

Informe Final

Construcción Mano Biónica en 3D

Integrantes: Diego Campanini
Eduardo Hormazábal
Profesor: Helmuth Thiemer
Auxiliar: Carlos Navarro
Ayudante: Rodrigo Maureira
Fecha: 14 de diciembre de 2015

Índice general

1. Introducción	5
2. Adquisición y Circuito de Adaptación de la señal	7
2.1. Adquisición de la señal	7
2.2. Circuito de adaptación de la señal	8
2.2.1. Diseño del circuito	8
2.2.2. Simulaciones circuito de adaptación de la señal	13
2.2.3. Construcción del circuito	15
3. Implementación y Entrenamiento Máquina de Aprendizaje	21
3.1. Conversión A/D	21
3.2. Preprocesamiento de la señal	22
3.2.1. Extracción de características	22
3.3. Resultados de la Extracción de Características	28
3.4. Antecedentes Support Vector Machine	33
3.4.1. Patrones Linealmente Separables	33
3.4.2. Patrones No Linealmente Separables	35
3.4.3. Aumento de dimensionalidad utilizando un kernel	36
3.5. Implementación de SVM	38
3.5.1. Entrenamiento off line	38
3.6. SVM con datos de Arduino	40
4. Construcción Mano Biómica e Integración de los Módulos	43
4.1. Impresión en 3D de la Mano Biómica	43
4.1.1. Modelo de la Mano	43
4.1.2. Modelo del Brazo y la Muñeca	44
4.2. Ensamblado de la Mano Biómica	46
4.2.1. Modelo de la Mano	46
4.2.2. Modelo del Brazo y la Muñeca	47
4.2.3. Modelo completo	48
5. Evaluación económica	50
6. Conclusiones y Trabajo Futuro	51

7. Anexos

54

Índice de figuras

1.1. Diagrama de bloques del sistema	6
2.1. Ubicación de los electrodos	7
2.2. Circuito amplificador diferencial	9
2.3. Circuito filtro pasa banda Sallen-Key de segundo orden	10
2.4. Circuito filtro rechaza banda	11
2.5. Circuito de amplificación de la señal de salida	12
2.6. Circuito sumador inversor para el offset.	12
2.7. Señales de entrada al circuito	13
2.8. Voltajes de salida de los 4 bloques principales	14
2.9. Diagrama de Bode de los tres bloques finales del circuito de adaptación de la señal de entrada	14
2.10. Salida del bloque de amplificación para una entrada sinusoidal de $V_{pp} = 10[mV]$ y $250[Hz]$	15
2.11. Salida del bloque de filtrado pasa banda para una entrada sinusoidal de $V_{pp} = 10[mV]$ y $250[Hz]$	16
2.12. Salida del bloque de filtrado pasa banda para una entrada sinusoidal de $V_{pp} = 10[mV]$ y $1[kHz]$	17
2.13. Salida del bloque rechaza banda para una entrada sinusoidal de $V_{pp} = 10[mV]$ y $50[Hz]$	18
2.14. Salida del bloque de amplificación final para una entrada sinusoidal de $V_{pp} = 10[mV]$ y $250[Hz]$	19
2.15. Salida del bloque del offset agregado para una medida por los electrodos.	20
3.1. Representación del skewness de tres distribuciones	25
3.2. Kurtosis baja y alta	26
3.3. Entradas de señales de $2 [Hz]$ al ADC	29
3.4. Entradas de Ruido al ADC	30
3.5. Espectros de Fourier de diferentes señales de entrada al Arduino	31
3.6. Espectros de Fourier calculados con FFT de Matlab para diferentes señales de entrada al Arduino	32
3.7. Separación de dos clases (círculos y circunferencias) mediante hiperplanos	34
3.8. Región de decisión caso no separable	35
3.9. Efecto del aumento de dimensionalidad para lograr trabajar en un espacio donde las clases sean fácilmente separables	37
3.10. Movimientos a clasificar. Superior izquierda: mano quieta. Superior derecha: apretar puño. Inferior izquierda: apretar cilindro. Inferior derecha: sostener peso.	38
3.11. Muestra de movimientos de la base de datos utilizada en [2]	39

3.12. Muestra de movimientos obtenidos con Arduino	41
4.1. Modelo de la mano que se imprimirá en 3D	43
4.2. Pernos que unen los dedos a las piezas de la palma	44
4.3. Ensamblaje de las piezas de los dedos	44
4.4. Modelo del brazo y muñeca	45
4.5. Ubicación de los servo motores dentro del brazo	45
4.6. Impresión de parte de la palma de la mano	46
4.7. Palma y muñeca de la mano ensamblados	47
4.8. Movimiento de un dedo accionado por un servomotor	47
4.9. Brazo y muñeca ensamblados	48
4.10. Sistema completo ensamblado	49

Capítulo 1

Introducción

El proyecto consiste en la construcción de una mano biónica accionada por señales mio-eléctricas provenientes del brazo, las cuales se deben acondicionar para una adecuada lectura de los datos, luego se muestreará para posteriormente construir un clasificador de movimientos mediante una técnica de inteligencia computacional llamada SVM [1], en donde finalmente se controlarán servomotores responsables de accionar los distintos movimientos de la mano.

El sensor utilizado para adquirir la señal corresponderá a un grupo de tres electrodos, uno de los cuales será la referencia a tierra con respecto a los otros dos y al circuito de adquisición de la señal. Es necesario que los electrodos estén relativamente cercanos para que el ruido se logre cancelar con facilidad mediante el amplificador diferencial.

Las señales biológicas con las que se trabajará serán del orden de 1 a 10 mV y de una frecuencia de entre 10 a 500 Hz. Estas serán adquiridas desde dos músculos ubicados en el antebrazo, los cuales son el Flexor Carpi Ulnaris y el Extensor Carpi Radialis.

Para acondicionar la señal se utiliza un circuito en base a Op-amps, el cual aparece reiteradas veces en la literatura. El primer bloque amplifica diferencialmente la señal, luego se aplica un filtro pasa banda de segundo orden tipo Sallen-Key para eliminar parte del ruido circundante, el cual proviene desde el cambio de potencial en células vecinas a los músculos que se desean medir su potencial, el tercer bloque será un filtro rechaza banda a 50 Hz para eliminar los efectos de la red , enseguida se efectuará una amplificación, y finalmente se agregará un offset de 1.7 V, ya que, el dispositivo de conversión análogo digital trabaja sólo con valores de la señal positivos.

Luego se procede a digitalizar la señal con el fin de obtener un muestreo de ésta y que se puedan realizar operaciones mediante un dispositivo que procese información digital. Ambas tareas, es decir, la conversión análogo a digital y el procesamiento de la señal, se realizan con un Arduino Due.

Una vez muestreada la señal, se le extraen diversas características tales como media, varianza, skewness, energía, etc. las cuales serán la entrada al clasificador SVM, el que se encargará de decidir a qué tipo de movimiento corresponde la señal captada. Esta decisión desencadena el cambio en los estados de los actuadores, que son los encargados de efectuar el movimiento satisfactorio de la mano biónica.

El proyecto se puede modularizar en 4 bloques, que son los que se explicarán en los capítulos posteriores. En la Figura 1.1 se muestra el orden de los bloques, esbozando una idea general del prototipo.

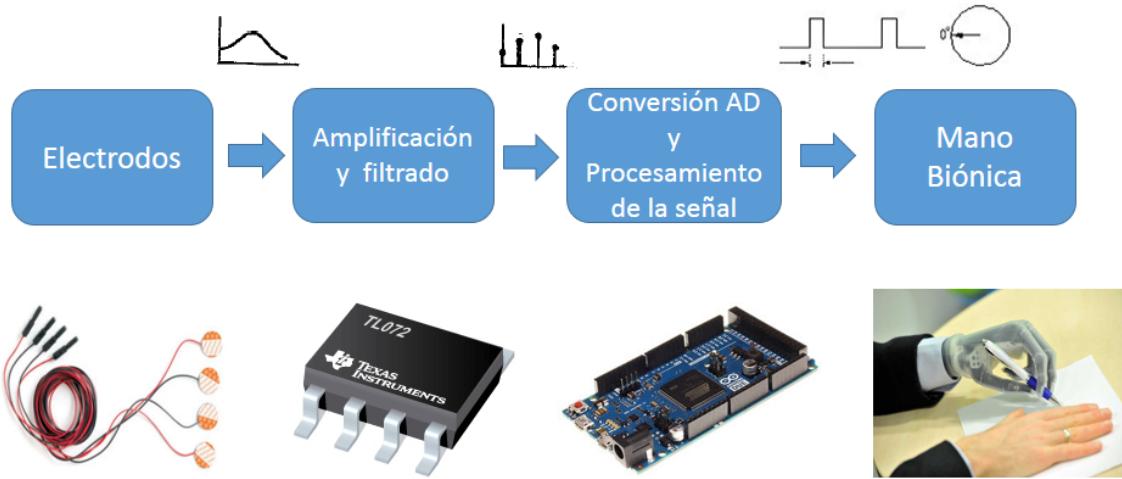


Figura 1.1: Diagrama de bloques del sistema

Capítulo 2

Adquisición y Circuito de Adaptación de la señal

2.1. Adquisición de la señal

Un aspecto fundamental para el correcto funcionamiento del sistema a construir, es la apropiada adquisición de la señal mio-eléctrica, desde la cual se extraerán características útiles para la futura clasificación, con la que se controlará el sistema mecánico de la mano. Para ello se utiliza un conjunto de 3 electrodos RED DOT Multiuso resistente a fluidos [7] para tener acceso a una señal de 1 canal.

La adquisición de la señal mio-eléctrica se genera al ubicar los tres electrodos en las posiciones que indica la Figura 2.1. Como se observa en el bosquejo los 3 electrodos se ubican relativamente cerca entre sí, lo que facilita la substracción del ruido al momento de diferenciar las señales. Cabe destacar que la señal extraída es obtenida con la diferencia de los voltajes de dos electrodos. Sin embargo es posible extender esta configuración para obtener dos canales, pues el electrodo que se emplea como tierra se puede utilizar como referencia mientras los dos restantes inducirían, cada uno, un voltaje con respecto a éste.

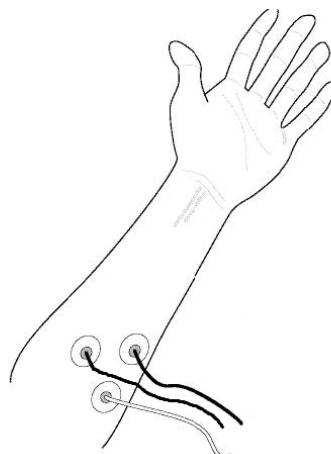


Figura 2.1: Ubicación de los electrodos

2.2. Circuito de adaptación de la señal

2.2.1. Diseño del circuito

En esta sección se explica el diseño teórico del circuito de adaptación de la señal proveniente de los electrodos, en donde se explican las etapas de amplificación y filtrados tanto pasa-banda como rechaza-banda.

Circuito Amplificador de entrada

Las señales de entrada al circuito serán del orden de 1 a 10 mV pico-pico, es por esto que resulta necesario una primera etapa de amplificación de las señales. El circuito propuesto actúa como un amplificador diferencial de las señales de entrada y además atenúa el ruido común a ambas, proporcionando una alta tasa de rechazo al modo común (CMRR por sus siglas en inglés).

El circuito amplificador se basa en la estructura interna del amplificador INA 121, el cuál tiene como una de sus aplicaciones principales la amplificación de señales fisiológica. Los Op-amp utilizados en todos los bloques serán los de la familia TL072.

Para implementar la amplificación se utilizarán tres Op-amp TL072, una resistencia $R1 = 10 K\Omega$, dos resistencias $R2 = 120 K\Omega$, una resistencia $R4 = 100 K\Omega$ y un potenciómetro de $100\text{--}100 K\Omega$ para controlar la ganancia. La cuál está dada por la siguiente función de transferencia:

$$\frac{V_{out}}{V_{in}} = \left(1 + \frac{2 \cdot R2}{R1}\right) \cdot \left(\frac{R4}{R3}\right) \quad (2.1)$$

La topología final del circuito amplificador se aprecia en la Figura 2.2, en la cual las señales de entrada son V_{in1} y V_{in2} (provenientes desde 2 electrodos distintos). Mencionar que la tierra se obtiene con un tercer electrodo que se puede ubicar en el codo, y además se utilizan resistencias conectadas entre las entradas y tierra para compensar la baja impedancia de las mediciones.

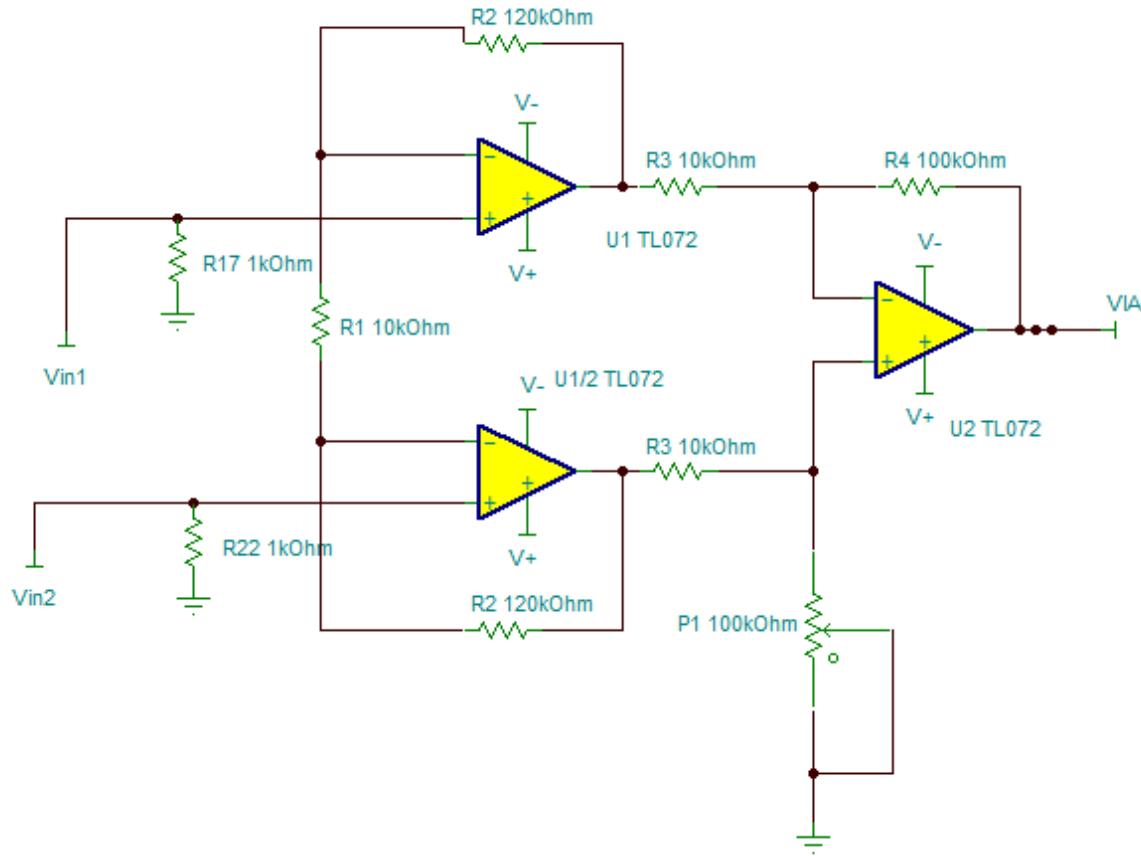


Figura 2.2: Circuito amplificador diferencial

Filtro pasa banda

El filtro pasa banda que se utilizará es del tipo Sallen-Key de segundo orden, la banda de frecuencia será entre 0,04 Hz y 500 Hz, de esta forma se abarca el rango de las señales fisiológicas. La arquitectura del circuito se presenta en la figura 2.3, los valores de los condensadores se fijaron en $C_1 = 1 \mu F$ y $C_2 = 0,1\mu F$, luego considerando una ganancia deseada de 2 para los dos sub-bloques que forman este filtro, se procede a determinar los valores de los otros elementos del circuito por medio de las siguientes fórmulas:

$$f_l = 0,04 = \frac{1}{2\pi R_5 \cdot C_1} \quad (2.2)$$

$$f_h = 500 = \frac{1}{2\pi R_8 \cdot C_2} \quad (2.3)$$

$$G_l = 2 = 1 + \frac{R_6}{R_5} \quad (2.4)$$

$$G_h = 2 = 1 + \frac{R_9}{R_{10}} \quad (2.5)$$

Finalmente las resistencias a utilizar serán $R5 = 3,3 M\Omega$, $R6 = R7 = 10 K\Omega$, $R8 = 3,6 K\Omega$, $R9 = R10 = 10 K\Omega$. Mencionar que la entrada a este bloque corresponde a la salida de la etapa de amplificación anterior.

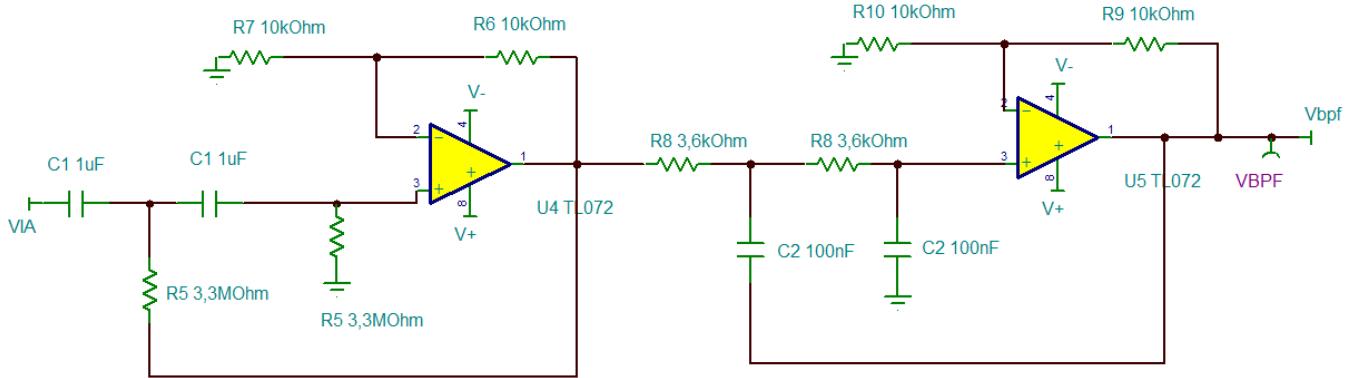


Figura 2.3: Circuito filtro pasa banda Sallen-Key de segundo orden

Filtro rechaza banda

Se implementará un filtro rechaza banda a 50 Hz, para evitar los efectos de la red eléctrica en la obtención de las señales. El circuito se aprecia en la Figura 2.4, los valores de los componentes utilizados son $C3=270pF$, $C4=540pF$, $R11 = R12 = 6 M\Omega$ y un potenciómetro de $100 K\Omega$. Los valores anteriores señalados se calculan en base a las siguientes ecuaciones:

$$C3 = C4 = \frac{C5}{2} \quad (2.6)$$

$$R11 = 2 \cdot R12 \quad (2.7)$$

$$f_n = \frac{1}{2\pi C3 \cdot R11} \quad (2.8)$$

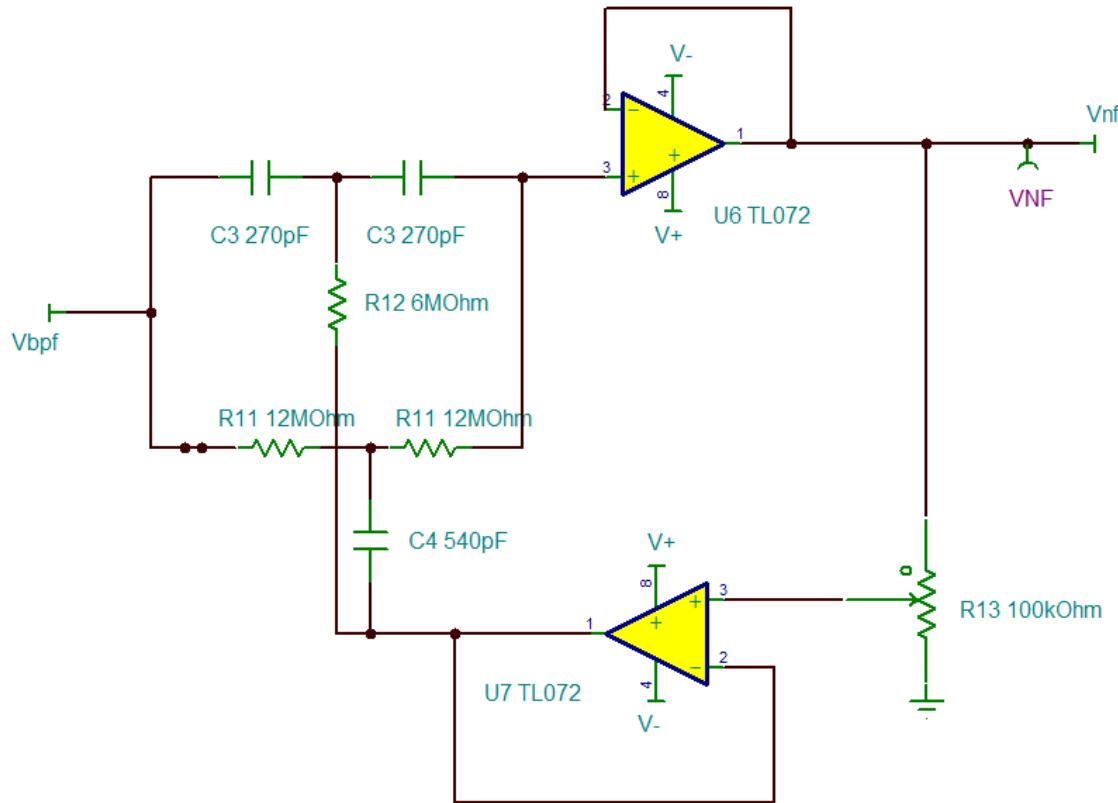


Figura 2.4: Circuito filtro rechaza banda

Amplificación final

El penúltimo bloque del circuito contempla una amplificación para obtener una amplitud apropiada de la señal que ingresará a la etapa de conversión análogo a digital. El circuito que realiza esta amplificación se muestra en la Figura 2.5, para esto se utilizaron dos resistencias de $1 K\Omega$ y un potenciómetro de $100 K\Omega$ para ajustar la ganancia, luego la salida de este bloque V_{outFin} es la entrada de la etapa de conversión análogo a digital.

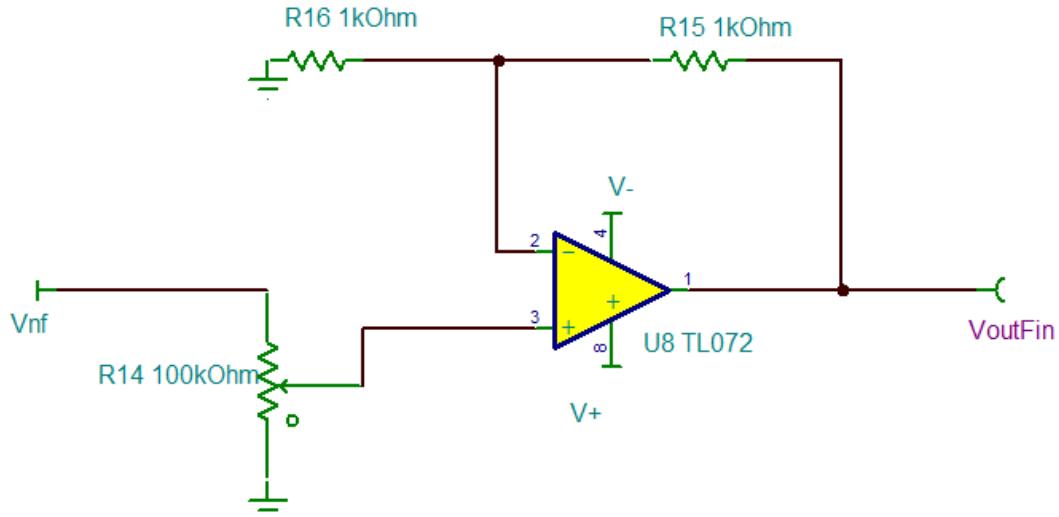


Figura 2.5: Circuito de amplificación de la señal de salida

Offset agregado

El último bloque del circuito corresponde a la suma de una componente continua al voltaje amplificado. Esto se debe a que el conversor análogo-digital puede procesar de forma correcta voltajes entre 0 y 5 [V], mientras que la salida del amplificador de la sección 2.2.1 está centrada en 0[V].

Para ello se utiliza un sumador inversor (ver Figura 2.6) en donde una de las entradas a sumar es la salida del amplificador final, y la otra es un voltaje continuo con un valor tal que no se obtengan voltajes negativos con las señales reales adquiridas con los electrodos.

También se incluye un seguidor inversor, de manera que la impedancia de salida del circuito se acople correctamente al conversor análogo digital del Arduino, correspondiente al capítulo siguiente.

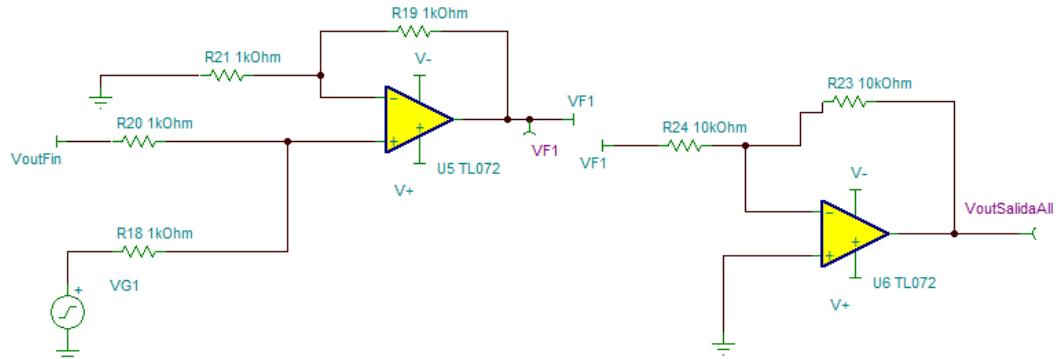


Figura 2.6: Circuito sumador inversor para el offset.

2.2.2. Simulaciones circuito de adaptación de la señal

Simulación señal de entrada

Para realizar las simulaciones del circuito se utilizó el *software* TINA. La alimentación de los Op-amp fue de $\pm 9V$ continuos y la señal de entrada se simuló mediante dos fuentes sinusoidales, una representando la señal fisiológica con frecuencia de 250 Hz y amplitud 10 mV_{pp} (Figura 2.7) y la otra fuente simuló el ruido de la red mediante una sinusode de 2 mV_{pp} a 50 Hz.

Las entradas utilizadas no se asemejan a las que se obtendrán realmente con los electrodos, ya que, estas serán no estacionarias ni periódicas, además tendrán mucho más ruido que el simulado, sin embargo, se consideran las entradas antes mencionadas puesto que permiten observar el comportamiento de los bloques del circuito, fundamentalmente en lo que respecta a la amplificación de la señal y su el diagrama de Bode de los bloques.

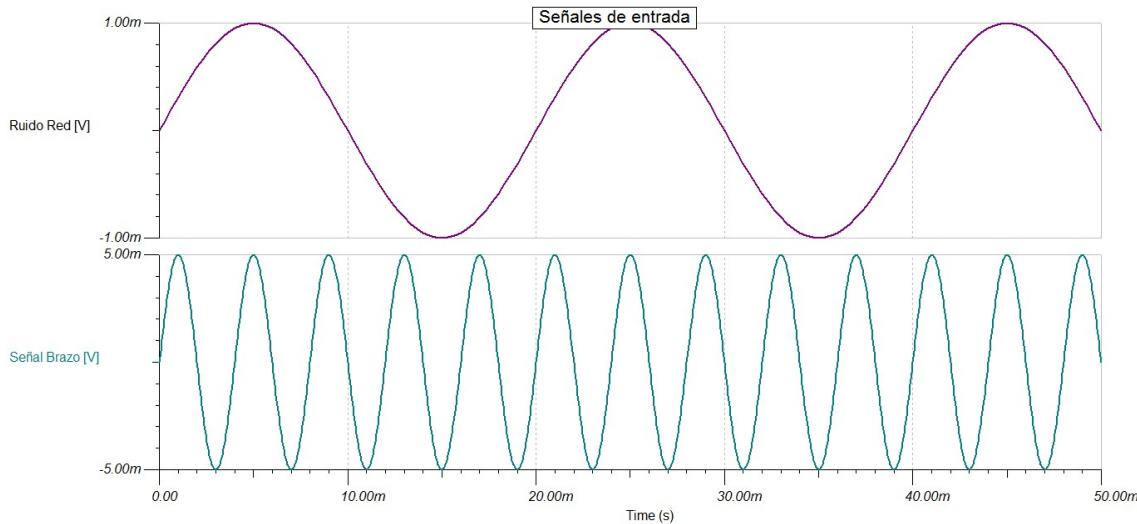


Figura 2.7: Señales de entrada al circuito

Simulación bloques circuito

Los resultados obtenidos en la simulación (Figura 2.8) resultaron concordantes con los cálculos teóricos realizados en base a los cuales se escogieron los parámetros. La respuesta del primer bloque (V_{outIA}) muestra que se obtiene una amplitud de salida de aproximadamente 1.25 V (Figura 2.8 curva azul), lo cual implica una amplificación de $1.25 \text{ V}/5 \cdot 10^{-3} \text{ mV} = 250$, que fue la consideración de diseño impuesta.

Posteriormente la señal de salida del bloque amplificador ingresa al filtro pasa banda con el cual se logran amplitudes de 5 V promedio (Figura 2.8 curva amarilla), por lo que está cumpliendo con el factor 4 de amplificación considerado en el diseño. Además desde el diagrama de Bode (Figura 2.9 curva amarilla) se aprecia que el filtro corresponde a uno pasa banda en el rango de frecuencias diseñado, es decir, entre 0.04 Hz y 500 Hz.

La salida del filtro pasa banda mantiene la amplitud de las señal (Figura 2.8 curva verde) con res-

pecto al bloque anterior. El efecto de este bloque se visualiza desde el diagrama de Bode, ya que, se tiene una disminución en la ganancia a los 50.77 Hz, lo cual resulta cercano a los 50 Hz esperados en base a los cálculos (Figura 2.9 curva verde). Finalmente se obtiene la señal de salida desde el último bloque (V_{outFin}), esta la cual mantiene la forma de la señal de entrada, pero ahora con una amplitud que alcanza aproximadamente los 6.2 V (Figura 2.8 curva roja), además se sigue apreciando en el diagrama de Bode los efectos de los filtros antes mencionados (Figura 2.9 curva Roja).

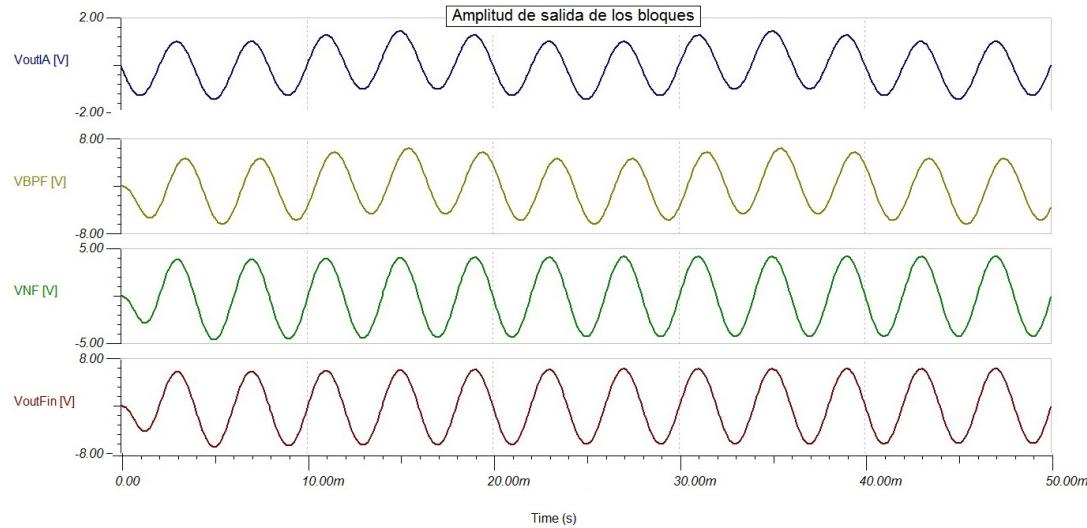


Figura 2.8: Voltajes de salida de los 4 bloques principales

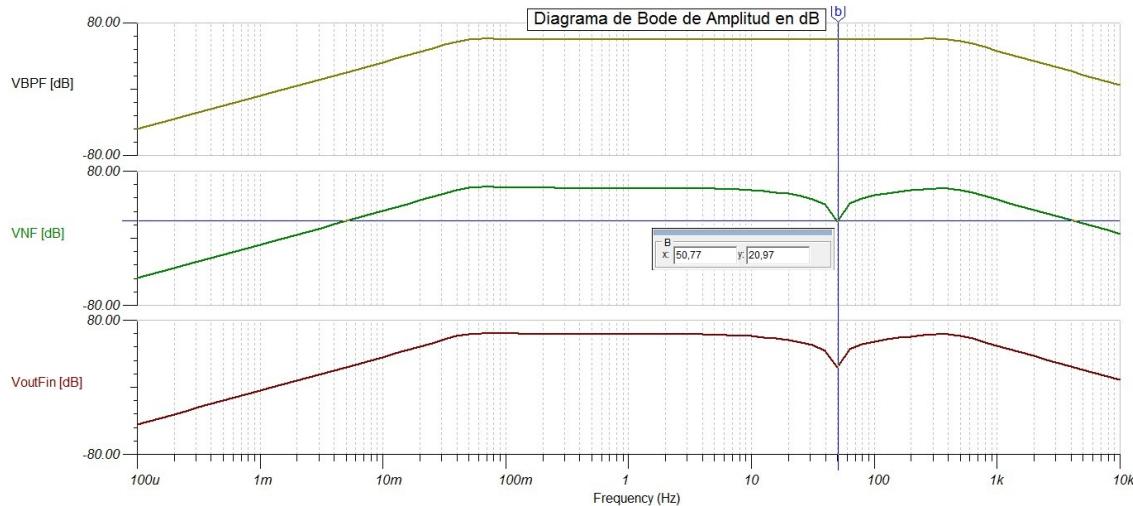


Figura 2.9: Diagrama de Bode de los tres bloques finales del circuito de adaptación de la señal de entrada

2.2.3. Construcción del circuito

El armado del circuito se realizó en una *protoboard* siguiendo el orden de los bloques de la sección 2.2.1. La comprobación de cada bloque se llevó a cabo con un generador de funciones para la señal de entrada, y un osciloscopio para visualizar el efecto del bloque respectivo a la entrada.

Círcuito Amplificador de entrada

Luego de armar el bloque de la sección 2.2.1 se comprobó el correcto funcionamiento de éste, dándole una entrada de referencia $V_{in} = 10\sin(2\pi 250t)[mV]$, es decir una señal sinusoidal de $250[Hz]$ de frecuencia y una amplitud máxima de $10[mV]$, es decir la amplitud máxima producida por una señal mioeléctrica [2].

En la Figura 2.10 se observa que la salida del bloque efectivamente amplifica la señal, con una ganancia de 280 que está dentro del rango esperado de la ganancia teórica, considerando las variaciones de los valores en las resistencias utilizadas.

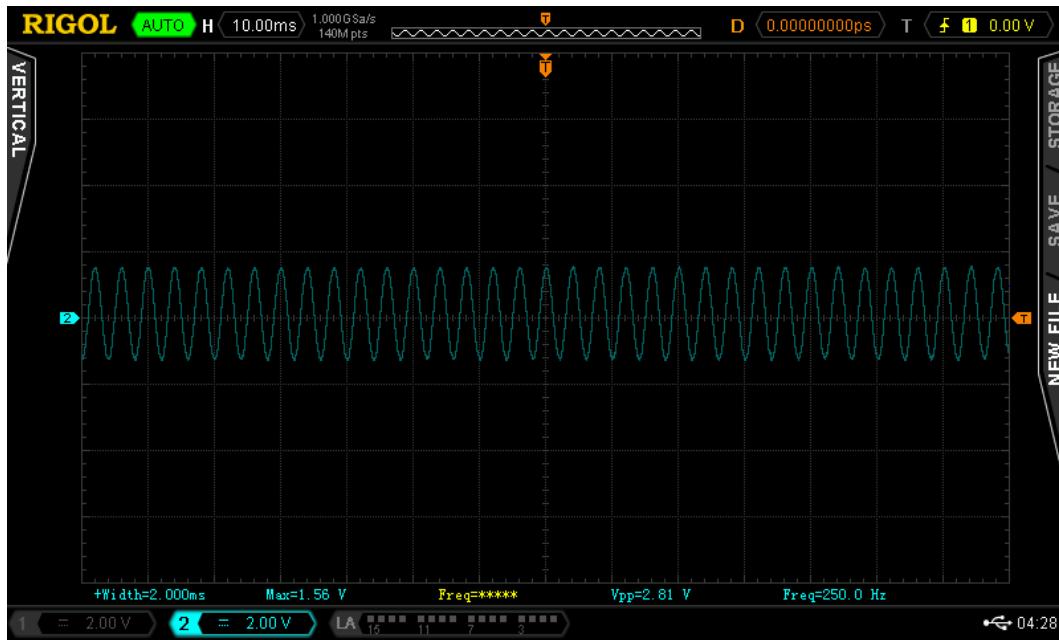


Figura 2.10: Salida del bloque de amplificación para una entrada sinusoidal de $V_{pp} = 10[mV]$ y $250[Hz]$.

Filtro pasa banda

La salida del circuito anterior es conectada a la entrada del bloque del filtro pasa banda, por lo que la entrada a este bloque es del orden de $[V]$ (gracias a la amplificación anterior). Con el objetivo de comprobar el filtrado de las frecuencias relativamente altas, es decir mayor a $500[Hz]$, se realizaron dos mediciones: una para $250[Hz]$ y otra para $1[kHz]$.

En las Figuras 2.11 y 2.12 se muestran las señales de entrada al bloque de filtrado pasa banda (señal inferior de color amarillo) y la respuesta del bloque (señal superior de color azul). En la Figura 2.11 se

observa una ganancia de aproximadamente 5, mientras que en la Figura 2.12 la amplificación resultante es menor a la unidad, que es lo esperado según el diseño realizado en la sección 2.2.1.

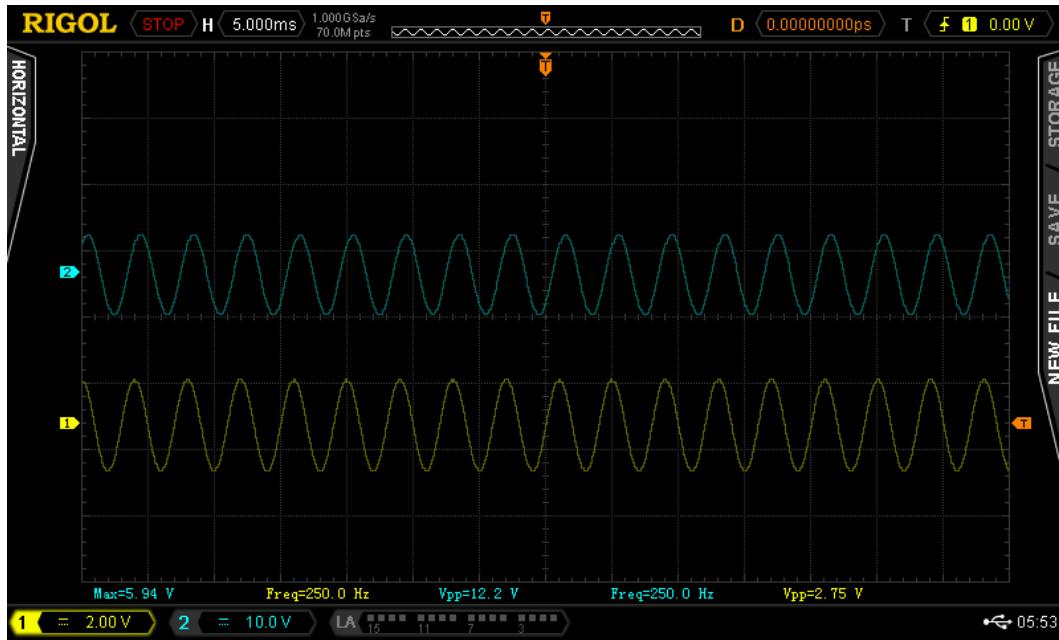


Figura 2.11: Salida del bloque de filtrado pasa banda para una entrada sinusoidal de $V_{pp} = 10[mV]$ y $250[Hz]$.

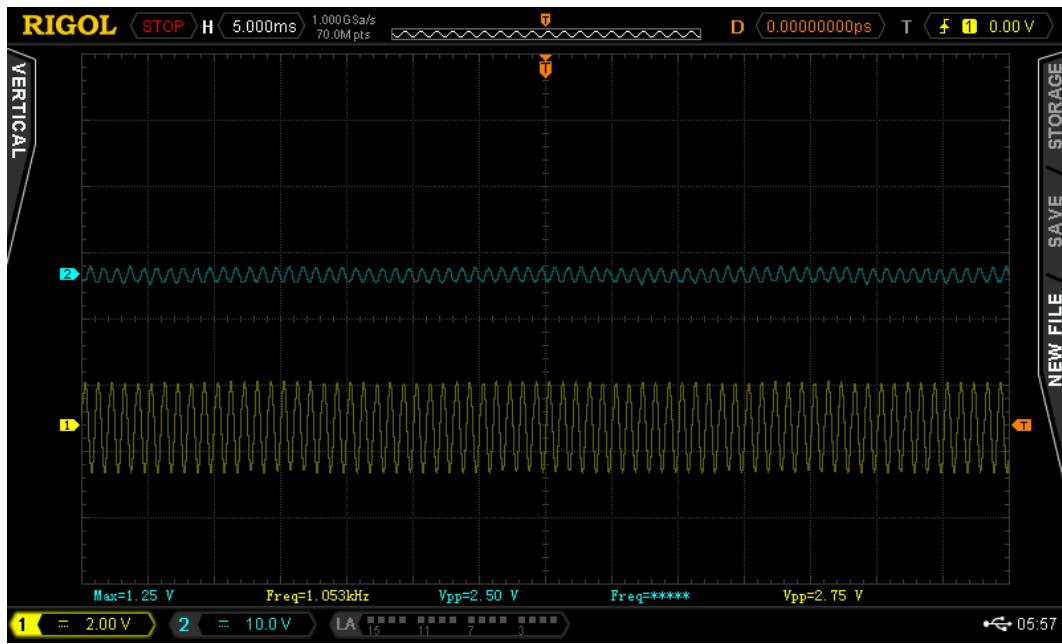


Figura 2.12: Salida del bloque de filtrado pasa banda para una entrada sinusoidal de $V_{pp} = 10[mV]$ y $1[kHz]$.

Filtro rechaza banda

Para la construcción de este bloque es esencial la precisión en los valores de las resistencias y capacidades. Es por esto que se utiliza un potenciómetro, encargado de ajustar el filtrado óptimo para los $50[Hz]$ de la red.

Una vez ajustado el potenciómetro se obtuvo el filtrado mostrado en la Figura 2.13 en donde la señal superior amarilla corresponde a la entrada al bloque, mientras que la señal inferior azul corresponde a la salida de éste. Como se observa en la figura prácticamente se eliminan los $50[Hz]$, mientras que para las demás frecuencias el bloque es visto como una ganancia unitaria.

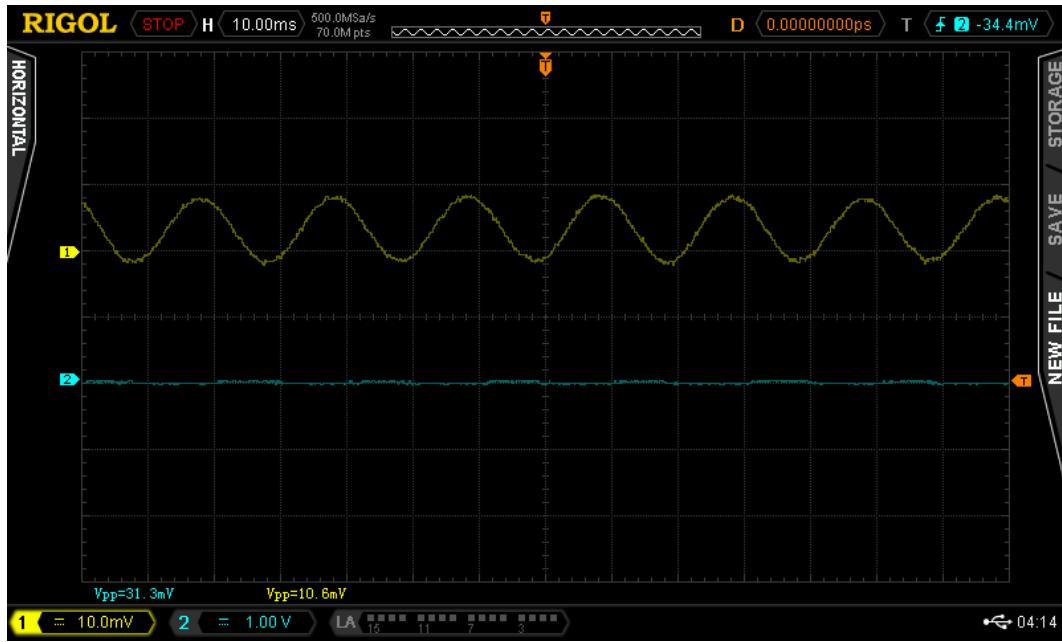


Figura 2.13: Salida del bloque rechaza banda para una entrada sinusoidal de $V_{pp} = 10[\text{mV}]$ y $50[\text{Hz}]$.

Amplificación final

El último ajuste en la ganancia también se regula con un potenciómetro, y así obtener el rango deseado de la señal acondicionada. Para efectos prácticos se reguló la resistencia variable para obtener una ganancia de 1,2, como se muestra en la Figura 2.14 (señal superior azul corresponde a entrada del bloque, mientras que señal inferior amarilla corresponde a salida del bloque). En esta se observa que la amplitud *peak-peak* es de 12,5[V] mientras que en el simulado es de 12,4[V], lo que ratifica el buen comportamiento del circuito construido con respecto a las simulaciones.

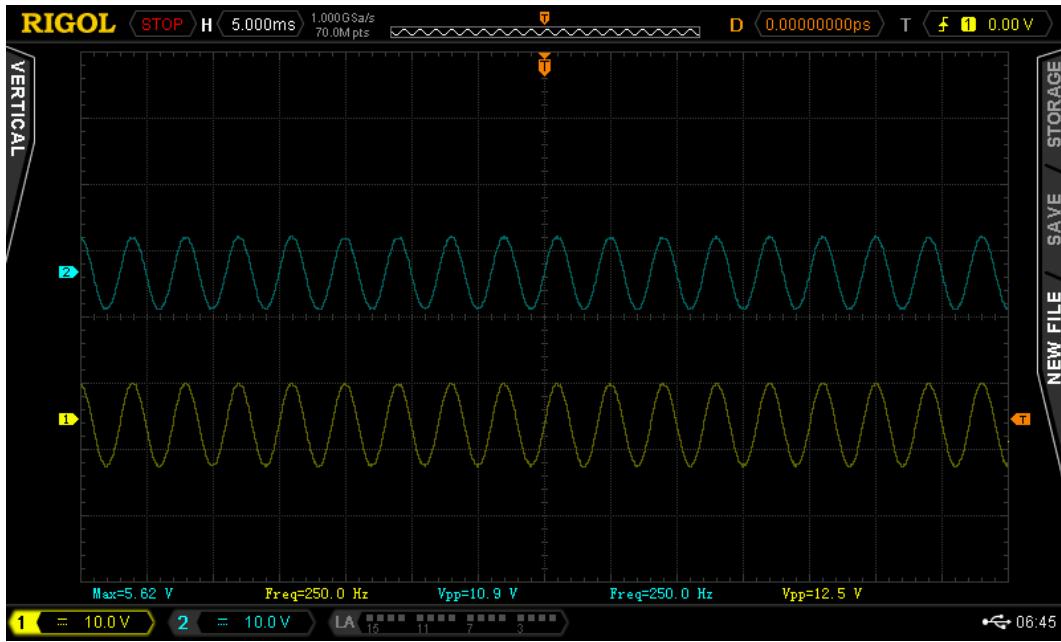


Figura 2.14: Salida del bloque de amplificación final para una entrada sinusoidal de $V_{pp} = 10[mV]$ y $250[Hz]$.

Offset agregado

Por último se construyó el circuito que le agrega el offset a la señal. En el caso de las pruebas realizadas se dio un offset de $1,5[V]$, evitando así que el rango de la señal de salida tome valores negativos.

En la Figura 2.15 se muestra una señal procesada por el circuito definitivo, con las mediciones realizadas por los electrodos. El momento de la perturbación corresponde a una contracción muscular categorizada como fuerte, realizada por el sujeto de pruebas.

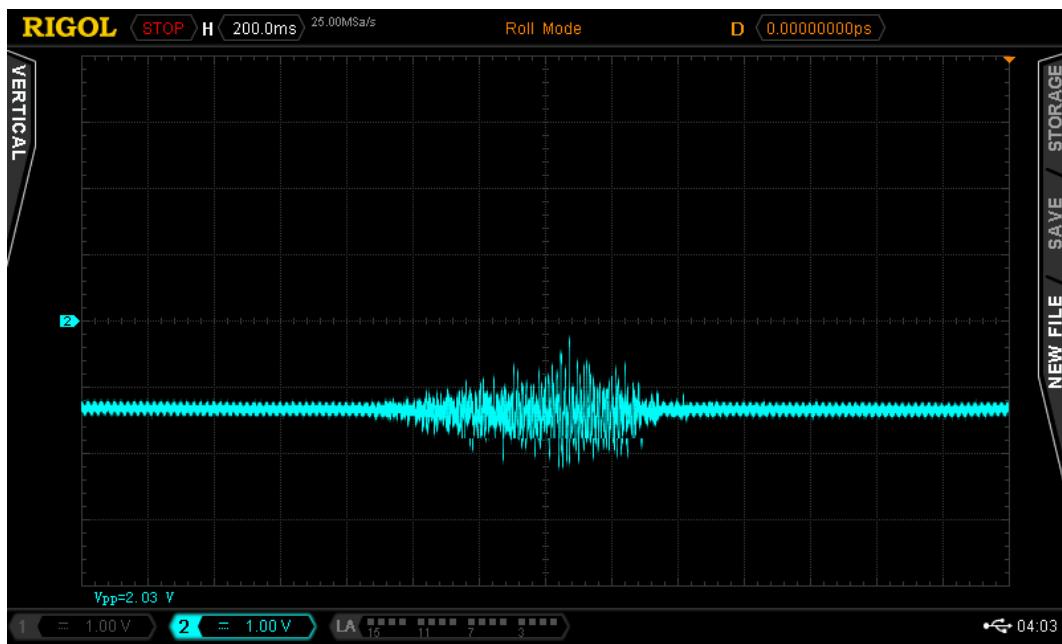


Figura 2.15: Salida del bloque del offset agregado para una medida por los electrodos.

Capítulo 3

Implementación y Entrenamiento Máquina de Aprendizaje

3.1. Conversión A/D

Para muestrear la señal analógica proveniente del circuito anterior se utilizará el conversor A/D del Arduino. En particular el Arduino DUE posee un conversor análogo-digital de 12 bits, con una frecuencia de muestreo máxima de aproximadamente $25[kHz]$. Esto hace posible muestrear la señal a una frecuencia mucho mayor que $500[Hz]$, que es la la frecuencia utilizada en el *paper* en que está basado el proyecto [2].

Para visualizar la señal digitalizada por el Arduino se utilizará una comunicación serial entre Arduino y Ordenador, que permite obtener los valores medidos directamente en el espacio de trabajo del software, y así graficarlos rápidamente. El código a utilizar para depurar el proceso de envío de datos es el mostrado en el Código fuente 3.1, en donde el vector *times* corresponde al tiempo en microsegundos y el vector *val* corresponde a los valores obtenidos del ADC para N muestras.

Código fuente 3.1: Envío de datos por Serial

```
1 Serial.println("INICIO");
2 for (int j = 0; j<N; j++){
3     Serial.print(times[j]);
4     Serial.print(",");
5     Serial.print(val[j]);
6     Serial.println(";");
7 }
8 Serial.println("FIN");
```

Por otro lado, la conversión análoga-digital se realiza con la función *analogRead* de Arduino, a un periodo aproximado de 2 microsegundos, como se muestra en el Código fuente 3.2. Cabe destacar que se transforman los datos medidos por *analogRead* en torno a valores entre $-2,5$ y $2,5$, que son formas de visualización ad'hoc a los voltajes análogos a la entrada del ADC. Además se observa que se calcula la media *mean* de los datos mientras éstos se obtienen, con el fin de disponer de ese parámetro para futuros cálculos.

Código fuente 3.2: Adquisición de datos

```

1 mean = 0;
2 while (i<N){
3     time2 = micros();
4     if (time2<time1+2200 && time2>time1+1800){
5         time1 = micros();
6         if (i<N){
7             val[i] = analogRead(analogInPin)*5.0*0.25/1024 - 2.5;
8             times[i] = time1;
9             mean += val[i];
10            i++;
11        }
12    }
13 }
14 mean = mean/N; // Media de los datos en [V]

```

3.2. Preprocesamiento de la señal

3.2.1. Extracción de características

La extracción de características tiene un rol fundamental en los resultados del clasificador, incluso puede influir más en el resultado de la clasificación la selección de características que el tipo de clasificador que se utilice.

El objetivo de esta etapa es extraer información desde los datos recolectados, tal que, se pueda discriminar con el mínimo de error los movimientos a clasificar. Las características deben ser seleccionadas de tal forma que se maximice la separabilidad entre las clases y que se minimice la redundancia entre características.

En este trabajo se proponen una serie de características que serán extraídas desde la señal original y su primer IMF, en el domino del tiempo y en el de frecuencia. Para determinar la calidad de las características se medirá la correlación entre clases por cada característica y la correlación entre características. A continuación se mencionan las características más usadas en la implementación de un sistema en base a electromiogramas, junto al código Arduino que implementa la respectiva extracción,

- **Integral del Electromiograma (IEMG):** Es el valor promedio del valor absoluto del EMG, matemáticamente se expresa de la siguiente forma:

$$IEMG = \frac{1}{N} \cdot \sum_{k=1}^N |x_k| \quad (3.1)$$

En la ecuación anterior x_k representa la k-ésima muestra de los datos obtenidos desde el EMG. En el Código fuente 3.3 se observa la implementación en Arduino.

Código fuente 3.3: Extracción de IEMG

```

1 double iemg = 0; // IEMG
2 for (int k = 0; k < N; k++)
3 {

```

```

4     iemg += abs(val[k]);
5 }
6 iemg = iemg/N;

```

- **Slope Sign Changes (SSC):** Cuenta el número de veces que la pendiente de la señal cambia de signo. Para esto se toman tres muestras desde el EMG, dadas por x_{k-1} , x_k y x_{k+1} , luego el npumero de slope sign changes biene dado por $\sum f(x)$, con f(x) calculado como sigue:

$$f(x) = \begin{cases} 1 & \text{si } (x_k < x_{k+a} \text{ y } x_k < x_{k-1}) \text{ o } (x_k > x_{k+a} \text{ y } x_k > x_{k-1}) \\ 0 & \text{otro caso} \end{cases} \quad (3.2)$$

En el Código fuente 3.4 se observa la implementación en Arduino. Para este procedimiento se deben definir dos valores anteriores val_prev y val_prev_prev con los que se comparará la señal actual. También se considera la posibilidad de que el cambio de signo de la pendiente no sea entre solamente tres puntos, lo que implica la presencia de un *if* en el código.

Código fuente 3.4: Extracción de SSC

```

1 double val_prev = -2.5;
2 double val_prev_prev = -2.5;
3 int ssc = 0; // Slope Sign Changes
4 for (int k = 0; k < N; k++)
5 {
6     ssc += (val_prev < val[k] && val_prev < val_prev_prev) ||
7         (val_prev > val[k] && val_prev > val_prev_prev);
8     if (val_prev != val[k]) // Caso en que la pendiente es nula
9         val_prev_prev = val_prev;
10    val_prev = val[k];
11 }

```

- **Waveform length (WL):** Es una variación acumulativa del EMG que puede indicar el grado de variación de la señal del EMG, esta se calcula como sigue:

$$WL = \sum_{k=1}^{N-1} (|x_{k+1} - x_k|) \quad (3.3)$$

En el Código fuente 3.5 se observa la implementación en Arduino. Se observa que se debe usar un valor anterior val_prev de la misma forma que para la extracción del SSC, lo que significa que estos valores se podrían compartir al momento de integrar todas las características.

Código fuente 3.5: Extracción de WL

```

1 double val_prev = -2.5;
2 double wl = 0; // Waleform Length
3 for (int k = 0; k < N; k++)
4 {
5     wl += abs(val[k] - val_prev);
6     val_prev = val[k];
7 }

```

- **Amplitud de Willison (WAMP):** Es el número de veces que la diferencia de amplitud de la señal EMG entre dos muestras, excede un determinado umbral. Lo anterior se puede escribir como $WAMP(x) = \sum_{k=1}^{N-1} f(|x_{k+1} - x_k|)$, con f dado por:

$$f(x) = \begin{cases} 1 & \text{si } x > \text{threshold} \\ 0 & \text{otro caso} \end{cases} \quad (3.4)$$

En el Código fuente 3.6 se observa la implementación en Arduino. De la misma forma que WL, también es posible compartir el valor anterior val_prev . En este caso se debe definir un umbral, el que por defecto es $thr = 0,1$.

Código fuente 3.6: Extracción de WAMP

```

1 double val_prev = -2.5;
2 double thr_wamp = 0.1; // umbral de Willison
3 int wamp = 0; // Amplitud de Willison
4 for (int k = 0; k < N; k++)
5 {
6     wamp += abs(val[k]-val_prev)>thr_wamp;
7     val_prev = val[k];
8 }
```

- **Varianza:** Es una medida de la densidad de potencia del EMG, destacar que la varianza es muy sensible a los *outliers*. La varianza se puede calcular como sigue:

$$VAR = \frac{1}{N-1} \sum_{k=1}^N (x_k)^2 \quad (3.5)$$

En el Código fuente 3.7 se muestra la implementación en Arduino. Se observa que para este estadístico es necesario utilizar la media calculada previamente en el muestreo de datos del conversor analógico-digital.

Código fuente 3.7: Extracción de Varianza

```

1 double var = 0; // Varianza
2 for (int k = 0; k < N; k++)
3 {
4     var += (val[k]-mean)*(val[k]-mean);
5 }
6 var = var/(N-1);
```

- **Skewness:** Es una medida de la simetría de la función de densidad de probabilidad en torno a un promedio. La asimetría puede tomar valores positivos, negativos, cero e incluso o indefinido. Una distribución simétrica tiene skewness igual a cero, si tiene una cola hacia la izquierda el skewness es negativo y si tiene una cola hacia la derecha el skewness es positivo, lo anterior se muestra en la Figura 3.1. El cálculo de esta característica se puede realizar de la siguiente forma:

$$S = \frac{E(x - \mu)^3}{\sigma^3} \quad (3.6)$$

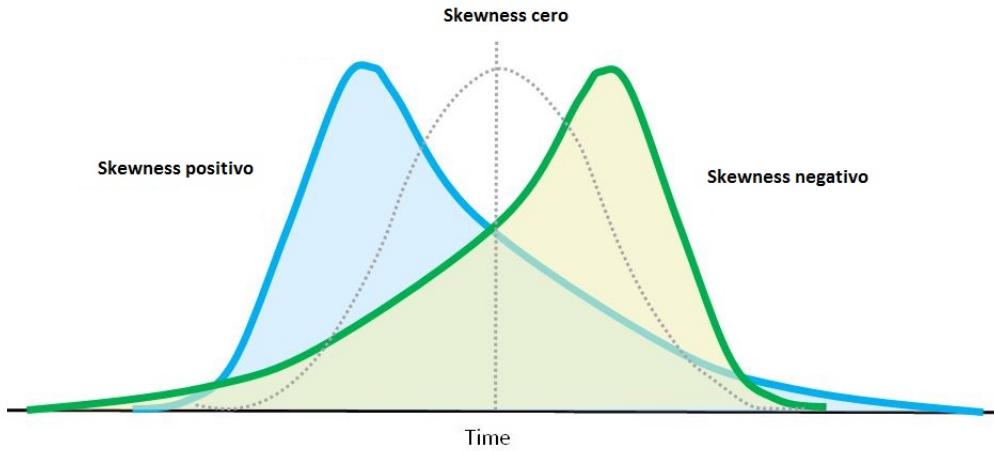


Figura 3.1: Representación del skewness de tres distribuciones

En el Código fuente 3.8 se muestra la implementación en Arduino. Se observa que para este estadístico es necesario utilizar la media y posteriormente se debe normalizar el Skewness por la varianza. Esto es posible realizar en conjunto con el cálculo de la varianza, pues esta última solamente se necesita al finalizar el ciclo *for*, en donde ya se obtuvo la varianza.

Código fuente 3.8: Extracción de Skewness

```

1 double ske = 0; // Skewness
2 for (int k = 0; k < N; k++)
3 {
4     ske += (val[k]-mean)*(val[k]-mean)*(val[k]-mean);
5 }
6 ske = ske/(N*pow(var,1.5));

```

- **Kurtosis:** Es una medida de la forma de la función de distribución, indica que tan achata o puntiaguda es una distribución (Figura 3.2), una forma de calcular esto es la siguiente:

$$S = \frac{E(x - \mu)^4}{\sigma^4} \quad (3.7)$$

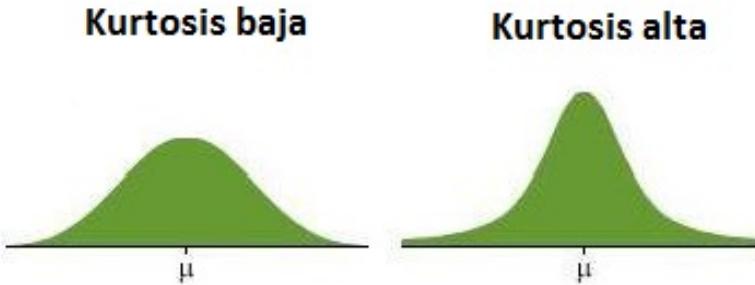


Figura 3.2: Kurtosis baja y alta

En el Código fuente 3.9 se muestra la implementación en Arduino. Se observa que para este estadístico se realiza un procedimiento similar al del cálculo del Skewness, por lo que es posible integrarlo en un mismo ciclo *for* junto a las demás características.

Código fuente 3.9: Extracción de Kurtosis

```

1 double kur = 0; // Kurtosis
2 for (int k = 0; k < N; k++)
3 {
4     kur += (val[k]-mean)*(val[k]-mean)*(val[k]-mean)*(val[k]-mean);
5 }
6 kur = kur/(N*(var*var));

```

- **Desviación estándar:** La desviación estándar es la raíz cuadrada de la varianza. Puede ser interpretada como una medida de incertidumbre. Matemáticamente se puede expresar de la siguiente forma:

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{k=1}^N (x_k - \bar{x})^2} \quad (3.8)$$

La desviación estándar es una característica dependiente de la varianza, por lo que no se realiza un cálculo de ésta. Se debe recordar que el objetivo de la extracción de características es obtener valores con la mayor independencia posible para lograr un clasificador robusto.

- **Energía para todas las muestras:** Se calcula la energía de todos los datos de la muestra, originalmente se pretende contar con 2000 muestras, ya que, se muestrearán por 2 s a 500 Hz. La energía se determina como sigue:

$$E = \sum_{k=1}^N |x_k|^2 \quad (3.9)$$

En el Código fuente 3.10 se muestra la implementación en Arduino.

Código fuente 3.10: Energía de todas las muestras

```

1 double ene_muestra = 0; // Energia de una muestra
2 double ene = 0; // Energia total
3 for (int k = 0; k < N; k++)
4 {
5     ene_muestra = val[k]*val[k];
6     ene += ene_muestra;
7 }
```

- **Energía acumulada por intervalos:** Se usará análisis por ventana deslizante sin solapamiento, esto quiere decir, que se calculará la energía por intervalos de muestras, por ejemplo, si se tienen 2000 muestras, se considerará el cálculo de energía cada 100 muestras.

La energía acumulada se puede definir como una fracción de la energía de todas las muestras, es decir para un intervalo de tiempo menor. Es por esto que se puede definir en conjunto con la energía total, como se muestra en el Código fuente 3.11.

Código fuente 3.11: Energía por intervalos

```

1 const int N_ENE = 5;      // Numero de intervalos
2 double ene_int[N_ENE];   // Energia por intervalos
3 double ene_muestra = 0; // energia de una muestra
4 for (int k = 0; k < N; k++)
5 {
6     ene_muestra = val[k]*val[k];
7     ene_int[(int)k*N_ENE/N] = ene_muestra;
8 }
```

3.3. Resultados de la Extracción de Características

Para comprobar la correcta conversión análogo-digital del Arduino y las extracciones de características, se generaron un conjunto de señales de baja frecuencia mediante el generador de señales. Sin embargo todas las señales deben tener características en común por las restricciones del ADC.

Como la frecuencia de muestreo utilizada es de $500[Hz]$, se debe tener en consideración señales de frecuencia menor o igual a $250[Hz]$ por consecuencia del teorema de muestreo de Nyquist-Shannon. La cantidad de puntos por señal es asignada a $N = 1000$. En particular para una buena observación de los datos graficados posteriormente, las señales generadas tienen una frecuencia de no más de $2[Hz]$.

Por otro lado se debe tener en consideración que el ADC del Arduino DUE soporta voltajes entre 0 y $3,3[V]$, por lo que las señales se deben escalar a $3,3[V_{pp}]$, con un offset de $1,7[V]$. Una vez realizado este acondicionamiento el Arduino puede transformar los bits entregados por el ADC a la escala que se desee.

En la Figura 3.3 se muestran 4 señales diferentes a las que se sometió el Arduino. Se observa que la escala de voltaje es la esperada con respecto a la normalización de la señal efectuada por el código Arduino. Además la resolución del ADC basta como para identificar señales típicas.

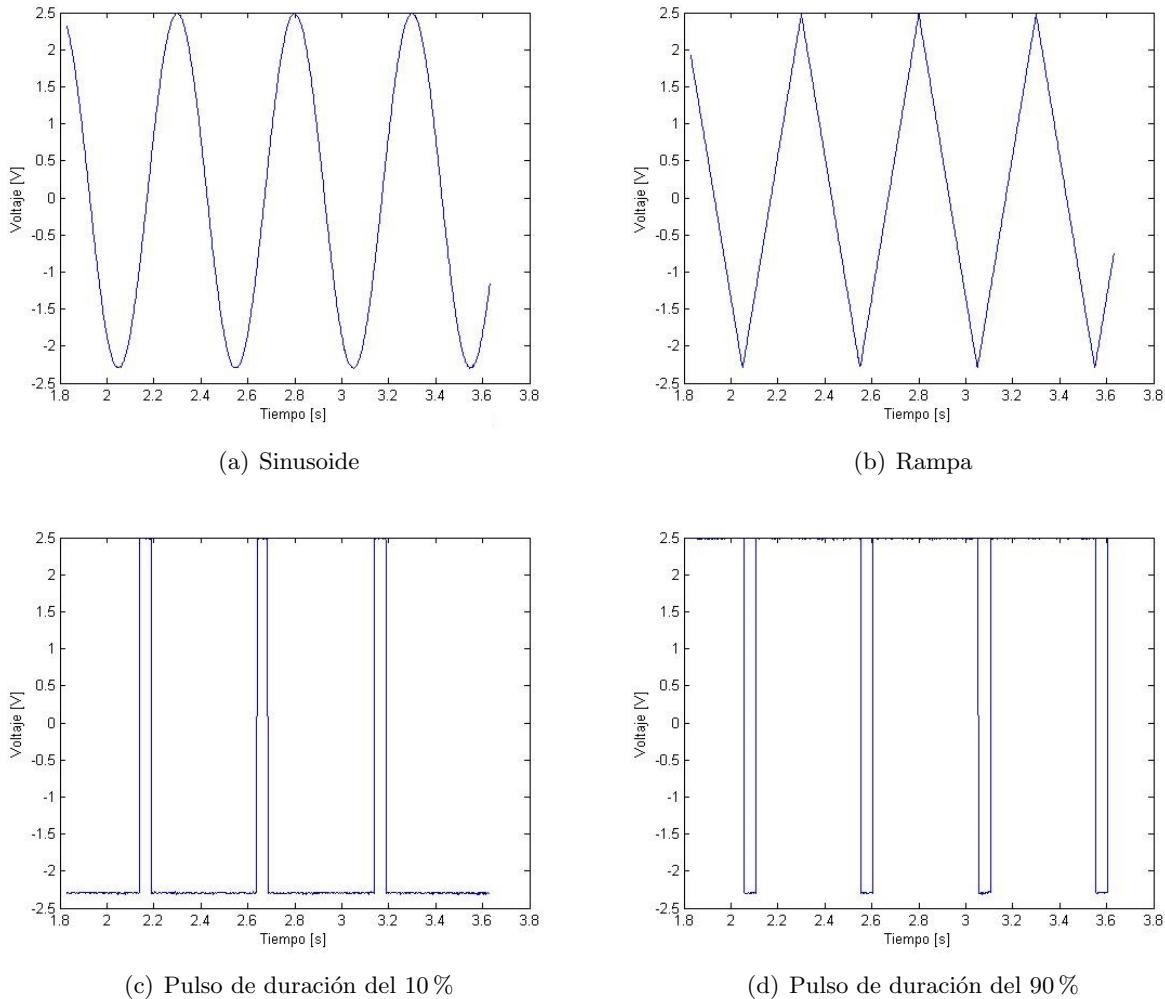


Figura 3.3: Entradas de señales de 2 [Hz] al ADC

Por otro lado, en la Figura 3.4 se muestran señales ruidosas. En primera instancia la Figura 3.4(a) corresponde al ADC con entradas en circuito abierto. En ella se observa que el voltaje medido varía entre los 50[mV] y 0[mV], lo que significa una energía total relativamente pequeña. Por otro lado la Figura 3.4(b) corresponde a una entrada ruidosa de amplitud pico-pico de 3,3[V] y un offset de 1,7[V].

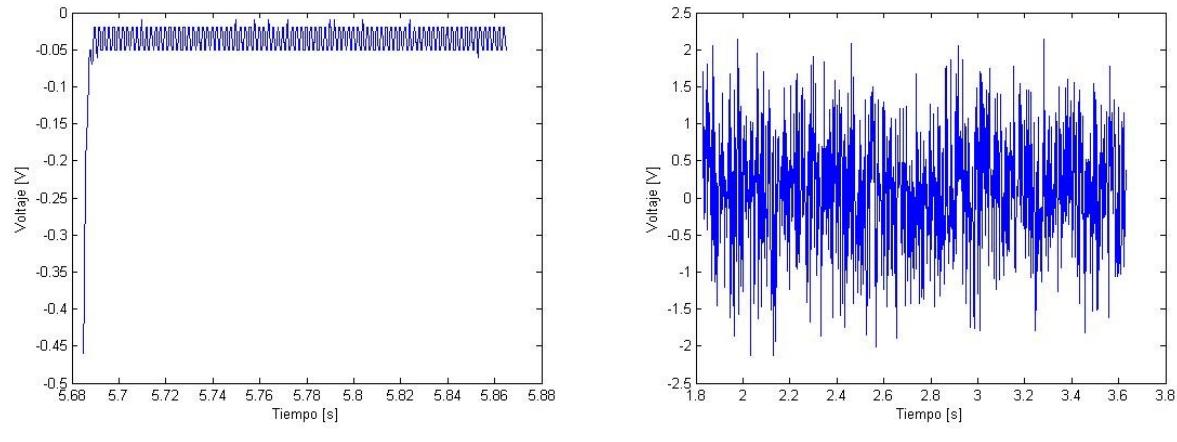


Figura 3.4: Entradas de Ruido al ADC

En el Cuadro 3.1 se muestran las principales características de cada una de las señales anteriores. A simple vista se observa que cada señal tiene características particulares que las diferencian entre ellas, lo que significa que hacer un clasificador para este tipo de señales es factible. En la etapa de entrenamiento, del siguiente módulo, se realizará la extracción de características para entradas obtenidas con los electrodos, ante distintos movimientos de la mano.

Cuadro 3.1: Características de distintas señales de frecuencia 2 [Hz]

	IEMG	SSC	WL	WAMP	Media	VAR	Skewness	Kurtosis	Energía
Sinusoide	1.53	28	39.47	0	-0.03	2.86	0.10	1.50	2857.57
Rampa	1.20	8	38.78	0	-0.01	1.90	0.08	1.79	1895.21
Pulso 10 %	2.31	529	33.75	6	-1.90	1.74	3.02	10.12	5350.05
Pulso 90 %	2.47	511	47.83	8	1.95	2.28	-2.46	7.04	6079.83
Ruido CA	0.04	178	8.60	0	-0.04	0.00	-8.38	107.25	2.21
Ruido blanco	0.67	668	946.33	936	0.10	0.67	-0.10	2.59	682.80

Para comprobar que el cálculo de características implementado en Arduino estuviera correcto, se procedió a obtener las características del cuadro 3.1, para la misma entrada sinusoidal, obteniendo valores similares .Por ejemplo para IEMG se obtuvo 1.53 con Arduino y 1.5256 con Matlab, difiriendo en los decimales, debido a que Arduino aproxima sus resultados numéricos con menor representación decimal que Matlab. El código con el cuál se calcularon las características en Matlab se encuentra en anexos.

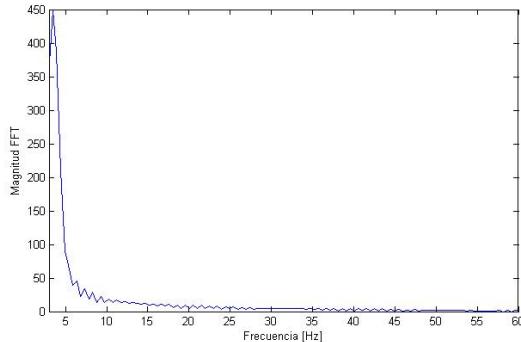
Por otro lado se calculó la transformada de Fourier en Arduino con la librería SplitRadixReal FFT. En la Figura 3.5 se muestran tres espectros de Fourier obtenidos a partir de los valores entregados por Arduino.

La Figura 3.5(a) corresponde a la FFT de la sinusoide de la Figura 3.3(a), en donde se observa un *peak* de $3,4[Hz]$ cercano a la frecuencia original de $2[Hz]$. Además la presencia de frecuencias altas es

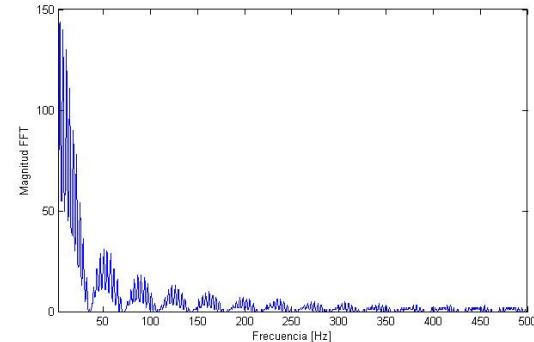
despreciable, lo que se espera de una señal sinusoidal limpia.

La Figura 3.5(b) corresponde a la FFT del pulso de la Figura 3.3(c), en donde se observa un peak cercano a la frecuencia principal de $2[\text{Hz}]$. También aparece la función *sinc* típica de los pulsos, lo que demuestra una vez más la efectividad de la FFT.

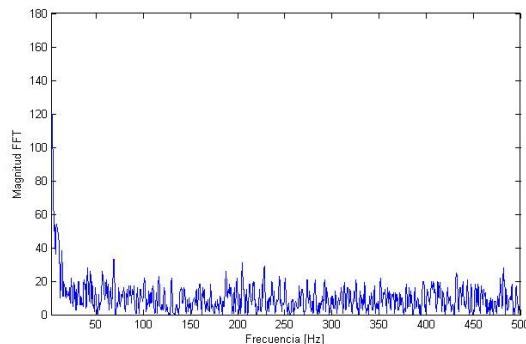
Por último la Figura 3.5(c) corresponde a la FFT del ruido blanco de la Figura 3.4(b). Se observa que la señal posee una gran cantidad de frecuencias a lo largo de todo el ancho de banda, lo que es característico del espectro de Fourier del ruido blanco.



(a) Espectro de Fourier sinusode de 2 Hz



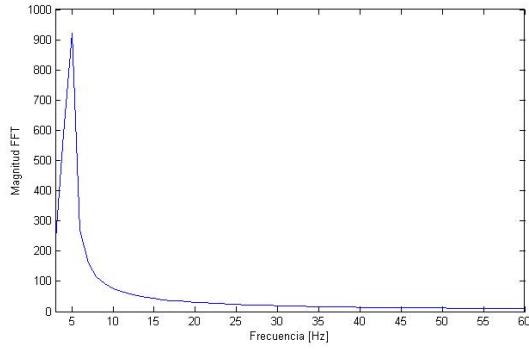
(b) Espesctro de Fourier pulso de 2 Hz de duración del 10 %



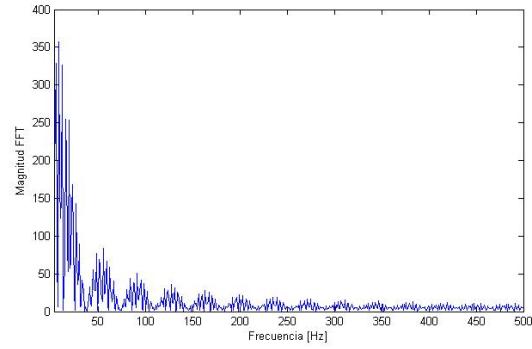
(c) Espectro de Fourier de ruido blanco

Figura 3.5: Espectros de Fourier de diferentes señales de entrada al Arduino

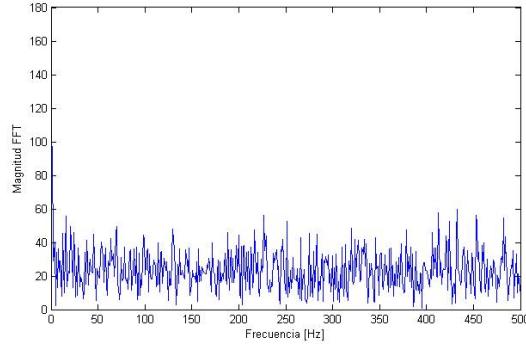
Con el fin de verificar de otra forma la efectividad de la librería de la transformada de Fourier, se calculó en Matlab la Fast Fourier Transform (FFT) para las tres señales anteriores (Figura 3.6). Claramente se observa que los espectros de Fourier tanto de Arduino como de Matlab son similares para las tres señales de muestra, lo que comprueba la efectividad de la transformada de Fourier implementada en Arduino.



(a) Espectro de Fourier de Matlab sinusoide de 2 Hz



(b) Espectro de Fourier de Matlab pulso de 2 Hz de duración del 10 %



(c) Espectro de Fourier de Matlab de ruido blanco

Figura 3.6: Espectros de Fourier calculados con FFT de Matlab para diferentes señales de entrada al Arduino

3.4. Antecedentes Support Vector Machine

En el contexto de aprendizaje el *Support Vector Machine* es un método que permite la clasificación de clases. En términos básicos el SVM encuentra un plano que separa a dos clases de acuerdo a sus características (en este caso, las calculadas en la sección anterior). Lo anterior corresponde a separar las características/patrones de forma lineal. Esto se puede extender para planos no lineales. A continuación se describe el método para ambos casos.

3.4.1. Patrones Linealmente Separables

Para separar las dos clases se puede definir la ecuación del hiperplano que las separa, la cual se expresa como sigue:

$$\mathbf{w}^t \cdot \mathbf{x} + b = 0 \quad (3.10)$$

Donde \mathbf{w} es un vector normal al hiperplano (vector de pesos), \mathbf{x} es un vector de entrada y b es el bias que corresponde a un valor constante. Para un cierto \mathbf{w} y b , la separación entre el hiperplano y el conjunto de datos más cercano se denomina *margen*. Cuando los datos son no separables se pueden seleccionar dos planos tal que entre ellos no se encuentre ningún dato, para seleccionar toda la porción de los datos de una clase o la otra, estos dos planos se pueden definir como:

$$\mathbf{w}^t \cdot \mathbf{x} + b \geq 1 \quad (3.11)$$

$$\mathbf{w}^t \cdot \mathbf{x} + b \geq -1 \quad (3.12)$$

Luego por geometría se puede determinar que la distancia entre los dos planos definidos por las igualdades de las ecuaciones (3.11) y (3.12) está dada por $\frac{2}{\|\mathbf{w}\|}$, por lo tanto, para maximizar el margen se tiene que minimizar \mathbf{w} , además se puede obtener que la distancia desde el origen al hiperplano óptimo, dado por la ecuación (3.10), es $\frac{b}{\|\mathbf{w}\|}$. En la Figura (3.7) se pueden observar los hiperplanos con línea segmentada que definen el margen y el hiperplano óptimo, para un problema de dos clases.

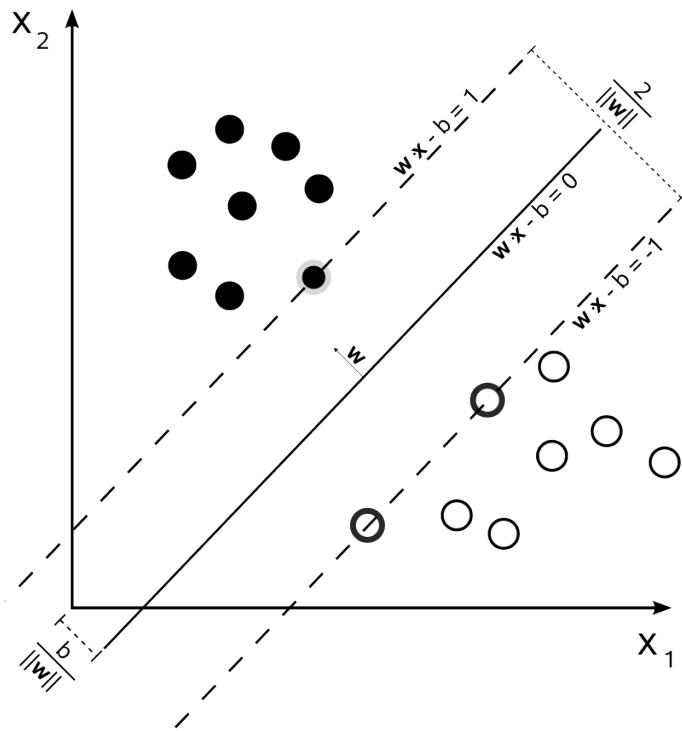


Figura 3.7: Separación de dos clases (círculos y cuadrados) mediante hiperplanos

En el problema de minimización antes expuesto se puede cambiar sin alterar el resultado $\|w\|$ por $\frac{\|w\|^2}{2}$, donde el $\frac{1}{2}$ se agrega por conveniencia matemática, además la minimización de la expresión $\frac{\|w\|^2}{2}$ está restringida a cumplir con $y_i(w^t \mathbf{x}_i + b) \geq 1$, donde y_i puede tomar valores de $+1$ o -1 dependiendo a qué clase pertenezca. En virtud de lo antes expresado este es un problema de optimización con restricciones y se puede resolver utilizando los multiplicadores de Lagrange, entonces la función lagrangeana para el problema, se puede expresar como:

$$\mathcal{L}(w, b, \alpha_i) = \frac{w^t w}{2} - \sum_{i=1}^N \alpha_i [d_i(w^t \mathbf{x}_i + b) - 1] \quad (3.13)$$

Luego aplicando las condiciones de optimalidad se pueden obtener los parámetros del clasificador (w y b) de la siguiente forma:

$$w = \sum_{i=1}^N \alpha_i d_i \mathbf{x}_i \text{ con } \alpha_i \neq 0 \quad (3.14)$$

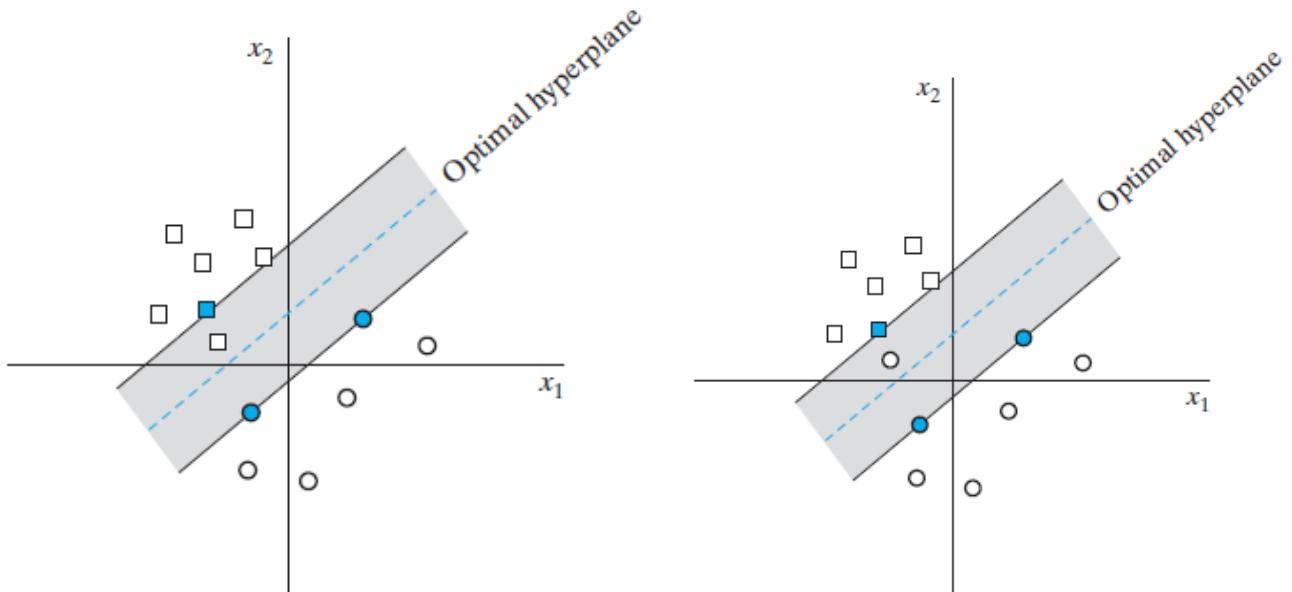
$$b = 1 - \sum_{i=1}^N \alpha_i d_i \mathbf{x}_i^t \mathbf{x} \quad (3.15)$$

3.4.2. Patrones No Linealmente Separables

En el caso en que los patrones no sean linealmente separable, no se podrá encontrar un hiperplano que separe de forma exacta los datos, por lo tanto, en este caso se busca encontrar un hiperplano tal que se minimice la probabilidad de clasificación errónea. Para tratar con estos datos mal clasificados, se introduce en la ecuación del hiperplano de decisión, una variable escalar no negativa denominada ξ_i , que recibe el nombre de *slack variables*, entonces la ecuación de la superficie de decisión se expresa como:

$$d_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad i = 1, 2, \dots, N \quad (3.16)$$

La variable ξ_i mide el grado de clasificación errónea, para $0 < \xi_i \leq 1$ los datos caen dentro de la región de separación y en el lado correcto, como se aprecia en la Figura 3.8 (a), por el contrario cuando $\xi_i > 1$ los datos caen en el lado incorrecto de la separación, como se observa en la Figura 3.8 (b)



(a) Datos mal clasificados dentro del margen pero en el lado correcto (b) Datos mal clasificados dentro del margen pero en el lado incorrecto

Figura 3.8: Región de decisión caso no separable

En este casos los vectores de soporte son aquellos que satisfacen la ecuación (3.16), incluso pueden existir vectores que cumplan con $\xi_i = 0$. El funcional que se desea minimizar con respecto a \mathbf{w} se puede escribir como sigue:

$$\Phi(\mathbf{w}, \xi_i) = \frac{\mathbf{w}^T \mathbf{w}}{2} + C \sum_{i=1}^N \xi_i \quad (3.17)$$

En la ecuación (3.17) el parámetro C controla la compensación entre la maximización del margen y la correcta clasificación de los datos. Si C se escoge como un valor grande quiere decir que existe una mayor penalización de los errores, mencionar que el segundo término de la ecuación (3.17) se relaciona con el

error de clasificación. La forma dual del problema se puede expresar de la siguiente forma:

$$Q(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j \quad (3.18)$$

$$\text{Sujeto a: } \sum_{i=1}^N \alpha_i d_i = 0 \quad (3.19)$$

$$0 \leq \alpha_i \leq C \quad (3.20)$$

En el problema dual anterior, lo que se busca determinar son los multiplicadores de Lagrange $\{\alpha_i\}_{i=1}^N$, tal que se maximice la función objetivo $Q(\alpha)$, el conjunto de α_i definen el vector de soporte buscado.

3.4.3. Aumento de dimensionalidad utilizando un kernel

Destacar otro aspecto presente en el SVM el cual se relaciona con la idea que el espacio original de características puede ser trasladado a un espacio de mayor dimensión, donde el problema sí sea separable, esta traspaso de una dimensión a otra se realiza por una función denominada *Kernel*, dentro de las más comunes destacan la polinomial, gaussiana y tangente hiperbólica.

Si \mathbf{x} es el vector que representa los datos de entradas al clasificador, que pertenecen a un espacio de dimensión m_0 , entonces se denota $\varphi(\mathbf{x})$ como una función no lineal que aumenta la dimensión del espacio de entrada. Dada esta transformación se puede definir un hiperplano actuando como la superficie de decisión, lo cual se escribe de la siguiente forma:

$$\sum_{j=1}^{\infty} w_j \varphi_j(\mathbf{x}) = 0 \quad (3.21)$$

El término $\{w_j\}_{j=1}^{\infty}$ en la ecuación(3.21), denota un conjhunto de vectores pesos infinitamente largos, que transforman las características del espacio de entrada al espacio de salida. La ecuación (3.21) se puede escribir matricialmente como:

$$\mathbf{w}^T \phi(\mathbf{x}) = 0 \quad (3.22)$$

La ecuación (3.14) se puede escribir de la siguiente forma:

$$\mathbf{w} = \sum_{i=1}^{N_s} \alpha_i d_i \phi(\mathbf{x}_i) \quad (3.23)$$

En la ecuación anterior N_s es el número de vectores de soporte y el vector de características es expresado como $\phi(\mathbf{x}_i) = [\varphi_1(\mathbf{x}_1), \varphi_2(\mathbf{x}_2), \dots]^T$. Usando las ecuaciones (3.22) y (3.23), se obtiene lo siguiente:

$$\sum_{i=1}^{N_s} \alpha_i d_i \phi^T(\mathbf{x}_i) \phi(\mathbf{x}_i) = 0 \quad (3.24)$$

El término $\phi^T(\mathbf{x}_i) \phi(\mathbf{x}_i)$ define un producto punto, además este es el denominado Kernel de SVM, en virtud de los anterior, se tiene que el Kernel está dado por:

$$k(\mathbf{x}, \mathbf{x}_i) = \phi^T(\mathbf{x}_i) \phi(\mathbf{x}_i) \quad (3.25)$$

$$= \sum_{j=1}^{\infty} \varphi(\mathbf{x}_i) \varphi(\mathbf{x}) \quad \text{con } i = 1, 2, \dots, N_s \quad (3.26)$$

Finalmente para encontrar los elementos del vector de soporte que permite maximizar el margen de separación entre clases, se obtiene de resolver el problema dual de la sección anterior, dado por las ecuaciones (3.18), (3.19) y (3.20). La única diferencia es que en la ecuación (3.18) el término $\mathbf{x}_i^t \mathbf{x}_j$, se reemplaza por $k(\mathbf{x}_i, \mathbf{x}_j)$.

Para ilustrar el beneficio de un aumento de la dimensión de los datos de entrada, se muestra la Figura 3.9, en la cual en primera instancia los datos de las dos clases (puntos rojos y azules) se ubican sobre una recta, donde no es posible separar los datos por un hiperplano lineal, sin embargo, al trasladarse a un espacio de dos dimensiones, mediante un kernel cuadrático, se obtiene que las clases son separables fácilmente por un hiperplano lineal.

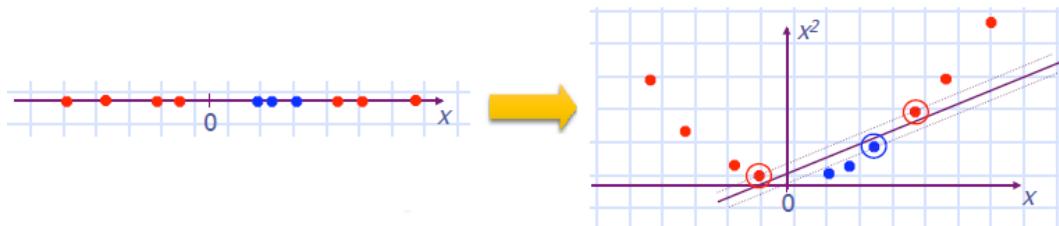


Figura 3.9: Efecto del aumento de dimensionalidad para lograr trabajar en un espacio donde las clases sean fácilmente separables

Para la implementación de SVM, se determinarán los vectores de soporte en un computador, debido a la capacidad de cálculo, es decir, el proceso de entrenamiento no se realizará en el Arduino Due. Una vez obtenidos los vectores de soporte se implementará el clasificador en el Arduino. Se pretende utilizar un kernel gaussiano para el aumento de dimensionalidad, debido a su efectividad, para trabajar con señales provenientes de electromiogramas. La base de datos para realizar el entrenamiento se obtendrá con el circuito implementado en el módulo 2.

3.5. Implementación de SVM

Recordando que la utilización de SVM tiene el fin de clasificar 4 movimientos realizados con el antebrazo, los que se captan a través de electrodos, se deben definir los 4 tipos previamente al entrenamiento para extraer los datos con los que funcionará el algoritmo. Los movimientos son:

- Mano quieta
- Apretar puño
- Apretar cilindro
- Sostener peso

Los movimientos a procesar se pueden ver en la Figura 3.10.

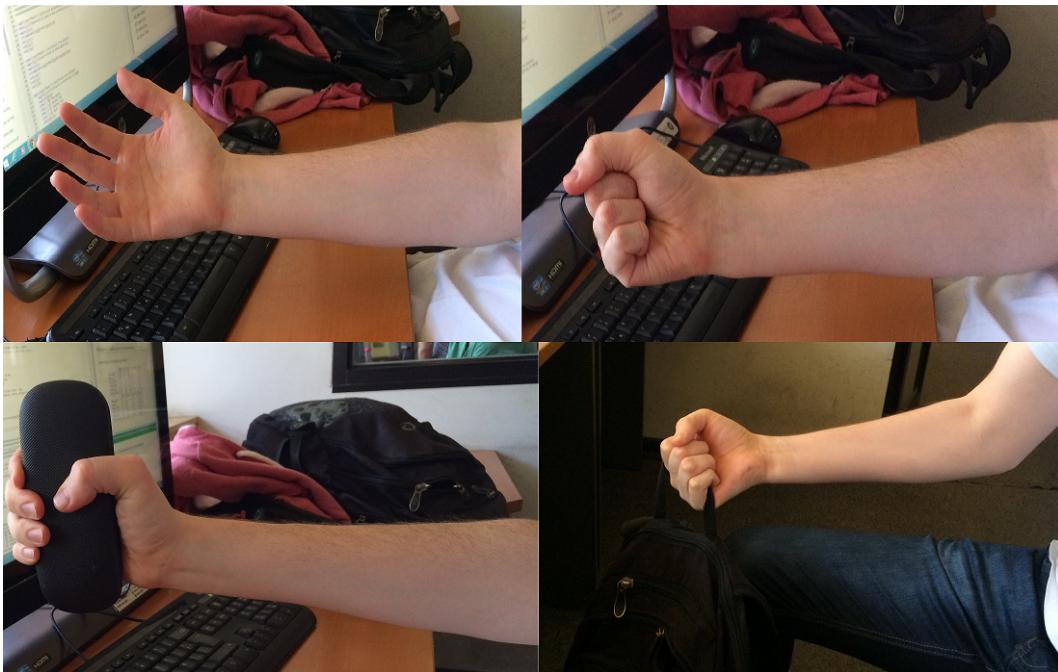


Figura 3.10: Movimientos a clasificar. Superior izquierda: mano quieta. Superior derecha: apretar puño. Inferior izquierda: apretar cilindro. Inferior derecha: sostener peso.

3.5.1. Entrenamiento off line

Para generar un correcto sistema de clasificación mediante SVM, se desarrolló en Matlab un *script* que recibe las series de tiempo de diferentes movimientos, le calcula las características consideradas previamente, construye el clasificador y retorna sus parámetros. Esto se testeó con la base de datos utilizados en [2].

En la Figura 3.11 se visualizan dos muestras correspondientes a diferentes movimientos a clasificar, pero que pertenecen al par de movimientos más similar entre todos (movimiento cilíndrico y esférico). Cabe

destacar que a simple vista no es posible distinguir una gran diferencia entre ambos. Sin embargo al extraer las características de éstos y junto con su posterior separación mediante SVM se obtiene un error de clasificación del 6,7 %, para un total de 60 muestras.

Los parámetros generados por el clasificador corresponden a un Kernel Gaussiano de la forma:

$$k_{\sigma}(\mathbf{x}, \mathbf{y}) = e^{-\frac{(\mathbf{x}-\mathbf{y})(\mathbf{x}-\mathbf{y})^T}{2\sigma^2}} \quad (3.27)$$

donde el ancho de banda encontrado es $\sigma = 2,1659$. Además para una mejor generalización, las características se normalizaron de forma de tener media 0 y varianza unitaria.

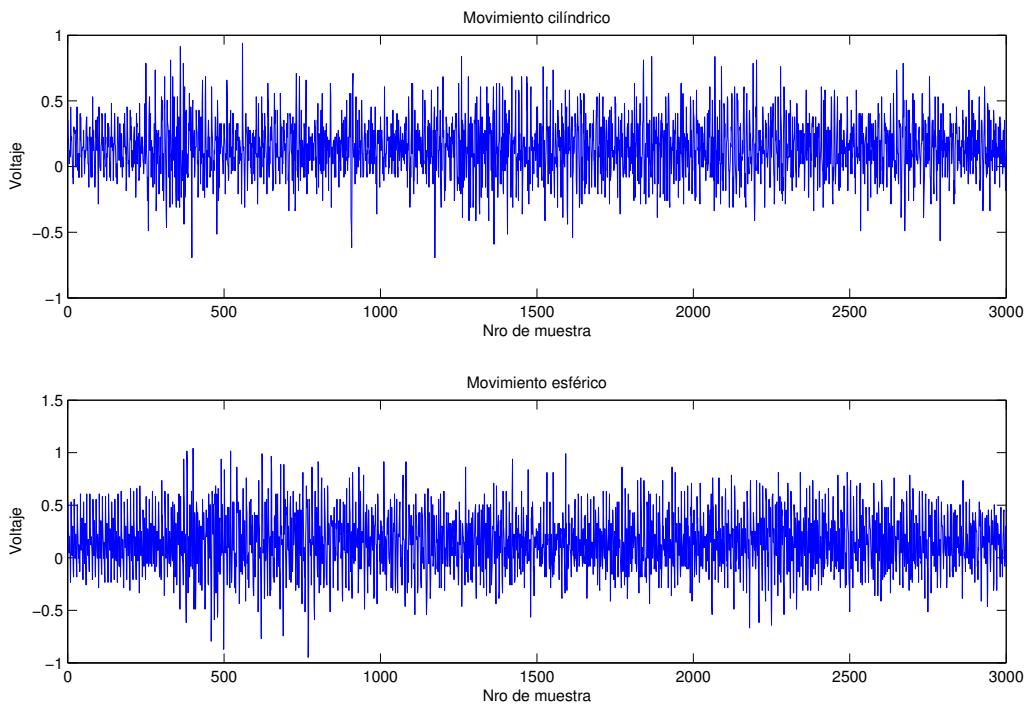


Figura 3.11: Muestra de movimientos de la base de datos utilizada en [2]

El proceso de entrenamiento y verificación se muestra en el Código fuente 3.12, el que sería extendible para cualquier base de datos que siga la estructura utilizada. En particular, se podrán entregar los datos muestreados con el Arduino para construir el clasificador de acuerdo a los movimientos reales a utilizar.

Código fuente 3.12: Entrenamiento y Clasificación mediante SVM.

```

1 %% Cargar datos
2 male1 = load('male_1.mat');
3 %% Movimientos
4 mov1 = male1.cyl_ch1;
5 mov2 = male1.spher_ch1;
```

```

6 %% Obtener caracteristicas
7 c_mov1 = get_caract(mov1);
8 c_mov2 = get_caract(mov2);
9 %% Juntar muestras y asignar clases (1: mov1, -1: mov2)
10 data = [c_mov1;c_mov2];
11 class = ones(size(data,1),1);
12 class(size(c_mov1,1)+1:size(data,1)) = -1;
13 %% Entrenar modelo
14 model = fitcsvm(data,class,'KernelFunction','rbf',...
15 'BoxConstraint',Inf,'ClassNames',[1,-1],...
16 'Standardize',true,'KernelScale','auto');
17 %% Resultados clasificacion fallida
18 CVmodel = crossval(model);
19 misclass = kfoldLoss(CVmodel);
20 misclass
21 %% Clasificar una muestra
22 muestra = data(11,:);
23 score = get_score(muestra,model.SupportVectors,...
24 model.SupportVectorLabels,model.KernelParameters.Scale,...
25 model.Alpha,model.Sigma,model.Mu,model.Bias);
26 clase = sign(score)

```

Como el clasificador logra un porcentaje de clasificaciones correctas superior al 90 % para las clases más similares, se infiere que es posible construir múltiples clasificadores conectados en cascada para decidir a cuál de todos los movimientos corresponde la serie de tiempo. Esto se implementará en el código Arduino final.

3.6. SVM con datos de Arduino

Se muestrearon 4 movimientos diferentes, con 15 muestras para cada uno. La tasa de muestreo es de 500 muestras por segundo, con un total de 1024 muestras. Los movimientos son los siguientes:

- Mano quieta.
- Apretar puño.
- Sostener peso con brazo horizontal.
- Apretar cilindro.

En la Figura 3.12 se observa una muestra para cada movimiento. De la misma forma que en la sección anterior, si bien existen movimientos distinguibles a simple vista (cerrar puño), la mayoría se confunde entre ellas

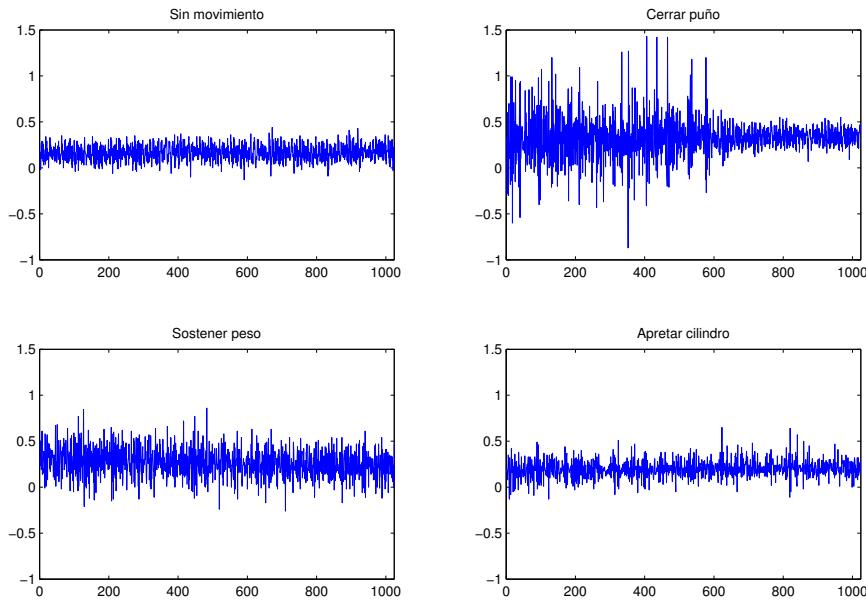


Figura 3.12: Muestra de movimientos obtenidos con Arduino

Al clasificar cada movimiento uno-contra-uno se obtuvieron los errores mostrados en el Cuadro 3.2. Como se observa el mayor error ocurre entre los movimientos Apretar puño y Sostener peso, el que alcanza un 20 % de clasificaciones erróneas. Sin embargo se debe recordar que solamente se está utilizando 1 canal, por lo que aún no se han extraído las características totales a utilizar en el proyecto.

Por otro lado, se observa que las demás tasas de error de clasificación no superan el 10 %, lo que es un buen indicador para el único canal utilizado. Debido a esto y a los resultados de la sección anterior, se verifica que conectar clasificadores uno-contra-uno en cascada es una solución factible para el problema de múltiples clases, que es lo que se buscaba en este módulo.

Finalmente, sólo basta transcribir el código implementado en Matlab al Arduino. Esto se hace directamente pues el Arduino solamente recibe los parámetros del clasificador (generados en Matlab), sin la necesidad de usar alguna librería adicional. El código de la clasificación para un clasificador se puede ver en el Código fuente 3.13. En él se puede observar que se utilizan arreglos definidos previamente, los que corresponden a los parámetros del clasificador:

- bias: sesgo del clasificador.
- sv: vectores de soporte del clasificador.
- label: etiquetas de clase de los vectores de soporte.
- mu: media de las características. Utilizada para que la media de la entrada al clasificador sea igual a 0.

- sigma: varianza de las características. Utilizada para normalizar la varianza de la entrada al clasificador al valor unitario.
- scale: parámetro del kernel gaussiano. Corresponde a su desviación estándar.
- alpha: peso de las características. Determina qué característica influye más al momento de clasificar.

Código fuente 3.13: Obtención de puntaje de clasificación. El signo del puntaje indica la clase de las características x de una señal

```

1 float get_score_rbf_SVM( float x[ nro_caract ] ){
2     float score = bias;
3     for ( int i=0; i<nro_vectores_soporte ; i++ ){
4         float kernel = 0.0;
5         for ( int j=0; j<nro_caract ; j++ ){
6             kernel = kernel + ((x[j]-mu[j])/sigma[j] - sv[i][j])*((x[j]-mu[j])/sigma[j] - sv[i][j]);
7         }
8         score = score + alpha[i]*label[i]*exp(-kernel/(2*scale*scale));
9     }
10    return -score;
11 }
```

Cuadro 3.2: Tasas de error de clasificación entre clases

	Mano quieta	Apretar puño	Sostener peso	Apretar cilindro
Mano quieta	0	0.0690	0.0230	0.0294
Apretar puño	0.0690	0	0.2000	0.0303
Sostener peso	0.0230	0.2000	0	0.0571
Apretar cilindro	0.0294	0.0303	0.0571	0

Capítulo 4

Construcción Mano Biómica e Integración de los Módulos

4.1. Impresión en 3D de la Mano Biómica

4.1.1. Modelo de la Mano

Las piezas que se utilizarán en la construcción serán impresas con PLA de color negro de 1.75 mm de diámetro. La primera parte a imprimir será la mano (Figura 4.1), la cual consta de 5 dedos formados por 6 piezas cada uno. Además cuenta con una palma que posee 3 porciones, y posee un recubrimiento el cual se aprecia de color amarillo en la Figura 4.1.

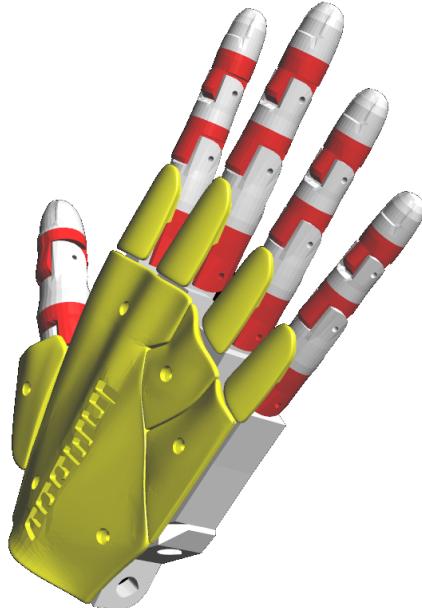


Figura 4.1: Modelo de la mano que se imprimirá en 3D

La unión entre las porciones que forman la palma de la mano, se realizarán con dos pernos (Figura 4.2), los cuales también se imprimirán en material de PLA. Por su parte la unión entre las 6 piezas de los dedos se implementará con alambre galvanizado de 1.24 mm y 0.71 mm de diámetro, como se muestra en la Figura 4.3. Mencionar que las partes de los dedos que no se ensamblan con alambre, se unen con pegamento epoxy.

Para lograr el movimiento de las falanges artificiales de la mano biónica se utilizará hilo de pescar, que será ajustado de tal forma que la palma esté siempre abierta. El hilo de pescar simulará los tendones de un cuerpo humano real, la contracción de los ligamentos se realizará por medio de servo motores.

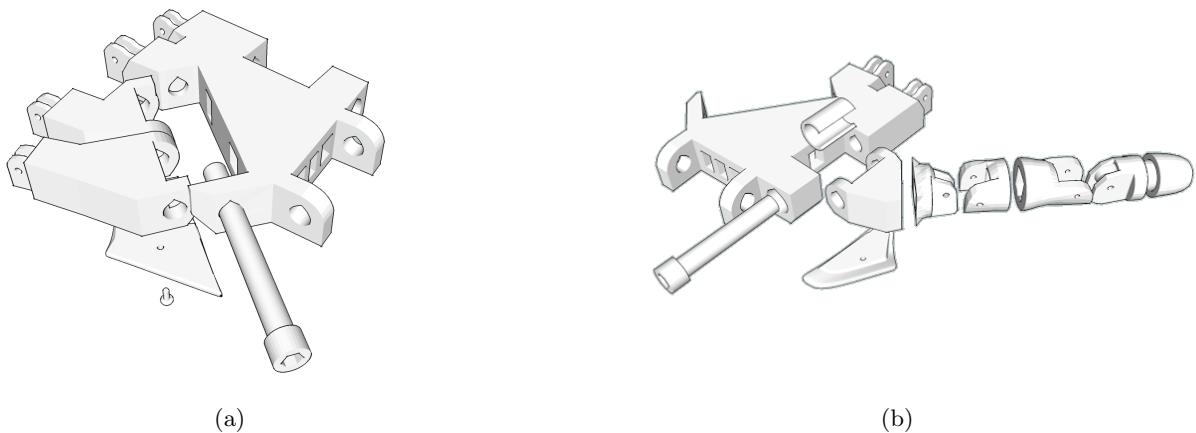


Figura 4.2: Pernos que unen los dedos a las piezas de la palma

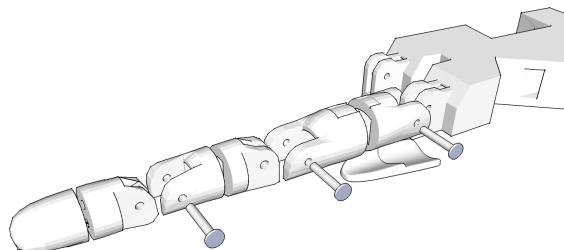
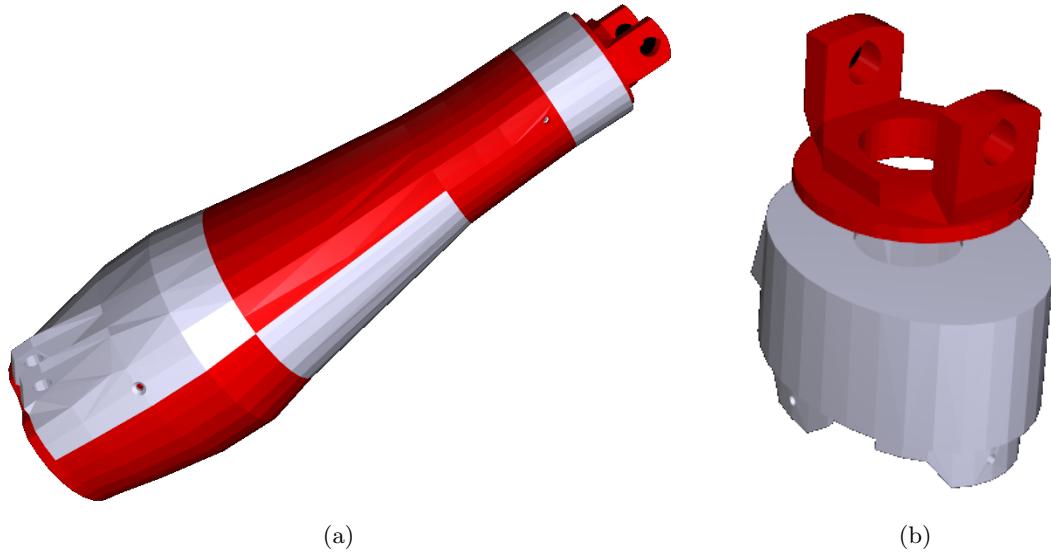


Figura 4.3: Ensamblaje de las piezas de los dedos

4.1.2. Modelo del Brazo y la Muñeca

El siguiente paso es construir el brazo y la muñeca. Esta última es solamente un elemento de unión entre la mano y el brazo, ya que no se ha contemplado movimiento de ésta. El modelo completo se observa

en la Figura 4.4 (a). La muñeca está compuesta por dos partes las cuales se pueden observar en la Figura 4.4 (b).



(a)

(b)

Figura 4.4: Modelo del brazo y muñeca

La unión entre las distintas piezas se realizará con pernos impresos en 3D, pegamento epoxy y alambre galvanizado de 1.24 mm y 0.71 mm. Un aspecto importante en la construcción del brazo, es que dentro de este se dispondrá un lugar para los 5 servomotores (Figura 4.5), los cuales se ensamblarán al brazo por medio de tornillos.

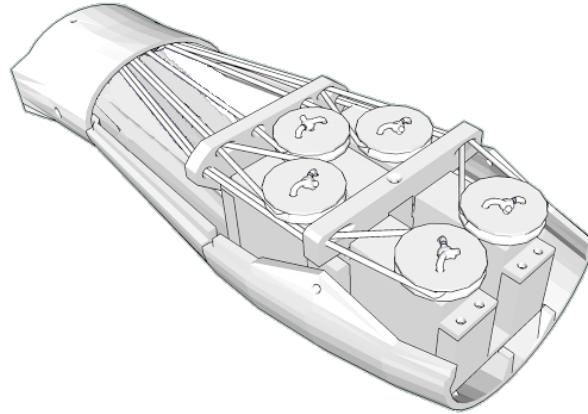


Figura 4.5: Ubicación de los servomotores dentro del brazo

4.2. Ensamblado de la Mano Biónica

4.2.1. Modelo de la Mano

En primera instancia se imprimió el conjunto completo de piezas que componen la mano. En la Figura 4.6 se observa el proceso de impresión de una de las partes de la mano, el cual tardó aproximadamente 2 horas 30 minutos.

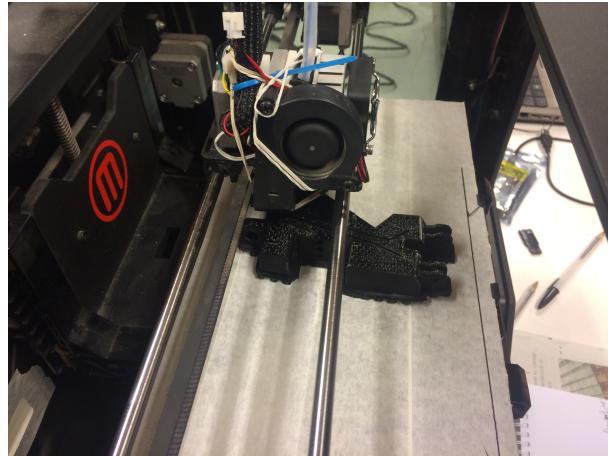


Figura 4.6: Impresión de parte de la palma de la mano

Cabe destacar que el proceso de impresión es relativamente lento, pues en promedio cada pieza tardaba, para estar lista, entre 1 hora 30 minutos y 2. Además se debía estar pendiente del proceso pues en ocasiones la impresora sufría imperfecciones que comprometían la correcta producción de la pieza en cuestión.

Una vez imprimidas todas las piezas de la palma de la mano, se procedió a ensamblarla de acuerdo a la Figura 4.2. Para esto se utilizaron limas para suavizar las conexiones móviles. El resultado de este ensamblado se muestra en la Figura 4.7, junto con la muñeca que actúa de soporte. Como se observa, las piezas de los dedos no están presentes puesto que éstos deben conectarse a los "tendones accionados por los servomotores, lo cual se realizará más adelante.

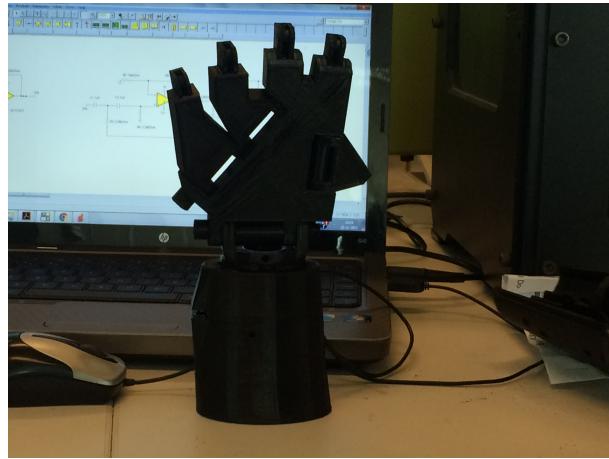


Figura 4.7: Palma y muñeca de la mano ensamblados

En el caso de los dedos, éstos se ensamblaron utilizando pegamento epoxy como unión de piezas, y alambre galvanizado para los grados de libertad de las articulaciones. En la Figura 4.8 se muestra esta construcción. Para probar el correcto funcionamiento del movimiento de los dedos se conectó un servomotor y se hizo girar, de tal forma de poder observar la contracción (Figura 4.8(a)) y la extensión (Figura 4.8(b)).

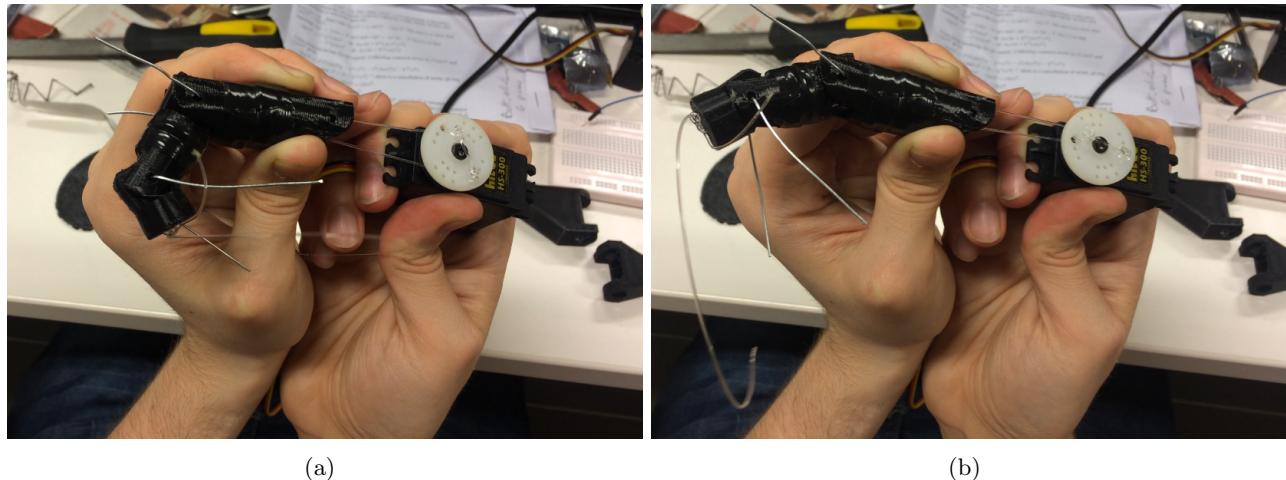


Figura 4.8: Movimiento de un dedo accionado por un servomotor

4.2.2. Modelo del Brazo y la Muñeca

Las piezas del brazo se unieron utilizando pegamento epoxy y tornillos, de acuerdo a la estructura de cada una. Este ensamblado también contempla la inclusión de los servomotores (como se ve en la Figura 4.5). Cabe destacar que se debe ser cuidadoso al momento de integrarlos, pues los espacios son estrechos pues se intenta optimizar el volumen ocupado por el sistema de accionamiento de los tendones. En la figura 4.9 se muestra el resultado del ensamblaje, con 4 servomotores.



Figura 4.9: Brazo y muñeca ensamblados

Destacar que los cables de todos los servomotores convergen a una misma salida del brazo, lo que significa que para controlar los movimientos de la mano basta con conectarse por ese extremo solamente, sin la necesidad de comprometer sectores más cercanos a la mano.

4.2.3. Modelo completo

Para unir los modelos del brazo, muñeca y mano se debe utilizar pegamento epoxy, tornillos y limas de acuerdo a la forma de ensamblado de cada pieza. En particular la mano y la muñeca se conectan con un perno impreso en 3D, mientras que la muñeca con el brazo lo hacen con pegamento epoxy y tornillos.

La integración de los cables que actúan como tendones debe ser cuidadosa, pues para generar el correcto movimiento de los dedos de acuerdo al giro de los servomotores los cables tienen que permanecer tensos sin interactuar con los demás.

Una vez ensamblado todo el sistema que comprende al movimiento controlado junto a sus actuadores (Figura 4.10), resta solamente conectar los servomotores al controlador Arduino para realizar los movimientos que están asociados al clasificador diseñado en la sección 3.6.



Figura 4.10: Sistema completo ensamblado

Capítulo 5

Evaluación económica

Considerando los principales componentes que forman parte del sistema, se obtuvo la lista de precios del cuadro 5.1: Se observa que la parte electrónica (componentes electrónicos y Arduino) contempla un 22% del total de costos, mientras que los materiales que componen la parte actuadora de la mano se acercan al 50%.

Cabe destacar que los precios están calculados de acuerdo al costo unitario de cada producto, cotizados la mayor parte en Olimex [10]. Es por esto que el costo total de la construcción del producto podría verse disminuido en gran cantidad. Incluso materiales como los electrodos o el filamento PLA son reutilizables para nuevas implementaciones del prototipo propuesto en este informe.

Con respecto a las horas hombre trabajadas, se debe considerar que gran parte del tiempo consiste en imprimir las piezas de la mano, lo que se podría paralelizar con los demás procesos correspondientes a los distintos bloques del prototipo, en donde incluso se podría automatizar parte de las construcciones, como lo es la placa PCB.

Cuadro 5.1: Tabla de costos estimados de producción

Elemento de producción	Costo estimado
Electrodos x 50	\$6,200
Placa PCB y componentes	\$10,000
Arduino Due	\$20,000
Servomotores x 5	\$35,000
Filamento PLA	\$30,000
Baterías recargables x 2	\$30,000
Varios	\$2,000
Total	\$136,200
Horas Hombre	340 horas

Capítulo 6

Conclusiones y Trabajo Futuro

Es factible la construcción de un circuito de amplificación y filtrado de señales de baja amplitud, del orden de los mV, tal como se logró verificar por medio de las simulaciones. Además la construcción de este dispositivo considera elementos de bajo costo. Esto se reafirma en la práctica al momento de construir el circuito, el que logró resultados esperados, entregando una señal discriminatoria en presencia de contracciones musculares.

Mencionar que las ganancias de los distintos bloques son ajustados en el laboratorio y los valores de los parámetros antes expuestos varían dependiendo de la señal de entrada real, por lo que se deben calibrar utilizando los potenciómetros del circuito. Esto es necesario sobre todo para el filtro rechaza banda de los 50 [Hz], con el fin de encontrar la ganancia mínima para dicha frecuencia.

El circuito construido es sensible al ruido de la red, comportándose distinto dependiendo del nivel de perturbaciones presentes, esto afecta en alguna medida al clasificador. Lo anterior se apreció al trabajar con el circuito conectado a la red, para alimentar los Opamps y el Arduino conectado a un PC. Se obtuvieron distintos resultados de clasificación probando en el laboratorio de electrónica y en el laboratorio de energía, ambos ubicados en el edificio de electrotecnologías.

Se realizó también una implementación con el circuito operando con baterías y el Arduino conectado a un PC para su alimentación, en este caso se debe tener en cuenta, que producto de la descarga de las baterías, el nivel de amplificación disminuirá, por lo que, las características podrían resultar ser muy distintas a las usadas para entrenar, esto luego de un tiempo prolongado de uso, por lo cual el clasificador disminuirá su rendimiento. Una solución a esto es agregar un sensor que mida el nivel de carga de las baterías y definir un rango de tensión, en el cual el clasificador opera adecuadamente, al salirse de este intervalo, las baterías se deben cambiar.

Para disminuir el tiempo de respuesta del Arduino Due, el clasificador debe entrenarse offline en un PC, para luego utilizar los vectores de soporte en la implementación de SVM en el Arduino. Otro punto a destacar es que entrenando con un solo canal se obtuvieron errores de clasificación que oscilaron entre 2 y un 7 %, salvo para la clasificación entre sostener peso y apretar puño, en el cual se obtuvo hasta un 20 % de error en algunos casos.

Para mejorar el sistema existen varias formas, una es considerar la utilización de Opamp INA 121 que

tienen como aplicación la amplificación de señales biológicas, además se podría considerar la utilización de la impresión de PCB de forma profesional y no hecha de forma casera, ya que, se producen irregularidades en las pistas de cobre. Se podría también realizar la etapa de adquisición y conversión de análogo a digital de la señal, con un dispositivo ADS1208, con lo cual no habría necesidad de construir el circuito amplificador implementado en este trabajo.

Para mejorar la mano se puede considerar imprimirla con material ABS, que es el que utiliza para hacer las piezas de legos, con este se obtienen mejores detalles de las terminaciones. Otra mejora se puede plantear en la construcción de la palma de la mano, la cual en el modelo actual, no posee flexibilidad y limita la prolabilidad de los movimientos.

Hoy en día el estado del arte en la construcción de prótesis biónicas, se está encargando de desarrollar manos con sensores capaces de hacer sentir a las personas lo que la mano biónica está palpando. Esto resultaría en una importante mejora al trabajo, para lograr esto, se necesitaría de un equipo multidisciplinario, tal que se logre transmitir las sensaciones al sistema nervioso de las personas.

Bibliografía

- [1] Mika, S., Schäfer, C., Laskov, P., Tax, D., and Müller, K.-R. Handbook of Computational Statistics. Springer, 2004, ch. Support Vector Machines, pp. 841-876.
- [2] Sapsanis, C.; Georgoulas, G.; Tzes, A., EMG based classification of basic hand movements based on time-frequency features,in Control & Automation (MED), 2013 21st Mediterranean Conference, pp.716-722, 25-28 June 2013
- [3] Rahman K K, M.; Nasor, M.,Multipurpose Low Cost Bio-Daq System for Real Time Biomedical Applications, 2015 International Conference on Information and Communication Technology Research (ICTRC2015)
- [4] P. Duhamel and H. Hollmann, "Split-radix FFT algorithm", Electron. Lett. 20 (1), 14–16 (1984).
- [5] Support Vector and Kernel Machines. Nello Cristianni.
- [6] Haykin,S.(2009).Neuronal Networks and Learning Machines.
- [7] [http://www.3msalud.cl/enfermeria/soluciones-productos/
electrodo-red-dot-multiuso-resistente-a-fluidos](http://www.3msalud.cl/enfermeria/soluciones-productos/electrodo-red-dot-multiuso-resistente-a-fluidos)
- [8] Arduino. [Online]. Disponible en: <https://www.arduino.cc/en/Reference/AnalogRead>
- [9] <https://coolarduino.wordpress.com/2014/09/25/splitradixreal-fft-library>
- [10] MCI Ltda. Olimex Chile. [Online]. Disponible en: <http://www.olimex.cl/>

Capítulo 7

Anexos

Código fuente 7.1: Cálculo Amplitud de Willison

```
1 function [ nwa ] =wa( simfs )
2 %Calcula Willison Amplitude (WAMP)
3 nwillia=0;
4 fc=size(simfs);
5 nwa=zeros(fc(1),1);
6 for i=1:fc(1)
7     m=abs( diff(simfs(i,:)));
8     for p=1:length(m);
9         if m(p)>0.1
10             nwillia=nwillia+1;
11         end
12     end
13     nwa(i,:)=nwillia;
14     nwillia=0;
15 end
16 end
```

Código fuente 7.2: Cáculo Waferom Length

```
1 function [ nwlen ] = wlen( simfs )
2 % Calcula el Waveform Length
3 fc=size(simfs);
4 nwlen=zeros(fc(1),1);
5 for i=1:fc(1)
6     m=diff(simfs(i,:));
7     nwlen(i,:)=sum(abs(m)) ;
8 end
9 end
```

Código fuente 7.3: Cálculo slope sign change

```
1 function [ nssc ] = ssc(simfs)
2 count=0;
```

```
3 fc=size(simfs);
4 nssc=zeros(fc(1),1);
5 for i=1:fc(1)
6     m=diff(simfs(i,:));
7     for p=1:length(m)-1
8         if (m(p+1)>0 && m(p)<0) || (m(p+1)<0 && m(p)>0)
9             count=count+1;
10        end
11    end
12    nssc(i,:)=count;
13    count=0;
14 end
15 end
```

Código fuente 7.4: Cálculo energía

```
1 function [ tenergy ] =energy( simfs )
2 fc=size(simfs);
3 tenergy=zeros(fc(1),1);
4 for i=1:fc(1)
5     mm=abs(simfs(i,:));
6     tenergy(i,:)=sum(mm.^2);
7 end
8 end
```