad del servo dependiendo de la falange en la que se encuentre el movimiento.

A continuación, se muestra el código.

```
#include <Servo.h>
#include "INA226.h"
#include "Wire.h"

//#define       Pin_Touch_Pulgar
7                 // Aquí se define el pin del sensor capacitivo
#define       Pin_Touch_Indice
6                 // Aquí se define el pin del sensor capacitivo
#define       Pin_Touch_Medio
5                 // Aquí se define el pin del sensor capacitivo
#define       Pin_Touch_Anular
4                 // Aquí se define el pin del sensor capacitivo
#define       Pin_Touch_Menique
3                 // Aquí se define el pin del sensor capacitivo

//#define       PIN_SW_ON_Menique                    0
//#define       PIN_SW_ON_Indice                    1
#define       PIN_SW_Medio                         2
#define       PIN_SW_Indice                        8
#define       PIN_SW_Pulgar                       7

const int     Pin_Analogica_X_Joystick              =
A2;               // seleccionar la entrada para el sensor

Servo         Servo_Pulgar;  // crea el objeto servo
Servo         Servo_Indice;
Servo         Servo_Medio;
Servo         Servo_Anular;
Servo         Servo_Menique;

INA226        INA(0x40);

int           Posicion_Pulgar                      =
0;                // posicion del servo
int           Posicion_Indice                      = 0;
int           Posicion_Medio                      = 0;
int           Posicion_Anular                      = 0;
int           Posicion_Menique                      = 0;

int           Limite Minimo Servo         = 0;
```

```
int           Limite_Maximo_Servo         = 125;
int           Posicion_primera_falange    = 45;
int           Correccion_primera_falange  = 2;
int           Posicion_segunda_falange    = 90;
int           Correccion_segunda_falange  = 3.6;
int           Retardo                     = 100;
int           Valor_Joystick_X            = 0;                    //
variable que almacena el valor raw (0 a 1023)
int           Valor_Joystick_X_0_100      = 0;                    //
Valor  en %
int           Valor_Sensor_touch          = 0;
float         Tension_INA                 = 0;
float         Corriente_INA               = 0;
float         Potencia_INA                = 0;

void setup() {

   // Iniciar comunicación serie
   Serial.begin(250000);

   pinMode(PIN_SW_Indice,                   INPUT);
   pinMode(PIN_SW_Medio,                    INPUT);
   pinMode(PIN_SW_Pulgar,                   INPUT);

//   pinMode(Pin_Touch_Pulgar,                INPUT);
   pinMode(Pin_Touch_Indice,                INPUT);
   pinMode(Pin_Touch_Medio,                 INPUT);
   pinMode(Pin_Touch_Anular,                INPUT);
   pinMode(Pin_Touch_Menique,               INPUT);

   Servo_Pulgar.attach(9);
   Servo_Indice.attach(10);
   Servo_Medio.attach(11);
   Servo_Anular.attach(12);
   Servo_Menique.attach(13);
}

void loop() {
```

```
delay(Retardo);
Valor_Joystick_X = analogRead(Pin_Analogica_X_Joystick);   //
realizar la lectura
Valor_Joystick_X_0_100 = int(map(Valor_Joystick_X, 0, 1023, 0.0,
100.0));   // cambiar escala a 0.0 % - 100.0 %

Serial.println  (Posicion_Pulgar);
//Serial.print  (" Retardo=");
//Serial.println(Retardo);
//Serial.println(Valor_Joystick_X_0_100);

           if ( Valor_Joystick_X_0_100 <= 50 )
              {
              Retardo = int(map(Valor_Joystick_X, 0, 512, 1.0,
100.0));
                  if (Posicion_Pulgar >= 0)
                       {Retardo = Retardo;}
                  if (Posicion_Pulgar >= Posicion_primera_falange)
                       { Retardo = Retardo *
Correccion_primera_falange;}
                  if (Posicion_Pulgar >= Posicion_segunda_falange)
                       { Retardo = Retardo *
Correccion_segunda_falange;}
              }

          if ( Valor_Joystick_X_0_100 >= 52 )
              {
              Retardo = int(map(Valor_Joystick_X, 512, 1023, 100.
0, 1.0));
                  if (Posicion_Pulgar >= 0)
                        {Retardo = Retardo;}
                  if (Posicion_Pulgar >= Posicion_primera_falange)
                       {Retardo = Retardo *
Correccion_primera_falange;}
                  if (Posicion_Pulgar >= Posicion_segunda_falange)
                       {Retardo = Retardo *
Correccion_segunda_falange;}
              }

          if ( digitalRead(PIN SW Pulgar)==1 &&
```

```
Valor_Joystick_X_0_100<=50 )
               {
                 if (Posicion_Pulgar <= Limite_Maximo_Servo)
                     {
                      Posicion_Pulgar += 1;
                      Servo_Pulgar.write(Posicion_Pulgar);
                     }
                }

          if ( digitalRead(PIN_SW_Pulgar)==1 &&
Valor_Joystick_X_0_100>=52 )
               {
                 if (Posicion_Pulgar >= Limite_Minimo_Servo)
                     {
                      Posicion_Pulgar -= 1;
                      Servo_Pulgar.write(Posicion_Pulgar);
                     }
                }




          if ( digitalRead(PIN_SW_Indice)==1 &&
Valor_Joystick_X_0_100<=50 )
               {
                 if (Posicion_Indice <= Limite_Maximo_Servo)
                     {
                      Posicion_Indice += 1;
                      Servo_Indice.write(Posicion_Indice);
                     }
                }

          if ( digitalRead(PIN_SW_Indice)==1 &&
Valor_Joystick_X_0_100>=52 )
               {
                 if (Posicion_Indice >= Limite_Minimo_Servo)
                     {
                      Posicion_Indice -= 1;
                      Servo_Indice.write(Posicion_Indice);
                     }
```

```
                }


          if ( digitalRead(PIN_SW_Medio)==1 &&
Valor_Joystick_X_0_100<=50 )
               {
                 if (Posicion_Medio <= Limite_Maximo_Servo)
                     {
                      Posicion_Medio += 1;
                      Servo_Medio.write(Posicion_Medio);
                      Servo_Anular.write(Posicion_Medio);
                      Servo_Menique.write(Posicion_Medio);
                     }
                }

          if ( digitalRead(PIN_SW_Medio)==1 &&
Valor_Joystick_X_0_100>=52 )
               {
                 if (Posicion_Medio >= Limite_Minimo_Servo)
                     {
                      Posicion_Medio -= 1;
                      Servo_Medio.write(Posicion_Medio);
                      Servo_Anular.write(Posicion_Medio);
                      Servo_Menique.write(Posicion_Medio);
                     }
                }


}
```

A.2. Cámara termográfica

La cámara termográfica está basada en el sensor AMG883 el cual contiene a su vez una matriz de sensores IR infrarrojos de 8x8 (64 sensores). Se ha realizado un código que adjudica a cada componente del array de 64 números un color determinado para su posterior visionado en un display.

Por otro lado, el programa calcula la temperatura máxima y mínima, al mismo tiempo que calcula la temperatura media de la matriz.

```
#include <TFT.h>  // Arduino LCD library
#include <SPI.h>

int cs = 10;
int dc =  9;
int rst = 8;

TFT TFTscreen = TFT(cs, dc, rst);

// position of the line on screen
int xPos = 0;
int yPos = 0;

#include <Wire.h>
#include <Adafruit_AMG88xx.h>

Adafruit_AMG88xx amg;


//low range of the sensor (this will be blue on the screen)
#define MINTEMP 22

//high range of the sensor (this will be red on the screen)
#define MAXTEMP 34

float Temp_min = 0;
float Temp_max = 0;
float Temp_med = 0;
float Temp     = 0;

//the colors we will be using
const uint16_t camColors[] = {0x480F,
0x400F,0x400F,0x400F,0x4010,0x3810,0x3810,0x3810,0x3810,0x3010,0x3
010,
0x3010,0x2810,0x2810,0x2810,0x2810,0x2010,0x2010,0x2010,0x1810,0x1
810,
0x1811,0x1811,0x1011,0x1011,0x1011,0x0811,0x0811,0x0811,0x0011,0x0
011,
0x0011,0x0011,0x0011,0x0031,0x0031,0x0051,0x0072,0x0072,0x0092,0x0
0B2,
```

```
unsigned long delayTime;
float pixels[AMG88xx_PIXEL_ARRAY_SIZE];
uint16_t displayPixelWidth, displayPixelHeight;

void setup() {
    Serial.begin(9600);
    Serial.println(F("AMG88xx pixels"));

    bool status;

    // default settings
    status = amg.begin();
    if (!status) {
        Serial.println("Could not find a valid AMG88xx sensor,
check wiring!");
        while (1);
    }

    Serial.println("-- Pixels Test --");

    Serial.println();

    // initialize the display
  TFTscreen.begin();

  // clear the screen with a pretty color
  TFTscreen.background(255,255,255);
    displayPixelWidth = (TFTscreen.width() / 8)-9;
    displayPixelHeight = (TFTscreen.height() / 8)-5;
    delay(100); // let sensor boot up
}


void loop() {
    //read all the pixels
    amg.readPixels(pixels);
```

```
    Temp_max = 0;
    Temp_min = 100;
    Temp_med = 0;
    Temp = 0;

    //Serial.print("[");
    for(int i=1; i<=AMG88xx_PIXEL_ARRAY_SIZE; i++){

      //if (pixels[i] >= Temp_max) {Temp_max = pixels[i]}
      //if (pixels[i] <= Temp_min) {Temp_min = pixels[i]}
      Temp = Temp + pixels[i];
      if (pixels[i] >= Temp_max) {
      Temp_max = pixels[i];
      }
    }

    Temp = Temp / AMG88xx_PIXEL_ARRAY_SIZE;

    Serial.println(Temp_max);
    Serial.println(Temp_min);
    Serial.print ("Temperatura media = ");
    Serial.println(Temp);
    //Serial.println();
    //delay a second
    //delay(1000);

  //TFTscreen.setTextSize(1);
  //TFTscreen.stroke(255, 0, 255);
'/  TFTscreen.text(Temp, 0, 0);
'/  TFTscreen.text((int) Temp,0, 110);

  for(int i=0; i<AMG88xx_PIXEL_ARRAY_SIZE; i++){
    uint8_t colorIndex = map(pixels[i], MINTEMP, MAXTEMP, 0, 255)
    colorIndex = (uint8_t)constrain((int16_t)colorIndex,
{int16_t)0, (int16_t)255);
    //draw the pixels!
    TFTscreen.fillRect(displayPixelHeight * floor(i / 8),
ìisplayPixelWidth * (i % 8),
```

A.3. Dactilológico

Dado que el modelo de Simulink se carga en la Raspberry imposibilitando la modificación de las constantes del mismo, este pequeño programa se ha realizado para poder interactuar con el modelo mientras está en funcionamiento. El programa "traduce" cada letra del abecedario leída por teclado, a un número entero y lo envía vía I2C a la RaspBerry.

Para su implementación se ha añadido la biblioteca Wire.h que nos permite trabajar con I2C.

```
#include <Wire.h>

char letranueva;
byte salida = 0;
void setup() {

  Serial.begin(115200);
  letranueva = 0;
  Wire.begin(0x48); //i2c bus adresss 0x48
  Wire.onRequest(requestEvent); //register event

}

void loop() {
  delay(100);
    switch (letranueva){
    case 'a':
    salida=1;
    break;

    case 'b'|'B':
    salida=2;
    break;

    case 'c'|'C':
    salida=3;
    break;

    case 'd'|'D':
    salida=4;
    break;

    case 'e'|'E':
    salida=5;
    break;

    case 'f'|'F':
    salida=6;
    break;
```

```
    case 'z'|'Z':
    salida=27;
    break;
  }
}


void requestEvent(){
  Wire.write(salida);
  Serial.println(salida);
}
void serialEvent(){
  letranueva=Serial.read();
}
```

A.4. Control agarre general

Este programa ha sido explicado en detalle en Capítulo.5, a continuación, se muestra el código fuente más relevante.

```
int EMG = 0; int OutEMG = 0;
int Touch = 0; int OutTouch = 0;
int Fuerza = 0; int OutFuerza = 0;
int Temperatura = 0; int OutTemperatura = 0;
int Distancia = 0; int OutDistancia = 0;
int Agarre = 0; int OutAgarre = 0;

#define       OnEMG                    7
#define       OnTouch                  6
#define       OnFuerza                 5
#define       OnTemperatura            4
#define       OnDistancia              3
#define       OnAgarre                 0

void setup() {

    // Iniciar comunicación serie
   Serial.begin(250000);

   pinMode(OnEMG,                  INPUT);
   pinMode(OnTouch,                INPUT);
   pinMode(OnFuerza                INPUT);
   pinMode(OnTemperatura,          INPUT);
   pinMode(OnDistancia,            INPUT);
   pinMode(OnAgarre,               INPUT);

}

void loop() {

  if ( digitalRead(OnEMG)==1){
    EMG = analogRead(A1);
    if (EMG > 400) {
      OutEMG = 1;
    }
      else{
      OutEMG = 0;
    }
  }
```

```
  if ( digitalRead(OnDistancia)==1){
    Distancia = analogRead(A2);
    if (Distancia < 200) {
      OutDistancia = 1;
    }
      else{
      OutDistancia = 0;
    }
  }

  if ( digitalRead(OnTouch)==0){ //Activo en bajo

      OutTouch = 1;
    }
      else{
      OutTouch = 0;
    }

if ( digitalRead(Agarre)==1){

      OutAgarre = 1;
    }
      else{
      OutAgarre = 0;
    }
  }

  if ( digitalRead(OnFuerza)==1){

    if (OutAgarre = 1){
      Fuerza = analogRead(A3);
    if (Fuerza < 610) {
      OutEMG = 1;
    }
    }
    if (OutAgarre = 0){
      Fuerza = analogRead(A3);
    if (Fuerza < 610) {
      OutEMG = 1;
    }
```

```
    }

if ( digitalRead(onTemperatura==1){
    Distancia = analogRead(A4);
    if (Distancia > 80) {
      OutDistancia = 1;
    }
      else{
      OutDistancia = 0;
    }
  }

  }



}
```

A.5. Adquisición de datos para base de datos Red Neuronal

Para la adquisición de los datos del dataset del Capítulo.6, se ha realizado un programa en Arduino El programa está diseñado para capturar cómo interfiere el movimiento de cada dedo en cada una de las ubicaciones de los electrodos. De esta manera, el programa detecta el "peak" de la señal mientras está siendo pulsado un botón, lo almacena y lo etiqueta.

```
int PulgarPin = 7;
int ValPulgar = 0;
int maxPulgar = 0;

int IndicePin = 6;
int ValIndice = 0;
int maxIndice = 0;

int MedioPin = 5;
int ValMedio = 0;
int maxMedio = 0;

int AnularPin = 4;
int ValAnular = 0;
int maxAnular = 0;

int PinkyPin = 3;
int ValPinky = 0;
int maxPinky = 0;

  int muestra[26];
  int cont=0;

void setup() {
  pinMode(PulgarPin, INPUT);
  Serial.begin(2000000);
}

void loop() {
  ValPulgar = digitalRead(PulgarPin);
  ValIndice = digitalRead(IndicePin);
  ValMedio = digitalRead(MedioPin);
  ValAnular = digitalRead(AnularPin);
  ValPinky = digitalRead(PinkyPin);

  cont=0; maxPulgar=0; maxIndice=0; maxMedio=0; maxAnular=0;
maxPinky=0;
  memset(muestra, 0, sizeof(muestra)); //Clear the ARRAY
```

```
&& (ValAnular == 0) && (ValPinky == 0)){
    delay(250);
    ValPulgar = digitalRead(PulgarPin);
    ValIndice = digitalRead(IndicePin);
    ValMedio = digitalRead(MedioPin);
    ValAnular = digitalRead(AnularPin);
    ValPinky = digitalRead(PinkyPin);
  }

// PULGAR
  if (ValPulgar == 1){
   while (ValPulgar == 1){
    muestra[cont]=analogRead(A1);
    if(muestra[cont]>maxPulgar){
      maxPulgar=muestra[cont];
    }
    cont++;
    delay(75);
    ValPulgar = digitalRead(PulgarPin);
   }
    Serial.print("MaxPulgar: ");
    Serial.println(maxPulgar);
 memset(muestra, 0, sizeof(muestra)); //Clear the ARRAY
 cont=0; maxPulgar=0; maxIndice=0; maxMedio=0; maxAnular=0;
maxPinky=0;
  }
// INDICE
  if (ValIndice == 1){
   while (ValIndice == 1){
    muestra[cont]=analogRead(A1);
    if(muestra[cont]>maxIndice){
      maxIndice=muestra[cont];
    }
    cont++;
    delay(75);
    ValIndice = digitalRead(IndicePin);
   }
    Serial.print("MaxIndice: ");
    Serial.println(maxIndice);
 memset(muestra, 0, sizeof(muestra)); //Clear the ARRAY
```

```
 cont=0; maxPulgar=0; maxIndice=0; maxMedio=0; maxAnular=0;
maxPinky=0;
  }
// MEDIO
  if (ValMedio == 1){
   while (ValMedio == 1){
      muestra[cont]=analogRead(A1);
      if(muestra[cont]>maxMedio){
        maxMedio=muestra[cont];
      }
      cont++;
      delay(75);
      ValMedio = digitalRead(MedioPin);
     }
      Serial.print("MaxMedio: ");
      Serial.println(maxMedio);
   memset(muestra, 0, sizeof(muestra)); //Clear the ARRAY
   cont=0; maxPulgar=0; maxIndice=0; maxMedio=0; maxAnular=0;
maxPinky=0;
    }
  // ANULAR
    if (ValAnular == 1){
     while (ValAnular == 1){
      muestra[cont]=analogRead(A1);
    if(muestra[cont]>maxAnular){
      maxAnular=muestra[cont];
    }
    cont++;
    delay(75);
    ValAnular = digitalRead(AnularPin);
   }
    Serial.print("MaxAnular: ");
    Serial.println(maxAnular);
 memset(muestra, 0, sizeof(muestra)); //Clear the ARRAY
  cont=0; maxPulgar=0; maxIndice=0; maxMedio=0; maxAnular=0;
maxPinky=0;
  }
// PINKY
  if (ValPinky == 1){
   while (ValPinky == 1){
```

```
    muestra[cont]=analogRead(A1);
    if(muestra[cont]>maxPinky){
      maxPinky=muestra[cont];
    }
    cont++;
    delay(75);
    ValPinky = digitalRead(PinkyPin);
   }
    Serial.print("MaxPinky: ");
    Serial.println(maxPinky);
    memset(muestra, 0, sizeof(muestra)); //Clear the ARRAY
     cont=0; maxPulgar=0; maxIndice=0; maxMedio=0; maxAnular=0;
naxPinky=0;
  }
}
```

Anexo B. Tecnologías

Durante el transcurso del proyecto se han diseñado y fabricado varios circuitos integrados y se han realizado modificaciones en el modelo 3D de la prótesis. Aunque no son objetivos concretos del trabajo, la realización de estas tareas ha sido clave en el desarrollo del mismo.

B.1. Modificaciones en el modelo 3D

Debido a la elección del sensor Flex como solución para la parametrización de la posición de los dedos, se han realizado modificaciones en los ficheros 3D originales con el objetivo de albergarlos en el interior de los dedos. Se han realizado unos pequeños canales rectangulares de 6 x 1.5 mm y un agujero adicional encima de la ranura de 2 mm de diámetro para el paso posibilitar el paso de del muelle encargado de realizar el movimiento de relajación. Este

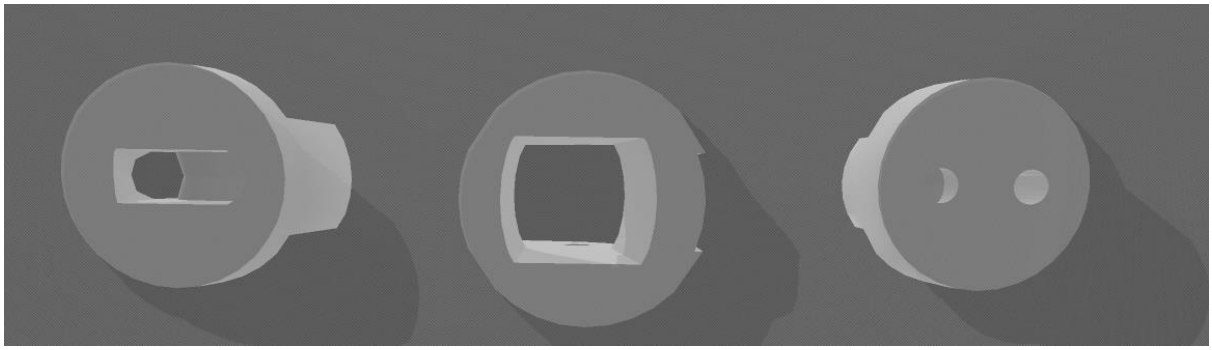


Figura.B.1 *Detalle falanges modelo 3D original*

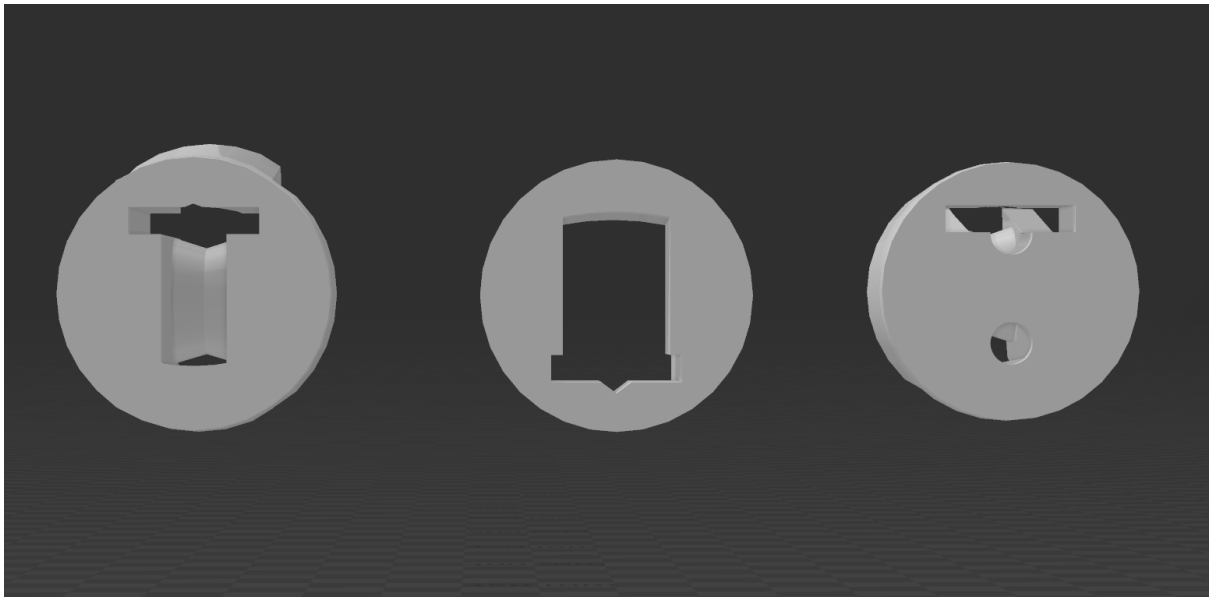


Figura.B.2 *Detalle falanges modelo 3D modificado*

Estas modificaciones se han realizado en los dedos pulgar índice y medio ya que son los dedos más relevantes para el control de posición y además son los que mejor permiten la implantación del sensor FLEX.

B.2. Impresión 3D

Para el presente proyecto se contaba con una prótesis perteneciente a un proyecto anterior, sin embargo, para el objetivo del control de este trabajo las holguras la calidad de la impresión y las holguras en el montaje de los dedos imposibilitan el buen funcionamiento de nuestro objetivo específico. Por ello, se ha decidido realizar de nuevo la impresión de los dedos, palma y muñeca para su montaje sobre el antebrazo anterior.

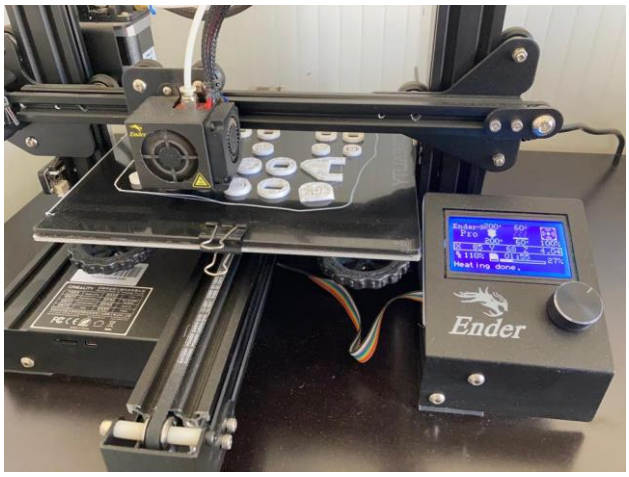


Figura.B.3 *Inicio de la impresión de las falanges con la impresora Ender 3 pro*

Figura.B.4 *Final de la impresión de las falanges con la impresora Ender 3 pro*

Para el montaje de la mano se han seguido los pasos mostrados en la página web de InMoov.

B.3. Características circuitos impresos

Para el control de la prótesis y prototipos como para la toma de señales se han diseñado, fabricado y utilizado un total de 3 circuitos impresos. El diseño de las placas se ha llevado a cabo mediante el software EAGLE. A continuación, se muestran las características de las placas:

- **Shield control joystick 5 servos:**

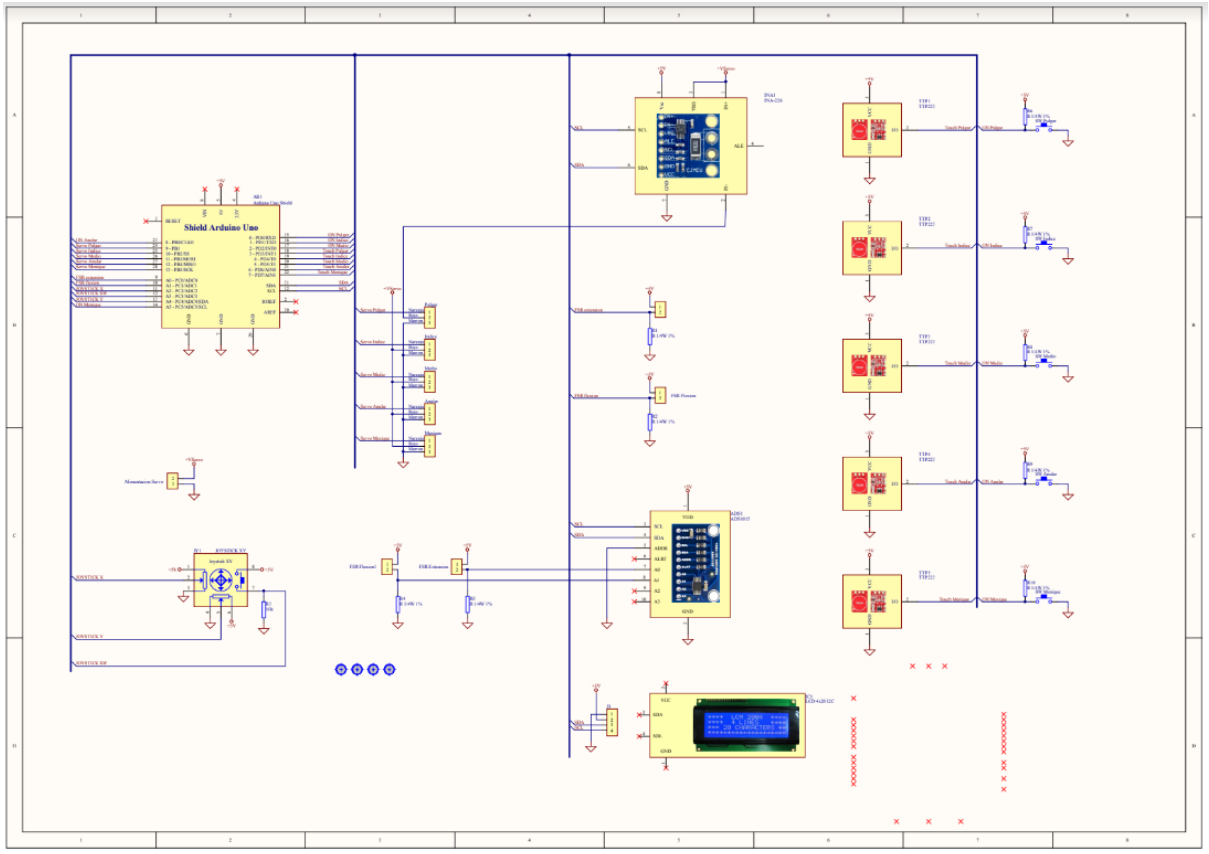


Figura.B.5 *Esquema circuito y componentes Shield 5 servos*

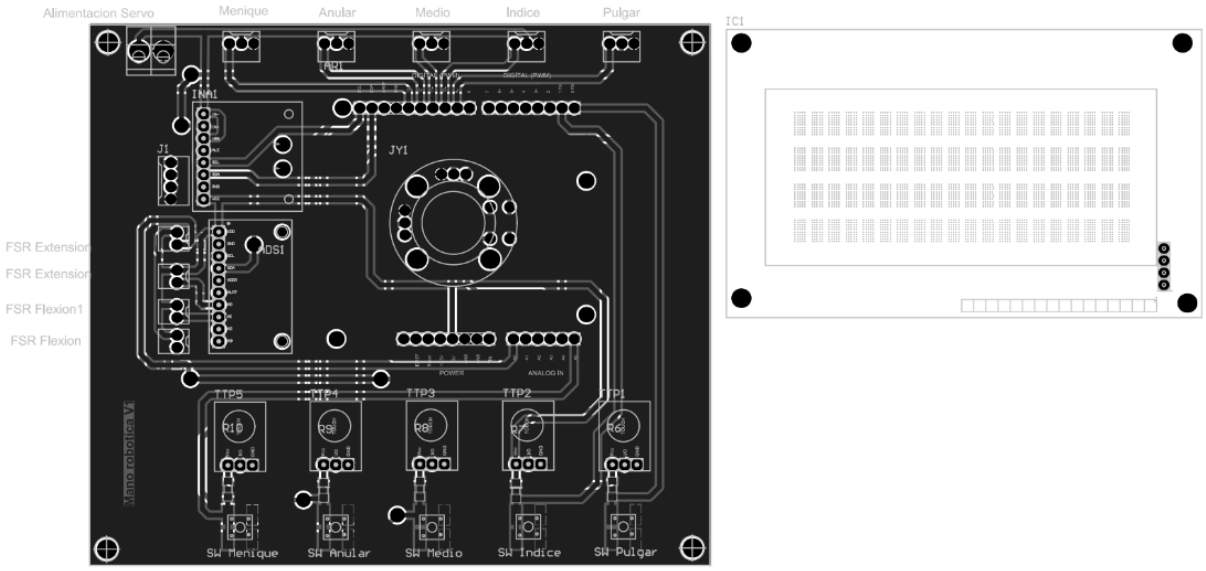


Figura.B.6 *Plantilla circuito Shield 5 servos*

Comment	Description	Designator	Footprint	LibRef	Quantity
CON POSTE 2	Conector poste mach	[NoValue], FSR Exten	Conector MOLEX 2	CON POSTE 2	4
ADS1015	Conversor Analogico-	ADS1	ADS1015 + Zocalo	ADS1015	1
CON POSTE 2	Conector poste mach	Alimentacion Servo	Clema PCB - 5.08 2 p	CON POSTE 2	1
CON POSTE 3	Conector poste mach	Anular, Indice, Medio,	Conector MOLEX 3	CON POSTE 3	5
Arduino Uno Shield		AR1	Arduino UNO Shield	Arduino Uno Shield	1
LCD 4x20 I2C	LCD 4x20 con interfa	IC1	LCD 4x20 I2C	LCD 4x20 I2C	1
INA-226	36V, 16-bit, ultra-prec	INA1	INA-226 + Zocalo	INA-226	1
CON POSTE 4	Conector poste mach	J1	Conector MOLEX 4	CON POSTE 4	1
JOYSTICK XY	JOYSTICK XY	JY1	Jostick dual XY	JOYSTICK XY	1
R 1/4W 1%	Resistencia 1% 1/4W	R1, R2, R4, R5, R6, R	1206 R 3D	R 1/4W 1%	9
10k	Resistencia 1% 1/4W	R3	1206 R 3D	R 1/4W 1%	1
SW SPST MOMENTAN	Pulsador para circuito	SW Anular, SW Indice	SW SPST MOMENTAN	SW SPST MOMENTAN	5
Separador 10 mm +	Separador circuito im	T1, T2, T3, T4	Tornillo + separador	Separador 10 mm +	4
TTP223	TTP223 Button Modul	TTP1, TTP2, TTP3, T	TTP223 + Zocalo	TTP223	5

Tabla.B.1 *Componentes Shield 5 servos*

- **Shield control joystick 5 motores lineales:**

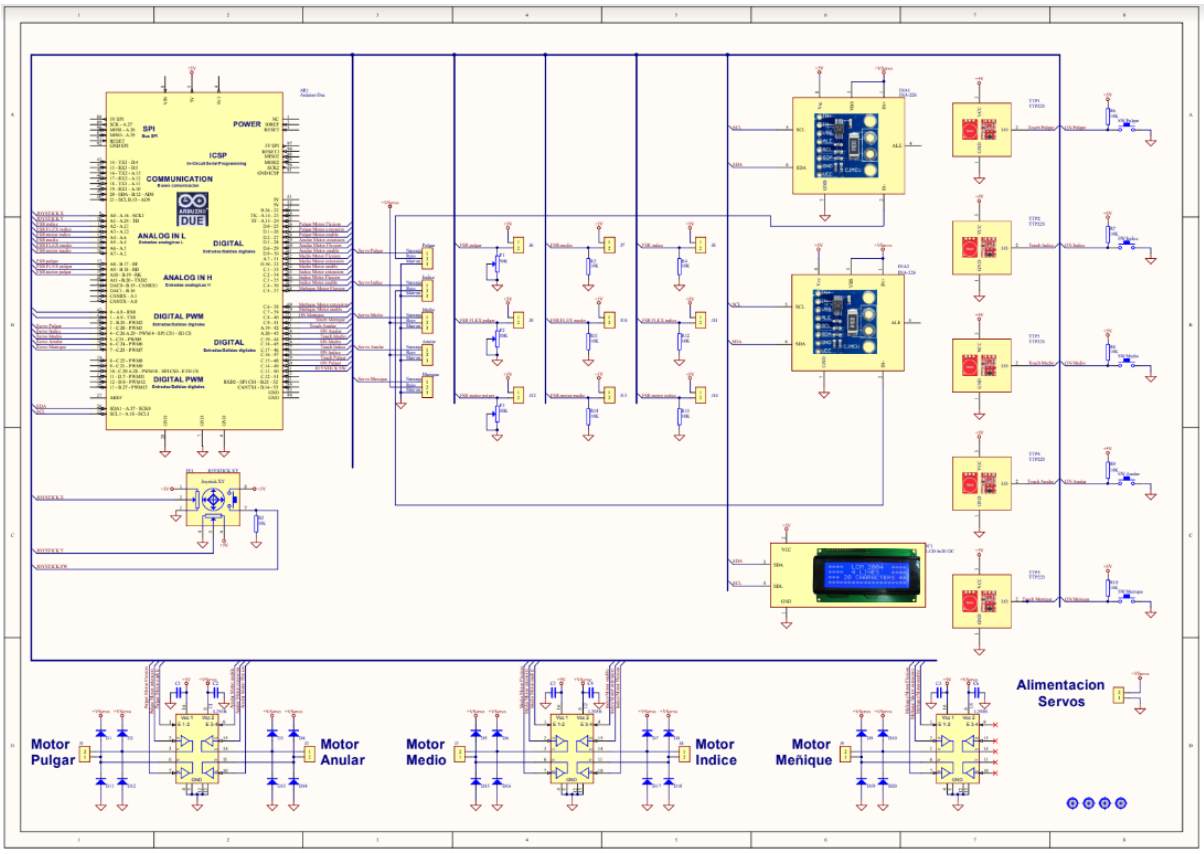


Figura.B.7 *Esquema circuito y componentes Shield 5 motores lineales*

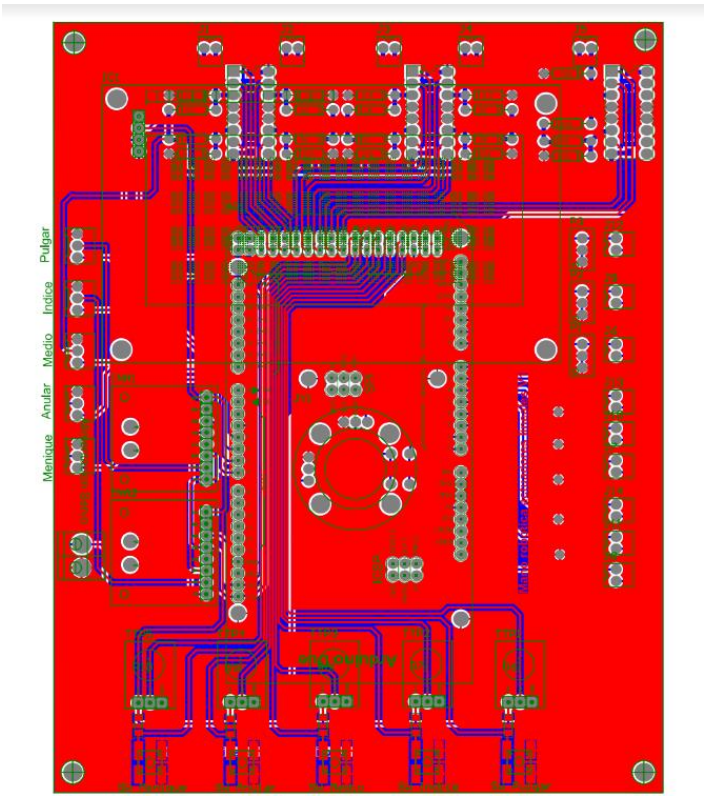


Figura.B.8 *Plantilla circuito Shield 5 motores lineales*

Comment	Description	Designator	Footprint	LibRef	Quantity
CON POSTE 2	Conector poste mach	Alimentacion Servo	Clema PCB - 5.08 2 p	CON POSTE 2	1
CON POSTE 3	Conector poste mach	Anular, Indice, Medio,	Conector MOLEX 3	CON POSTE 3	5
Arduino Due		AR1	Arduino Due	Arduino Due	1
C 100nF/50V SMD	Cond cerámico SMD 1	C1, C2, C3, C4, C5, C	1206 C 3D	C 100nF/50V SMD	6
D 1N4007	Diodo 1N4007	D1, D2, D3, D4, D5, D	Diodo 0.4	D 1N4007	20
LCD 4x20 I2C	LCD 4x20 con interfa	IC1	LCD 4x20 I2C	LCD 4x20 I2C	1
INA-226	36V, 16-bit, ultra-prec	INA1, INA2	INA-226 + Zocalo	INA-226	2
CON POSTE 2	Conector poste mach	J1, J2, J3, J4, J5, J6, J	Conector MOLEX 2	CON POSTE 2	14
JOYSTICK XY	JOYSTICK XY	JY1	Jostick dual XY	JOYSTICK XY	1
50K	Potenciometro multiv	P1, P2, P3	POT Multivuelta verti	POT MV	3
10K	Resistencia 1% 1/4W	R2, R3, R4, R6, R7, R	1206 R 3D	R 1/4W 1%	12
SW SPST MOMENTAN	Pulsador para circuito	SW Anular, SW Indic	SW SPST MOMENTAN	SW SPST MOMENTAN	5
Separador 10 mm +	Separador circuito im	T1, T2, T3, T4	Tornillo + separador	Separador 10 mm +	4
TTP223	TTP223 Button Modu	TTP1, TTP2, TTP3, T	TTP223 + Zocalo	TTP223	5
L293B	L293x Quadruple Ha	U1, U2, U3	DIP 16 + Zocalo 3D	L293B	3

Tabla.B.2 *Componentes Shield 5 motores lineales*

- **Shield Temperatura y Distancia**

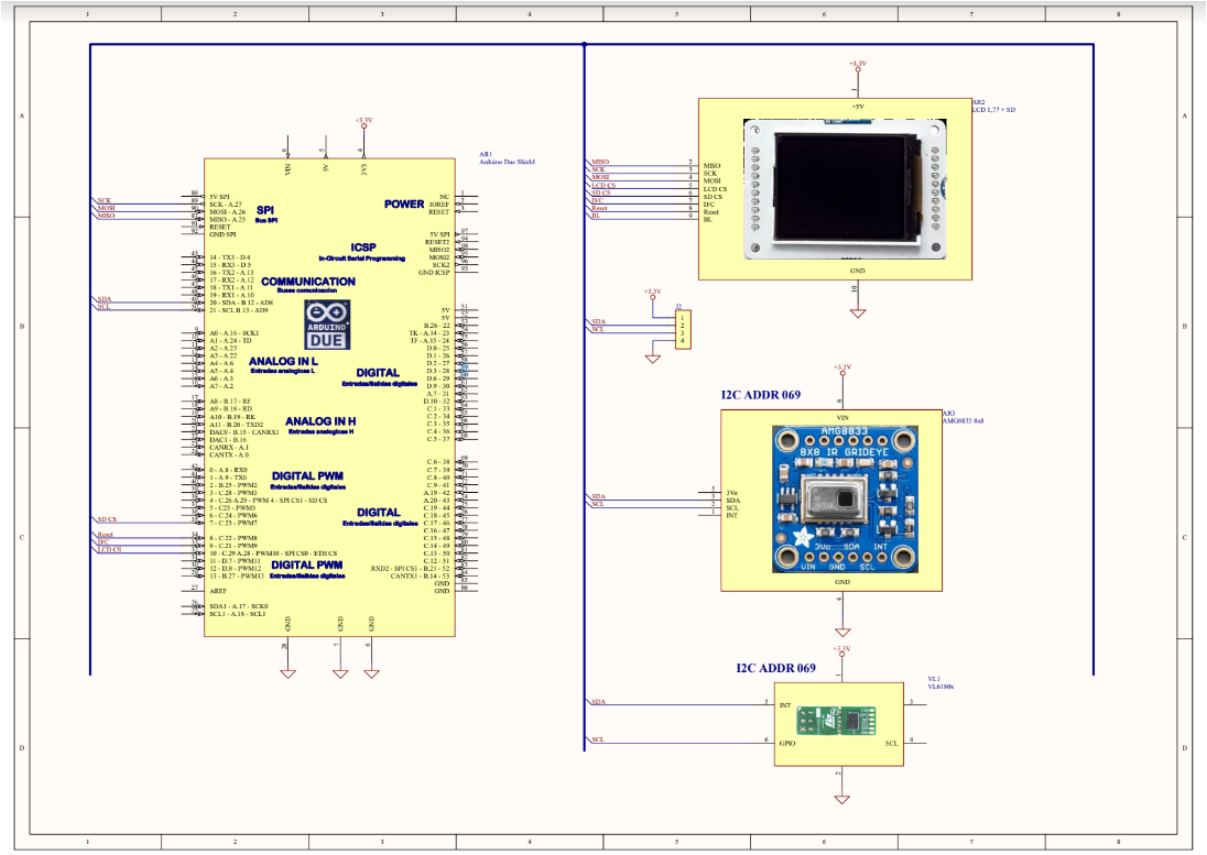


Figura.B.9 *Esquema circuito y componentes Shield Temperatura y Distancia*