UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

DISEÑO DE UN SISTEMA EMBEBIDO PARA EL PROCESAMIENTO DE SEÑALES EEG Y EMG PARA APLICACIONES DE hBMI

POR

ING. ALDEBARAN ALFONSO ALONSO CARREÓN

EN OPCIÓN AL GRADO DE MAESTRÍA EN CIENCIAS
DE LA INGENIERÍA ELÉCTRICA

DICIEMBRE, 2016

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA
SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO

DISEÑO DE UN SISTEMA EMBEBIDO PARA EL PROCESAMIENTO DE SEÑALES EEG Y EMG PARA APLICACIONES DE hBMI

POR

ING. ALDEBARAN ALFONSO ALONSO CARREÓN

EN OPCIÓN AL GRADO DE MAESTRÍA EN CIENCIAS
DE LA INGENIERÍA ELÉCTRICA

SAN NICOLÁS DE LOS GARZA, NUEVO LEÓN, MÉXICO DICIEMBRE 2016

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis «Diseño de un sistema embebido para el procesamiento de señales EEG y EMG para aplicaciones de hBMI», realizada por el alumno Ing. Aldebaran Alfonso Alonso Carreón, con número de matrícula 1441339, sea aceptada para su defensa como opción al grado de Maestría en Ciencias en Ingeniería Eléctrica.

El Comité de Tesis

Dr. Miguel Angel Platas Garza
Asesor

Dra. Griselda Quiroz Compeán
Revisor

Dr. David Gutiérrez Ruiz
Revisor

Vo. Bo.

Dr. Simón Martínez Martínez
Subdirector de Estudios de Posgrado

San Nicolás de los Garza, Nuevo León, Diciembre 2016

Agradecimientos

Quiero agradecer primeramente a mi asesoror el Dr. Miguel Angel Platas por todo su paciencia, enseñanzas, conocimientos y guía durante el desarrollo de esta investigación. Así como a la Dra. Griselda Quiroz Compéan por su conocimiento y aportaciones como co-asesora de tesis, también agradezco a todo el grupo de trabajo del área de Mecatrónica del Posgrado de Ingeniería Eléctrica.

Agradezco a la Facultad de Ingeniería Mecánica y Eléctrica (FIME), la Universidad Autónoma de Nuevo León, al Consejo Nacional de Ciencia y Tecnología (CONACYT) por otorgarme la beca número 390751 y los medios para realizar esta investigación, y al Programa para el Desarrollo Profesional Docente (PRODEP), para el Tipo Superior por la ayuda financiera para asistir y exponer parte del presente trabajo en el 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society llevado a cabo en Orlando, Florida, USA del 16 al 20 de agosto del 2016, a través del proyecto 103.5/15/14156.

También deseo agradecer a Miguel Tovar, Carlos Arvizu, Victor Medrano, David Castillo, Antonio Zalapa, Luis Mercado y a los demás compañeros y amigos de generación quienes no podran negar que nos divertimos mucho durante esta etapa.

Finalmente agradezco a mis padres Luis Alfonso Alonso Estrada y Alma Mireya Carreón Martinez que me apoyaron en todo momento con su cariño incondicional, sus enseñanzas de vida y orientación a lo largo de mi vida. También a mis hermanas, Alma Emirette por prestarme su computarora en tiempos de crisis y Arianne Ayn que siempre me ha demostrado su apoyado.

A mi familia y a mis seres queridos...

Índice general

Resumen

Ing. Aldebaran Alfonso Alonso Carreón.
Candidato para el grado de Maestro en Ciencias en Ingeniería Eléctrica.

Universidad Autónoma de Nuevo León.
Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio:

Diseño de un sistema embebido para el procesamiento de señales EEG y EMG para aplicaciones de hBMI

Número de páginas: 158.

Resumen

En los últimos años, la combinación de tecnologías de control mioeléctrico e interfaces cerebro-máquina (Brain Machine Interface, BMI) en un número creciente de estudios relacionados con el control de prótesis, exoesqueletos, o cualquier dispositivo ajeno al cuerpo humano, han hecho avances significativos en el campo de la rehabilitación de la locomoción humana, esto con el fin de proporcionar los medios suficientes para restaurar la función motriz de algún paciente. Sin embargo, para realizar dicha tarea se requiere de un sistema que sea capaz de estimar una variable cinemática partiendo de señales fisiológicas.

Dicho sistema necesita de un modelo matemático que relacione la variable cinemática con la señal fisiológica, sin embargo debido a la complejidad del modelo matemático, se planteó dividir el problema en dos casos de estudio diferentes, proponiendo una etapa de clasificación para determinar el punto de operación del paciente, y un decodificador que estima la variable cinemática (según el punto de operación del paciente).

En esta tesis se abordan los problemas de pre-procesamiento, clasificación y decodificación de señales EEG y EMG respectivamente, se propone un esquema de clasificador/decodificador para estimar una variable cinemática durante la locomoción humana, se realiza un estudio comparativo de diversos algoritmos para cada etapa del esquema propuesto. Se presentan simulaciones, comparaciones e implementación de los algoritmos descritos en la literatura revisada, utilizando "hardware in the loop"para probar los esquemas propuestos en esta investigación.

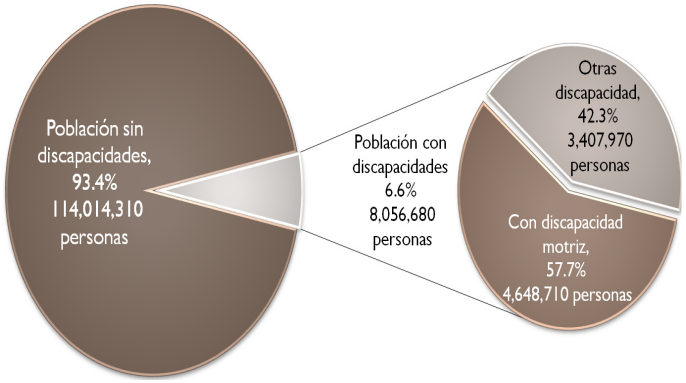
Firma del asesor: _____

Dr. Miguel Angel Platas Garza

Capítulo 1

Introducción

Tomando en cuenta estadísticas como las que se muestran en la Figura 1.1 [1–3], existe un número creciente de personas con discapacidades motrices, lo anterior ha llevado al desarrollo de líneas de investigación centradas en el desarrollo de interfaces cerebro computadora, interfaces cerebro maquina o enfoques híbridos de los mismos.



(a)

País	Año	Población con discapacidad motriz
E.U.A	2010	30.6 millones
Unión Europea	2011	11.8 millones
México	2012	2.5 millones de hombres 3 millones de mujeres

(b)

Figura 1.1: Datos estadísticos de ENSANUT(a,b), U.S Census Bureau, Eurostat (b).

Las Interfaces Cerebro-Computadora (BCI, por sus siglas en Inglés) o las Interfaces Cerebro-Máquina (BMI, por sus siglas en Inglés) son sistemas que usan las señales fisiológicas de un usuario (generalmente discapacitado) con el fin de proveerle un canal alternativo de comunicación con algún otro dispositivo externo y con esto mejorar la calidad de vida de dicho usuario [4].

La clasificación de estos sistemas puede ser según el objetivo con el que se desee procesar la señal; si se emplea para el procesamiento, análisis o diagnostico por computadora se le denomina BCI, o por el contrario, si se usa para la manipulación de un robot, prótesis, ortesis, exoesqueletos o algún otro actuador se conoce como BMI. También pueden ser clasificadas según la señal con la que se desee procesar, por lo que se le conoce como interfaz hibrida [5] (hBCI o hBMI) al sistema que procese alguna otra señal fisiológica además de la cerebral.

La arquitectura de estos sistemas puede variar según sea el objetivo del mismo, sin embargo las etapas que comúnmente conforman la arquitectura de una BCI (o BMI) se muestran en la Figura 1.2.

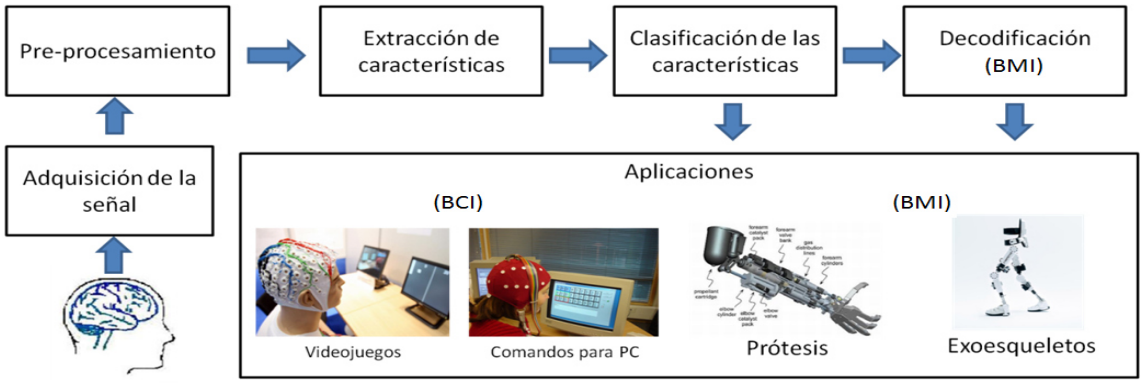


Figura 1.2: Arquitectura típica para una BCI o una BMI.

A continuación se da una breve descripción de cada una de las etapas fundmentales, así como algunas metodologías que se utilizan en los sistemas BCI's para efectuar su tarea.

1. **Adquisición de la señal**: En esta etapa se adquiere y discretiza la señal que posteriormente se procesará. Existen diversos métodos para la adquisición, los cuales son clasificados como invasivos y no invasivos; los invasivos se caracterizan por adquirir la señal directamente de la corteza cerebral como el electrocorticograma (ECoG), mientras que los no invasivos adquieren la señal de manera indirecta, tal como el encefalograma (EEG), que adquiere la señal

a través de electrodos colocados en posiciones especificas del cuero cabelludo. Algunos ejemplos se enlistan en la Figura 1.3.

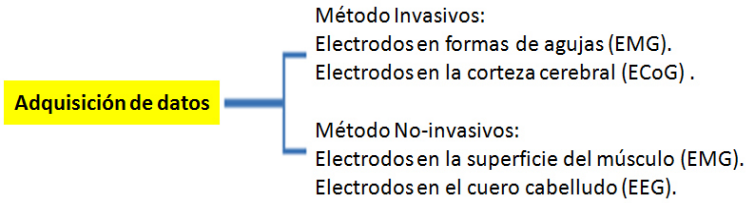


Figura 1.3: Métodos para la adquisición de la señal deseada [4, 5].

2. **Pre-procesamiento**: En esta etapa se busca acondicionar las señales de entrada con el fin de tener un mejor desempeño para las etapas posteriores del procesamiento, debido a que las señales fisiológicas suelen ser susceptibles a ruido involuntario como el parpadeo, la respiración o cualquier otro ruido interno o externo que afecte a la salida del procesamiento. En esta parte se aplican algoritmos para separación y reducción de los efectos de los artefactos (señales parásitas para nuestros fines) de la señal origen. Este ruido generalmente es despreciado usando técnicas enlistadas en la Figura 1.4.

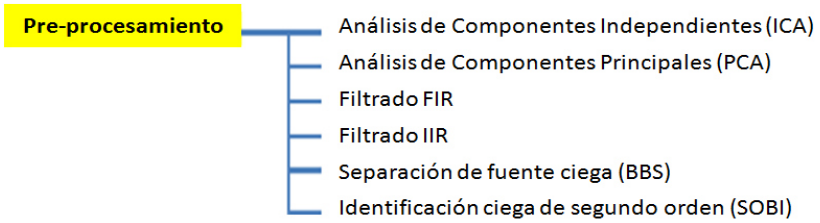


Figura 1.4: Métodos para el pre-procesamiento de una señal [4–8].

3. **Extracción de características**: Esta etapa se encarga de obtener la información más relevante de las señales, dicha información es resumida en un vector de características. Estas características pueden obtenerse por las técnicas descritas en la Figura 1.5.

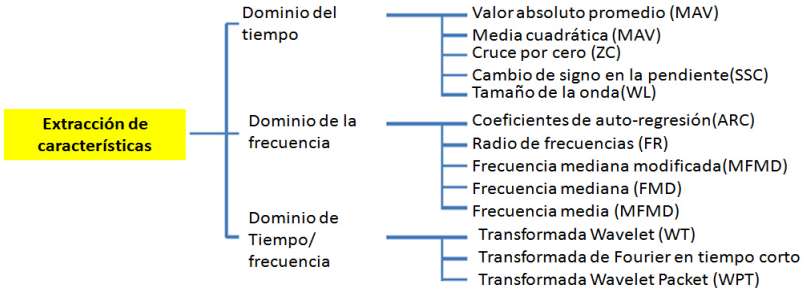


Figura 1.5: Métodos para la extracción de características de una señal [4–6, 9].

4. **Clasificación**: Esta etapa es la que se encarga de buscar una relación entre el vector de características con un cierto criterio de clasificación y que derive una señal de control. Algunas categorías de clasificadores se muestran en la Figura 1.6.

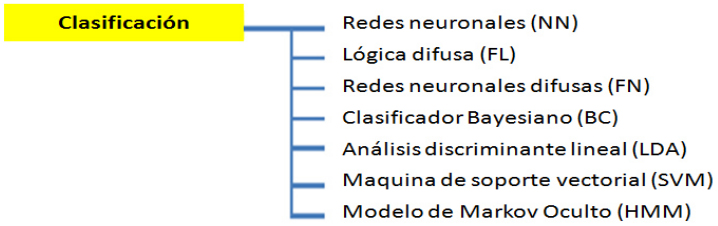


Figura 1.6: Métodos para la clasificación de las características de una señal [4–6, 9].

5. **Decodificación**: Las BMI son las que requieren de esta etapa, debido a que se busca es establecer la relación entre la señal de entrada y la salida de control para el dispositivo final, en esta etapa se involucran algoritmos como los enlistados en la Figura 1.7.

Figura 1.7: Métodos para la decodifición de la señal deseada [10–14].

Este capítulo está dedicado a los preliminares de este trabajo, mostrando las partes

fundamentales que conforman las BCI y BMI; el resto del capítulo está distribuido de la siguiente manera: En la Sección 1.1 se explican las razones por las cuales estas tecnologías han tenido un gran auge en los últimos años. En 1.2 se muestran algunos de los avances que se han realizado en esta línea de investigación. En 1.3 se resumen las áreas de oportunidad de las BMI actuales. En 1.4 se plantea la idea con la cual se piensa abordar el problema a resolver. En 1.5 se plantean los alcances de este trabajo mediante la definición del objetivo general y los particulares. En 1.6 se describe la metodología con la que se trabajo a lo largo del estudio. En 1.7 se describen las contribuciones y en 1.8 se describe el orden de los capítulos de esta tesis.

1.1 Motivación

Iniciando con una perspectiva nacional, los resultados del Censo de México en el 2010 relacionados al porcentaje de las causas por la cual existe el padecimiento motriz en la población con discapacidades se muestran en la Figura 1.8; mientras que de acuerdo a la Encuesta Nacional de Salud y Nutrición (ENSANUT) 2012, para la población de seis años o más, el tipo de discapacidad (consecuencia de la deficiencia en la persona afectada) que tuvo la prevalencias más altas fue la dificultad para caminar, moverse, subir o bajar miembros corporales (4.9 % en hombres y 5.8 % en mujeres) tal como se muestra en la Figura 1.1 [3].

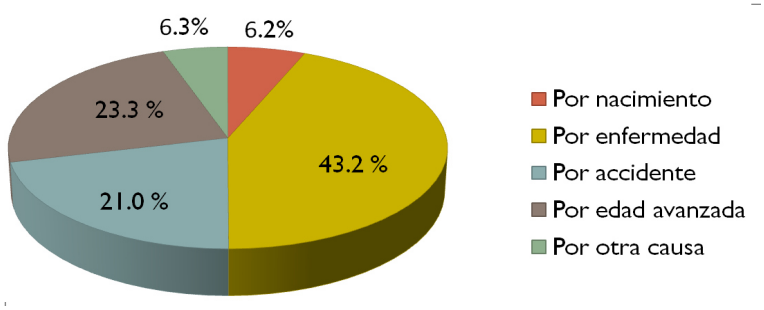


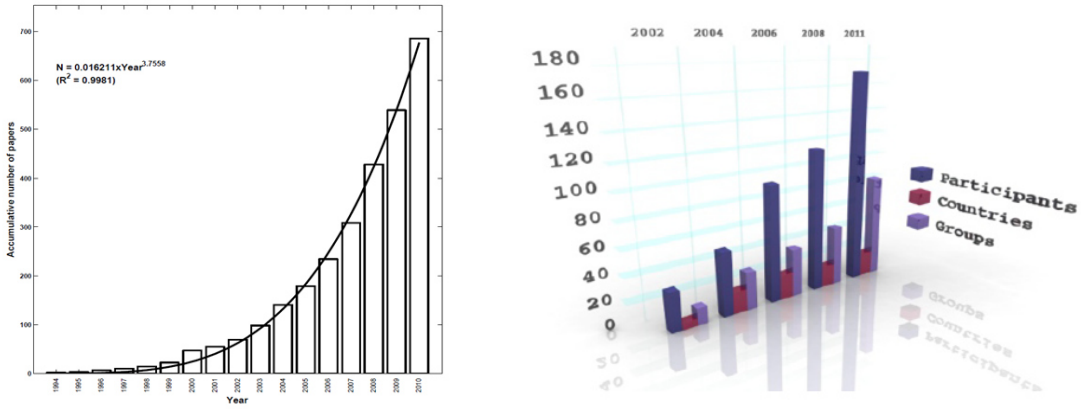
Figura 1.8: Resultados estadísticos de las causas del por las que la población presenta problemas motrices del Censo 2010.

Abordando una perspectiva internacional; en los Estados Unidos durante el 2010, alrededor de 30.6 millones de personas tenían una discapacidad asociada con sus actividades ambulatorias tales como dificultad al caminar, subir o bajar escaleras o tienen la necesidad de usar una silla de ruedas, andador, bastón o muletas. En el 2011, en la Unión Europea, había alrededor de 11.8 millones de personas con discapacidad para caminar [1, 2]. Finalmente, cada año, 15 millones de personas en el mundo sufren un accidente cerebro vascular (ACV). De estos, 5 millones mueren y otros 5 millones quedan permanentemente discapacitados [15].

Dichas estadísticas han impulsado la investigación sobre métodos para permitirle al usuario recuperar sus habilidades motrices o establecer comandos basados en señales fisiológicas, dando como resultado una nueva línea de investigación llamada Interfaz cerebro-computadora (BCI), el cual permite a pacientes con discapaciades motrices, interactuar con el ambiente.

1.2 Antecedentes

El origen de las tecnologías BCI nació con el descubrimiento de la naturaleza eléctrica del cerebro. El término de Interfaz Cerebro-Computadora (BCI) fue primeramente utilizado por Jacques Vidal (1973) [16] mientras medía potenciales eléctricos del cerebro en el cuero cabelludo por medio de un electroencefalograma (EEG). En 1990 surgieron los primeros intentos de utilizar EEG y han continuado desde entonces hasta permitir su uso a un número relativamente grande de pacientes humanos [16]. Debido a la mejora continua en dispositivos de EEG y a computadoras más rápidas, se han ofrecido nueva posibilidades, por lo que el número de grupos de investigación de BCI incrementa cada año, tal como se muestra en la Figura 1.9.



(a) (b)

Figura 1.9: Descripción gráfica de la tendencia en la investigación de BCI. a) Representa de trabajos relacionados por año. b) Representa el incremento en participantes por año [16, 17].

En las últimas décadas las líneas de investigación que involucran las BCIs o BMIs se han diversificado, llegando a cubrir aplicaciones puntuales como en [19], en el que se controla una prótesis con una BCI basada en señales SSVEP (señales neurales que son respuestas naturales de algún estímulo visual), otro caso es el de [18] donde se diseño un sistema embebido para la manipulación de una cama de hospital usando una BCI basada en señales SSVEP.

Entre las investigaciones relacionadas con la implementación de un sistema BCI se destacan: el desarrollo un sistema embebido de una BCI basada en tecnología FPGA (Field Programable Gate Array) para el control de dispositivos caseros [20], el diseño de un sistema BCI para control de una cama de enfermería basada en tecnología FPGA [21], el desarrollo de un sistema confiable BCI para marcar un teléfono móvil [22]; en [23] se describe el diseño de un hardware portable basado en la plataforma CompactRIO en el que se implementó el algoritmo ICA y en [24] se diseño un sistema portable para la adquisición y control de datos ambulatorios basadose en filtros pasivos, amplificadores lineales y amplificadores de instrumentación; para el control, usaron dos microcontroladores H9S12C32 de Motorola.

En la Tabla 1.1 se resumen algunos otros trabajos que están relacionados.

Actividad	Tipo de señal	Posición de los electrodos	Método de extracción de características	Método de clasificación	Precisión en a clasificación
Control del movimiento del cursor para pacientes con ALS.	SCP	Cz (lóbilo central z)	Wavelet de Haar	i)MLP, ii)PNN(RBF), iii)SVM.	i)MLP: 80.87% ii)PNN(RBF): 68.35% iii)SVM: 68.33%
Deletreo de palabras.	i)P300, ii)SSVEP	Fz, Cz, P3, Pz, P4, PO7, Oz y PO8.		LDA	i)P300: 77.27% ii)SSVEP: 69.23%
Control del movimiento de los dedos.	ERP	128 electrodos.	Espectrograma (Power spectral density,PDS).	i)LDA, ii)SVM.	i)LDA: 81.63% ii)SVM: 75.41%
Control del movimiento del cursor.	ERD	C3 y C4.	Transformada rápida de Fourier.	Clasificador de distancia Mahalanobis.	80%
Activación de un robot móvil vía imaginación.	Señales de imaginación	F3, Fz, F4, C3, Cz, C4, P3 y P4.	Transformada Wavelet Packet	i)MMN, ii)HM.	i)MMN: 88.75% ii)HM: 91%
Control de movimiento de una pequeña pelota.	SSVEP	O1 y O2.		Clasificador Fisher	Clasificador Fisher: 80%
Deletreo de palabras.	P300	Fz, Cz, Pz, Oz, C3 y C4.		i)FLDA, ii)LSA, iii)SWLDA.	i)FLDA: 95.75% ii)LSA: 89.45% iii)SWLDA: 90.73%
Movimiento de la mano por medio de la imaginación.	Señales de imaginación	FC4, FC3, C1, C2, C3, C4, C5, C6, CP3, CP4.	Espectrograma (Power spectral density,PDS)	i)MLP, ii)Lógica difusa.	i)MLP: 90% ii)Lógica difusa: 78%
Control del agarre de una mano	Señales de imaginación	F3, F4, Fz, Pz, C3, C4, y Cz.		PNN adaptativo y estadístico.	PNN adaptativo y estadístico: 85.8%

Tabla 1.1: Revisión bibliográfica de algunos trabajos relacionados a BCIs [4].

La mayoría de los trabajos desarrollos en la literatura se basan en pre-procesar y/o clasificar tareas para la generación de comandos y no cuentan con una etapa de decodificación de variables cinemáticas del cuerpo humano [20–24]. Es decir, las investigaciones en BCIs por su lado suelen tener una arquitectura limitada hasta la etapa de clasificación; sin embargo existen trabajos patentados en los que se busca la decodificación de alguna la variable cinemática [25–27]. Asimismo, muchos de los desarrollos que se enfocan en la decodificación y clasificación de miembros superiores exclusivamente [16,19,28], siendo un area de oportunidad el enfoque sobre miembros inferiores.

Los trabajos anteriormente mencionados [11–14,16,24,29], son ejemplos de sistemas en los que utilizan alguna faceta de la señal cerebral unicamente, sin embargo existen trabajos en donde se emplean otro tipo de señales, por ejemplo en [30] se utilizan las señales electromiográficas (EMG) para controlar una ortesis de un miembro inferior humano mediante el uso de una función que relaciona la señal mioeléctrica

y la fuerza requerida por el usuario. En [31], controlan un exosqueleto de la unión del codo con la señal mioeléctrica y un modelo basado en el modelo matemático de Hill.

Sin embargo existen pocos trabajos [18,19] en los que se plantea un proceso de decodificación de variables cinemáticas relacionando las dos señales EEG y EMG. Solamente en [26] se plantea un esquema de implementación en un dispositivo, sin embargo, en este dispositivo analógico no es posible embeber más de una velocidad simultáneamente, ya que por cada velocidad de operación deseada es necesario realizar una etapa de entrenamiento previa por el método de gradiente descendente, requiriendo de una señal cinemática de referencia. En la Tabla 1.1 se resumen otros trabajos relacionados a BMIs en la actividad de locomoción humana.

Autor(es)	Tipo de señal	Método de decodificación	Actividad base	Parámetros decodificados
Pressaco et al. (2012)	EEG	Filtro Wiener	Caminata en una caminadora.	Coordenadas de x, y, z de la cadera, rodilla y tobillo. Ángulos de la cadera y rodilla.
An H et al. (2014)	EEG, EMG	Filtro de Kalman "Unscented" de orden 10	Caminata con un exoesqueleto.	El ángulo de la unión de la cadera, rodilla y tobillo.
Úbeda et al. (2014)	EEG	Regresión lineal	Caminata sobre una caminadora en 3 velocidades distintas.	Ángulo de la unión de la rodilla.

Tabla 1.2: Revisión bibliográfica de algunos trabajos relacionados a la etapa de decodificación de una BMIs [10–14].

Por último, la mayoría de las investigaciones en BMIs (también en BCIs) se realiza en entornos controlados como en clínicas o laboratorios especializados, utilizando electroencefalogramas de grado médico, equipo de grabación, una posición de electrodos estandarizada con una resolución espacial y temporal muy alta y la asistencia de un experto. Otro aspecto técnico importante es la existencia de trabajos en los que se realiza procesamiento en una computadora, lo cual limita la portabilidad del dispositivo para el usuario final [4, 5, 9, 26, 29]. Debido a esto, existen diversos

trabajos que mencionan y trabajan sobre la importancia de abordar el tema de portabilidad y la operación en tiempo real de sistemas BCIs [4], por lo que es necesario desarrollar una plataforma portable, de bajo costo y que permita a los investigadores y usuarios trabajar en línea. Debido a esto se tiene la convicción de que hace falta diseño de un sistema embebido compacto para el procesamiento de señales EEG y/o EMG relacionadas a la locomoción humana.

1.3 Planteamiento del problema

La comunidad científica ha discutido la finalidad de las BCIs, la cual busca brindarle al usuario el mayor apoyo posible, procesando las señales fisiológicas del mismo, sin embargo la mayor parte del procesamiento de las señales fisiológicas de una BCI o BMI actuales se realiza en una computadora (PC), haciendo al usuario dependiente de una PC. Además de esta limitante se encuentran otras como el operar con un solo tipo de señal fisiológica, lo cual vuelve al sistema suceptible a las propiedades de la señal a procesar.

Otro aspecto importante es la etapa de decodificación que necesita de un modelo matemático que relacione la señal fisiológica con la VC (variable cinemática), sin embargo dicho modelo requiere de varios parametros del usuario, los cuales resulta complicado obtenerlos en la práctica [31]. También se han usado métodos para la identificación de parámetros, sin embargo el modelo es limitado fijando un solo punto de operación, causando que baje el desempeño en algún otro punto.

1.4 Hipótesis

Debido a las desventajas identificadas anteriormente se propone que con el uso de un esquema de clasificador/decodificador basado en señales EEG/EMG (Figura 1.10) respectivamente, con el cual se plantea mejorar el proceso de estimación de variables cinemáticas relacionadas a la locomoción humana a distintas velocidades.

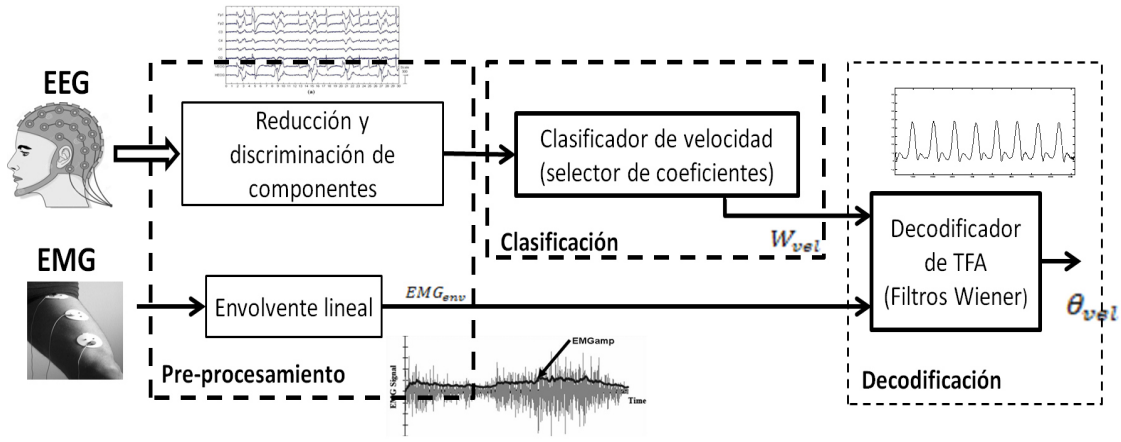


Figura 1.10: Esquema clasificador/decodificador propuesto.

1.5 Objetivos

1.5.1 Objetivo general

Diseñar un dispositivo digital portátil que permita la estimación de los ángulos de las articulaciones durante la locomoción humana, a partir del procesamiento de señales electroencefalográficas y electromiográficas. El propósito de este dispositivo es que sea una interfaz entre las señales fisiológicas del usuario y un dispositivo de asistencia motriz, tal como una prótesis o un exoesqueletos.

1.5.2 Objetivos particulares

1. Decodificar una variable cinemática de un miembro inferior (rodilla).
2. Evaluar y comparar el desempeño de los algoritmos propuestos.
3. Implementar los algoritmos que comforman la hBMI.

1.6 Metodología

1. Revisión bibliográfica general
2. Obtener datos experimentales de señales EMG y EEG durante la caminata a diferente velocidad.
3. Pre-selección de algoritmos a evaluar para cada etapa de la hBMI.
4. Establecer la etapa de pre-procesamiento para las señales EEG y EMG.
5. Diseño y simulación de algoritmos seleccionados.
6. Realizar una serie de simulaciones numéricas para seleccionar los algoritmos de cada etapa a implementar.
7. Selección de Hardware.
8. Implementación del sistema propuesto.
9. Conclusiones del desempeño de la implementación.

1.6.1 Esquema Propuesto

La metodología propuesta para realizar las estimaciones del presente trabajo consta de tres etapas de procesamiento de señal (Figura 1.10): i) Pre-procesamiento de señales EEG y EMG, ii) Clasificación de velocidad de acuerdo a señal EEG y iii) Decodificación de VCs a partir de señal EMG y velocidad clasificada. La estimación de las posiciones angulares de distintas articulaciones relacionadas a miembros inferiores se realiza en línea a partir de un conjunto de señales (EEG, EMG).

1.7 Contribuciones

Con esta tesis nosotros proveemos de procedimientos que en conjunto forman el desarrollo de una hBMI, la contribución del trabajo es el siguiente:

1. Comparación y evaluación de algoritmos para cada etapa de la hBMI.
2. El diseño de un dispositivo para decodificar el ángulo de la rodilla durante la locomoción humana a distintas velocidades.
3. El diseño de una esquema para decodificar una VC en una actividad no definida.

1.8 Organización del trabajo

El contenido de este trabajo está organizado de tal manera que la metodología, esquema propuesto e implementación del sistema sean lo más comprensible posible para el lector. El Capítulo 2 tiene el objetivo de presentar los algoritmos que son incluidos en esta investigación, esto se logra comenzando con la descripción de las propiedades más relevantes (para este trabajo) de las señales a procesar (EMG y EEG); seguido por una descripción más amplia de cada una de las etapas del sistema, estableciendo los algoritmos (provenientes de la literatura revisada) que fueron evaluados. Los métodos de pre-procesamiento para las señal EMG y EEG que aborda este capítulo son: i) Envolvente lineal, ii) Envolvente suavizada por el método de la transformada Hilbert, iii) Análisis de componentes Principales (PCA) y iv) Análisis de componentes independientes (ICA). Para la extracción de características se discute los tipos de buffer que fueron evaluados y el uso de la transformada de Fourier para determinar el vector de características (entrada del clasificador) de la señal EEG. Para el diseño del clasificador se describe los métodos siguientes: i) Clasificador de Bayes (caso lineal) y para abordar la posibilidad de que el problema sea linealmente no separable se consideró: ii) Perceptrón multicapa (MLP) y iii) Máquina de soporte vectorial (SVM). Para el diseño del decodificador se presentan los siguientes algoritmos: i) filtro Wiener, ii) filtro LMS, iii) filtro RLS y iv) Perceptrón Multicapa para abarcar los modelos no lineales. El Capítulo 3 se da a conocer los algoritmos seleccionados, a partir de la simulación de experimentos controlados; el capítulo comienza con la descripción del protocolo de experimentación, seguido por

la simulación numérica de cada uno de los algoritmos propuestos por etapa, continuando con un análisis comparativo de los resultados parciales y se presenta una simulación en Simulink de todo el sistema. El Capítulo 4 está destinado a presentar los resultados; partiendo de la descripción el hardware seleccionado para la fase de implementación y se muestra los paradigmas y resultados de la implementación. El trabajo finaliza con el Capítulo 5, donde se plantean las conclusiones y trabajo a futuro. Además, esta tesis contiene tres anexos, que describen: i) Códigos empleados en esta investigación, ii) Hardware alternativo para la implementación.

Capítulo 2

Métodos propuestos para el sistema hBMI

El objetivo de este capítulo es describir los métodos necesarios para cada etapa del sistema propuesto; por lo cual se inicia con la Sección 2.1 y 2.2; discutiendo brevemente algunas características de las señales fisiológicas (neural, muscular) con las que se trabajaron, asimismo se describen los métodos de adquisición de ambas señales. En la Sección 2.3 se describen los métodos evaluados para la etapa de preprocesamiento de ambas señales, abordando la señal EEG con métodos como PCA, ICA, blanqueamiento y centrado de la señal. Para la señal EMG se propone el cálculo de la envolvente suavizada por el método de la transformada Hilbert y filtrado lineal. En 2.4 se aborda la etapa de decodificación, este capítulo explora métodos adaptativos de filtrado como: i) filtro Wiener, ii) filtro LMS, iii) filtro RLS. Sin embargo con el fin de incluir la posibilidad de que el sistema a identificar sea no lineal, se discute la metodología de redes neuronales artificiales. Finalmente, en 2.5 se plantea el problema de clasificación, se explican las características principales que se tomaron en cuenta para así clasificar la tarea y los métodos que se evaluaron en este trabajo.

2.1 Señales cerebrales

Las señales cerebrales representan la actividad eléctrica neuronal rítmica del sistema nervioso central. El tejido neural puede generar potenciales eléctricos oscilatorios por medio de las neuronas individuales o por las interacciones entre las neuronas. En las neuronas individuales, las oscilaciones pueden aparecer ya sea como oscilaciones en el potencial de membrana o como patrones rítmicos de los potenciales de acción, que a su vez producen la activación oscilatoria de las neuronas post-sinápticas. A nivel de los conjuntos neuronales, la actividad sincronizada de un gran número de neuronas puede dar lugar a oscilaciones macroscópicas, que se pueden observar en el EEG.

Una señal EEG es el registro de la actividad eléctrica cerebral causada por la excitación sináptica de las dendritas de las neuronal piramidales en la corteza cerebral y es captada por electrodos en la superficie de cuero cabelludo [32]. Si bien existen diversos métodos para interpretar la actividad neuronal del cerebro (basados en magnetismo, flujo de sangre del cerebro, entre otros); El electroencefalograma ha sido el enfoque más aceptado para la implementación de sistemas BCI [4,5] debido a la posibilidad de poder desarrollar un sistema completamente portable sin intervenir en la integridad física del usuario.

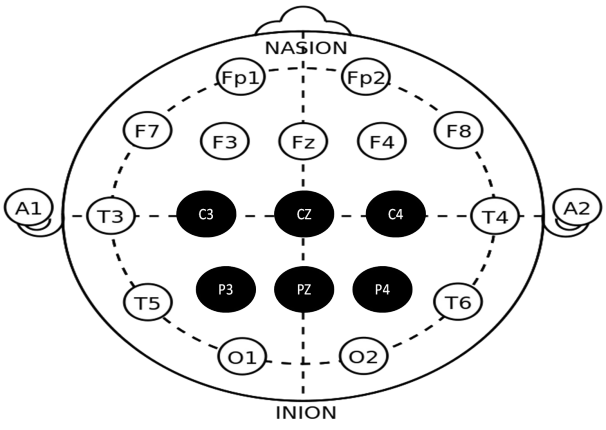


Figura 2.1: Descripción gráfica del sistema internacional 10-20, marcando en color negro la localización de electrodos más relevantes para este trabajo.

Con el fin de optimizar los resultados de este método, la localización de los electrodos ha sido estandarizada por el sistema internacional 10-20, el cual dispone de 19 electrodos, y es usado para la mayoría de las aplicaciones, sin embargo también existe la convención 10-10 y 10-5, que son configuraciones con alta densidad de arreglos de sensores lo cual incrementa el número de electrodos de manera impráctica [33]. Para este trabajo se consideraron los seis electrodos del sistema internacional 10-20, tal como se muestra en la Figura 2.1.

Las ondas cerebrales poseen amplitudes que van desde los $\pm 10\mu V$ hasta $\pm 100\mu V$ sobre la superficie del cuero cabelludo. Estas ondas existen en la banda de frecuencias de 0.5 y 100 Hz y tienen mucha dependencia de la actividad en la corteza cerebral. Generalmente estas ondas no suelen poseer alguna forma característica, sin embargo algunos ritmos suelen clasificarse en ritmos α, β, γ, μ, θ y δ [32,37]. En la Figura 2.2 se muestra las formas generalizadas de las principales ondas de la actividad cerebral.

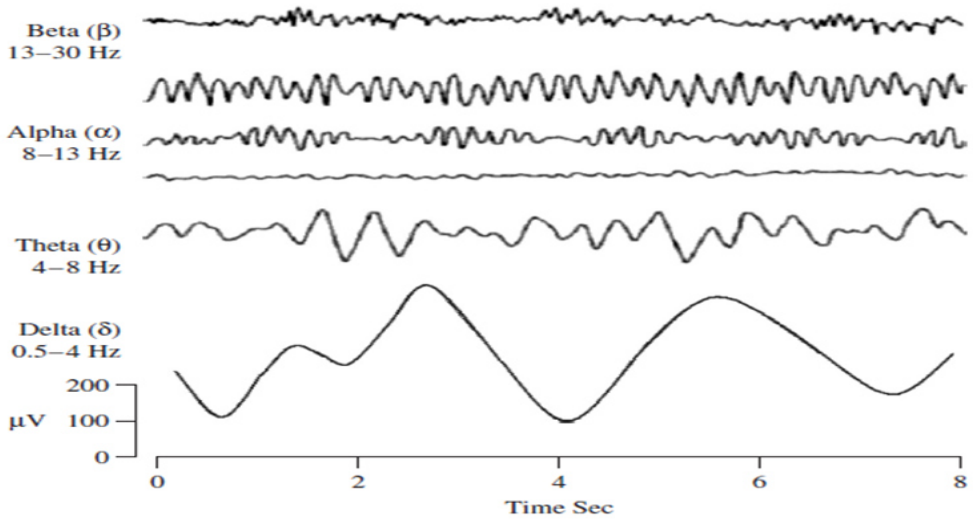


Figura 2.2: Los cuatro ritmos cerebrales normales dominantes, de alta a baja frecuencia [32].

Las ondas α poseen frecuencias de 8 a 13 Hz. Se registran en sujetos normales y despiertos, con conciencia relajada y sin mucha concentración sobre alguna tarea. Estas ondas son las más comunes de todo el conjunto de ondas de la actividad cere-

bral; su amplitud está comprendida entre 10 y $50\mu V$, estas ondas suelen localizarse sobre todo en la zona occipital.

Las ondas β poseen frecuencias de 14 a 30 Hz; se registran usualmente en las zonas parietal y frontal. La actividad de las ondas β está estrechamente vinculada a la conducta motriz, el pensamiento activo, la atención activa, concentración en el mundo exterior o resolución de problemas que requieren de concentración. La amplitud normal de estas ondas esta debajo de 30 μV. Las ondas θ poseen frecuencias de 4 a 7.5 Hz y estás se asocian al subconsciente, inspiración creativa y meditación profunda, también suelen ser más recurrentes en la infancia aunque también pueden presentarse en la etapa adulta en períodos de estrés emocional y frustración. Se localizan en áreas como las zonas parietal y temporal.

Las ondas δ poseen frecuencias inferiores a 4 Hz y se presentan durante el sueño profundo, al despertar y por lo general es la onda que más suele estar contaminada por artefactos de movimiento. Con respecto a las ondas γ suelen existir en rango de frecuencia de 30-100 Hz aproximadamente. Se cree que representan la unión de diferentes conjuntos de neuronas, juntos en una red con el propósito de llevar a cabo una determinada función cognitiva o motriz. Las ondas μ se encuentra sobre la corteza motora, que es una banda de aproximadamente de oreja a oreja. Una persona suprime patrones de ondas μ cuando él o ella realiza una acción motriz o cuando él o ella imagina y realizar una acción motora.

Las señales eléctricas detectadas a lo largo del cuero cabelludo por un EEG que no tienen origen cerebral se denominan artefactos. Las señales adquiridas en un EEG generalmente están contaminadas por tales artefactos. La amplitud de los artefactos puede ser muy grande en relación con el tamaño de la amplitud de las señales de interés. Además, las señales EEG pueden captar señales fisiológicas como el ECG (electrocardiograma), EOG (electrooculograma) y el EMG.

2.2 Señales mioeléctricas

Desde el siglo XVII se sabía de la existencia de la actividad eléctrica en los músculos, siendo un resultado del estudio de los músculo de los peces raya. Sin embargo, en 1890 el doctor francés E. J. Marey, realizó el primer registro de corrientes eléctricas en los músculos. A partir de ese suceso, se establecieron nuevas líneas de investigación enfocadas en el perfeccionamiento de esta técnica, llegando a desarrollar electrodos capaces de captar la actividad eléctrica de célula a célula en una electromiografía.

Básicamente, la electromiografía es una prueba que se usa para analizar el funcionamiento del sistema nervioso periférico y los músculos asociados. La electromiografía se desarrolla utilizando un instrumento médico llamado electromiógrafo, para producir un registro de señales mioeléctricas (señales EMG) mediante electrodos especiales que captan los potenciales eléctricos que activan las células musculares, cuando éstas son activadas neuralmente o eléctricamente para realizar contracciones.

Esta diferencia de potencial de la membrana muscular, generalmente existe alrededor de -80 a 30 mV, mientras que el rango típico de frecuencia de la actividad muscular es de alrededor 10 a 250 Hz dependiendo del tamaño del músculo [34]. Dado que la señal EMG es básicamente una sumatoria algebraica de unidades potenciales de acción del área medida, esta señal de hecho muchas veces es considerada como la mejor candidata para ser una señal de control, debido a que su comportamiento puede ser interpretado como el reflejo directo de un intento de movimiento [30, 31].

Este registro puede obtenerse por métodos invasivos o no invasivos, que corresponden a EMG intramuscular o de superficie, respectivamente. En el caso de una EMG intramuscular, se utilizan electrodos en forma de aguja, los electrodos se inserta en el músculo que se desea estudiar. Al ser un método invasivo la señal obtenida es más limpia, sin embargo un electrodo por si mismo adquiere una perspectiva local de la actividad del músculo, por lo que se requiere de más electrodos en posiciones

distintas, para obtener una perspectiva más cercana a la global de la actividad del músculo, lo que involucra mayor cantidad de electrodos en este método. Por el contrario una EMG superficial tiene una perspectiva local mucho más alta, debido a que se usa un electrodo de superficie (por lo que abarca una mayor superficie del músculo) colocado en la piel del músculo que se desea estudiar, la posición de los electrodos para este trabajo se muestra en la Figura 2.3.

En ambos casos se utilizan circuitos de amplificación y filtrado para adquirir la señal de los electrodos, por lo tanto para este trabajo se toma en cuenta solo el método no invasivo.

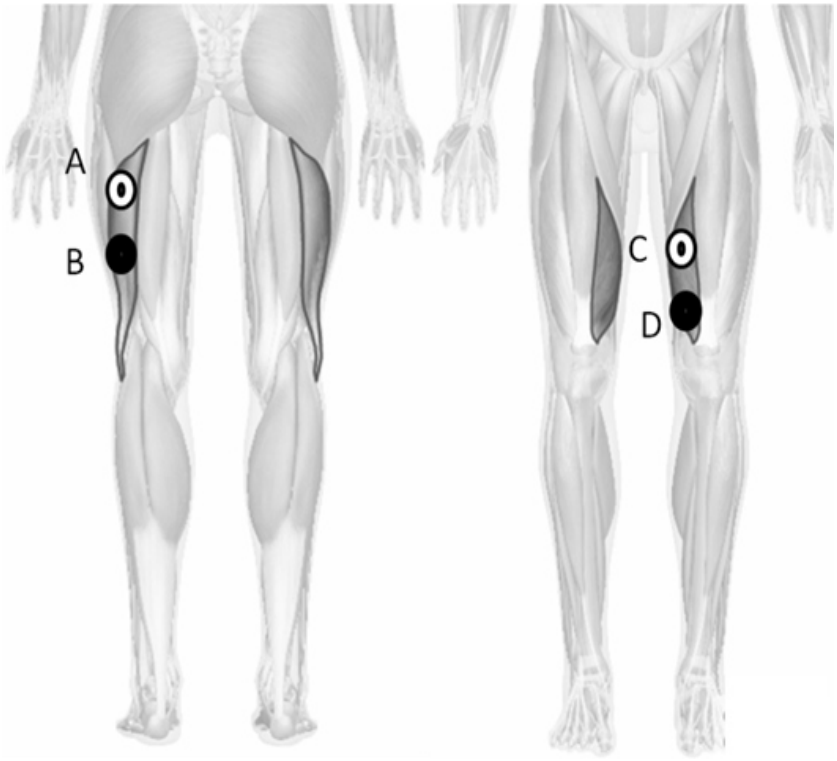


Figura 2.3: A y B son los electrodos positivos y negativos colocados en el *bíceps femoris* (flexor) respectivamente. C y D son los electrodos positivos y negativos colocados en los *vastus medialis* (extensor) respectivamente.

Las señales EMG y EEG son consideradas en naturaleza, como realizaciones de un proceso aleatorio por lo que se puede hacer una aproximación con una función de distribución Gaussiana. Otro aspecto a considerar es la variación de señales

fisiológicas entre diferentes personas, de igual manera estas señales son susceptibles al estrés, cansancio, fatiga muscular, adormecimiento, etc.

2.3 Pre-procesamiento

Tal como se muestra en la Figura 2.4 las señales medidas suelen estar contaminadas. por lo que en esta etapa se busca acondicionar las señales con fin de eliminar componentes no deseados tales como ruido por artefactos y de esta manera incrementar de la relación señal-ruido. La etapa de pre-procesamiento se aplica al conjunto de señales EEG y EMG siguiendo distintos procedimientos que son descritos a continuación.

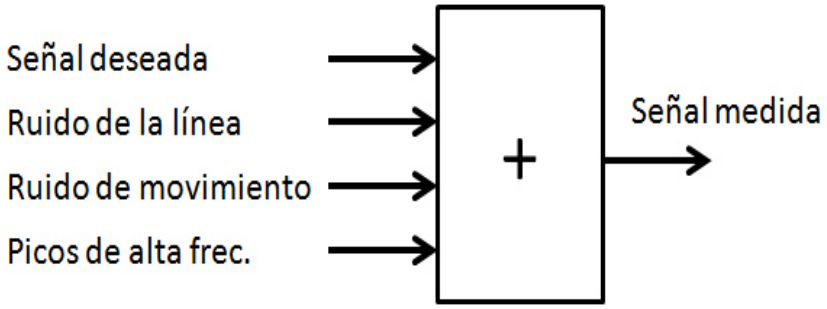


Figura 2.4: Componentes que conforman una señal medida por métodos convencionales.

2.3.1 Pre-procesamiento de señales EMG

El proceso que genera la señal EMG es catalogada como un proceso no estacionario, debido a esto se propone calcular la envolvente suavizada de la señal [30, 34], ya que las propiedades de la señal EMG cruda muestra información más compleja para ser procesada. La envolvente de una señal oscilante es aquella curva suave que delinea los extremos de la señal original, tal como se muestra en la Figura 2.5; para este trabajo se considera la envolvente superior.

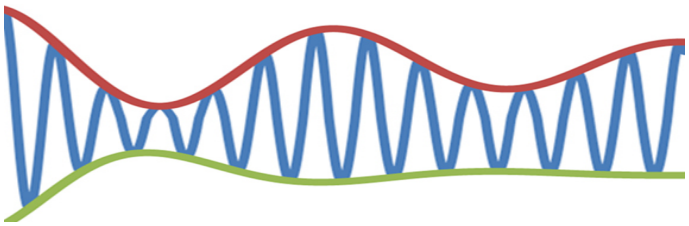


Figura 2.5: Envolvente de la señal: la señal en rojo representa la envolvente superior, mientras que en verde se muestra la envolvente inferior de una señal oscilatoria (azul).

Existen diferentes métodos para el cálculo de la envolvente de una señal, en este trabajo se abarcan dos diferentes métodos: A) Filtrado lineal [34] y B) transformada Hilbert [36]; ambos métodos se describen gráficamente en la Figura 2.6.

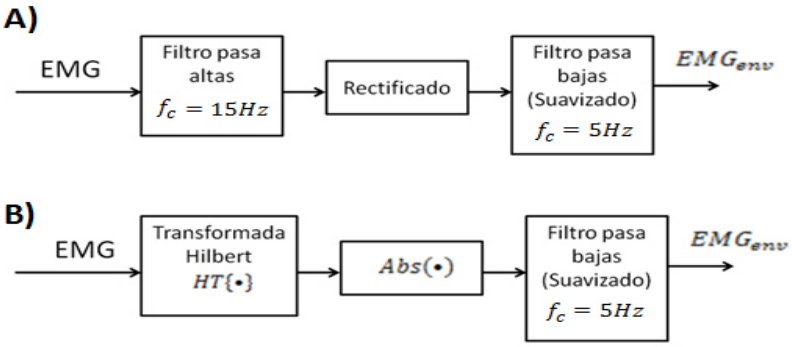


Figura 2.6: Diagrama de bloques de los dos métodos a evaluar para el pre-procesamiento de señales EMG.

A) Filtrado Lineal

- **Filtro butterworth pasa altas**

Los rangos de frecuencias en el que se encuentra la información deseada de una señal EMG son 10-250 Hz [34]. Por lo que las frecuencias ajenas a este rango de frecuencia tienen que ser filtradas, ya que son consideradas como artefactos de la señal EMG. Para la solución de este tipo de problema es posible aplicar un filtro

de Butterworth; el cual es uno de los filtros electrónicos más conocidos, diseñado para producir la respuesta más plana posible hasta llegar a la frecuencia de corte. En otras palabras, la salida se mantiene constante casi hasta la frecuencia de corte, luego disminuye a razón de $20n$ dB por década (ó $6n$ dB por octava), donde n es el número de polos del filtro. El diseño de un filtro Butterworth puede realizarse en base a su función de transferencia [35]; la función de transferencia para un filtro Butterworth pasa bajas de orden n es el siguiente:

$$|A_c(jw)|^2 = \frac{1}{1 + (\frac{j\omega}{j\omega_c})^{2n}} \tag{2.1}$$

Donde ω_c es la frecuencia de corte (rad/s) en el cual la respuesta cae 3 dB, ω es la frecuencia (rad/s). En la Figura 2.7 se muestra la respuesta del filtro al aumentar el orden.

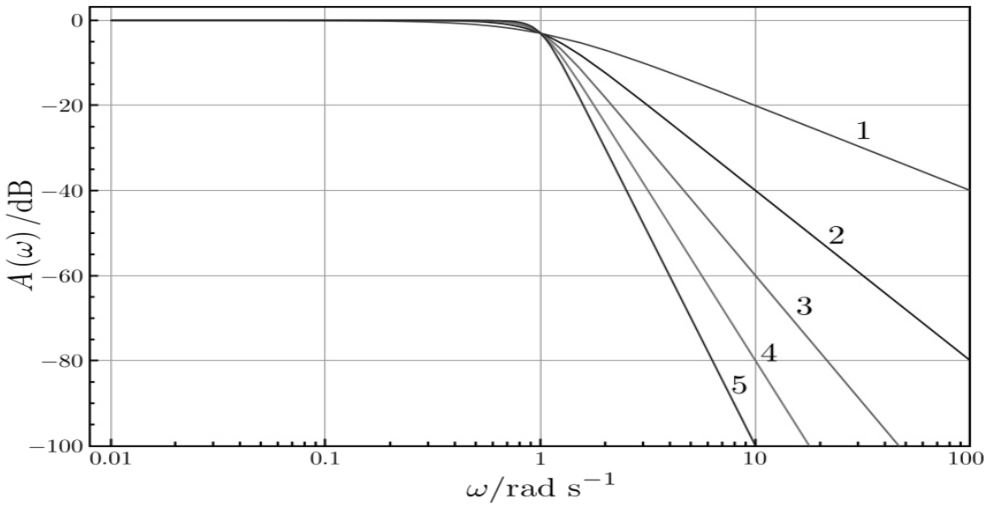


Figura 2.7: Grafica de la ganancia de un filtro pasa baja de orden 1 al 5 y una frecuencia de corte de 1 Hz.

- **Rectificado**

Con el propósito de obtener la envolvente superior, se requiere que la señal EMG sea completamente positiva, es decir la señal tiene que ser rectificada, existen dos tipos de rectificación: i) rectificación de media onda y ii) rectificación de onda

completa. En la rectificación de media onda, se elimina la parte negativa de la señal, mientras que la rectificación de onda completa cambia los valores negativos de la señal a positivos. La operación de valor absoluto satisface la rectificación de onda completa por lo que se conserva la información de la señal completa.

- **Suavizado**

La energía del músculo puede ser cuantificada con el proceso de suavizado, dicho proceso puede ser realizado mediante los siguientes métodos [34, 37]:

Media Móvil (MA)

$$MA_t = \frac{1}{N} \sum_{i=t-N+1}^{t} x_i \tag{2.2}$$

Valor absoluto media (MAV)

$$MAV_t = \frac{1}{N} \sum_{i=t-N+1}^{t} |x_i| \tag{2.3}$$

Media cuadrática (RMS)

$$RMS_t = \sqrt{\frac{1}{N} \sum_{i=t-N+1}^{t} x_i^2} \tag{2.4}$$

Donde t es el tiempo inicial, x_i es la i-ésima muestra de la señal que se pretende suavizar y N es el tamaño de la ventana. Los métodos MAV y RMS no requieren la etapa previa de rectificado, debido que utilizan operadores que transforman la señal a positiva.

Media aritmética por medio de un filtro pasa bajas.

Una señal cualquiera se puede expresar siempre como $s(t) = s_{dc} + u_{ac}(t)$, donde $s_{dc}(t)$ es la componente continua y $u_{ac}(t)$ la componente alterna. Si se calcula el promedio aritmetico de $s(t)$ se obtiene: $E\{S(t)\} = S_{dc} + E\{U_{ac}(t)\}$. Con la particularidad que al ser $U_{ac}(t)$ la componente alterna, su promedio tendiende a 0 a medida que se aumenta el numero de muestras o a medida que su frecuencia es más alta (pero siempre menor que la frecuencia de Nyquist).

Debido a que la respuesta al impulso de un filtro de media móvil corresponde a un filtro pasa bajas; el usar un filtro pasabajas también suaviza la señal. Para este trabajo el proceso de suavizado se realiza utilizando un filtro Butterworth pasa bajas.

B) Envolvente suavizada por el método de la transformada Hilbert

Para hallar la envolvente de la señal analítica se precisa el uso de la transformada de Hilbert($\mathcal{H}\{\cdot\}$) [38, 39]. La transformada de Hilbert se define como la convolución de una señal $x(t)$ con la función $\frac{1}{\pi t}$:

$$\mathcal{H}\{x(t)\} = x(t) * \frac{1}{\pi t} = \frac{1}{\pi}\int_{-\infty}^{+\infty} \frac{x(\tau)}{t-\tau}\,\mathrm{d}\tau \tag{2.5}$$

La operación de convolución puede realizarse en el dominio de la frecuencia como una simple multiplicación, por lo tanto se puede calcular la transformada de Hilbert de la siguiente manera:

$$\mathcal{F}\{\frac{1}{\pi t}\} = -j\mathrm{sgn}(\omega), \qquad \mathcal{F}\{x(t)\} = X(\omega),$$

$$\mathcal{F}\{\mathcal{H}\{x(t)\}\} = -j\mathrm{sgn}(\omega)X(\omega) \tag{2.6}$$

En la Figura 2.8 se muestran la forma del módulo y de la fase de la función de transferencia de la transformada Hilbert.

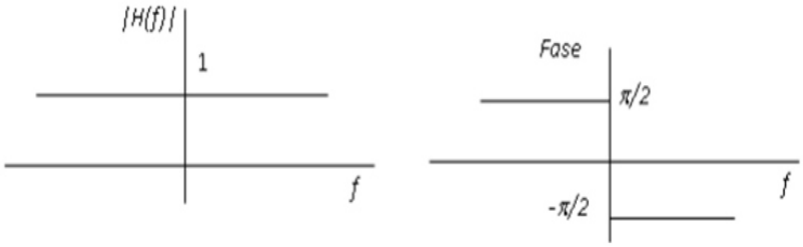


Figura 2.8: Magnitud y fase de la transformada de Hilbert.

Realizar dicha transformación a una señal, implica que todas las componentes positivas de la señal se retrasan $\frac{\pi}{2}$, mientras que las componentes negativas se adelantan $\frac{\pi}{2}$.

Dada estas propiedades y que cualquier función se puede expresar como una suma ponderada de exponenciales complejas, es posible escribir a la función exponencial compleja de la siguiente forma:

$$e^{j\Omega t} = \cos(\Omega t) + j\sin(\Omega t) = \cos(\Omega t) - j\mathcal{H}\{cos(\Omega t)\} \tag{2.7}$$

La ecuación 2.7 se puede expresar como una función analítica:

$$a_x(t) = x(t) - j\mathcal{H}\{x(t)\}, \tag{2.8}$$

calculando el espectro de la señal $a_x(t)$ se obtiene:

$$A_x(\omega) = F\{a_x(t)\} = 2X(\omega)u(\omega), \tag{2.9}$$

con esta ecuación se concluye que el espectro de la señal analítica de una señal $x(t)$ sólo contiene sus frecuencias positivas y la energía de la señal $x(t)$ es igual a la de la señal analítica.

Con lo anterior mencionado, es posible definir la envolvente $e_x(t)$ de una señal real $x(t)$ como la magnitud de la señal analítica $a_x(t)$:

$$e_x(t) = |a_x(t)| = \sqrt{x(t)^2 + \mathcal{H}\{x(t)\}^2} \tag{2.10}$$

En los puntos de intersección de la función y su envolvente, la envolvente $e_x(t)$ será tangente a la función $x(t)$. Por lo que se concluye que la envolvente debe ser mayor o igual que la función en todo punto.

Como la reconstrucción de la señal $x(t)$ se hace a partir de la envolvente, que es la parte de la señal en la que se manifiestan los picos, parece pertinente aplicar el proceso de suavizado anteriormente discutido.

2.3.2 Pre-procesamiento de señales EEG

Para realizar una prueba de EEG generalmente se utiliza el estándar internacional 10-20, el cual involucra alrededor de 19 electrodos, en estas aplicaciones típicas, la cantidad de información adquirida por los electrodos es inmensa, es decir la dimensión del problema a resolver incrementa por cada una de las mediciones de los electrodos, la dimensión de la entrada afecta de manera directa a la carga computacional del sistema embebido [40].

Este problema suele resolverse con la caracterización en la posición donde se adquiere la señal de interés, es decir se tiene el conocimiento de la posición exacta donde la señal de interés es emitida, con ese conocimiento basta con analizar los electrodos más cercanos a dicha posición para obtener un buen resultado, sin embargo en la actualidad las señales cerebrales no se han caracterizado por completo, causando que en algunas ocasiones ésta aproximación pueda no ser válida, por otro lado existen métodos para la reducción de la dimensión basados en la identificación de componentes principales.

Análisis de componentes principales

El análisis de componentes principales (PCA) es una técnica utilizada para reducir la dimensión de un conjunto de datos [41]. Intuitivamente ésta metodología sirve para encontrar las causas de variación de un cierto conjunto de datos y darles un orden según su importancia. En términos de mínimos cuadrados, el PCA busca la proyección en el que los datos puedan ser representados de la mejor manera. Entre las principales aplicaciones del PCA se encuentra: el análisis exploratorio de datos y la construcción de modelos predictivos. La transformación a componentes principales también se asocia a otra factorización de una matriz, la descomposición de valores singulares (en Inglés, SVD), ya que el ejecutar PCA es equivalente a usar el SVD en una matriz de covarianza $\mathbf{X}$.

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{W}^T \tag{2.11}$$

Donde Σ es una matriz rectangular de $n \times p$, con $n, p \in \Re$, de números positivos $\sigma_{(x)}$ llamados valores singulares de $\mathbf{X}$ en la diagonal; $\mathbf{U}$ es una matriz cuadrada y unitaria de $n \times n$, las columnas de esta matriz son vectores unitarios ortogonales de tamaño n llamados los vectores singulares de izquierda de $\mathbf{X}$; $\mathbf{W}$ es una matriz cuadrada unitaria de $p \times p$, las columnas de esta matriz son vectores unitarios ortogonales de tamaño n llamados los vectores singulares en la derecha de $\mathbf{X}$.

En términos de la factorización, la matriz $\mathbf{X}^T\mathbf{X}$ se puede desarrollar de la siguiente manera:

$$\mathbf{X}^T\mathbf{X} = \mathbf{W}\Sigma\mathbf{U}^T\mathbf{U}\Sigma\mathbf{W}^T = \mathbf{W}\Sigma^2\mathbf{W}^T$$

Comparando con la factorización de eigenvectores de $\mathbf{X}^T\mathbf{X}$ estableciendo que los valores singulares de derecha $\mathbf{W}$ de $\mathbf{X}$ son equivalentes a los eigenvectores de $\mathbf{X}^T\mathbf{X}$, mientras que los valores singulares $\sigma_{(k)}$ de $\mathbf{K}$ son iguales a la raíz cuadrada de los eigenvalores $\lambda_{(k)}$ de $\mathbf{X}^T\mathbf{X}$.

Usando la descomposición de valores singulares la matriz $\mathbf{T}$ se describe como:

$$\mathbf{T} = \mathbf{X}\mathbf{W} = \mathbf{U}\Sigma\mathbf{W}^T\mathbf{W} = \mathbf{U}\Sigma \tag{2.12}$$

De esta manera, las columnas de $\mathbf{T}$ están dadas por uno de los vectores singulares de la izquierda de $\mathbf{X}$ multiplicados por su correspondiente valor singular. Existen algoritmos eficientes para el cálculo de la SVD de $\mathbf{X}$ sin tener que formar la matriz $\mathbf{X}^T\mathbf{X}$, por lo que el cálculo de SVD se ha convertido en una manera estándar de calcular PCA de datos mezclados.

Análisis de componentes independientes (ICA)

Otro problema muy común en las señales EEG es la baja relación entre la señal y el ruido (SNR, signal-noise ratio) y esto se debe a la influencia de los artefactos inmersos en la señal EEG, si bien se podría generalizar este problema con el paradigma de la fiesta de coctel, que consiste en tratar de reconstruir las señales de origen con cierta dimensión, partiendo de señales observadas en distintos puntos y con la

misma dimensión.

En la literatura, este problema suele ser resuelto por la técnica conocida como el Análisis de Componentes Independientes (en Inglés, ICA) [41], método desarrollado para separar, en componentes aditivos, una señal multivariante suponiendo que sus componentes tienen independencia estadística y son no-Gausianos. El algoritmo ICA es similar al PCA, en el sentido de que en ambos casos se realiza una transformación lineal de los datos originales. Éste es un caso especial de separación ciega de las señales, donde se supone que las señales observadas tienen la siguiente estructura:

$$x_j = a_{j1}s_1 + a_{j2}s_2 + \cdots + a_{jn}s_n \qquad \forall j = 1, 2, \ldots, n \tag{2.13}$$

Donde s_n es la n-ésima señal independiente, x_j es la j-ésima señal mezclada con las señales independientes, a_{jn} son los coeficientes de la combinación lineal de las señales independientes.

Denotando 2.13 en forma matricial se obtiene:

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{2.14}$$

$$\mathbf{s} = \mathbf{A}^{-1}\mathbf{x} = \mathbf{W}\mathbf{x}$$

Donde $\mathbf{s}$ es el vector de señales independientes, $\mathbf{x}$ es el vector de señales mezcladas con las señales independientes y $\mathbf{A}$ es la matriz de transformación.

La técnica ICA se basa en la hipótesis que establece que las señales originales son estadísticamente independientes, debido a que los procedimientos que se realizan en el algoritmo, se basan en distribuciones probabilísticas de las señales originales y de las observaciones (mezclas); ya que si las señales originales tienen independencia estadística, las señales recuperadas también deben de serlo. Por lo tanto, la condición más importante para el ICA es la independencia estadística. Ya que se buscará maximizar la independencia de los componentes basandose en la distribución probabilística de las señales mezcladas.

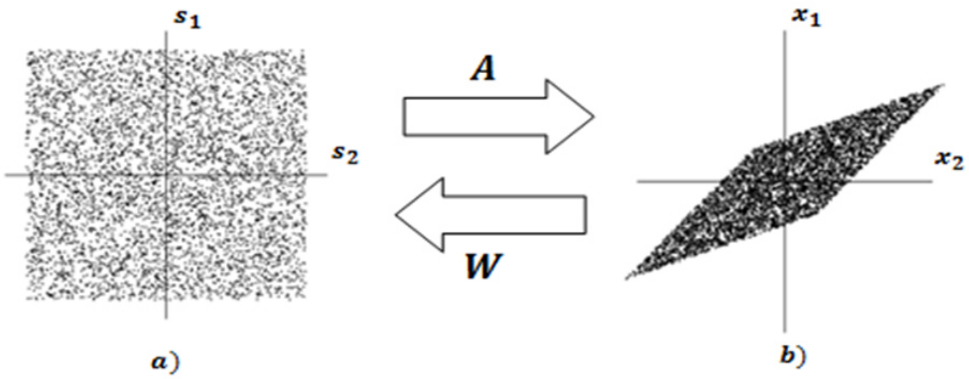


Figura 2.9: a) La distribución conjunta de los componentes independientes s_1 y s_2 (los cuales se modelan como una distribución uniforme). b) La distribución conjunta de los componentes mezclados x_1 y x_2

El problema (Figura 2.9) consiste en encontrar una transformación que encuentre $\mathbf{A}^{-1}$ a partir de la cual se mezclaron los componentes principales, maximizando la no-Gaussianidad.

No-Gaussianidad

En este algoritmo requiere que los componentes independientes no tengan distribución Gaussiana ya que de tener dicha propiedad, hace que la distribución conjunta sea cuasi-simétrica (Figura 2.10) [41]. Por consiguiente no contiene la información necesaria para identificar las columnas de la matriz de mezclado $\mathbf{A}$.

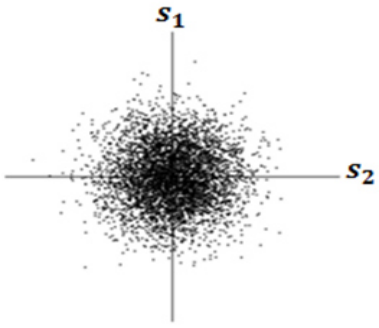


Figura 2.10: La distribución conjunta de los componentes independientes los cuales se modelan como una distribución Gaussiana.

El teorema del límite central, en la teoría estadística, establece que la distribución de la suma de variables aleatorias independientes tiende a semejarse a una distribución Gaussiana en comparación de las señales originales de referencia [41]. Tomando en cuenta que puede existir un conjunto finito de N transformaciones lineales posibles, con diferentes grados de distribución Gaussiana, se puede intuir que existe una transformación lineal con el menor grado de distribución Gaussiana, es decir, se puede utilizar la no gausianidad como una función objetivo a maximizar para encontrar la transformación lineal cuyo grado de distribución Gaussiana sea la menor.

Para usar esta propiedad como función a optimizar es necesario cuantificar la medida de no gausianidad de una variable aleatoria; existen diferentes métodos para medir la no gausianidad entre ellos se encuentra:

- Curtosis
- Negentropia

Ambos métodos presentan ventajas y desventajas, en el caso de la curtosis el principal ventaja de este método es que requiere de pocas muestras, sin embargo, esta característica también puede ser un problema, ya que hace al método susceptible de la valores atípicos o ruido, es decir el método de curtosis no es robusto; el método de negentropía es en cierto sentido el estimador optimo de no Gaussianidad, sin embargo es mucho más complicado el análisis de la misma ya que requiere de una estimación de la función de densidad probabilística (pdf). Debido a esto se pretende utilizar una aproximación a la negentropía ($J(y)$) ya que se usa una aproximación a la pdf; La aproximación clásica es:

$$J(y) \approx \sum_{i=1}^{N} k_i [E\{G_i(y)\} - E\{G_i(v)\}]^2 \tag{2.15}$$

Donde k_i son constantes positivas, v es una variable Gaussiana con media cero y varianza unitaria, por último $G_i(\cdot)$ es una función no cuadrática el cual puede elegirse

según la aplicación, algunas de las siguientes funciones son [29, 41]:

$$G_1(u) = \frac{1}{a_i}\log(\cosh(a_i u)) \tag{2.16}$$

$$G_2(u) = u\exp(\frac{-u^2}{2}) \tag{2.17}$$

$$G_3(u) = \tanh(a_i u) \tag{2.18}$$

Donde a_i es un parametro constante y existe en el intervalo [1-2].

Con fin de de optimizar el algoritmo ICA, se proponen dos etapas que incluyen i) Centrado (media-cero) ii) Blanqueamiento.

Proceso de centrado

El pre-proceso más básico y necesario es el centrar las señales observadas x_i [41], esto se realiza de manera sencilla restado el vector de medias o promedios $\mathbf{m} = E\{\mathbf{x}\}$, haciendo que $\hat{\mathbf{x}}$ sea de media-cero y por ende $\hat{\mathbf{s}}$ también será de media-cero. Una vez que se obtiene un resultado del algoritmo ICA se puede regresar la señal a su forma original sumando el vector $\mathbf{m}$.

Proceso de blanqueamiento

Una transformación de blanqueamiento [41] es una transformación de descorrelación que transforma un conjunto arbitrario de variables que tienen una matriz de covarianza conocida M en un conjunto de nuevas variables cuya covarianza es la matriz identidad (lo que significa que no están correlacionados y todos tienen varianza distinta de cero).

La transformación se llama "blanqueamiento"porque cambia el vector de entrada en un vector de entrada de ruido blanco. Esta transformación se diferencia de una transformación general de descorrelación, en que este último sólo hace que el covarianzas igual a cero, de manera que la matriz de correlación puede ser cualquier matriz diagonal.

Este proceso se realiza usando el algoritmo PCA (descrito anteriormente), específicamente con la descomposición de valores singulares. Otra ventaja de usar PCA como proceso de blanqueamiento es la posibilidad de reducir la dimensión de los componentes independientes y asi obtener los componentes principales.

Los pasos a seguir para efectuar el algoritmo ICA se muestran a continuación:

Algoritmo 1: Resumen de algoritmo ICA

Proceso de centrado de vector de señales observadas

1. Calcular $\hat{\mathbf{x}} = \mathbf{x} - E\{\mathbf{x}\}$

Proceso de blanqueamiento (Algoritmo PCA)

2. Calcular $E\{\hat{\mathbf{x}}\hat{\mathbf{x}}^T\}$
3. Descomponer $\hat{\mathbf{x}}\hat{\mathbf{x}}^T \rightarrow \mathbf{E}\Sigma\mathbf{E}^T$
4. Calcular $\tilde{\mathbf{x}} = \mathbf{E}\Sigma^{-\frac{1}{2}}\mathbf{E}^T\hat{\mathbf{x}}$

Algoritmo ICA

5. Seleccionar $\mathbf{w}$ aleatorios

repetir

6. Calcular $\mathbf{w}^+ = E\{\tilde{\mathbf{x}}G_i(\mathbf{w}^T\tilde{\mathbf{x}})\} - E\{G_i(\mathbf{w}^T\tilde{\mathbf{x}})\}$
7. Calcular $\mathbf{w} = \dfrac{\mathbf{w}^+}{||\mathbf{w}^+||}$

hasta que *Calcular hasta llegar a converger*;

Limitaciones del algoritmo ICA

El modelo ICA presenta las siguientes limitaciones [42]:

1. Dado que se asume que las señales observadas son originadas por una combinación lineal de sus componenetes, no se posible restaurar el orden de los mismos.

2. Este método no da información, ni el signo de la varianza de la señal original; sin embargo, al realizar el pre-procesamiento descrito anteriormente, es posi-

ble estandarizar las características iniciales x_j de modo que su varianza sea unitaria.

2.4 Decodificación

El enfoque del proceso de decodificación (o estimación) consiste en determinar una aproximación de un valor útil para algún propósito, incluso si los datos de entrada pueden estar incompletos. Sin embargo, se busca que ésta aproximación se derive de la mayor y mejor información posible o disponible. El problema de decodificación puede ser visto como la identificación del sistema desconocido que produce la variable deseada, en nuestro caso, el sistema de interés toma como entrada las señales EMG y produce una estimación de la variable cinemática deseada como su salida, como se muestra en la Figura 2.11.

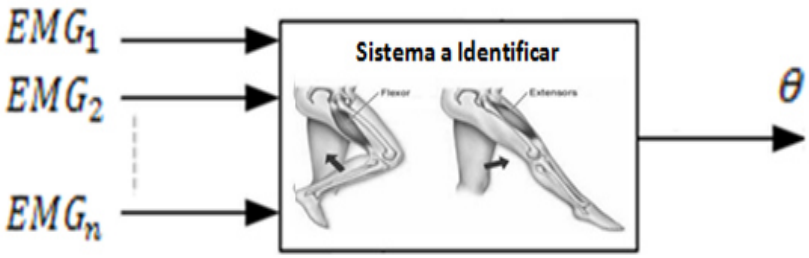


Figura 2.11: Problema de decodificación.

En este trabajo se emplean dos enfoques diferentes los cuales son: filtros adaptativos y redes neuronales [10–14]. Para el filtro adaptativo, se supone que el sistema puede ser representado como un modelo lineal, un filtro FIR que está conectado para formar una retroalimentación del error, tal como se muestra en la Figura 2.12. Un filtro FIR es definido por la siguiente ecuación:

$$y(n) = \sum_{k=0}^{N-1} w_k x(n-k) = \mathbf{w}^T \mathbf{x}, \tag{2.19}$$

donde, $x(n)$ es la señal de entrada, $y(n)$ es la señal de salida en la n-ésima muestra, w_k representa el k-ésimo coeficiente del filtro FIR y N es el orden del filtro; este filtro actualizara sus propios coeficientes (respuesta al impulso) de acuerdo con un proceso de adaptación. El proceso de adaptación en lazo cerrado implica el uso de una función de costo, que es un criterio a minimizar para obtener un comportamiento óptimo del filtro, el cual depende de un algoritmo específico, que determina cómo se debe modificar el filtro para reducir al mínimo la función de costo en la siguiente iteración.

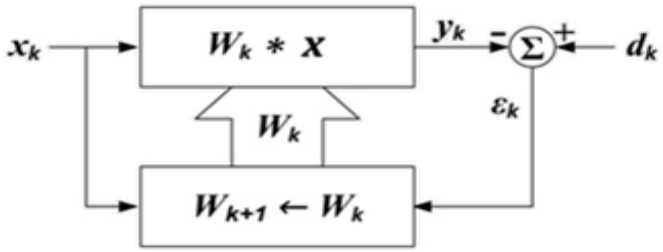


Figura 2.12: Diagrama de bloques de un filtro adaptativo. Donde $\mathbf{x}_k$ es la señal de entrada, d_k es la respuesta deseada (salida del sistema desconocido), $\mathbf{y}_k$ es la señal de salida del filtro y ϵ_k es la señal de error en la k-ésima iteración.

Finalmente, se optó por explorar el uso de redes neuronales artificiales (RNA) debido a la necesidad de cubrir la posibilidad de que el modelo a identificar sea no lineal, tomando en cuenta el potencial que poseen las RNAs para aprender el comportamiento de ciertos sistemas no lineales. El elemento clave de este paradigma es su estructura de procesamiento de información y la naturaleza paralela del sistema en conjunto. Dicha estructura se compone de un gran número de elementos de procesamiento altamente interconectados y trabajan juntos para resolver problemas específicos. Las RNAs, también pueden aprender con el ejemplo. Los métodos de aprendizaje más populares son: i) aprendizaje supervisado o no supervisado. En este trabajo la RNA fue entrenada usando aprendizaje supervisado como se muestra en la Figura 2.13.

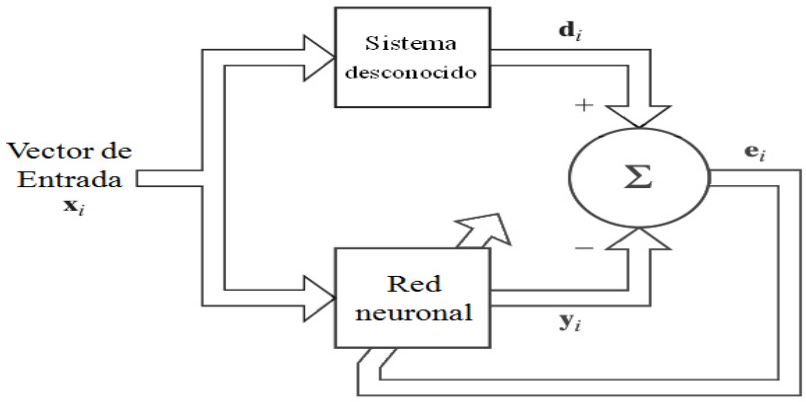


Figura 2.13: Estructura de una red neuronal artificial con aprendizaje supervisado. Donde $\mathbf{x}_i$ es el vector de entrada, $\mathbf{y}_i$ es el vector de salida de la RNA, $\mathbf{d}_i$ es el vector deseado y $\mathbf{e}_i$ es el vector de error en la i-ésima muestra.

2.4.1 Filtros Adaptativos

Cuando se habla de filtros adaptativos, está implícito que los parámetros que caracterizan al filtro, tales como el ancho de banda y frecuencias de los ceros, entre otros, cambian con el tiempo, esto es, los coeficientes de los filtros adaptativos son variantes en el tiempo y dependen de una regla de adaptación, en contraposición a los coeficientes de la mayoría de los filtros (selectivos en frecuencia) que son, en la mayoría de los casos, sistemas invariantes con el tiempo. En este trabajo se abordan los siguientes esquemas de adaptación:

- Filtro Wiener
- Filtro LMS
- Filtro RLS

Filtro Wiener

El filtro Wiener es un tipo de filtro óptimo lineal, el cual se basa en un enfoque estadístico, en donde los coeficientes del filtro $\hat{\mathbf{w}}(n)$ son calculados de tal manera que se minimice el error cuadrático medio (en ingles, MSE) que existe entre la señal deseada y la entrada del filtro; este método considera una señal deseada $d(n)$, un filtro FIR con M coeficientes:

$$\mathbf{w} = [w_0, w_1, \ldots, w_{M-1}]^T,$$

un vector de entrada de la siguente forma:

$$\mathbf{x}_n = [x(n), x(n-1), \ldots, x(n-M+1)]^T.$$

La salida del filtro se obtiene de 2.19,

$$y(n) = [w_0, w_1, \ldots, w_{M-1}] \begin{bmatrix} x(n) \\ x(n-1) \\ \cdots \\ x(n-M+1) \end{bmatrix} = \mathbf{w}^T\mathbf{x}_n = \mathbf{x}_n^T\mathbf{w}, \tag{2.20}$$

y el error está definido como

$$e(n) = d(n) - y(n) = d(n) - \mathbf{w}^T\mathbf{x}_n, \tag{2.21}$$

Los coeficientes del filtro optimo $\mathbf{w}_o$ se calculan para minimizar la varianza como función de costo mediante el MSE

$$\begin{aligned} J(\mathbf{w}) &= E\{(e(n))^2\} \\ &= E\{(d(n) - \mathbf{w}^T\mathbf{x}_n)^2\} \\ &= E\{(d(n)^2 - 2(\mathbf{w}^T\mathbf{x}_n)d(n) + (\mathbf{w}^T\mathbf{x}_n)^2\} \\ &= \sigma_d^2 + \mathbf{w}^T E\{\mathbf{x}_n\mathbf{x}_n^T\}\mathbf{w} - 2\mathbf{w}^T E\{\mathbf{x}_n d(n)\} \end{aligned} \tag{2.22}$$

donde $E\{\cdot\}$ denota el valor esperado, σ_d es la varianza de la señal deseada, el término $E\{\mathbf{x}_n\mathbf{x}_n^T\}$ está dado por

$$E\{\mathbf{x}_n\mathbf{x}_n^T\} = E\left\{\begin{bmatrix} x(n) \\ x[n-1] \\ \vdots \\ x[n-M+1] \end{bmatrix}[x(n), x[n-1], \ldots, x[n-M+1]]\right\} = \mathbf{R}_{xx}, \quad (2.23)$$

y representa la matriz de autocorrelación de la entrada con una dimensión es $M \times M$, mientras que el término

$$E\{\mathbf{x}_n d(n)\} = E\left\{\begin{bmatrix} x(n) \\ x[n-1] \\ \vdots \\ x[n-M+1] \end{bmatrix} d(n)\right\} = \begin{bmatrix} r_{dx}[0] \\ r_{dx}[-1] \\ \vdots \\ r_{dx}[1-M] \end{bmatrix} = \mathbf{R}_{dx}, \quad (2.24)$$

es conocido como el vector de correlación cruzada entre la entrada y la salida deseada y es de M elementos. La función de costo queda definida de la siguiente manera:

$$J(\mathbf{w}) = \sigma_d^2 + \mathbf{w}^T\mathbf{R}_{xx}\mathbf{w} - 2\mathbf{w}^T\mathbf{R}_{dx}. \quad (2.25)$$

Para obtener la solución del filtro óptimo ($\hat{\mathbf{w}}_o$) se requiere calcular la derivada de la función de costo e igualarla a cero, considerando

$$\frac{\partial \mathbf{w}^T\mathbf{R}_{xx}\mathbf{w}}{\partial \mathbf{w}} = 2\mathbf{R}_{xx}\mathbf{w}, \qquad \frac{\partial \mathbf{w}^T\mathbf{R}_{dx}}{\partial \mathbf{w}} = \mathbf{R}_{dx}, \qquad \frac{\partial \sigma_d^2}{\partial \mathbf{w}} = 0$$

$$\frac{\partial J(\hat{\mathbf{w}}_o)}{\partial \hat{\mathbf{w}}_o} = 2\mathbf{R}_{xx}\hat{\mathbf{w}}_o - 2\mathbf{R}_{dx} = 0, \quad (2.26)$$

simplificando 2.26 obtenemos la solución para el filtro Wiener

$$\hat{\mathbf{w}}_o = \mathbf{R}_{xx}^{-1}\mathbf{R}_{dx}. \quad (2.27)$$

Los pasos para el diseño de un filtro Wiener se resumen a continuación:

Algoritmo 2: Solución Wiener

$\mathbf{x} = [x(n), x(n-1), ..., x(n-M+1)]^T$

$\mathbf{R}_{dx} = E\{\mathbf{x}d(n)\}$

$\mathbf{R}_{xx} = E\{\mathbf{x}\mathbf{x}^T\}$

$\hat{\mathbf{w}}_o = \mathbf{R}_{xx}^{-1}\mathbf{R}_{dx}$

Ventajas del filtro Wiener

- Para su diseño se necesita conocer solamente las propiedades estadísticas de segundo orden: la correlación cruzada entre la entrada y la salida deseada.
- No presenta retrasos inherentes.

Desventajas del filtro Wiener

- Es necesario que el proceso sea estacionario de segundo orden.
- Este algoritmo es completamente dependiente de la correlación entre la entrada y la salida, por consecuencia, si la entrada del filtro no está correlacionada con la salida deseada el desempeño del filtro será bajo.
- Se asume al sistema lineal, por lo que no es posible identificar sistemas no lineales.

Para el lector interesado que desee profundizar en el tema, la metodología fue interpretada de [43–45], en los cuales además de describir la deducción de la técnica, discuten el comportamiento del error, modelos de regresión, diversos ejemplos y aplicaciones; entre otros temas de interés.

Filtro LMS

El filtro LMS (del Inglés, Least Mean Square) es una clase de filtro adaptativo que usa como regla de adaptación el promedio del cuadrado del error (MSE), el filtro se adapta al error en el instante actual (ciclo actual) y con ello determina los coeficientes del filtro, la señal de error esta definida como la diferencia entre la señal de salida del filtro y la señal deseada. Esta técnica pertenece al tipo de algoritmos del gradiente desendiente, es decir, se basa en el gradiente de una función evaluada en un punto $\mathbf{w}(n)$, el cual indica la dirección de máximo incremento de la función en ese punto; sin embargo, el utilizar la dirección contraria al gradiente es análogo a minimizar la función de coste; por lo que el algoritmo LMS consiste en restarle a los coeficientes actuales el gradiente instantáneo.

Para determinar el algoritmo LMS se parte de la ecuación 2.26, la cual puede interpretarse como una estimación instantánea del gradiente, la ecuación toma la siguiente forma:

$$\frac{\partial \hat{\mathbf{J}}(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{x}_n \mathbf{x}_n^T \mathbf{w}(n) - \mathbf{x}_n d(n). \tag{2.28}$$

Sustituyendo 2.20 en 2.27 se obtiene:

$$\frac{\partial \hat{\mathbf{J}}(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{x}_n y(n) - \mathbf{x}_n d(n) = \mathbf{x}_n (y(n) - d(n)) = -e(n)\mathbf{x}_n \tag{2.29}$$

La dirección del gradiente $-\Delta J(\mathbf{w})$ es ponderada por el parámetro μ, el cual se le denomina como paso y con ello se puede controlar la exploración de la solución por iteración, por lo tanto, se define el algoritmo de adaptación para el filtro LMS como

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu e(n)\mathbf{x}(n) \tag{2.30}$$

El resumen de la metodología se enlista a continuación:

Algoritmo 3: Filtro LMS

$\hat{\mathbf{w}}(0) = 0$

repetir

$\mathbf{x}(n) = [x(n), x(n-1), ..., x(n-M+1)]^T$

$\hat{\mathbf{w}}(n) = [\hat{w}_0(n), \hat{w}_1(n), ..., \hat{w}_{M-1}(n)]^T$

$y(n) = \hat{\mathbf{w}}^H(n) \cdot \mathbf{x}(n)$

$e(n) = d(n) - y(n)$

$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu e^*(n)\mathbf{x}(n)$

hasta que *Calcular desde n=1, 2, ..., N*;

Donde, $e(n)$ es la señal error, $y(n)$ es la salida del filtro adaptativo en la muestra n, N es el número de ciclos de adaptación y μ es el tamaño del paso por ciclo.

Ventajas del filtro LMS

- Permite seguir cambios en las estadísticas de las señales (tracking).
- Una implementación sencilla y una baja carga computacional $M + 1$ multiplicaciones y $M - 1$ sumas por cada iteración del algoritmo en el caso de señales reales.
- No requiere de la inversión de la matriz de correlación

Desventajas del filtro LMS

- Necesita mayor número de iteraciones para llegar a la convergencia, tiene una convergencia muy lenta y ésta depende del parámetro μ.
- Al ser un método lineal, podría tener problemas para identificar sistemas no lineales.

Para el lector interesado que desee profundizar en el tema, la metodología fue interpretada de [43–45], en los cuales además de describir la deducción de la técnica, se discute el comportamiento del error, análisis de convergencia y el comportamiento del transitorio; entre otros temas de interés.

Filtro RLS

El algoritmo RLS (del Inglés, Recursive Least Squares) es un caso particular del filtro de Kalman (donde se considera un factor de olvido como la matriz de transición de estados del sistema desconocido) y se usa en filtros adaptativos para determinar los coeficientes de un filtro adaptativo basado en minimizar una función de costo lineal de mínimos cuadrados (LS, Least squares) ponderando a la señal de entrada; este método se las arregla para calcular la inversa de la matriz de correlación de forma recursiva y tiene una rápida tasa de convergencia. Sin embargo, la complejidad del algoritmo hace que la carga computacional sea mayor, en comparación de otros algoritmos.

Para definir el algoritmo RLS, se empieza con el cálculo usando las condiciones iniciales conocidas luego se actualiza el viejo estimado basándonos en la informacion contenida en los nuevos datos, posteriormente se minimiza la función de costo $J(n)$ que posee la siguiente forma

$$J(n) = \sum_{i=1}^{n} \beta(n,i)|e(i)|^2 \tag{2.31}$$

donde n es el tamaño de la variable observada, $e(i)$ es el error correspondiente a 2.21, $\beta(n,i)$ son los factores de ponderación, en general, este factor se define para asegurar que los datos en el pasado distante sean olvidados y así llegar a seguir las variaciones estadísticas de los datos observables. Este factor tiene la siguiente propiedad

$$0 < \beta(n,i) \leq 1, \qquad i = 1, 2, \ldots, n. \tag{2.32}$$

La ponderación por factor de olvido (λ) es uno de los casos de ponderación

más comunes, el cual es definido de la siguiente forma

$$\beta(n,i) = \lambda^{n-i}, \qquad i = 1, 2, \ldots, n. \tag{2.33}$$

Siendo λ definida como una constante positiva próxima y menor que la unidad. Cuando $\lambda = 1$ se dice que el algoritmo posee memoria infinita, dando como resultado el método ordinario de mínimos cuadrados. Sustituyendo 2.33 en 2.31 resulta lo siguiente

$$\begin{aligned} J(n) &= \sum_{i=1}^{n} \lambda^{n-i} |e(i)|^2 \\ &= \sum_{i=1}^{n} \lambda^{n-i} (d(i) - \mathbf{x}^H(i)\mathbf{w}(n))^* (d(i) - \mathbf{x}^H(i)\mathbf{w}(n)) \\ &= \sum_{i=1}^{n} \lambda^{n-i} (d^2(i) - d^*(i)\mathbf{x}^H\mathbf{w} - \mathbf{x}^H\mathbf{w}d^*(i) + \mathbf{w}^H\mathbf{x}\mathbf{x}^H\mathbf{w}) \end{aligned} \tag{2.34}$$

Para obtener la solución del filtro se requiere calcular la derivada de la función de costo e igualarla a cero, considerando

$$\frac{\partial \mathbf{w}^H\mathbf{x}\mathbf{x}^H\mathbf{w}}{\partial \mathbf{w}} = 2\mathbf{x}\mathbf{x}^H\mathbf{w}, \qquad -\frac{\partial 2\mathbf{w}^H\mathbf{x}d^*(i)}{\partial \mathbf{w}} = -2\mathbf{x}d^*(i), \qquad \frac{\partial d^2(n)}{\partial \mathbf{w}} = 0$$

$$\frac{\partial J(\mathbf{n})}{\partial \hat{\mathbf{w}}} = \sum_{i=1}^{n} \lambda^{n-i} (2\mathbf{x}\mathbf{x}^H\hat{\mathbf{w}} - 2\mathbf{x}d^*(i)) \tag{2.35}$$

Separando 2.35 se obtiene

$$\sum_{i=1}^{n} [\lambda^{n-i}\mathbf{x}(i)\mathbf{x}^H(i)]\hat{\mathbf{w}}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{x}(i)d^*(i), \tag{2.36}$$

el cual tiene la forma de

$$\Phi(n)\hat{\mathbf{w}}(n) = \mathbf{Z}(n), \tag{2.37}$$

siendo

$$\Phi(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{x}(i)\mathbf{x}^H(i), \tag{2.38}$$

$$\mathbf{Z}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{x}(i)d^*(i), \tag{2.39}$$

donde $\mathbf{Z}(n)$ es el vector de correlacción cruzada de la entrada y la salida de tamaño M y $\Phi(n)$ es la matriz de autocorrelación de la entrada cuya dimensión es $M \times M$. Aislando el termino correspondiente a $i = n$ de 2.38 obtenemos

$$\Phi(n) = \lambda \left[\sum_{i=1}^{n-1} \lambda^{n-1-i}\mathbf{x}(i)\mathbf{x}^H(i)\right] + \mathbf{x}(n)\mathbf{x}^H(n), \tag{2.40}$$

Aplicando un retrazo en 2.38 resulta

$$\Phi(n-1) = \sum_{i=1}^{n-1} \lambda^{n-1-i}\mathbf{x}(i)\mathbf{x}^H(i), \tag{2.41}$$

sustituyendo 2.41 en 2.40 se obtiene

$$\Phi(n) = \lambda\Phi(n-1) + \mathbf{x}(n)\mathbf{x}^H(n). \tag{2.42}$$

De manera similar se desarrolla $\mathbf{Z}(n)$ para tomar la siguiente forma

$$\mathbf{Z}(n) = \lambda\mathbf{Z}(n-1) + \mathbf{x}(n)d^*(n). \tag{2.43}$$

La ecuación 2.37 puede tomar la siguiente forma

$$\hat{\mathbf{w}}(n) = \Phi^{-1}(n)\mathbf{Z}(n), \tag{2.44}$$

el término $\Phi^{-1}(n)$, en la práctica, resulta ser una operación que consume una considerable cantidad de recursos computacionales, debido a que generalmente el número de peso M es grande y se desea calcular este término para $n = 1, 2, \ldots, \infty$; sin embargo el lema de la inversión de una matriz puede ser usado para realizar el cálculo de $\Phi^{-1}(n)$.

Lema de inversión de una matriz

Sean $\mathbf{A}$ y $\mathbf{B}$ dos matrices de $M \times M$, definidas positivas y que estén relacionadas mediante

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^H, \tag{2.45}$$

donde $\mathbf{D}$ es otra matriz definida positiva cuya dimensión sea de $N \times M$ y $\mathbf{C}$ es una matriz de $M \times N$. La matriz $\mathbf{A}^{-1}$ se puede expresar de la siguiente manera.

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^H\mathbf{B}, \tag{2.46}$$

Dada la ecuación 2.44, se identifican cada elemento de 2.46 como

$$\mathbf{A} = \Phi(n),$$
$$\mathbf{B}^{-1} = \lambda\Phi(n-1),$$
$$\mathbf{C} = \mathbf{x}(n),$$
$$\mathbf{D} = 1.$$

Sustituyendo 2.46 se obtiene

$$\Phi^{-1}(n) = \lambda^{-1}\Phi^{-1}(n-1) - \frac{\lambda^{-2}\Phi^{-1}(n-1)\mathbf{x}(n)\mathbf{x}^H(n)\Phi^{-1}(n-1)}{1+\lambda^{-1}\mathbf{x}^H(n)\Phi^{-1}(n-1)\mathbf{x}(n)}, \tag{2.47}$$

definiendo $\mathbf{k}(n)$ y $\mathbf{P}(n)$ como

$$\mathbf{P}(n) = \Phi^{-1}(n) \tag{2.48}$$

y

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{x}(n)}{1+\lambda^{-1}\mathbf{x}^H(n)\mathbf{P}(n-1)\mathbf{x}(n)}. \tag{2.49}$$

la cual puede expresarse de la siguiente manera

$$\mathbf{k}(n)[1+\lambda^{-1}\mathbf{x}^H(n)\mathbf{P}(n-1)\mathbf{x}(n)] = \lambda^{-1}\mathbf{P}(n-1)\mathbf{x}(n), \tag{2.50}$$

$$\mathbf{k}(n) = \lambda^{-1}\mathbf{P}(n-1)\mathbf{x}(n) - \lambda^{-1}\mathbf{k}(n)\mathbf{x}^H(n)\mathbf{P}(n-1)\mathbf{x}(n). \tag{2.51}$$

La ecuación 2.47 toma la siguiente forma

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{x}^H(n)\mathbf{P}(n-1), \tag{2.52}$$

y 2.51 se simplifica a

$$\mathbf{k}(n) = \mathbf{P}(n)\mathbf{x}(n). \tag{2.53}$$

El siguiente paso es sustituir 2.43 y 2.52 en 2.44

$$\begin{aligned}\hat{\mathbf{w}}(n) &= \lambda\mathbf{P}(n)\mathbf{Z}(n-1) + \mathbf{P}(n)\mathbf{x}(n)d^*(n)\\ &= \mathbf{P}(n-1)\mathbf{Z}(n-1) - \mathbf{k}(n)\mathbf{x}^H(n)\mathbf{P}(n-1)\mathbf{Z}(n-1)\\ &+ \mathbf{P}(n)\mathbf{x}(n)d^*(n)\end{aligned} \tag{2.54}$$

dado 2.48, 2.53 y sustituyendo en 2.54 la ecuación 2.44 atrasada una muestra da el siguiente resultado

$$\begin{aligned}\hat{\mathbf{w}}(n) &= \hat{\mathbf{w}}(n-1) - \mathbf{k}(n)\mathbf{x}^H(n)\hat{\mathbf{w}}(n-1) + \mathbf{P}(n)\mathbf{x}(n)d^*(n) \\ &= \hat{\mathbf{w}}(n-1) - \mathbf{k}(n)\mathbf{x}^H(n)\hat{\mathbf{w}}(n-1) + \mathbf{k}(n)d^*(n) \\ &= \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)[d^*(n) - \mathbf{x}^H(n)\hat{\mathbf{w}}(n-1)].\end{aligned} \tag{2.55}$$

Definiendo el error de estimación a priori como

$$\begin{aligned}\xi(n) &= d(n) - \mathbf{x}^T(n)\hat{\mathbf{w}}^*(n-1) \\ &= d(n) - \hat{\mathbf{w}}^H(n-1)\mathbf{x}(n).\end{aligned} \tag{2.56}$$

Donde el producto punto de $\xi(n)$ representa el estimado de la respuesta deseada $d(n)$ basada en los coeficientes calculados para la iteración $n-1$. Por último se sustituye 2.56 en 2.55 y se obtiene

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n). \tag{2.57}$$

Finalmente la ecuación 2.57 describe la base para actualizar los coeficientes en cada iteración del algoritmo RLS.

Ventajas del filtro RLS

- Permite seguir cambios en las estadísticas de las señales (tracking).
- El algoritmo RLS tiene una parte de búsqueda y otra parte predictiva dentro del mismo.
- Su capacidad recursiva iterativa. Con la cual con muy pocas iteraciones disminuye rápidamente el error cuadrático.

Desventajas del filtro RLS

- El algoritmo RLS presenta una mayor complejidad númerica en comparación con el algoritmo LMS.
- Necesitan mayor número de coeficientes o parámetros para funcionar, lo cual es una limitante.
- Al ser un método lineal, podría tener problemas para identificar sistemas no lineales.

Los pasos para el diseño del filtro RLS se enlista a continuación.

Algoritmo 4: Resumen del filtro adaptivo RLS

$\hat{\mathbf{w}}(0) = 0$

$\hat{\mathbf{P}}(0) = \delta^{-1}\mathbf{I}$

repetir

$\mathbf{x}(n) = [x(n), x(n-1), ..., x(n-M+1)]^T$

$\hat{\mathbf{w}}(n) = [\hat{w}_0(n), \hat{w}_1(n), ..., \hat{w}_{M-1}(n)]^T$

$$k(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{x}(n)}{1 + \lambda^{-1}\mathbf{x}^H(n)\mathbf{P}(n-1)\mathbf{x}(n)}$$

$\xi(n) = d(n) - \mathbf{w}^H(n-1)\mathbf{x}(n)$

$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n)$

$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{x}^H(n)\mathbf{P}(n-1)$

hasta que *Calcular desde n=1, 2, ..., N*;

Para el lector interesado que desee profundizar en el tema, la metodología fue interpretada de [43–45], en los cuales además de describir la deducción de la técnica, se discute la robustez, convergencia y aplicaciónes del algoritmo; entre otros temas de interés.

2.4.2 Redes neuronales artificales

Actualmente se han hecho intentos por decodificar la cinemática de la locomoción humana usando redes neuronales, obteniendo resultados prometedores (para arquitecturas avanzadas) [11, 25], sin embargo el costo computacional para el entrenamiento es muy costoso.

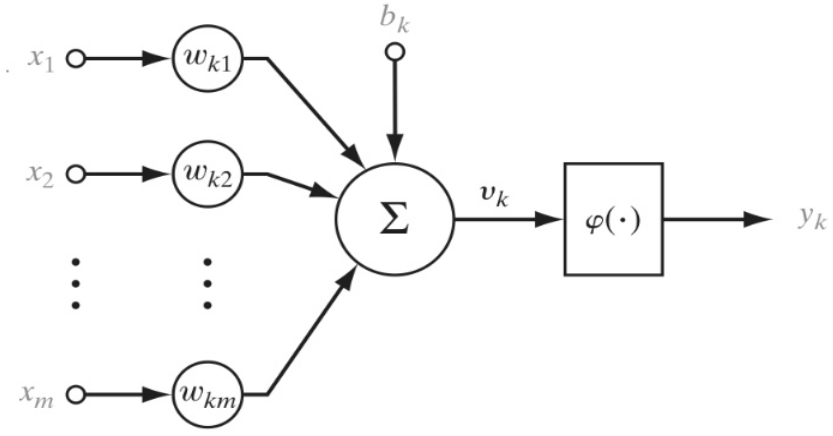


Figura 2.14: Diagrama de bloques de una neurona artificial no lineal [44].

El modelo más sencillo de una RNA se ilustra en la Figura 2.14, el cual es llamado como Perceptrón de Rosenblatt, que tiene como base el modelo de una neurona de McCulloch-Pitts. En términos matemáticos, la neurona k puede ser descrita con el siguiente par de ecuaciones

$$u_k = \sum_{j=1}^{m} w_{kj} x_j, \tag{2.58}$$

$$y_k = \varphi(u_k + b_k), \tag{2.59}$$

donde $x_1, x_2, \ldots, x_m$ son la señales de entrada, $w_{k1}, w_{k2}, \ldots, w_{km}$ son los pesos sinápticos de la neurona k, u_k es una la combinación lineal de la entrada con los pesos, b_k es el bias, y_k es la señal de salida de la neurona y $\varphi(\cdot)$ es una función no lineal que se conoce como función de activación; las funciones de activación más comunes son

- Función logística (sigmoidal)

$$\varphi_k(v_k(n)) = \frac{1}{1 + e^{-av_k(n)}} \tag{2.60}$$

- Función de la tangente hiperbólica

$$\varphi_k(v_k(n)) = a \tanh(bv_k(n)) \tag{2.61}$$

- Función signo

$$\varphi_k(v_k(n)) = \text{sgn}(v_k(n)) = \begin{cases} +1 & si\ v_k(n) \geq 0 \\ -1 & si\ v_k(n) < 0 \end{cases} \tag{2.62}$$

La aplicación más común para este modelo es el paradigma de clasificación (el tema se aborda en detalle en la Sección 2.5.2), usando 2.62 como función de activación. Naturalmente este modelo se puede extrapolar hacia arquitecturas más sofisticadas como el perceptrón multicapa.

Perceptrón Multicapa

El perceptrón multicapa (MLP) es una modificación del perceptrón lineal estándar. Básicamente un MLP es un modelo de red neuronal artificial que mapea conjuntos de datos de entrada en un conjunto de salidas deseadas. Un MLP consiste en múltiples capas de nodos (neuronas), con cada capa conectada totalmente a la siguiente (como se muestra en la Figura 2.15). A excepción de los nodos de entrada, cada nodo es una neurona (o elemento de procesamiento) con una función activación no lineal.

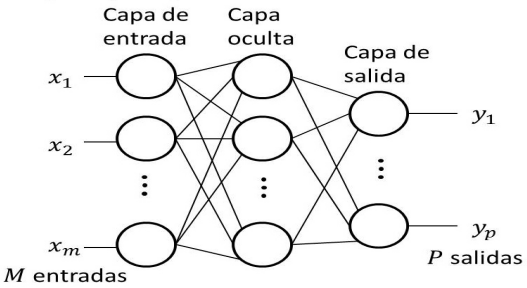


Figura 2.15: Arquitectura básica de un perceptrón multicapa.

Generalmente el MLP es capaz de aprender patrones mediante aprendizaje supervisado. El aprendizaje supervisado es una técnica de aprendizaje automático que se basa en los datos de entrada y salidas deseadas para ajustar los pesos de la red neuronal con el fin de aproximar una función no lineal o algún tipo de relación. Las técnicas típicas de aprendizaje supervisado incluyen [50]:

- Adaline
- Madaline
- Retropropagación
- Máquina de Boltzmann

El objetivo del aprendizaje es predecir con precisión los valores de salida para un conjunto dado de valores de entrada después de que la red sea expuesta en una serie de datos de entrenamiento. En este trabajo se centra en la técnica de aprendizaje supervisado denominada retropropagación (método resumido en el algoritmo 5) que se deduce partiendo de N pares de datos de entrenamiento que consisten en datos de entrada con su salida deseada correspondiente, $(x(n), d(n))$ para $n = 1, 2, \ldots, N$; lo que se busca es diseñar una red multicapa de propagación directa $(f(x))$ comenzando por

$$\mathbf{d}(n) = f(\mathbf{x}(n)) + \mathbf{e}(n) \tag{2.63}$$

siendo P el número de neuronas de salida, 2.58 se le conoce como campo local inducido y 2.59 es la representación salida de la red; planteamos minimizar el error definido como

$$e_j(n) = d_j(n) + y_j(n), \tag{2.64}$$

$$E(n) = \frac{1}{2}\sum_{j=1}^{P} e_j^2(n), \tag{2.65}$$

el algoritmo de retropropagación consiste en aplicar una corrección $\Delta w_{ji}(n)$ a los pesos sinápticos $w_{ji}(n)$, que se obtiene calculando la derivada parcial del error instantáneo $E(n)$ total con respecto a los pesos $w_{ji}(n)$, esto se logra usando la regla del

cálculo conocida como la regla de la cadena, describiendo al gradiente de la siguiente manera

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}. \tag{2.66}$$

La ecuación 2.66 representa un factor de sensibilidad el cual determina la dirección del espacio de búsqueda de los pesos. Derivando 2.65 con respecto a $e_j(n)$, 2.64 con respecto a $y_j(n)$, 2.59 con respecto a $v_j(n)$ y finalmente 2.58 con respecto a $w_{ji}(n)$ se obtiene

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n), \tag{2.67}$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1, \tag{2.68}$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi'(v_j(n)), \tag{2.69}$$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_j(n). \tag{2.70}$$

sustituyendo 2.67, 2.68, 2.69, 2.70 en 2.66 da el siguiente resultado

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi'_j(v_j(n))y_i(n). \tag{2.71}$$

Definiendo $\Delta w_{ji}(n)$ como

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)}, \tag{2.72}$$

donde η es el radio de aprendizaje. La ecuación 2.72 puede corresponde a

$$\Delta w_{ji}(n) = -\eta \delta_j(n) y_j(n), \tag{2.73}$$

donde $\delta_j(n)$ es el gradiente local descrito por

$$\delta_j(n) = e_j(n)\varphi'_j(v_j(n)). \tag{2.74}$$

una vez definida $\Delta w_{ji}(n)$, la actualización de los pesos $w_{ji}(n)$ se realiza con la siguiente ecuación

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \eta \delta_j^l(n) y_i^{l-1}(n) \tag{2.75}$$

la ecuación 2.75 actualiza los pesos relacionados con la neurona l, sin embargo $\delta_j(n)$, descrita por 2.74, solo corresponde a las neuronas de la capa de salida, ya que para

las neuronas de las capas ocultas no se tiene una respuesta deseada específica, el gradiente local para una neurona j de alguna capa oculta se define como

$$\begin{aligned}\delta_j(n) &= -\frac{\partial E(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} \\ &= -\frac{\partial E(n)}{\partial y_j(n)}\varphi_j'(v_j(n)),\end{aligned} \tag{2.76}$$

se deriva el error instantáneo de la neurona de salida k (2.65) con respecto a la salida de la neurona oculta j y se expresa como

$$\begin{aligned}\frac{\partial E(n)}{\partial y_j(n)} &= \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} \\ &= \sum_k e_k \frac{\partial e_k(n)}{\partial v_k(n)}\frac{\partial v_k(n)}{\partial y_j(n)}\end{aligned} \tag{2.77}$$

sustituyendo 2.59 en 2.64 y derivando con respecto a $v_k(n)$ se obtiene

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n)), \tag{2.78}$$

continuando con la ecuación 2.58 la cual se deriva con respecto $y_j(n)$

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{ji}(n), \tag{2.79}$$

considerando las ecuaciones 2.74, 2.78, 2.79; la ecuación 2.77 se reescribe de la siguiente manera

$$\begin{aligned}\frac{\partial E(n)}{\partial y_j(n)} &= -\sum_k e_k \varphi'(v_k(n)) w_{kj}(n), \\ &= -\sum_k \delta_k(n) w_{kj}(n),\end{aligned} \tag{2.80}$$

por último se sustituye 2.80 en 2.76 y resulta en

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n). \tag{2.81}$$

La ecuación 2.81 define el gradiente local para neuronas ocultas y con ello se completa

las formulas para enunciar el método de entrenamiento para esta RNA.

Algoritmo 5: Perceptrón multicapa

Inicializar pesos y umbrales (aleatorios).

Presentar los ejemplos de entrenamiento.

repetir

Propagación directa

(por cada neurona j con peso k en la capa l)

$v_j^{(l)}(n) = \sum_i w_{ji}^{(l)}(n) y_i^{l-1}(n)$

$y_j^{(l)}(n) = \varphi_j(v_j(n))$

$y_j^{(0)}(n) = x_j(n)$

Retro-propagación *(por cada neurona j con peso k en la capa oculta l ó capa de salida L)*

$$\delta_j^l(n) = \begin{cases} e_j^{(L)}(n)\varphi_j^{'}(v_j^{(L)}(n)) \\ \varphi_j^{'}(v_j^{(l)}(n)) \sum_k \delta_k^{(l+1)} w_{kj}^{(l+1)}(n) \end{cases}$$

Actualización *(para cada neurona j con peso k en la capa l)*

$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \eta \delta_j^l(n) y_i^{l-1}(n)$

hasta que *Calcular desde n=1, 2, ..., N*;

Donde, $\varphi(\cdot)$ es la función de activación y δ es el gradiente local de la neurona.

Esta arquitectura de RNA puede distinguir los datos que no son linealmente separables, sin embargo, cuando MLP tiene dificultades para imitar un determinado comportamiento dinámico por sí mismo, con esta motivación planteada, se puede intuir que al añadir una etapa anterior se podría mejorar la rendimiento del MLP. Por lo tanto, la adición de una memoria a corto plazo (como se muestra en la Figura 2.16) a la entrada proporcionará más información de la entrada a la red y transformando a la red en una red dinámica [44]; una forma sencilla de conseguir esta modificación es mediante el uso de retrazos discretos de tiempo.

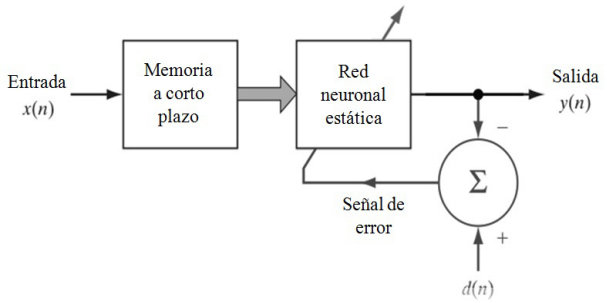


Figura 2.16: Estructura de una red neuronal dinámica [44].

La descripción de la manera en el que se realiza el diseño de la arquitectura de la red se muestra en Figura 2.17a y el entrenamiento de la red se describe en el algoritmo 5 y en la Figura 2.17b.

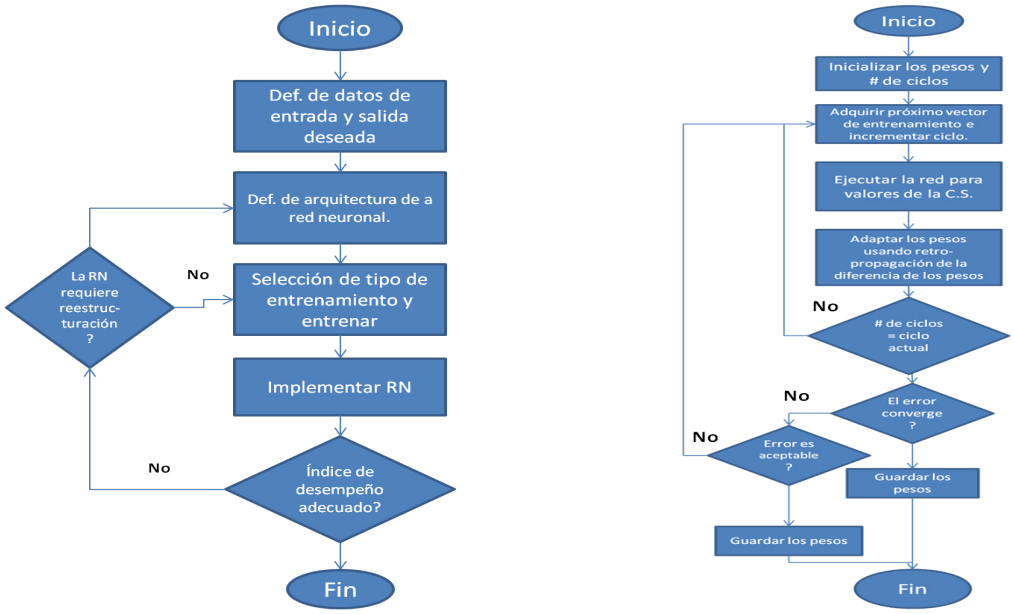


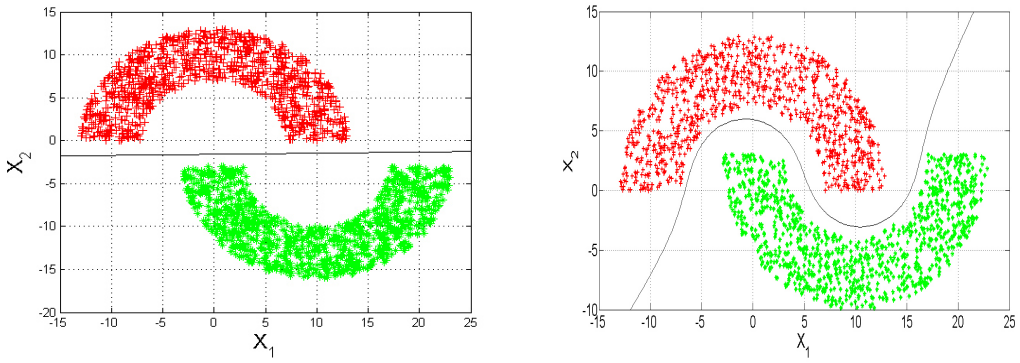
Figura 2.17: a) Diagrama del diseño y b) entrenamiento de una RN.

Para el lector interesado que desee profundizar en el tema, la metodología fue interpretada de [44], en el cual además de describir la deducción de la técnica, se discute la redes convolucionales, filtrado nolineal y aplicaciónes del algoritmo; entre otros temas de interés.

2.5 Clasificación

La clasificación es la actividad de definir un modelo de una función discriminante, capaz de identificar a que categoría pertenece una nueva observación de un grupo de características [44, 46]. Los clasificadores se dividen en dos tipos: lineales y no lineales. Los clasificadores lineales generalmente son más robustos que los no-lineales [6], debido a que tiene pocos parámetros libres para sintonizar y por lo tanto son menos propenso a exceso de ajuste. Mientras que los no lineales son más adecuados para encontrar la solución cuando la estructura de los datos es muy compleja. En particular, cuando la fuente de los datos a clasificar no se conoce con exactitud o se consideran como variables aleatorias, una posible solución es usar métodos para encontrar transformaciones no lineales, por ejemplo RNA o métodos basados en kernel.

El problema de clasificación puede ser linealmente separable o no linealmente separable tal como se ilustra en la Figura 2.18.



(a) (b)

Figura 2.18: Problema de clasificación de la doble luna usando SVM: a) linealmente separable b) no linealmente separable.

Donde las muestras en rojo y en verde representan las dos actividades distintas a clasificar.

En este trabajo se abordan los siguientes esquemas de clasificación [4–6, 9]:

- Clasificador de bayes (lineal).
- Máquina de soporte vectorial (SVM).
- RNA Perceptrón multicapa (MLP).

2.5.1 Extracción de características

Dado que la señal adquirida generalmente está expresada como una secuencia de muestras muy grandes, resulta impractico utilizar la señal completa como entrada de un clasificador y mas aún con las señales fisiológicas, que presentan un cierto grado de aleatoriedad; debido a esto, se pretende usar las características principales de la señal, para mejorar la eficiencia de clasificación [47]. Como se menciono en el capítulo 1, las caracteristicas pueden ser adquiridas en el dominio tiempo, frecuencia o tiempo-frecuencia.

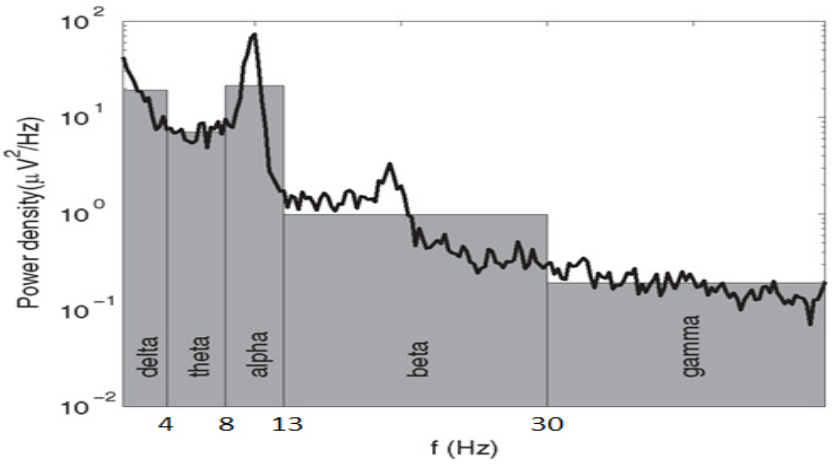


Figura 2.19: Espectro de una señal EEG.

Esta etapa deberá calcular y presentar las características preseleccionadas para un clasificador; el cual deberá estimar la velocidad de operación del usuario usando como entrada un vector de características de la señal EEG, por lo tanto estas caracteristicas serán adquiridas a patir del espectro de la señal, obtenido mediante la

Trasformada discreta de Fourier (DFT), posteriormente se seleccionan los componentes o rangos de frecuencia correspondientes a las ondas α y β (como se muestra en la Figura 2.19), debido a que estas ondas están relacionadas a las actividades motrices del ser humano [4–6, 9, 32].

Transformada discreta de Fourier

La transformada de Fourier se describe como

$$X(e^{j\omega t}) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt \tag{2.82}$$

si a (2.82) se le aplicase cualquier señal en tiempo discreto $x(nT)$ de tamaño N y que sea completamente sumable; se conoce como transformada de Fourier en tiempo discreto (DTFT), la transformada se expresa de la siguiente forma.

$$X(e^{j\omega T}) = F\{x(nT)\} = T\sum_{n=-\infty}^{\infty} x(nT)e^{-j\omega nT}, \tag{2.83}$$

donde T es el periodo de muestro y debido a la relación $X(e^{j(\omega+2\pi)}) = X(e^{j\omega}e^{j2\pi}) = e^{j\omega})$ se puede apreciar que una DTFT produce un espectro periódico con período 2π. Cabe aclarar que $X(e^{-j\omega t})$ es una función continua en el dominio de la frecuencia; sin embargo, es posible discretizarlo si se muestrea N veces sobre todo su período (2π); por lo tanto 2.83 se modifica de la siguiente forma

$$X(e^{j\frac{2\pi}{NT}k}) = T\sum_{n=0}^{N-1} x(nT)e^{-j\frac{n\pi}{NT}k}, \tag{2.84}$$

Normalizando $T = 1$ y definiendo W_n como

$$W_n = e^{-j\frac{2\pi}{NT}} \tag{2.85}$$

Se obtiene la transformada discreta de Fourier (DFT)

$$X(k) = \sum_{n=0}^{N-1} x(n){W_n}^{kn}, \tag{2.86}$$

siendo la operación inversa (IDFT) descrita como

$$x(n) = \frac{1}{N}\sum_{n=0}^{N-1} X(k){W_n}^{-kn}, \tag{2.87}$$

Con el cual es posible calcular el espectro de secuencias finitas.

2.5.2 Métodos de clasificación

Clasificador de Bayes (CB)

El clasificador de Bayes [44, 46], minimiza el error de clasificación tomando en cuenta una función de riesgo promedio:

$$\begin{aligned}\mathcal{R} = c_{11}p_1 \int_{\mathcal{H}_1} p_{\mathbf{x}}(\mathbf{x}|\mathcal{C}_1)d\mathbf{x} + c_{22}p_2 \int_{\mathcal{H}_2} p_{\mathbf{x}}(\mathbf{x}|\mathcal{C}_2)d\mathbf{x} \\ +c_{21}p_1 \int_{\mathcal{H}_2} p_{\mathbf{x}}(\mathbf{x}|\mathcal{C}_1)d\mathbf{x} + c_{12}p_1 \int_{\mathcal{H}_1} p_{\mathbf{x}}(\mathbf{x}|\mathcal{C}_2)d\mathbf{x},\end{aligned} \tag{2.88}$$

donde c_{ij} es el costo de seleccionar la clase C_i cuando C_j es el correcto, p_i es la probabilidad apriori de $\mathbf{x}$, $\mathcal{H}_i$ es el espacio de observaciones de la clase i y $p_{\mathbf{x}}(\mathbf{x}|\mathcal{C}_i)$ es la pdf condicional de sus argumentos. Desarrollando la (2.88) se tiene,

$$\mathcal{R} = c_{21}p_1 + c_{22}p_2 + \int_{\mathcal{H}_1} [p_2(c_{12} - c_{22})p_{\mathbf{x}}(\mathbf{x}|\mathcal{C}_2) - p_1(c_{21} - c_{11})p_{\mathbf{x}}(\mathbf{x}|\mathcal{C}_1)] \tag{2.89}$$

analizando la (2.89) se define la siguiente condición,

$$\mathcal{T} = p_2(c_{12} - c_{22})p_{\mathbf{x}}(\mathbf{x}|\mathcal{C}_2) < p_1(c_{21} - c_{11})p_{\mathbf{x}}(\mathbf{x}|\mathcal{C}_1) \tag{2.90}$$

la cual asigna a $\mathbf{x}$ a la clase $\mathcal{C}_1$ si la condición es cierta o de lo contrario se asigna a la clase $\mathcal{C}_2$. Definiendo un umbral y una función de verosimilitud de la (2.90) se obtiene:

$$\Lambda(\mathbf{x}) > \xi \tag{2.91}$$

$$\Lambda(\mathbf{x}) = \frac{p_{\mathbf{x}}(\mathbf{x}|\mathcal{C}_1)}{p_{\mathbf{x}}(\mathbf{x}|\mathcal{C}_2)} \tag{2.92}$$

$$\xi = \frac{p_2(c_{12} - c_{22})}{p_1(c_{21} - c_{11})} \tag{2.93}$$

Bajo la suposición de un ambiente Gaussiano, el clasificador de Bayes se reduce a un clasificador lineal de la siguiente forma

$$\log \Lambda(\mathbf{x}) = (\mu_1 - \mu_2)^T \mathbf{C}^{-1}\mathbf{x} + \frac{1}{2}(\mu_2^T \mathbf{C}^{-1}\mu_2 - \mu_1^T \mathbf{C}^{-1}\mu_1) \tag{2.94}$$

$$\log \xi = 0 \tag{2.95}$$

$$y = \mathbf{w}^T \mathbf{x} + b \tag{2.96}$$

Donde,

$$y = \log \Lambda(\mathbf{x}) \tag{2.97}$$

$$\mathbf{w} = (\mathbf{C}^{-1})^T (\mu_1 - \mu_2) \tag{2.98}$$

$$b = \frac{1}{2}(\mu_2^T \mathbf{C}^{-1} \mu_2 - \mu_1^T \mathbf{C}^{-1} \mu_1) \tag{2.99}$$

Máquina de soporte vectorial (SVM)

La máquina de soporte vectorial consiste en la construcción de un hiperplano, basándose en un subconjunto de datos de entrenamiento que maximizan la distancia entre ambas clases, con los cuales se establece una superficie de decisión [44].

Para el caso de patrones linealmente separables un hiperplano se describe por:

$$y = \mathbf{w}^T \mathbf{x} + b, \tag{2.100}$$

Con el siguiente criterio de clasificación

$$\begin{aligned} \mathbf{w}^T \mathbf{x}_i + b \geq +1 \quad \text{para} \quad d_i = +1 \\ \mathbf{w}^T \mathbf{x}_i + b \leq -1 \quad \text{para} \quad d_i = -1 \end{aligned}$$

el cual se puede formular de la siguiente manera

$$d_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \quad \text{para} \quad i = 1, 2, \ldots, N \tag{2.101}$$

donde, d_i es la clase a la que pertenece $\mathbf{x}_i$ y el margen de separación (ρ) esta dado por

$$\rho = \frac{2}{\|\mathbf{w}_o\|}, \tag{2.102}$$

Con esta información se puede formular una función Lagrangiana tomando en cuenta una funcion de costo de $\mathbf{w}$ análoga a maximizar (2.102) y las restricciones previamente definidas de la ecuación (2.101).

$$\mathbf{J}(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \alpha_i[d_i(\mathbf{w}^T\mathbf{x}_i + b) - 1], \tag{2.103}$$

donde α_i son multiplicadores de Lagrange, los cuales son positivos $\forall i$. La solución del problema de optimización-restricción se obtiene en un punto silla, por lo que derivando (2.103) con respecto a $\mathbf{w}$ y b se obtiene

$$\mathbf{w} = \textstyle\sum_{i=1}^{N} \alpha_i d_i \mathbf{x}_i \quad \sum_{i=1}^{N} \alpha_i d_i = 0$$

Sustituyendo en (2.103)

$$\mathcal{Q}(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j d_i d_j \mathbf{x}_i^T\mathbf{x}_j \tag{2.104}$$

Debido a que la (2.104) depende puramente de α, se puede utiizar $\mathcal{Q}(\alpha)$ como función objetivo para determinar los multiplicadores de Lagrange y así calcular $\mathbf{w}$. Por ultimo el bias se calcula utilizando los datos de algún vector de soporte

$$b = -d_j + \sum_{i=1}^{N} \alpha_i d_i \mathbf{x}_i^T\mathbf{x}_j \tag{2.105}$$

Para el caso linealmente no separable, se realiza una transformación no lineal ($\phi(\mathbf{x}_i)$) con el fin de aumentar la dimensión del problema y así encontrar un hiperplano que finalmente separe las dos clases, Se define una función de kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi^T(\mathbf{x}_i)\phi(\mathbf{x}_j) \tag{2.106}$$

Aplicando la transformación se obtiene:

$$\mathcal{Q}(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j d_i d_j \phi^T(\mathbf{x}_i)\phi(\mathbf{x}_j) \tag{2.107}$$

$$\mathcal{Q}(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j d_i d_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{2.108}$$

Donde $0 \leq \alpha \leq C$ y C es definido positivo. El truco del kernel consiste en:

- El determinar un kernel especifico es una condición suficiente.
- No se requiere conocer el vector de pesos $\mathbf{w}$.
- Basta con conocer la matriz simétrica $\mathbf{K}$ definida como

$$\mathbf{K} = \{k(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1}^{N} \tag{2.109}$$

donde $k(\mathbf{x}, \mathbf{x}_i)$ es un kernel de Mercer, el cual puede seleccionarse a partir de la siguiente tabla.

Tipo de SVM	Kernel de Mercer ($k(\mathbf{x}, \mathbf{x}_i)$)
Función de base radial	$\exp(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}_i\|^2)$
Polinomio	$(\mathbf{x}^T\mathbf{x} + 1)^p$
Perceptrón de dos-capas	$\tanh(\beta_0\mathbf{x}^T\mathbf{x} + \beta_1)$

Tabla 2.1: Tabla de Kernel de Mercer más comunes en la literatura [44].

Perceptrón Multicapa (MLP)

Nuevamente se aborda la metodología de RNA perceptrón multicapa pero a diferencia de la sección 2.4.2, ésta sección está enfocada en la clasificación, problema que se diferencia por tener una salida que solo puede tomar valores de un conjunto finito de categorías posibles, mientras que en la sección pasada se abordaba el problema de la identificación, donde la salida podría tomar cualquier valor del conjunto de los reales.

Entre las razones para explorar nuevamente esta metodología se encuentra: su capacidad de clasificar tanto datos con relaciones linealmente separables como no linealmente separables; además de aprender las relaciones directamente de los datos que están siendo modelados. También cumple restricciones de tiempo real, que son una característica importante para este trabajo. Como pioneros en el desarrollo de control con señales fisiológicas en tiempo real basada en el reconocimiento patrones,

en [48] utilizaron un perceptrón multicapa (MLP) para clasificar características en el dominio del tiempo, y fueron capaces de clasificar cuatro tipos de movimiento de las extremidades, con una tasa de error de alrededor del 10 %. Posteriormente, en [49] aplicaron una red neural MLP para reconocer seis patrones de movimiento basados en una características en el dominio del tiempo y su entropía, lograron una precisión promedio de aproximadamente el 95 %. La técnica de diseño y entrenamiento es descrita en la sección 2.4.2 (algoritmo 5 y Figura 2.18).

Capítulo 3

Diseño, análisis comparativo y simulación

El objetivo de este capítulo es simular, comparar y seleccionar los métodos presentados en el capítulo 2 para diseñar el sistema propuesto en la hipótesis (como se muestra en la Figura 1.10) y posteriormente seguir a la implementación de cada etapa; iniciando con la Sección 3.1, donde se describen los protocolos de experimentación, los cuales presentan los datos necesarios para diseñar el sistema digital. La Sección 3.2 está destinada a detallar el diseño, la cual se divide en la Sección 3.2.1, donde se simulan los algoritmos propuestos para la etapa de pre-procesamiento de las señales fisiológicas y se define el método que se considera para el diseño posterior. Posteriormente en la Sección 3.2.2 se define el método que se implementará para la etapa de decodificación, esto se realiza mediante la simulación de los métodos propuestos y tomando en cuenta la complejidad computacional y el error cuadrático promedio normalizado (NMSE) para seleccionar la técnica indicada. En la Sección 3.2.3, se describen los diferentes tipos de buffers (ventana) a considerar para el procesamiento de la señal cerebral, se simulan los diferentes tipos de clasificadores y se selecciona uno de ellos, partiendo del porcentaje de aciertos y la complejidad de la implementación. Por último, en la Sección 3.3 se simula la hBMI y se determina si es factible la implementación del mismo.

3.1 Protocolo de experimentación

En ésta tesis se considera datos experimentales realizados anteriormente por el cuerpo académico del DIE de FIME-UANL [51, 52].

3.1.1 Señal EEG

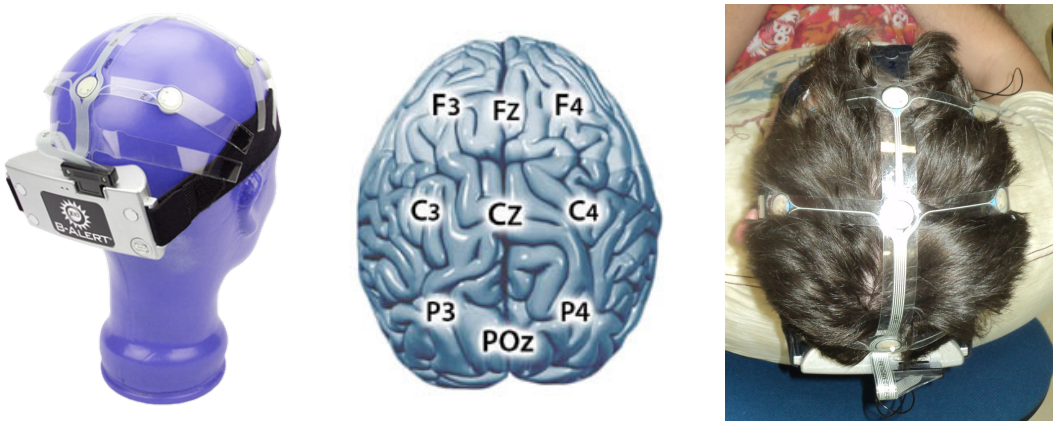
La señal EEG fue adquirida de un experimento [51] que involucra a un paciente de 25 años que no presentaba ninguna patología física o neural relacionada a los miembros inferiores, a este sujeto se le pidió que caminara a dos distintas velocidades en la caminadora "XTERRA trail racer 3.0"; la primera velocidad se definió de acuerdo a la velocidad con la que el sujeto camina normalmente (aprox. 1.43 segundos por ciclo); posteriormente se incremento la velocidad a 1.13 segundos por ciclo (esta velocidad sigue siendo confortable para el sujeto y no se considera como trotar).

Figura 3.1: Fotografía del experimento para la adquisición de la señal EEG [51].

Al iniciar el experimento (Figura 3.1) se le solicitó al sujeto parpadear tres veces (como marcador para el procesamiento posterior), después se le pidió al sujeto

caminar en la primera velocidad durante un minuto, posteriormente se notifica al paciente de manera verbal el cambio de velocidad (la velocidad fue aumentando hasta alcanzar la velocidad 2), una vez que el sujeto alcanzara la velocidad 2 se le pidió que parpadeara tres veces, después de un minuto en la velocidad 2, nuevamente se le informo verbalmente al sujeto el cambio de velocidad (disminuyendo la velocidad hasta alcanzar la velocidad 1), una vez que el sujeto alcanzara la velocidad 1 se le pidió que parpadeara tres veces, el experimento concluye después de un min. Este experimento se repitió un total de 10 veces con descansos de 10 a 15 segundos [51].

Se utilizó el dispositivo B-ALERT X10 de Advanced Brain Monitoring Ltd tal como se muestra en la Figura 3.2; para la adquisición de las señales EEG, este dispositivo opera de manera wireless, posee 9 canales para EEG y el décimo canal para EOG o ECG, adquiere las señales con una frecuencia de muestreo de 256 Hz, tiene una resolución de 16 bits y rango ± 1000 μV [54]:



(a) (b) (c)

Figura 3.2: Adquisición de la señal EEG. a) Sistema B-ALERT X10. b) Esquema de la posición de los electrodos del sistema. c) Fotografía del sistema sobre el paciente [51].

En este experimento se registraron los 10 canales disponibles, con un preprocesamiento preestablecido para eliminar artefactos oculares. Para este trabajo se consideró el electrodo C3 (debido a que esa área está relacionada con la acti-

vidad motriz) junto con la señal EOG que es usada como marcador del estado del experimento tal como en la Figura 3.3, la cual muestra un registro de una repetición.

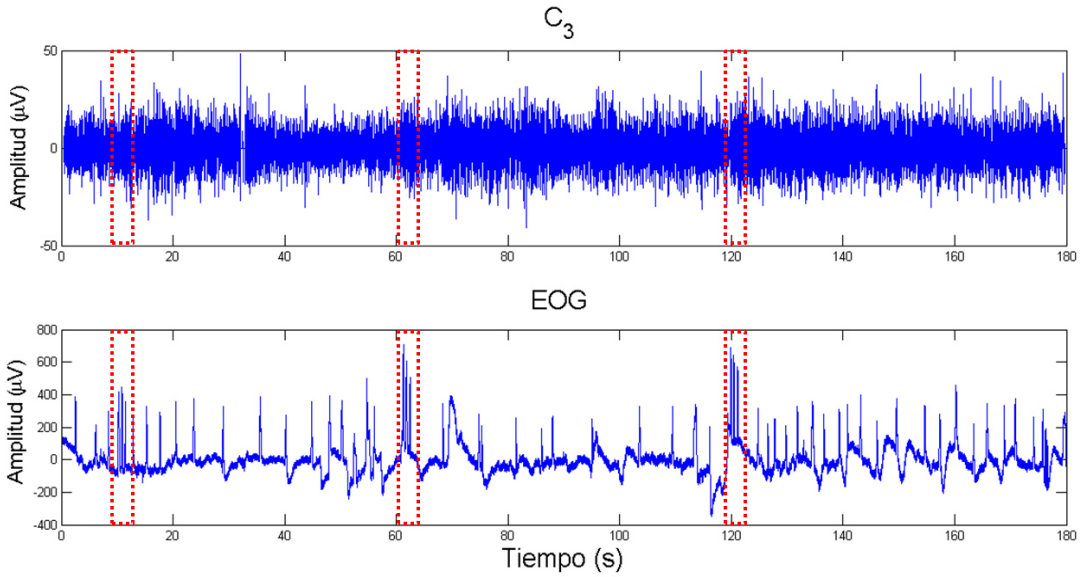


Figura 3.3: Señales EEG del electrodo C3 y EOG de un experimento, el cuadro rojo marca los tres parpadeos que indican el cambio de actividad del paciente [51].

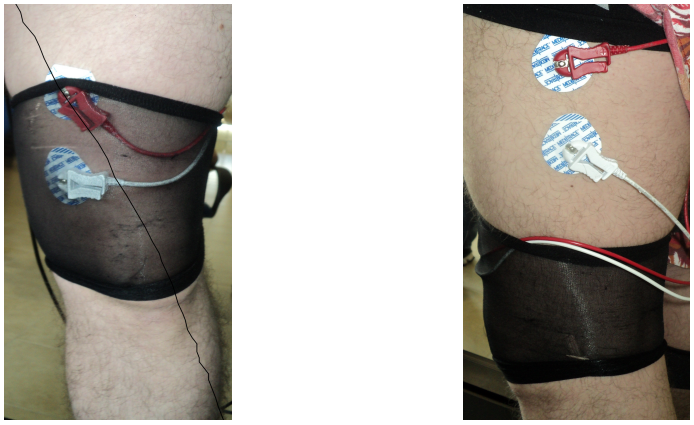
3.1.2 Señal EMG y variable cinemática

Para el registro de la actividad de las señales EMG, y la variable cinemática que para este trabajo corresponde al ángulo tibio-femoral (TFA), se realizó un experimento similar al descrito anteriormente, en este experimento se usaron tres pacientes que realizaron varias caminatas en la XTERRA trail racer 3.0, estos pacientes no presentaron ningún historial de problemas fisiológicos que afecten a la prueba.

El experimento consiste en registrar dos velocidades diferentes de caminata, para la cual se le solicito al paciente que parpadeara tres veces al cambio de la velocidad de 4 millas/hora a 6 millas/hora y viceversa (se cambió la velocidad dos veces por prueba), con el fin de sincronizar las señales en la etapa de procesamiento, cada velocidad tuvo una duración de 1 min [52].

Para adquirir las señales EMG se utilizo el sistema MP36 de BIOPAC tal

como se muestra en la Figura 3.4, usando electrodos desechables EL503. El sistema adquiere las señales con una frecuencia de muestreo de 1000Hz, con resolución de 24 bits, y con rango ajustable entre 2000 μV y 2 V [55].



(a) (b)

Figura 3.4: Sujeto usando los electrodos EL503 que se conectan al sistema MP36 colocados en el a) *Vastus medialis* (extensor) y en b) *Bícep femoris* (flexor) [52].

Para este estudio se consideró la señal mioeléctrica del músculo extensor debido a que ambas señales se consideran como complementarias una de otra; en la Figura 3.5 se muestra el registro completo de la señal EMG de un experimento y en la Figura 3.6 se muestra la señal mioeléctrica en cada velocidad.

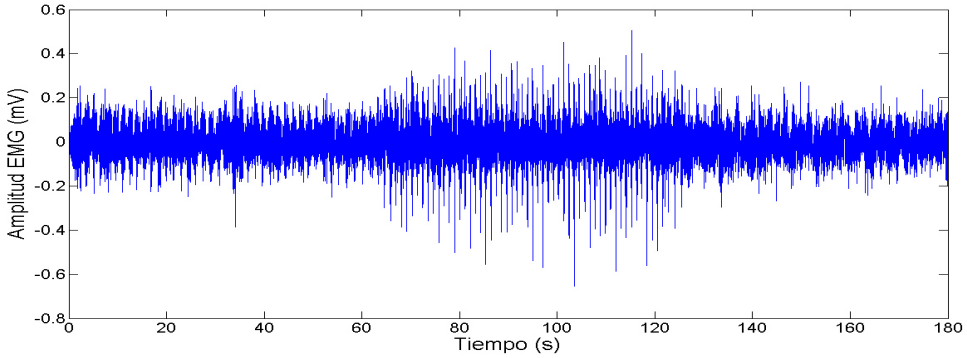
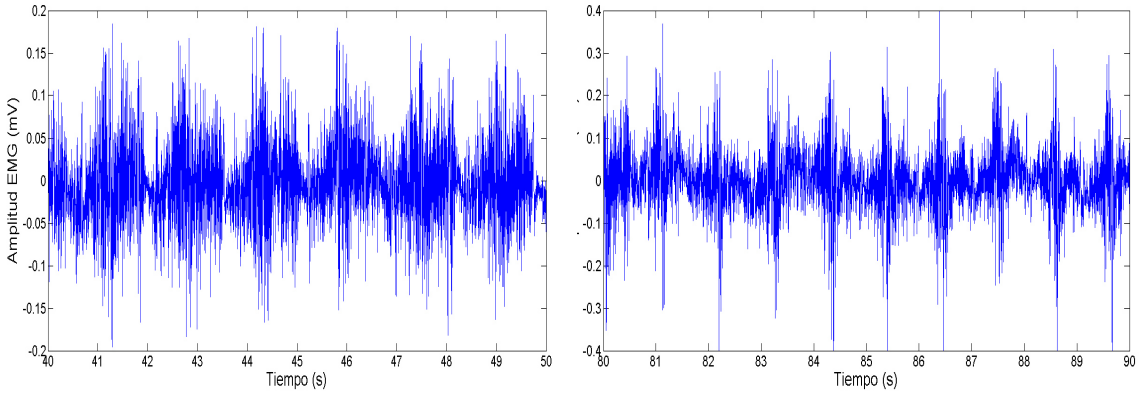


Figura 3.5: Señal EMG durante todo un experimento [52].



(a) (b)

Figura 3.6: Registro de actividad EMG durante: a) velocidad 1 (4 mph) y b) velocidad 2 (6 mph) [52].

En el caso de la adquisición del TFA, se utilizó una cámara de video Sony Handyman HDR-CX190, la cual registraba la posición de unos índices blancos (sobre un fondo negro como se muestra en la Figura 3.7), posteriormente se procesa el video fuera de línea y se calcula el TFA. En [52] se describe con mayor profundidad el método para calcular la VC partiendo del video. Este sistema adquiere las señales con una frecuencia de muestreo de 29 cuadros/seg.

(a) (b)

Figura 3.7: Adquisición del TFA (θ). a) Ambiente del experimento para grabar del video. b) Convención de los índices (cadera, rodilla, tobillo) [52].

En la Figura 3.8 se muestra el registro completo del TFA de un experimento y en la Figura 3.9 se muestra dicho ángulo en cada velocidad.

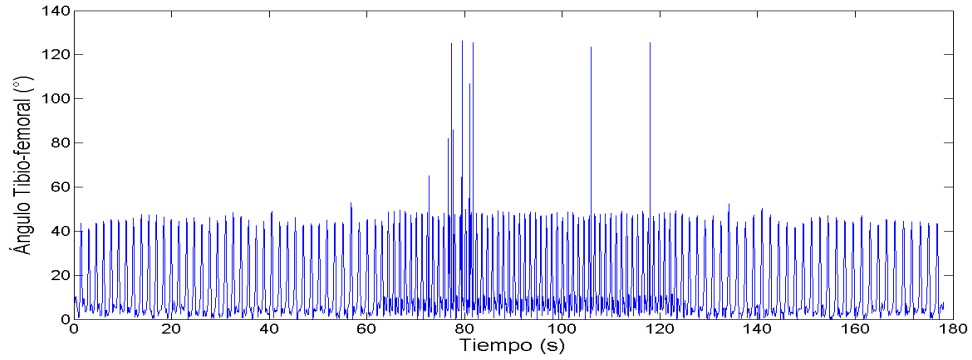
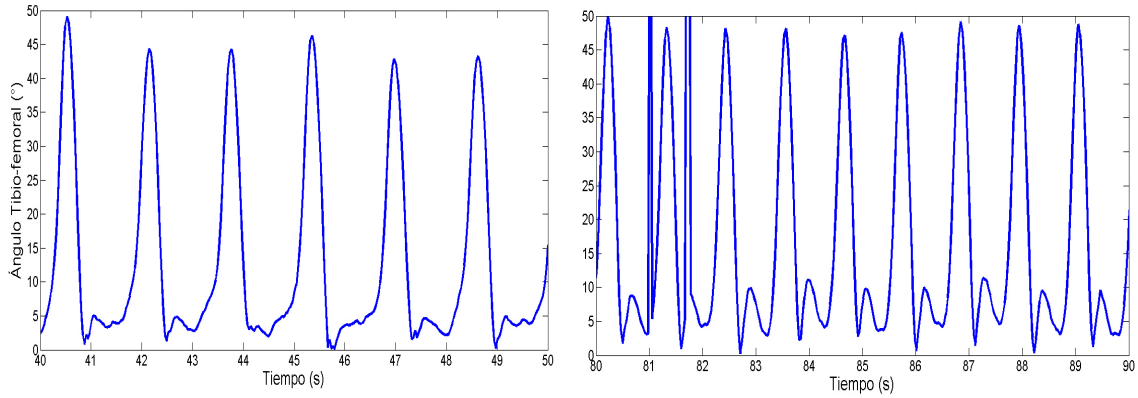


Figura 3.8: Ángulo tibio-femoral durante todo un experimento [52].



(a) (b)

Figura 3.9: Registro de actividad del TFA durante: a) velocidad 1 (4 mph) y b) velocidad 2 (6 mph) [52].

3.2 Diseño y análisis comparativo

El diseño del sistema digital se realizó en la plataforma *MATLAB*® de Mathworks, mientras que la simulación del sistema se realizó en el módulo *Simulink*®.

3.2.1 Diseño del pre-procesamiento

Pre-procesamiento de EMG

Resumiendo lo discutido en la Sección 2.3.1, se propuso calcular la envolvente suavizada de la señal EMG para posteriormente usarla como entrada para la etapa de decodificación. Para este trabajo se evaluaron los siguientes métodos:

- Método de la trasformada Hilbert que consiste en:
  - Transformada Hilbert (Calcular la señal analítica).
  - Módulo de la señal.
  - Filtro Butterworth pasa bajas de 5th orden con $f_c = 5$ Hz.
- Método de filtrado lineal que consiste en:
  - Filtro Butterworth pasa altas de 5th orden con $f_c = 15$ Hz.
  - Módulo de la señal.
  - Filtro Butterworth pasa bajas de 5th orden con $f_c = 5$ Hz.

En la Figura 3.10 se muestra una comparación gráfica de los dos métodos.

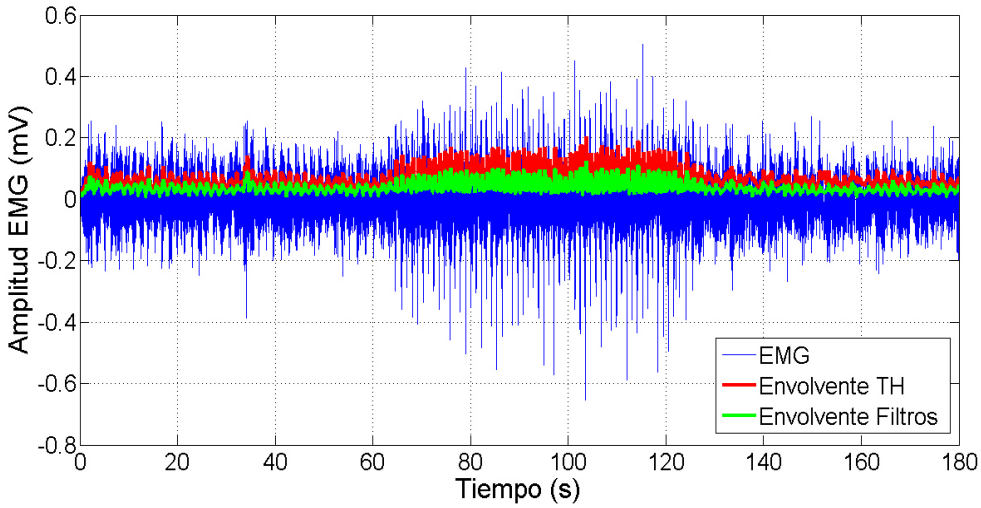
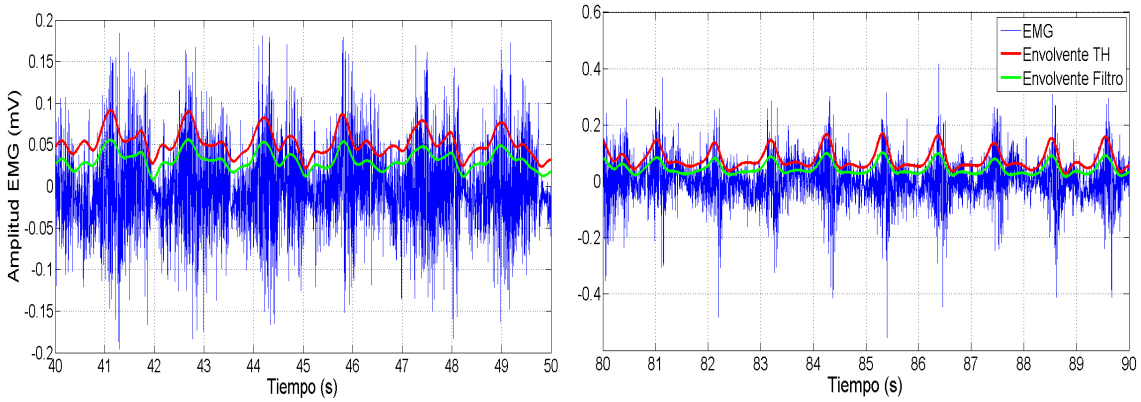


Figura 3.10: Envolvente lineal de una EMG.

Estos métodos tienen un retraso aproximado de 300 muestras, dado que la señal fue muestreada a 1000Hz, el retraso se aproxima a 0.3 seg. El resultado de despreciar el retraso inherente del uso de filtros se muestra en la Figura 3.11. En



(a) (b)

Figura 3.11: Envolvente lineal de una EMG en: a) velocidad 1 (4 mph) y b) velocidad 2 (6 mph).

conclusión el método de la transformada Hilbert cubre de mejor manera la señal EMG en comparación del método tradicional de filtrado, sin embargo este último, puede ser configurado en los sistemas de adquisición de datos actuales sin problemas.

El código de la implementación de estos métodos puede encontrarse en el Apéndice A.1, para desarrollar dicho código se usaron comandos como “butter” (comando para diseñar un filtro Butterworth) y “hilbert” (el cual calcula la señal analítica de su argumento).

Pre-procesamiento de EEG

En la Sección 2.3.2 se presenta el algoritmo ICA para discriminar artefactos, en este trabajo se usó un toolbox llamado FastICA desarrollado por [53], el cual es un método eficiente para efectuar el algoritmo ICA.

Discriminación de artefactos de la señal EEG

Como se mencionó en la Sección 3.1.1, la señal EEG fue pre-procesada para eliminar la señal EOG sobre los canales EEG, por lo cual esta etapa no fue incluida en el sistema digital final, sin embargo en esta tesis se presenta un ejemplo de la aplicación del algoritmo ICA para remover los componentes EOG sobre los canales EEG.

El experimento que contempla este ejemplo fue la adquisición de la señal EEG mientras el sujeto de prueba levantaba su tobillo hasta que el TFA tuviera un ángulo aproximado de 180° (el TFA parte de 90°) mientras se mantiene sentado, para adquirir la señal se usó el sistema MOBITA-W-20 EEG y se adquirió la señal EEG con artefactos oculares (EOG).

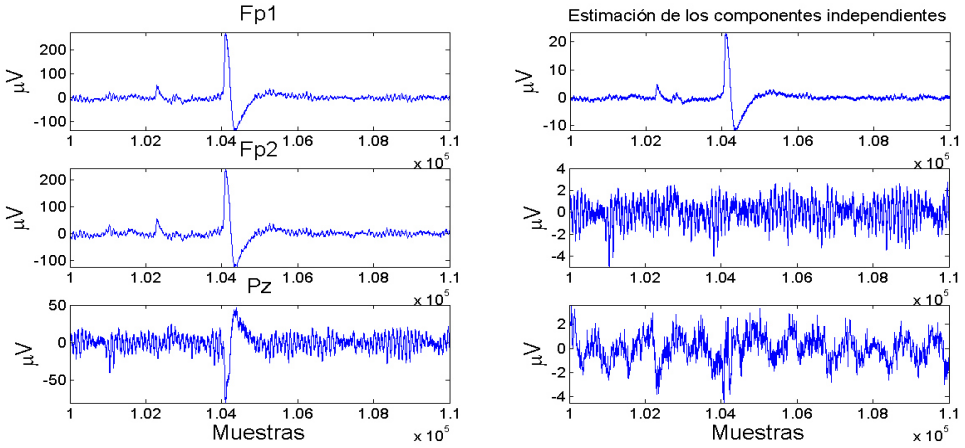


Figura 3.12: Discriminación de componentes de la señal EEG del experimento completo.

En la Figura 3.12 se muestra del lado izquierdo las variables observadas en un intervalo donde el sujeto de prueba parpadeó, Fp1 corresponde al electrodo frontal 1, Fp2 es el electrodo frontal 2 (ambas señales registran la señal EOG) y Pz el electrodo en la zona central del lóbulo parietal (señal EEG con parpadeo).

De lado derecho se muestra el resultado de aplicar el algoritmo ICA, separando las señales estadísticamente independientes. Como se puede apreciar en la primera gráfica del lado derecho corresponde a la señal EOG, la segunda gráfica es la señal EEG (Pz) sin el parpadeó y la tercera gráfica es una señal de artefactos independientes de la señal EEG y EOG. Como se mencionó anteriormente, la desventaja de esta técnica es el alterar la magnitud de la señales originales, sin embargo usando la matriz de mezclado A y generando un nuevo vector de señales originales (reemplazando los componentes no deseados por ceros) es posible recuperar las señales observadas sin los componentes no deseados, como se muestra en la Figura 3.13.

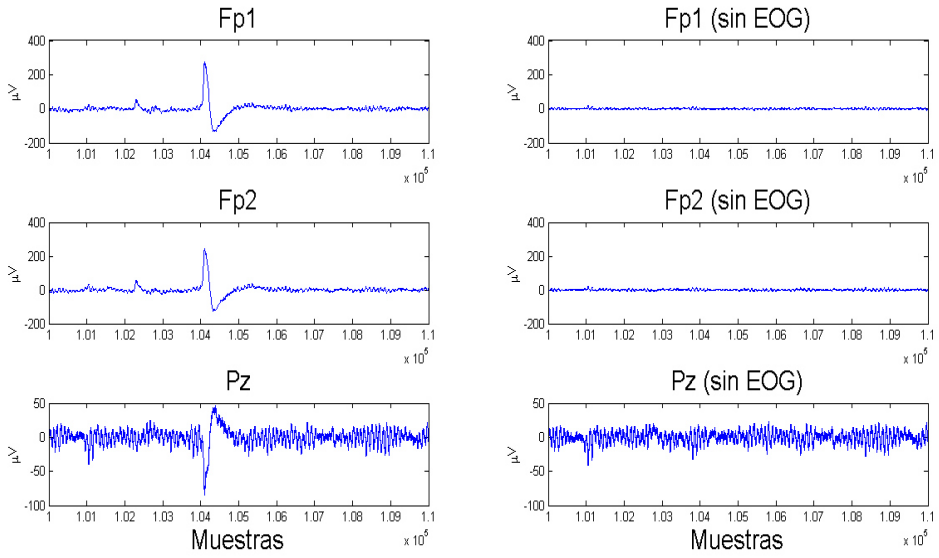


Figura 3.13: Señal EEG (sin los componentes EOG) del experimento completo.

En conclusión el algoritmo ICA es capaz de eliminar el ruido por parpadeo.

3.2.2 Diseño del decodificador

Haciendo referencia a la Sección 2.4, este trabajo aborda los métodos de filtro óptimo, filtros adaptativos y redes neuronales artificiales. Por lo tanto se define la señal deseada como el TFA y la envolvente de la señal EMG del músculo extensor como la entrada de esta etapa.

Dado que las tres señales fueron adquiridas a diferente frecuencia de muestro (f_s), se define la frecuencia de muestro de nuestro sistema digital a 256 Hz, debido a esta especificación, la señal del TFA (f_s = 29 Hz) requiere de una interpolación y la señal EMG (f_s = 1000) Hz requiere de decimación; para realizar la interpolación se usa la técnica denominada como "zero padding" (Apéndice A.1) el cual consite en agregar muestras de ceros al espectro de la señal original y transformar el espectro resultante al dominio del tiempo, dando como resultado una interpolación con carácter sinusoidal. Para realizar la decimación se aplicó el algoritmo descrito en el Apéndice A.1, el cual elimina muestras conforme a un factor y así reducir la frecuencia de muestreo. La señal TFA fue normalizada entre 34.26 y -156.27, mientras que la señal EMG se normalizó entre los valores 0.2 y 2.45e-12; Por último, a ambas señales se les aplicó un offset para hacerlas de media cero. Para realizar el entrenamiento o los ciclos de adaptación, se seleccionó un intervalo de las señales EMG y TFA para cada velocidad de operación, para así obtener mejores resultados.

En las Figuras 3.14 y 3.15 se muestran los intervalos usados para el diseño de esta etapa.

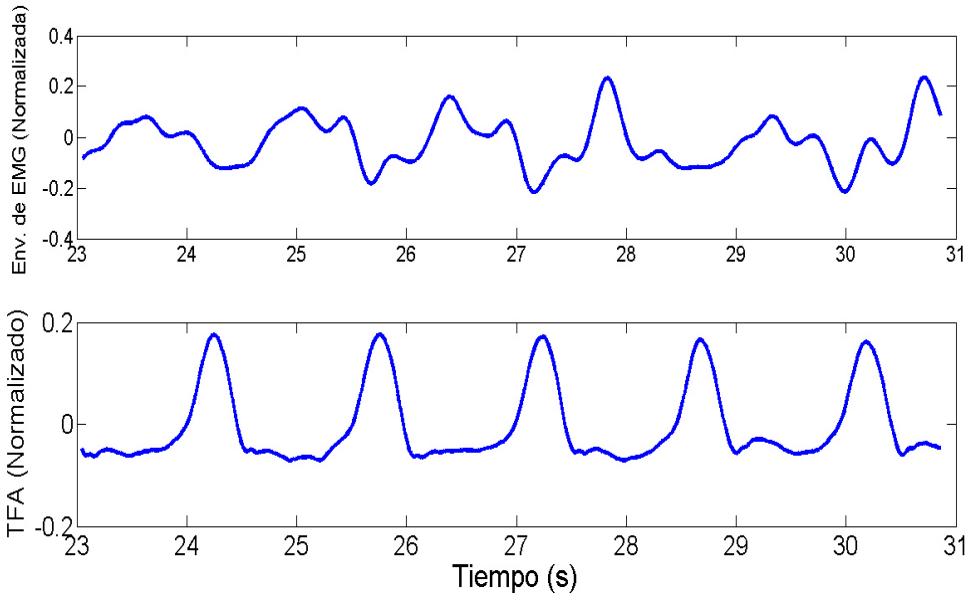


Figura 3.14: Intervalo de las señales EMG y TFA para la velocidad 1.

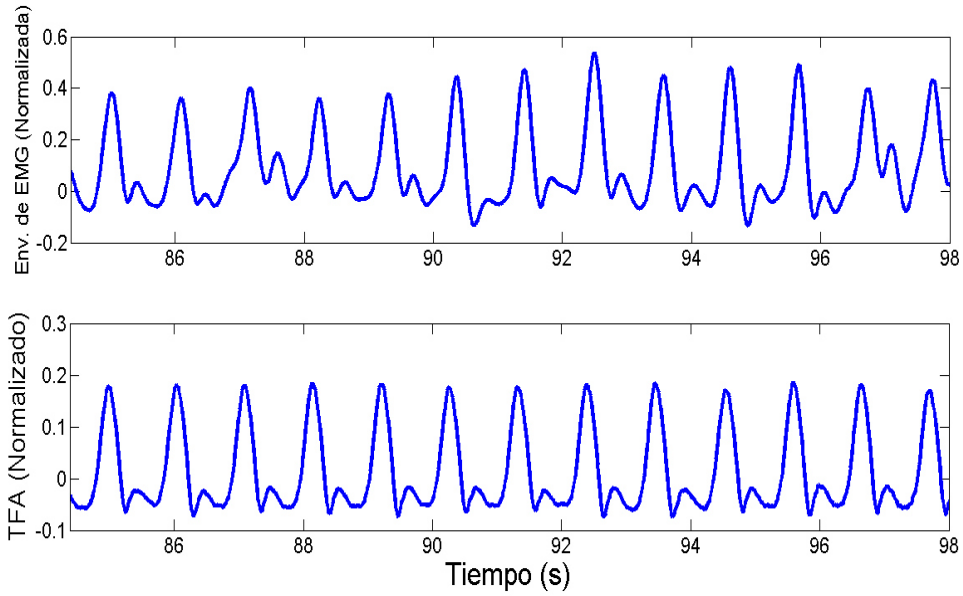


Figura 3.15: Intervalo de las señales EMG y TFA para la velocidad 2.

Filtro Wiener

La deducción de esta técnica se discute en la Sección 2.4.1, mientras que el código empleado se describe en el Apéndice A.1. Las características del diseño son los siguientes:

Filtro Wiener de la velocidad 1:

- Orden: 500.
- Tamaño de la señal (N): 2001 muestras.
- Frecuencia de muestreo (f_s): 256 Hz.

En la Figura 3.16 se muestran los resultados del diseño usando un filtro Wiener sobre el intervalo seleccionado.

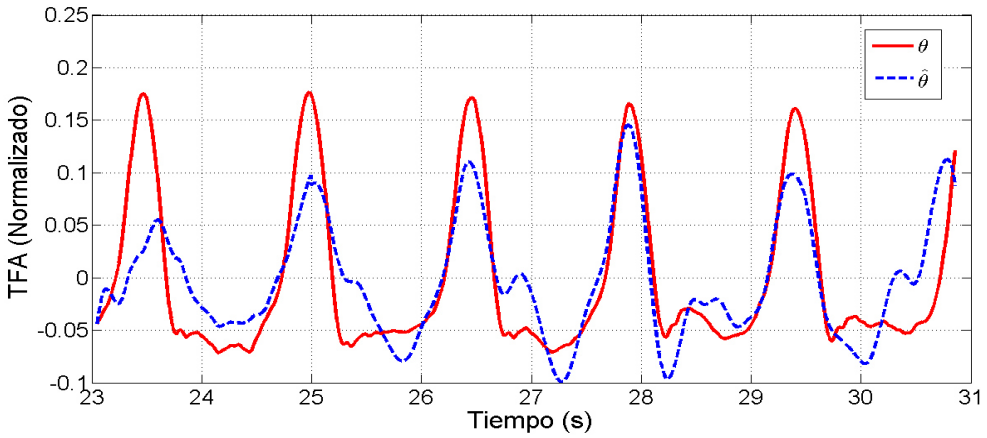


Figura 3.16: Intervalo decodificado por el filtro Wiener para la velocidad 1.

Filtro Wiener de la velocidad 2:

- Orden: 500.
- Tamaño de la señal (N): 4801 muestras.
- Frecuencia de muestreo (f_s): 256 Hz.

En la Figura 3.17 se muestran los resultados del diseño usando un filtro Wiener sobre el intervalo seleccionado.

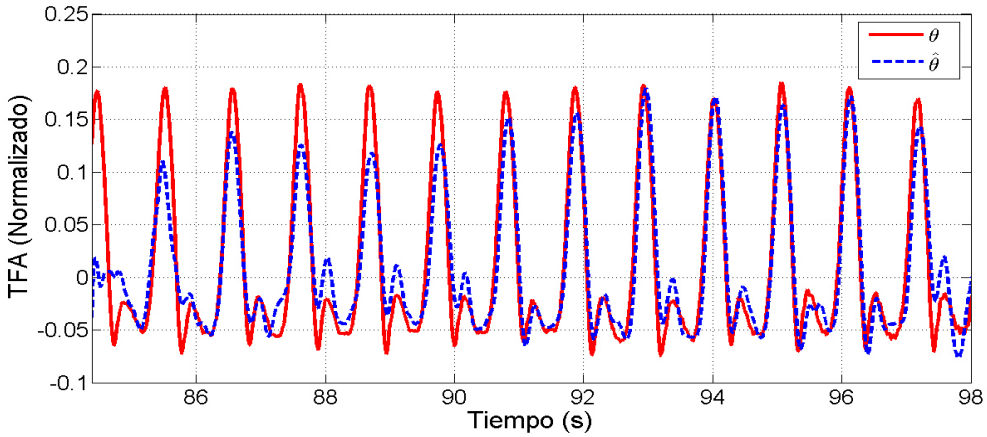


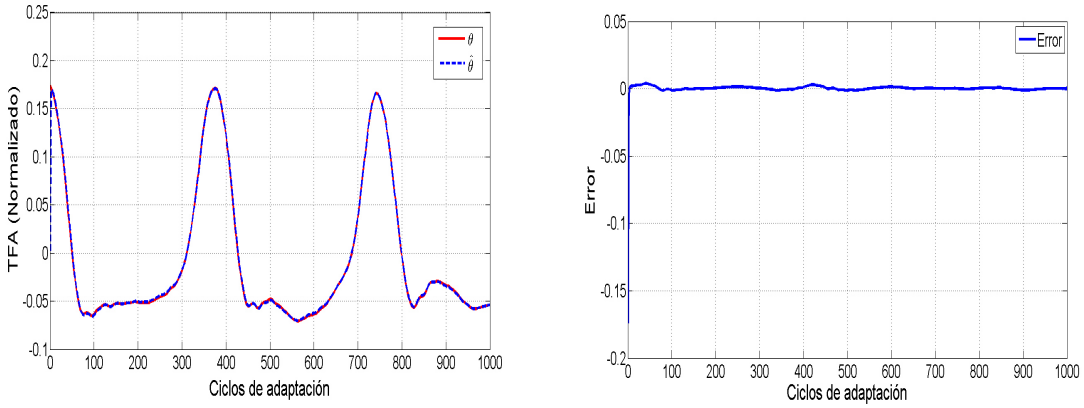
Figura 3.17: Intervalo decodificado por el filtro Wiener para la velocidad 2.

Filtro LMS

En la Sección 2.4.1, se describió el desarrollo de este método, además, se discute la cualidad de modificar los coeficientes del filtro por iteración; esta característica refuerza la perspectiva local sobre la global, es decir, el ciclo actual tiene una mayor influencia en el resultado que los anteriores, hasta el punto de poder despreciar las primeras iteraciones. Debido a esto se propuso el registrar los coeficientes del filtro por iteración y promediarlos para aumentar la perspectiva global. A continuación se presenta las características del diseño de los filtros, junto con la simulación en ambas actividades.

Filtro LMS de la velocidad 1:

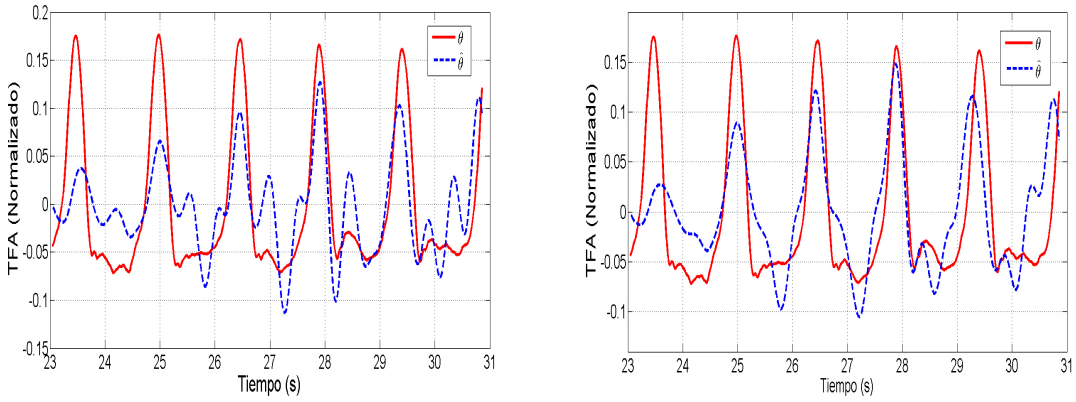
- Orden: 500.
- Tamaño de la señal (N): 2001 muestras.
- Frecuencia de muestreo (f_s): 256 Hz.
- Ciclos de adaptación: 1000.
- Tamaño del paso (μ): 0.324.



(a) (b)

Figura 3.18: Resultados durante el ciclo de adaptación de la velocidad 1. a) Estimación del ángulo durante y b) Estimación del ángulo durante el ciclo de adaptación.

En la Figura 3.18 se muestran los resultados del diseño usando un filtro LMS sobre el intervalo seleccionado. Y la simulación del diseño del filtro se muestra en la Figura 3.19.



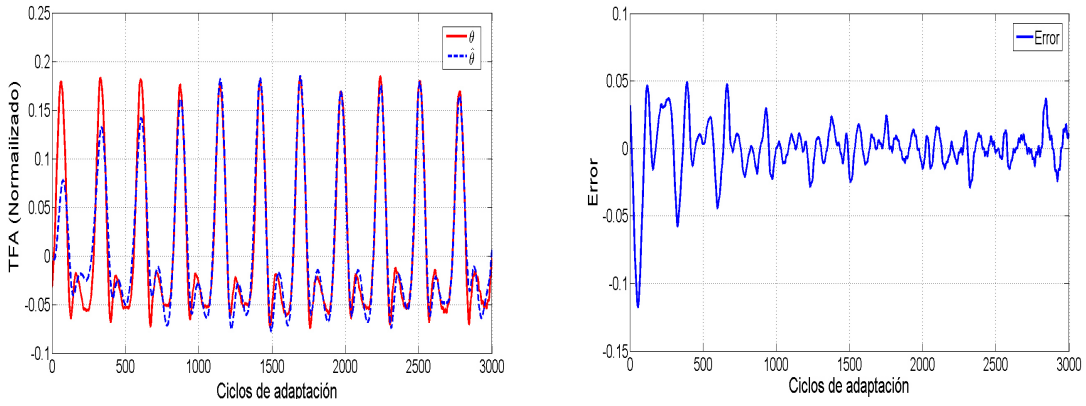
(a) (b)

Figura 3.19: Simulación sobre el intervalo de la velocidad 1: a) Filtro LMS y b) Filtro LMS promediado.

Filtro LMS de la velocidad 2:

- Orden: 500.
- Tamaño de la señal (N): 4801 muestras.
- Frecuencia de muestreo (f_s): 256 Hz.
- Ciclos de adaptación: 3000.
- Tamaño del paso(μ): 0.0017.

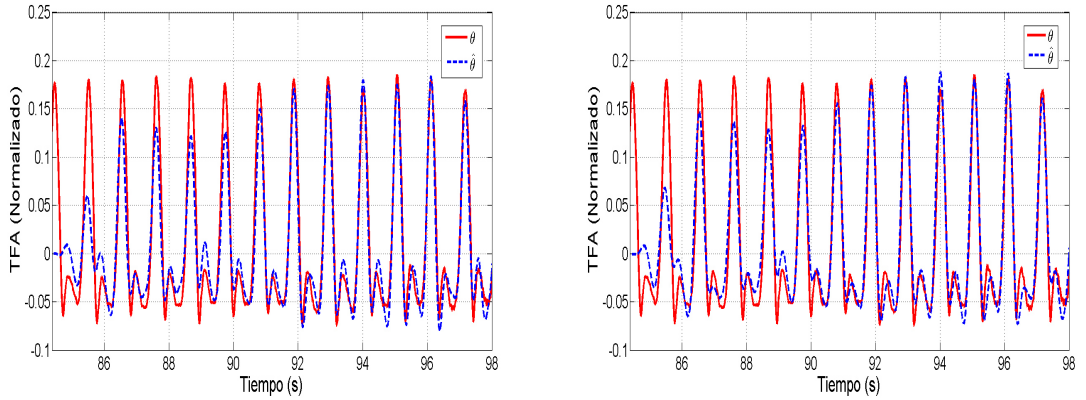
En la Figura 3.20 se muestran los resultados del diseño usando un filtro LMS sobre el intervalo seleccionado.



(a) (b)

Figura 3.20: Resultados durante el ciclo de adaptación de la velocidad 2. a) Estimación del ángulo y b) Estimación del ángulo durante el ciclo de adaptación.

Y la simulación del diseño del filtro se muestra en la Figura 3.21.



(a) (b)

Figura 3.21: Simulación sobre el intervalo de la velocidad 2: a) Filtro LMS y b) Filtro LMS promediado.

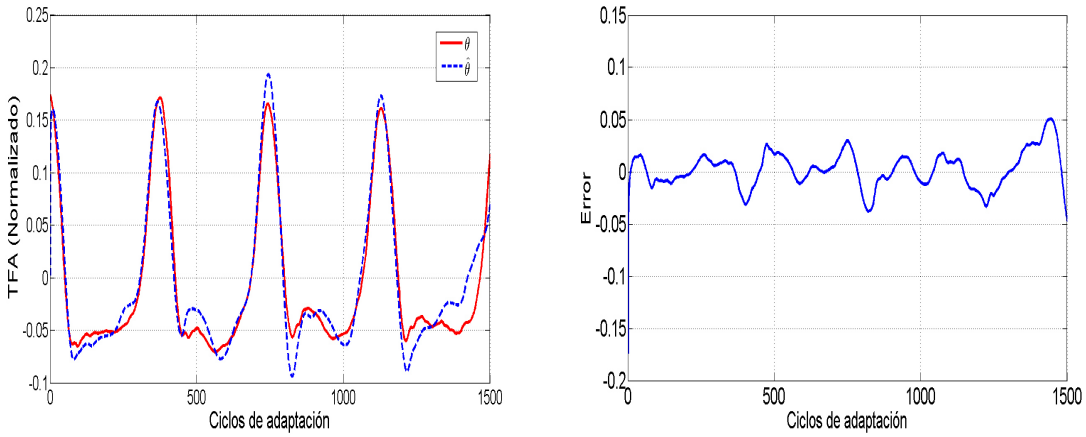
Para el lector interesado, el código que se usó para simular el método se puede encontrar en el Apéndice A.1.

Filtro RLS

La deducción de esta técnica se encuentra en la Sección 2.4.1, donde ademas se menciona que esta técnica posee un factor de olvido, por lo que presenta una mayor perspectiva global que el algoritmo LMS y por lo cual se optó por no promediar el registro de coeficientes como en el método anterior. A continuación se presenta las características del diseño de los filtros, junto con la simulación en ambas actividades.

Filtro RLS de la velocidad 1:

- Orden: 500.
- Tamaño de la señal (N): 2001 muestras.
- Frecuencia de muestreo (f_s): 256 Hz.
- Ciclos de adaptación: 1500.
- Factor de olvido (μ): 0.997.



(a) (b)

Figura 3.22: Resultados durante el ciclo de adaptación de la velocidad 1. a) Estimación del TFA y b) Error en la estimación durante el ciclo de adaptación.

En la Figura 3.22 se muestran los resultados del diseño usando un filtro RLS sobre el intervalo seleccionado. Y la simulación del diseño del filtro se muestra en la Figura 3.23.

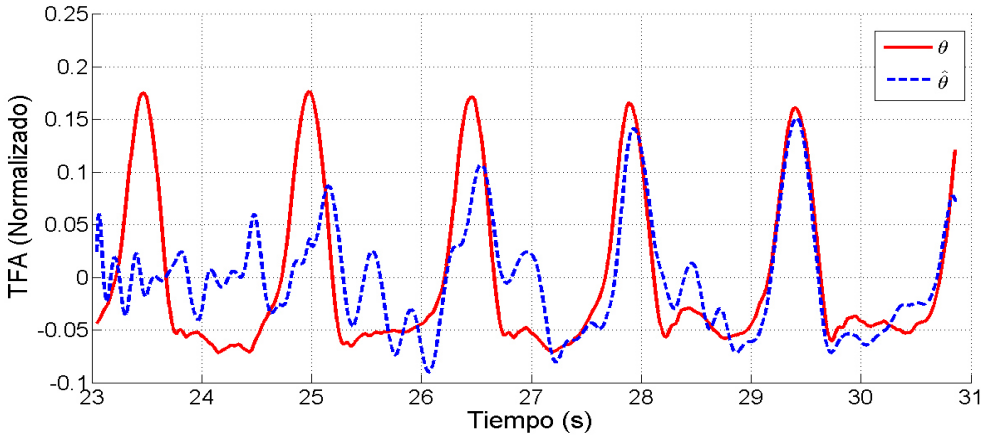
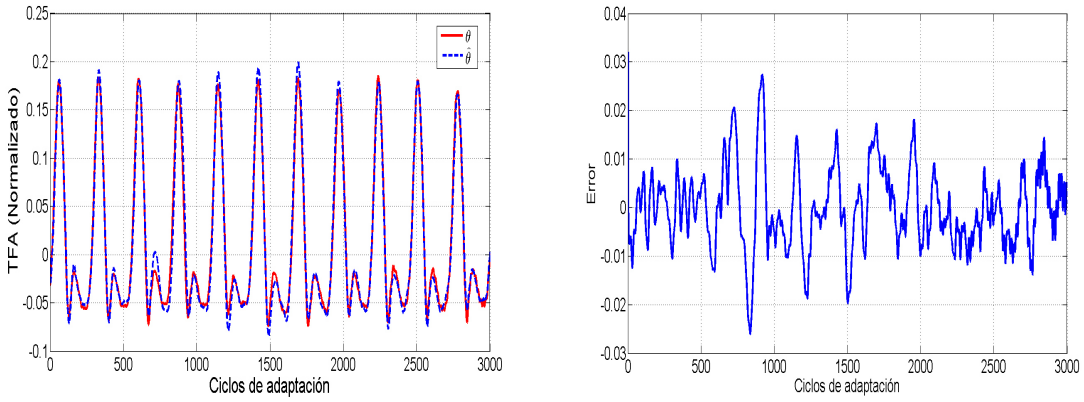


Figura 3.23: Intervalo decodificado por el filtro RLS para la velocidad 1.

Filtro RLS de la velocidad 2:

- Orden: 500.
- Tamaño de la señal (N): 2001 muestras.
- Frecuencia de muestreo (f_s): 256 Hz.
- Ciclos de adaptación: 3500.
- Factor de olvido (μ): 0.997.

En la Figura 3.24 se muestran los resultados del diseño usando un filtro RLS sobre el intervalo seleccionado.



(a) (b)

Figura 3.24: Resultados durante el ciclo de adaptación de la velocidad 2. a) Estimación del TFA y b) Error en la estimación durante el ciclo de adaptación.

Y la simulación del diseño del filtro se muestra en la Figura 3.25.

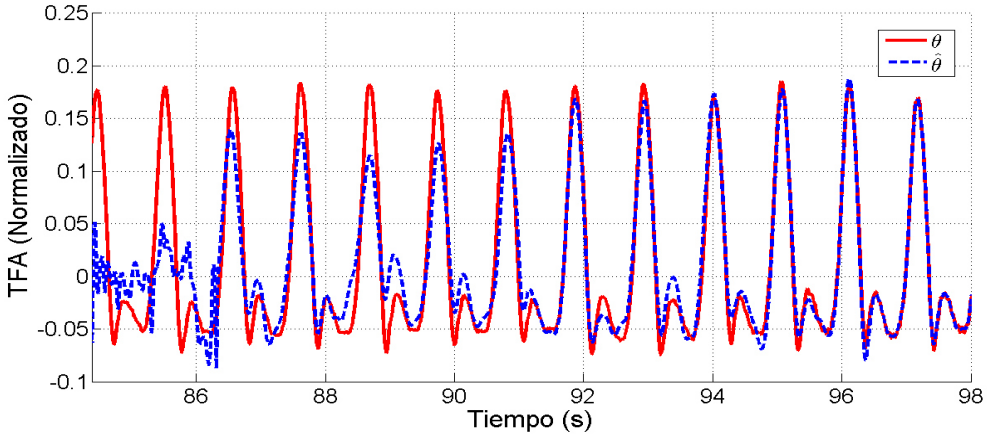


Figura 3.25: Intervalo decodificado por el filtro RLS para la velocidad 2.

Para el lector interesado, el código que se usó para simular el método se puede encontrar en el Apéndice A.1.

Perceptrón Multicapa (decodificador)

La metodología de MLP se aborda en la Sección 2.4.2, así mismo se menciona los dos tipos de paradigmas evaluados en esta tesis; la RNA estática y la RNA dinámica (con memoria incluida) cuya estructura se muestra en la Figura 2.16. El diseño de la red neuronal se realizó usando el toolbox de MATLAB conocido como "Neural Network Toolbox" ("nntool", comando para abrir el gestionador de RNAs, el cual permite crear y configurar RNA o importar y exportar datos para el diseño). Se configuró el gestionador para una RNA estática de la siguiente manera:

Características de RNA de la velocidad 1:

- Arquitectura:
  - Número de entradas: 1
  - Número de capas ocultas: 3
  - Número de neuronas en la capa 1: 20
  - Número de neuronas en la capa 2: 10
  - Número de neuronas en la capa 3: 5
  - Número de neuronas en la capa de salida: 1
- Tamaño de la señal (N): 2001 muestras
- Frecuencia de muestreo (f_s): 256 Hz
- Parámetros de entrenamiento:
  - Número de épocas: 1000
  - Gradiente mínimo: 1x10^{-7}
  - Tamaño del paso: 0.001
  - Número de épocas para la convergencia: 167

En la Figura 3.26 se muestra la simulación de la RNA sobre la velocidad 1.

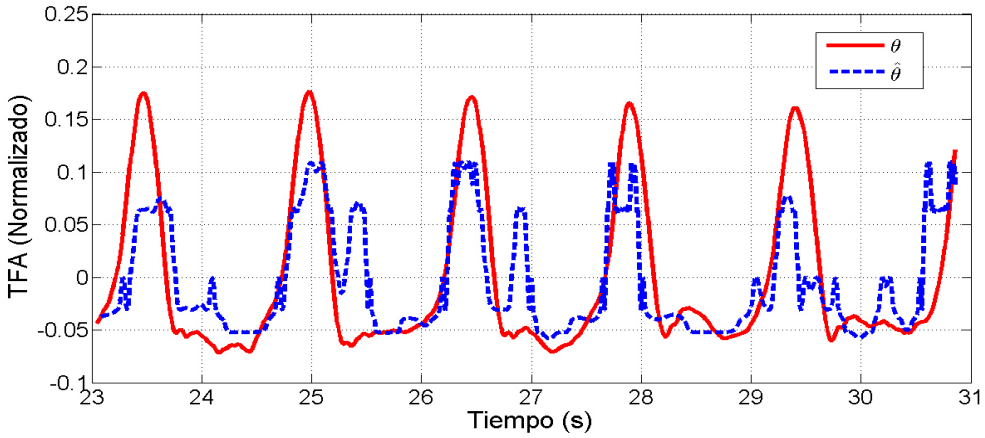


Figura 3.26: Intervalo decodificado por la RNA (1-20-10-5-1) para la velocidad 1.

Características de RNA de la velocidad 2:

- Arquitectura:
  - Número de entradas: 1
  - Número de capas ocultas: 3
  - Número de neuronas en la capa 1: 20
  - Número de neuronas en la capa 2: 10
  - Número de neuronas en la capa 3: 5
  - Número de neuronas en la capa de salida: 1
- Tamaño de la señal (N): 4801 muestras
- Frecuencia de muestreo (f_s): 256 Hz
- Parámetros de entrenamiento:
  - Número de épocas: 1000
  - Gradiente mínimo: 1x10^{-7}
  - Tamaño del paso: 0.001

- Número de épocas para la convergencia: 17

En la Figura 3.27 se muestra la simulación de la RNA sobre la velocidad 2.

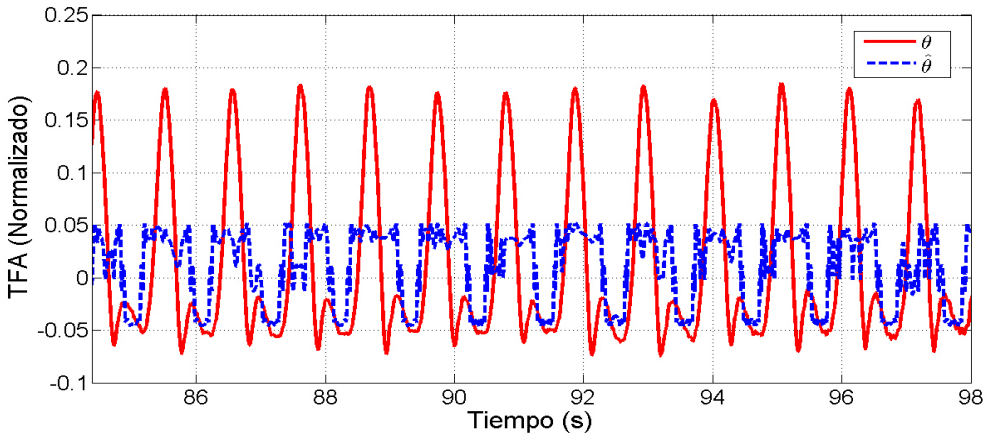


Figura 3.27: Intervalo decodificado por la RNA (1-20-10-5-1) para la velocidad 2.

A continuación se describe las cualidades de las RNA dinámica evaluadas.

Características de RNA de la velocidad 1:

- Arquitectura:
  - Número de retrasos en la entrada: 10
  - Número de entradas: 10
  - Número de capas ocultas: 3
  - Número de neuronas en la capa 1: 20
  - Número de neuronas en la capa 2: 10
  - Número de neuronas en la capa 3: 5
  - Número de neuronas en la capa de salida: 1
- Tamaño de la señal (N): 2001 muestras
- Frecuencia de muestreo (f_s): 256 Hz

- Parámetros de entrenamiento:
  - Número de épocas: 1000
  - Gradiente mínimo: 1x10^{-7}
  - Tamaño del paso: 0.001
  - Número de épocas para la convergencia: 194

En la Figura 3.28 se muestra la simulación de la RNA sobre la velocidad 1.

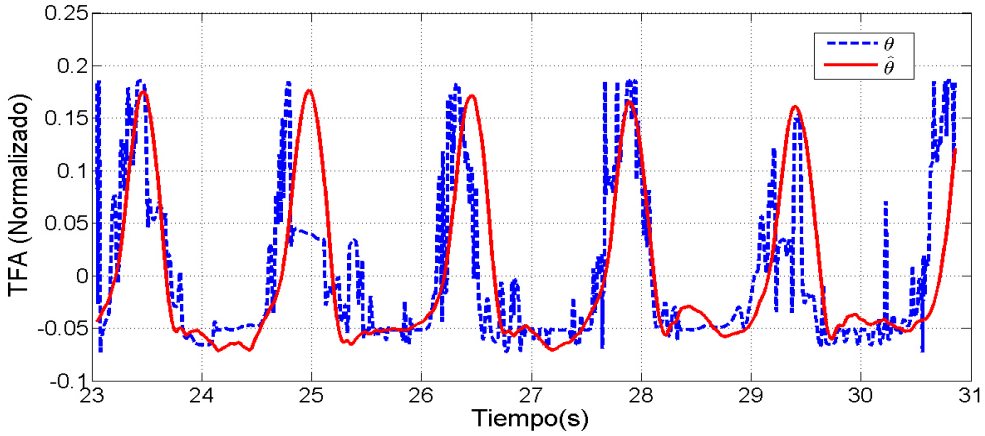


Figura 3.28: Intervalo decodificado por la RNA (10-20-10-5-1) para la velocidad 1.

Características de RNA de la velocidad 2:

- Arquitectura:
  - Número de retrasos en la entrada: 10
  - Número de entradas: 10
  - Número de capas ocultas: 3
  - Número de neuronas en la capa 1: 20
  - Número de neuronas en la capa 2: 10
  - Número de neuronas en la capa 3: 5
  - Número de neuronas en la capa de salida: 1

- Tamaño de la señal (N): 4801 muestras
- Frecuencia de muestreo (f_s): 256 Hz
- Parámetros de entrenamiento:
  - Número de épocas: 1000
  - Gradiente mínimo: 1x10^{-7}
  - Tamaño del paso: 0.001
  - Número de épocas para la convergencia: 235

En la Figura 3.29 se muestra la simulación de la RNA sobre la velocidad 2.

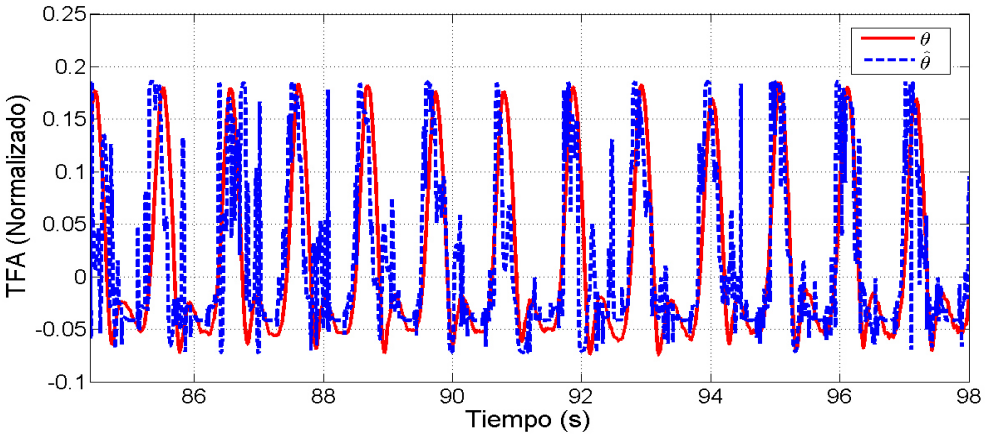


Figura 3.29: Intervalo decodificado por la RNA (10-20-10-5-1) para la velocidad 2.

En conclusión, se puede observar que las RNA dinámicas obtienen un mejor resultado que las estáticas, sin embargo las RNA dinámicas parecen poseer ruido de alta frecuencia por lo que posiblemente con un filtro pasa baja se obtenga un mejor desempeño.

Análisis comparativo

Se realizó un análisis comparativo de los algoritmos candidatos para la etapa de decodificación, el error cuadrático promedio normalizado (NMSE) fue el criterio de evaluación. El NMSE está definido por la siguiente ecuación:

$$NMSE = \frac{\sum_{N}^{n=0}[d(n) - y(n)]^2}{\sum_{N}^{n=0}[d(n)]^2}, \tag{3.1}$$

Donde $y(n)$ es la salida del método seleccionado en la muestra n y $d(n)$ es la señal deseada.

Algoritmo	Vel .1	Vel. 2
F. Wiener	0.3377	0.2629
F. LMS	0.4619	0.3020
F.LMS P	0. 4652	0.2958
F. RLS	0.5506	0.3030
MLP (estática)	0.4725	0.7995
MLP (dinámica)	0.5608	0.5977

Tabla 3.1: Tabla comparativa de los resultados de los algoritmos propuestos para la etapa de decodificación.

Conforme a los resultados Tabla 3.1 se selecciona al filtro Wiener como la técnica a implementar como la etapa de decodificación.

3.2.3 Diseño del clasificador

En la Figura 3.30 y 3.31 se muestra gráficamente el resultado de aplicar el decodificador sobre todo el experimento.

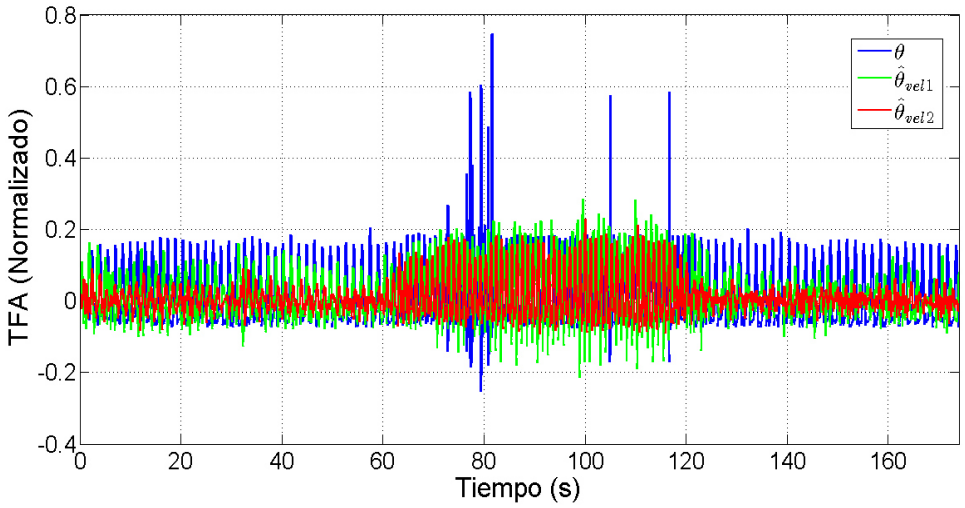
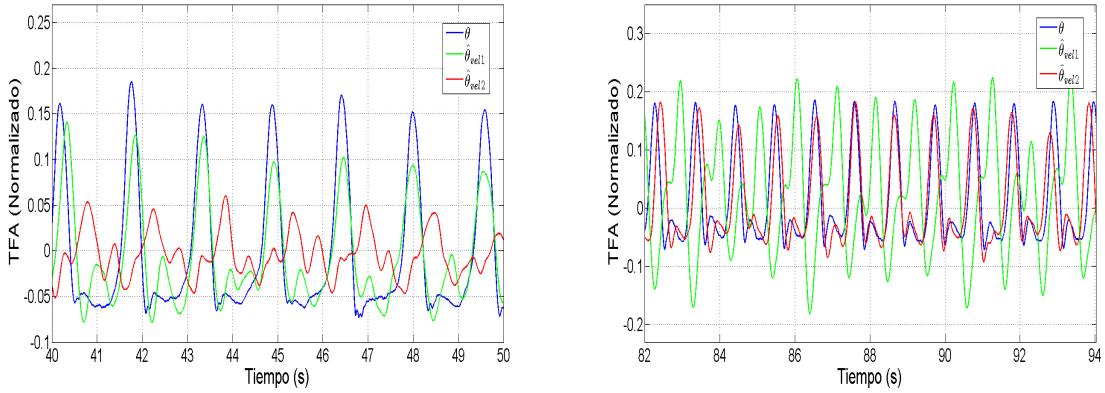


Figura 3.30: Estimación del TFA de cada filtro en todo un experimento.



(a) (b)

Figura 3.31: Estimación del TFA durante la a) velocidad 1 (4 mph) y b) velocidad 2 (6 mph).

A pesar de los resultados obtenidos en el diseño del decodificador, se observa que el uso de un solo filtro no es suficiente para replicar el movimiento en su totalidad, debido a esto se propone usar un banco de filtros que engloben una cierta actividad específica. Esto nos lleva al problema de encontrar la manera de seleccionar el modelo apropiado para la tarea que el usuario pretenda ejecutar.

Con ello se plantea el diseño de un clasificador basado las características relacionadas en la actividad correspondiente de la señal EEG del usuario, y con ella deberá ser capaz de estimar la tarea del usuario. La señal EEG registrada en electrodo C3 se establece como la entrada del clasificador.

Extracción de características

En la Sección 2.5.1 se describió el objetivo de esta etapa, el cual se resume en el cálculo del vector de características, que se llevara a cabo considerando la magnitud del espectro de la señal EEG (usando el comando de MATLAB "fft", que calcula la trasformada rápida de Fourier) en una ventana fija de 256 muestras, está ventana puede ser con traslape o sin traslape, el tipo de ventana se definirá más adelante conforme al resultado del diseño del clasificador. Posteriormente generamos un nuevo vector de 15 elementos con los componentes correspondientes a las ondas α y β. Para el lector interesado, el código empleado para la extracción de características se describe en el Apéndice A.1.

Clasificador de Bayes

El algoritmo que se evaluará es el clasificador ingenuo de Bayes, el cual es una aproximación del clasificador de Bayes descrito en la Sección 2.5.2, ésta aproximación se logra haciendo la suposición de que las características son condicionalmente independientes de la clase. Se usó el comando "NaiveBayes.fit" de MATLAB para diseñar este clasificador. A continuación se muestra el resultado de clasificador de la señal EEG del C3, el cual usa una ventana sin traslape que dio lugar a 189 vectores de entrenamiento, considerando como las Tarea 1 y -1 como las velocidad 1 y 2 respectivamente.

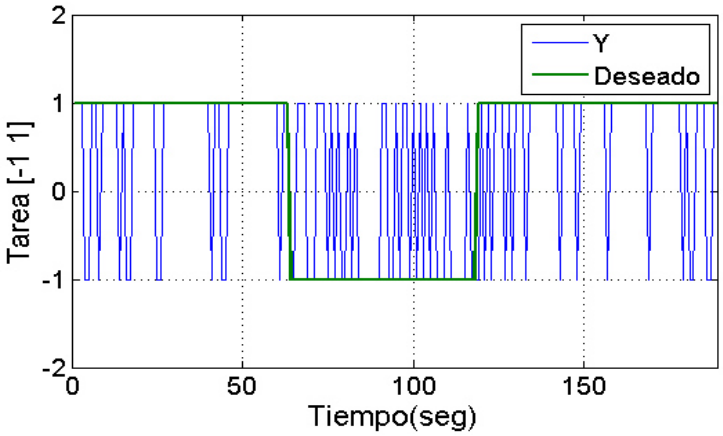


Figura 3.32: Estimación de la tarea del usuario usando el clasificador de Bayes con la señal del C3 en todo un experimento.

El clasificador de Bayes con ventana sin traslape de la Figura 3.32, el cual obtuvo un resultado de 74.76 % de porcentaje de aciertos (PA). Para una ventana con traslape y corrimiento de una muestra, se obtuvieron 44810 vectores de entrenamiento, para los cuales el algoritmo no fue capaz de sintetizar un resultado.

Máquina de soporte vectorial

Éste método fue descrito en la Sección 3.2.3. Para diseñar este clasificador se usó el comando de MATLAB "svmtrain", el cual define una función discriminante partiendo de un conjunto de datos con su clase (tarea) correspondiente, sin embargo el usuario puede definir la función de kernel, la función de kernel que se estableció para este trabajo fue la del polinomio de 5° orden (por obtener los mejores resultados). A continuación se muestra el resultado de clasificador con una ventana sin traslape de la señal EEG del C3:

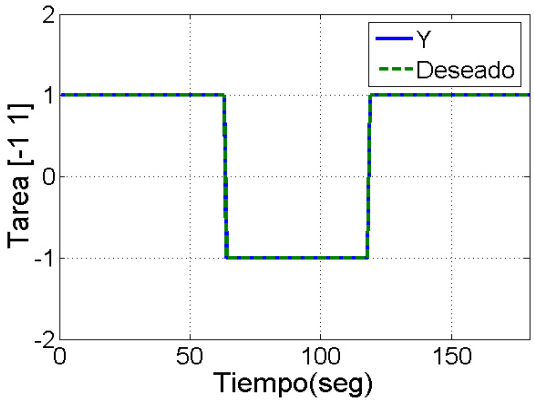


Figura 3.33: Estimación de la tarea del usuario usando una SVM y la señal C3 en todo un experimento.

La SVM de la Figura 3.33 obtuvo un resultado de 100 % PA, por otra parte el modelo requiere de 126 vectores de 15 elementos, que representa un poco más del 65 % de los datos de entrenamiento; esto indica que el método tiene dificultad para generalizar resultados para los dos caso, dicho esto el modelo puede ser simplificador usando los multiplicadores de Lagrange (α), ya que este parámetro indica la influencia del VS asociado, por lo que estableciendo un umbral se pueden discriminar los menos influyentes, el umbral u esta descrito por la siguiente ecuación:

$$u = m[\min(\alpha)], \tag{3.2}$$

donde m es una constante en los $\mathbb{R}$ definida por el usuario y α representa el vector de todos multiplicadores de Lagrange del modelo. Solamente serán tomados en cuenta los vectores de soporte cuyos multiplicadores $\hat{\alpha}$ tengan una magnitud que sobrepase el umbral u, es decir:

$$\hat{\alpha} = \begin{cases} \alpha_i & \alpha_i > u \\ 0 & \text{en otro caso} \end{cases} \tag{3.3}$$

A continuación se muestra el resultado de las SVMs en su versión simplificada.

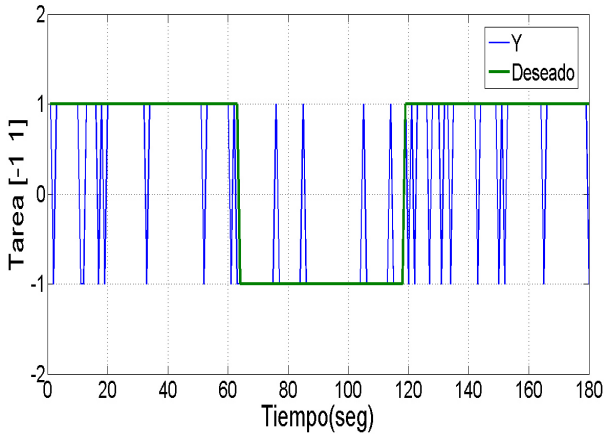


Figura 3.34: Estimación de la tarea del usuario usando una SVM y la señal C3 en todo un experimento con m=250.

La SVM de la Figura 3.34 obtuvo un resultado de 86.77 % PA, el modelo requiere de 87 vectores de 15 elementos, que representa el 46.03 % de los datos de entrenamiento. Finalmente se evalúa el uso de una venta con traslape y corrimiento de una muestra.

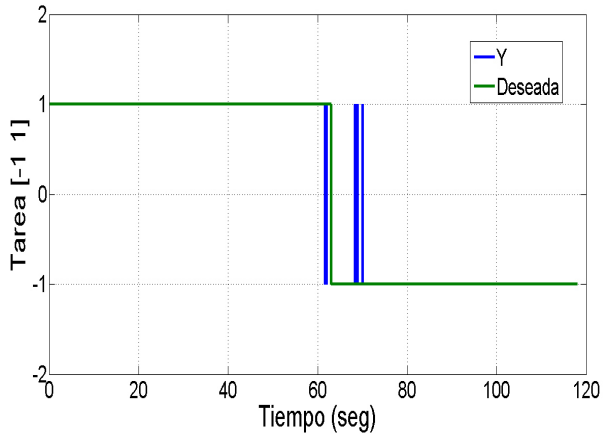


Figura 3.35: Estimación de la tarea del usuario usando una SVM con ventana con traslape (VT) y la señal C3 en todo un experimento.

La SVM de la Figura 3.35 obtuvo un resultado de 99.9 % (solo 23 errores) PA, el modelo requiere de 974 VS, que representa el 3.22 % de los datos de entrenamiento. Siendo un total de 16558 parámetros fijos (incluyendo α y bias) del tipo doble (64 bits, 1059712 bits en total), que conlleva a un problema para el almacenamiento en memoria.

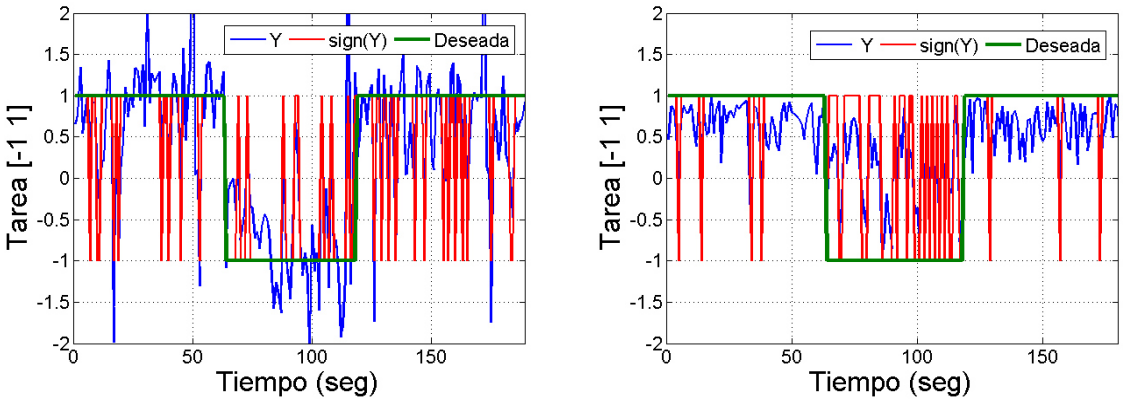
En conclusión, a pesar del buen resultado en la clasificación, el modelo presenta el inconveniente de requerir de un gran número de parámetros, que conlleva a más recursos de memoria.

Perceptrón Multicapa (clasificador)

Siguiendo la metodología expuesta anteriormente (secciones 2.4.2 y 2.5.2), usando la función signo(sgn()), la cual limita los resultados de la RNA a un conjunto finito de categorías posibles. Se evaluaron dos arquitecturas diferentes para cada señal. Las características de la etapa de aprendizaje son:

- Tamaño de la señal de entrenamiento con ventana sin traslape: 189x15
- Tamaño de la señal de entrenamiento con ventana con traslape: 30267x15
- Frecuencia de muestreo (f_s): 256 Hz
- Parámetros de entrenamiento:
  - Número de épocas: 1000
  - Gradiente mínimo: 1x10^{-7}
  - Tamaño del paso: 0.001
  - Número de épocas para la convergencia: 16

A continuación se muestran los resultados usando la señal muestreada con una ventana sin traslapes.



(a) (b)

Figura 3.36: Tarea estimada usando la señal del C3 y una a) RNA de 15-50-1 b) RNA de 15-50-20-1.

Los resultados de las dos arquitecturas propuestas se muestran en la Figura 3.36. La arquitectura 15-100-1 obtuvo un PA de 83.06 % mientras que la RNA 15-50-20-1 consiguió un PA de 80.42 %. Finalmente en la Figura 3.37 se muestra el resultado de usar la señal muestreada con una ventana con traslape y corrimiento de una muestra.

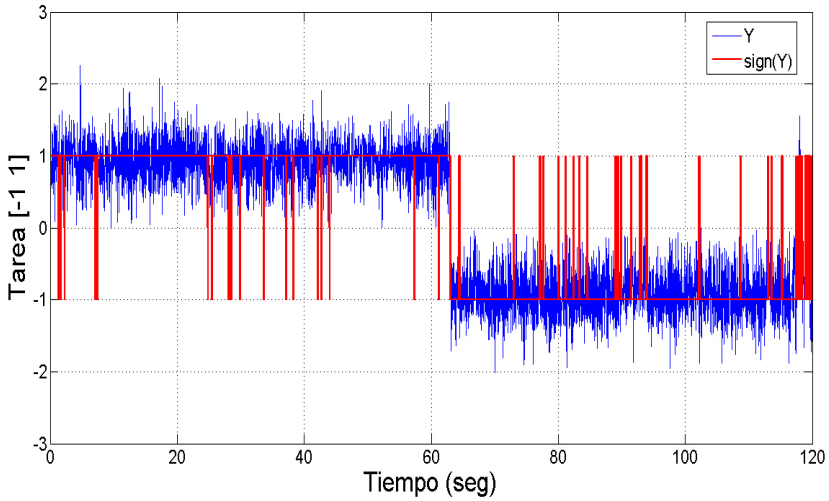


Figura 3.37: Tarea estimada por RNA (15-100-1) usando la señal del C3.

La arquitectura 15-100-1 usando una ventana con traslape tuvo un resultado similar al de la SVM, sin embargo se requieren de mucho menos parámetros para implementar esta técnica.

Análisis comparativo

El análisis comparativo consistió en evaluar el porcentaje de aciertos (PA) de los algoritmos propuestos tal como se muestra en la Tabla 3.2.

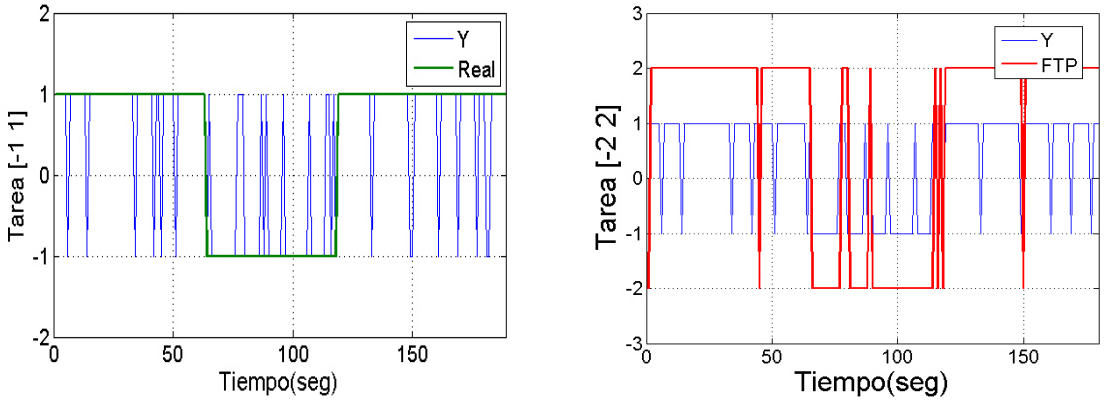
Clasificador	Porcentaje de aciertos (%)
Clasificador de Bayes	74.07
RNA (15-50-20-1)	80.42
RNA (15-50-1)	83.06
SVM	86.77

Tabla 3.2: Tabla comparativa de los resultados de la etapa de clasificación.

Por lo tanto se define la SVM como la técnica a implementar para la etapa de clasificación, dado que obtuvo los mejores resultados. En el Apéndice A.1 describe el código para la implementación de este método.

Protector de falsos verdaderos

Con fin de mejorar el rendimiento del clasificador se le agrega una etapa de protección a falsos verdaderos (FTP), el cual consiste en un filtro pasa baja (promedio de tres muestras) y un umbral para delimitar la salida del clasificador a dos números enteros. En la Figura 3.38 se muestra el incremento en la efectividad del clasificador al usar el FTP.



(a) (b)

Figura 3.38: Estimación de la tarea del usuario con: a) SVM (86.77 %) y b) SVM con FTP (93.05 %).

3.3 Simulación

En esta sección se discute la simulación del sistema, que se desarrolló en el entorno de Simulink, cuyo diagrama de bloques se muestra en la Figura 3.39.

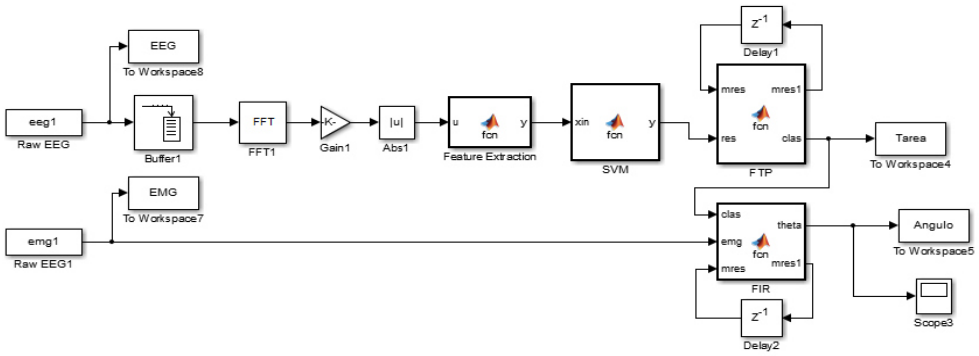


Figura 3.39: Sistema hBMI completo en Simulink.

Para el lector interesado, los códigos empleados para esta simulación, se describe en el Apéndice 5.1. En la Figura 3.40 se muestran los resultados de la simulación.

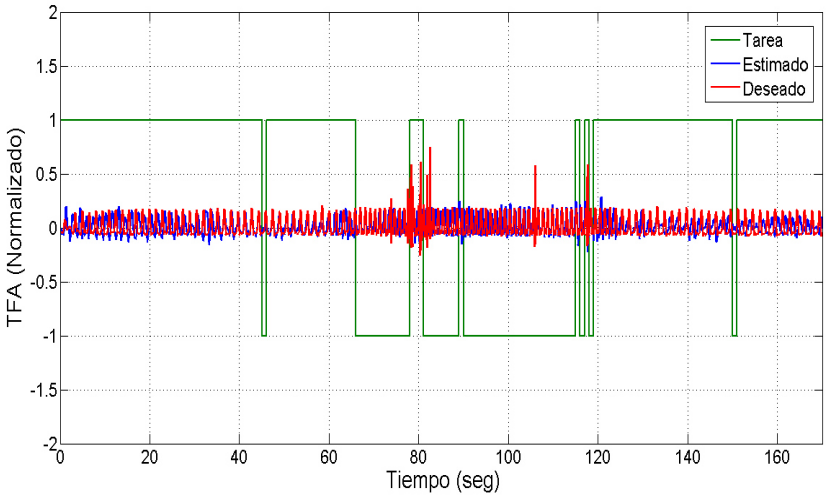
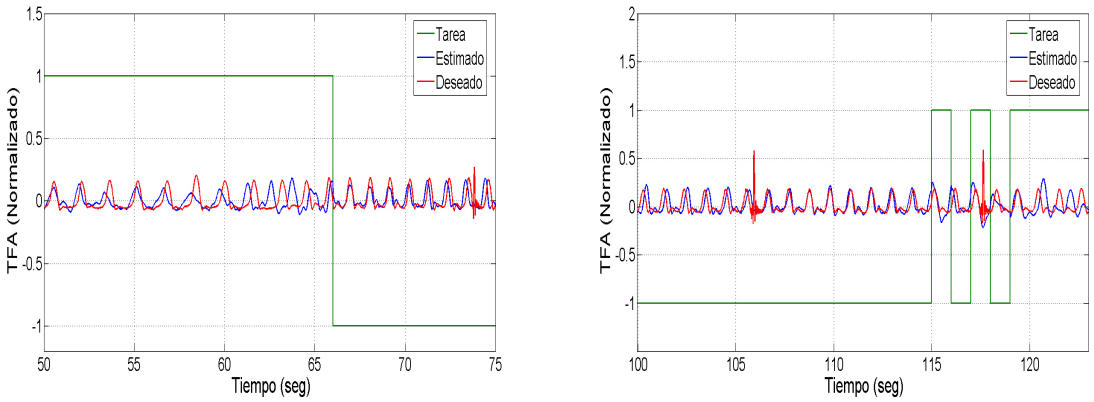


Figura 3.40: TFA estimado por la hBMI en todo el experimento.



(a) (b)

Figura 3.41: Estimación del TFA en un intervalo de comutación de a) velocidad 1-2 y b) velocidad 2-1.

Mientras que en la Figuras 3.41 se hace un énfasis en el momento en que el sistema debe conmutar. Por último se realizó un análisis comparativo (Tabla 3.3) entre los filtros de cada velocidad y la hBMI, usando nuevamente el NMSE como criterio de evaluación y ventanas correspondientes a las velocidades de operación.

Estados	Filtro Vel .1	Filtro Vel .2	hBMI
Velocidad 1	0.0038	0.0085	0.0043
Velocidad 2	0.0338	0.0134	0.0140
Total	0.0376	0.0219	0.0183

Tabla 3.3: Tabla comparativa de los resultados de los filtros individuales y la hBMI.

La hBMI presenta mayor error que los filtros en la velocidad a la que fueron diseñados, esto se debe a los errores en la clasificación, sin embargo se concluye que de manera global si se mejora el proceso de decodificación empleando la hBMI, en comparación de usar los filtros por si solos.

Capítulo 4

Implementación

El objetivo de este capítulo es detallar la implementación de los algoritmos seleccionados del capítulo 3 para cada etapa del sistema hBMI; comenzando con la Sección 4.1, donde se describe las características del hardware propuesto y el entorno de programación del mismo. Continuando con la Sección 4.2 la cual está enfocada en presentar los esquemas, paradigmas, tiempos de computo (por etapa) y el entorno para la simulación del tipo "hardware in the loop". Y finalmente la Sección 4.3 muestra los resultados de la implementación. Para el lector interesado, en el Apéndice A.2 se describe la implementación del sistema propuesto en un hardware de National Instruments (NI myRIO-1900).

4.1 Hardware

Para seleccionar el hardware apropiado, se definen las siguientes características necesarias en el sistema:

1. Convertidor Analógica-Digital (ADC) con resolución de 10 bits y $f_s \geq 256$ Hz.
2. Convertidor Digital-Analógica (DAC) con resolución de 10 bits y $f_s \geq 256$ Hz.
3. Frecuencia de reloj mayor a 40MHz.
4. Portátil y económico (menor a $1500.00 MN).

Los ADC y DAC son necesarios debido a que tanto la señal de entrada como la salida del sistema se definieron como señales analógicas y con estos convertidores el sistema digital será capaz de procesar la información. Uno de los factores que determina el costo de un convertidor es la resolución, dado a que se desea que el sistema sea económico, se considera un convertidor de 10 bits, ya que el valor mínimo que puede adquirir dicho convertidor es de 4.88 mV y 2.92 mV, para sistemas que operan a 5 V y 3.3 V, lo cual no representa problema para los dispositivos de adquisición actuales, que son capaces de ajustar la magnitud de la señal de entrada, en el caso de la salida, el sistema deberá ser capaz de escalar la señal de salida para hacer despreciable el error de cuantificación. Debido a que la mayor energía de la señal EMG existe entre 7-100 Hz y las bandas de frecuencias de interés de las señales EEG son menores a 30Hz, se requiere de un muestreo por lo menos de 256 Hz (para satisfacer el criterio de Nyquist), por lo tanto la tasa de conversión de los ADC y DAC tiene que ser mayor a lo indicado. El sistema digital debe tener la capacidad de calcular la DFT, efectuar la SVM, estimar el TFA y mantener la frecuencia de muestreo por lo tanto se considera reloj de 40 MHz.

Con todo lo mencionado anteriormente, se seleccionó el microcontrolador (MCU) de 32 bits "ATSAM3X8E" [56] de ATMEL, que posee los recursos deseados para el sistema propuesto. Además en el mercado actual existe una tarjeta de desarrollo compacta enfocada en este MCU, conocida como Arduino DUE (Figura 4.1), la cual fue diseñada como "hardware libre"por *Arduino*® [58].

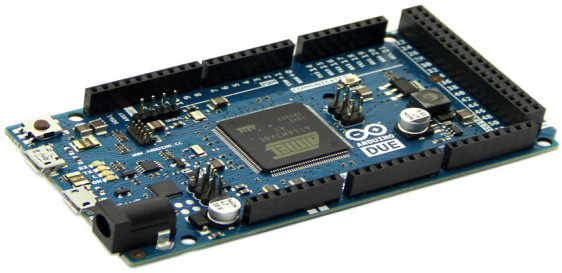


Figura 4.1: Tarjeta de desarrollo Arduino DUE [58].

A continuación se muestra un resumen de características más relevantes (para este trabajo) del Arduino Due [56, 58]:

- Procesador: ATSAM3X8E (MCU, ARM Cortex-M3).
- Frecuencia del reloj: 84 MHz.
- Memoria interna: 512 KB de memoria FLASH, 96 KB SRAM.
- Canales ADC (entradas analógicas): 16 canales de 12 bits (resolución) con una tasa de conversión de 1 MHz.
- Canales DAC (salidas analógicas): 2 canales de 12 bits con una tasa de conversión de 1 MHz.
- Interrupciones disponibles: 30.
- Voltaje de operación: 3.3 V.
- Otras características: Contadores de 9 canales de 32 bits, reloj en tiempo real (RTC), contador de tiempo real (RTT), 54 salidas digitales.

Para programar el Arduino DUE, la empresa Arduino desarrolló su propio IDE (entorno de programación), sin embargo para este trabajo, se usó el software ATMEL Studio 7.0 y se programó en lenguaje C, debido a que en este software es posible depurar en tiempo real, con la herramienta necesaria (con el programador/debuger ATMEL ICE).

4.2 Esquema de implementación

4.2.1 Esquema de programación

En la Figura 4.2, se muestra un diagrama con los elementos que serán incluidos en el sistema embebido.

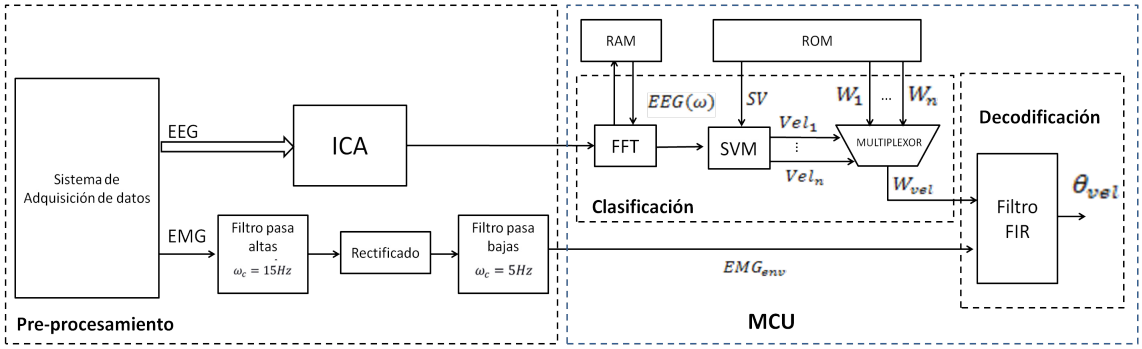


Figura 4.2: Principales procesos que se realizan en el sistema embebido propuesto.

El primer enfoque (programación secuencial) tomado para la implementación del sistema embebido se muestra la Figura 4.3.

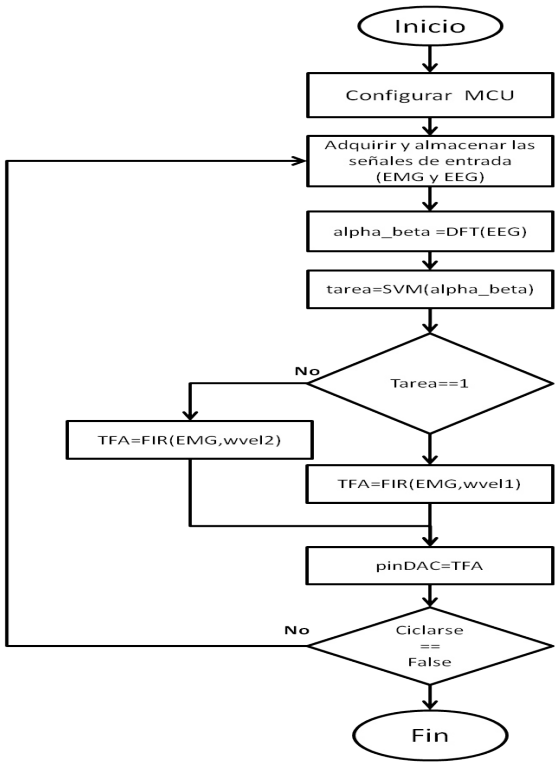


Figura 4.3: Diagrama de flujo de la programación secuencial del sistema embebido.

Este enfoque tiene la finalidad de medir el tiempo de ejecución por etapa, esto se realiza con un contador embebido en hardware del MCU.

Etapa	Tiempo de ejecución(ms)
DFT y EC	14.5
SVM	3-4
PFV	0.84
FIR	1
Total	~20

Tabla 4.1: Tabla de los tiempos de ejecución de cada etapa del sistema implementado en ATSAM3X8E.

En la Tabla 4.1 se muestra el tiempo de procesamiento del MCU para ejecutar cada una de las etapas del sistema propuesto, además demuestra que no es factible usar el enfoque secuancial, dado que tiene un tiempo de computo aproximado de 20 ms ($f_s = 50\ Hz$). Por otro lado, al cambiar el esquema de programación por uno que maneje interrupciones, hace posible mantener una $fs = 256\ Hz$, esto se consigue alterando el flujo normal del programa, seleccionando las tareas que se quieran ejecutar en una interrupción periódica.

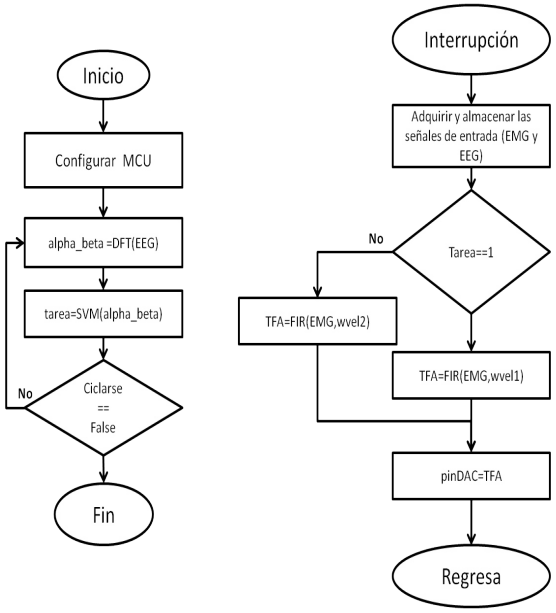
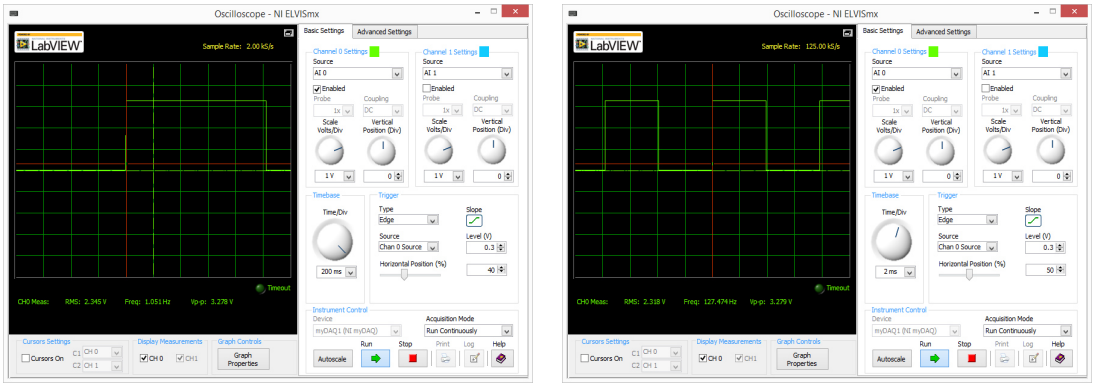


Figura 4.4: Diagrama de flujo de la programación manejando la interrupción del sistema embebido.

La interrupción que se agregó es accionada por un contador embebido en el hardware interno del MCU, ésta interrupción se programó para ejecutarse cada 3.906 ms ($f_s = 256\ Hz$). Y como se muestra en la Figura 4.4, la interrupción ejecuta la etapa de decodificación, la adquisición de las señales y el despliegue de la señal de salida. Para asegurarnos de operar según el requisito del muestreo, durante la interrupción conmutamos un pin del MCU y como se observa en la Figura 4.5, el sistema opera con una $f_s = 254.948\ Hz$.



(a) (b)

Figura 4.5: Experimento para medir la frecuancia de muestreo. a) Frecuencia del clasificador y b) Frecuencia del decodificador.

Para el lector interesado, el código que incluye la configuración del MCU (interrupciones, contadores, ADC, DAC, etc.) y las funciones (filtro FIR, SVM, RNA, TFP, DFT, etc.) se describen en el Apéndice B.

4.2.2 Hardware in the loop

Para probar el desempeño de la hBMI en un ambiente más realista, se simularon las señales fisiológicas de un posible usuario con un dispositivo NI-myDAQ (Figura 4.6), es decir, este dispositivo de instrumentación se encarga de suministrar las señales de entrada (EEG y EMG) para el sistema embebido y además, adquiere y grafica la señal de salida del sistema.

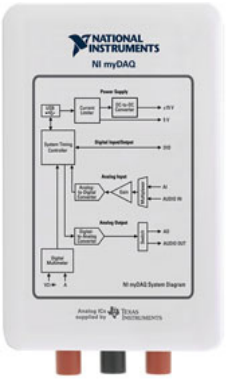
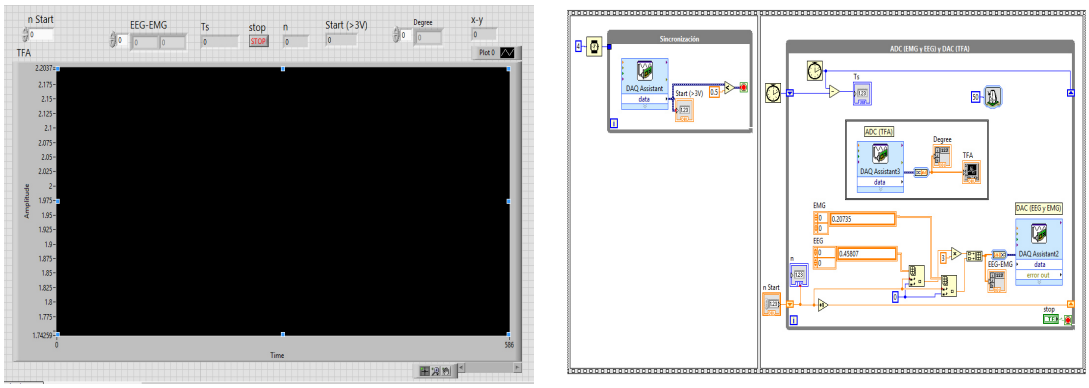


Figura 4.6: Instrumento NI-myDAQ para la simulación de la hBMI.

Algunas de las características relevantes para la simulación son [57]:

- Características de las entradas analógicas:
  - Número de canales: 2 diferenciales.
  - Resolución del ADC: 16 bits.
  - Capacidad Máxima de muestro: 200 kS/s.
  - Rango: ±10 V.
- Características de las salidas analógicas:
  - Número de canales: 2 (con referencia a tierra).
  - Resolución del DAC: 16 bits.
  - Capacidad Máxima de muestro: 200 kS/s.
  - Rango: ±10 V.
  - Corriente de salida máxima: 2 mA.

Eventualmente para programar la simulación, se diseño una interfaz virtual en el software $LabVIEW^{®}$ de NI. Dicha interfaz se muestra en la Figura 4.7, sin embargo, debido a que este software está basado un entorno gráfico, en la Figura 4.7 se muestra la programación (diagrama de bloques) de la interfaz.



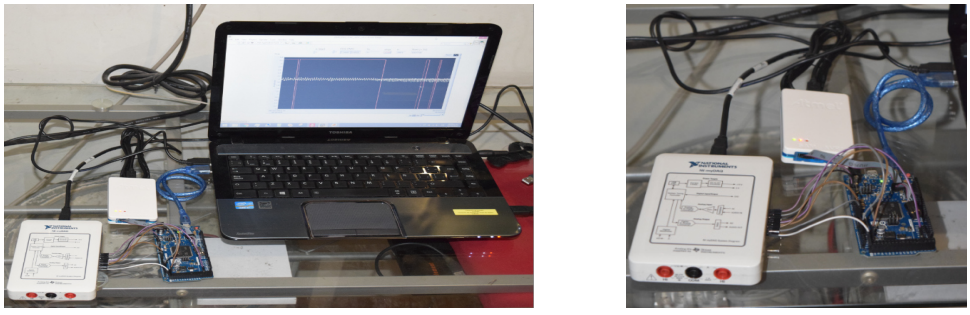
(a) (b)

Figura 4.7: Interfaz gráfica para simular un experimento. a) Panel frontal y b) Diagrama de bloques de la interfaz.

A pesar de las características discutidas anteriormente, la frecuencia de muestro de la simulación se ve afectada debido a la interacción y las operaciones que se realizan entre la interfaz y el dispositivo, por lo cual la simulación opera a 20Hz aproximadamente y se restringió la frecuencia de muestreo del MCU para cubrir las exigencias de la simulación.

4.3 Resultados

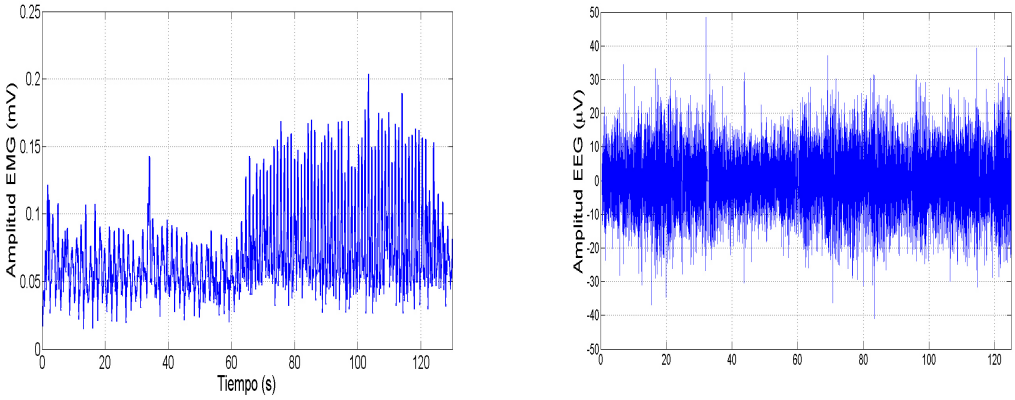
En la Figura 4.8 se muestra el ambiente para la prueba de la implementación.



(a) (b)

Figura 4.8: Hardware de pruebas para evaluar la implementación.

Las señales que fueron embebidas en el myDAQ se muestra en la Figura 4.9.



(a) (b)

Figura 4.9: Señales utilizadas para simular un experimento. a) Envolvente de la EMG y b) Señal EEG del electrodo C3.

Dado que la salida del sistema embebido es una señal analógica de 0 a 3.3 V, se le sumó una constante (offset) de 2.5 para obtener solo valores positivos y además, se escaló la salida con un factor de 1.25 para tener mejor resolución. En la Figura 4.10 se muestran los resultados (exportados a MATLAB) de la implementación comparados con la señal deseada.

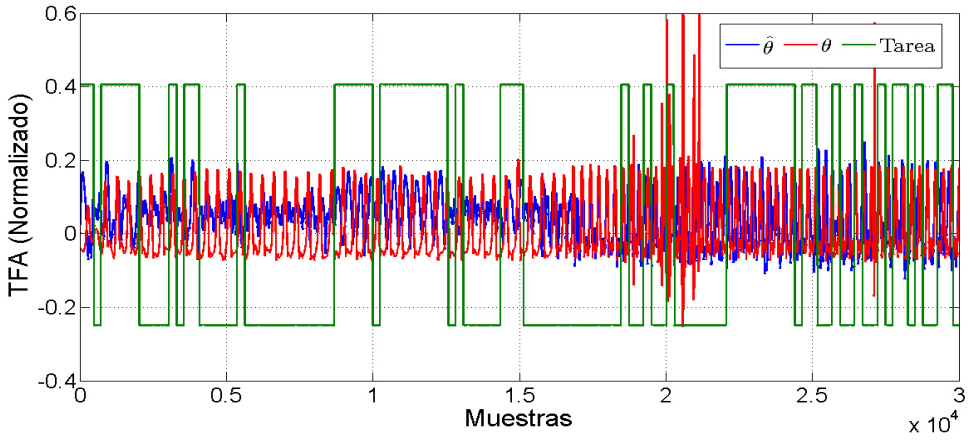


Figura 4.10: Salidas de la hBMI (SVM con VST) y la señal de deseada.

El PA del clasificador durante la prueba de la implementación fue de 50.77 %,

este resultado es inaceptable para cualquier clasificador, además de afectar directamente el desempeño del decodificador. En la Sección 3.2.3 se evaluó la posibilidad de usar clasificadores con ventanas con traslapes, los cuales obtenían un mejor desempeño, con la desventaja de requerir un modelo más complejo. Dado los resultados anteriores, se optó por cambiar el tipo de ventana a una con traslape (VT), por consiguiente el modelo del clasificador que representa menor carga computacional fue el MLP (15-100-1).

En la Figura 4.11 se muestran los resultados (exportados a MATLAB) de la implementación comparados con la señal deseada.

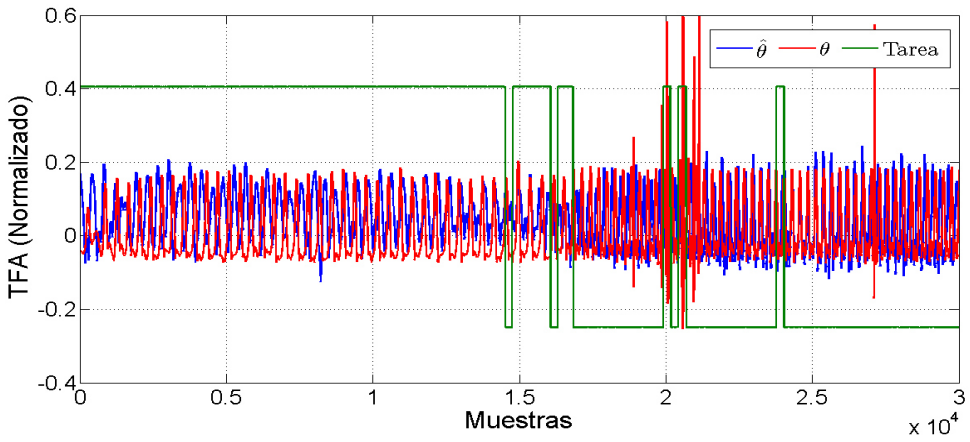


Figura 4.11: Salidas de la hBMI (MLP con VT) y la señal de deseada.

Finalmente, en la Tabla 4.2, se compararon ambos resultados, usando como criterios el NMSE y el PV, y con ello se concluye que mejora el rendimiento.

Criterio	hBMI (SVM con VST)	hBMI (MLP con VT)
PA (%)	50.77	95.09
NMSE	0.0369	0.0145

Tabla 4.2: Tabla comparativa entre los resultados de la prueba de implementación.

Capítulo 5

Conclusiones y trabajo a futuro

5.1 Conclusiones

La principal contribución de este proyecto de investigación consiste en el diseño de un sistema digital para decodificar (estimar) una VC durante diferentes tareas, el esquema embebido en el sistema digital corresponde a un clasificador/decodificador, donde el clasificador estima la tarea del usuario, basándose en la señal EEG (del electrodo C3), mientras que el decodificador estima la VC partiendo de la señal EMG y un banco de filtros, el cual conmuta entre filtros dependiendo de la tarea del usuario.

El diseño del sistema comprende principalmente de tres etapas: 1) Etapa de preprocesamiento, donde se definen los procesos necesarios para obtener las señales de entrada. Para obtener la señal EEG, se propone usar el algoritmo ICA para eliminar las EOG u otras señales parásitas; para el caso de la señal EMG, se propone calcular la envolvente de la señal con el método de filtrado lineal. 2) Etapa de clasificación, en la cual se define el vector de características (relacionada con el HGC) junto con el modelo del clasificador. Se exploraron diversas metodologías para definir el clasificador, sin embargo, la SVM con VTS obtuvo el mejor resultado basándonos en el PA y la cantidad de memoria que requiere el modelo, seguido por MLP con VT. 3) Etapa de decodificación, en donde se define el modelo que relaciona la señal EMG con VC (específicamente el TFA). Para ello se probaron diferentes enfoques, siendo el filtro Wiener el algoritmo con mejor desempeño de acuerdo al NMSE.

Una vez definido los algoritmos que comprenden al sistema, se realizó una simulación de la hBMI usando señales adquiridas de varios experimentos relacionados. Esta simulación afirma la hipótesis, mostrando que la hBMI propuesta tiene un mejor desempeño que solo usar el método seleccionado para la decodificación.

El objetivo general se cumple con la implementación de la hBMI sobre la tarjeta de desarrollo Arduino DUE®. Esta plataforma es tarjeta electrónica basada en un MCU de 32 bits (ATSAM3X8E), el cual se programa en lenguaje C.

La implementación fue puesta a prueba, usando la técnica "harware in the loop" en un hardware de instrumentación (NI-mydaq), en donde se pretende emular un experimento; suministrando y adquiriendo las señales de entrada y de salida respectivamente, de la hBMI.

El desempeño de la implementación de la hBMI se determinó usando el PA y NMSE como criterios de evaluación del clasificador y decodificador respectivamente. Dado los resultados de la implementación, se llego a la conclusión de que el clasificador SVM con VST no tenia un buen desempeño (un PA de casi 50 %), por lo que se cambió el clasificador a un MLP con VT, mejorando su desempeño a un 95 %.

En resumen, los siguientes puntos concluyen los objetivos particulares:

- El sistema se implementó satisfactoriamente emulando el ángulo de la rodilla durante la caminata a distintas velocidades, sin embargo siguiendo la metodología expuesta en este trabajo, es posible extrapolar la hBMI a ejecutar más actividades relacionadas a señales EEG y EMG.
- Según nuestro análisis comparativo, el filtro Wiener y el MLP con VT resultaron ser los mejores candidatos para la implementación de nuestro sistema.
- La hBMI fue implementada y probada usando hardware comercial y compacto.

5.1.1 Recomendaciones

Finalmente, cabe mencionar que este trabajo ha presentado una revisión del estado del arte, definiciones y procedimientos que permiten continuar con una línea de investigación. Sin embargo, los siguientes aspectos deben ser considerados:

- Para el entrenamiento y la simulación de los algoritmos propuestos se usaron señales de experimentos similares, lo ideal seria que dichas señales se obtuvieran de un solo experimento de un solo usuario.
- Dado que el dispositivo de instrumentación estaba limitado, no fue posible probar el sistema a una f_s de 256Hz.

5.2 Trabajo a futuro

Dentro de los temas en los que se puede profundizar o explorar están:

- Explorar metodologías de aprendizaje no supervisado o de aprendizaje de refuerzo.
- Acondicionar la salida del sistema ante un cambio abrupto.
- Explorar técnicas distintas a las que se utilizaron en este trabajo.
- Acondicionar y probar el sistema para diferentes tareas.
- Probar la hBMI con un usuario.

Nomenclaturas

BMI	Brain Machine Interface (Interfaz Cerebro Máquina).
BCI	Brain Computer Interface (Interfaz Cerebro Computadora).
hBMI	hybrid-Brain Machine Interface (Interfaz Cerebro Máquina híbrida).
EMG	Señal mioeléctrica (proveniente de un Electromiograma).
EEG	Señal cerebral (proveniente de un Electroencefalograma).
VC	Variable Cinemática
TFA	Tibio-Femoral Angle, θ (Ángulo tibiofemoral).
PCA	Principal Component Analysis (Análisis de Componentes Principales).
ICA	Independent Component Analysis (Análisis de Componentes Independientes).
RNA	Red neuronal artificial.
MLP	Multi-Layer Perceptron (Perceptrón Multicapa).
SVM	Support Vector Machine (Máquina de Soporte Vectorial).
LMS	Least Mean Squares (Mínimos Cuadrados Promedios).
RLS	Recursive Least Squares (Mínimos Cuadrados Recursivos).
FIR	Finite Impulse Response (Respuesta al impulso Finita).
MSE	Mean Square Error (Error Cuadrático Medio).
NMSE	Normalized Mean Square Error(Error cuadrático medio normalizado).
CP	Componente principal.

Bibliografía

[1] EUROSTAT. Your key to european statistics, 2011. $http : //appsso.eurostat.ec.europa.eu/nui/show.do?dataset = hlth_dp040\&lang = en$ $[Online; accessed 17 - February - 2015]$.

[2] Matthew W. Brault. Americans with disabilities: 2010. In *Current Population Reports P70-131, U.S. Census Bureau, Washington, DC*, 2012.

[3] J.P. Gutiérrez, J. Rivera-Dommarco, T. Shamah-Levy, S. Villalpando-Hernádez, L. Cuevas-Nasu A. Franco, M. Romero-Martínez, and M. Hernández-Ávila. Encuesta nacional de salud y nutrición 2012. In *Resultados Nacionales, Instituto Nacional de Salud Pública (MX), Cuernavaca, México*, 2012.

[4] W.; Khuan L.Y. Norani, N.A.M.; Mansor. A review of signal processing in brain computer interface system. In *IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES)*.

[5] V. Stankevich, P.; Spitsyn. A review of brain-computer interface technology. In *The 2015 International Siberian Conference on Control and Communications (SIBCON)*.

[6] Ali Bashashati, Mehrdad Fatourechi, Rabab K Ward, and Gary E Birch. A survey of signal processing algorithms in brain computer interfaces based on electrical brain signals. *Journal of Neural Engineering*, 4(2):R32, 2007.

[7] Xun Zhang, François-Benoît Vialatte, Chen Chen, Apurva Rathi, and Gérard Dreyfus. Embedded implementation of second order blind identification (sobi)

for real time applications in neuroscience. *Cognitive Computation*, 7(1):56–63, 2015.

[8] Amar Kachenoura, Laurent Albera, Lotfi Senhadji, and Pierre Comon. Ica: A potential tool for bci systems, Nov 2007.

[9] S. Sun and J. Zhou. A review of adaptive feature extraction and classification methods for eeg-based brain-computer interfaces. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 1746–1753, July 2014.

[10] Nathan A. et al. Fitzsimmons. Extracting kinematic parameters for monkey bipedal walking from cortical neuronal ensemble activity. In *Frontiers in Integrative Neuroscience*, 2009.

[11] Thierry Castermans, Matthieu Duvinage, Guy Cheron, and Thierry Dutoit. Towards effective non invasive brain-computer interfaces dedicated to gait rehabilitation systems. *Brain sciences*, 4(1):1–48, 2013.

[12] Alessandro Presacco, Ronald Goodman, Larry Forrester, and Jose Luis Contreras-Vidal. Neural decoding of treadmill walking from noninvasive electroencephalographic signals. *Journal of neurophysiology*, 106(4):1875–1887, 2011.

[13] An H Do, Po T Wang, Christine E King, Sophia N Chun, and Zoran Nenadic. Brain-computer interface controlled robotic gait orthosis. *Journal of neuroengineering and rehabilitation*, 10(1):1, 2013.

[14] A. Úbeda, D. Planelles, A Costa, E. Hortal, E. Ianez, and J. M. Azorin. Decoding knee angles from eeg signals for different walking speeds. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1475–1478, Oct 2014.

[15] World Health Organization. Stroke, cerebrovascular accident, 2012. $http://www.who.int/topics/cerebrovascular_accident/en/$ [Online; accessed 17-February-2015].

[16] Samuel T. Claton. Brain-computer interface control of an anthropomorphic robotic arm. 2011. Tesis Doctoral. Massachusetts Institute of Technology.

[17] B. Allison C. Guger and E. Leuthardt. Recent advances in brain-computer interface research: a summary of the bci award 2012 and bci research trends. In *In Brain-Computer Interface Research*, pages 105–109, 2014. Springer Berlin Heidelberg.

[18] Zhang Xiaodong et al. Brain-myoelectricity artificial limb control device and method based on scene steady-state visual evoking. Mar. 11 2015. CN Patent App. CN 201,410,617,265.

[19] Qiu Zu Rong et al. Artificial hand using muscle electrical and electroencephalogram cooperative control and controlling method thereof. Oct. 7 2009. CN Patent 100,546,553.

[20] B. Kais, F. Ghaffari, O. Romain, and R. Djemal. An embedded implementation of home devices control system based on brain computer interface. In *2014 26th International Conference on Microelectronics (ICM)*, pages 140–143, Dec 2014.

[21] K. K. Shyu, Y. J. Chiu, P. L. Lee, M. H. Lee, J. J. Sie, C. H. Wu, Y. T. Wu, and P. C. Tung. Total design of an fpga-based brain computer interface control hospital bed nursing system. *IEEE Transactions on Industrial Electronics*, 60(7):2731–2739, July 2013.

[22] C. W. Feng, J. C. Chang, W. C. Chen, and W. C. Fang. A reliable brain computer interface implemented on fpga for mobile dialing system. In *Consumer Electronics - Taiwan (ICCE-TW), 2015 IEEE International Conference on*, pages 110–111, June 2015.

[23] A. Palumbo, B. Calabrese, G. Cocorullo, M. Lanuzza, P. Veltri, P. Vizza, A. Gambardella, and M. Sturniolo. A novel ica-based hardware system for reconfigurable and portable bci. In *Medical Measurements and Applications, 2009. MeMeA 2009. IEEE International Workshop on*, pages 95–98, May 2009.

[24] T. E. Doyle, A. Ieta, and Z. Kucerovsky. Design of a non-volatile ambulatory data acquisition and control system for a brain-computer interface. In *2007 3rd International IEEE/EMBS Conference on Neural Engineering*, pages 346–349, May 2007.

[25] T. Dutoit T. Castermans and M. Duvinage. Method to determine an artificial limb movement from an electroencephalographic signal. Feb. 21 2013. US Patent App. 13,521,339.

[26] R. Sarpeshkar B. Rapoport and W. Wattanapanitch. Low-power analog-circuit architecture for decoding neural signals. Jan. 8 2013. US Patent 8,352,385.

[27] B. Corneil B. Greger S. Musallam, R. Andersen and H. Scherberger. Cognitive control signals for neural prosthetics. Oct. 13 2005. US Patent App. 11/086,534.

[28] Usb (universal serial bus) 3.0-based five-finger myoelectric artificial limb embedded measurement and control system and usb 3.0 data transmission method of system, March 12 2014. CN Patent App. CN 201,310,641,815.

[29] A Palumbo, F Amato, B Calabrese, M Cannataro, G Cocorullo, A Gambardella, PH Guzzi, M Lanuzza, M Sturniolo, P Veltri, et al. An embedded system for eeg acquisition and processing for brain computer interface applications. In *Wearable and Autonomous Biomedical Devices and Systems for Smart Environment*, pages 137–154. Springer, 2010.

[30] Christian Fleischer and Günter Hommel. *Embedded Control System for a Powered Leg Exoskeleton*, pages 177–185. Springer Netherlands, Dordrecht, 2006.

[31] J. Rosen, M. Brand, M. B. Fuchs, and M. Arcan. A myosignal-based powered exoskeleton system. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 31(3):210–222, May 2001.

[32] Saeid Sanei and Jonathon A Chambers. *EEG signal processing*. John Wiley & Sons, 2013.

[33] Valer Jurcak, Daisuke Tsuzuki, and Ippeita Dan. 10/20, 10/10, and 10/5 systems revisited: their validity as relative head-surface-based positioning systems. *NeuroImage*, 34(4):1600–1611, February 2007.

[34] Peter Konrad. *The ABC of EMG: A Practical Introduction to Kinesiological Electromyography*. 2005.

[35] J.G. Proakis and D.G. Manolakis. *Tratamiento digital de señales*. Fuera de colección Out of series. Pearson Educación, 2007.

[36] Yijun Wang, Xiaorong Gao, Bo Hong, and Shangkai Gao. Practical designs of brain–computer interfaces based on the modulation of eeg rhythms. In *Brain-Computer Interfaces*, pages 137–154. Springer, 2009.

[37] Swati Vaid, Preeti Singh, and Chamandeep Kaur. Eeg signal analysis for bci interface: A review. In *2015 Fifth International Conference on Advanced Computing & Communication Technologies*, pages 143–147. IEEE, 2015.

[38] K. T. Deepak, K. Ramesh, N. Adiga, and S. R. M. Prasanna. Speech and egg polarity detection using hilbert envelope. In *TENCON 2015 - 2015 IEEE Region 10 Conference*, pages 1–6, Nov 2015.

[39] Rueda Cebollero Guillem. Procesado de señales electroencefalográficas para determinar características espectrales de episodios epilépticos. 2011. Tesis Licenciatura. Universitat de Lleida.

[40] J. F. Beche, S. Bonnet, T. Levi, R. Escola, A. Noca, G. Charvet, and R. Guillemaud. Real-time adaptive discrimination threshold estimation for embedded neural signals detection. In *2009 4th International IEEE/EMBS Conference on Neural Engineering*, pages 597–600, April 2009.

[41] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4):411–430, 2000.

[42] Juliana Mu noz y Jorge Rivera y Edison Duque. Análisis de componentes principales e independientes aplicados a reducción de ruido en señales electrocardiográficas. *Scientia Et Technica*, 2(39), 2008.

[43] S. Haykin. *Adaptive Filter Theory*. Pearson Education, 2014.

[44] S.O. Haykin. *Neural Networks and Learning Machines*. Pearson Education, 2011.

[45] A.D. Poularikas and Z.M. Ramadan. *Adaptive Filtering Primer with MATLAB*. Electrical Engineering Primer Series. CRC Press, 2006.

[46] E. Alpaydin. *Introduction to Machine Learning*. Adaptive computation and machine learning. MIT Press, 2014.

[47] Mohammadreza Asghari Oskoei and Huosheng Hu. Myoelectric control systems-a survey. *Biomedical Signal Processing and Control*, 2(4):275–294, 2007.

[48] Bernard Hudgins, Philip Parker, and Robert N Scott. A new strategy for multifunction myoelectric control. *IEEE Transactions on Biomedical Engineering*, 40(1):82–94, 1993.

[49] J. Zhao, Z. Xie, L. Jiang, H. Cai, H. Liu, and G. Hirzinger. Emg control for a five-fingered underactuated prosthetic hand based on wavelet transform and sample entropy. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3215–3220, Oct 2006.

[50] Y.C. Shin and C. Xu. *Intelligent Systems: Modeling, Optimization, and Control*. Automation and Control Engineering. CRC Press, 2008.

[51] Luis Antonio Mercado-Cerda. Control of a tibio-femoral virtual joint by means of a hybrid eeg-emg scheme. Master's thesis, Universidad Autónoma de Nuevo León, 2014.

[52] Angel Omar Martínez-Mata. Análisis de la señalización electroencafalográfica (eeg) y electromiográfica (emg) de la locomoción humana. Master's thesis, Universidad Autónoma de Nuevo León, 2016.

[53] Jaakko Särelä Hugo Gävert, Jarmo Hurri and Aapo Hyvärinen. *The FastICA package*, 1996-2005. http://research.ics.aalto.fi/ica/fastica/about.shtml.

[54] INC BIOPAC Systems. *B-Alert X10 Setup Manual*, 2010. https://www.biopac.com/wp-content/uploads/b-alert-x10-user-manual.pdf.

[55] INC BIOPAC Systems. *BSL Harware Guide*, 2013. https://www.biopac.com/wp-content/uploads/MP36-MP35-MP45.pdf.

[56] ATMEL. *SAM3X / SAM3A Series, SMART ARM-based MCU, Atmel-11057C-ATARM-SAM3X-SAM3A-Datasheet*, 2015. http://www.atmel.com/ru/ru/Images/Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A_Datasheet.pdf.

[57] National Instruments. *Specifications NI myDAQ*, 2014. http://www.ni.com/pdf/manuals/373061f.pdf.

[58] Banzi Massimo; Cuartielles David; et. al. Arduino official web site, Jun 2013. http://www.arduino.cc.

[59] National Instruments. *USER GUIDE AND SPECIFICATIONS NI myRIO-1900*, 2013. http://www.ni.com/pdf/manuals/376047a.pdf.

Índice de figuras

Índice de tablas

Apéndice A

Códigos

A.1 Códigos de diseño

Envolvente suavizada

```
function [y1, data, xgraf] = Envelope_AAAC(data,fs)
%-------------------------------------------------------------------
% Este código fue realizado en la plataforma MATLAB.
%
%   data => es la señal de entrada de tamaño NxM siendo M<N,
%           de la cual se desea calcular la envolvente.
%   fs   => es la frecuencia de muestro de "data".
%   y1   => es un vector de 2*NxM, que contiene la envolvente
%            suavizada por el método de la transf. Hilbert
%            y por filtrado lineal.
%   xgraf=> es un vector de tiempos.
%-------------------------------------------------------------------
    xgraf=linspace(0,(max(size(data)))/fs,(max(size(data))));
%-------------------------------------------------------------------
%   Envolvente por método de la transformada de Hilbert
%-------------------------------------------------------------------
    HTD = hilbert(data);
    envh=abs(HTD);
%-------------------------------------------------------------------
```

```
%   Envolvente método de filtrado lineal
%-------------------------------------------------------------------
    [b,a]=butter(5, 15/fs,'high');
    envf=filter(b,a,data);
    envf1=abs(envf);
% Etapa de suavizado con filtro IIR Butterworth
    [l1,k1]=butter(5, 5/fs,'low');
    env=filter(l1,k1,envh);             %envolvente (transf. hilbert)
    env1=filter(l1,k1,envf1);           %envolvente (filtro lineal)
    y1=[env env1];
end
```

Filtro Wiener

```
function [y, ww] = FWiener_AAAC(data,target,P)
%-------------------------------------------------------------------
% Función para diseñar un filtro Wiener de orden "P".
%   Este código fue realizado en la plataforma MATLAB.
%
%   data   => es el vector de entrada.
%   target => es el vector deseado.
%   P      => es el orden del filtro Wiener.
%   ww     => es el vector de coeficientes de la respuesta al
%   impulso del filtro Wiener.
%   y      => es el vector estimado del vector deseado.
%-------------------------------------------------------------------
    [f1,c1]=size(data);
    [f2,c2]=size(target);
    if f2<c2
        target=target';
    end
    if f1>c1
```

```
        data=data';
    end
    [N,K]=size(target);
    r=xcorr(data);
    Rxx=toeplitz(r(N:N+P-1));
    ryx_=xcorr(target,data);
    ryx=ryx_(N:N+P-1);
    ww=inv(Rxx)*ryx;
    y=conv(ww,data);
end
```

Filtro LMS

```
function [yd,ydm, w, wm] = FLMS_AAAC(data,target,P,N1,mu,sel)
%--------------------------------------------------------------------
% Función para diseñar un filtro LMS de orden "P".
%   Este código fue realizado en la plataforma MATLAB.
%
%   data   => es el vector de entrada.
%   target => es el vector deseado.
%   P      => es el orden del filtro Wiener.
%   N1     => parámetro para el número de ciclos de adaptación(N1-P).
%   mu     => es factor de paso.
%   sel    => selector de método de actualización.
%   w      => es el vector de coeficientes de la respuesta al
%             impulso del filtro LMS.
%   wm     => es el vector de coeficientes de la respuesta al
%             impulso del filtro LMS promediado.
%   yd     => es el vector estimado del vector deseado con "w".
%   ydm    => es el vector estimado del vector deseado con "wm".
%--------------------------------------------------------------------
    [f1,c1]=size(data);
```

```
    [f2,c2]=size(target);
    x=data;
    d=target;
    if c2<f2
       d=target';
    end
    if f1<c1
       x=data';
    end
    w=zeros(P,N1-P);
    s=1;
    for i=P:1:N1-1
        xn=flip(x(i-P+1:i));
        y(s)=w(:,s)'*xn;
        e(s) = d(i) - y(s);
        switch(sel)
%-------------------------------------------------------------------
%                          Método Normal
%-------------------------------------------------------------------
        case 'normal'
            w(:,s+1) = w(:,s) + mu*(e(s)*xn);
%-------------------------------------------------------------------
%                  Método del signo del error
%-------------------------------------------------------------------
        case 'signo1'
            w(:,s+1) = w(:,s) + mu * sign(e(s))*xn;
%-------------------------------------------------------------------
%                  Método de signo del error 2
%-------------------------------------------------------------------
        case 'signo2'
            w(:,s+1) = w(:,s) + mu * sign(e(s)*xn);
        end
```

```
        s=s+1;
    end

    for t1=1:P
        wm(t1)=mean(w(t1,1:s-1));
    end
    wm=wm';
    yd=conv(w(:,s-1),x);
    ydm=conv(wm,x);
end
```

Filtro RLS

```
function [yd, w] = FRLS_AAAC(data,target,p,lam,M)
%--------------------------------------------------------------------
% Función para diseñar un filtro RLS de orden "P".
%   Este código fue realizado en la plataforma MATLAB.
%
%   data   => es el vector de entrada.
%   target => es el vector deseado.
%   p      => es el orden del filtro Wiener.
%   M      => parámetro para el número de ciclos de adaptación(M-P).
%   lam    => es factor de olvido.
%   w      => es el vector de coeficientes de la respuesta al
%             impulso del filtro RLS.
%   yd     => es el vector estimado del vector deseado con "w".
%--------------------------------------------------------------------
    [f1,c1]=size(data);
    [f2,c2]=size(target);
    x=data;
    d=target;
    if c2<f2
```

```
        d=target';
    end
    if f1<c1
        x=data';
    end
    k=zeros(p,1);
    w=zeros(p,M+1);
    lam1=1/lam;
    s=1;
    for n=p:M-p
        xn=flip(x(n-p+1:n));
        k(:,s) = (P*xn)*(lam1/(1+lam1*(xn')*P*xn));
        y(s)= w(:,s)'*xn;
        e(s) = d(n) - y(s);
        w(:,s+1) = w(:,s)+(k(:,s)*e(s));
        P = (lam1*P)-(k(:,s)*xn'*lam1*P);
        s=s+1;
    end
    yd=conv(x,w(:,s));
end
```

Extracción de características

```
function [FE] = FE_AAAC(data1,Fs,col,coh)
%--------------------------------------------------------------------
% Función para extraer los componentes espectrales de una señal.
%   Este código fue realizado en la plataforma MATLAB.
%
%   data1    => es el vector de entrada.
%   col   => es el índice de la frecuencia menor.
%   coh   => es el índice de la frecuencia mayor.
%   Fs => es la Frecuencia de muestreo de "data1".
```

```
%   FE => es el vector resultante con la banda deseada (de col a coh).
%--------------------------------------------------------------------
   [f c]=size(data1);
   b1=1;
   fdata=[];
   Fdata=[];
   aux=[];
   cc=[];
   ff=[];
   window=Fs-1;
   for i=1:max([f,c])
      fr=data1(b1:window+b1);
      fdata(b1:window+b1,:)=2*abs((fft(data1(b1:window+b1)))
                           /length(data1(b1:window+b1)));
      cc=[cc; data1(b1:window+b1)'];
      ff=[ff; fdata(b1:window+b1,:)'];
      NFFT=length(fdata(b1:window+b1,:));
      aFdata=fdata(b1:NFFT/2+b1,:);
      Fdata=[Fdata;aFdata];
      b1=b1+1;
      freq=Fs/2*linspace(0,1,NFFT/2+1);
      st=find(col-1<freq&freq<col+1);
      ven=find(coh-1<freq&freq<coh+1);
      aux=[aux;aFdata(st:en,:)'];
   end
   FE=aux;
end
```

Máquina de soporte vectorial

```
function [res] = SVM_AAAC(xin,svmstruct1,col,coh)
%--------------------------------------------------------------------
```

```
% Función para ejecutar una SVM  usando el comando "svmtrain",
%   esta estructura debera usar el kernel del polinomio de orden N.
%   Este código fue realizado en la plataforma MATLAB.
%
%   xin          => es el vector de entrada.
%   svmstruct1   => estructura generada por el comando "svmtrain".
%   res          => clase estimada.
%-----------------------------------------------------------------------
    sv = svmstruct1.SupportVectors;     % vectores de soporte
    alpha = svmstruct1.Alpha;           % alfas
    bias = svmstruct1.Bias;             % bias
    p=svmstruct1.KernelFunctionArgs{:}; % orden del polinomio
    escalar=1;
    if escalar==1
    if ~isempty(svmstruct1.ScaleData)
        for c = 1:size(xin, 2)
            xin(:,c) = svmstruct1.ScaleData.scaleFactor(c) * ...
            (xin(:,c) +  svmstruct1.ScaleData.shift(c));
        end
    end

    if ~isnumeric(p)
        p=3;
    end
    end
    pp = (sv*xin.');
    K = pp;
    for i = 2:p
        K = K.*(1 + pp);
    end
    auxc = K'*alpha + bias;
    res = -1*sign(auxc);
```

```
end
```

Error cuadrático promedio normalizado

```
function [nmse] = NMSE_AAAC(x,y)
%--------------------------------------------------------------------
% Función para calcular el error cuadrático promedio normalizado.
%   Este código fue realizado en la plataforma MATLAB.
%
%   x      => es el vector de entrada.
%   y      => es el vector deseado.
%   nmse   => es el error cuadrático promedio normalizado.
%--------------------------------------------------------------------
[f1,c1]=size(x);
[f2,c2]=size(y);
aux=max([c1 c2 f1 f2]);
e=0;
den=0;
for n=1:aux
e=e+(((y(n)-x(n))^2));
den=den+(y(n))^2;
end
nmse=e/den;
end
```

Interpolación

```
function [y] = Interpolation_AAAC(x, Fh,Fl)
%--------------------------------------------------------------------
% Función para interpolar una señal "x", usando la técnica
%   "zero padding".
%   Este código fue realizado en la plataforma MATLAB.
```

```
%
%   x    => es el vector de entrada y debe ser un vector fila,
%           de la cual se desea interpolar.
%   Fh   => es la frecuencia de muestro a la que se desea interpolar.
%   Fl   => es la frecuencia de muestro inicial.
%   y    => es el vector resultante y debe ser un vector fila,
%-------------------------------------------------------------------
   l1=Fh/Fl;
   [Nf,N]=size(x);
   X=(fft(x))/length(x);
   l=(l1-1)*N;
   Xz=[X(1,1:round(N/2)) zeros(1,round(l/2)) zeros(1,1)
   zeros(1,round(l/2)) X(1,round(N/2)+1:end)];
   y=real(ifft(Xz)*length(Xz));
end
```

Decimación

```
function [res]=downSampling_AAAC(var,Fmo,Fmd)
%-------------------------------------------------------------------
%   Función para disminuir la frecuencia de muestro.
%   Este código fue realizado en la plataforma MATLAB.
%   var -> es el vector de entrada.
%   Fmo -> es la frecuencia de muestreo de la señal original.
%   Fmd -> es la frecuencia de muestreo de la señal deseada.
%   res -> es el vector resultante.
%-------------------------------------------------------------------
   N=length(var);
   fact=round(Fmo/Fmd);
   s=1;
   for i=0:fact:N
      if i<N
```

```
         res(s)=var(i+1);
         s=s+1;
      else
         res(s)=var(i);
      end
   end
end
```

Normalización

```
function [y,a_max,a_min] = AAAC_norm(a)
%--------------------------------------------------------------------------
%   Función para normalizar un vector "a".
%   Este código fue realizado en la plataforma MATLAB.
%
%   a -> es el vector de entrada.
%   a_max -> valor máximo de "a".
%   a_min -> valor mínimo de "a".
%   y -> es el vector resultante.
%--------------------------------------------------------------------------
   a_max=max(a);
   a_min=min(a);
   y=(a-a_min)/((a_max-a_min)+.00000000000000001);
   if y>1
      y=1;
   end
   if y<0
      y=0;
   end
end
```

A.2 Código del hBMI en ATSAM3X8E

```
#include "sam.h"
#include "math.h"
double emg=0;
double eeg=0;
//matriz de la transf. de Fourier
float ftp[15][256]={...}; //parte real.
float fti[15][256]={...}; //parte imaginaria.
double Xp[15];
double Xi[15];
//var buffer de EEG
double bEEGt[256];
double bEEG[256];
double X[15];
double eeg_t=0;
%-----------------------------------------------------------------------
//modelo filtro FIR
float filter[2][500]={...};      //filtro Wiener.
double FIR[500];                 //filter memory.
double emg_t=0;
double dec=0;
uint32_t dec_aux=0;
%-----------------------------------------------------------------------
//modelo RNA MLP
double x1_step1_xoffset[15]={...};
double x1_step1_gain[15]={...};
signed char x1_step1_ymin=-1;
double b1[100]={...};
double IW1_1[15][100]={{...}};
double LW2_1[100]={...};
double b2=-0.34148433677486439;
```

```
%--------------------------------------------------------------------
signed char y1_step1_ymin = -1;
signed char y1_step1_gain = 1;
signed char y1_step1_xoffset = -1;
%--------------------------------------------------------------------
//variables auxiliares
unsigned int u=0;
unsigned int ind=0;
unsigned int o=0;
unsigned char sh=0;
unsigned char comp;
unsigned char lck=0;
unsigned char ciclo=0;
unsigned char Tarea_m[200];
double ANN_m[200];
%--------------------------------------------------------------------
//Functions
%--------------------------------------------------------------------
double FIR_filter(unsigned char tarea, unsigned int orden, double x);
void fourier_transform(unsigned char tam, unsigned int tam_v);
unsigned char svm(unsigned char lowbound,unsigned char upperbound,
                  unsigned char orden);
unsigned char ANN(unsigned char nX,unsigned char nHL,
                  unsigned char nOL);
double mean(unsigned char upperbound);
double absolute(double data);
unsigned char protector_FV(unsigned char tarea,
                           unsigned char tamano_memoria);
%--------------------------------------------------------------------
void TC0_Handler(void) //función que ejecuta a interrupción.
{
     uint32_t dummy;
```

```
    if (tarea==0)
    {
       PIOB->PIO_ODSR&=~PIO_PB27; //LED off
       PIOD->PIO_ODSR&=~PIO_PD2; //registra la vel2
    }
    else
    {
       PIOB->PIO_ODSR|=PIO_PB27; //LED on
       PIOD->PIO_ODSR|=PIO_PD2; //registra la vel1
    }
    dummy=TC0->TC_CHANNEL[0].TC_SR; //elimina bandera
    dummy+=1;
    if (lck==0) //el if es para multiplicar la fs.
    {
        ADC->ADC_CR=ADC_CR_START;             //Comienza el ADC
        while(((ADC->ADC_ISR)&0x07)!=0x07);
    eeg=((ADC->ADC_CDR[1])*3.3)/4095-.015;
emg=((ADC->ADC_CDR[2])*3.3)/4095-.015;
        eeg_t=(eeg/3)*((48.396-(-40.906))+.00000000000000001)
              +(-40.906);
        emg_t=emg*((0.70865-(-0.21738))+.00000000000000001)
              +(-0.21738);
        bEEGt[ind]=eeg/3;
        dec=2+(2.5*(FIR_filter(tarea,500,emg_t)));//filtro Wiener
        dec_aux=(dec*512)/0.85+1536;
        if (DACC->DACC_ISR&DACC_ISR_TXRDY)
        {
           DACC->DACC_CDR=dec_aux; //desplegamos en el DAC
        }
        ind+=1;
        if (ind>255)//255
        {
```

```
            comp=1;
            sh=0;
            ind=0;
        }
        asm("NOP");
        lck=0;
        o+=1;
    }
 else
    {
        lck+=0;
    }
    if (sh==0)     //actualiza el buffer.
    {
         for (u=0;u<256;u++)
         {
             bEEG[u]=bEEGt[u];
         }
         sh=1;
    }

}
%-----------------------------------------------------------------------
static void configure_dac(void) //Función para configurar DAC.
{
    //Declare a clock to the DAC peripheral.
    PMC->PMC_PCER1|=PMC_PCER1_PID38;    //Activamos el reloj del DAC
    //DACC Channel Enable
    DACC->DACC_CHER=DACC_CHER_CH0;      //Activamos el canal 0.
}
%-----------------------------------------------------------------------
static void configure_adc(void) //Función para configurar ADC.
```

```
{
     //Declare a clock to the ADC peripheral.
     PMC->PMC_PCER1|=PMC_PCER1_PID37;
     //ADC Channel Enable
     ADC->ADC_CHER|=ADC_CHER_CH1|ADC_CHER_CH0|ADC_CHER_CH2;
     //Activamos los canales que vamos a ocupar
     //ADC Sequence 1
     ADC->ADC_SEQR1=0x00000210;
     //Se realiza primero el canal 0 y despues el canal 1
     //Analog Settling Time
     ADC->ADC_MR|=ADC_MR_FREERUN; //elegimos Free-run mode
     //Analog Settling Time
     ADC->ADC_MR|=ADC_MR_USEQ_REG_ORDER; //elegimos Free-run mode
     ADC->ADC_CR=ADC_CR_START; //Comienza la conversión
}
%--------------------------------------------------------------------
static void configure_tc0(void) //Función para configurar el contador.
{
     // Enable TC0 (27 is TC0)
     PMC->PMC_PCER0 = 1 << 27;
     // Disable TC clock
     TC0->TC_CHANNEL->TC_CCR = TC_CCR_CLKDIS;
     // Disable interrupts (del contador TC0)
     TC0->TC_CHANNEL->TC_IDR = 0xFFFFFFFF;
     // Clear status register (se limpia con solo leerlo).
     TC0->TC_CHANNEL->TC_SR;
     // Set Mode
     TC0->TC_CHANNEL->TC_CMR = TC_CMR_CPCTRG |
                               TC_CMR_TCCLKS_TIMER_CLOCK5;
     // Compare Value(Se establece el valor del registro C.)
     TC0->TC_CHANNEL[0].TC_RC = 1503; //110=3.9ms o 256Hz, 50ms=1600
     // funcion que habilita las posibles interrupciones.
```

```
    NVIC_EnableIRQ((IRQn_Type) ID_TC0);
    //Se configura la interrupción cuando la cuenta del
    contador sea igual a la del registro.
    TC0->TC_CHANNEL->TC_IER = TC_IER_CPCS;
    // Reset counter (SWTRG) and enable counter clock (CLKEN)
    TC0->TC_CHANNEL[0].TC_CCR = TC_CCR_CLKEN | TC_CCR_SWTRG;
}
%--------------------------------------------------------------------
static void configure_io(void) //Función que configura los PIO
{
    // Enable IO
    PIOB->PIO_PER = PIO_PB27; //LED Arduino.
    PIOD->PIO_PER = PIO_PD0; //PiN de inicio.
    PIOD->PIO_PER = PIO_PD2; //PIN del estado del Clasificador
    // Set to output
    PIOB->PIO_OER = PIO_PB27;
    PIOD->PIO_OER = PIO_PD0;
    PIOD->PIO_OER = PIO_PD2;
    // Enable write
    PIOB->PIO_OWER = PIO_PB27;
    PIOD->PIO_OWER = PIO_PD0;
    PIOD->PIO_OWER = PIO_PD2;
    // Disable pull-up
    PIOB->PIO_PUDR = PIO_PB27;
    PIOD->PIO_PUDR = PIO_PD0;
    PIOD->PIO_PUDR = PIO_PD2;
}
%--------------------------------------------------------------------
%--------------------------------------------------------------------
int main(void) //Programa principal del MCU
{
    /* Initialize the SAM system */
```

```
     SystemInit();
     configure_dac();     //Iniciamos DAC
     configure_adc();     //Iniciamos ADC
     configure_io();      //Iniciamos PIOs
     PIOD->PIO_ODSR&=~PIO_PD0; //iniciamos mydaq
     for (uint32_t v=0;v<410000;v++) //delay.
     {
          asm("NOP");
     }
     configure_tc0();     //Iniciamos contador.
while (1)
{
     if (comp==1)
     {
          fourier_transform(15,256);
          //tarea=svm(15,86,5);         //Clasificación por SVM.
          tarea=ANN(15,100,1);          //Clasificador(ANN 15-100-1)
          tarea=protector_FV(tarea,3);  //TFP
          comp=0;//comp=1->VT,comp=0->VST
          ciclo+=1;
          sh=0;
          if (ciclo==118) //frena simulación
          {
               asm("NOP");
          }
     }
}
}
%-------------------------------------------------------------------
%-------------------------------------------------------------------
double mean(unsigned char upperbound)     //Función promedio.
{
```

```
     unsigned i=0;
     double aux=0;
     double aux1=0;
     for (i=0;i<=(upperbound-1); i++)
     {
          aux=memoria_tarea[i]+aux;
     }
     aux1=aux/upperbound;
     return aux1;
}
/*unsigned char svm(unsigned char lowbound,unsigned char upperbound,
unsigned char orden)
{
     unsigned char i=0;
     unsigned char j=0;
     unsigned char tarea=0;
     double xin[lowbound];
     double aux1=0;
     double aux;
     double K[upperbound];
     //scaling & shifting
     for(j=0;j<=(lowbound-1);j++)
     {
     //xin[j]=(input_xn[j]+ssF[1][j])*ssF[0][j];
     xin[j]=(X[j]+ssF[1][j])*ssF[0][j];
     }
     //dot product
     for (i=0;i<=(upperbound-1);i++)
     {
          for(j=0;j<=(lowbound-1);j++)
          {
               aux=xin[j]*sv_alp[i][j];
```

```
            aux1=aux1+aux;
        }
        pp[i]=aux1;
        K[i]=pp[i];
        aux1=0;
        aux=0;
    }
    //Kernel Function
    for (i=1;i<=(orden-1);i++)
    {
        for(j=0;j<=(upperbound-1);j++)
        {
            K[j]=K[j]*(1+pp[j]);
        }
    }
    aux=0;
    //dot product
    for (i=0;i<=(upperbound-1);i++)
    {
        aux=K[i]*sv_alp[i][lowbound];
        aux1=aux1+aux;
    }
    aux=aux1+bias;
    SVM_m[ciclo]=aux;
    aux=aux*(-1)-1e11;
    SVM_m2[ciclo]=aux;
    if (aux>0)
    {
        tarea=1;      //tarea negativa
    }
    else
    {
```

```
          tarea=0;      //tarea positiva
     }


return tarea;
}*/
double absolute(double data)     //Función modulo.
{
     double aux1=0;
     if (data<=0)
     {
          aux1=data*(-1);
     }
     else
     {
          aux1=data;
     }
     return aux1;
}
double FIR_filter(unsigned char tarea, unsigned int orden, double x)
{                    //Filtro Wiener
     signed int u=0;
     unsigned int v=0;
     double aux;
     double aux1=0;
     //memory for convolution
     for (u=(orden-1);u>=1;u--)
     {
          FIR[u]=FIR[u-1];
     }
     FIR[0]=x;
     //convolution (dot product)
     for (v=0;v<=(orden-1);v++)
```

```
     {
          aux=FIR[(orden-1)-v]*filter[tarea][v];
          aux1=aux1+aux;
     }
     return aux1;
}
unsigned char protector_FV(unsigned char tarea,unsigned char tamano_memoria)
{              //Ejecuta el protector contra falsos positivos.
     signed char u=0;
     double aux=0;
     unsigned char res=0;
     signed char tarea_actual=0;
     if(tarea==0)
     {
          tarea_actual=1;
     }
     else
     {
          tarea_actual=-1;
     }
     aux=absolute(mean(3));

     if(tarea_actual!=memoria_tarea[1])
     {
          if(aux<=0.5)
          {
               res=tarea_actual;
          }
          else
          {
               res=memoria_tarea[1];
          }
```

```
     }
     else
     {
          res=tarea_actual;
     }
     //memory update
     for (u=(tamano_memoria-1);u>=1;u--)
     {
          memoria_tarea[u]=memoria_tarea[u-1];
     }
     memoria_tarea[0]=tarea_actual;
     if(res==1)
     {
          tarea=0;
     }
     else
     {
          tarea=1;
     }
     return tarea;
}
void fourier_transform(unsigned char tam, unsigned int tam_v)
{
     unsigned char i = 0;
     unsigned int j = 0;
     double aux=0;
     double aux1=0;
     for (i = 0; i <= (tam - 1); i++)
     {
          Xp[i]=0;
          Xi[i]=0;
          for (j = 0; j <= (tam_v-1); j++)
```

```
          {
               aux = bEEG[j] * ftp[i][j];
               Xp[i] = Xp[i] + aux;
               aux1 = bEEG[j] * fti[i][j];
               Xi[i] = Xi[i] + aux1;
          }
     }
     for (i = 0; i <= (tam - 1); i++)
     {
          X[i]=sqrt((Xp[i]*Xp[i])+(Xi[i]*Xi[i]))/256;
          X[i]*=2;
     }
}
unsigned char ANN(unsigned char nX,unsigned char nHL,unsigned char nOL)
{
     unsigned char i=0;
     unsigned char j=0;
     unsigned char tarea=0;
     double xin[nX];
     double fi[nHL];
     double aux1=0;
     double aux=0;
     //escalar la entrada
     for (i=0;i<=(nX-1);i++)
     {
          xin[i]=X[i]-x1_step1_xoffset[i];
          xin[i]=xin[i]*x1_step1_gain[i];
          xin[i]=xin[i]+x1_step1_ymin;
     }
     for (i=0;i<=(nHL-1);i++)
     {
          for (j=0;j<=(nX-1);j++)
```

```
          {
               aux=aux+xin[j]*IW1_1[j][i];
          }
          aux=aux+b1[i];
          fi[i]=(2/(1+exp(-2*aux)))-1;
          aux=0;
     }
     for (i=0;i<=(nOL-1);i++)
     {
          for (j=0;j<=(nHL-1);j++)
          {
               aux=aux+fi[j]*LW2_1[j];
          }
          aux=aux+b2;
          aux1=aux-(y1_step1_xoffset);
          aux1=aux1*y1_step1_gain;
          aux1=aux1+y1_step1_ymin;
     }
     ANN_m[ciclo]=aux1;
     if (aux1>0)
     {
          tarea=1;      //tarea positiva
     }
     else
     {
          tarea=0;      //tarea negativa
     }
     Tarea_m[ciclo]=tarea;
     return tarea;
}
```

Apéndice B

hBMI en NI-myRIO

Generalmente, en términos de desarrollo e innovación, se inicia el diseño sobre la plataforma disponible que más recursos tenga. En cuestión de hardware, una alternativa a nuestra disposición con muchos recursos computacionales, fue el dispositivo NI myRIO (Figura B.1), el cual fue la primera opción para la implementación de la hBMI. Este dispositivo se selecciono por las siguientes características [59]:

- Procesador: Xilinx Zynq-7010 (MCU-FPGA), a 667MHz, 256 MB de almacenamiento no volátil, 512 MB DRAM para control y análisis determinanticos.
- FPGA Xilinx Z-7010 (cerca de 430k celdas lógicas) reconfigurable para el procesamiento en línea y control personalizado.
- 4 entradas analógicas de 12 bits, 2 salidas analógicas de 12 bits.
- Compatibilidad con el software Labview y Xilinx ISE.

Figura B.1: Plataforma de desarrollo myRIO-1900.

El myRIO se programa utilizando el software LabVIEW®, el cual maneja un entorno grafico para efectuar la programación, tal como se muestra en las Figuras B.3 y B.2.

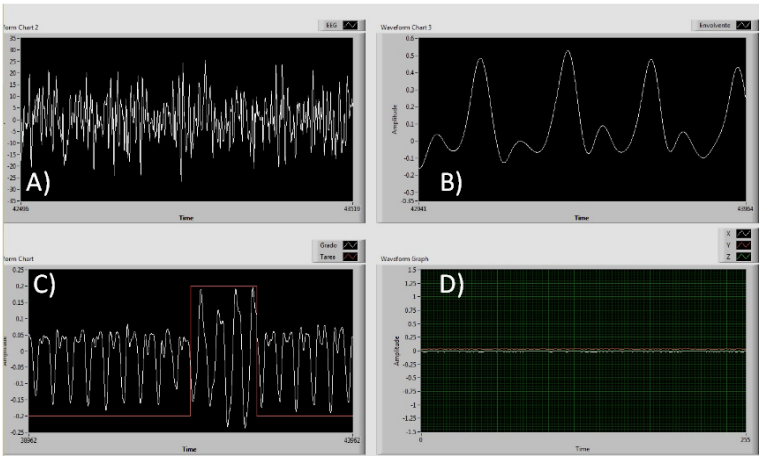


Figura B.2: Panel frontal de la hBMI, donde se muestra en A) la señal EEG, B) la señal EMG C) la salida de la hBMI y D) la señal del acelerómetro.

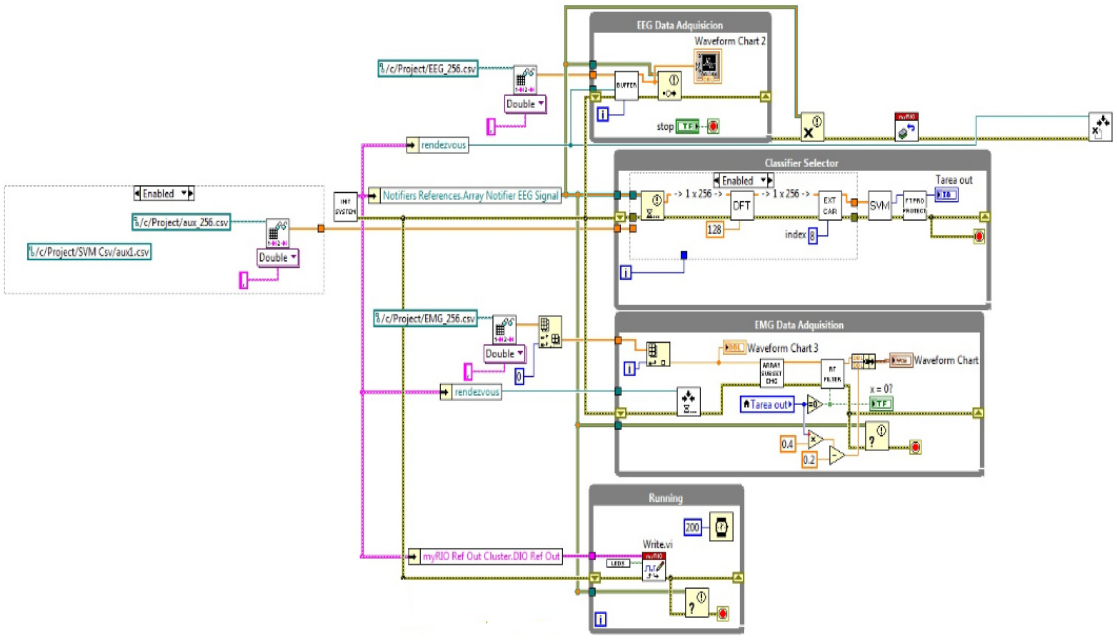


Figura B.3: Diagrama de bloques de la hBMI.

En la Figura B.2 se muestra el panel frontal de la interfaz durante la simulación de la hBMI (como en el capítulo 4), la gráfica superior izquierda despliega la señal

EEG, mientras que la derecha muestra la envolvente de la señal EMG y la gráfica inferior izquierda muestra la estimación del TFA y la tarea del usuario. El diagrama de bloques de la Figura B.3 representa la programación del sistema (análoga a la Figura 4.2). A continuación se muestran cada uno de los subVI's que componen el diagrama de la Figura B.3.

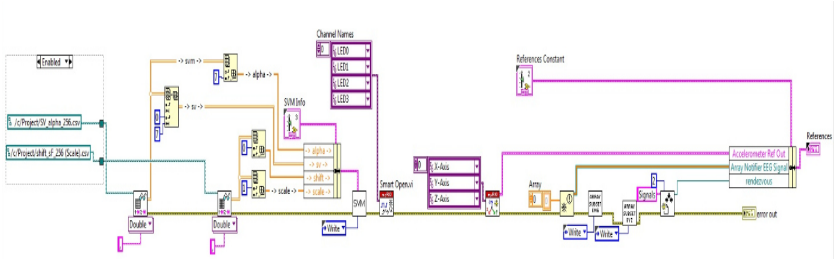
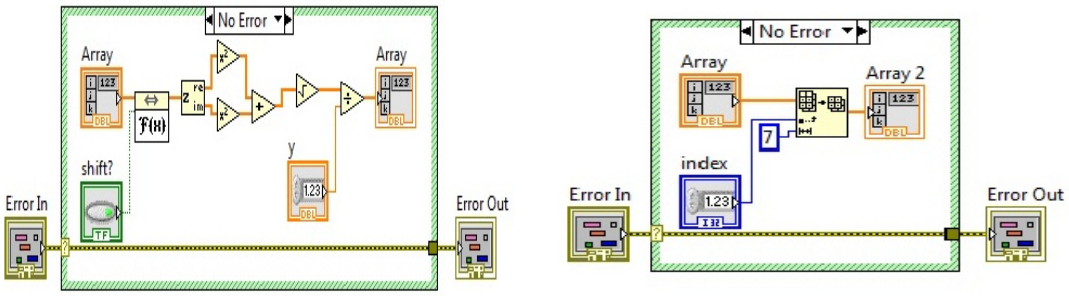
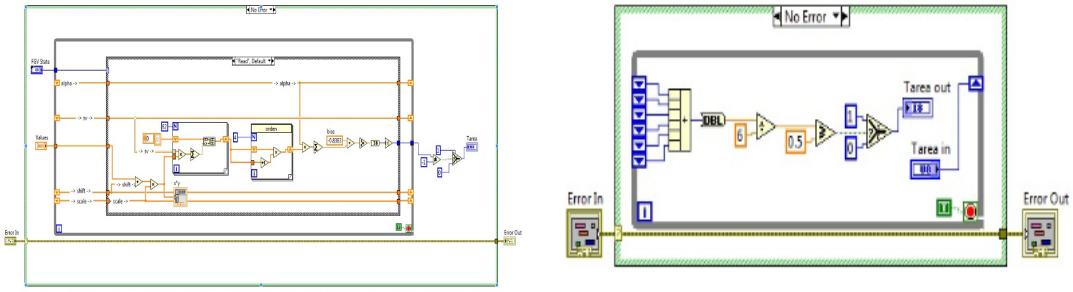


Figura B.4: Etapa de inicialización del sistema.



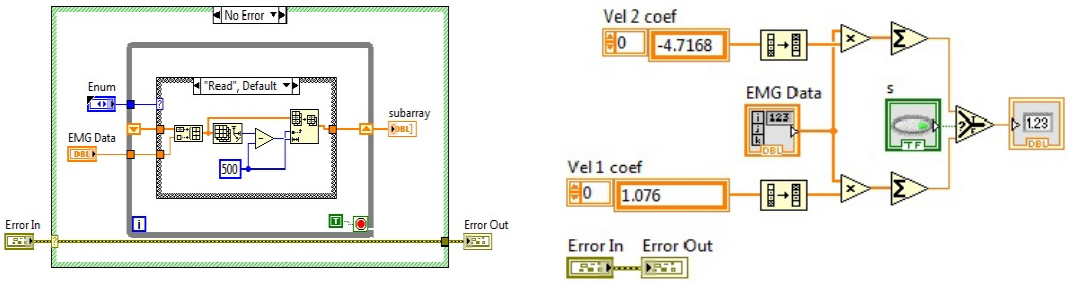
(a) (b)

Figura B.5: Bloques de la a)Etapa DFT (Transformada discreta de Fourier) y de la b)Etapa de extracción de características.



(a) (b)

Figura B.6: Bloques de la a)Etapa SVM y de la b)Etapa protector de falsos verdaderos.



(a) (b)

Figura B.7: Bloques de la a)Etapa buffer EMG y de la b)Etapa del filtro FIR.

Este dispositivo se considero como una alternativa, debido a que se tenia pensado explorar los dispositivos FPGA en caso de que la velocidad de un MCU no fuera lo suficientemente rápido. A pesar de ser dispositivo funcional, muchas de las características de la tarjeta (como el acelerómetro, wifi, entradas y salidas de audio, etc.) no se utilizaron, por lo que el siguiente paso sería la optimización del diseño, esto se consiguió migrando a la plataforma Arduino DUE (ATSAM3X8E) tal como se describe en el capítulo 4.

Apéndice C

Productividad académica

El siguiente artículo con título “A Classifier/decoder of Kinematic Tasks During the Human Gait Cycle” fue aceptado para presentarse como poster en “38th AnnualInternational Conference of the IEEE Engineering in Medicine and Biology Society (EMBC?16)” llevada acabo en Disney?s Contemporary Resort at Walt Disney WorldrResort, Lake BuenaVista (Orlando), Florida USA, del 17 al 20 de Agosto del 2016, resultando como productode la investigación realizada en esta tesis.

A classifier/decoder of kinematic tasks during the human gait cycle

A. Alonso-Carreón, M. Platas-Garza, G. Quiroz

***Abstract*— We propose a Hybrid-Brain Machine Interface (hBMI) that uses electromyographic (EMG) and electroencephalographic (EEG) signals to estimate the tibio-femoral angle (TFA) of the human gait cycle (HGC). A support vector machine (SVM) is used to classify the operating speed of the user from EEG signals. The superficial EMG signals and a bank of Wiener Filters (WF) are used to decode the TFA.**

I. Introduction

In recent years, there has been an increasing attention in the design of external devices that can recognize the user's intention of movement; since there are many people with motor disabilities. These devices require a decodification of neural and muscular signals for different task. For the HGC, methods to solve these problems rely on BMI's, nevertheless, for some tasks, the neural signal is not enough to reproduce the action, and hBMI's are required. However, there are few works for the hBMI's [1], and generally, the decodification is limited to a certain operation point established by the training data [1]-[4]. Therefore, in this work we propose a classifier/decoder scheme for EEG/EMG to estimate a kinematic variable during the HGC as shown in Fig 1. The scheme uses a classifier to switch between two decoders specifically designed to operate for each speed tested.

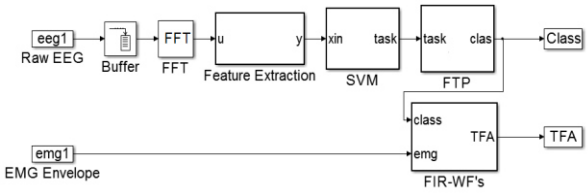


Figure 1: The propose hBMI system in MATLAB/Simulink®

II. Procedures and Results

In order to acquire the EEG/EMG/TFA data, this activity was recorded in three patients (23-25 years old) during 10 repetitions of HGC in a treadmill at different speeds. The speed of operation changes every minute from 4 (V1) to 6 (V2) and then to 4 MPH again. Next we present the design.

Classification. The EEG was acquired through the Alert X10 system. We use the C3 electrode, since it is related to the movement of the right side of the body. The EEG signal was divided into non-overlapping windows of 256 samples each. The FFT was used to compute the spectrum. We chose α and β bands from the spectrum, creating a vector of 7 features for each window. This feature vector was the input to the classifier.

*Autors are from Universidad Autónoma de Nuevo León, FIME
Av. Universidad S/N, Ciudad Universitaria, C.P. 66451, San Nicolás de los Garza, Nuevo León, México. Phone: +52 81 83294020 ext. 5773.

A SVM was trained with a polynomial function of 5 order as the Mercer kernel function. 3 of the 10 repetitions were used to train the SVM generating a total of 830 feature vectors and a classification accuracy of 86.77%. Finally, we add a false positives protector (FTP), based in a moving average filter of 3th order with a threshold, improving the performance of the classifier to 93.05%.

Decodification. The EMG was acquired through the MP36 system from BIOPAC. Disposable electrodes were placed in quadriceps femoris and vastus medialis oblique. The envelope of the EMG was computed by applying the modulus of a Hilbert Transform, and a smoothing phase. For each speed, we design a 500 order FIR-WF, with the envelope as input and the TFA (recorded by a vision system) as target.

*Results***.** The results of the classification/decodification scheme are shown in Fig 2 for the speeds V1 and V2. The normalized mean square error was used to compare the performance of both filters (without classifier) and the commutated system (hBMI) and the results were 0.0376, 0.0219 and 0.0183 respectively. Therefore, it shows an improvement in the decoding performance.

Furthermore, the computational complexity is low in comparation with other schemes as neural networks [3]-[4].

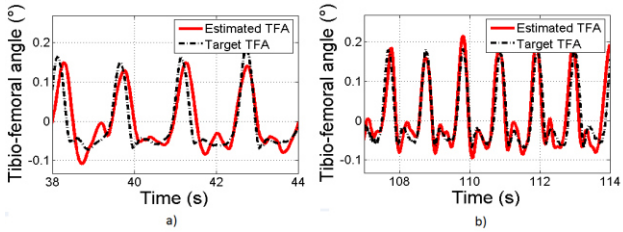


Figure 2: Estimated angle for a) V1 and b) V2, the TFA was normalized between 34.26° and -156.27°.

Acknowledgment

Authors are members of the research network "Biosystems and Biomechatronics" and thank to PRODEP-SEP for financial support. Aldebaran Alonso thanks to CONACYT for the scholarship grants number 390751.

References

[1] J.R. Millán, et al., "Combining brain–computer interfaces and assistive technologies: state-of-the-art and challenges", *Frontiers in neuroscience*, vol. 4, pp. 161, 2010.

[2] G. Chéron, et al., "From spinal central pattern generators to cortical network: integrated BCI for walking rehabilitation", *Neural plasticity,* vol. 2012, pp.1-13, 2012.

[3] Md R. Ahsan, et al., "EMG signal classification for human computer interaction: a review", *European Journal of Scientific Research*, vol. 33, no 3, pp. 480-501, 2009.

[4] T. Castermans, et al., "Towards effective non-invasive brain-computer interfaces dedicated to gait rehabilitation systems", *Brain sciences*, vol. 4, no 1, pp. 1-48, 2013.

Preprint submitted to 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. Received May 13, 2016.

38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society

UANL
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

A CLASSIFIER/DECODER OF KINEMATIC TASKS DURING THE HUMAN GAIT CYCLE.

A. Alonso-Carreón, M. Platas-Garza, G. Quiroz
Posgrado en Ingeniería Eléctrica
Facultad de Ingeniería Mecánica y Eléctrica, Universidad Autónoma de Nuevo León
{aldebaran.alonsocr, miguel.platasgrz, griselda.quirozcm}@uanl.edu.mx

Abstract

We propose a "Hybrid-Brain Machine Interface" (hBMI) that uses electroencephalographic (EEG) and electromyographic (EMG) signals to estimate the tibio-femoral angle (TFA) of the human gait cycle (HGC). A support vector machine (SVM) is used to classify the operating speed of the user from EEG signals. The superficial EMG signals and a bank of Wiener Filters (WF) are used to decode the TFA.

Introduction

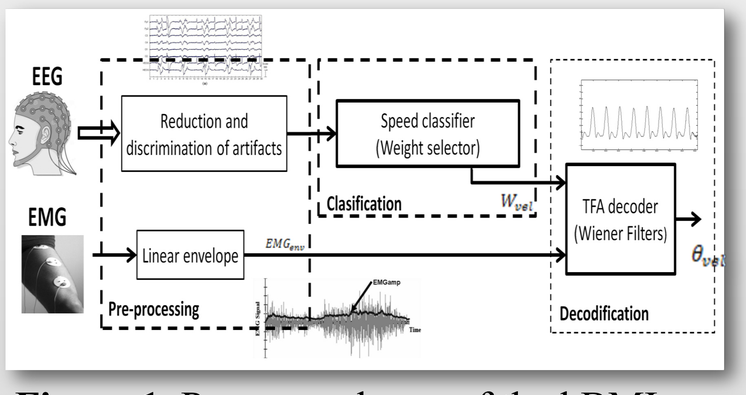


Figure 1. Propose scheme of the hBMI.

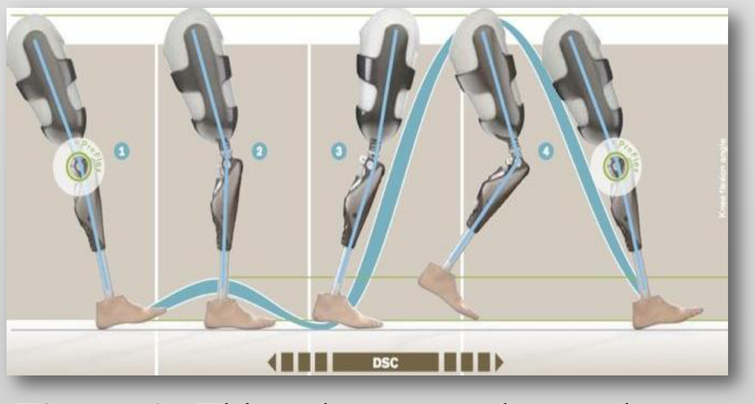


Figure 2. This scheme can be used to control assistive devices.

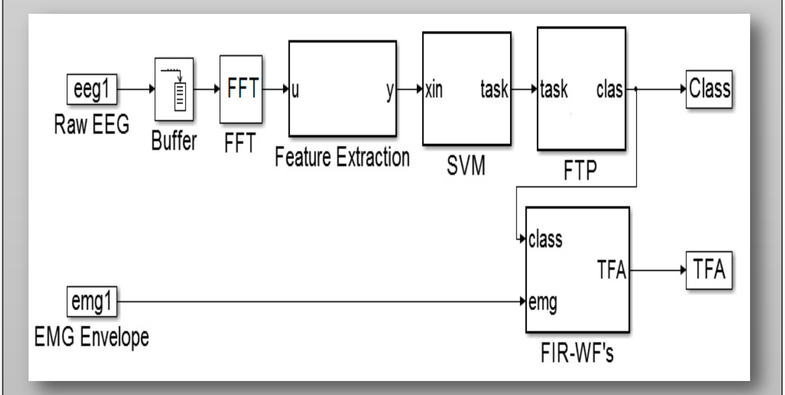


Figure 3. Propose scheme.

Results

Decoding with Wiener filters.

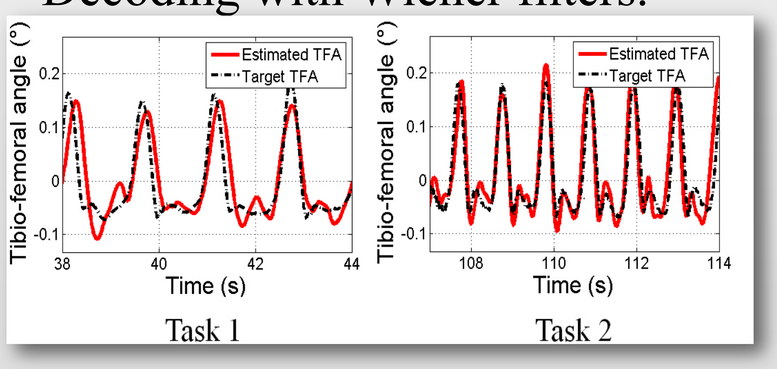


Figure 4. Results of the decoder for each task: speed 1 (4 mph) and speed 2 (6 mph). For each task, we design a 500 order FIR-WF, with the envelope of the EMG as input and the TFA as desired response. The TFA was normalized between 34.26° and -156.27°.

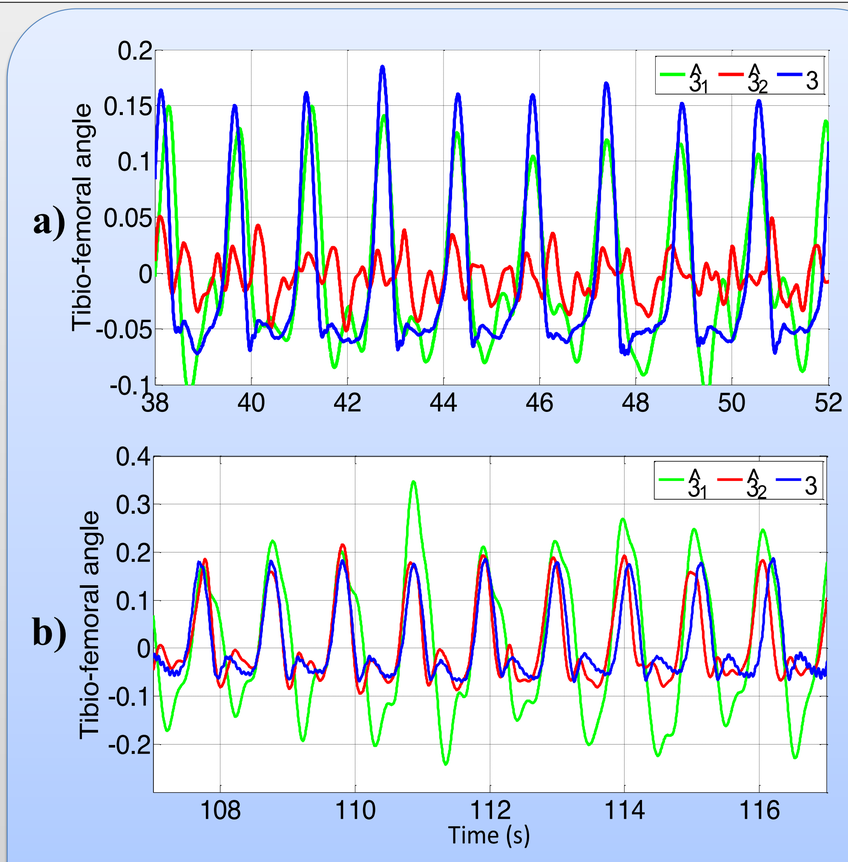


Figure 5. As shown in the graphs, each filter has an independent operation point.

Classification with SVM.

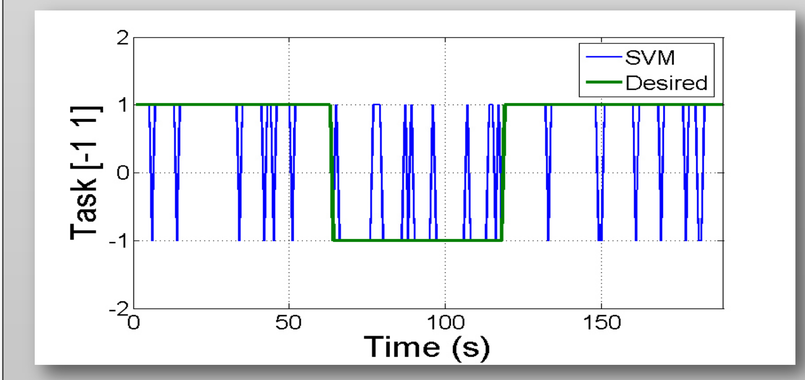


Figure 6. Results of the classifier (SVM) with 86.77% of precision. We use the EEG signal as input (specifically the C3 electrode and the frequency bands α and β as the feacture vector) to estimate the user intent of speed.

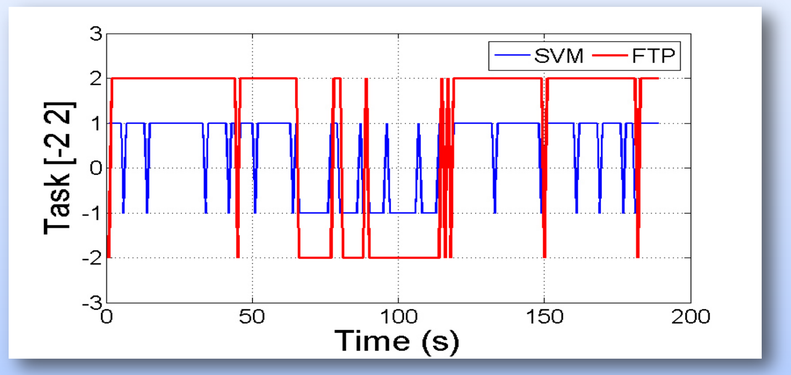
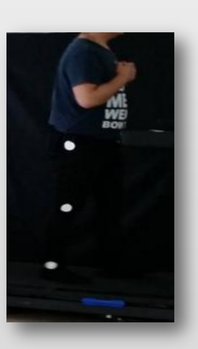


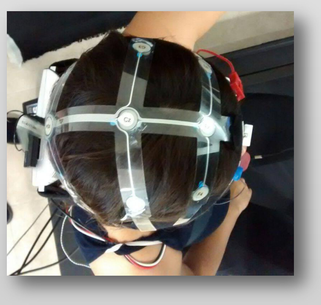
Figure 7. Adding a false positive protector (moving average filter) the performance improves to 93.07%.

Experimental protocol



The EEG (classifier input) was acquired through the B-Alert X10® system from Advanced Brain Monitoring Ltd.

The TFA was recorded by a vision system as the target signal.



The EMG (decoder's input) was acquired with the MP36® system from BIOPAC in quadriceps femoris and vastus medialis oblique.

Summery

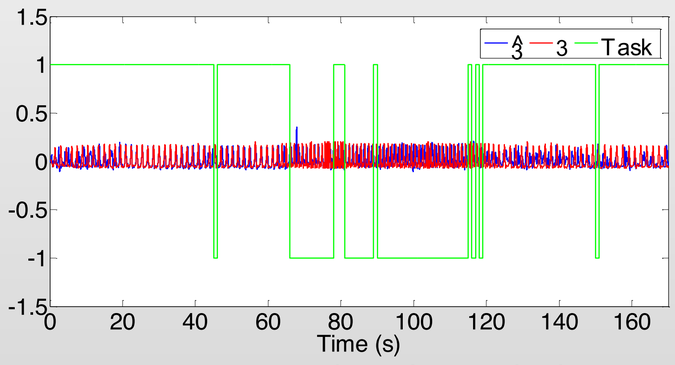


Figure 8. Estimated tasks and the TFA (normalized) during all the experiment.

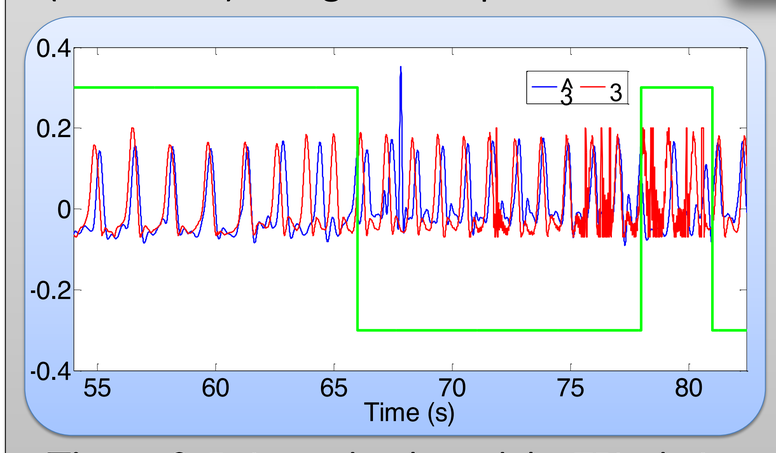


Figure 9. Estimated tasks and the TFA during a commutation between tasks.

Normalized mean square error:

$$NMSE = \frac{\sum_{n=0}^{N}[T(n) - \hat{T}(n)]^2}{\sum_{n=0}^{N}[T(n)]^2}$$

Experiment state	Speed I filter	Speed 2 filter	hBMI
Speed I	0.0038	0.0085	0.0043
Speed 2	0.0338	0.0134	0.0140
Total	0.0376	0.0219	0.0183

We made a comparison between the desired signal and the estimations of the filters by using the normalized mean square error (NMSE) as a criterium to evaluate the performance of the hBMI system.

Conclusion

The EEG signal pattern during the HGC at different speeds seems to have distinctions. This differences are used to improve the decoding process by commuting filters designed for each task.

References

[1] Millán J d. R, et al.(2010) Combining Brain–Computer Interfaces and Assistive Technologies: State-of-the-Art and Challenges .*Frontiers in Neuroscience*; 4:161.
[2] G. Cheron, et al. (2012), From Spinal Central Pattern Generators to Cortical Network: Integrated BCI for Walking Rehabilitation, Neural Plasticity, vol. 2012, Article ID 375148, 13 pages.
[4] Ahsan, Md R., et al. (2009) EMG signal classification for human computer interaction: a review. European Journal of Scientific Research, vol. 33, no 3, p. 480-501.
[5] Castermans, T.; Duvinage, M.; Cheron, G.; Dutoit, T. Towards. (2014) Effective Non-Invasive Brain-Computer Interfaces Dedicated to Gait Rehabilitation Systems.BrainSci.4,1-48.

Authors are members of the research network "Biosystems and Biomechatronics" and thank to PRODEP-SEP for financial support (103.5/15/14156) .Aldebaran Alonso thanks CONACYT for granting the scholarship number 390751.