UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño de un sistema inteligente de monitoreo de ondas EEG y generador de pulsos binaurales para combatir desórdenes de sueño en los atletas

Trabajo de graduación presentado por José Pablo Muñoz Núñez para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2019

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño de un sistema inteligente de monitoreo de ondas EEG y generador de pulsos binaurales para combatir desórdenes de sueño en los atletas

Trabajo de graduación presentado por José Pablo Muñoz Núñez para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2019

Vo.Bo.:

(f) _____

Dr. Luis Alberto Rivera

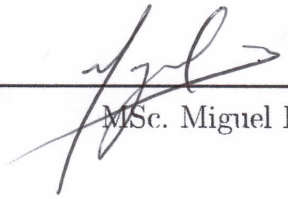
Tribunal Examinador:

(f) _____

Dr. Luis Alberto Rivera

(f) _____

MSc. Carlos Alberto Esquit



(f) _____

MSc. Miguel Enrique Zea

Fecha de aprobación: Guatemala, 5 de diciembre de 2019.

Prefacio

Guatemala es un país que se encuentra en una constante lucha contra una gran cantidad de problemáticas sociales tales como la corrupción, la desnutrición y la falta de educación. Día a día los guatemaltecos tienen que enfrentarse a distintas situaciones relacionadas a estas problemáticas que hacen que la realidad actual de nuestro país sea complicada. Sin embargo, dejando momentáneamente todo el escenario negativo en el que estamos sumergidos, tampoco hay que olvidar que también somos un país que tiene un gran potencial en el área científica, social y deportiva. Es por eso que todos aquellos que tuvimos el privilegio de estudiar o de contar con más oportunidades y recursos que el resto, debemos hacer todo lo posible para retribuirle de alguna forma a la sociedad que nos vio crecer.

Los atletas guatemaltecos constantemente han demostrado que tienen todo el talento y potencial para competir al máximo nivel internacional. Brindarles más herramientas y equipos para su preparación puede ser un diferenciador importante que les permita lograr aun mejores resultados. Aunque este trabajo sea el desarrollo de una primera fase que todavía no es aplicable para hacer experimentación, con más años de trabajo se puede llegar a desarrollar un sistema práctico que les permita a los atletas dormir mejor y aliviar las altas presiones que tienen que manejar día a día. Además, un dispositivo que permita hacer más eficiente el sueño de las personas tiene el potencial de ser un aporte científico que ponga en alto el nombre de Guatemala.

Para el desarrollo de este trabajo se le da especial agradecimiento a la Universidad del Valle de Guatemala por brindarme a lo largo de 5 años de carrera todas las capacidades, herramientas y destrezas para ser un miembro productivo y con valores en la sociedad. Además, también se le agradece al Ing. Luis Pedro Montenegro por el constante acompañamiento, disponibilidad y aportes que tuvo durante el transcurso del año para la elaboración de este proyecto. También se le agradece al Dr. Luis Alejandro López por sus compartir sus conocimientos y por el gran interés y disponibilidad que tuvo en apoyar esta investigación. Y finalmente, un especial agradecimiento a mi asesor personal, el Dr. Luis Alberto Rivera, por su increíble dedicación, compromiso y motivación que proporcionó durante todo el año y que fue fundamental para presentar estos valiosos resultados para la Universidad del Valle y la comunidad guatemalteca.

Índice

Lista de figuras

Lista de cuadros

Resumen

Este proyecto consistió en el desarrollo de una primera fase, dividida en cinco etapas, de un sistema de clasificación automática de las etapas del sueño. Este sistema una vez finalizado sería capaz de generar pulsos binaurales para combatir trastornos del sueño en los atletas. En la primera etapa se realizó una conexión efectiva entre un Electro-Cap y una Cyton Board de OpenBCI para hacer lecturas de las ondas cerebrales. Posteriormente, en la segunda etapa se desarrolló un algoritmo en Python capaz de generar pulsos binaurales de cualquier frecuencia y duración, en donde los audios finales fueron examinados en el analizador de espectros de la herramienta Audacity y Sonic Visualizer. La tercera etapa consistió en la adquisición de los datos crudos en tiempo real de la Cyton Board, lo cual se logró mediante un algoritmo en Python que también permitía almacenar los datos leídos en un archivo CSV.

En la cuarta etapa se realizó el procesamiento de los datos mediante el diseño e implementación de filtros digitales en tiempo real. Para validar esta etapa las señales filtradas se compararon con las señales de la interfaz de OpenBCI y se pudieron obtener exactamente los mismos resultados. Finalmente, en la última etapa se utilizó la base de datos Sleep-EDF Database para adquirir polisomnografías de múltiples personas junto a sus hipnogramas respectivos. Se extrajeron las características Mean Absolute Value (MAV) y Zero Crossing (ZC) de cuatro distintas señales presentes en las polisomnografías para entrenar una red neuronal (RN) y una máquina de vectores de soporte (SVM). Las mejores clasificaciones fueron obtenidas a partir de las RN, en donde se llegaron a obtener rendimientos superiores al 90 % al clasificar las etapas del sueño según las guías de la Academia Estadounidense de Medicina del Sueño (AASM).

Abstract

This is an abstract of the development of the first stage of a project that aims to be an Automatic Sleep Stage Classification (ASSC) system that is capable of playing binaural beats to help atheltes deal with sleep disorders. In the first stage of the project the connection between an OpenBCI Cyton Board and an Electro-Cap was achieved in order to read and observe brainwaves in real time. In the second stage, an algorithm was developed in Python capable of generating binaural beats of any desired frequency and duration. The results of this stage were validated using Audacity and Sonic Visualizer spectrum analyzer. The next stage consisted in reading the Cyton Board raw data in real time, which was also accomplished with a Python code that simultaneously saves a registry of all the data that is read in a CSV file.

The fourth stage consisted of a Digital Signal Processing (DSP) where filters were designed and applied to the raw data in real time. The results of the DSP were compared with the OpenBCI user interface signals and the results were identical. Finally, for the final stage, the Sleep-EDFx Database was used in order to acquire biosignals from sleep studies along with their hypnograms. The time domain features, Mean Absolute Value (MAV) and Zero Crossing (ZC), were extracted from four different signals available in the database recordings in order to train two types of classifiers, a neural network (NN) and a Support Vector Machine (SVM). The best performances were obtained by neural networks with accuracies over 90 % when classifiyng the sleep stages according to the American Academy of Sleep Medicine (AASM).

CAPÍTULO 1

Introducción

A pesar de los grandes avances en la ciencia en las últimas décadas, muchos aspectos del sueño de una persona siguen siendo un enigma para la comunidad científica. Aunque falten años de investigaciones para comprender en su totalidad esta necesidad biológica, con los estudios actuales es evidente que se trata de una actividad indispensable para la recuperación física y psicológica del organismo. Aún cuando se reconoce la importancia y beneficios de dicha actividad, las exigencias, presiones y agotamiento en los deportistas de alto rendimiento, han provocado que con los años se hayan vuelto más susceptibles a tener trastornos de sueño. Dado que un atleta profesional debe de optimizar al máximo sus horas de sueño para tener el mejor rendimiento y recuperación posible, con este trabajo se busca crear el diseño de un sistema capaz de monitorear la actividad cerebral de un atleta mientras duerme y tratar de controlar dicho sueño mediante audios especiales conocidos como pulsos binaurales.

Este documento tiene el objetivo de presentar resultados que sirvan como base y fundamento para la continuación y desarrollo de nuevas fases de este proyecto. También se espera que este documento pueda contribuir a la comunidad científica en temas como el comportamiento de las ondas cerebrales y su asociación a las etapas del ciclo del sueño y principalmente en la incidencia que pueden tener los pulsos binaurales en la actividad cerebral de una persona. Cabe mencionar que este trabajo de investigación se hizo por medio de los conocimientos adquiridos durante cinco años de la carrera de Ingeniería Electrónica, con bastas consultas a diversos medios sobre los temas relacionados y con la asistencia y apoyo de maestros y especialistas en el área.

Se espera que con esta investigación todo lector interesado en el tópico pueda comprender a detalle todos los pasos realizados para la elaboración de esta primera fase del sistema. Además, se espera que con el aporte de este trabajo de graduación, en un futuro nuevos estudiantes de la Universidad del Valle quieran seguir mejorando y avanzando este proyecto. Esto con la finalidad de poder brindarle a la comunidad de Guatemala una solución a un problema con el que muchas personas, no solo atletas, se sienten identificadas y afecta su vida diaria.

CAPÍTULO 2

Antecedentes

El estudio de las señales eléctricas del cuerpo humano para entender mejor su comportamiento es un campo de estudio que ha tenido un crecimiento importante en los últimos años. Empresas "*open source*" como OpenBCI han contribuido significativamente en los avances y descubrimientos en esta área de la ciencia. Este tipo de empresa proporciona todo el equipo, software y guías necesarias de una manera accesible para que todo aquel interesado en estas señales eléctricas pueda hacer sus propias experimentaciones para compartir sus hallazgos con la comunidad científica. Hoy en día hay una gran cantidad de información de investigaciones que han utilizado el equipo y software de OpenBCI para llevar a cabo todo tipo de proyectos como lo que se realizó en [1] o en [2].

En el 2018 en la Universidad del Valle de Guatemala se trabajaron dos proyectos, [3] y [4], que utilizaron el equipo de la empresa OpenBCI para realizar lecturas cerebrales por medio de la Cyton Board y de cascos con electrodos impresos en 3D. En ambos trabajos lo que se buscó llevar a cabo fueron terapias de neuroretroalimentación o *neurofeedback*. Dichas terapias se basan en realizar sesiones monitorizadas de la actividad cerebral de una persona mientras realiza determinadas tareas y según la respuesta del cerebro se le da a la persona un estímulo positiva o negativa dependiendo de la reacción. La idea de todo este proceso es entrenar el cerebro a controlar ciertos procesos cerebrales para que potencialmente una persona sea capaz manejar mejor escenarios donde necesite mayor concentración o reducir el estrés.

También existen investigaciones donde se estudia el impacto que tienen los pulsos binaurales en las ondas cerebrales como en el caso de [5] o bien [6]. Dentro de toda esta temática existe cierta incertidumbre sobre cual es el alcance real que tienen estos pulsos para sincronizar las ondas cerebrales o inducir ciertos estados mentales. Sin embargo, en los trabajos mencionados anteriormente los resultados son positivos e invitan a realizar más investigación en el tema.

Finalmente, la clasificación de las etapas del sueño en base a señales del cuerpo humano también es un fenómeno que ha sido ampliamente investigado. En la mayoría de estas investigaciones se presentan resultados optimistas que permiten llegar a pensar que sí es posible

desarrollar un sistema capaz de realizar una identificación de las etapas del sueño sin la necesidad de la intervención de un experto. En [7] se hace una recopilación de múltiples trabajos que han tratado de hacer la clasificación mediante la extracción de diferentes características y el uso de distintos algoritmos de reconocimiento de patrones.

CAPÍTULO 3

Justificación

A pesar de que una persona común y corriente duerme alrededor de una tercera parte de su vida, la investigación acerca de los efectos negativos que puede tener la privación del sueño es escasa. Dormir es un proceso fundamental para que tanto cuerpo y mente se restauren del día a día, especialmente para los atletas, para quiénes la recuperación es esencial para mantenerse en forma y poder exigirse al máximo. Al contrario de lo que naturalmente se podría llegar a pensar, los atletas tienen una mayor tasa de desórdenes de sueño que una persona regular, en su mayoría desórdenes de conciliación o mantenimiento, es decir, su sueño es menos eficiente que el resto personas y esto puede tener consecuencias que afecten su salud [8]. Según un estudio realizado por la Universidad de Finlandia Oriental y la Clínica de Sueño Oivauni, un 22 % de los atletas estudiados reportaron tener problemas para dormir mientras se encontraban fuera de temporada. Además, estando fuera de temporada un 4 % de ellos también comentaron tomar algún medicamento para poder dormir mejor. Estos números aumentan significativamente a un 46 % y 17 % respectivamente cuando los atletas están en plena temporada de competición [9].

El análisis del sueño no es una prioridad para los preparadores físicos de los atletas, principalmente por la poca accesibilidad que se tiene este tipo de estudio, al alto costo del procedimiento y al tiempo que requiere invertirle. Sin embargo, si se dispusiera de un sistema cómodo, no invasivo y automático que pudiera ayudar a mejorar la calidad del sueño de manera instantánea, se podría llegar a tener un diferenciador importante para los atletas en términos de rendimiento, salud y recuperación a nivel competitivo. Además, con la capacidad de poder monitorear los sueños de una forma más sencilla, se puede extraer información para los perfiles psicológicos deseados por el Comité Olímpico Guatemalteco (COG). Cabe mencionar que un futuro los resultados de este proyecto también pueden llegar a contribuir a la poca investigación existente sobre cómo la calidad del sueño puede afectar directamente el estado mental y rendimiento de los atletas.

Adicionalmente, en la comunidad científica existe cierta incertidumbre acerca de qué tan grande es el impacto que existe en las ondas cerebrales al estimular el cerebro con audios especiales como los pulsos binaurales, pulsos monoaurales o tonos isocrónicos. Estos tres

tipos de audios se creen que son capaces de crear un seguimiento, arrastre o sincronización de la frecuencia de las ondas cerebrales con la frecuencia de los audios, sin embargo, con los datos existentes no hay nada concluyente sobre sus efectos. Por ejemplo, a pesar de que en [10] determinan que los resultados del estudio realizado no muestran ningún efecto de seguimiento significativo de las ondas cerebrales a los pulsos, sí encontraron que una estimulación de pulsos binaurales puede ayudar a una persona a llegar a un estado de relajación más profundo.

Por otro lado, en [6] concluyen que la estimulación de pulsos binaurales es un método valioso, accesible y no invasivo para mejorar tanto el sueño de los atletas como su estado mental al momento de despertarse. Además, también concluyen que el mejoramiento del sueño por medio de la estimulación de pulsos binaurales puede ser un factor que pueda influenciar positivamente el rendimiento psicofísico de un atleta. Cabe mencionar que la principal diferencia entre los estudios mencionados es que en [10] hacen pruebas con la estimulación de pulsos durante periodos de 5 minutos, mientras que en [6] aplican los pulsos de manera constante durante toda la noche. Dado el interés de la comunidad científica en continuar explorando los potenciales beneficios de los pulsos binaurales, este proyecto será un valioso aporte para todos los interesados en su estudio.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Diseñar un sistema que reproduzca tonos estéreo que generen pulsos binaurales en el cerebro usando lecturas de un electroencefalograma para realizar pruebas de monitoreo e inducción del sueño.

4.2. Objetivos específicos

- Realizar el acoplamiento y la conexión del gorro Electro-Cap con el controlador Cyton Board para visualizar la actividad cerebral empleando la interfaz de lectura de datos de OpenBCI.
- Desarrollar un algoritmo que genere los archivos de audio con tonos estéreo acorde al pulso binaural y duración deseada para reproducirlos acorde a la etapa del ciclo del sueño en que la persona se encuentre.
- Desarrollar un algoritmo que lea los datos en crudo de los ocho canales de la Cyton Board para realizar el procesamiento de señales necesario para la obtención correcta de las ondas cerebrales.
- Entrenar algoritmos básicos de aprendizaje automático (*Machine Learning*) para reconocer distintos patrones en las ondas cerebrales que permitan identificar las etapas del ciclo de sueño de una persona.
- Realizar pruebas de funcionamiento con los electroencefalogramas de distintas personas para evaluar si la identificación de las ondas cerebrales y reproducción de audios es correcta acorde a la etapa del sueño en la que el individuo se encuentre.

CAPÍTULO 5

Alcance

Este proyecto consiste en el desarrollo de una primera fase de lo que se espera termine siendo un sistema final que les permita a los atletas combatir los desórdenes de sueño mediante la aplicación inteligente de pulsos binaurales. Esta primera fase se lleva a cabo a través del desarrollo de tres módulos principales, la electroencefalografía (EEG), la generación de los pulsos binaurales y el reconocimiento de patrones en las ondas cerebrales. Cabe mencionar que este proyecto no es una implementación de un sistema final que ya pueda ser aplicado en pacientes. Sin embargo, presenta los resultados necesarios para apoyar y promover futuras fases del mismo.

El primer módulo consiste en utilizar todo el equipo y software necesario para poder realizar una EEG, y así poder monitorear y registrar la actividad cerebral de una persona. Dentro de este mismo módulo también se incluye el proceso de hacer los códigos pertinentes y utilizar las herramientas necesarias, para adquirir los datos numéricos de las ondas cerebrales y procesarlos mediante un Procesamiento de Señales Digital (DSP). Por otro lado, el segundo módulo se basa en el desarrollo de un algoritmo capaz de generar y reproducir los archivos de audio que originan los pulsos binaurales en un individuo. Este algoritmo recibe como parámetros la frecuencia central de los audios, la frecuencia de los pulsos binaurales, la frecuencia de muestreo del archivo y la duración deseada. Por último, en el tercer módulo se toman señales grabadas durante un estudio de sueño y se les extraen características que permiten distinguir una etapa del sueño de otra. Por medio de estas características se entrenan algoritmos de reconocimiento de patrones para crear modelos capaces de clasificar las señales acorde a las distintas etapas del ciclo del sueño. El producto de esta primera fase es un sistema que acopla todos los módulos mencionados anteriormente en una simulación final, que emula el funcionamiento que tendría que tener un sistema ya completo.

CAPÍTULO 6

Marco teórico

6.1. Ondas cerebrales

La raíz de todos los pensamientos, emociones y comportamientos del ser humano son producto de la interacción entre de neuronas dentro del cerebro. Esta interacción produce una actividad eléctrica conocida como ondas cerebrales, ya que por las características químicas de estas células, al conectarse unas con otras se generan pequeños pulsos eléctricos de forma sincronizada. Las ondas cerebrales son detectadas por medio de sensores llamados electrodos en el cuero cabelludo y son clasificadas principalmente según su velocidad en ciclos por segundo (Hz). Cada onda cerebral puede ser asociada a un estado cerebral distinto, es decir, las ondas cerebrales cambian conforme las actividades que realiza una persona y lo que siente al realizarlas [11].

Por lo general, cuando predominan las frecuencias bajas en las ondas cerebrales, la actividad cerebral se asocia a estados emocionales como cansancio, relajación o sueño. Por otro lado, cuando las ondas tienen una mayor frecuencia las personas normalmente se encuentran activas, concentradas o en un estado de alerta. En el Cuadro 1 se clasifican las distintas ondas cerebrales donde se describen los estados emocionales con los que se les asocia. En la Figura 1 se puede apreciar la forma de estas ondas cuando son estudiadas por un especialista. La clasificación del Cuadro 1 simplemente es un estándar para identificar las ondas cerebrales, sin embargo, en la práctica se observan y analizan aspectos y patrones mucho más complejos que pueden reflejar distintas condiciones dependiendo de la ubicación en la cabeza donde se generaron [11].

Onda	Rango de frecuencias (Hz)	Características principales
Delta	0.5 - 4	Son las ondas de menor frecuencia y son conocidas por estar presentes en la etapa del sueño profundo del ciclo del sueño. Se caracterizan por ser ondas de alta amplitud y por la regeneración y sanación que ocurre en el cuerpo cuando estas ondas están presentes.
Theta	4 - 7	Estas ondas son producidas durante estados de meditación profunda y durante periodos de concentración intelectual. Las ondas Theta se asocian a un estado emocional cuyas características principales son la memoria plástica, la estimulación de la imaginación y creatividad y un incremento en la capacidad de aprendizaje.
Alfa	7 - 12	Las ondas alfa aparecen durante estados de relajación y meditación del cuerpo, por lo que se consideran como el estado de descanso del cerebro. La actividad de las ondas alfa aumenta considerablemente simplemente con cerrar los ojos.
Beta	13 - 39	La generación de ondas Beta se presenta cuando la persona se encuentra despierta y en plena actividad mental, es decir, estas ondas son asociadas a un estado normal de una persona cuando esta activa, alerta y tomando decisiones.
Gamma	> 40	Son las ondas más rápidas y de menor amplitud. Pueden aparecer en cualquier parte del cerebro y se asocian a estados donde una persona esta resolviendo problemas lógicos o matemáticos, favoreciendo el procesamiento de la información.

Cuadro 1: Clasificación de las ondas cerebrales [11]

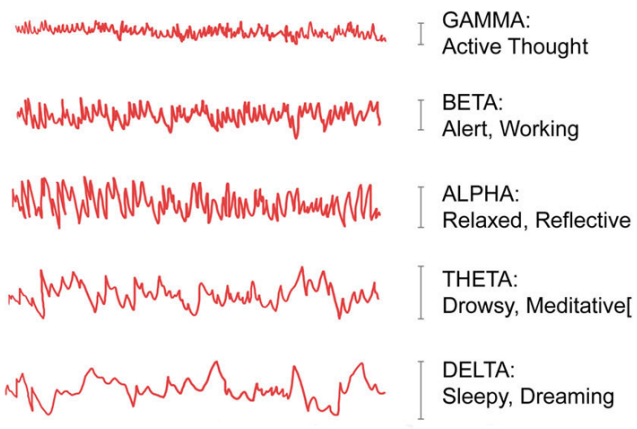


Figura 1: Las ondas cerebrales [12]

6.1.1. El ciclo del sueño

Cuando una persona duerme su inconsciente va progresado a través de distintas etapas del sueño que conforman lo que se conoce como el ciclo del sueño. A las primeras cuatro etapas se les denomina como etapa 1, etapa 2, etapa 3 y etapa 4 y se clasifican como sueño NREM por el acrónimo que se forma con sus siglas en inglés (*Non-Rapid Eye Movement*), ya que durante estas etapas apenas existe movimiento ocular. Por otro lado, la última etapa es conocida y clasificada como la etapa REM también llamada así por sus siglas en inglés (*Rapid Eye Movement*). En esta etapa, al contrario de lo que ocurre en las etapas NREM, los ojos se mueven constante y rápidamente. A lo largo de toda la noche una persona atraviesa todas las etapas de manera cíclica, empezando por la etapa 1 y terminando en la etapa REM. Generalmente completar el ciclo del sueño dura entre 90-110 minutos, por lo que se recorre entre 4 y 6 veces por noche dependiendo de cuanto tiempo se duerma [13].

Durante el transcurso de la noche cada etapa tiene una duración aproximada entre 5-15 minutos dependiendo de cuanto tiempo lleve la persona dormida. Comúnmente, durante los primeros ciclos las etapas de sueño REM son cortas con largos periodos de la etapa 4. Conforme avanza la noche la situación se revierte y el sueño REM aumenta su duración mientras que la etapa 4 aparece por menos tiempo e incluso llega a desaparecer en los últimos tramos de la noche. En la Figura 2 se muestra lo que se conoce como un hipnograma del sueño, el cual es un registro gráfico de las distintas etapas del sueño que se genera según las mediciones de señales bioeléctrica en el cuerpo mientras una persona duerme. Este gráfico es el que les permite a los expertos diagnosticar entre los distintos trastornos de sueño que existen. En la Figura 2 lo que se muestra es un hipnograma estándar del sueño de una persona, en donde se puede apreciar como se va alterando el tiempo que duran las etapas conforme transcurre la noche [14]. A continuación, se describen las distintas etapas con información obtenida de [13].

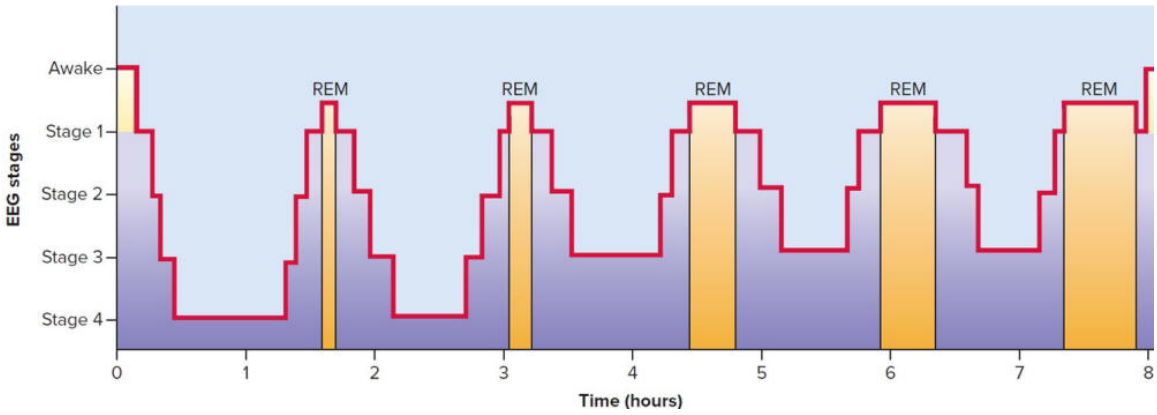


Figura 2: Hipnograma estándar del sueño [15]

- Etapa 1 o somnolencia: Es la primera etapa y la más corta del sueño NREM, en la que apenas hay una pequeña presencia de movimiento de en los ojos. Durante este periodo de tiempo el tono muscular empieza a relajarse y las ondas cerebrales se comienzan a volver más lentas. Durante esta etapa cualquier ruido o distracción leve podría despertar a la persona, y corresponde a un 5 % del ciclo del sueño.

- Etapa 2 o sueño ligero: Esta etapa también pertenece al sueño NREM y básicamente la persona sigue profundizando su sueño. Despertar a una persona durante esta etapa ya no es tan sencillo como en el caso anterior. Esta etapa comprende un 50 % del ciclo del sueño en donde la frecuencia cardíaca, respiración y temperatura del cuerpo comienzan a disminuir.

- Etapa 3-4 o sueño profundo: Dado que son expertos en el sueño los que manualmente anotan y clasifican las etapas, la cantidad de etapas en el ciclo del sueño depende de las guías que se utilicen para identificarlas. Por muchos años solamente existían las guías de clasificación desarrolladas por Rechtschaffen y Kales en 1968, también conocidas como las guías R&K, en donde la etapa del sueño profundo se separa en dos etapas distintas, la etapa 3 y la etapa 4. Con el paso del tiempo surgen nuevas guías de clasificación más modernas como el caso de guías desarrolladas en el 2007 por la Academia Estadounidense de Medicina del Sueño en donde las etapas 3 y 4 se combina en una sola etapa que en conjunto describen el sueño profundo. Estas guías también se les conoce como las guías AASM por las siglas en inglés de *American Academy of Sleep Medicine*. En cualquier caso, este bloque del ciclo es la etapa más profunda del sueño NREM. Durante este periodo de tiempo las ondas cerebrales se vuelven todavía más lentas y el tono muscular esta prácticamente completamente inhibo. Esta etapa es la más reparadora de todas y despertar a una persona con un estímulo externo es mucho más complicado que en las etapas anteriores, en total comprende un 20 % del ciclo del sueño.

- Sueño REM: Al contrario de las etapas anteriores, en esta parte del sueño hay un movimiento rápido de los ojos ya que es la etapa en la que una persona sueña. Debido al sueño de la persona las ondas cerebrales también se vuelven más activas que en las etapas anteriores, además de que la frecuencia cardíaca y respiratoria también aumentan. Sin embargo, hay una atonía muscular completa para que la persona no actúe ni se mueva durante sus sueños. Esta etapa comprende un 25 % del total del ciclo.

6.1.2. Ondas cerebrales durante el ciclo del sueño

El estudio de los sueños es un campo que ha crecido considerablemente durante las últimas décadas debido a que ahora se cuenta con mucho mejor equipo para estudiar a detalle las ondas cerebrales durante el sueño. Cabe mencionar que en un principio se creía que toda actividad cerebral cesaba completamente mientras una persona dormía. Hoy en día se tiene claro que conforme se atraviesa el ciclo del sueño las ondas cerebrales se vuelven más lentas y más ruidosas, es decir, van reduciendo su frecuencia y aumentando su amplitud de voltaje. Esta tendencia continúa hasta llegar al sueño REM donde las ondas vuelven a ser rápidas y pequeñas. En la Figura 3 se puede apreciar la forma que las ondas van tomando conforme se va transicionando a una nueva etapa del ciclo [14].

Además de las ondas cerebrales características mencionadas anteriormente, durante la etapa 2 del ciclo del sueño también es común que aparezcan los husos del sueño y complejos K, las cuales son características únicas de estas etapas. Los husos del sueño son ondas oscilatorias que aparecen de forma repentina y breve y se cree que están relacionados con la transferencia de energía eléctrica en el cerebro. Por otro lado los complejos K, al contrario

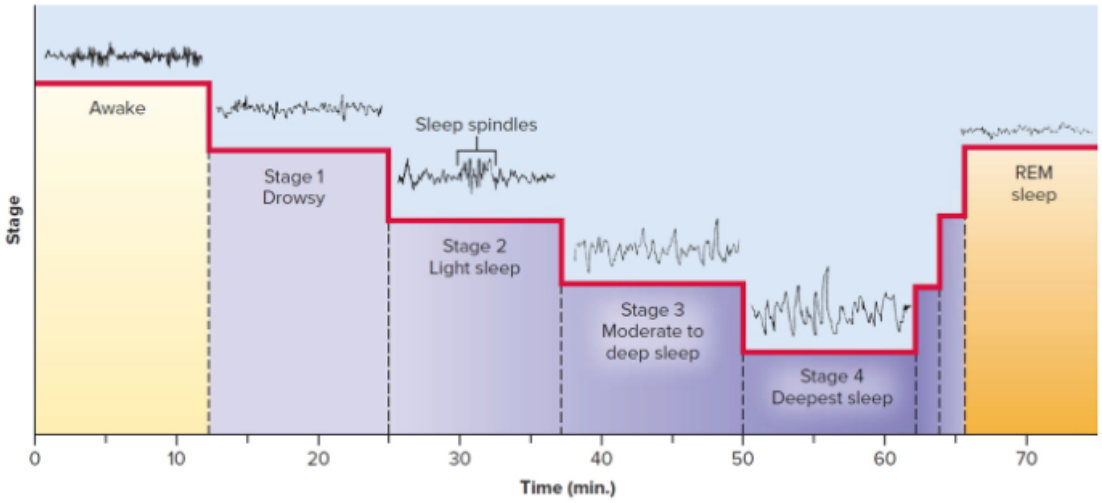


Figura 3: Ondas cerebrales durante las etapas del sueño [15]

de los husos del sueño, son ondas largas y lentas que también pueden aparecen de forma repentina o como reacción a un estímulo externos mientras la persona duerme [14]. Ambos sucesos se pueden comprender mejor con la Figura 4. En el Cuadro 2 se recopila la asociación de las ondas cerebrales con las distintas etapas del ciclo del sueño.

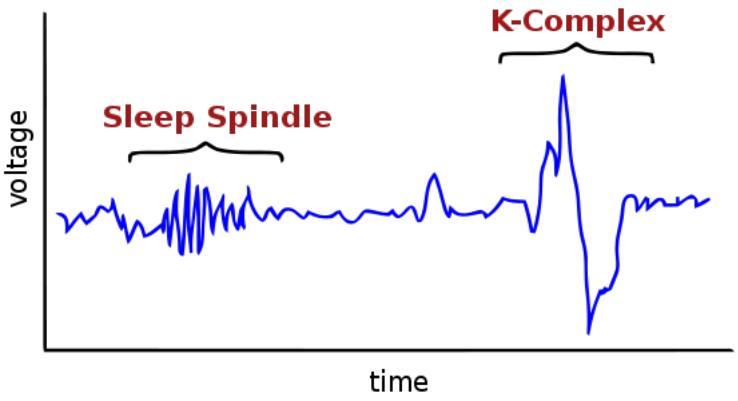


Figura 4: Husos del sueño y complejos-K [16]

Etapa	Frecuencia (Hz)	Amplitud (uV)	Onda cerebral
Despierto	15-50	< 50	Beta
Pre-sueño	8-12	50	Alfa
Etapa 1	4-8	50-100	Theta
Etapa 2	4-15	50-100	Husos de sueño y complejos K
Etapa 3-4	1-4	100-200	Delta
REM	15-30	< 50	-

Cuadro 2: Características de las ondas cerebrales durante el ciclo del sueño [13]

6.1.3. Electroencefalograma

Un electroencefalograma o EEG es un examen médico que se realiza para monitorear la actividad eléctrica en la corteza cerebral con el fin de obtener diagnósticos sobre su funcionamiento, analizar comportamientos de la persona e incluso detectar alguna enfermedad cerebral como epilepsia u otra enfermedad degenerativa. Para poder realizar este tipo de examen es necesario utilizar electrodos que se colocan en la cabeza de la persona para poder obtener las señales eléctricas del cerebro tal como se observa en la Figura 5. Dichas señales son tan pequeñas, en la escala de micro voltios (uV), que además de los electrodos también se requiere de un equipo capaz de detectar, amplificar y procesar las señales para poder observar gráficamente las ondas cerebrales de forma correcta [17].

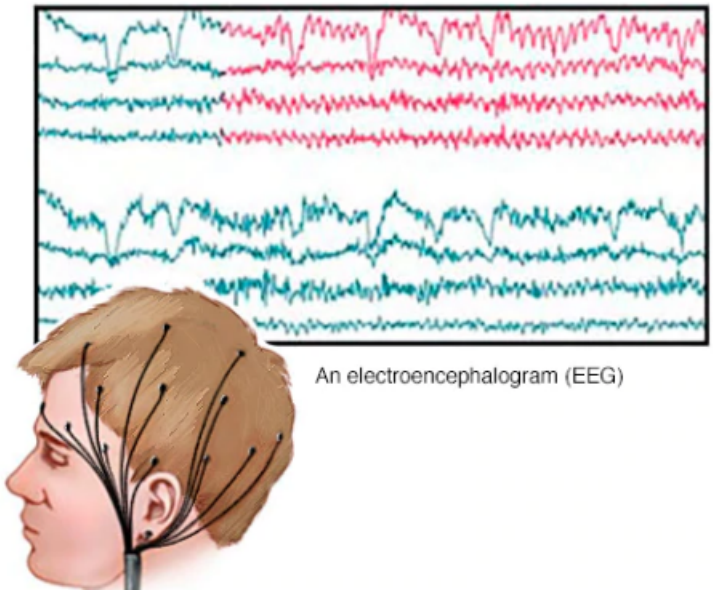


Figura 5: Ejemplo de un electroencefalograma [18]

Una de las formas más comunes para realizar un EEG era utilizar electrodos húmedos. En este escenario los electrodos se encontraban incrustados en un gorro especial y por medio de un gel se realizaba la conexión con el cuero cabelludo de la persona. Cuando se realiza este tipo de estudio con este tipo de electrodo se recomienda que la persona sujeta al análisis llegue con el cabello apropiadamente lavado y sin aplicarse ningún tipo de producto o cosmético. También se pueden realizar EEG con electrodos secos cuya principal ventaja es que pueden hacer las lecturas directamente, sin necesidad de gel o pasta, pero pueden resultar más incómodos para el paciente para estudios de larga duración [18].

La electroencefalografía más común se realiza en una clínica especializada y dura aproximadamente 30 minutos, para este caso no se requiere ningún tipo de preparación previa por parte del paciente. Durante el estudio se le pide al paciente realizar distintas actividades como respirar profundamente, abrir y cerrar los ojos y ver unas luces intermitentes para analizar la respuesta del cerebro.

Con el paso de los años, realizar este tipo de estudio se ha vuelto más accesible y el interés en experimentar con las ondas cerebrales ha incrementado. Hoy en día las aplicaciones de un electroencefalograma pueden abarcar desde terapias de neuroretroalimentación hasta estudios de sueño. Cabe mencionar que los estudios de sueño en los que además de utilizar la EEG, también se monitorean otras señales del cuerpo por medio de electrocardiogramas (ECG), electrocolugramas (EOG) y electromiogramas (EMG), se les conoce como polisomnografías (PSG) [17].

6.1.4. Sistema Internacional 10-20

El Sistema Internacional 10-20 es un estándar internacionalmente reconocido para la colocación y descripción de electrodos en el cuero cabelludo para un estudio de electroencefalografía. Este sistema se basa en distancias porcentuales de 10 y 20 porciento entre el nasión e inión y otros puntos específicos y reconocibles en el cráneo de una persona. Los puntos fijos o de referencia son el lóbulo frontal denominado con la letra (F), el lóbulo central (C), el parietal (P), el temporal (T), el occipital (O) y a las referencias se les asigna la letra (A) [19].

Los electrodos de referencia por lo general se colocan en los lóbulos de orejas, ya que por su cercanía al cráneo y poco tejidos resultan ser un buen punto neutro para medir la diferencia de potencial de estos puntos respecto al resto de electrodos. Todos los electrodos que se encuentran dentro de la línea central del cerebro utilizan el subíndice Z. Por otro lado, los electrodos que se encuentran en el hemisferio izquierdo del cerebro utilizan subíndices de números impares, mientras que los del hemisferio derecho utilizan subíndices pares [19]. La ubicación de los electrodos con el sistema 10-20 se puede apreciar y comprender claramente a partir de la Figura 6.

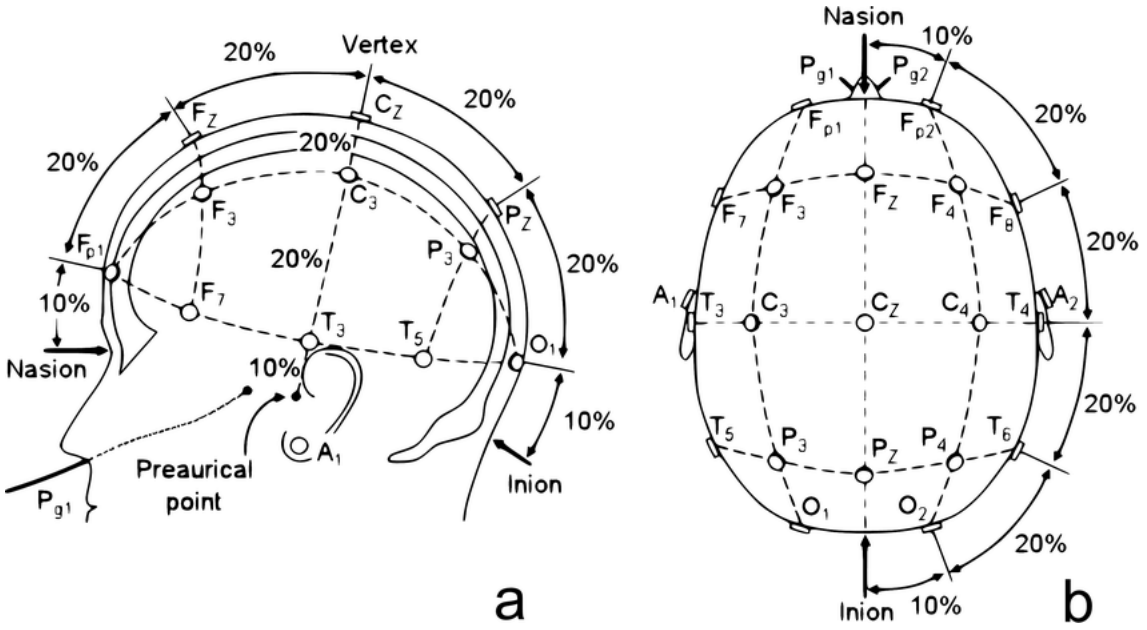


Figura 6: Ubicación de electrodos según el Sistema Internacional 10-20 [20]

6.1.5. Electro-Cap

El Electro-Cap es un gorro que tiene adheridos electrodos en la tela elástica que lo forma y es muy utilizando en aplicaciones de EEG. Los electrodos están ubicados en la tela de tal forma que al colocarlo en la cabeza de una persona quedan posicionados cumpliendo con los estándares del sistema 10-20. Estos gorros se vuelven especialmente útiles cuando se necesita un estudio detallado que requiera la lectura de ondas cerebrales desde distintos puntos alrededor del cerebro, ya que pueden contar hasta con 19 electrodos. Tienen la gran ventaja de evitar el proceso de medir y calcular la ubicación y colocación de cada electrodo según el sistema 10-20, lo cual no ocurre con otras alternativas que miden EEG [21].

El Electro-Cap cuenta con electrodos húmedos de estaño puro. Para hacer una conexión entre el electrodo y el cuero cabelludo del paciente es necesario aplicar un gel conductor en los electrodos que se quiera utilizar, llamado Electro-Gel, por medio de una jeringa con punta roma. Además del gel , el gorro y la jeringa, el sistema completo del Electro-Cap también incluye los electrodos de referencia que deben colocarse de preferencia uno en cada oreja del sujeto en el que se realizará el estudio. En conjunto, todos los componentes del sistema permiten facilitar la recepción de la señal al maximizar el contacto del metal del electrodo con el cuero cabelludo y permitiendo una baja resistencia en las lecturas a través de la piel [21]. El Electro-Cap junto a sus demás complementos se pueden observar en la Figura 7.

Figura 7: Sistema completo del Electro-Cap [21]

El Electro-Cap posee una terminal DB-25 donde se puede acceder a los distintos electrodos que posee el gorro. En algunos casos los electrodos de referencia ya están incorporados dentro del gorro y en otras versiones es necesario conseguirlos por separado [21]. En el Cuadro 3 se detallan a que electrodos están conectados pines de salida de DB-25.

Ubicación	Color	Pin
FP1	Café	1
F3	Rojo	2
C3	Naranja	3
P3	Amarillo	4
O1	Verde	5
F7	Azul	6
T3	Morado	7
T5	Gris	8
GND	Blanco	9
FZ	Negro	10
A1	-	11
OZ	-	12
-	-	13
FP2	Café	14
F4	Rojo	15
C4	Naranja	16
P4	Amarillo	17
O2	Verde	18
F8	Azul	19
T4	Morado	20
T6	Gris	21
CZ	Blanco	22
PZ	Negro	23
A2	-	24

Cuadro 3: Pines de salida del Electro-Cap [21]

6.1.6. OpenBCI

OpenBCI se describe como una comunidad de código abierto, también conocido como *open source*, de investigadores, ingenieros, científicos, etc. que tienen como objetivo que todos aquellos que tengan pasión e interés en las señales eléctricas del cuerpo y cerebro humano tengan a su alcance las herramientas necesarias para descubrir nuevas cosas acerca de su comportamiento. El logo de esta comunidad se encuentra en la Figura 8. Esta comunidad provee de sistemas que pueden ser usados para monitorear la actividad cerebral (EEG), la actividad muscular (EMG) y la actividad cardiaca (ECG). Además de estos sistemas, esta comunidad también proveen de tutoriales y software para poder utilizar sus equipos correctamente y tener una interfaz amigable donde se puedan interactuar con las mediciones [22].

Figura 8: Logo de OpenBCI [22]

Específicamente para la parte de la EEG tienen a disposición cascos y gorros con electrodos para monitorear las ondas cerebrales. Incluso tienen disponibles los diseños de sus cascos y placas para que una persona los pueda replicar por su cuenta como una opción más económica. Por otro lado, también cuentan con una interfaz de usuario de llamada OpenBCI_GUI, la cual está disponible para Windows, MAC y Linux. Esta interfaz permite entre otras cosas, observar gráficamente y en tiempo real las lecturas de hasta 8 canales, observar el espectro en frecuencia de las señales, e incluso hasta desplegar un mapa que indica que partes del cerebro están más activas, todas estas características se ilustran en la Figura 9.

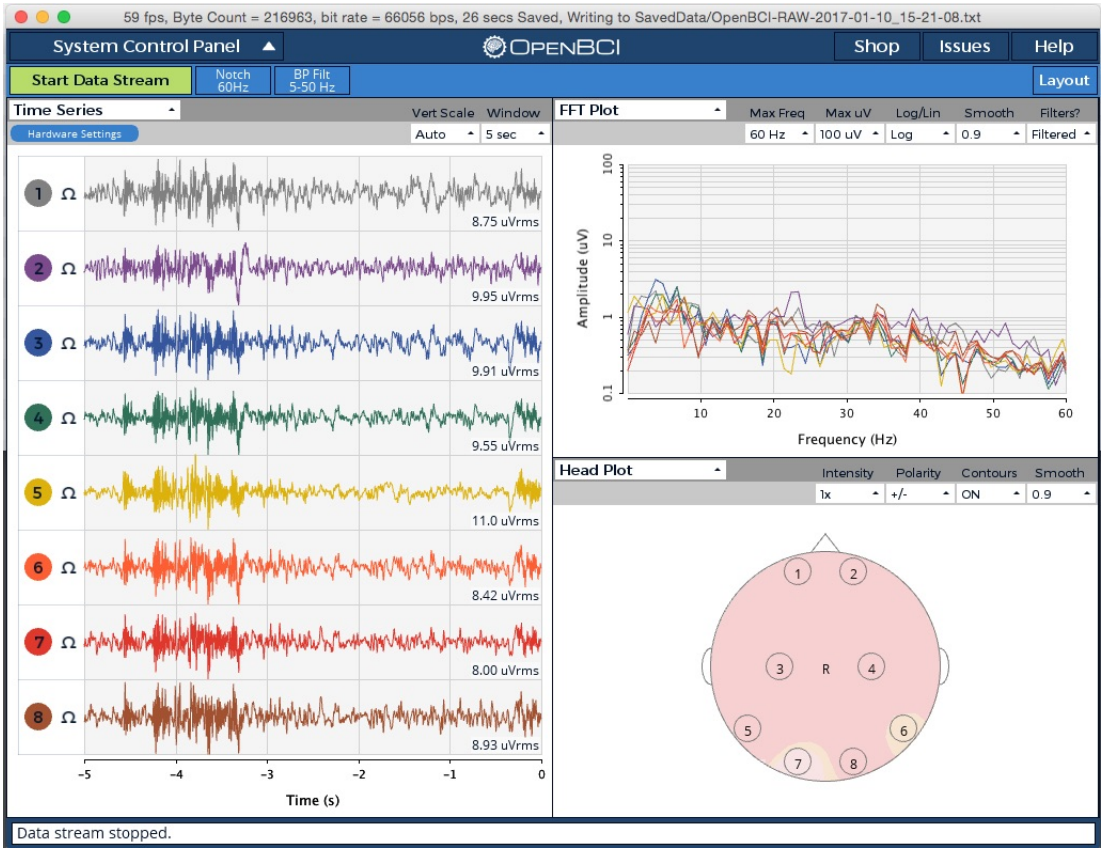


Figura 9: Interfaz de usuario de OpenBCI [22]

La interfaz tiene muchas otras funcionalidades como modificar las escalas de las gráficas, el filtrado de los datos o la ganancia de los canales. También es tan flexible con el usuario que permite modificar el diseño de la interfaz pudiendo mostrar más o menos ventanas con todas las herramientas adicionales que poseen. Por ejemplo, en lugar de utilizar el diseño por defecto de la Figura 8, se podría modificar la distribución de ventanas para también desplegar la gráfica del acelerómetro, o cualquier otra de las herramientas adicionales. Todas estas facilidades las proporcionan con el fin de que el usuario pueda interactuar sencillamente con diferentes escenarios según requiera su estudio [22].

Esta empresa incluso ofrece el código de la interfaz en Processing, lenguaje de programación basado en Java que se enfoca en el diseño digital, para que cualquier persona pueda realizar modificaciones en el código según considere necesario. Además, invitan a las perso-

nas a colaborar con el diseño de más herramientas que puedan ser de utilizad en la interfaz. Este tipo de acceso lo proveen con el fin de que entre la comunidad científica de todas partes del mundo se compartan conocimientos para tratar de aportar soluciones a las mayores incógnitas y problemas que rodean al ser humano [22].

6.1.7. Cyton Board y Dongle

La Cyton Board es uno de los hardware certificados de OpenBCI de bajo costo y alta calidad que cuenta con 8 canales (N1-N8) en los que se pueden muestrear señales biológicas a señales digitales. Adicionalmente, cuenta con un pin de referencia (SRB) y otro para la reducción de ruido (BIAS). La placa también cuenta con un acelerómetro interno del cual se pueden realizar lecturas sin necesidad de tener que conectar nada a los pines [22]. La tasa de muestro por defecto de esta placa es de 250 Hz y aunque el ADS1299 integrado esta equipo para soportar tasas mayores, la conexión inalámbrica con el Dongle y la conexión serial ya no podría soportar estas tasas mayores. La Cyton Board cuenta tiene los siguientes aspectos técnicos:

- Alimentación de 3-6V (Con baterías únicamente)
- Microcontrolador PIC32MX250F128B
- Convertidor analógico ADS1299
- Módulo RFduino BLE para comunicación Bluetooth
- Espacio para tarjeta de almacenamiento Micro SD
- Reguladores de voltaje de 3V,-2.5V y +2.5V

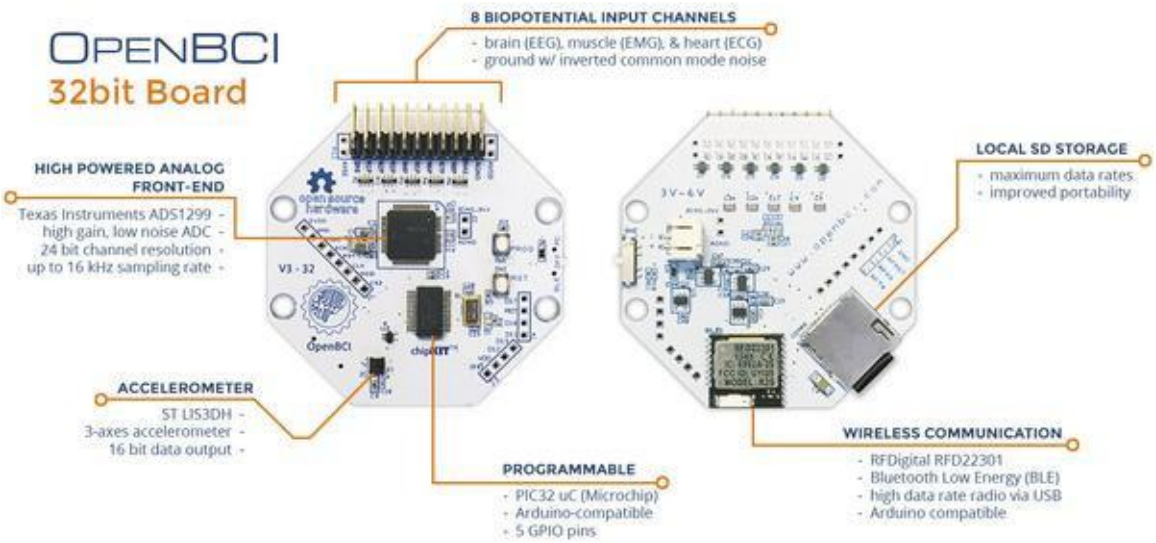


Figura 10: Características de la Cyton Board [22]

En la Figura 10 se pueden observar ambas caras de la Cyton Board junto a sus componentes principales. Para poder realizar la comunicación entre la placa y la computadora se utiliza un dispositivo intermediario conocido como Dongle, dicho dispositivo se puede observar a detalle en la Figura 11. Este módulo encarga de recibir los datos de la Cyton Board por medio de una conexión inalámbrica con Bluetooth, por lo cual ambos dispositivos cuentan con un módulo de Arduino llamado RFduino BLE que permite establecer este tipo de comunicación . Con el RFduino BLE se pueden conseguir mayores tasas de transmisión que con los módulos convencionales. Posteriormente el Dongle de OpenBCI transmite los datos recibidos por el puerto serial de la computadora o procesador con un baudaje de 115200 [22]. Los aspectos técnicos de este dispositivo son:

- Alimentación vía USB (Únicamente)
- Módulo RFduino BLE para comunicación Bluetooth
- FTDI FT231XQ-R para comunicación Serial

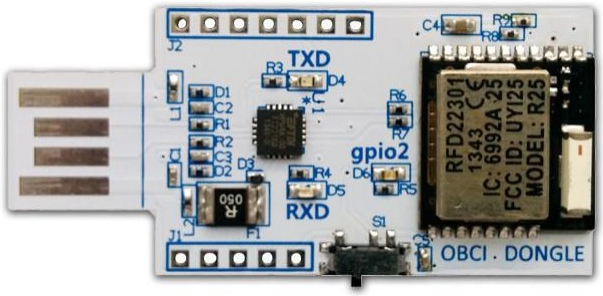


Figura 11: Dongle de OpenBCI [22]

Los tramas de datos que se envían de la Cyton al Dongle y la computadora se componen de 33 bytes. En la Figura 12 se puede ver a detalle la composición de la trama. Dado que los 8 canales del ADS1299 de la Cyton componene 24 bytes de la trama total, se cuentan con 3 bytes o 24 bits de información por cada canal. Los datos de los canales son enteros con signo y deben ser multiplicados en software por un factor de escalamiento para convertir los datos de la trama a valores de cuentas por segundo a unidades científicas como los microvoltios (uV). El cálculo del factor de escalamiento depende de la ganancia que se le den a los canales, tal como se puede ver en la ecuación posterior. La ganancia por defecto de la Cyton Board es de x24 [22].

$$\text{Factor de escalamiento} = \frac{4.5V * Ganancia}{2^{23} - 1}$$

Luego de esta multiplicación ya se obtienen lo que se conoce como los datos en crudo, los cuales son datos sin ningún tipo de procesamiento. Al hacer una sesión de estudio con la interfaz de OpenBCI todos los datos en crudo de la sesión quedan registrados en un archivo de texto. Los datos que se grafican en dicha interfaz sí están procesados y filtrados, sin embargo, se registran los valores en crudo para que un investigador pueda realizar su propio procesamiento de datos según las necesidades de su estudio [22].

Stop Byte	Accelero-meter	8-Channel ADS	Sample Counter	Header
1 byte	6 bytes	24 bytes	1 byte	1 byte

Figura 12: Trama de datos Cyton Board

6.2. Procesamiento de señales

El procesamiento digital de señales (PDS) es un área de la ingeniería que se encarga de procesar, analizar, manipular, amplificar e interpretar señales discretas a través de un conjunto de operaciones y transformaciones matemáticas. Dichas operaciones y transformaciones se realizan con el fin de poder tener una mejor aproximación de la información a analizar. El procesamiento digital surge principalmente debido a la dificultad de manipular señales analógicas con un procesamiento analógico ya que se deben de utilizar componentes como capacitores, resistencias e inductores cuyos valores no son exactos debido a las tolerancias inherentes que poseen. Además, el procesamiento analógico está expuesto a que vibraciones mecánicas o cambios de temperatura afecten el desempeño de dichos componentes [23].

Debido a estas deficiencias, a la hora de procesar una señal se prefiere la exactitud, bajo costo y baja susceptibilidad al ruido que brinda un procesamiento digital. El esquema más simple y común en el que se utiliza un PDS se ejemplifica en la Figura 13. En dicho sistema se muestrea una señal analógica o continua a una señal digital por medio de un Convertidor Analógico/Digital (ADC). Posteriormente ya con una señal digital se puede realizar todo el procesado necesario, como amplificar o eliminar el ruido de la señal, en un dispositivo de cómputo. Finalmente, el resultado del procesamiento se reconvierte a una nueva señal analógica por medio de un Convertidor Digital/Analógico (DAC) para que ya pueda ser usada en alguna aplicación según sea el caso. La idea detrás de la del diagrama de la Figura 13 se puede ver implementada en la industria desde las redes de telecomunicaciones hasta aplicaciones específicas con microcontroladores [23].

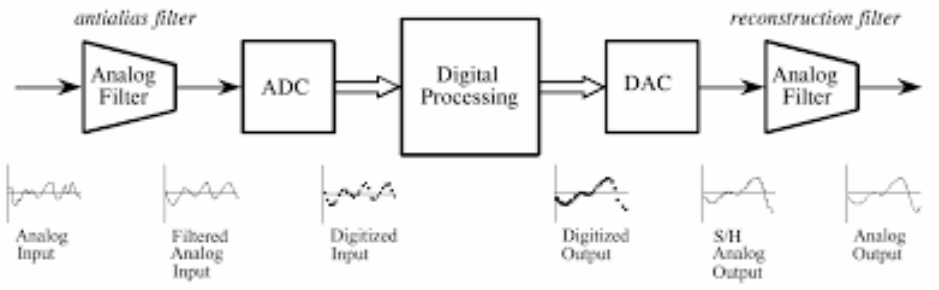


Figura 13: Diagrama de bloques de un sistema DSP [24]

6.2.1. Filtros eléctricos

Un filtro eléctrico es considerado como cualquier medio que atraviesa y modifica una señal. La idea principal detrás de los filtros es que al recibir una señal de entrada solamente

permitan el paso de cierto rango deseado de frecuencias, atenuando todas las frecuencias que no estén en dicho rango. Debido a esta característica, a los filtros eléctricos también se les conoce como circuitos de frecuencia selectiva. Generalmente el rango de frecuencias que se desea que pase libremente de la entrada a la salida, sin atenuación, se le conoce como banda de paso mientras que el conjunto de frecuencias que se desea atenuar y que no lleguen a la salida se les conoce como banda de rechazo. Hoy en día casi cualquier dispositivo que se comunique por medio de señales eléctricas, como televisores, satélites o teléfonos, emplea filtros de frecuencia selectiva para disminuir el efecto de frecuencias fuera del rango interés [25]. Dentro de las aplicaciones más comunes de un filtro se encuentran los dos casos siguientes:

- Separación de señales: En algunos casos hay señales que se combinan, por ejemplo una señal de entrada puede adquirir ruido o interferencias provenientes del medio o algún factor externo. En este caso los filtros se diseñan para que en la banda de rechazo se atenúen las frecuencias del ruido o interferencia y en la salida se obtenga nuevamente la señal original [26].

- Recuperación de señales: En otros casos una señal de entrada puede ser distorsionada, que a diferencia del caso anterior, una distorsión solo afecta a la señal en determinadas porciones y no modifican todos los componentes de la señal. Por ejemplo, el canal de transmisión puede alterar la señal solo en determinado conjunto frecuencias y es necesario aplicar filtros para recuperar la señal que se quería transmitir [26].

Cabe mencionar que en ningún caso los filtros pueden comportarse de manera ideal, es decir, ningún diseño es capaz de eliminar o filtrar perfectamente las frecuencias deseadas, sino que solamente se atenúan. Este concepto se ilustra perfectamente en la Figura 14 en donde la gráfica de la izquierda representaría la respuesta ideal de un filtro pasa altas, dejando pasar todas las frecuencias superiores a una frecuencia determinada y eliminando el resto. Por otro lado en la gráfica en la derecha se observan las características típicas del comportamiento de un filtro real, en el que además de las bandas ya mencionadas también existe una banda de transición donde comienza a atenuarse o disminuirse el efecto de ciertas frecuencias a partir de una frecuencia determinada. Todo diseño de filtros gira entorno a esta frecuencia determinada, la cual es denominada la frecuencia de corte (fc) y se caracteriza porque en esa frecuencia la amplitud de la señal se atenúa a un 70.7 % o lo que es equivalente a una pérdida de -3dB de la amplitud de voltaje máximo [25].

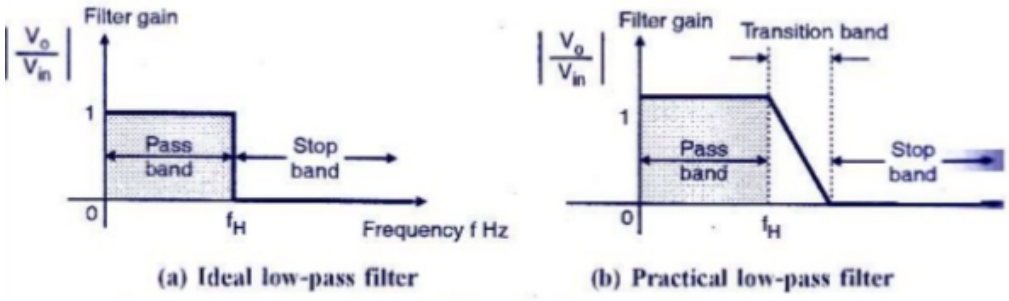


Figura 14: Respuesta en frecuencia de un filtro ideal versus un filtro en la práctica [27]

Los filtros o circuitos selectores de frecuencia se clasifican según la ubicación de su banda de pasa y existen cuatro tipos principales: filtros pasa bajas, pasa altas, pasa banda y rechaza banda. Como su nombre lo indica los filtros pasa bajas tienen una banda de paso para todas los voltajes con frecuencias inferiores a su frecuencia de corte, mientras que todos los voltajes de entrada fuera de esta banda de paso, es decir que tengan frecuencias mayores a la frecuencia de corte, tienen su magnitud atenuada por el filtro. En el caso de los filtros pasa altas pasa todo lo contrario y las frecuencias menores a la frecuencia de corte son las que se ven atenuadas. Tanto los filtros pasa bandas como los rechaza banda pueden ser creados combinando un filtro pasa altas junto a un pasa bajas. Para estos filtros se tienen dos frecuencias de corte, en el caso del pasa bandas se atenúa todo lo que está fuera de estas frecuencias y en el caso del rechaza banda se atenúe todo lo que esta dentro de ellas. En la Figura 15 se puede apreciar la respuesta de los distintos tipos de filtros junto a la ubicación de sus frecuencias de corte [25].

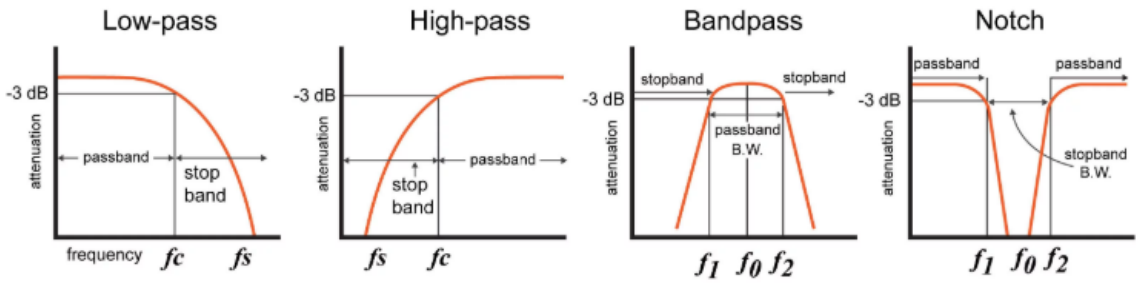


Figura 15: Respuesta de los cuatro tipos de filtros selectores de frecuencia [28]

Los filtros también pueden dividirse en filtros analógicos, los cuales filtran señales continuas mediante componentes eléctricos, y filtros digitales, los cuales trabajan con señales o secuencias discretas en el tiempo y generalmente se implementan por medio algoritmos recursivos en software. Los filtros analógicos pueden dividirse también en filtros pasivos y activos. Por un lado los filtros pasivos como su nombre su indica se construyen exclusivamente con componentes eléctricos pasivos, resistencias, capacitores e inductores, no necesitan de una alimentación externa para funcionar y no son capaces de darle una ganancia adicional a la banda de paso del filtro. Sin embargo por las mismas razones mencionadas en la sección 6.2 estos filtros no son muy viables en aplicaciones donde se requiera mucha precisión. Por otro lado los filtros activos, además de resistencias y capacitores, también utilizan componentes activos como amplificadores operaciones y transistores en circuitos con retroalimentación. Estos componentes además de atenuar la señal en la banda de rechazo también también permiten amplificar la señal de entrada en su banda de paso y son los que más comúnmente se ven en circuitos eléctricos debido a que presentan una mayor facilidad para ajustarse [25].

Los filtros también se pueden dividir según la forma de su respuesta en frecuencia y se debe de seleccionar un tipo dependiendo de que requiera la aplicación, para tomar una decisión se deben de considerar factores, además de la respuesta en frecuencia, como la selectividad, fase y complejidad [29]. A continuación se presentan los cuatro tipo principales, los cuales también se pueden comprender mejor con la Figura 16.

- Bessel: Se caracteriza por tener no tener una banda de transición empinada pero que

tiene la mejor fase, es decir que las frecuencias no presentan ningún retardo en el tiempo o *delay* al pasar por él. Por el otro lado, su complejidad es grande y requiere de muchos componentes para armarlo [29].

- Butterworth: Es el tipo de filtro más utilizado por que tiene la banda de transición más empinada sin presentar ondulaciones en su banda de paso o rechazo. Se puede obtener una banda más empinada dependiendo del orden del filtro. Además, su complejidad es menor y no requiere de tantos componentes como el filtro Bessel [29]. Con este tipo de respuesta sería con lo que mejor se puede aproximar la respuesta ideal de la Figura 14

- Chebyshev: Es conocido por presentar una ondulación en su banda de paso o rechazo, esta ondulación es mayor entre más empinada se quiere que sea la banda de transición [29].

- Elíptico: Para este filtro la ondulación existe tanto para la banda de paso como la de rechazo pero lo compensa teniendo la banda de paso más empinada de todos los filtros [29].

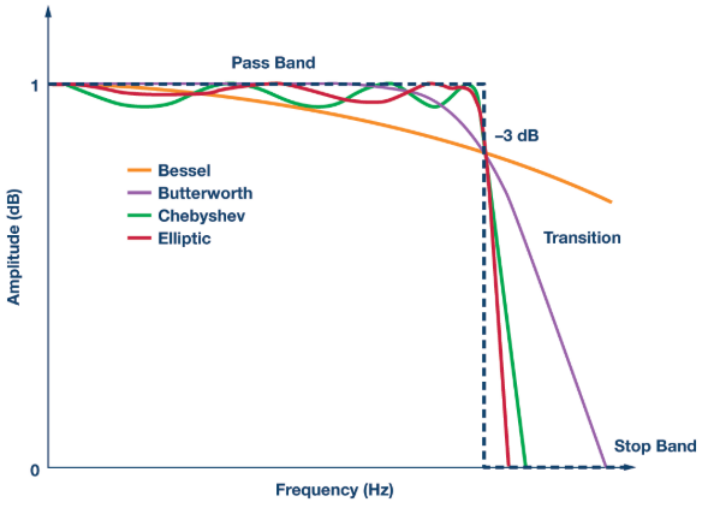


Figura 16: Tipos de respuestas en frecuencia de los filtros [30]

6.2.2. Filtros digitales

Los filtros digitales son filtros que operan sobre señales discretas o digitales y su desempeño es altamente superior al de los filtros analógicos. Para poder implementar este tipo de filtros se requiere de operaciones matemáticas que reciban como entrada una secuencia de números y la modifiquen para dar como salida otra secuencia de números con el objetivo de resaltar o atenuar ciertas características de la señal de entrada. En el diagrama de la Figura 13 este proceso comprendería lo que es bloque de *Digital Processing*. Muchas veces por las múltiples ventajas que puede proporcionar un filtro digital sobre el analógico, el proceso de muestrear una señal se hace solo con el objetivo de aplicar un filtro digital.

Para la implementación de este tipo es necesario encontrar los coeficientes de una ecuación de diferencias, las cuales son el análogo a las ecuaciones diferenciales en tiempo discreto. Sin embargo, este proceso puede resultar muy complicado y exhaustivo de realizar a mano, especialmente para filtros muy específicos o complejos. Por esta razón, cuando se habla de filtros digitales es común que tanto el diseño como la implementación se hagan por medio de software con herramientas como Python o Matlab para hacer que el proceso sea significativamente más simple que en los filtros analógicos [31]. Dentro de los filtros digitales existen dos tipos principales:

- Filtros FIR: Los filtros FIR o filtros de Respuesta al Impulso Finita es un tipo de filtro puramente digital, es decir que no tienen una contraparte en el mundo analógico. Se caracterizan por ser filtros versátiles que son más estables y se adaptan con mayor facilidad a nuevas señales. Su salida en cada instante de tiempo es un promedio de la muestra actual y muestras anteriores de la señal de entrada. Es decir que sus ecuaciones de diferencias no dependen de la salida del filtro, es por eso que también se les llama filtros no recursivos [31].
- Filtros IIR: Los filtros IIR o filtros de Respuesta Infinita al Impulso se comportan de manera más similar a los filtros analógicos. Generalmente se prefieren sobre los filtros FIR cuando se quiere una banda de transición estrecha o empinada sin aumentar significativamente la demanda computacional del filtro. En este caso el filtro sí tiene una parte recursiva en sus ecuaciones por lo que la salida en cada instante de tiempo depende del valor actual de la entrada, valores pasados de la entrada y valores pasados de la señal de salida, razón por la cual también se les llaman filtros recursivos [31].

6.3. Pulsos binaurales

Desde las civilizaciones antiguas se hacían rituales para influenciar el comportamiento de las personas a través de sonidos con ciertos ritmos constantes y repetitivos, dichos rituales se creían que tenían un gran impacto en la sanación de cuerpo y espíritu de las personas. Con el avance de la tecnología, la influencia de la música en el comportamiento cerebral ha podido ser más estudiado y comprendido, llegándose a desarrollar entre muchos otros estudios, la teoría de los pulsos binaurales. Todo estos efectos de los pulsos binaurales se comienzan investigar por el físico Heinrich Wilhelm Dove en 1839 cuando se percata accidentalmente que un sujeto que estaba siendo expuesto a 2 tonos con frecuencias distintas, percibía un pulso con un ritmo más lento.

Sin embargo, fue hasta en 1923 cuando la investigación de los pulsos binaurales empezó a tomar una mayor relevancia. Esto fue gracias al Dr. Gerald Oster quién publicó un artículo llamado *Auditory Beats in the Brain* [32], en donde analizaba el potencial de los pulsos para diagnosticar discapacidades auditivas e incluso otras enfermedades. Por ejemplo, en su artículo presentó evidencias que la poca capacidad de detectar los pulsos binaurales era un indicio para una detección temprana de la enfermedad de Parkison. En este artículo también se centra en la posibilidad que tienen estos pulsos para ayudar a mejorar la concentración, reducir la ansiedad o o eliminar el insomnio basado en que escuchar estos pulsos involucra otro tipo de respuestas neuronales que escuchar un audio convencional [33].

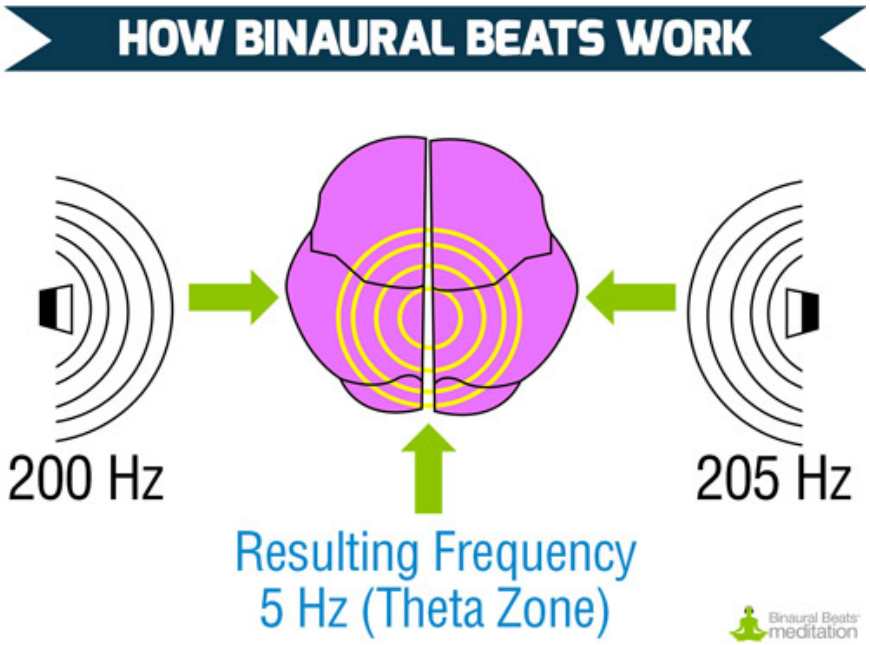


Figura 17: Funcionamiento de los pulsos binaurales [33]

Con muchos años de estudios ya se ha determinado que pulsos binaurales son una ilusión auditiva que se produce en los núcleos olivares del cerebro, los cuales son la parte encargada de controlar la percepción tridimensional del sonido. Esta ilusión se crea dentro del cerebro cuando una persona se ve sometida a dos tonos con frecuencias ligeramente distintas, uno a través del oído izquierdo y otro a través del derecho. Dado que este tipo de sonido no se encuentra normalmente en la naturaleza, para poder procesar dicho audio, el cerebro resta las frecuencias de ambos tonos generando una tercera frecuencia dentro del cerebro, la cual es a la que se le denomina pulso binaural.

En la Figura 17 se ilustra y se comprende mejor este concepto. La teoría de los pulsos binaurales se basa en la idea de que las ondas cerebrales lentamente se sincronizan con la frecuencia de los pulsos. Hoy en día existen muchos estudios en los que se analiza las capacidades de esta teoría tratando de inducir ciertos estados cognitivos según la aplicación de pulsos [33]. En los trabajos [5] y [34] se menciona que para que exista el fenómeno de los pulsos binaurales los tonos deben ser tonos puros, es decir los audios deben ser ondas armónicas que varían el tiempo de forma sinusoidal. Cualquier otro tipo onda, como triangular o cuadrada, tiene armónicos en más frecuencias que harían que el efecto de los pulsos binaurales disminuya o no se produzca.

El efecto solamente se produce cuando se están utilizando audífonos y el sonido es estéreo para que puedan haber una frecuencia distinta en cada canal. Los audífonos son principalmente debido a que el efecto de combinar las frecuencias debe de ocurrir dentro del cerebro y no fuera de el. En caso de reproducir un audio estéreo con frecuencias ligeramente distintas en bocinas u otro medio la combinación de las sonidos ocurre en el aire y se produciría lo que se conoce como un pulso monoaural. Por último, con tonos con una diferencia de frecuencia mayor a los 30 Hz ya no se percibirían los pulsos y los tonos se escucharían como dos audios distintos [33].

6.3.1. Archivos WAV

Los archivos WAV o *Wavefile Audio Format* son un formato de almacenamiento de audio digital, normalmente sin compresión de datos, desarrollado por Microsoft e IBM que utiliza la extensión .wav, su ícono se observa en la Figura 18. Este tipo de archivo puede ser abierto y reproducido por cualquier reproductor de audio y pueden almacenar todo tipo de audios, desde música hasta grabaciones de voz. Debido a que no tienen una compresión de datos estos archivos tienden a ocupar más memoria que otros tipos de almacenamiento como MP3 ó MP4 que sí comprimen los datos. Sin embargo, debido a esta falta de compresión son más accesibles para examinar el contenido del archivo que el resto de formatos.

Este tipo formato permite leer y escribir archivos tanto mono como estéreo de distintas frecuencias y normalmente utilizan frecuencia de muestreo por defecto de 44100 muestras/segundo. Según el teorema de muestreo de Nyquist, con este formato se podrían reconstruir archivos de audio de frecuencias hasta 22050Hz. Bajo esta tasa de muestreo cada muestra tiene un tamaño de 2 bytes/16 bits con signo para los audios mono y 4 bytes/24 bits con signo para el caso de audio estéreo. Este es un tipo de formato flexible que permite escribir archivos con diferentes tasas de muestreo, sin embargo, la mejor calidad posible es la que obtiene a partir de las 44100 muestras/segundo [35].

Figura 18: Ícono de un archivo WAV [36]

6.3.2. Espectro en frecuencias

El espectro en frecuencias de una señal es un gráfico que muestra la descomposición de una señal de entrada eléctrica, acústica u óptica en una escala de amplitudes y fases respecto de la frecuencia. Un analizador de espectro permite observar una señal en el dominio de la frecuencia mediante la implementación de la Transformada de Fourier, la cual es una herramienta que obtiene la intensidad o potencia que presenta cada componente espectral que constituye una señal basada en el tiempo. En decir, con el espectro de frecuencias se obtiene un gráfico de intensidad versus frecuencia de una señal, donde lo que se puede observar es la distribución de la intensidad o amplitud que presenta cada frecuencia de una señal. La contraparte de utilizar un analizador de espectro es ver una señal manifestarse en el tiempo a través de un osciloscopio [37].

El eje vertical se presenta en una escala logarítmica el nivel en dBm (decibelio-milivatio) del contenido espectral de la señal. Mientras que en el eje horizontal se representa la frecuencia en una escala que es función del número de muestras capturadas. En la gráfica superior de la Figura 19 se puede ver una señal de audio en el tiempo, mientras que en la gráfica inferior se ve el espectro frecuencial de esa misma señal. Claramente se puede observar que la frecuencia dominante de la señal de audio se encuentra cerca de los 100 Hz. [37].

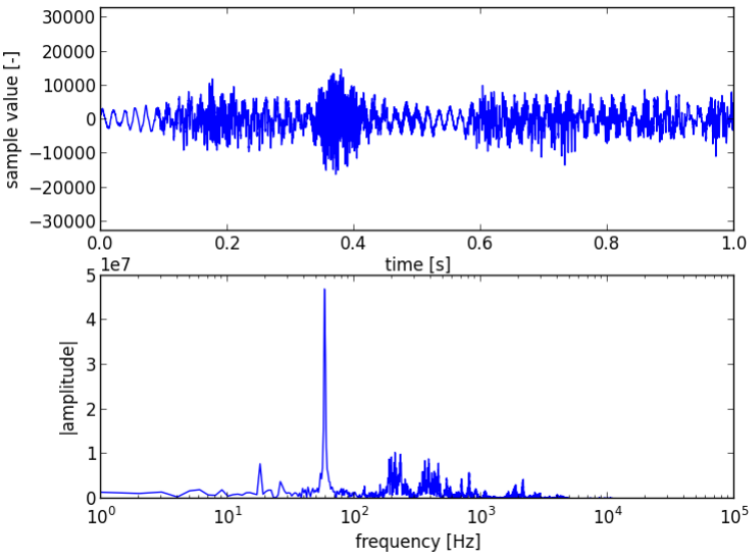


Figura 19: Señal de audio en el dominio del tiempo vrs dominio de la frecuencia [36]

6.4. Aprendizaje automático

El aprendizaje automático, comúnmente conocido como *Machine Learning*, es una rama de la Inteligencia Artificial (IA) que tien e como objetivo desarrollar algoritmos y técnicas que le permitan a una computadora aprender y reconocer comportamientos y patrones a partir de información que se le suministre al equipo. Básicamente con este tipo de IA las computadoras son capaces de detectar patrones en un flujo de datos numéricos y tomar decisiones de forma propia en base al reconocimiento de dichos patrones.

En otras palabras, con el aprendizaje automático las computadoras son capaces de determinar y tomar acciones sin la necesidad de ser programadas explícitamente para tomar dicha acción. Hay distintos tipos de aprendizaje automático dependiendo del tipo de datos y señales que se quieren analizar, incluso, con algunos algoritmos la computadora es capaz de predecir comportamientos futuros de un objeto en base a las propiedades que ha aprendido del objeto en el pasado [38].

6.4.1. Redes Neuronales Artificiales (RNA)

Las Redes Neuronales Artificiales, también llamadas RNA, son una herramienta para resolver problemas de clasificación supervisada. Se les llama redes neuronales debido a que su organización, comportamiento y estructura se basa en las redes neuronales biológicas. El elemento más básico de un sistema neuronal es la neurona y su análogo en una RNA sería una neurona artificial. Tratando de emular un sistema neuronal biológico, las neuronas artificiales se organizan en capas y estas capas se organizan en estructura jerárquica similar a la que se encuentra en el cerebro. En la Figura 20 se puede observar el modelo estándar de una red neuronal y como se conforman por un conjunto de capas [39].

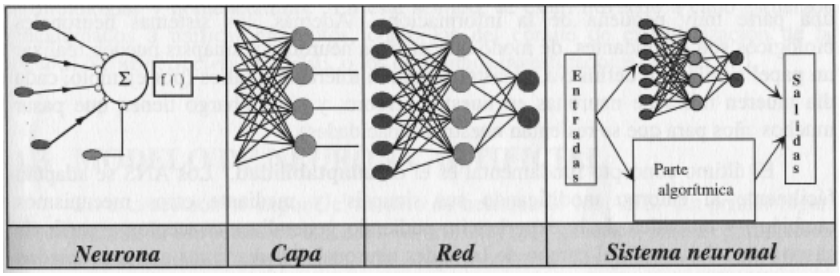


Figura 20: Modelo estándar de una red neuronal artificial [39]

El objetivo de las redes neuronales es emular las siguientes funcionalidades de una red biológica:

- **Procesamiento en paralelo:** Las redes de neuronales son capaces de operar en paralelo un proceso producto de la gran cantidad de neuronas que intervienen en dicho proceso.

- **Memoria distribuida:** En estas redes no hay posiciones de memoria definidas si no que la información se distribuye a lo largo de la red, existiendo redundancia para evitar la pérdida de información.

- **Adaptabilidad:** Las redes de neuronales son capaces de aprender de la experiencia, siendo capaces de modificar su comportamiento en respuesta a su entorno.

Por estas funcionalidades y muchas otras más las redes neuronales son común en el proceso de reconocimiento de patrones. En dicho proceso se entrena una red neuronal para asignar clases determinadas a un conjunto de entradas. Dadas las características de la red, una vez entrenada es capaz de clasificar y aprender de patrones que nunca antes había visto [39].

6.4.2. Máquinas de Vectores de Soporte (SVM)

Las Máquinas de Vectores de Soporte o SVM por las siglas sus en inglés (*Support Vector Machine*) fueron originalmente diseñadas para resolver problemas de clasificación binaria, sin embargo, hoy en día son comúnmente utilizadas para resolver problemas de agrupamiento, regresión y clasificación y han sido utilizadas en numerosas aplicaciones y diversos campos como la visión artificial, reconocimiento de patrones, análisis de series temporales, etc.

Dentro del área de clasificación los SVM buscan encontrar la mejor separación posible entre clases y son considerados como clasificadores lineales dado que inducen separadores lineales, en caso de que la clasificación sea en dos dimensiones, para hacer dicha separación. Aunque realmente lo más común es que los problemas de clasificación con SVM sean multidimensionales, induciendo en lugar de una línea como en el caso de dos dimensiones, un hiperplano en un espacio denominado espacio de características [40].

En cualquier caso, el objetivo del SVM es encontrar la línea o hiperplano óptimo, de entre las infinitas posibles soluciones, que maximice el margen de separación entre clases en un espacio. Para llevar esto a cabo se utilizan los vectores de soporte, los cuales son los puntos que definen el margen máximo de separación que tiene el hiperplano o separador lineal. Se les conoce como vectores porque tienen la misma dimensión del espacio que contiene los puntos que se quieren clasificar. En la Figura 21 se puede observar como se encuentra el hiperplano óptimo que separa ambas clases. También es común que existan escenarios donde pareciera que no es posible encontrar un hiperplano que pueda separar las clases correctamente. Para estos casos se utiliza lo que se conoce como el método Kernel, en el cual básicamente se expande la dimensionalidad del espacio de características para encontrar una separación entre clases [40]. Este escenario se ilustra en la Figura 22.

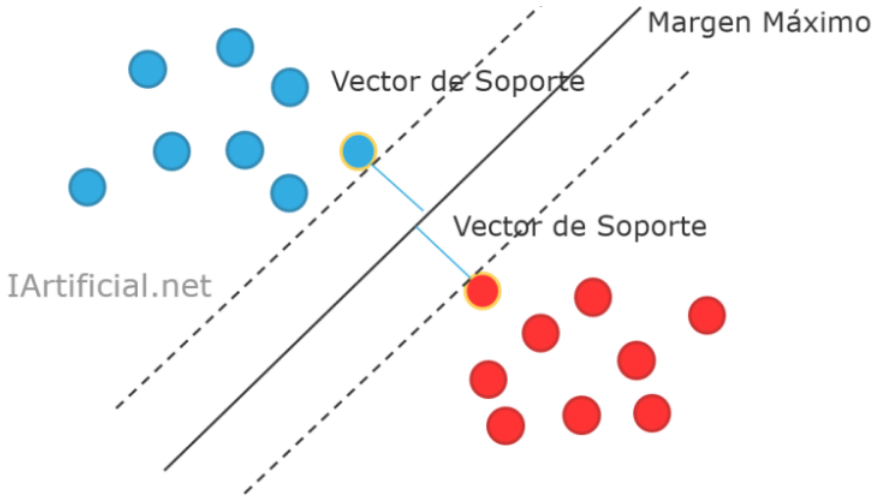


Figura 21: Hiperplano óptimo de separación entre clases [41]

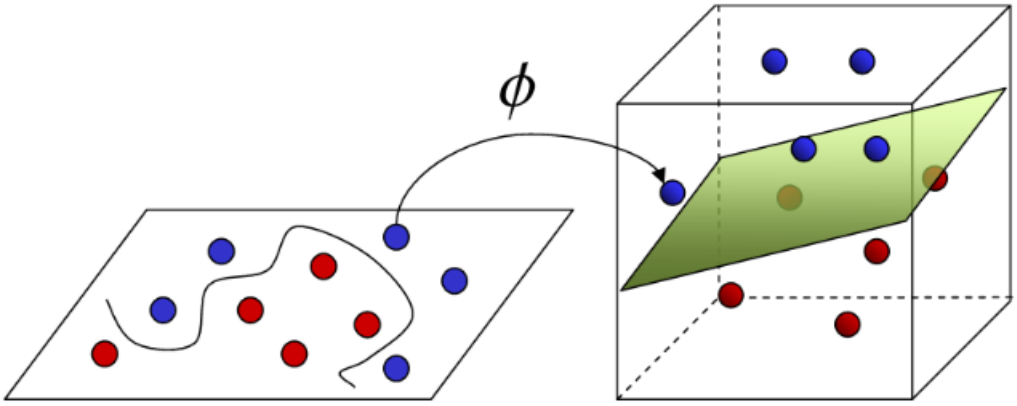


Figura 22: Uso de una nueva dimensión para encontrar una separación entre clases [41]

CAPÍTULO 7

Conexión de Electro-Cap con la Cyton Board

7.1. Planteamiento del proyecto

Luego de seleccionar la línea de investigación: "Ciencia Aplicada al Deporte" para desarrollar o continuar un proyecto, el primer paso consistió en leer y familiarizarse con todos los proyectos realizados en la Universidad del Valle bajo esta rama. Cabe mencionar que dentro de esta línea de investigación se tiene un convenio con el Comité Olímpico Guatemalteco (COG) y todos los proyectos realizados son con el fin de contribuir a esta organización para que los atletas cuenten con tecnología innovadora y necesaria para competir al más alto nivel.

Luego de leer sobre lo que se había estado realizado previamente dentro del departamento de electrónica y mecatrónica de la universidad, se investigó sobre como la tecnología y la ciencia están impactando el deporte a nivel mundial. Esta búsqueda se hizo con el fin de obtener ideas y plantear proyectos que le podrían llegar a interesar al COG según las innovaciones que se están viendo hoy en día en el deporte. Luego de plantear ciertas propuestas interesantes, se optó por continuar con los proyectos previamente comenzados en la universidad con el fin de llegar a desarrollar proyectos totalmente terminados que puedan ser utilizados perfectamente por los atletas y entrenadores nacionales.

Sin embargo, dentro de la investigación realizada se percató que los atletas tienen la tendencia de tener más trastornos de sueño que el resto de personas, por lo que en base al producto comercial Sleep Shepherd, se sugirió y aprobó darle un nuevo enfoque a los proyectos de neuroretroalimentación que se habían llevado durante el transcurso del 2018. El artefacto Sleep Shepherd, que se observa en la Figura 23, utiliza un sensor para monitorear la actividad cerebral del sueño de una persona y envía pulsos binaurales acorde a las lecturas del sensor para tratar de ayudar a la persona a dormir mejor. La idea del proyecto a desarrollar se basaba en este mismo concepto pero en una versión propia y mejorada, dado que las

reseñas del producto no eran muy favorecedoras y al consultar con un especialista del sueño llamado Dr. Luis Alejandro López, comentó que ese tipo de sensores realmente no pueden medir lo que ocurre durante el sueño. Más detalles de la reunión con este doctor se presentan en el Capítulo 10.4.

El proyecto consistiría en utilizar la placa Cyton Board y el gorro de electrodos Electro-Cap para hacer las lecturas de la actividad cerebral de una persona. Posteriormente, se utilizarían métodos de Clasificación Automática de las Etapas del Sueño o ASSC por sus siglas en inglés *Automatic Sleep Stage Classification*, para así detectar en que parte del ciclo del sueño se encuentra la persona en base a su actividad cerebral. Finalmente, dependiendo de la etapa del sueño detectada se reproduciría el pulso binaural correspondiente a esa etapa para hacer el sueño mas eficiente.

Figura 23: Artefacto Sleep Shepherd [42]

7.2. Familiarización con las herramientas

Dado que el año anterior se trabajó con el software y equipo de OpenBCI, se confirmó que el procesador Cyton Board era la mejor opción para llevar a cabo la etapa de la lectura de las ondas cerebrales. El principal motivo detrás de esta decisión fue que ya se contaba con los trabajos de graduación del año pasado por lo que servirían como base para aprender a utilizar el equipo.

Además, ya que la empresa OpenBCI es de código abierto también se tendrían a disposición códigos, tutoriales y foros para darle un uso correcto al equipo. Finalmente, también se decidió utilizar la Cyton Board debido a que realizar desde cero el hardware y software para procesar las ondas cerebrales consumiría demasiado tiempo y se desvaría demasiado de los objetivos planteados en este trabajo. En la Figura 24 se puede observar la mencionada página de OpenBCI donde se encuentran los materiales de apoyo.

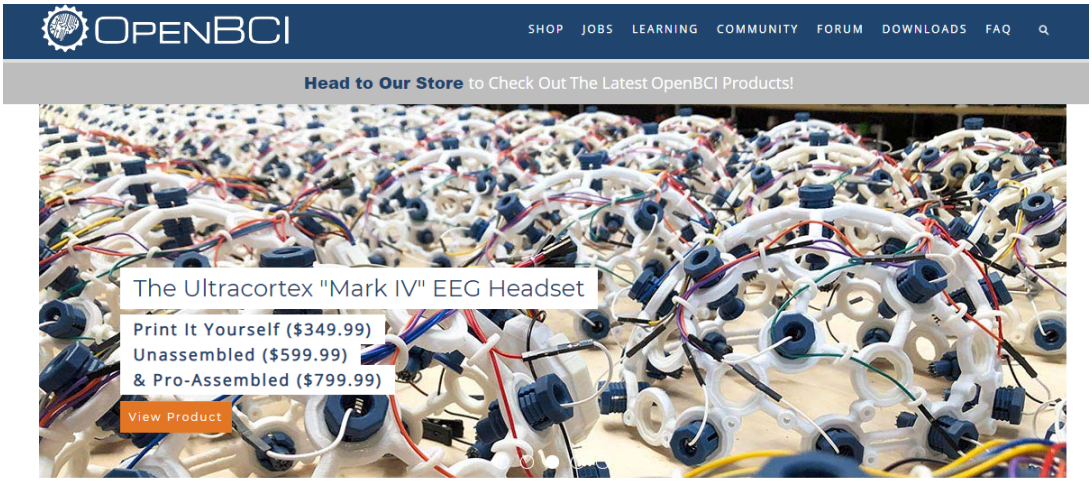


Figura 24: Página de inicio de la empresa OpenBCI [22]

Luego de tomar esa decisión, el siguiente paso fue leer todo el material disponible en esta página para conocer el equipo con el que se iba a estar trabajando, específicamente la Cyton Board, su Dongle, y los electrodos de referencia. El equipo completo de OpenBCI a utilizar se puede observar en la Figura 25. De esta forma, se estudiaron desde las propiedades físicas del equipo hasta la estructura de las tramas de datos que se envían durante la comunicación entre dispositivos. Posteriormente se prosiguió a familiarizarse con todas las aplicaciones, configuraciones y capacidades que tiene el software que provee esta empresa para hacer las lecturas de cualquier tipo de señal biomédica.

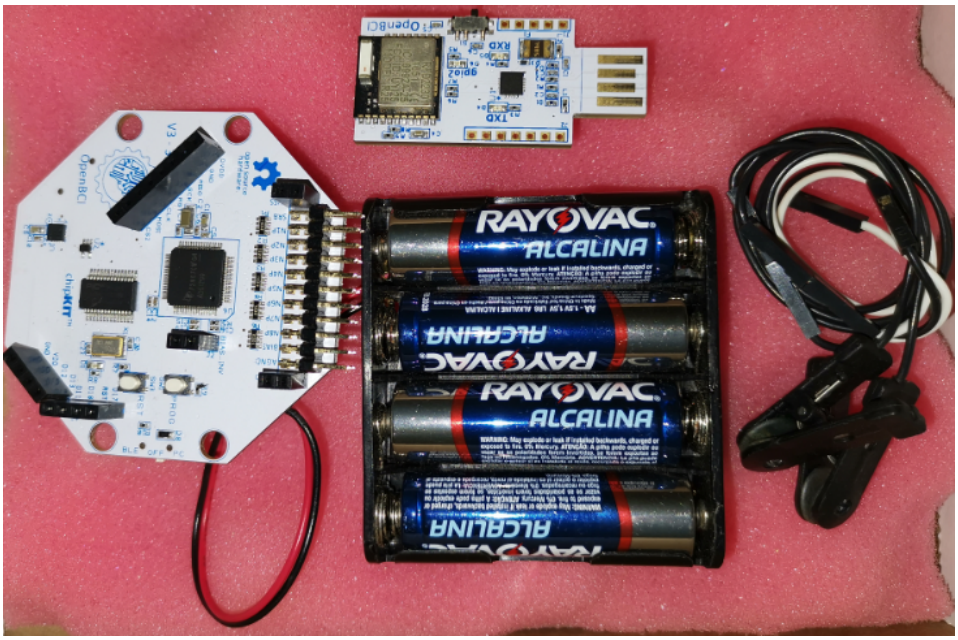


Figura 25: Equipo completo de OpenBCI

Finalmente, siguiendo todos los tutoriales dentro de la página se pudo instalar el software OpenBCI-GUI en una computadora Windows de 64-bits, junto a los drivers necesarios para hacer lecturas del Dongle en un puerto serial de la computadora. Al terminar de leer todas las recomendaciones de uso, se siguieron los pasos de una guía para comenzar a realizar las primeras pruebas de funcionamiento. Al término de dicha guía, la conexión de Cyton Board y el Dongle con la interfaz fue efectiva, y ya se podían apreciar mediciones en tiempo real del ruido generado al tocar con los dedos alguno de los 8 canales de la placa tal como se observa en la Figura 26.

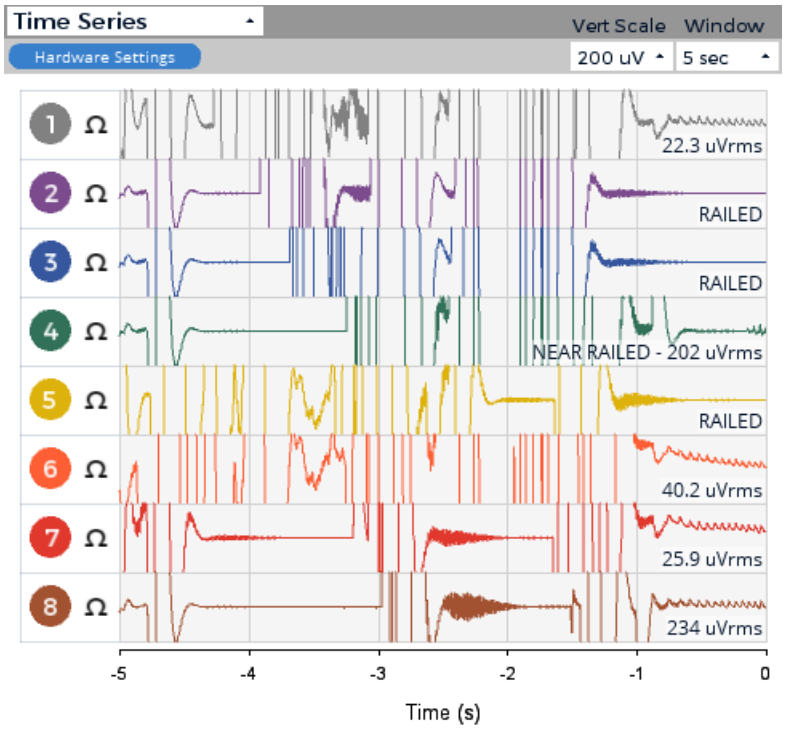


Figura 26: Ruido medido en todos los canales de la interfaz

7.3. Pruebas con el Electro-Cap

Dado que en los proyectos pasados se utilizó la Cyton Board junto a los cascos Ultracortex Mark IV, también de OpenBCI, se determinó que se tenía que buscar otra opción para realizar los electroencefalogramas dado que con uno de estos cascos no sería factible dormir por aspectos claros de comodidad. Afortunadamente en años anteriores se utilizó el gorro de electrodos Electro-Cap, el cual sería una opción mucha más factible para realizar un estudio de sueño.

Además de una mayor comodidad, el Electro-Cap también tenía las grandes ventajas de poseer más electrodos que los cascos y que sus cables estaban conectados a una terminal DB-25 por lo que no había riesgo de que se enredaran durante la noche. Sin embargo, dado que ambos equipos eran de compañías distintas era necesario hacer pruebas de compatibilidad para verificar que si pudieran obtener lecturas correctas a partir de ambos. El equipo del Electro-Cap que se tenía disponible se puede apreciar en la Figura 27.

Figura 27: Equipo para utilizar el Electro-Cap

Para estas primeras pruebas simplemente se conectaron *jumpers* en los pines de la terminal DB-25 hasta los pines de lectura de la Cyton, seleccionado los pines según los electrodos deseados acorde a la Cuadro 2. Evidentemente este tipo conexión no era lo más apropiado dado las dimensiones distintas que tienen los pines del DB-25 con el de los jumpers, provocando lo que podría ser una conexión muy ruidosa e inestable.

Sin embargo, esta prueba solo se realizó con el propósito de verificar rápidamente la factibilidad de conexión entre la Cyton y el Electro-Cap en lo que se evaluaban distintas opciones para tener una conexión más sólida y eficiente. Cabe mencionar que en esta prueba los electrodos de referencia fueron colocados uno en cada oreja, el primero hacia el pin SBR, canal de referencia para el resto de pines, y el otro hacia al canal BIAS para reducción de ruido. Los electrodos comúnmente se colocan en las orejas dado su diferencia de potencial neutral respecto a los canales ubicados en la cabeza.

Estos electrodos de referencia fueron los que se utilizaron en los proyectos pasados y que venían junto al kit del casco Ultracortex Mark IV que se compró para la realización de esos proyectos. También es importante mencionar que el gel que utiliza el Electro-Cap para hacer la conexión de los electrodos con el cuero cabelludo estaba vencido desde hace 2 años, sin embargo para aprovechar el tiempo en lo que se solicitaba y se obtenía un nuevo gel, se decidió hacer las pruebas con el gel que se tenía disponible en ese momento a pesar de las malas condiciones en las que se encontraba.

Posteriormente, se conectaron electrodos ubicados en diferentes lóbulos del cerebro hacia la Cyton Board y se prosiguió a realizar la primera prueba. Desafortunadamente todos los canales utilizados mostraron el mensaje *railed* indicando que el controlador no detectó ninguna diferencia de potencial entre los electrodos y las referencias tal como se muestra en la Figura 28. Por más que se tratara de crear perturbaciones que pudieran detectadas por el sistema, tales como morder o levantar las cejas, el resultado siempre fue el mismo.

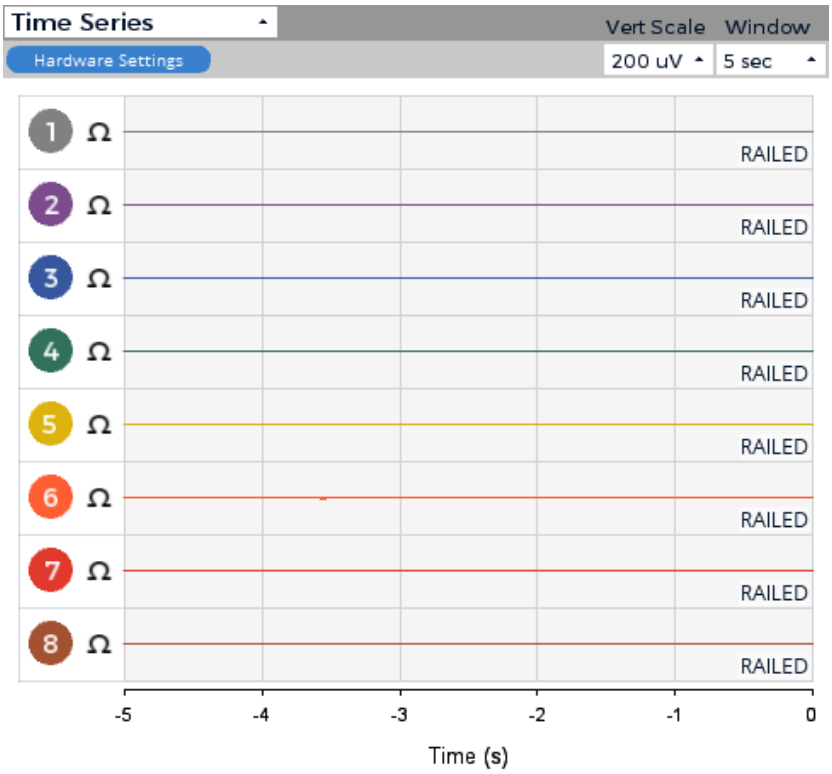


Figura 28: Mensaje de error durante la primera prueba

7.3.1. Nuevas pruebas de conexión

Debido a las múltiples fuentes de error presentes durante la primera prueba, se fueron descartando posibles factores que pudieran haber provocado las fallas en las lecturas. El primer paso para realizar las nuevas pruebas de conexión fue obtener un nuevo gel conductor para la lectura de los electrodos, el cual solicitado y entregado por el departamento de Electrónica. Posteriormente, se era necesario hacer un adaptador más apropiado para mejorar la conexión del DB-25 hacia los pines de la Cyton.

Para llevar a cabo esta última tarea se cortaron por la mitad unos cuantos cables para realizar conexiones también conocidos como *jumpers*. Posteriormente, se peló una pequeña porción de los cables por el extremo que fue cortado. Luego se soldaron dichos extremos de los cables a algunos pines deseados de un adaptador hembra del DB-25 especialmente diseñado para ser soldado. Finalmente, se utilizaron y calentaron unos tubos termoencogibles para asegurar la conexiones. El resultado final del adaptador se puede apreciar en las Figura 29.

Una vez se adquirió el nuevo gel y se terminó de fabricar el adaptador, se volvió a realizar la prueba de conexión bajo las mismas condiciones que en el caso anterior. Para el adaptador se eligieron por el momento solo algunos electrodos básicos en una electroencefalografía para realizar esta prueba. A pesar de contar con un adaptador estable y un gel conductor nuevo, el resultado de esta prueba mostró el mismo mensaje de error por lo que se debía de seguir explorando otras posibles fallas antes de descartar la factibilidad de la conexión.

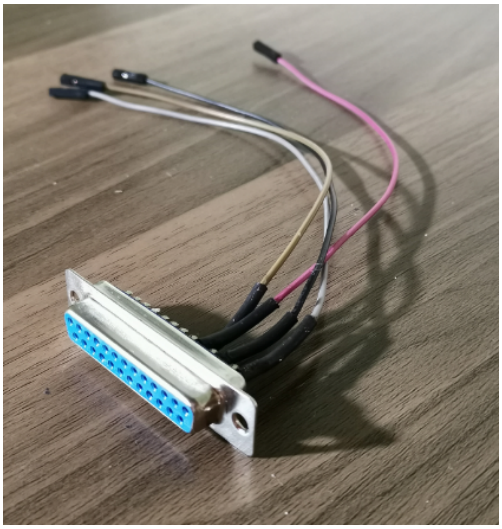


Figura 29: Adaptador soldado para la conexión con el DB-25

Dado estas nuevas fallas el siguiente paso a tomar fue descartar que el problema estuviera relacionado con el equipo de OpenBCI, por lo que se prosiguió a realizar las mismas pruebas pero en este caso utilizando el casco de electrodos de OpenBCI, Figura 24, en lugar del Electro-Cap. En este caso ya se obtuvieron las señales esperadas como se puede observar en la Figura 30, cabe mencionar que el casco solo tenía 7 electrodos disponibles, dado que un electrodo no tenía su cable de conexión, por lo que el canal 1 estuvo apagado durante la prueba.

Además, el canal 4 permaneció desconectado para verificar la presencia de ruido. La perturbación que se observa en la Figura 31 fue una prueba en la que se aprieta el mentón para comprobar que si detectada en tiempo por el equipo. Con estos resultados se pudo confirmar que la Cyton Board, Dongle e interfaz estaban funcionado correctamente y el problema no estaba relacionado a una falla física o de configuración de estos equipos.

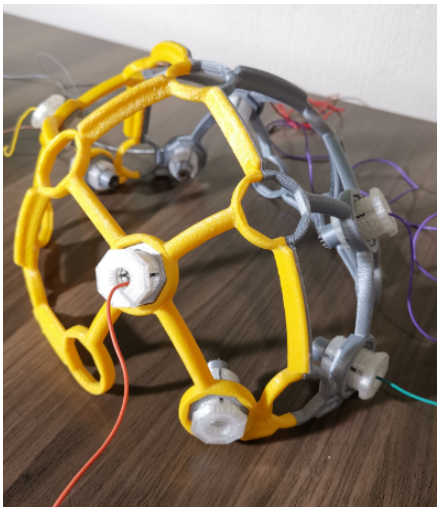


Figura 30: Casco de electrodos Ultracortex Mark IV de OpenBCI

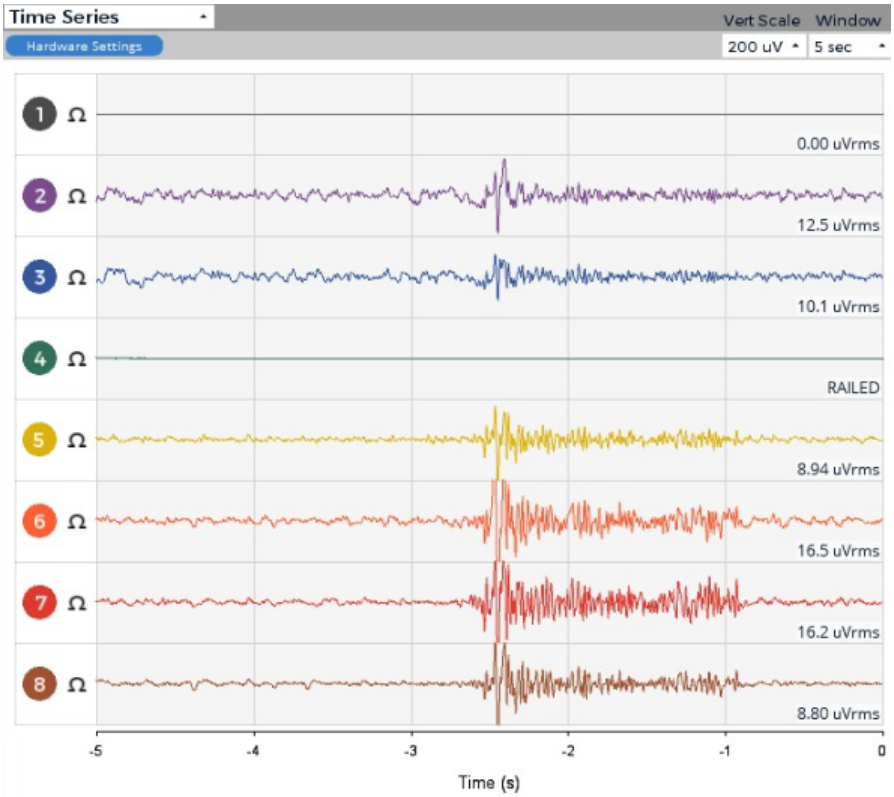


Figura 31: Señales obtenidas con casco Ultracortex Mark IV

Con las conclusiones del paso anterior se continuaron realizando pruebas para tratar de encontrar la razón del fallo en las lecturas al utilizar Electro-Cap. Con un multímetro se midió que existiera continuidad eléctrica de cada electrodo con su pin correspondiente en el terminal DB-25 según el Cuadro 3, en donde efectivamente todos los electrodos mostraron continuidad en su pin correspondiente. Posteriormente se midió la resistencia de cada electrodo del gorro a su pin del DB-25 para verificar que fuera menor a los 5kΩ, ya que si la resistencia fuera mayor a esa cifra las lecturas probablemente ya no serían correctas, según una consulta en el foro de ayuda de OpenBCI. Sin embargo, nuevamente todos los electrodos cumplieron con este requisito.

A pesar de que las últimas pruebas invitaban a pensar de que todos los electrodos estaban en condiciones óptimas para funcionar, se prosiguió a realizar una nueva prueba pero utilizando los electrodos que visiblemente se encontraban en mejor estado que el resto, ya que había varios electrodos que por su uso constante y mala limpieza se miraban bastante desgastados. Aun así, los resultados fueron los mismos que para los casos anteriores.

De igual forma, dado que al gorro no se le dio la limpieza adecuada en años anteriores, como una nueva alternativa se probó limpiar los electrodos con agua caliente, detergente, un cepillo especial e hisopos, lo cual es recomendable hacer cada cierto tiempo si el gorro se utiliza constantemente. Como una posible solución también se investigó que el electrodo Fz0 podría ser utilizado como una referencia en lugar de las orejas, sin embargo, a pesar de que los electrodos se miraban en muchas mejores condiciones y se tenía una nueva referencia, el resultado de la conexión continuó siendo negativo.

7.3.2. Pruebas finales

Dado que no se encontraban soluciones para el problema se consultó nuevamente en los foros de la empresa OpenBCI para verificar si era posible la conexión del Electro-Cap. Dentro del foro se comentó que el problema posiblemente era que se estaban utilizando electrodos de referencia inadecuados, ya que los que se estaban usando estaban diseñados para los electrodos de los cascos, los cuales están compuestos de electrodos secos. Dado que los electrodos del Electro-Cap son electrodos húmedos el tipo de metal es distinto y al combinar diferentes tipos de metal al hacer las lecturas se genera un efecto galvánico que no permite obtener las señales.

Con esta nueva información se prosiguió a conseguir los electrodos apropiados para el gorro y afortunadamente la universidad también contaba con ellos. Sin embargo dado que la terminal de dichas referencias es conocida como *touchproof*, también se tuvieron que construir unos adaptadores para estos electrodos de la misma manera que para el DB-25, con la diferencia que se soldaron unos jacks especiales para este tipo de terminal. En la Figura 32 se pueden ver estos nuevos electrodos mientras que en la Figura 33 se observa el resultado final de los adaptadores realizados. Se decidió no comprar adaptadores para esta conexión, ya que solamente se vendían en paquetes muy grandes y no se consideró un gasto necesario para solo dos electrodos.

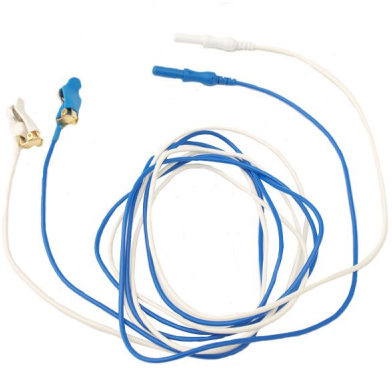


Figura 32: Electrodos de referencia para el Electro-Cap

Figura 33: Adaptadores para electrodos de referencia

Finalmente, con estas nuevas referencias ya se obtuvieron señales correctas del Electro-Cap a partir de la Cyton Board tal como se puede observar en la Figura 34. En dicho resultado lo que se puede apreciar es una prueba estándar de funcionamiento en los estudios EEG, en donde se cierran los ojos por unos segundos para aumentar la generación de ondas alfas, las cuales fueran captadas a través del electrodo 01 y 02, los cuales son el mejor lóbulo para ver este tipo de ondas.

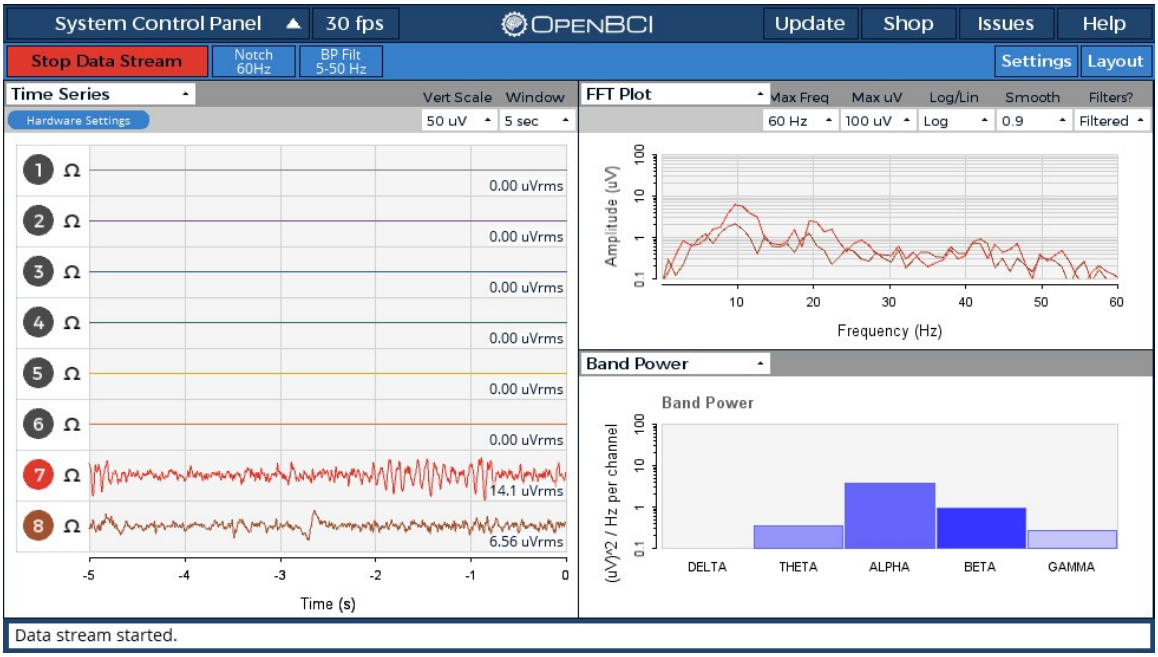


Figura 34: Lectura exitosa a partir del Electro-Cap

CAPÍTULO 8

Desarrollo de algoritmo generador de pulsos binaurales

8.1. Análisis de opciones

En una primera instancia se había planteado que la generación de los pulsos binaurales se realizara por medio de un hardware; sin embargo, luego de consultas a asesor personal, catedráticos y algunas investigaciones, se determinó que esta opción sería contraproducente en comparación a realizar la misma tarea mediante software. Ya que a pesar de que para generar los pulsos binaurales solamente se tendrían que generar ondas sinusoidales de distintas frecuencias, esta podría terminar siendo una tarea que consumiera mucho tiempo y que no permitiera cumplir el resto de objetivos planteados.

Los circuitos de audio no son nada trivial, ya que a pesar de que en teoría sí se podría llegar a fabricar este módulo, en la práctica se estaría lidiando con factores como ruido, interferencia, control de ganancia, entre otros, que podrían complicar demasiado una buena implementación. En conclusión, se llegó a esta decisión ya eventualmente el objetivo final de este proyecto, conforme se desarrollen más fases, será ayudar a los atletas a tener un sueño más eficiente, y por más tiempo que se le dedicara al hardware, jamás se igualaría la precisión, calidad y rapidez con la que se puede generar y manipular audio por medio de software.

Luego de este análisis se evaluó en que programa se desarrollarían los audios que generen los pulsos binaurales. Aunque existían varias aplicaciones que automáticamente generaban los pulsos, se decidió que era una mejor opción desarrollar los audios desde cero por medio de algún lenguaje de programación. De esta manera se tendría control sobre todas las variables con las que se generarían los tonos estéreo, tales como la frecuencia central, la diferencia entre ambas frecuencias, el número de muestras, la duración del archivo, etc. De esta forma para fases futuras del proyecto, se pueden llevar a cabo varias experimentaciones con pacientes evaluando el efecto que tienen distintas frecuencias y tiempo del estímulo.

Dentro de los principales lenguajes de programación para llevar a cabo esta tarea se consideró y evaluó utilizar Matlab, Java y Python. Finalmente se eligió Python debido a la simplicidad de su sintaxis, la gran cantidad de documentación que tiene, su manejo de listas y la ventajas que brindaban librerías como *numpy*, *struct*, *scipy.io.wavfile* para la lectura y escritura de archivos WAV. La elección de almacenar los audios en este tipo de formato se basó principalmente en que se trata de un tipo de formato que permite el análisis, lectura y escritura de datos de una manera mucho más accesible que cualquier otro tipo de formato que sí comprima los datos, como por ejemplo MP3.

Ya que se trabajaría con audios digitales sin compresión, los archivos ocuparían un mayor espacio en la memoria de la computadora. Sin embargo, no se esperaría la creación de demasiados archivos como que para la memoria fuera un factor a considerar, en cualquier caso la conversión de un archivo WAV a cualquier otro tipo de archivo para ahorrar espacio se podría hacer fácilmente con cualquier convertidor en línea.

8.2. Primeras pruebas

El objetivo de este algoritmo sería poder crear tonos estéreo donde ambos canales correspondieran a una onda sinusoidal pero de distinta frecuencia. De esta forma, en el oído derecho se captaría una frecuencia y en el izquierdo otra, produciendo el efecto de los pulsos binaurales. Para eventualmente poder escribir archivos de audio con formato WAV, primero había que comprender su estructura de almacenamiento de datos y analizar cómo se realiza el manejo de estos archivos en Python.

Para llevar esto acabo se hicieron unas pequeñas pruebas de lectura de archivos WAV descargados de una página con muestras gratis de archivos de audio. Cabe mencionar que al leer un archivo binario, lo que se obtiene es un una variable de tipo *string* que tiene empacados valores desde “0x00” hasta “0xff”, en donde “0x” es un indicativo que se esta utilizando el sistema hexadecimal. Dado que este sistema utiliza base 16, cada dos dígitos hexadecimales equivaldrían a 1 byte u 8 bits de información en el sistema binario. En la Figura 35 se pueden ver los primeros 50 bytes, incluido el encabezado, de un archivo de audio llamado “Yamaha-V50-Rock-Beat-120bpm.wav”.

```
In [2]: audio_file
Out[2]: 'RIFFj^\x05\x00WAVEfmt
\x10\x00\x00\x00\x01\x00\x02\x00D\xac\x00\x00\x10\xb1\x02\x00\x04\x00\x10\x00data\xd0]\x05
\x00\x00\x00\xff\xff\x02\x00'
```

Figura 35: Muestra de los datos de un archivo de audio

Para poder interpretar estos datos de una manera fácil se hizo uso de la librería *struct* de Python, la cual es utilizada para manejar *strings* empacados con datos binarios. Su funcionalidad principal es la de realizar conversiones de las muestras binarias a cualquier otro tipo de variable, proceso que también se le conoce como “desempacar” los datos. Cabe mencionar que esta librería también es capaz de realizar el proceso opuesto, conocido como “empacar” los datos, en donde las variables con las que se está trabajando pueden ser empacadas en datos binarios para que puedan ser interpretados por una computadora.

Esta librería es especialmente útil para evitar tener que realizar operaciones matemáticas tediosas para empacar o desempacar cada muestra de los datos de interés. Dado que se utiliza el sistema hexadecimal para representar los datos binarios que forman el archivo WAV, cada 4 dígitos hexadecimales equivalen a una muestra de dos bytes de un archivo tipo mono. En la Figura 36 se mira cómo se desempaqueta una muestra del archivo de audio anterior a un entero con signo de 16 bits. Por otro lado en la Figura 37 se puede observar el proceso opuesto en el que un entero fue empacado como un dato binario representado en el sistema hexadecimal.

```
In [3]: muestra = struct.unpack('h','\x10\xb1')

In [4]: muestra
Out[4]: (-20208,)
```

Figura 36: Muestra de datos desempacados de un archivo de audio tipo mono

```
In [5]: muestra = struct.pack('h',-20208)

In [6]: muestra
Out[6]: '\x10\xb1'
```

Figura 37: Muestra de datos empacados de un archivo de audio tipo mono

Con este nuevo entendimiento se comenzaron a realizar pruebas para primero poder escribir audios mono y luego pasar a los audios estéreo. Para llevar esto a cabo se determinó que había dos opciones principales para poder escribir el archivo de audio. La primera opción se basaba en el uso de la ya mencionada librería *struct* junto a la librería *wave*, mientras que en la segunda opción la escritura de los archivos se realizaba mediante las librerías *numpy* y *scipy.io.wavfile*. Aunque ambas opciones funcionaban bajo la misma lógica, se prefirió utilizar la segunda opción debido a que esas librerías tenían un funcionamiento más simple que permitía obtener el mismo resultado pero con considerablemente menos líneas de código.

Para estas primeras pruebas se escribió un audio mono a partir de una señal casi cuadrada que se obtuvo al sumar tres componentes sinusoidales de la serie de Fourier de una señal cuadrada. Para poder validar que efectivamente se creó el archivo de audio esperado se utilizó la aplicación Audacity, la cual por muchos años ha sido un editor de audio muy reconocido. En la Figura 38 se puede observar la señal de audio que se generó y que efectivamente cumplía con lo esperado, la logística detrás de como se desarrolló el archivo se describe en la siguiente sección dado que el principio de funcionamiento es el mismo tanto para archivos mono como estéreo y se resumen en la Figura 43.

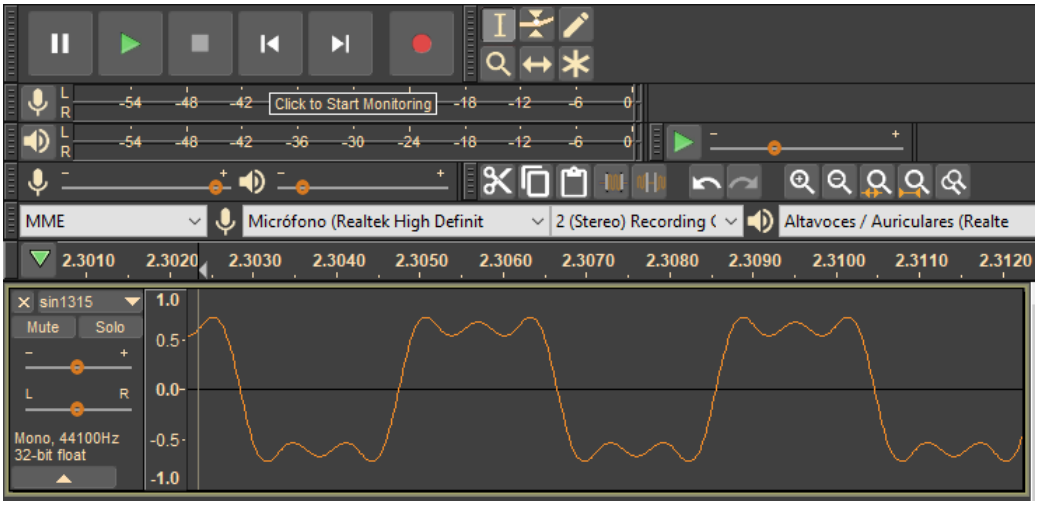


Figura 38: Archivo mono de una onda cuadrada

8.3. Desarrollo de algoritmo

Una vez se manejaron los archivos mono, se pasó a hacer pruebas para generar tonos estéreo. Cabe recordar que el objetivo de estos archivos era obtener una señal sinusoidal pura para cada canal, donde cada señal fuera de distinta frecuencia. De esta forma la resta entre ambas frecuencias sería la frecuencia del pulso binaural que se buscaría generar en una persona. El primero paso del proceso fue importar las librerías *numpy* y *scipy.io.wavfile* para así poder utilizar sus módulos en la creación de los archivos. Con las herramientas de las librerías ya disponibles se prosiguió a crear una onda sinusoidal de manera digital utilizando como base la siguiente fórmula:

$$y = \sum_{n=0}^{N} \frac{4}{\pi} \sin(\frac{2\pi * n}{N})$$

en donde N es el número de muestras que tiene un período completo de la señal y n es una variable iterativa para discretizar la señal a lo largo de ese período. Para encontrar dicho valor N se debe de multiplicar la tasa de muestreo (Tm) del archivo con el período de la señal deseada (1/T). Por ejemplo, si se quisiera obtener una onda de 100 Hz, el número de muestras N en un periodo se determinaría de la siguiente manera:

$$N = \frac{Muestras}{Periodo} = Tm * \frac{1}{T} \longrightarrow \frac{44100\ Muestras}{Segundo} * \frac{1}{100Hz} = \frac{441\ Muestras}{Periodo}$$

Con este resultado y la fórmula mencionada anteriormente se obtiene el muestreo de un período completo de la señal. Dado que una sinusoidal es una función periódica, el siguiente paso fue encontrar el número de veces que se debía de repetir el periodo muestreado para que el archivo de audio tuviera la duración deseada.

Continuando con el ejemplo anterior, si se quisiera que el archivo fuera de un minuto primero se debían de encontrar el número de muestras totales que tendría dicha señal, tal como se muestra con la ecuación a continuación:

$$t = \frac{Cm}{Tm} \longrightarrow 60 segundos * \frac{441\ Muestras}{Periodo} = 2,646,000\ muestras$$

donde Cm es la cantidad de muestras totales. Con este valor ya calculado se prosiguió a calcular la cantidad de periodos (Cp) necesarios para obtener una señal de una duración deseada, tal como se muestra en el cálculo posterior:

$$Cp = 2,646,000\ muestras * \frac{1\ Periodo}{441\ muestras} = 6,000\ periodos$$

Con estos resultados se muestreó un período de la señal y se replicó 6000 veces para que el archivo tuviera una duración de 60 segundos para el ejemplo planteado. En las sección final se detalla la duración de los archivos finales que fueron requeridos. El resultado de este proceso se almacenó en un arreglo de datos de Numpy, dado que es un requerimiento de Scipy.io.wavfile para poder escribir el archivo WAV. Al utilizar esta librería el formato del archivo WAV depende del tipo de arreglo de Numpy donde estén almacenados los datos.

En este caso el tipo de arreglo era de tipo flotante de 32 bits por lo que los valores máximos y mínimos que se podrían escribir en el archivo eran de 1.0 y -1.0 respectivamente. Al muestrear la sinusoidal efectivamente había valores mayores y menores a este rango, se normalizó la señal dividiendo cada dato dentro del valor máximo de la sinusoidal, de esta forma solo se tendrían datos dentro del margen requerido y se evitaría que la señal se acotara en los puntos máximos y mínimos de la señal a la hora de escribir el archivo. Luego de realizar todos estos pasos se tendría la señal completa de uno de los dos canales del archivo de audio de tipo estéreo.

Todo este proceso debía de repetirse en base a la señal deseada para el segundo canal, en este caso se calculó para obtener una señal de 120 Hz, recordando que la diferencia de frecuencias entre ambos canales sería la frecuencia del pulso binaural a generar. Al obtener los datos del segundo canal en un arreglo de Numpy distinto, solamente se debían de estructurar ambos arreglos en un nuevo arreglo bidimensional, de dos columnas y con un número de filas igual a la de cantidad muestras de las señales. Finalmente, se escribió el archivo estéreo utilizando la función *write* de la librería Scipy.io.wavfile, a la cual se le ingresaron como parámetros el nombre del archivo, la tasa de muestreo utilizada y el arreglo bidimensional con las señales para cada canal.

En la Figura 39 podemos ver el archivo generado desde Audacity, en donde se puede apreciar que efectivamente la duración del archivo es de un minuto. En la Figura 40 se observa un acercamiento en las señales y claramente se pueden distinguir las sinusoidales generadas y que ambas tienen una frecuencia distinta. Sin embargo, para poder validar que las señales sí fueran de la frecuencia que fue calculada, se utilizó el analizador de espectro de Audacity para poder examinar las intensidades de cada componente frecuencial del archivo.

En la Figura 41 se puede apreciar el espectro del archivo de audio y se pudo comprobar que las señales sí tenían la frecuencia calculada dado que los picos de mayor intensidad estaban ubicados precisamente en 100 Hz y 120 Hz. Para tener una mejor perspectiva también se utilizó la herramienta Sonic Visualizer para poder apreciar mejor donde se encontraban exactamente los picos espectrales con mayor intensidad tal como se muestra en la Figura 42. Los pasos resumidos de como desarrollaron los pulsos binaurales se encuentran en el diagrama de la Figura 43.

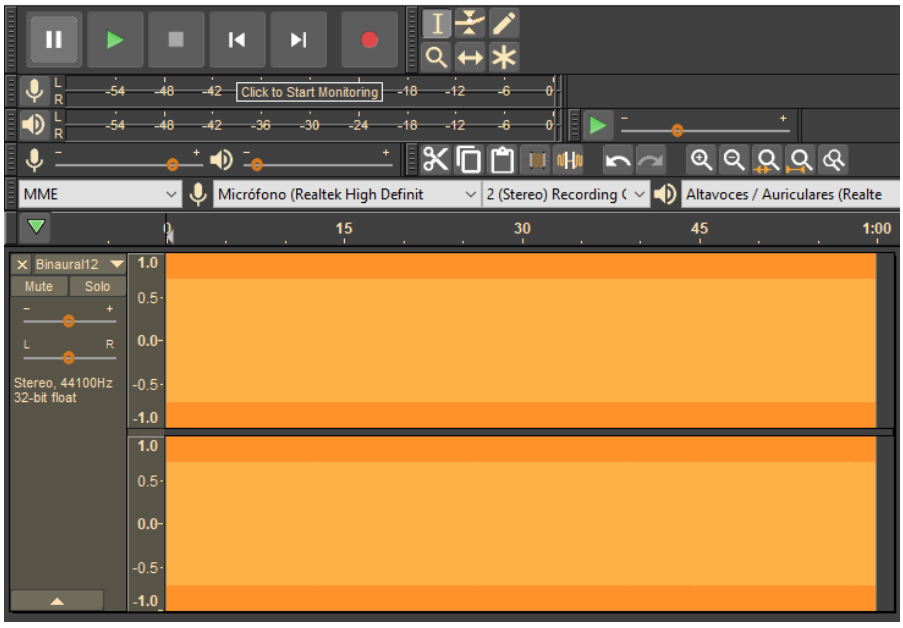


Figura 39: Archivo estéreo de un minuto de duración

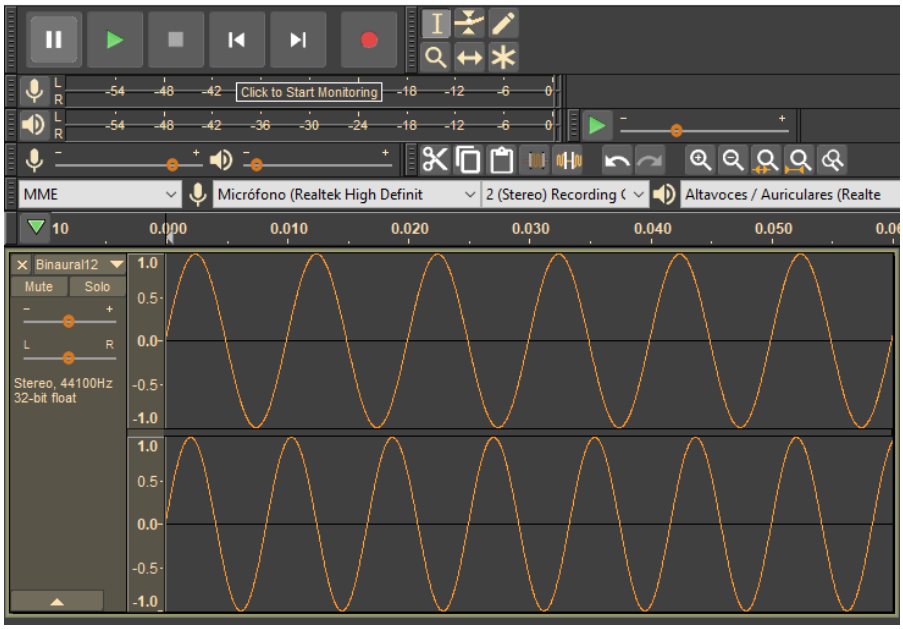


Figura 40: Señales sinusoidales generadas en el archivo de audio

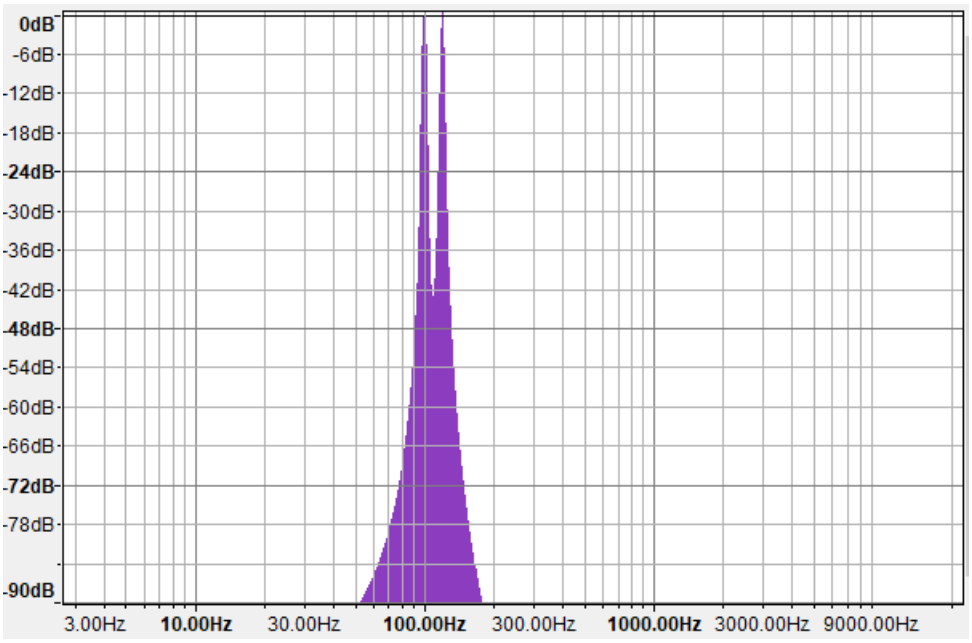


Figura 41: Espectro en frecuencia del archivo generado en Audacity

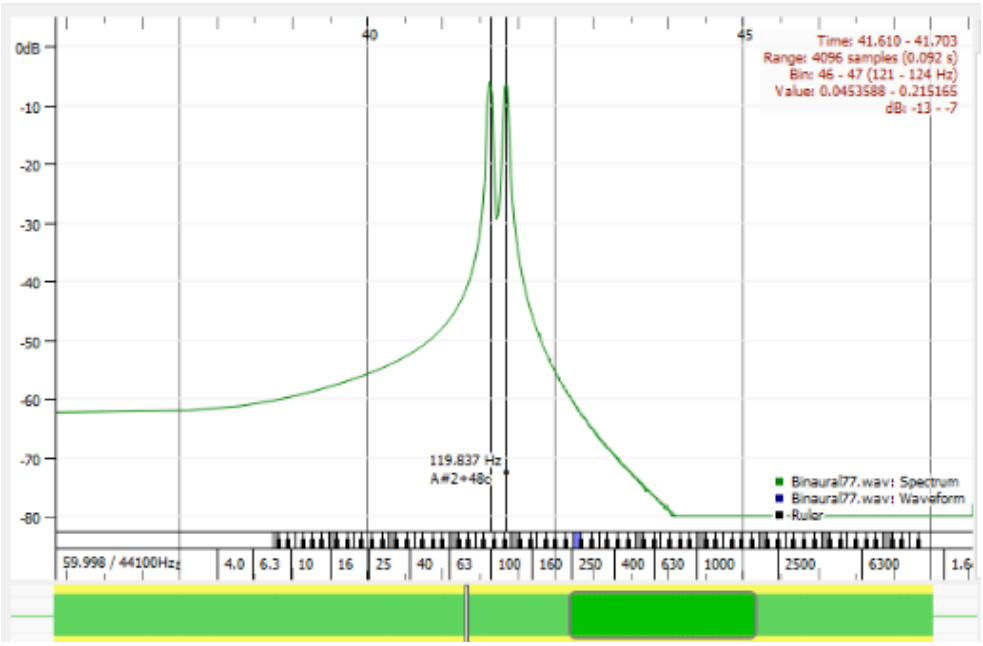


Figura 42: Espectro en frecuencia del archivo generado en Sonic Visualizer

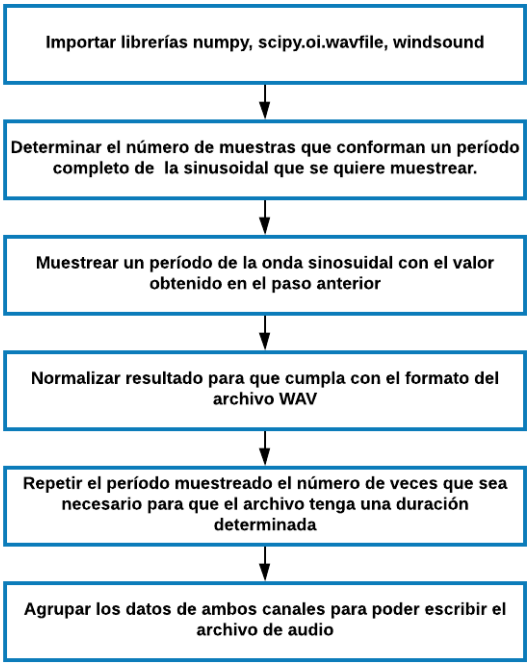


Figura 43: Diagrama de pasos seguidos para generar los pulsos binaurales

8.4. Detalles finales

Aunque ya se obtuvieran resultados satisfactorios, para poder hacer más pruebas era muy tedioso tener que calcular todos los datos de la sección anterior para cada prueba. Por dicha problemática se decidió automatizar y parametrizar el código dentro de una sola función a la que al ingresarle la frecuencia central de las señales, tasa de muestro del archivo, frecuencia del pulso binaural deseado, duración deseada y un identificador del archivo, realizaría todos los cálculos de manera automática y almacenaría el audio generado dentro de la carpeta donde estuviera el código. Además, se importó la librería windsound para poder crear una pequeña función que reprodujera los audios desde Python para no tener la necesidad de abrir un reproductor externo para escuchar los audios.

Con esta nueva función ya se podían generar archivos de manera instantánea, obteniendo distintos audios con solo variar los parámetros de la función. De esta forma se llevaron a cabo múltiples pruebas generando audios con distintas frecuencias centrales y distinta separación entre frecuencias (frecuencia del pulso binaural) para escuchar los diferentes efectos en el sonido. Todos estos archivos también se reprodujeron para comparar los resultados obtenidos con los audios producidos por la aplicación Gnaural, un generador reconocido de pulsos binaurales, en la Figura 44 se puede ver la interfaz de inicio de dicha aplicación. La percepción general era que los audios generados en Python sonaban exactamente igual a los de Gnaural. De igual forma se generó un archivo con frecuencia central en 100 Hz y unos pulsos de 20 Hz para replicar los resultados de la Figura 42. El resulto fue prácticamente idéntico, como se observa en la Figura 45, a excepción de que este audio presentaba más ruido en otras frecuencias por lo que los audios generados en Python eran incluso más puros.

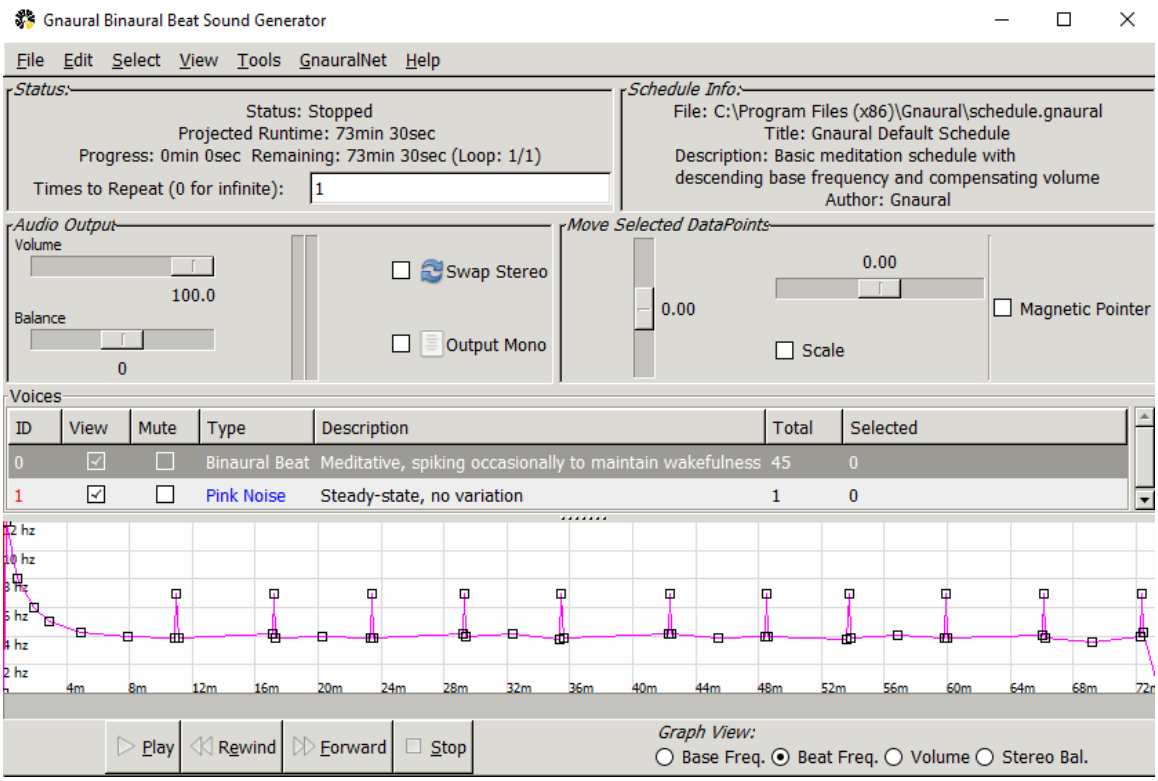


Figura 44: Interfaz de Gnaural

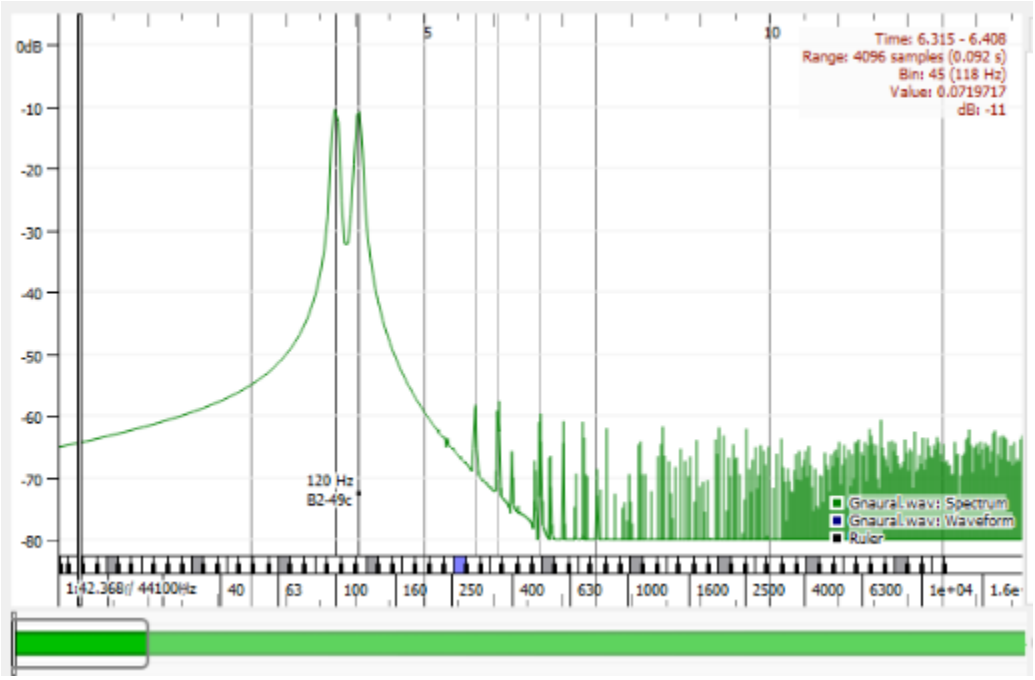


Figura 45: Pulso binaural generado en Gnaural

Como se mencionó anteriormente, los archivos WAV tienden a ocupar más espacio en memoria debido a que no hay una comprensión de datos y por su alta tasa de muestreo, 44100 muestras/segundo, lo cual puede llegar a ser significativo si un archivo es de larga duración. Es por eso que una última prueba fue reducir el parámetro de la tasa de muestreo, ya que al reducirla el número de muestras por se reduciría y por ende, el tamaño del archivo, pero sin perder la calidad del sonido. Al comenzar a bajar la frecuencia de muestreo no se percibía un cambio en los audios, hasta que eventualmente se llegó a un punto que al bajar demasiado esta tasa, pese a seguir cumpliendo con el criterio de Nyquist, el audio ya no se escuchaba tan bien como en el caso original. A pesar de que si hubo un margen donde se logró reducir el peso de los archivos sin perder calidad perceptible en el audio, la determinación final fue que para los audios finales se utilizaría la frecuencia estándar de los archivos WAV. Se llegó a esta determinación debido a que no se tendrían que generar muchos archivos y ninguno tendría una duración muy prolongada, por lo que al utilizar la tasa de 44100 muestras/segundo se estaría garantizando tener audios de la mejor calidad posible para generar los efectos de los pulsos binaurales.

CAPÍTULO 9

Obtención de los datos EEG

9.1. Análisis de opciones

Aunque la conexión con la OpenBCI_GUI ya hubiera sido exitosa y se pudieran ver correctamente las señales del Electro-Cap, dicha interfaz solo permite observar gráficamente las señales más no interactuar con la data numérica en tiempo real. Cabe mencionar que al utilizar la interfaz sí se genera un registro de los datos adquiridos y se almacenan en un archivo de texto al término de una sesión. Sin embargo, no hay forma de acceder a estos datos conforme se utiliza la interfaz y lo que se almacena son datos en crudo, es decir datos que todavía no han sido procesados, por lo que todavía no conforman las señales que se ven gráficamente. Un ejemplo de un registro de la interfaz de OpenBCI se observa en la Figura 46, en dicho registro se encuentran los datos de los ocho canales de entrada de la Cyton más los tres datos de su acelerómetro.

9.2. Users.py

Dado que para llegar a realizar la identificación de las etapas del ciclo del sueño sería necesario procesar e interactuar con los datos conforme se hicieran las lecturas, era necesario buscar otro medio para comunicarse con la Cyton Board. OpenBCI es una empresa en constante crecimiento que continuamente se actualiza y se adapta a lo que requieren sus usuarios. Es por eso que sus materiales, equipos y guías son mejorados constantemente y así brindarle a la comunidad científica las mayores facilidades posibles al hacer uso de sus servicios para llevar a cabo sus investigaciones. Al momento de evaluar que plataforma utilizar para la obtención de los datos, la página de OpenBCI ofrecía una sección de software externos compatibles con los que se podía utilizar la Cyton Board.

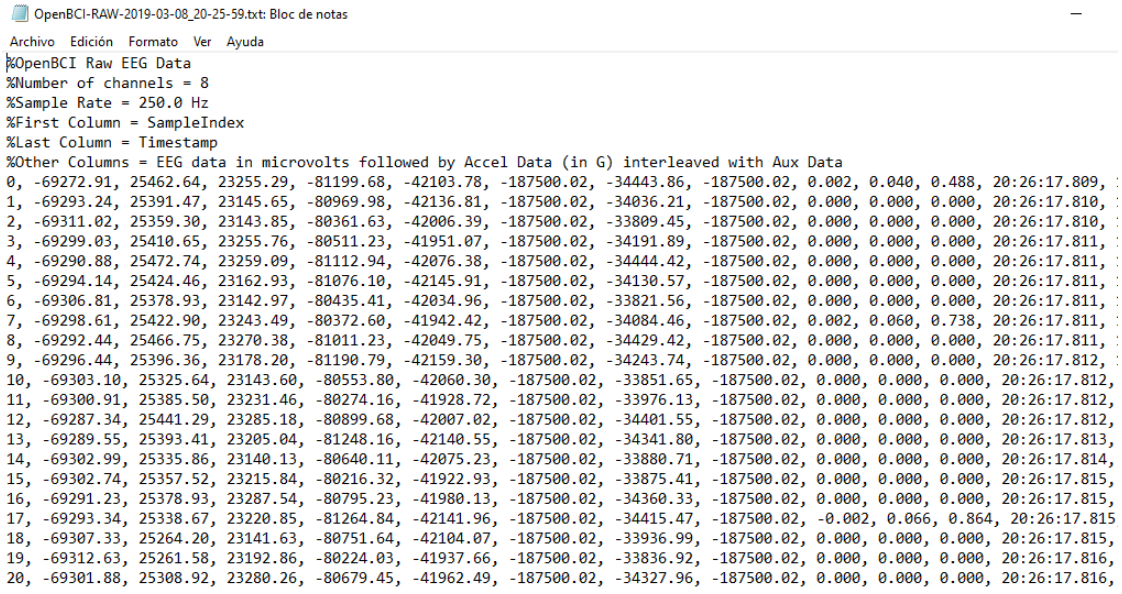


Figura 46: Datos almacenados al utilizar la OpenBCI_GUI

Dentro de estas opciones se encontraba Python, Matlab en conjunto con otros *toolboxes* adicionales, OpenVibe, Lab Stream Layer (LSL), entre otros. Luego de evaluar todas estas alternativas se decantó por utilizar Python por varias razones. Primero, esta opción contaba con más documentación que el resto de plataformas y contaba con un archivo para realizar la conexión desarrollado propiamente por OpenBCI. Segundo, aparentaba ser una conexión más simple que el resto de alternativas ya que no requería de herramientas adicionales para hacer una lectura de los datos, como en el caso de Matlab, OpenVibe y LSL.

Además, muchas de las opciones solamente eran para realizar un post-análisis, es decir, sólo eran un medio para procesar los datos luego de ser capturados por la OpenBCI_GUI y no para leer los datos en tiempo real. Y por último, dado que el algoritmo para generar pulsos binaurales ya se había desarrollado en Python, también era favorecedor que la lectura de datos estuviera en el mismo medio para cuando se tuvieran que usar ambos programas en simultáneo.

Para poder hacer una lectura de datos en Python, OpenBCI brindaba la opción de utilizar un código llamado Users.py. Este archivo puede ser encontrado en un repositorio digital junto al resto de archivos e instrucciones necesarias para su funcionamiento correcto. Luego de hacer todas las instalaciones necesarias y leer sobre cómo se utilizaba el archivo, se prosiguió a tratar de realizar la conexión con la Cyton Board. Cabe mencionar nuevamente que la página de OpenBCI se actualiza constantemente y puede ser que al momento de leer este documento haya cosas que hayan cambiado para hacer sus códigos mas eficientes.

Este código debía ser corrido desde la terminal de Windows especificando el puerto donde se iban a leer los datos del Dongle de manera serial. Adicionalmente se debía de colocar si se deseaba utilizar algún *plugin* adicional. Para este caso se habilitó que se pudieran imprimir los datos en pantalla, lo cual era el objetivo de principal de esta sección. Al correr el archivo se obtuvo la pantalla mostrada en la Figura 47 indicando que la conexión fue exitosa. Este menú de inicio mostraba que la placa estaba lista para recibir comandos.

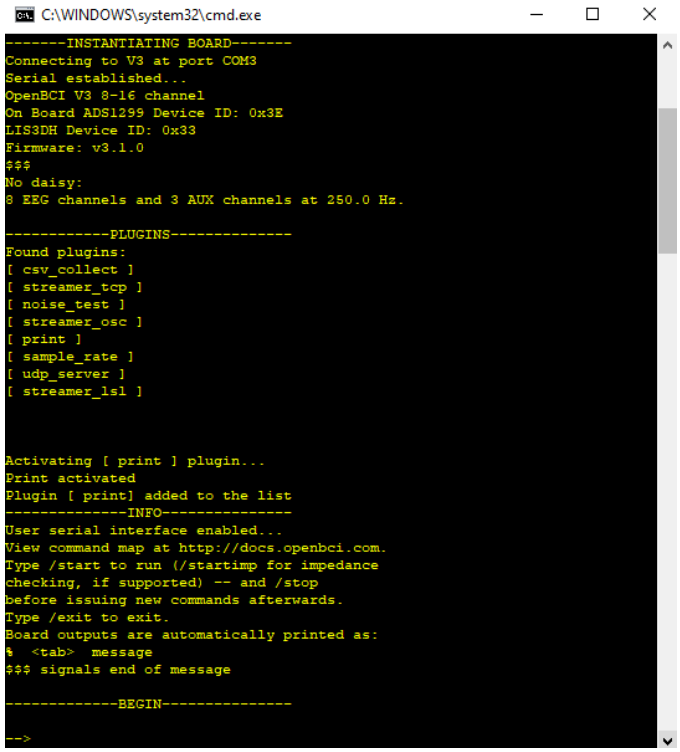


Figura 47: Conexión con la Cyton Board a través de Users.py

La Cyton Board se comunica usando comandos en forma de caracteres ASCII, los cuales están asignados a realizar funciones específicas en la placa. Los más comunes se resumen en el Cuadro 4, para más configuraciones revisar la página de OpenBCI.

Comando	Función
1,2,3,4,5,6,7,8	Apagar canal
!,@, #, $, %, ^,&,*	Encender canal
v	Reset
b	Comenzar transmisión
s	Detener transmisión

Cuadro 4: Algunos comandos para comunicarse con la Cyton Board

Con el conjunto de todos los comandos se pueden realizar básicamente las mismas configuraciones que hacen con la interfaz como modificar ganancias o cambiar las referencias. Sin embargo, los comandos descritos en el Cuadro 4 serían los de mayor interés. Al escribir los comandos en la terminal, se pudo iniciar correctamente un flujo de datos como se puede apreciar en la Figura 48, en donde se puede observar para cada muestra su número de identificación o ID , el cual es un número de 0 a 255, los datos de los 8 canales de entrada y los 3 datos del acelerómetro. Al comparar estos valores con los registros generados al utilizar la interfaz de OpenBCI se pudo concluir que los datos en crudo obtenidos eran correctos. Aunque con este código solamente se pudieran ver los datos a través de la terminal, estos archivos servirían para tener una idea base de como realizar el algoritmo para leer los datos y como un medio de comparación a la hora de leer los datos en la consola de Python.

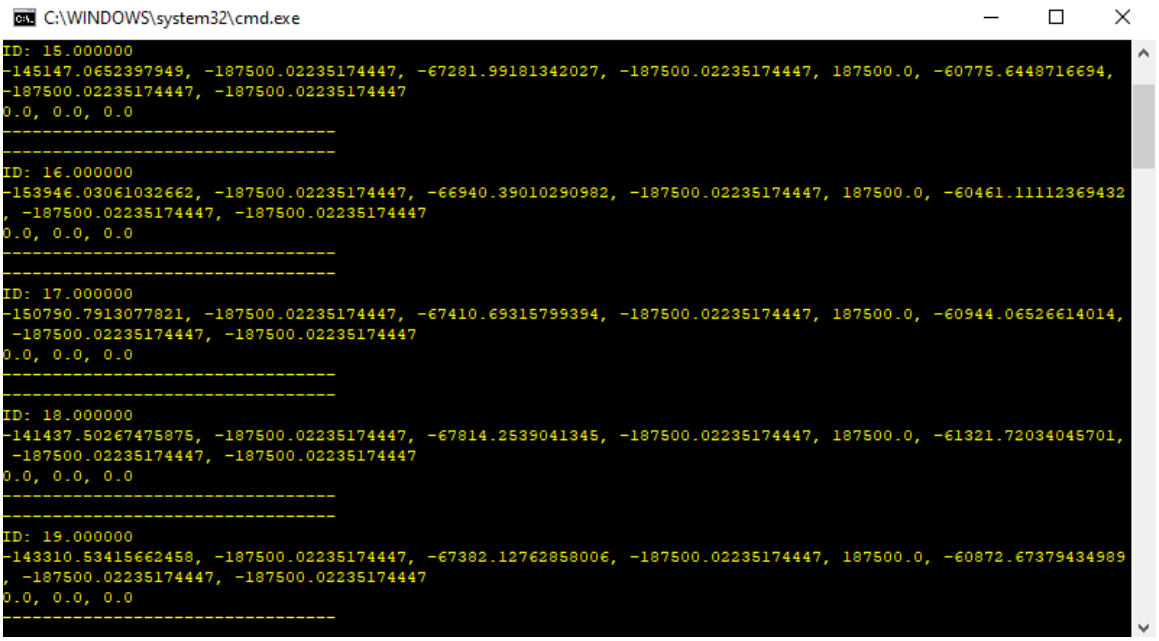


Figura 48: Lectura de datos de la Cyton Board a través de Users.py

9.3. Desarrollo de algoritmo para lectura de datos

Al explorar el código de User.py junto al resto de archivos descargados se percató que tratar de acoplar algo similar a lo que se hace en esos códigos sería una tarea sumamente compleja que consumiría demasiado tiempo. Dado que solamente se querían leer los datos, no había necesidad de hacer algo tan complejo con tantas funcionalidades como lo hacen en Users.py. Por esta razón, se decidió realizar pruebas empezando desde lo más simple, leer en el puerto serial.

El código se comenzó importando la librería *serial* de Python para luego proseguir a inicializar la conexión estableciendo el puerto y el baudaje para la comunicación, para este caso se utilizó el "COM3" y 115200 baudios/s. Con la función *write* de la librería *serial* al ejecutar el programa se enviarían a la placa los comandos "v","d" y "b". El primero sería para reiniciar la placa y asegurarse que estuviera lista para la comunicación, el segundo para configurar todos los canales en su modo por defecto, y el último para iniciar la lectura y transmisión de los datos. Luego de escribir estos comandos, se realizó un bucle donde se utiliza constantemente la función *read* para leer cada byte de la transmisión de datos hasta que encontrar el encabezado de la trama, el cual es el valor hexadecimal "a0". Se utilizó la librería binascii y la función hexlify para obtener un *string* con la representación hexadecimal de los datos binarios y así poder identificar cada vez que se detectara el encabezado. Siempre que se encontrara ese byte que marcaba el inicio de una trama de datos se proseguía a leer los 25 bytes siguientes. De esta manera se obtendría el identificador de cada trama y los valores de los 8 canales de entrada, el resto de bytes no era necesario leerlos ya que contenían los datos del acelerómetro.

Para desempacar los datos binarios del identificador se utilizó la librería *struct* tal como en la sección anterior, con la excepción de que se especificó que se quería obtener una variable de tipo *char* sin signo. De esta forma se obtendría el número de 0-255 que identifica cada trama. A continuación, los siguientes 24 bytes fueron segmentados en secciones de 3 bytes y fueron asignados a una variable distinta para separar la información de cada canal. Sin embargo, obtener una representación decimal de estos datos no iba ser tan sencillo como en el caso del identificador ya que no se podía utilizar la librería *struct* para la conversión. La razón de este problema es que no existe ningún tipo de variable estándar que utilice 3 bytes de memoria por defecto, todas utilizan ya sea 1, 2 o 4 bytes para representar las cantidades que tienen almacenadas.

Debido a esta situación se probó utilizar nuevamente la función hexlify, así convirtiendo el resultado en una variable de tipo entero y multiplicando el resultado por el factor de escalamiento de la Cyton Board para obtener un valor en microvoltios. Adicionalmente, el programa se configuró para que al presionar las teclas Crtl+C se detuviera la transmisión de datos y se cerrara el puerto serial. Al correr el código se obtuvo el resultado de la Figura 49.

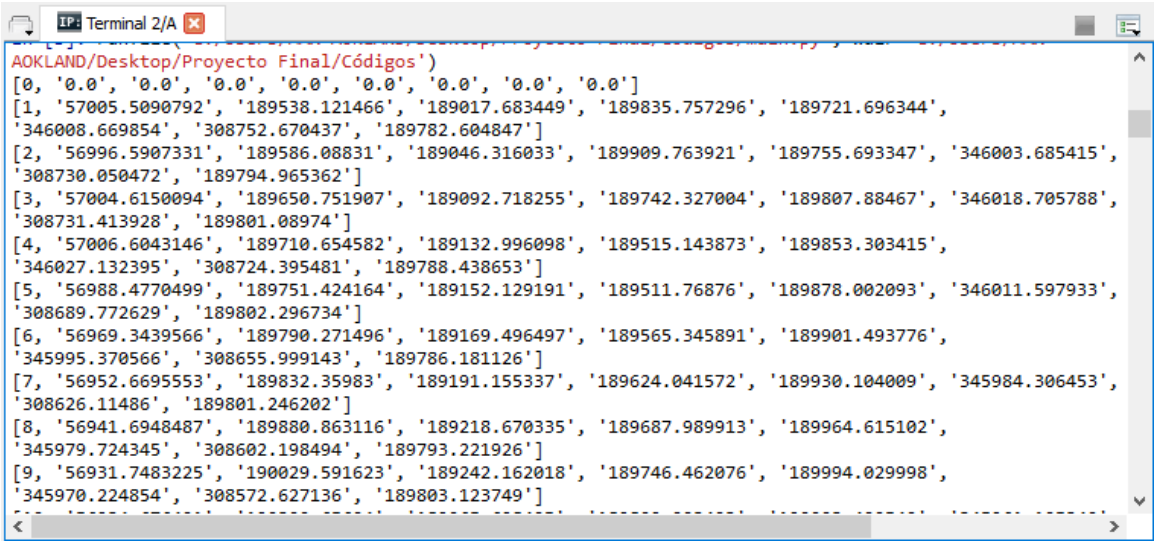


Figura 49: Lectura de datos crudos desde Python

En dicha imagen se ve como se imprime una lista que contiene en su primera casilla el identificador de cada trama y en el resto de casillas los canales de entrada, del 1 al 8. En un principio parecía que se había obtenido lo que se esperaba, sin embargo, al comparar los resultados contra lo que se obtenía en Users.py, se evidenció que valores realmente no eran los mismos. El problema era que al hacer la conversión, el valor de la respuesta era sin signo y los valores de los canales sí son con signo. Para poder obtener correctamente los valores se realizó una función basada en un segmento de código de OpenBCI para transformar un entero de 24 bits a un entero de 32 bits. Al tener una variable con 32 bits ya se podría utilizar la librería struct para obtener la conversión a un valor decimal. Con esta nueva función los resultados obtenidos se observan en la Figura 50 negativos en lo datos. Al hacer la misma comparación que en el caso anterior, daba la impresión de que la adquisición de los datos crudos era correcta.

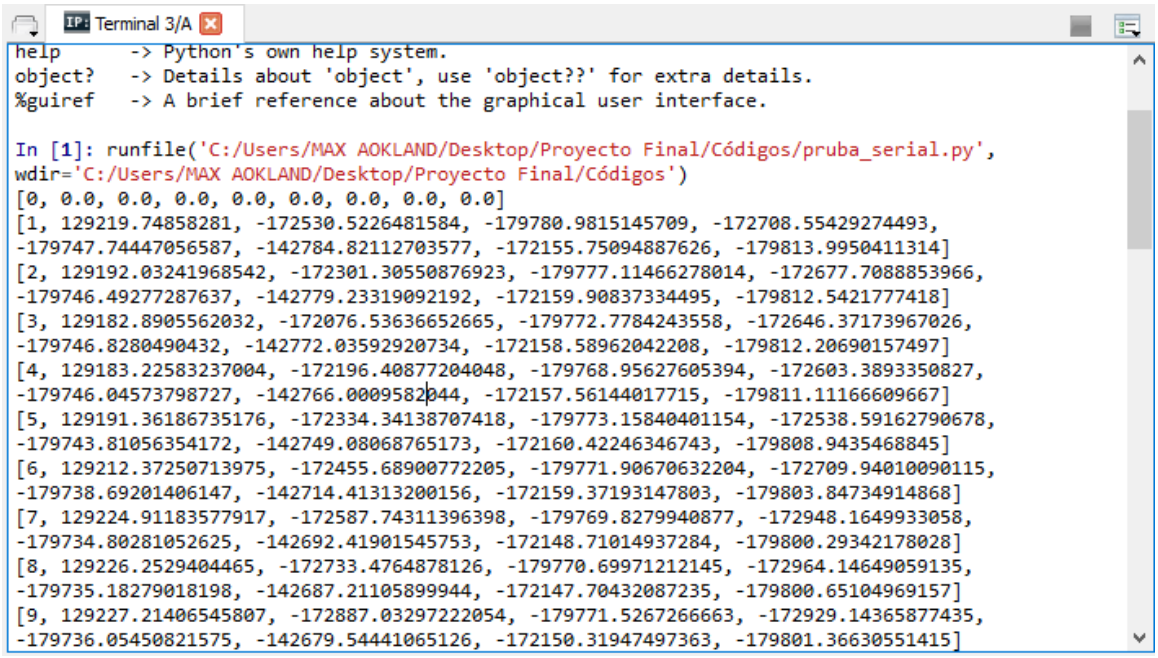


Figura 50: Nueva lectura de datos crudos desde Python

Dado que el puerto serial en Windows solo se puede comunicar con un software a la vez no se puede hacer una comparación directa utilizando los tres programas de manera simultánea. Por esta razón, se decidió que para analizar los datos de una manera más efectiva, era necesario tener un registro de los datos que se obtenían desde Python. Para esta tarea lo más simple era almacenar los resultados conforme se fueran obteniendo en un archivo CSV para lo cual Python también contaba con una librería llamada csv. Con este nuevo registro fue más fácil comparar los valores que se obtenían al utilizar Users.py, la OpenBCI_GUI y el código de Python. Al correr cada opción se trató en la mayor medida posible que las condiciones al tomar las mediciones fueran las mismas, razón la cual las cada opción se utilizó de manera breve y se corrió una opción tras la otra. Bajo estas consideraciones se pudo observar que los resultados de cada canal rondaban los mismos valores en las tres opciones, evidenciando que la toma de datos en el código era correcta.

9.4. Simulación de funcionamiento

Para acoplar los objetivos ya cumplidos se quiso juntar los códigos de los pulsos binaurales junto a la lectura de datos. En las primeras pruebas se percató que al reproducir un audio la ejecución del resto del código se quedaba en pausa hasta terminar la reproducción del audio. Por esta razón se exploró utilizar una programación multihilos, la cual permite hacer ejecuciones prácticamente en paralelo de ciertos segmentos deseados de código con el propósito de agilizar la ejecución total de un proceso. Para llevar esto a cabo se importó la librería Thread y se utilizó para que la reproducción del audio se ejecutara en un hilo secundario, en otras palabras, se ejecutara de forma separada a la toma de datos, el cual sería el hilo principal del código. De esta forma la toma de datos no se vería afectada cada vez que se tuviera que reproducir un audio.

Con el objetivo de emular lo que sería el funcionamiento del sistema, se configuró el código para que cada 100 muestras se reduzca la frecuencia de los pulsos en 3 Hz. También se configuró un retardo de tiempo de 1 micro segundo para que fuera más fácil observar las tramas de datos. En la sección de Anexos 16.1.1 se puede acceder a un vídeo de dicha simulación en donde también se muestra los audios que fueron generados y el archivo CSV con los datos crudos que fueron leídos. Con esta simulación se confirmó que era mejor tener un conjunto de audios previamente generados en lugar de generarlos conforme se fueran obteniendo los datos. Esta decisión fue tomada principalmente porque la cantidad de archivos de audio generados sería demasiado grande si se tomaran datos durante toda la noche. Además, se generarían muchos archivos redundantes cuando la persona se encuentre en una misma etapa de sueño.

CAPÍTULO 10

Procesamiento Digital de Señales (DSP)

10.1. Determinación del procesamiento a realizar

Con la obtención correcta de los datos crudos el siguiente paso era poder filtrarlos para obtener las mismas señales que se observan gráficamente con la OpenBCI_GUI. Para llevar a cabo esta tarea se hizo uso de la librería *signal* de scipy, la cual es la librería más recomendada para hacer un procesamiento de señales digital con Python. Se consultó nuevamente en los foros de ayuda de OpenBCI, este vez para saber cual era el procesamiento que se requería para obtener las señales que se obtenían desde su interfaz y se obtuvo como respuesta que era necesario la implementación de algunos filtros digitales.

El primero y que era el más indispensable de realizar era un filtro que eliminara un desface DC que posee el ADS1299 del la Cyton Board y que es bastante común en los amplificadores de ondas EEG. Este desfase, que puede ser tanto negativo como positivo, se sobrepone a las variaciones de voltaje en (uV) que detecta la Cyton y provoca que los valores sean sumamente elevados, como se pudo ver en los datos de la sección anterior. Este tipo de desfase ocurre a bajas frecuencias por que en el foro se recomendó utilizar cualquier tipo de filtro capaz de atenuar todas las frecuencias menores a 0.5 Hz y así remover el desfase de los datos.

Además, también se debía hacer un filtro pasa bandas que restringiera el rango de frecuencias que se querían monitorear. De esta forma se eliminaría el efecto de frecuencias no deseadas y se minimizaría el impacto de los los artefactos, los cuales son cualquier señal de origen no cerebral que perturba las señales cerebrales, como por ejemplo movimientos musculares de los ojos o mandíbula. Finalmente, también se recomendó como buena práctica realizar un filtro rechaza banda que eliminara el ruido generado por la corriente alterna que circula a 60 Hz en cualquier aparato que este conectado a la red eléctrica. Ya con estos conocimientos se podía proseguir a utilizar la librería para el diseño de los filtros.

10.2. Utilización de librería para diseño y uso de filtros

En la librería *signal* había múltiples opciones para crear cualquier tipo de filtro por lo que había que decidir de que tipos de iban a utilizar. La primera elección fue que se harían filtros *Butterworth* debido a que su respuesta en frecuencia no tiene ondulaciones en su banda de paso ni en la de rechazo, además, se puede obtener una banda de transición estrecha sin comprometer la forma plana de sus bandas. Posteriormente se eligió un filtro IIR sobre el filtro FIR, ya que aunque con ambos se pueda llegar al mismo resultado, los filtros IIR permiten obtener respuestas con transiciones muy buenas con funciones de transferencias más simples y que son menos demandantes computacionalmente. Finalmente, se eligió que los filtros fueran de segundo orden para obtener una banda de transición lo suficientemente lisa y estrecha y no demandar más poder computacional del requerido.

Se utilizó la función *butter* dentro de la librería *signal* para obtener los coeficientes, tanto del denominador como del numerador, de la función de transferencia que describe el filtro. Esta función es bastante útil porque ahorra el proceso de tener que calcular los coeficientes a mano. Para utilizar esta función simplemente se tenían que ingresar como parámetros el orden del filtro, las frecuencias de corte en radianes/muestra donde la magnitud se atenuaría en -3dB y que tipo de filtro se quería.

Se decidió que se harían dos filtros de 2ndo orden, primero un pasa banda con frecuencias de corte en 0.5 y 50 Hz para eliminar el desfase DC y delimitar las frecuencias con las que se quería trabajar, ya que el sistema es una aplicación para leer las ondas cerebrales durante el sueño nunca se vería una frecuencia mayor a los 50 Hz. Y un segundo filtro rechaza banda para eliminar completamente el ruido de la red eléctrica con frecuencias de corte 59 y 61 Hz para atenuar los 60 Hz. Para poder verificar que los filtros efectivamente fueran como se diseñaron se hizo un pequeño código para obtener la respuesta en frecuencia del filtro en base a los coeficientes previamente calculados y la frecuencia de muestreo del sistema, la cual para el caso de la Cyton es de 250 Hz. Los resultados se graficaron con la ayuda de la librería matplotlib y se pueden observar en la Figura 51.

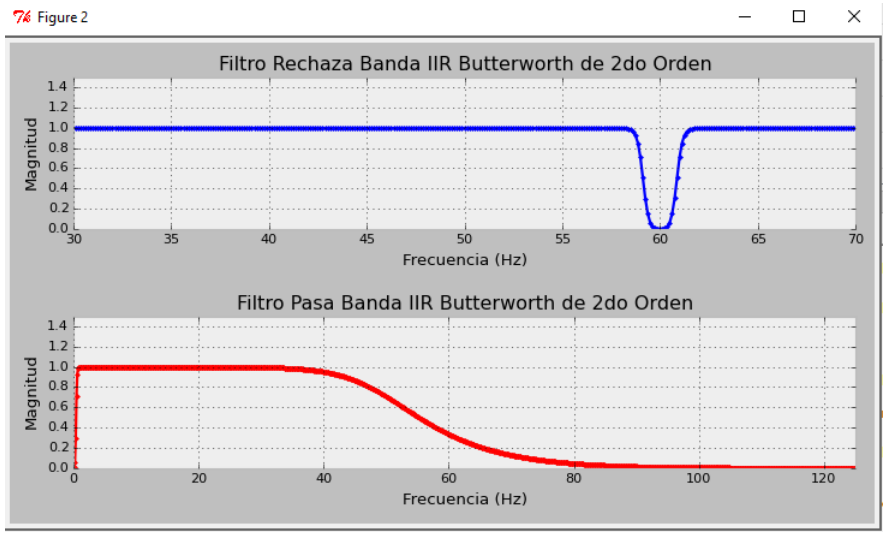


Figura 51: Respuesta en frecuencia de los filtros diseñados

Para verificar el funcionamiento de los filtros se hizo una pequeña prueba en la que se creo una señal sinusoidal de 1000 Hz y se le sumo como ruido una señal sinusoidal de 50 Hz, dicha señal puede observarse en la parte izquierda de la Figura 52 de color rojo. También de color rojo en parte inferior izquierda se puede observar el espectro de esa señal donde claramente se ven las dos frecuencias que la componen. Para aplicar el filtro se utilizó la función *lfilter* también de la librería *signal* la cual se encarga de filtrar los datos de una señal implementando la ecuación de diferencias que se obtiene a partir de los coeficientes calculados con la función *butter*.

De esta forma se diseñó un filtro pasa banda con frecuencias de corte en 950 y 1050 Hz con la función *butter* y con la función *lfilter* se le aplicó a la señal con ruido. Los resultados del filtrado se ven del lado derecho de la Figura 52 de color azul, donde se puede ver tanto en la señal como en el espectro, que efectivamente se removió completamente el ruido. Ya con una idea clara de como diseñar un filtro y aplicarlo a una señal, se podía proseguir a utilizar los filtros diseñados previamente para tratar de aplicarlos al flujo de datos que se obtuvo en el capítulo anterior.

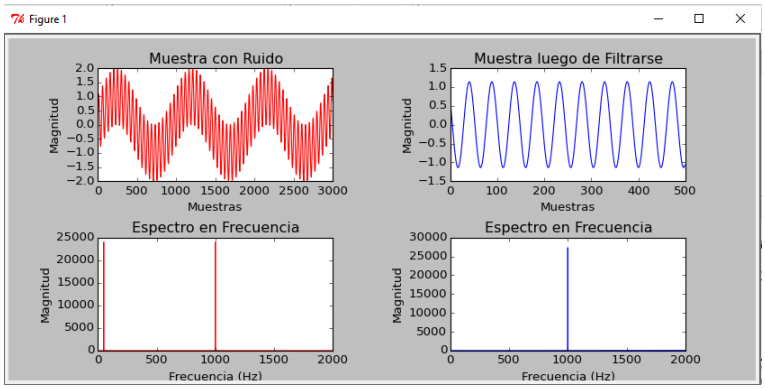


Figura 52: Filtrado de una señal con ruido

10.3. Primeras pruebas para filtrar datos en tiempo real

Para el desarrollo de esta sección se invirtió un tiempo importante aprendiendo a dominar las herramientas de la librería *matplotlib* para poder graficar datos en tiempo real. Ya que a pesar de que no se planeaba hacer una interfaz gráfica y que la idea detrás de detectar automáticamente las etapas del sueño era que no hubiera un especialista monitorizando las ondas, la mejor forma de visualizar el resultado de los filtros, considerando la gran cantidad de datos, era de forma gráfica. Cuando se dominó el aspecto de las gráficas se determinó que para obtener la mayor velocidad de actualización posible, lo mejor era solamente modificar los datos de la línea a graficar en lugar de crear una nueva figura para cada iteración.

A continuación se realizaron las primeras pruebas para filtrar directamente el flujo de datos leídos de la Cyton Board. Para empezar se probó que mientras se estuvieran leyendo los datos se fuera llenando un búfer de izquierda a derecha y al llenarse se enviaran los datos al filtro. Posteriormente se rotarían los valores del búfer una casilla a la izquierda para descartar el valor más antiguo, es decir el primer valor que entró al búfer, y adquirir un nuevo

dato que tomara lugar en la primera casilla del búfer de derecha a izquierda. Por último, los datos se filtrarían nuevamente. El procedimiento detrás de esta idea se comprende mejor con el diagrama de la Figura 53.

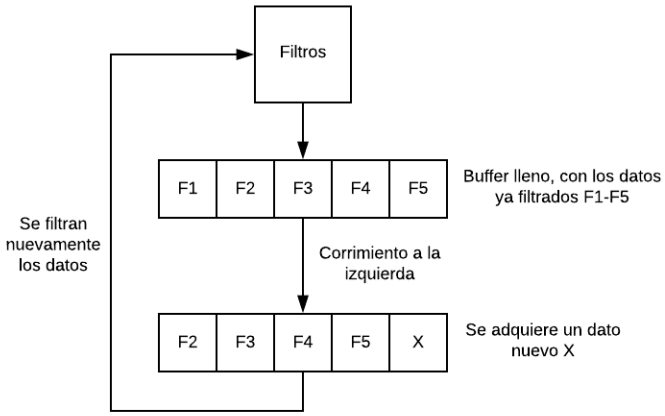


Figura 53: Primera prueba de filtrado en tiempo real

Para esta prueba solamente se estaban leyendo dos canales de la Cyton, los cuales estaban conectados a los electrodos frontales del casco Ultracortex Mark IV, se prefirió utilizar el casco para estas pruebas ya que era más fácil de quitar y poner ya que no había necesidad de aplicar gel a los electrodos. Esta primera prueba de filtrar los datos resultó fallida. Gráficamente ambos canales se comportaban de manera extraña y después de unos segundos las señalas se quedaban estancadas como en la Figura 54 sin importar que se hicieran movimientos para tratar de alterar las señales. Evidentemente se estaba haciendo mal y se hicieron pruebas cambiando el tamaño del búfer pero no hubo cambio significativos en la respuesta.

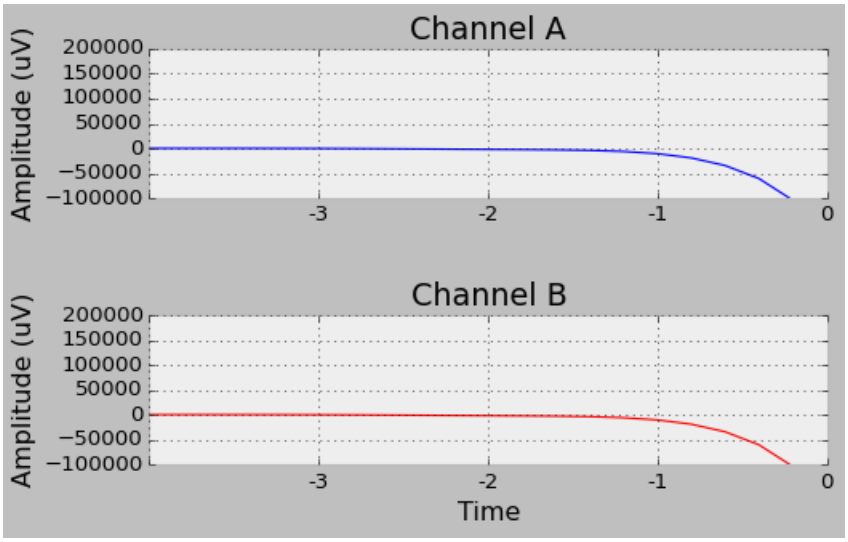


Figura 54: Prueba fallida de filtrado en tiempo real

10.4. EEGLAB

Durante el proceso de poder realizar el filtrado correcto del flujo de datos se tuvo una cita con el Dr. Luis Alejandro López, quién como parte de su trabajo es especialista en trastornos de sueño y trabaja en el Hospital El Pilar en la Ciudad de Guatemala. En esta cita se le presentó la idea del proyecto y pareció muy interesado en el potencial que tendría poder combatir trastornos de sueño sin la necesidad de medicamentos. De esta primera cita el doctor destacó que antes de seguir con cualquier otra sección del proyecto, era importante tener un registro visual de los datos para que un especialista como él los pudiera ver, analizar y etiquetar apropiadamente para crear un hipnograma del sueño como en la Figura 2. Ya que hasta ahora solo se obtenía un registro numérico de los datos crudos a través de archivos CSV, ahora se debía tener una forma de fácil de visualizar los datos capturados por la Cyton.

Esta sección del proyecto era importante dado que para futuras fases del proyecto, el análisis y comparación de hipnogramas sería una de las principales fuentes de validación del funcionamiento de un sistema final y los hipnogramas no se podían llegar a obtener sin la visualización gráfica de los datos que se procesaron durante la noche. Para llevar esta tarea a cabo se evaluaron las opciones de OpenBCI que servían para hacer un procesamiento posterior a la captura de datos, en otras palabras un procesamiento fuera de línea u *offline*. Dentro de las opciones se eligió el *toolbox* EEGLAB, la cual es una herramienta interactiva de Matlab para procesar datos continuos de señales eléctricas del cuerpo humano, por las facilidades y flexibilidad que presentaba a la hora de graficar grandes cantidades de datos.

Se realizó una pequeña grabación de la toma de datos a través de OpenBCI para servir como medida de comparación en las siguientes pruebas. Para la grabación se utilizaron los 7 canales disponibles del casco Ultracortex Mark IV y se puede ver en la sección 16.1.2. Los datos crudos del canal 3 de la grabación fueron exportados a matlab para poder ser utilizados con la interfaz de EEGLAB. Ya sabiendo utilizar la herramienta se hicieron todas las configuraciones pertinentes y se probó filtrar los datos con los filtros de EEGLAB para luego graficarlos. Los resultados se pueden ver en la Figura 55, la idea era ajustar la amplitud a 200 microvoltios y la ventana de tiempo de EEGLAB a 5 segundos para coincidir con la interfaz de OpenBCI_GUI y tener resultados comparables. En la Figura 55 se estarían viendo la primeros 5 segundos del vídeo 16.1.2 y evidentemente los resultados no coinciden.

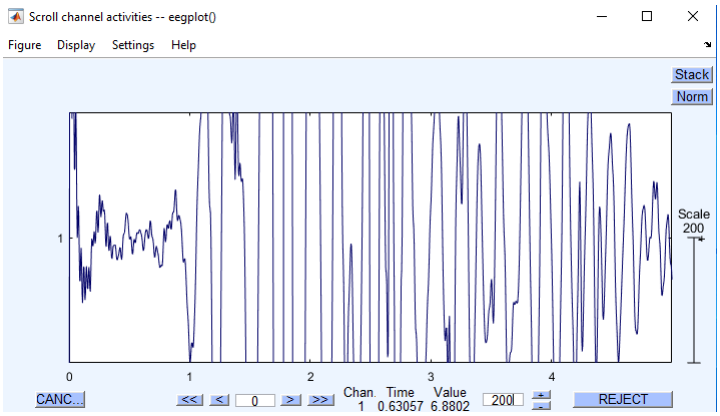


Figura 55: Señal registrada y filtrada en EEGLAB

Como una alternativa se creó un código que leyera los datos del archivo CSV del vídeo y los filtrara con los filtros diseñados anteriormente, la única modificación que se hizo fue cambiar el pasa bandas a una banda entre 5 Hz y 50 Hz dado que se esa era el filtro que utilizaba la OpenBCI_GUI. A diferencia de lo que se había estado probando, ahora no se estaba filtrando en tiempo real si no que se tomaba la señal entera registrada en el CSV para ingresarla directamente a la función lfilter. Los resultados de filtrar los ocho canales fueron almacenados en otro CSV para luego poder importarlos a Matlab. Dado que habilitando los 8 canales simultáneamente era más complicado observar las señales a detalle, nuevamente se importó solamente el canal 3 del vídeo 16.1.2 a EEGLAB. En un principio parecía que la prueba había fracaso, sin embargo, al reducir la figura a un tamaño similar al que se ve en la Open_BCI, bajo la misma escala y ventana de tiempo, se obtuvo exactamente la misma señal que en el vídeo de prueba 16.1.2. Se recorrió toda la señal a la largo del tiempo con la ventana de EEGLAB comparando los tiempos en que ocurrían ciertos eventos con el tiempo que ocurrían en el vídeo y el resultado siempre fue idéntico. Algunas capturas de esta prueba se pueden observar en la Figura 56 y en la Figura 57.

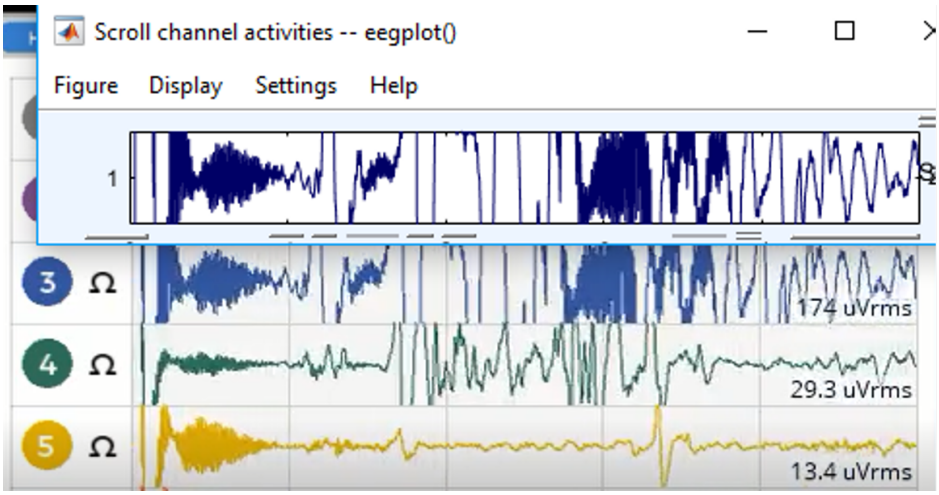


Figura 56: Señal registrada en EEGLAB comparada con la señal de la OpenBCI_GUI

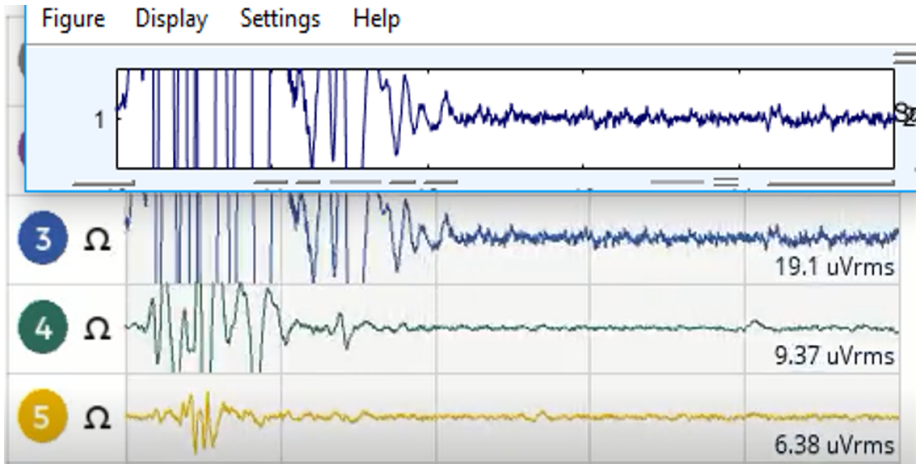


Figura 57: Señal registrada en EEGLAB comparada con la señal de la OpenBCI_GUI en otro instante de tiempo

Se hicieron pruebas con otros canales individualmente y el resultado fue igual de positivo, luego se prosiguió a juntar los ocho canales simultáneamente y el resultado se observa en la Figura 58. Ya que utilizando los mismos filtros que se habían diseñado previamente se pudo recrear exactamente la misma señal de la OpenBCI_GUI, todo indicaba que para el filtrado en tiempo real también era posible obtener los mismos resultados.

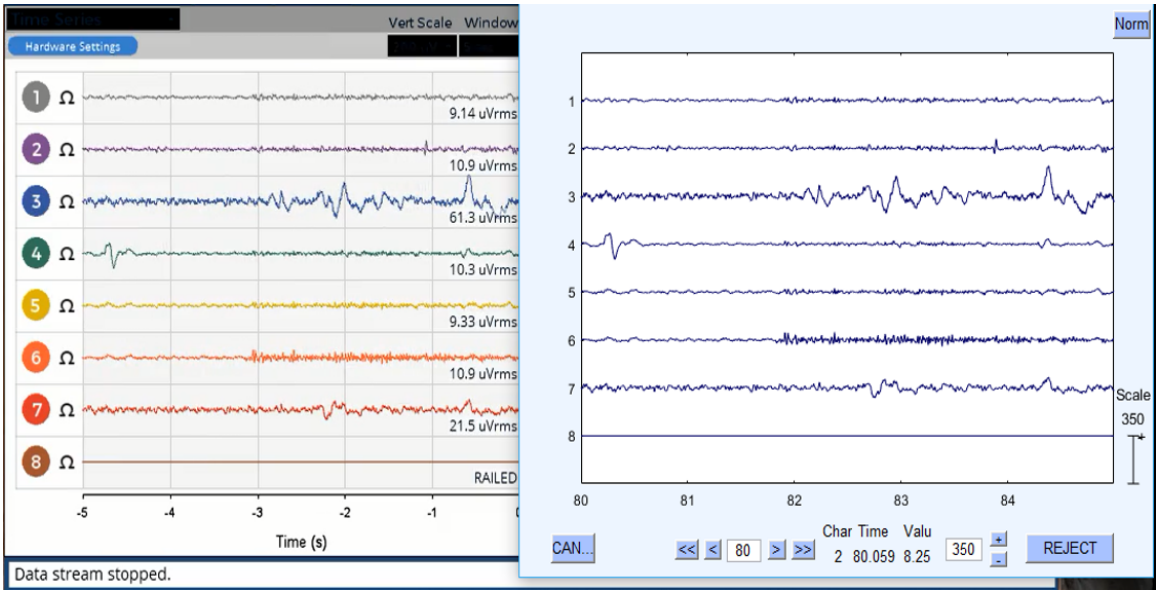


Figura 58: Registro de ocho canales desde EEGLAB

10.5. Siguientes pruebas de filtrado en tiempo real

Dado que OpenBCI es una empresa de código abierto además de la interfaz que tienen para Windows, MAC y Linux también cuentan con una interfaz hecha en Processing programada en C++, abierta para que cualquier persona que quiera pueda hacer sus modificaciones. Se instaló dicha interfaz para encontrar dentro del código como estaban haciendo la parte del filtrado y así poder replicar sus resultados. Dentro de todas las opciones que tiene la interfaz se observó que los filtros que se estaban utilizando eran un pasa bandas y un rechaza bandas con exactamente las mismas frecuencias de corte. Además, los coeficientes que devolvía la función *butter* eran los mismos que los coeficientes que ellos tenían ingresados, por lo que todo era cuestión de como utilizar filtros correctamente conforme se fueran leyendo los datos.

Después de mucho analizar el conjunto de códigos, se logró determinar que estaban utilizando dos búfers, uno pequeño de 30 casillas y uno más grande de 5000 casillas. La idea era que el búfer pequeño recibiera los datos crudos y al llenarse se trasladaran los datos al búfer más grande haciendo un corrimiento a la izquierda del tamaño del búfer pequeño. El búfer grande con los datos crudos sería el que se pasara por ambos filtros y el resultado de dicho filtrado es el que se desplegarían gráficamente en la interfaz, dicho proceso se comprende mejor con la Figura 59.

Se concluyó que el problema del enfoque anterior es que se estaban filtrando datos que ya habían sido filtrados, en cambio con este nuevo enfoque, siempre se filtran datos crudos. Con el fin de poder validar esta implementación lo que se hizo fue leer un canal del archivo CSV del vídeo de muestra 16.1.2 y tomar los datos con una frecuencia de 250 Hz, de esta forma se estaría emulando una lectura de datos en vivo desde la Cyton pero pudiendo comparar los resultados contra el vídeo 16.1.2. Los resultados de esta simulación fueron exitosos y se pueden apreciar en el siguiente vídeo 16.1.3. Como una comparación adicional también se almacenaron los datos filtrados en un archivo CSV y se compararon con los datos que fueron filtrados ingresando como entrada la señal entera del CSV con datos crudos. Dado que los datos eran exactamente los mismos se concluyó que el filtrado en tiempo real, bajo este nueva implementación, funcionaba correctamente.

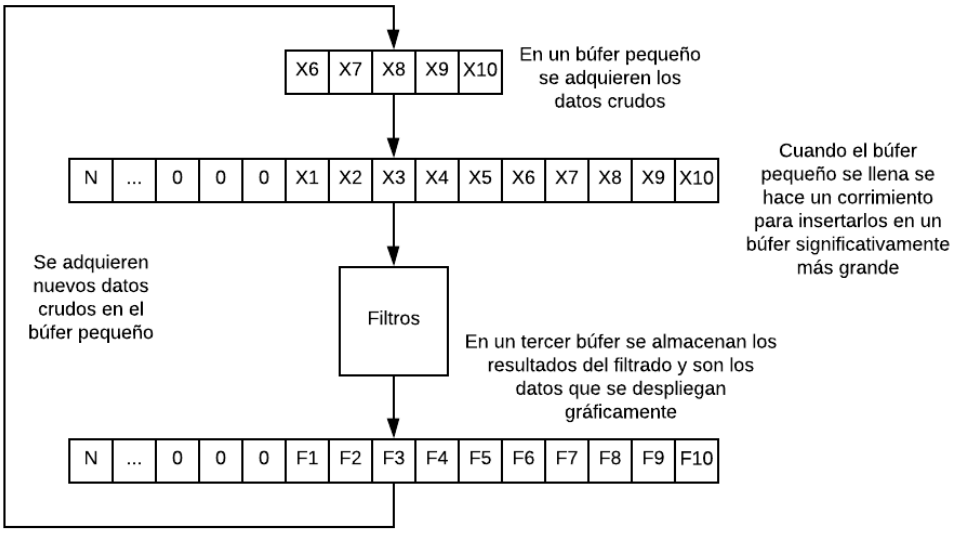


Figura 59: Nueva implementación de filtrado en tiempo real

CAPÍTULO 11

Clasificación automática de las etapas del ciclo del sueño

11.1. Lectura de trabajos relacionados

Ya que no se contaba con experiencia previa con temas relacionados a la clasificación de señales por medio de métodos basados en *Machine Learning*, ni de cómo estos eran aplicados en la detección de las etapas del sueño, la primera meta era informarse ampliamente sobre el tema. Los trabajos encontrados en [43], [7], [44] fueron referidos por el Dr. Luis Alberto Rivera, quién tiene una gran experiencia en *Machine Learning*, y precisamente sirvieron como base para comenzar a comprender cómo realizar un sistema de detección automática de las etapas del sueño.

En estos trabajos el objetivo era lograr una clasificación exitosa de las etapas del sueño utilizando un solo canal de EEG con el propósito de que el sistema asistiera a los especialistas en la tediosa y costosa tarea de inspeccionar todas las señales recolectadas en un estudio de sueño para detectar algún trastorno. La idea principal de estos sistemas es hacer el proceso de clasificación más rápido y analíticamente más acertado, dado que la clasificación manual está sujeta al error humano, existiendo incluso desacuerdos dentro de los mismos especialistas. En estos trabajos proponen utilizar un solo canal de EEG para que el sistema sea lo menos invasivo posible y que el sueño del paciente sea lo más eficiente posible.

De los trabajos mencionados, la investigación en [7] fue la que se utilizó como base principal para el desarrollo del sistema propio de clasificación debido a que además de explicar a detalle cómo es el proceso para crear este tipo de sistema y proponer una metodología innovadora para el diseño de uno, también hacen una amplia recopilación de distintos sistemas de detección automática de las etapas del sueño que se han publicado en los últimos años. En esta recopilación se mencionan todas las distintas técnicas que se utilizaron tanto en la etapa de extracción de características como en el uso de clasificadores, mostrando los resultados de acierto de todas estas variaciones y combinaciones.

11.2. Base de datos Sleep-EDF

Como punto de partida se decidió utilizar una base de datos pública que ha sido ampliamente utilizada en la mayoría de investigaciones relacionadas a la clasificación de las etapas del sueño, tal como se menciona en [7]. Esta base datos que se encuentra en este enlace es propiedad de PhysioNet y son quienes se han encargado de actualizarla constantemente. Hoy en día se pueden encontrar hasta 197 polisomnografías completas junto a sus hipnogramas respectivos, los cuales fueron manualmente anotados por especialistas en base a las guías de Rechtschaffen y Kales (R&K) de 1968. Para hacer las anotaciones los datos se dividieron en segmentos o épocas de 30 segundos, en donde se etiqueta claramente a qué etapa del sueño corresponde cada época.

Dentro de las 197 polisomnografías las primeras 153 eran las de principal interés dado que en el resto de grabaciones los sujetos de estudio tomaron Temazepam para analizar sus efectos en el sueño. En las polisomnografías de interés se hicieron grabaciones de alrededor de 20 horas para cada sujeto, en donde se registra la información de canales relevantes para poder interpretar el estado de la persona. La mayoría de polisomnografías contiene los datos de dos canales EEG en las posiciones Fpz-Cz y Pz-Oz, un canal de EMG en el mentón, un canal EOG con ubicación horizontal y en algunos otros casos también se incluyen datos de la respiración oronasal y temperatura corporal. Cabe mencionar que la frecuencia de muestreo para los canales EEG y EOG fue de 100 Hz mientras que para el resto de señales fue de 1 Hz.

Tanto las polisomnografías como los hipnogramas estaban almacenados en un formato .EDF lo cual es un formato común para estudios de sueño, sin embargo, para poder visualizar las grabaciones era necesario el uso de alguna una herramienta adicional, en este caso se instaló un visualizador de .EDFs llamado Polyman. Luego de aprender a utilizar esta herramienta se pudieron abrir y explorar distintos archivos de la base de datos. En la parte superior de la Figura 60 se ven los canales principales de la polisomnografía SC4001EO y en la parte inferior de color rojo se muestra la etiqueta del hipnograma que indica a que etapa del sueño corresponde esa época de 30 segundos.

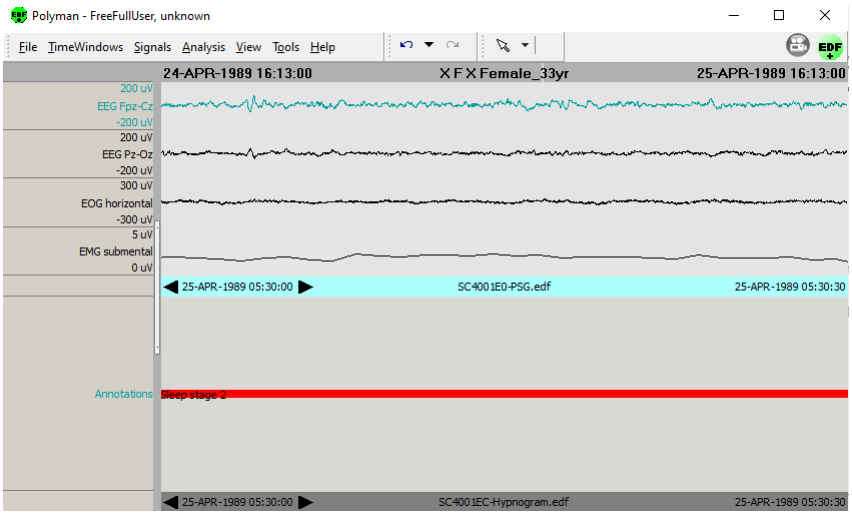


Figura 60: Visualización de polisomnografía SC4001E0 desde Polyman

11.3. Sleep EDFx Toolbox

Aunque ya se pudieran visualizar las grabaciones y anotaciones de los archivos de la base de datos, para poder llegar a hacer un sistema de clasificación era necesario poder acceder a los datos y a las etiquetas. Ya que por medio de Polyman y otros aplicaciones que trabajan con formato .EDF no fue posible realizar esta tarea, se buscaron alternativas y se encontró el trabajo presentado en [45]. En esta publicación se habla sobre el desarrollo de una herramienta para Matlab especialmente diseñada para trabajar con la base de datos de PhysioNet con el fin de facilitar su uso y estandarizar la forma en la que se toman los datos. De esta forma los resultados de todas las investigaciones relacionadas a algoritmos clasificadores del sueño que utilicen esta base de datos pueden ser comparables.

Aunque esta herramienta parecía ser exactamente lo que se necesitaba, en una primera instancia al instalar la herramienta en Matlab y tratar de utilizarla, las funciones descritas en [45] no devolvían nada. Explorando más a fondo los códigos de cada función se percató que posiblemente los fallos en el funcionamiento eran debido a que la página de PhysioNet había sido actualizada y los archivos ya no se encontraban de la misma manera que cuando se creó la herramienta. De esta forma se hizo una depuración de las funciones línea por línea para que finalmente la herramienta pudiera funcionar en la versión de Matlab 2017a y con el formato actual en el que se encuentra la base de datos. A continuación se describe brevemente qué se obtuvo de cada función de esta herramienta. Para obtener más detalles de las funciones se recomienda leer el trabajo en [45].

- **loadEDFxData():** Con esta función se descargaban automáticamente las polisomnografías dentro de una carpeta con el mismo nombre que la polisomnografía. Todas las carpetas se creaban dentro de un directorio que se especificaba al utilizar la función. Se configuró el código para descargar los primeros diez archivos de los sujetos que no hicieron pruebas con Temazepan.
- **convertEDFxToMat():** Esta función fue muy importante ya a partir del archivo de la polisomnografía en formato .EDF obtenía todos los datos numéricos de los canales disponibles en formato .MAT, el cual es el tipo de formato estándar que utiliza Matlab para cargar o guardar variables y otras estructuras. Para esta conversión era necesario tener EEGLAB, el cual ya se había instalado para la sección anterior y un *toolbox* llamado biosig.
- **downloadEDFxAnnotations():** A partir de correr esta función se descargaron los hipnogramas respectivos a cada polisomnografía descargada. Además de las anotaciones también se descargan archivos con información importante como la hora a la que el sujeto apagó las luces, la hora en que las encendió, las horas que a las comenzaron y terminaron las grabaciones para cada sujeto, etc.
- **processEDFxHypnogram():** Ya con las anotaciones descargadas este código permitía abstraer del hipnograma un vector unidimensional con todas las etiquetas que fueron marcadas por los especialistas. En este vector se identificaba con la letra 'W' a todas las épocas de 30 segundos que fueron clasificadas como que la persona estaba despierta. Los números 1-4 se identificaban a las siguientes cuatro etapas mientras que la letra R lo hacía para el sueño REM. Este vector también era almacenado en una variable .MAT para poder utilizarlo fácilmente desde Matlab.

- **loadEDFx():** Por último, dado que de las aproximadamente 20 horas de grabación por sujeto solo se estaba interesado en la parte en la que duermen, con esta función lo que se hacía era recortar tanto los datos de los canales como el vector del hipnograma a la porción de la grabación donde la persona estaba durmiendo. Para hacer esto recomendaban como forma de estandarización comenzar a tomar las etiquetas a partir del momento en que el sujeto de estudio apagó las luces y terminar al momento en el que las encendió. Esta función también permitía convertir el hipnograma anotado en base a las guías del sueño de R&K a las guías de sueño AASM. Para este caso se prefirió hacer el cambió a las guías AASM ya que son más actuales y tienen menos etapas, lo cual ayudaría a obtener mejores resultados en los algoritmos clasificadores.

11.4. Extracción de características

Ya con los datos limitados a solo la porción de la noche se podía proseguir a extraer características esenciales de las señales que permitieran entrenar un algoritmo clasificador a identificar cada señal. Los pasos a grandes rasgos que se siguieron para realizar este proceso se muestran en la Figura 61. Cabe mencionar que en esta sección del documento solo se detallará sobre el proceso de extracción mientras que la recopilación de todas las pruebas y resultados obtenidos se describen en la sección 11.6.

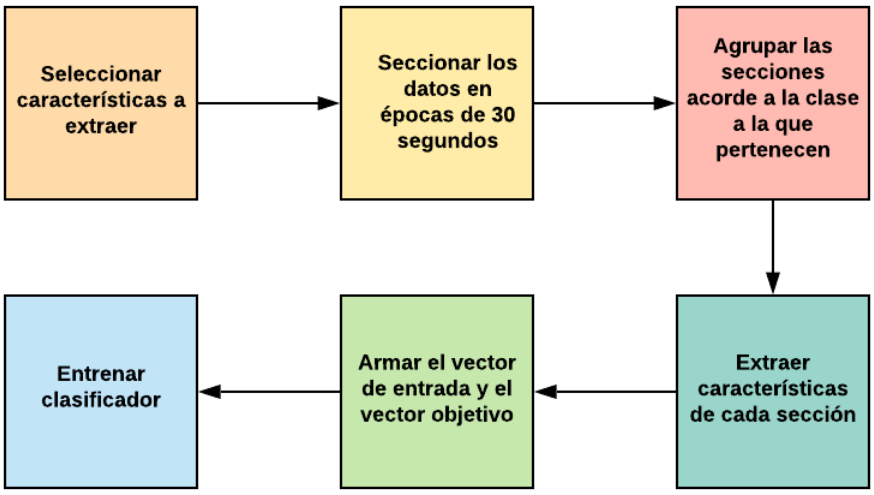


Figura 61: Proceso para extraer características de la base de datos

Tal como se indica en la Figura 61 el primer paso para la extracción era decidir que tipo de características podían aportar valor a la hora de identificar la señal. Cabe mencionar que la de selección de características puede llegar a ser un proceso sumamente complejo que puede llegar a tomar mucho tiempo dado a la gran variedad variantes y combinaciones de las que se puede elegir. Para seleccionar las características más adecuadas para entrenar un clasificador se debe de estudiar a detalle la naturaleza de las señales de interés y la cantidad de datos disponibles.

En la mayoría de casos es recomendable comenzar con una idea base e ir haciendo pruebas para ver que combinaciones están acercando al modelo a los porcentajes de acierto deseados. En este caso se contaba con el gran aporte de la recopilación de estudios que se encuentran en [7], en donde se muestra que para este tipo de clasificadores se han usado características en el tiempo, en frecuencia, combinando tiempo y frecuencia y características no lineales. Dado que los estudios que utilizaron características basadas en el tiempo presentaron buenos resultados, incluyendo los dos nuevos tipos de características que se presentaron en [7], se decidió que se iba a concentrar en este tipo de características a la espera de ver resultados.

Para comenzar a realizar pruebas lo antes posible la primera característica que se decidió implementar fue el *Mean Absolute Value* o MAV. Para obtener esta característica simplemente se calcula el valor absoluto de una señal y luego se saca su promedio. Como segunda característica se utilizó una función que calcula el *Zero Crossing* o ZC de una señal, en donde básicamente lo que hace es contar el número de veces que una señal atraviesa el cero. Esto se hace contando cada vez que hay un cambio de signo en los datos pero solamente cuando se supera un umbral que se ingresa a la función como parámetro.

La idea de ingresar un umbral en esta función es minimizar el efecto del ruido en la cuenta de cruzamientos por cero. Ya con estas dos características se esperaban resultados que fueran capaces de identificar las muestras según la etapa a la que corresponden de una manera satisfactoria. Ya que por ejemplo, apoyándonos en el Cuadro 2, se esperaría que una señal de una persona despierta sea de alta frecuencia pero de baja amplitud. Para esta señal el MAV no debería ser muy grande pero sí contar con un ZC elevado. En el caso contrario, una onda perteneciente al sueño profundo tendría una mayor amplitud pero menor frecuencia por lo se esperaría un MAV más alto pero un ZC menor.

Como una característica adicional que podía aportar información valiosa para el clasificador, también se implementó la característica presentada en [7] llamada *Maximum Minimum Distance* o MMD. Lo que hace esta función es calcular la distancia, como su nombre lo indica, entre el punto máximo de la señal contra el punto mínimo, para esta característica toda una muestra es divida en subsecciones y se calcula el MMD para cada subsección para que la característica final de la señal completa sea la suma del MMD de cada subsección. En la Figura 62 se puede apreciar mejor la idea detrás de esta característica.

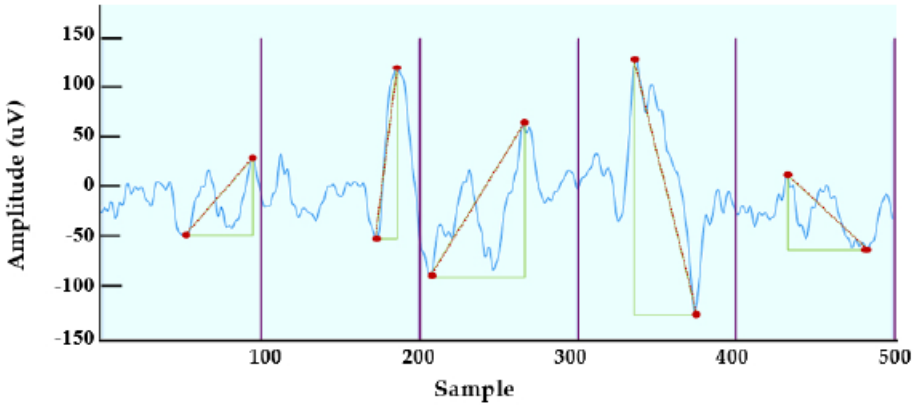


Figura 62: Característica Máximum Minimum Distance (MMD) [7]

Con las características listas los siguientes pasos de la Figura 61 son más directos. Debido a la gran cantidad de datos con los que se iba a trabajar y las numerosas pruebas que debían hacerse cambiando el número de clases, número de características y hasta el número de señales, el código de extracción de datos se hizo por medio de celdas en Matlab para que las modificaciones para cada prueba pudieran hacerse de una manera más dinámica.

Lo que se hizo a continuación fue tomar el vector unidimensional con los millones de datos de la señal y se segmentó en épocas de 30 segundos para que coincidieran con el vector de anotaciones. De esta forma se pasó de un vector unidimensional a una matriz con 3000 filas. Dado que el período muestreo es de 100 Hz, cada 3000 datos se captura una época de 30 segundos. El número de columnas de la matriz sería igual al número de etiquetas que tuviera el vector de anotaciones, en otras palabras, el número de épocas totales. El siguiente paso fue separar esta matriz en cinco matrices distintas, una para cada etapa del sueño según las guías AASM. En otras palabras, en cada matriz estarían todas las épocas que corresponden a una misma etapa por lo que el número de columnas de cada matriz corresponde al número de veces que se etiquetó esa etapa en el vector de anotaciones.

Posteriormente, se pasó a calcular las características de cada época de las cinco matrices, en donde los resultados de cada época se denominan vectores de características. Estos vectores iban siendo almacenados en una matriz, en donde cada columna correspondería a un vector de características. A partir de los resultados de la extracción se deben de generar los vectores de objetivos, los cuales se encargan de identificar cada vector de características según a la etapa del sueño a la que corresponden. Tanto el vector de características como el de objetivos se describen con mayor detalle en la siguiente sección. Finalmente, con ambos conjuntos de vectores ya estructurados, se prosiguió a entrenar los clasificadores.

11.5. Entrenamiento de clasificadores

De la recopilación en [7] también se pueden encontrar que se han utilizando una gran cantidad de distintos clasificadores para hacer la identificación de las etapas del ciclo del sueño. Para esta sección, se eligió utilizar una Red Neuronal Artificial o RNA y una Máquina de Vectores de Soporte mejor conocida una *Support Vector Machine* (SVM), dado sus buenos resultados en [7] y que se contaba con una mayor accesibilidad para utilizar estos clasificadores. Para el caso de la red neuronal se utilizó la *Neural Network App* de Matlab donde se tenía una interfaz para seleccionar los vectores con los que se entrenaría la red y seleccionar el porcentajes de datos que corresponderían para entrenar, validar y probar la red. Además, también contaba con distintas opciones para visualizar y analizar los resultados. La red neuronal que se implementa con esta interfaz se observa en la Figura 63.

Por otro lado para el SVM se utilizó un pequeño código que utilizaba la función fitcecoc de Matlab para entrenar el clasificador SVM y la función crossval, también de Matlab, para hacer una validación cruzada y obtener el rendimiento del modelo generado. Cabe mencionar que a la hora de utilizar los clasificadores cada etapa del sueño correspondería a una clase a identificar. Dado que se utilizaron las guías de AASM se tendrían cinco clases, donde la clase 1 correspondería a la etapa de "Despierto", las siguientes 3 clases representarían a las Etapas 1, 2 y 3, respectivamente y por último, la clase 5 correspondería a la etapa "REM".

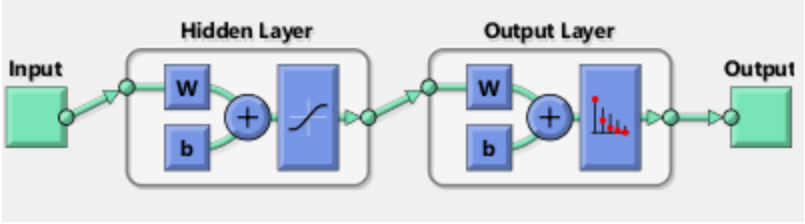


Figura 63: Red Neuronal de la Neural Network App

Para poder entrenar un clasificador a identificar patrones en una señal que nunca antes ha visto, es necesario ingresarle los vectores de características y vectores de objetivos de la porción de los datos que se asignarán para el entrenamiento. El rendimiento de la clasificación depende principalmente de la calidad y cantidad de las características que se hayan seleccionado y que componen la matriz de características. En esta matriz el número de filas corresponde al número de características que se quieran calcular, mientras que el número de columnas corresponde al número de épocas o muestras totales. En la matriz que se estructuró aparecen primero las características de todas las épocas de la etapa "Despierto", seguidas por todas las de la Etapa 1 2 y 3, hasta terminar con la etapa "REM". En la Figura 64 se puede ver un ejemplo de la matriz que contiene los vectores de características de una grabación, en donde la primera fila corresponde al resultado del MAV y la segunda al del ZC para cada época.

Input_Vector

2x838 double

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	13.7896	11.5370	14.5035	11.8348	13.4379	12.4544	12.2722	12.1854	11.4685	9.2049	11.9678	10.9987	9.3631
2	711	616	580	530	520	607	525	557	539	546	539	637	627

Figura 64: Ejemplo de vectores de características para la grabación "SC4002E0"

Posteriormente había que estructurar los vectores de objetivos para indicarle al clasificador a que etapa pertenece cada vector de características. Para el caso de la red neuronal estos vectores debían de estar compuestos solamente por unos y ceros, donde hubiera un número de filas igual al número de clases que se quieren clasificar, y donde se colocaría un número 1 en la fila que corresponde a la clase que pertenece un determinado vector de características. El resto de filas del vector se llenarían con ceros. En la Figura 65 se puede ver un ejemplo de la estructura de la matriz de objetivos que contiene todos estos vectores. Se puede apreciar que todos los número 1 están primera fila, indicando que todos los vectores de características que se encuentran en esas mismas posiciones pero en su respectiva matriz, pertenecen a la clase 1.

Aunque en este caso se haya ordenado la matriz de características para que aparecieran todas las épocas de cada clase juntas, el orden de esta matriz no influye en los resultados siempre y cuando en la matriz de objetivos se indique en la posición correcta a que clase pertenece cada vector de características. Sin embargo, sí resultaba más simple crear la matriz de objetivos agrupando los vectores por clase en lugar de tomarlos en el orden que se encontraban las anotaciones. Para el caso del SVM la matriz de objetivos es más simple y solamente se tiene que generar un vector unidimensional que identificara cada clase con etiquetas determinadas, por ejemplo, con números del 1 al 5 o con letras de la A a la E.

Target_Vector

5x837 double

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 65: Ejemplo de vectores de objetivos para cinco clases para la grabación "SC4002E0"

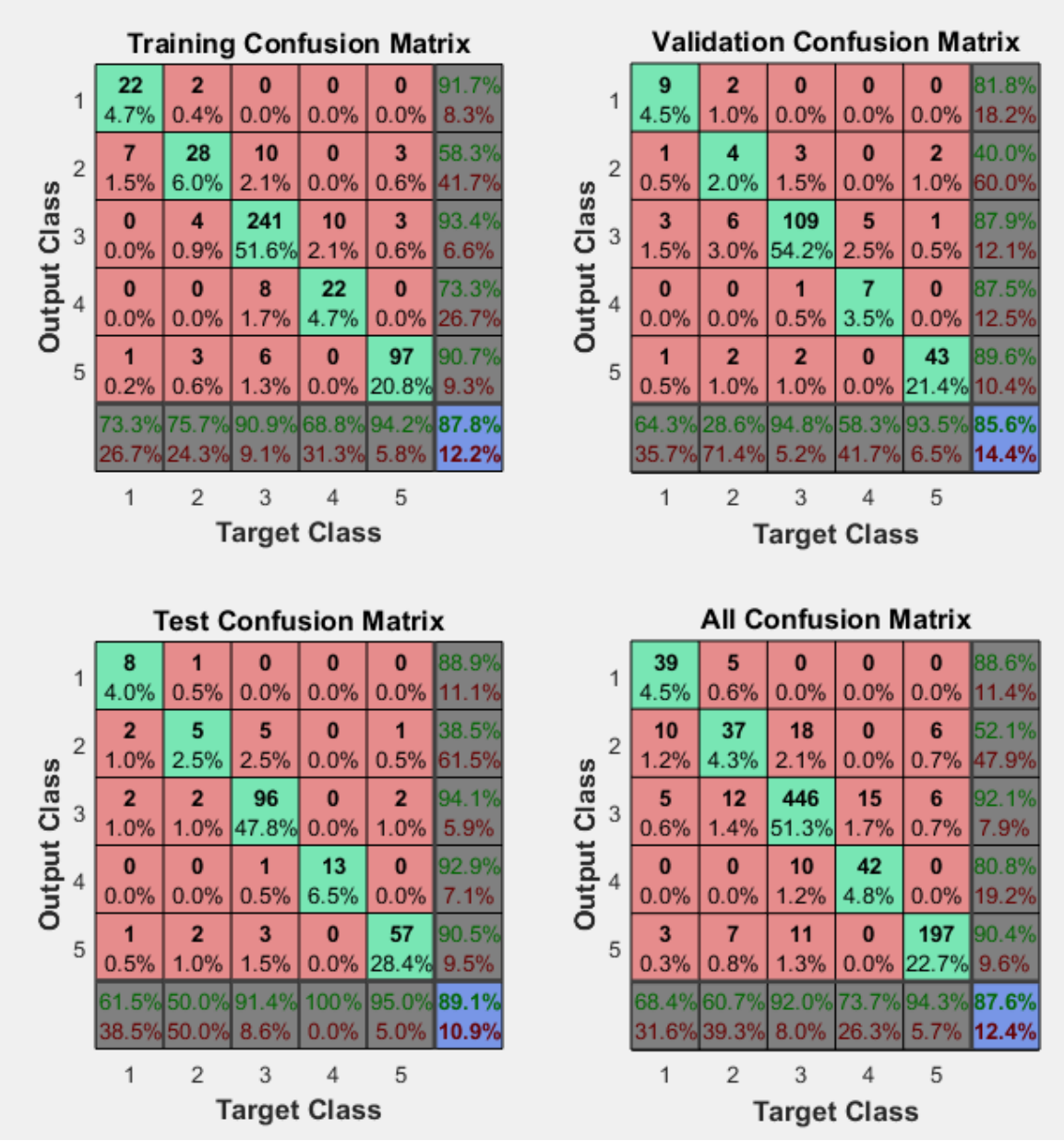


Figura 66: Ejemplo de matrices de confusión obtenidas de la Neural Network App

Para el caso de la red neuronal el rendimiento de la red generada se podía analizar mediante matrices de confusión. En estas matrices básicamente se indicaban los aciertos y errores a la hora clasificar cada clase. En la Figura 66 se puede ver un ejemplo de estas matrices. En la imagen están los resultados de la porción de los datos que se utilizaron para entrenar la red, validarla y probarla y en la última matriz del se recopila el resultado de acierto de todas las matrices. En estas matrices, las columnas son la clase real a la que pertenecían las épocas y las filas cómo realmente fueron clasificadas.

De esta forma la diagonal verde indica la cantidad de épocas que se encontraron de una clase determinada y que fueron correctamente clasificados como esa misma clase, mientras que el resto de cuadros rojos son épocas de una clase que fueron incorrectamente clasificadas como otra. Por ejemplo, enfocándonos en la última matriz, la red se topó con un total de 57 épocas que pertenecían a la clase 1, de estas 39 fueron correctamente clasificadas como clase 1, 10 incorrectamente clasificadas como clase 2, 5 incorrectamente clasificadas como clase 3 y 3 incorrectamente clasificadas como clase 5. Al final el resultado más importante era el porcentaje de acierto final del último cuadro azul, el cual también se obtenía al utilizar el clasificador SVM.

11.6. Pruebas y resultados

En esta sección se recompilan todos las pruebas, variaciones y combinaciones que se llevaron a cabo con el solo objetivo de obtener el clasificador con el mejor rendimiento posible. Cabe mencionar que dada la aleatoriedad que existe en los clasificadores para seleccionar los datos de entrenamiento, volver a entrenar un modelo con los mismos vectores da resultados que pueden variar para dar un mejor o peor resultado. Es por eso que es conveniente entrenar un modelo más de una vez y sacar promedios de los rendimientos para tener una mejor percepción de que tan altos porcentajes se pueden llegar a obtener con determinados vectores.

Las matrices de confusión fueron muy útiles para analizar que clases presentaban mayores problema, por ejemplo en la Figura 66 claramente se ve la clase 1 es la que peor porcentaje de acierto tiene, confundiéndose mucho entre la clase 3 y la clase 5. Aunque las matrices tuvieron un rol importante para el análisis de los resultados y toma de decisiones, para agilizar la realización de las pruebas y presentar cifras de manera compacta solo se concentró en los porcentajes de rendimiento totales. Para eso se creó un código que automáticamente entrenara los clasificadores con la red neuronal y el SVM recompilando los resultados en una matriz.

Para las primeras pruebas se comenzó probando solo con cuatro clases por lo que no se tomó en cuenta la clase 5 o etapa REM y tomando solo dos características, MAV y ZC. Por el momento solo se iba a estar utilizando el canal EEG “Fpz- Cz” para todas las pruebas. Los modelos fueron entrenados un total de tres veces para cada una de las grabaciones con los dos clasificadores. Los resultados que se pueden observar en la Figura 67 son el rendimiento promedio que se obtuvo de los tres entrenamientos que se hicieron por clasificador. En la parte inferior de la gráfica lo que se muestra es el nombre de la grabación, tal como las identifican dentro de la base de datos de PhysioNet, a la que corresponden los resultados.

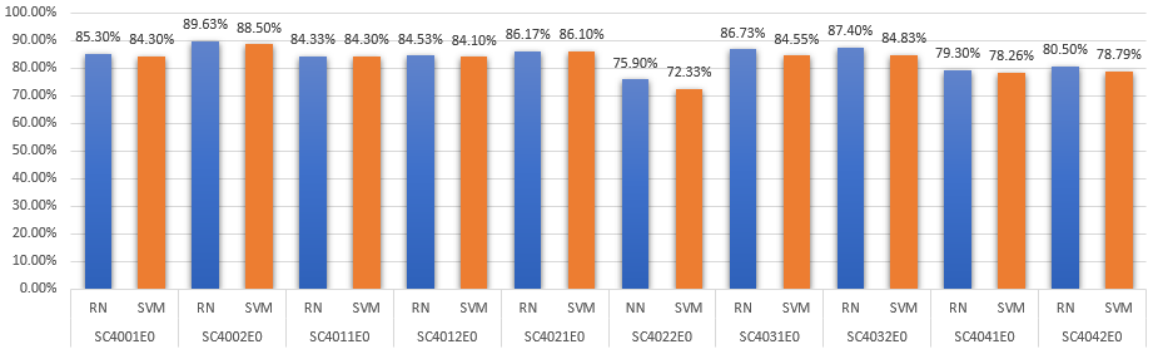


Figura 67: Resultados de clasificadores con extracción de MAV y ZC

De la figura anterior se puede ver los resultados para una primera prueba son bastante satisfactorios, sin embargo, solo se estaban considerando cuatro clases por el momento. De estos resultados se determinó que ambos clasificadores daban resultados muy similares, pero la red neural siempre fue un poco mejor. Para las siguientes pruebas donde ya se evaluarían cinco clases se decidió solo seguir utilizando la red neural debido a que daba ligeramente mejores resultados y porque tomaba significativamente menos tiempo entrenar la red.

Entre más datos y más características se consideraran este tiempo iba a aumentar todavía más para el SVM. La siguiente prueba consistió en analizar como se comportaba la red con cinco clases extrayendo las mismas dos características. Ahora que solo se iba a utilizar la red neuronal se podían hacer más corridas de una manera más rápida, por lo que los resultados que se muestran en la Figura 68 son el promedio de entrenar la red diez veces con los mismos vectores. Cabe mencionar que a partir de ahora el resto de resultados que se presentan en esta sección también son el promedio de entrenar diez veces la red.

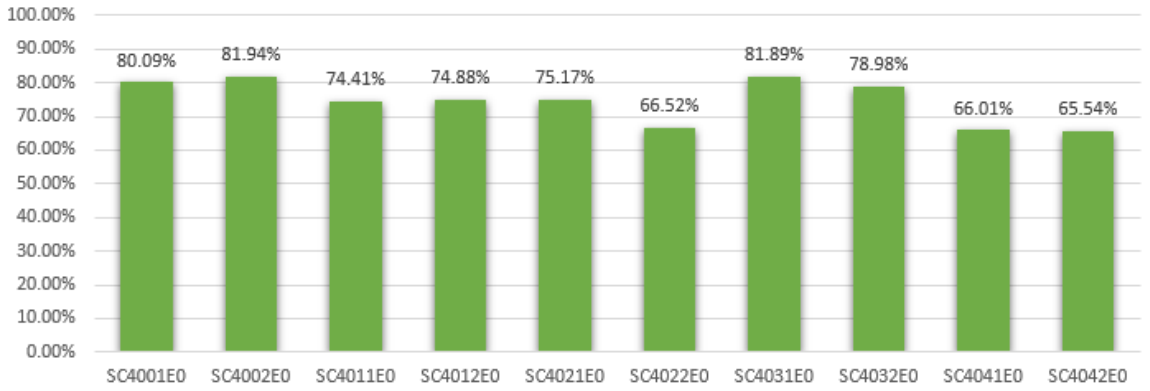


Figura 68: Resultados de red neuronal con extracción de MAV y ZC para cinco clases

Como era de esperarse al incluirse una clase más los porcentajes de efectividad bajaron considerablemente, aunque hubiera algunas redes que rondaban el 80 % de acierto, también había tres que estaban debajo del 70 %. En la Figura 69 se puede ver la matriz de confusión de una red para la última grabación SC4042E0 donde se evidencia claramente que las clases

que las clases que presentan muchísimo error son la clase 1 y 5, correspondientes a la Etapa 1 y Etapa REM del sueño. Esta baja efectividad era esperada dado que las señales EEG de ambas etapas son sumamente parecidas y los expertos que hacen las anotaciones requieren de otros canales como EOG y EMG para distinguir una etapa de otra.

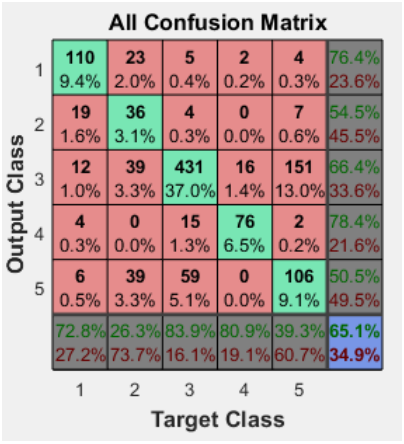


Figura 69: Matriz de confusión para red entrenada con grabación "SC4042E0"

La siguiente prueba fue agregar una característica adicional para ver si eso ayudaba a mejorar el rendimiento. En la siguiente prueba se utilizó también el MMD configurado para 30 subdivisiones por lo que cada 100 muestras calcularía el MMD y luego sumaría y sacaría el promedio de todos los MMD calculados. Los resultados de esta prueba se observan en el Cuadro 5 en el que se comparan directamente con los resultados de la Figura 68

Grabación	MAV-ZC (%)	MAV-ZC-MMD (%)
SC4001E0	80.09	80.3
SC4002E0	81.94	81.64
SC4011E0	74.41	74.92
SC4012E0	74.88	73.89
SC4021E0	75.17	74.91
SC4022E0	66.52	66.31
SC4031E0	81.89	81.06
SC4032E0	78.98	78.9
SC4041E0	66.01	66.11
SC4042E0	65.54	65.27

Cuadro 5: Porcentaje de rendimiento extrayendo dos y tres características

En el cuadro anterior se puede apreciar que los resultados se mantienen básicamente iguales, e incluso se puede decir que se pierde un poco de rendimiento al incluir al MMD como tercera característica. Cabe mencionar que se hicieron pruebas combinando MMD con ZC y MMD con MAV y aunque los resultados fueran bastante parecidos, los mejores rendimientos siempre se obtuvieron con la combinación de MAV y ZC. También se hicieron unas pequeñas pruebas con el SVM pero siguió mostrando las mismas tendencias que para las pruebas anteriores. Además, también se evaluó utilizar otras características basadas en

el tiempo como calcular la energía o entropía de la señal, pero tampoco se obtuvo alguna mejoría. Dado que el MMD ni las otras características estaban aportando para mejorar el rendimiento de la red se continuó solo utilizando MAV y ZC para el resto de pruebas.

Otro escenario que es bastante común para buscar conseguir un mejor rendimiento es hacer subdivisiones dentro de las mismas épocas, tal como se hizo con el MMD. La idea detrás de esto es que una señal puede tener sus características más distintivas en alguna sección específica de la época y calcularlas de la época total no se diferencian tanto del resto. Por ejemplo, una señal puede cruzar muchas veces por cero en la primera mitad de la época y en la segunda mitad prácticamente no cruzar. Por otro lado, otra señal cruza moderadamente el cero durante toda la época, por lo que al sacar el ZC de ambas señales el resultado podría llegar a ser muy parecido.

Al subdividir las épocas, la red podría aprovechar mejor las características seleccionadas para distinguir entre ambas señales. Al hacer una división la época se divide en 2 y los vectores de características se duplican, ya que se calculan las 2 características para cada mitad de la época. En el caso de hacer 2 divisiones las épocas se dividen en 3 y la matriz de características original con 2 filas pasaría a ser una con 6 filas. Los resultados de estas pruebas se resumen el la Figura 70 donde se tomaron como muestra las 5 grabaciones tuvieron mejor rendimiento en las pruebas anteriores.

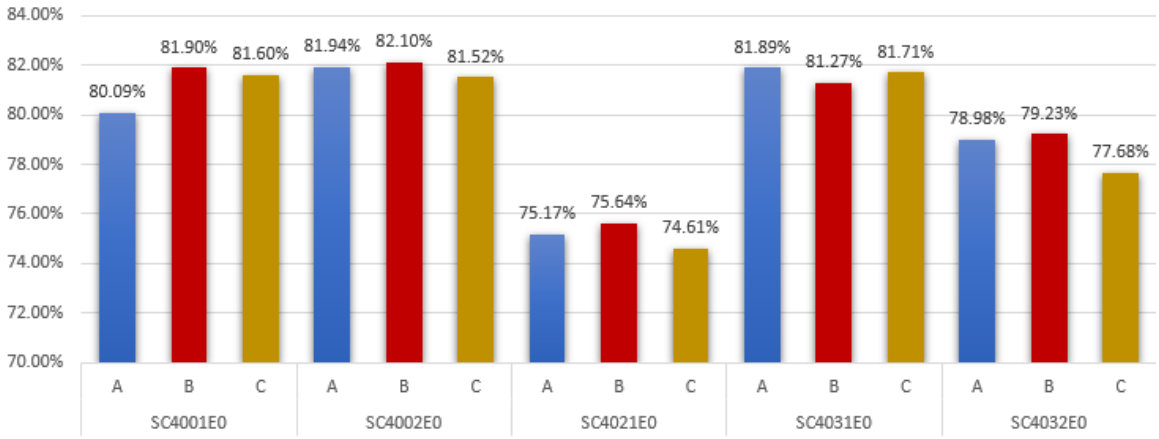


Figura 70: Resultados de red neuronal dividiendo las épocas (A = 0 divisiones, B = 1 división, C = 2 divisiones)

De los resultados anteriores se puede observar que al hacer las divisiones prácticamente no ganó mucho rendimiento, en cualquier caso parece que se obtiene un ligero mayor rendimiento cuando las épocas se dividen en dos, aunque esto no quiera decir que se repita para todos los casos y dada la aleatoriedad que mencionaba anteriormente tampoco se puede decir que se concluyente. Dado que la mejora no fue significativa el siguiente paso a seguir era agregar más canales de los que se tenían disponibles en la base de datos. De este paso se esperaba un crecimiento significativo en el rendimiento dado que el resto de señales son lo complementan una polisomnografía completa. En otras palabras, un especialista no puede identificar las etapas del sueño en base a solo un canal de EEG, por lo que para obtener un rendimiento más alto con solo un canal se deben de explorar otro tipo de procedimientos y otro tipo de características que puedan contribuir a una mejora.

Se fueron agregando los canales “Pz-Oz”, “EOG Horizontal” y “EMG Submental” secuencialmente y probando hacer una y dos divisiones en las épocas. Cabe mencionar que el canal de EMG fue muestreado a 1 Hz, por lo que en la base de datos hicieron una interpolación para que tuviera la misma cantidad de datos en las épocas que el resto de señales. Este canal daba una idea general del comportamiento muscular, pero no era la mejor forma de medir y analizar este tipo de actividad. Los mejores resultados se obtuvieron a partir de utilizar los 4 canales haciendo una sola división en las épocas. En el Cuadro 6 se muestra una recopilación de estos resultados donde además del promedio de las 10 veces que se entrenó, también se muestra el rendimiento máximo y mínimo de esos 10 entrenamientos.

Grabación	**Mín.** (%)	**Máx.** (%)	**Promedio** (%)
SC4001E0	86.7	90.0	88.6
SC4002E0	87.8	89.6	88.09
SC4011E0	88.0	89.6	88.66
SC4012E0	84.6	88.8	85.6
SC4021E0	87.9	90.6	89.67
SC4022E0	87.1	89.1	87.74
SC4031E0	85.5	92.8	89.27
SC4032E0	87.5	91.2	89.4
SC4041E0	81.4	84.2	82.78
SC4042E0	81.5	85.1	82.29

Cuadro 6: Rendimientos máximos, mínimos y promedios de las redes entrenadas con cuatro canales

Claramente los resultados subieron significativamente respecto a las pruebas anteriores pudiendo encontrar redes con un rendimiento máximo superior al 90 %. En [7] se menciona que los rendimientos promedio para todas los estudios que recompilaron rondan entre el 70 % y 94 % de acierto por lo que resultados que son completamente satisfactorios. En la Figura 71 se muestra un ejemplo de una matriz de confusión para una de las grabaciones que mejores resultados dio. Como se puede apreciar los porcentajes de acierto por clase también aumentaron considerablemente, sin embargo la clase 1 siempre se mantuvo con el porcentaje más bajo de todas las clases, posiblemente por su considerable parecido a la clase 2 y 5 y porque en algunas de las grabaciones era la clase que menos etiquetas tenía.

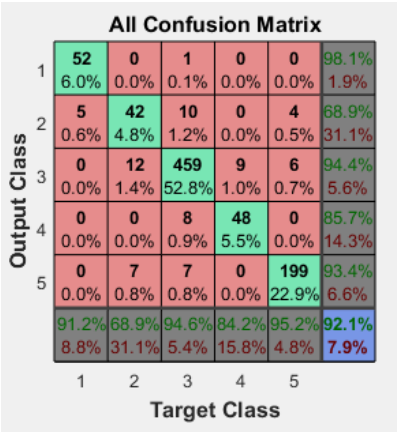


Figura 71: Matriz de confusión de red entrenada con cuatro canales para grabación “SC4031E0”

Como ya se mencionó anteriormente, el proceso de selección de características y entrenamiento de clasificadores puede llegar a ser sumamente complejo dado que tiene múltiples posibles variantes. Solo en [7] se mencionan hasta 33 tipos distintos de características, más de 12 tipos de clasificadores e incluso hasta 14 tipos de técnicas para ayudan en el proceso de selección de las mejoras características. A estas variantes se le tiene que sumar el número clases que se quieran clasificar, el número de canales disponibles y si se van a dividir las épocas. Dado que este proyecto se encuentra una fase inicial, los resultados se consideran bastante satisfactorios.

Cabe resaltar que el sueño de una persona es único por lo que cada grabación es completamente diferente una de otra aunque se hayan medido los mismos canales. Hay una cantidad de factores innumerables que pueden contribuir a los resultados que se obtienen. Por ejemplo, la hora de dormirse, el calor del cuarto, la edad de las personas, actividades que se realizaron antes de dormirse, comidas que se ingirieron ese día etc. En fin, dadas estas características del sueño, un entrenamiento a un clasificador de forma personalizada siempre va a dar los mejores resultados. Al tratar de hacer un modelo general que funcione efectivamente para cualquier persona en base a un entrenamiento con múltiples grabaciones, es posible que los resultados no igualen a los de un entrenamiento específico para una persona.

Aun así, era conveniente evaluar la factibilidad de crear un sistema que pueda aplicarse para cualquier persona. Esto con la idea de que al crear una base datos propia solo se requiera obtener anotaciones de algunas grabaciones iniciales para posteriormente poder utilizar el sistema en cualquier persona de manera directa. De esta forma se evitaría que un especialista del sueño tenga que intervenir para cada nuevo usuario. Para evaluar esta factibilidad lo que se hizo fue combinar grabaciones para entrenar las redes.

Se tomaron 9 de las 10 grabaciones y se combinaron sus datos para entrenar una red, posteriormente la grabación que no se utilizó para entrenar fue utilizada como entrada de la red para evaluar su rendimiento. Para esto ya se utilizaba propiamente el modelo de la red que se generó con el entrenamiento para que predijera las clases correspondientes a los datos de la grabación restante. Dado que se cuenta con las etiquetas de dicha grabación se podía obtener un porcentaje de acierto al comparar las etiquetas reales contra las que etiquetas que clasificó la red. En la Figura 72 se pueden observar el resultado de estas pruebas, en color amarillo se muestra el rendimiento de entrenar una red con todas las muestras a excepción de la que se indica abajo de dicha barra, mientras que las barras de color morado indican el porcentaje de acierto que tuvo la red al tener como entrada la grabación que no se utilizó para entrenar.

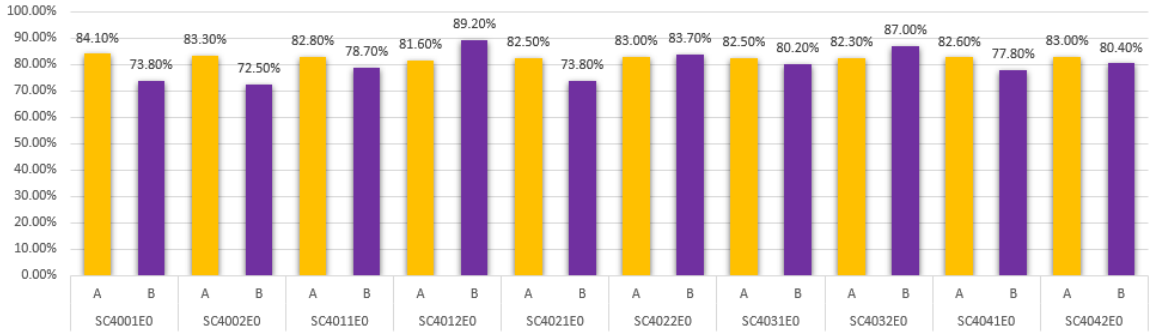


Figura 72: Resultados de red neuronal general (A = Rendimiento de red, B = Rendimiento de predicciones)

Como se puede observar, los resultados de combinar datos de distintos sujetos para entrenar las redes obtienen un menor rendimiento que lo que se observó en el Cuadro 6, tal como se esperaba. Sin embargo, estos resultados siguen siendo bastante positivos considerando que se están clasificando cinco clases. Por otro lado, los porcentajes de acierto de las predicciones son mejor de lo esperado, muchos rondando cerca del 80 % de certeza. Estos resultados invitan a pensar que hay suficientes similitudes y patrones reconocibles en las señales de distintas grabaciones. Dada esta intuición es probable que con pruebas más exhaustivas se pueda llegar a obtener porcentajes de acierto todavía mejores que posibiliten la elaboración de un sistema general. De ser esto posible se eliminaría la necesidad de tener que hacer un entrenamiento personalizado para cada usuario que use el sistema.

CAPÍTULO 12

Simulaciones finales

Para englobar todo lo que se abarcó en este trabajo se desarrollarían dos simulaciones que sirvieran como prueba de concepto y fundamento de lo que sería el funcionamiento del sistema al juntar todas las fases de este proyecto. En la Figura 73 se puede observar un resumen del proceso que debe de realizar el sistema final para reproducir los audios correctamente. Primero, el sistema debe de ser capaz de captar las señales EEG y filtrarlas correctamente para obtener señales correctas. Toda esta parte del proceso se abarcó en los Capítulos 7, 9 y 10.

Posteriormente, en base a las lecturas de las señales el sistema debe de poder extraer características en tiempo real que alimenten un clasificador previamente entrenado. La extracción y clasificación de las señales se resumen en el Capítulo 11. Por último el sistema, en base al resultado del clasificador, debe de poder reproducir un pulso binaural respectivo. La generación de los audios se cubre en el Capítulo 8. La idea central de ambas simulaciones es demostrar como el sistema es capaz de clasificar las etapas del sueño en tiempo real para así reproducir un pulso binaural acorde a la clasificación. Dado los buenos resultados que ofreció una frecuencia central de 250 Hz en [5], se eligió esta frecuencia para los pulsos a reproducir en ambas simulaciones. Se debe de resaltar que se deb de escuchar los audios a través de audífonos para percibir el fenómeno detrás de los pulsos binaurales.

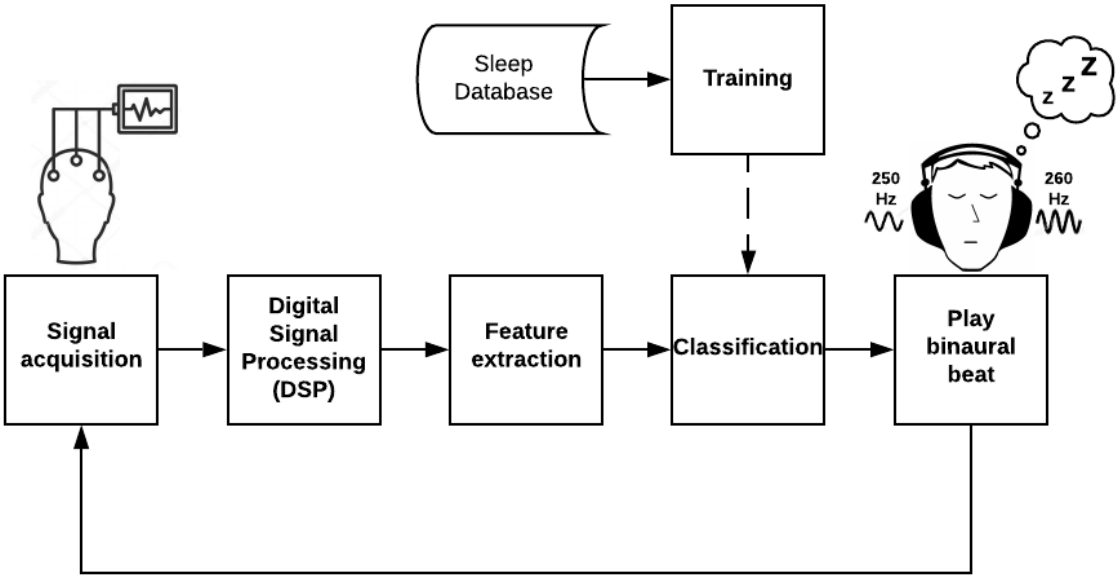


Figura 73: Concepto de funcionamiento

12.1. Simulación con base de datos

Se desarrolló una simulación en Matlab utilizando las polisomnografías de la base de datos mencionada en el capítulo anterior para demostrar como funcionaría un sistema final. El programa lee y despliega continuamente los datos de los canales EEG, EOG y EMG de una polisomnografía seleccionada, emulando como que si se estuvieran adquiriendo las señales en tiempo real. Mientras los datos se van leyendo, el sistema extrae un vector de características por cada época que transcurre y lo envía como señal de entrada a un clasificador previamente entrenado con señales de la base de datos. Dependiendo del resultado de la clasificación el sistema reproduce un pulso binaural determinado.

En esta simulación cuando se detectara la etapa de "Despierto" se reproduciría un pulso binaural de 8 Hz, uno de 6 Hz para la etapa 1, de 4 Hz para la etapa 2 y finalmente uno de 2 Hz para la etapa 3. Dadas las recomendaciones del Dr. Luis Alejandro López no se reproduciría ningún pulso durante la etapa REM para evitar despertar al sujeto de prueba. Para maximizar los resultados en un sistema final se deberán realizar pruebas con pacientes reales para determinar el impacto que tienen determinados pulsos binaurales haciendo pruebas con distintas frecuencias y volúmenes. Dicho escenario no estaba contemplado para los alcances de esta primera fase del proyecto.

En la Figura 74 se puede ver una captura de pantalla de la simulación. Lo que se observa en la parte superior de figura son cuatro ventanas en donde se simula la lectura de una señal de la polisomnografía a lo largo del tiempo. En la parte inferior izquierda se muestra un mensaje que indica cual es la etapa real de la última época procesada. Por otro lado en la parte inferior derecha se indica cual es la predicción del sistema para la última época procesada. Para agilizar la simulación se aceleró el tiempo de lectura de los datos para que las épocas fueran procesadas en un tiempo de 6 segundos aproximadamente en lugar de los

30 segundos originales. Los pulsos binaurales se hicieron de esa misma duración para que se escuchasen durante toda la época respectiva.

Adicionalmente, esta simulación tiene dos modalidades. Se puede correr para que se lean las señales de manera convencional o se puede seleccionar que se corra utilizando un vector de prueba. Este vector esta compuesto de épocas seleccionadas para hacer una simulación más corta donde se aprecien rápidamente todas las etapas del sueño. En caso de simular con el modo convencional se tendría que esperar un tiempo considerable antes de que se clasifiquen las últimas etapas del ciclo del sueño. Para una mejor comprensión se recomienda ver el vídeo de funcionamiento en la sección 16.1.4.

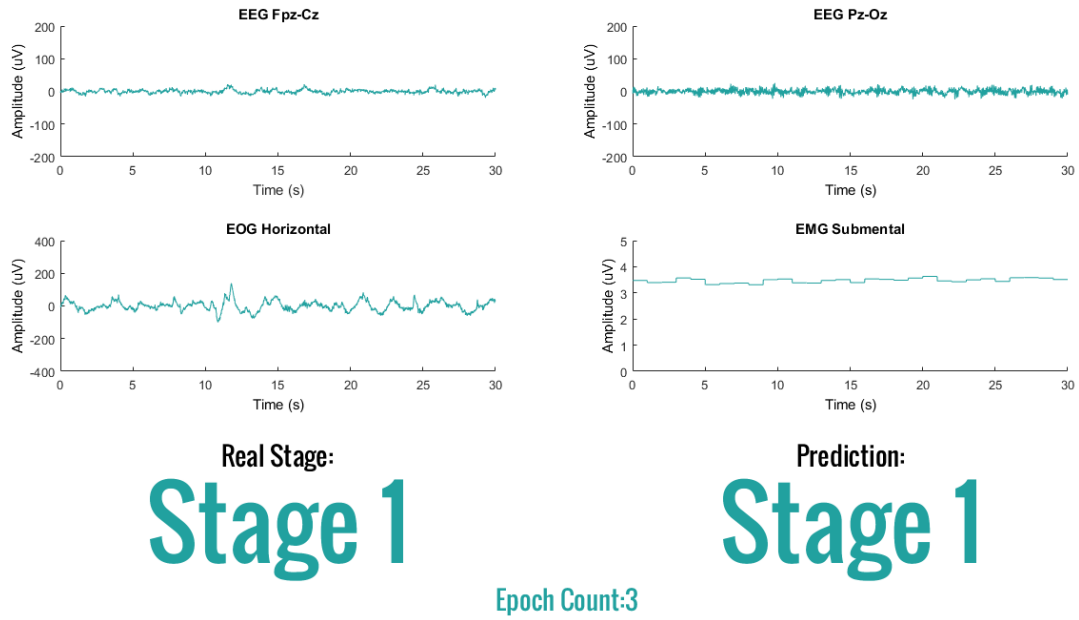


Figura 74: Simulación utilizando base de datos

12.2. Simulación con datos obtenidos en tiempo real

En la segunda simulación se siguen los mismos principios que en el caso anterior con la excepción de que en lugar de leer datos de la base de datos se estarían leyendo y procesando datos en tiempo real obtenidos desde la Cyton Board y el Electro-Cap, es decir, se estarían obteniendo las señales desde hardware propio. Para poder obtener el flujo de los datos crudos desde Matlab se utilizó la herramienta Lab Stream Layer (LSL) desde un código en Python. Este último código se encuentra dentro de los archivos de OpenBCI y solamente fue levemente modificado para que se acoplara al envío de datos deseado. De esta forma se estarían leyendo los datos crudos exactamente de la misma forma que en Capítulo 9 con la única diferencia que en lugar de imprimir los resultados en consola estos sería enviados directamente a Matlab a través de LSL. Para poder utilizar dicha herramiento fue necesario instalar la librería liblsl y el toolbox MinGW-w64 C/C++ Compiler.

De este flujo de datos solo se estaría procesando un solo canal para que la simulación se mantuviera rápida y eficiente. Se utilizó el mismo proceso de filtrado que en el capítulo 11 para obtener correctamente la señal leída. Posteriormente, igual que en la simulación anterior, se extraen las características cada época para que sean alimentadas a una red neuronal. Se recomienda ver el vídeo en 16.1.5 para observar a mayor detalle como funcionaría el sistema con un sujeto de prueba. En la Figura 75 se puede observar una captura de pantalla de dicho vídeo.

Cabe mencionar que para esta prueba no fue posible determinar el nivel de certeza de las predicciones dado que no se obtuvieron anotaciones de un experto para datos capturados desde la Cyton Board y el Electro-Cap. Por este motivo para entrenar la red neuronal se utilizó el canal EEG Fpz-Oz de la base de datos. La predicción del clasificador es puramente ilustrativa dado que hay muchos factores que podían afectar su certeza, como utilizar un solo canal en lugar de 4, entrenar con los datos del base de datos, utilizar el electrodo Fp1, estar completamente despierto, etc. Sin embargo, el propósito de la simulación era enseñar la funcionalidad de un sistema que operara en base al diagrama de la Figura 73.

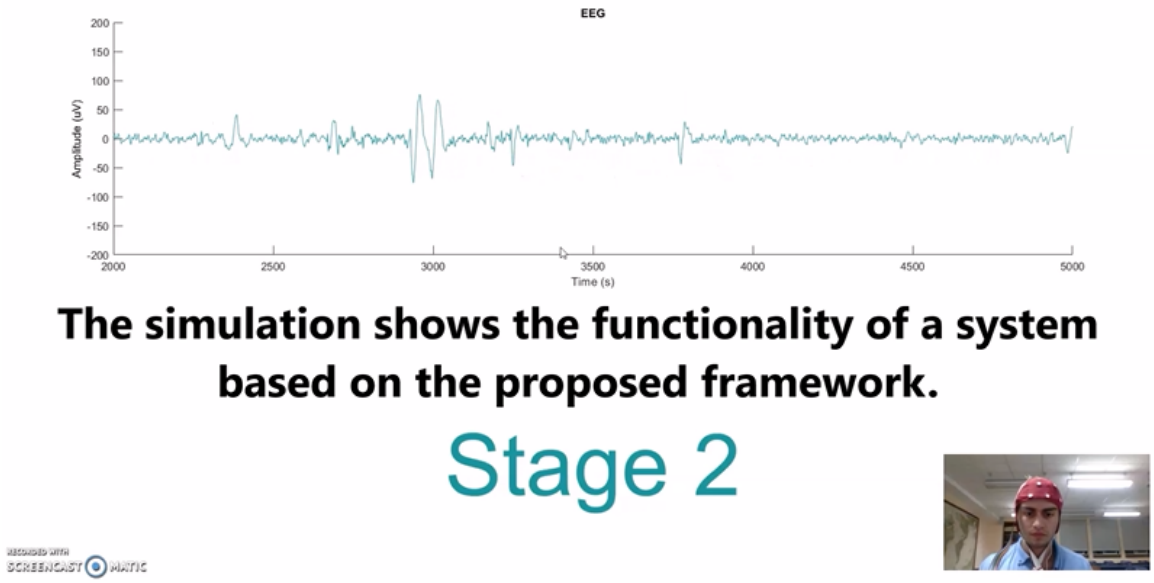


Figura 75: Simulación adquiriendo datos en tiempo real

CAPÍTULO 13

Conclusiones

- Los adaptadores realizados para obtener una conexión exitosa entre la Cyton Board y el Electro-Cap brindan la posibilidad de tener un sistema más cómodo y con más capacidad de medición que los cascos de OpenBCI que se habían utilizado en proyectos pasados de la universidad.
- El algoritmo que se desarrolló y automatizó para generar los pulsos binaurales probó ser una herramienta eficaz con la que fácilmente se pueden obtener audios estéreo matemáticamente correctos según sean requeridos.
- De todas las alternativas disponibles, tanto los resultados de la adquisición de los datos como el procesamiento de los mismos, demostraron que Python es una herramienta accesible y más que viable para formar parte de un sistema de procesamiento digital.
- El diseño e implementación de filtros IIR, Butterworth de 2ndo orden, permitieron replicar idénticamente las señales que se obtienen con la interfaz de la empresa OpenBCI.
- Las redes neuronales probaron ser un clasificador más efectivo que las máquinas de vectores de soporte para las características basadas en el tiempo que fueron seleccionadas.
- El rendimiento de la red al ser entrenada con un solo canal de EEG, pese a tener resultados aceptables, no pudo mejorarse significativamente con la adición de características ni con la subdivisión de las épocas.
- La recopilación de estudios en [7] demuestra que el rendimiento de clasificación obtenido con cuatro canales en base a las características de MAV y ZC son satisfactorios a pesar de que el enfoque del trabajo era aplicar un clasificador en una aplicación que funcionara en tiempo real.
- Los resultados obtenidos de entrenar redes neuronales con múltiples grabaciones dieron indicios positivos de que puede ser factible realizar un sistema de clasificación automática de las etapas del sueño de funcionamiento general y no de entrenamiento personalizado.

- El desarrollo de simulaciones tanto en Python como Matlab demostró que la programación multihilos es esencial para que los audios se puedan reproducir sin afectar la ejecución del resto de procesos del código.
- Las simulaciones finales demuestran la funcionalidad de un sistema capaz de adquirir señales y reproducir pulsos binaurales en base a la extracción de características y el entrenamiento de un clasificador.
- Dadas las altas exigencias del mundo moderno, un sistema que ayude a las personas a combatir los desórdenes de sueño sin la necesidad de medicamento sería invaluable.

CAPÍTULO 14

Recomendaciones

- Aunque los adaptadores para la conexión del Electro-Cap hayan funcionado correctamente, para la implementación de un sistema ya final se recomendaría hacer una pequeña placa para hacer la conexión entre el DB-25 y los pines de la Cyton, una vez se tenga bien definido del montaje de electrodos a utilizar. De esta forma quién use el sistema no se debe de preocupar de conectar cables además de que se tendría una conexión más segura en el que la persona que este durmiendo no se tenga que preocupar que se zafe o enrede algún cable.

- Aunque los pulsos binaurales se generen de forma correcta, para determinar una metodología de aplicación que mejoraría el sueño es necesario hacer muchas pruebas con pacientes reales para ver el efecto que tiene aplicarlos. Incluso, también se pueden considerar diversos estudios que dicen que los pulsos monoaurales tienen un mayor impacto que los binaurales y se consiguen simplemente reproduciendo los audios que ya se pueden generar en una bocina en lugar de audífonos.

- Como ya se mencionó anteriormente, OpenBCI se mantiene en constante actualización por lo que nuevas alternativas han surgido para conectarse a softwares externos, ha mejorado la documentación y se ha facilitado el uso de cualquiera de las opciones anteriores. Dada esta situación, para una fase futuro de este proyecto podría ser beneficioso analizar nuevamente las opciones ya sea para hacer más simple la conexión con Python o explorar adquirir los datos desde otra plataforma como Matlab.

- Para una fase futura la recomendación más importante sería la de evaluar integrar al sistema la lectura de ondas EOG y EMG. Primero, porque el proceso de entrenamiento con estos canales dio resultados realmente buenos. Y segundo y más importante, para entrenar modelos de clasificación con datos obtenidos de la Cyton, es necesario obtener una base de datos propia de gente durmiendo que haya sido debidamente etiquetada por un especialista. De las conversaciones con el Dr. Luis Alejando López se mencionó que no es posible hacer las anotaciones en base a solo lecturas de los canales EEG, principalmente para la etapa 1 y etapa REM. Dado que sin las anotaciones del especialista no es posible entrenar un clasificador, es indispensable obtener un contacto

que pueda hacer las anotaciones y determine que canales son los que definitivamente se necesitarían. Posteriormente ya se podría obtener la base de datos donde se pueda entrenar un modelo clasificador para luego realizar más pruebas que permitan concluir si es posible obtener un sistema general que ya no necesite de anotaciones para pacientes futuros.

- Dependiendo de las determinaciones que se tomen de la recomendación anterior y dada la gran cantidad de opciones que hay para hacer un sistema de detección de las etapas del sueño, también valdría la pena explorar hacer el entrenamiento de un modelo según un set de características y clasificadores más complejos en búsqueda de aumentar el porcentaje de certeza.

  [46]

CAPÍTULO 15

Bibliografía

[1] P. Schembri, R. Anthony y M. Pelc, «Detection of Electroencephalography Artefacts using Low Fidelity Equipment», en *Proceedings of the 4th International Conference on Physiological Computing Systems*, SCITEPRESS - Science y Technology Publications, 2017. DOI: 10.5220/0006398500650075. dirección: https://doi.org/10.5220/0006398500650075.

[2] C. Audette, *Control An Air Shark With Your Mind*, Accedido por última vez el 20 de marzo de 2019, 2015. dirección: https://spectrum.ieee.org/geek-life/hands-on/openbci-control-an-air-shark-with-your-mind.

[3] J. Aguirre, «Diseño, Análisis, y Desarrollo de un Sistema de Entrenamiento para Mejorar el Desempeño de los Atletas del Comité Olímpico Guatemalteco», Universidad del Valle de Guatemala, 2018.

[4] M. Godoy, «Sistema de Neurofeedback para mejorar el rendimiento de los Atletas del Comité Olímpico Guatemalteco. Diseño e implementación de módulo de recopilación de señales y módulo de retroalimentación», Universidad del Valle de Guatemala, 2018.

[5] N. Jirakittayakorn e Y. Wongsawat, «Brain Responses to a 6-Hz Binaural Beat: Effects on General Theta Rhythm and Frontal Midline Theta Activity», *Frontiers in Neuroscience*, vol. 11, jun. de 2017. DOI: 10.3389/fnins.2017.00365. dirección: https://doi.org/10.3389/fnins.2017.00365.

[6] V. Abeln, J. Kleinert, H. K. Strüder y S. Schneider, «Brainwave entrainment for better sleep and post-sleep state of young elite soccer players – A pilot study», *European Journal of Sport Science*, vol. 14, n.º 5, págs. 393-402, jul. de 2013. DOI: 10.1080/17461391.2013.819384. dirección: https://doi.org/10.1080/17461391.2013.819384.

[7] K. Aboalayon, M. Faezipour, W. Almuhammadi y S. Moslehpour, «Sleep Stage Classification Using EEG Signal Analysis: A Comprehensive Survey and New Investigation», *Entropy*, vol. 18, n.º 9, pág. 272, ago. de 2016. DOI: 10.3390/e18090272. dirección: https://doi.org/10.3390/e18090272.

[8] S. L. Halson, «Sleep and Athletes», *Sport Science Exchange*, vol. 28, n.º 167, págs. 1-4, 2016. dirección: https://secure.footprint.net/gatorade/prd/gssiweb/sf_libraries/sse-docs/bball-taskforce_sse_167.pdf?sfvrsn=2.

[9] H. Tuomilehto, V.-P. Vuorinen, E. Penttilä, M. Kivimäki, M. Vuorenmaa, M. Venojärvi, O. Airaksinen y J. Pihlajamäki, «Sleep of professional athletes: Underexploited potential to improve health and performance», *Journal of Sports Sciences*, vol. 35, n.º 7, págs. 704-710, mayo de 2016. DOI: 10.1080/02640414.2016.1184300. dirección: https://doi.org/10.1080/02640414.2016.1184300.

[10] X. Gao, H. Cao, D. Ming, H. Qi, X. Wang, X. Wang, R. Chen y P. Zhou, «Analysis of EEG activity in response to binaural beats with different frequencies», *International Journal of Psychophysiology*, vol. 94, n.º 3, págs. 399-406, dic. de 2014. DOI: 10.1016/j.ijpsycho.2014.10.010. dirección: https://doi.org/10.1016/j.ijpsycho.2014.10.010.

[11] C. Lavelle, I. Valentin, S. Roy y J. Roy, *What are Brainwaves?*, Accedido por última vez el 25 de marzo de 2019, 2007-2019. dirección: https://brainworksneurotherapy.com/what-are-brainwaves.

[12] Reducción Hipno-Gástrica, *¿Qué son las ondas cerebrales?*, Accedido por última vez el 20 de marzo de 2019, 2014. dirección: https://reduccionhipnogastrica.wordpress.com/2014/12/16/que-son-las-ondas-cerebrales/.

[13] T. A. Sleeping, *Stages of Sleep and Sleep Cycle*, Accedido por última vez el 4 de abril de 2019, 2019. dirección: https://https://www.tuck.com/stages/#brain_waves_during_rem_and_non_rem_sleep.

[14] R. K. Malhotra y A. Y. Avidan, «Sleep Stages and Scoring Technique», en *Atlas of Sleep Medicine*, Elsevier, 2014, págs. 77-99. DOI: 10.1016/b978-1-4557-1267-0.00003-5. dirección: https://doi.org/10.1016/b978-1-4557-1267-0.00003-5.

[15] HealthJade, *How to sleep better*, Accedido por última vez el 12 de abril de 2019, 2019. dirección: https://healthjade.net/how-to-sleep-better/.

[16] R. Ranjan, R. Arya, S. L. Fernandes, E. Sravya y V. Jain, «A fuzzy neural network approach for automatic K-complex detection in sleep EEG signal», *Pattern Recognition Letters*, vol. 115, págs. 74-83, nov. de 2018. DOI: 10.1016/j.patrec.2018.01.001. dirección: https://doi.org/10.1016/j.patrec.2018.01.001.

[17] T. T. García, «Manual básico para enfermeros en electroencefalografía», *Enfermería Docente 11*, págs. 29-33, 1994. dirección: http://www.sspa.juntadeandalucia.es/servicioandaluzdesalud/huvvsites/default/files/revistas/ED-094-07.pdf.

[18] Mayo Clinic Health System, *Electroencefalografía (EEG)*, Accedido por última vez el 12 de abril de 2019, 1998-2019. dirección: https://www.mayoclinic.org/es-es/tests-procedures/eeg/about/pac-20393875.

[19] E. Fernández, *Manual de laboratorio de fisiología (6a. ed.)* McGraw-Hill Interamericana, 2000.

[20] R. Sepulveda, O. Montiel, G. Diaz, D. Gutierrez y O. Castillo, «Classification of Encephalographic Signals using Artificial Neural Networks», *Computación y Sistemas*, vol. 19, n.º 1, mar. de 2015. DOI: 10.13053/cys-19-1-1570. dirección: https://doi.org/10.13053/cys-19-1-1570.

[21] Electro-Cap International, *Caps*, Accedido por última vez el 5 de mayo de 2019, 2019. dirección: https://electro-cap.com/index.cfm/caps/.

[22] OpenBCI, *Learning Materials and Guides*, Accedido por última vez el 10 de julio de 2019, 2019. dirección: https://docs.openbci.com/docs/01GettingStarted/GettingStartedLanding.

[23] L. Escobar, *Procesadores de Señales (DSPs) y Aplicaciones*, Accedido por última vez el 25 de julio de 2019, 2019. dirección: http://odin.fi-b.unam.mx/labdsp/files/ADSP/apuntes/dsp_apli0_17.pdf.

[24] M. Martínez, L. Gómez, A. J. Serrano, J. Vila y J. Gómez, *Introducción al Procesado Digital de Señales*, Accedido por última vez el 12 de agosto de 2019, 2009-2010. dirección: http://ocw.uv.es/ingenieria-y-arquitectura/1-1/tema1.pdf.

[25] J. W. Nilsson y S. A. Riedel, *Electric circuits*, 10.ª ed. Pearson, 2019.

[26] L. Jure, E. López y M. Rocamora, *Introducción a la Teoría del Procesamiento Digital de Señales (DSP) de Audio*, Accedido por última vez el 10 de agosto de 2019, 2019. dirección: https://www.eumus.edu.uy/eme/ensenanza/electivas/dsp/presentaciones/clase10.pdf.

[27] K.L. University, *Frequency Response of Filters*, Accedido por última vez el 4 de agosto de 2019, 2014. dirección: https://www.slideshare.net/junaidsk560/op-amp-applications-filters-cw.

[28] N. Davies, *An Introduction to Filters*, Accedido por última vez el 17 de agosto de 2019, 2017. dirección: https://www.allaboutcircuits.com/technical-articles/an-introduction-to-filters/.

[29] M. Martínez, L. Gómez, A. J. Serrano, J. Vila y J. Gómez, *Revisión de los tipos de filtros analógicos más comúnes*, Accedido por última vez el 12 de agosto de 2019, 2009-2010. dirección: http://ocw.uv.es/ingenieria-y-arquitectura/filtros-digitales/tema_2._revision_de_los_tipos_de_filtros_analogicos_mas_comunes.pdf.

[30] S. Xie, *Practical Filter Design Challenges and Considerations for Precision ADCs*, Accedido por última vez el 28 de agosto de 2019, 2015. dirección: https://www.analog.com/en/analog-dialogue/articles/practical-filter-design-precision-adcs.html.

[31] P. Cetta, *Filtros Digitales*, Accedido por última vez el 14 de agosto de 2019, 2012. dirección: https://www.pablocetta.com/pdfs/publicaciones/filtros.pdf.

[32] G. Oster, «Auditory Beats in the Brain», *Scientific American*, vol. 229, n.º 4, págs. 94-102, oct. de 1973. DOI: 10.1038/scientificamerican1073-94. dirección: https://doi.org/10.1038/scientificamerican1073-94.

[33] P. Lix, *Binaural Beats Meditation*, Accedido por última vez el 7 de septiembre de 2019, 2011-2019. dirección: https://www.binauralbeatsmeditation.com/the-science/.

[34] M. Garcia-Argibay, M. A. Santed y J. M. Reales, «Binaural auditory beats affect long-term memory», *Psychological Research*, vol. 83, n.º 6, págs. 1124-1136, dic. de 2017. DOI: 10.1007/s00426-017-0959-2. dirección: https://doi.org/10.1007/s00426-017-0959-2.

[35] A. Colmenar, *El sonido digital: formatos, captura, edición, manipulación, conversión y grabación*, Accedido por última vez el 10 de agosto de 2019, 2019. dirección: https://ocw.innova.uned.es/mm2/tm/contenidos/pdf/tema3/tmm_tema3_sonido_digital_presentacion.pdf.

[36] R. Smith, *Filtering a sound recording*, Accedido por última vez el 10 de julio de 2019, 2013. dirección: https://rsmith.home.xs4all.nl/miscellaneous/filtering-a-sound-recording.html.

[37] S. J. Espinoza, *Implementación de un Laboratorio Virtual para la Operación Centralizada de Equipos de Medición de Telecomunicaciones del CICTE*, Accedido por última vez el 5 de septiembre de 2019, 2011. dirección: https://repositorio.espe.edu.ec/bitstream/21000/2767/1/T-ESPE-030522.pdf.

[38] G. J. Giner, *Introducción al Aprendizaje Automático*, Accedido por última vez el 3 de agosto de 2019, 2018. dirección: https://br.escueladenegociosydireccion.com/big-data/introduccion-al-aprendizaje-automatico/.

[39] P. Larranaga, I. Inza y A. Moujahid, *Redes Neuronales*, Accedido por última vez el 8 de septiembre de 2019, 2018. dirección: http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t8neuronales.pdf.

[40] E. J. Carmona, *Tutorial sobre Máquinas de Vectores de Soporte (SVM)*, Accedido por última vez el 10 de septiembre de 2019, 2014. dirección: http://www.ia.uned.es/~ejcarmona/publicaciones/[2016-%5C%20Carmona]%5C%20SVM.pdf.

[41] J. Martínez, *Máquinas de Vectores de Soporte (SVM)*, Accedido por última vez el 10 de septiembre de 2019, 2019. dirección: https://iartificial.net/maquinas-de-vectores-de-soporte-svm/.

[42] M. Larsson, *Sleep Shepherd Blue*, Accedido por última vez el 15 de marzo de 2019, 2019. dirección: https://sleeptrackers.io/sleep-shepherd-blue/.

[43] A. Koushik, J. Amores y P. Maes, «Real-time Smartphone-based Sleep Staging using 1-Channel EEG», en *2019 IEEE 16th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, IEEE, mayo de 2019. DOI: 10.1109/bsn.2019.8771091. dirección: https://doi.org/10.1109/bsn.2019.8771091.

[44] K. A. I. Aboalayon y M. Faezipour, «Multi-class SVM based on sleep stage identification using EEG signal», en *2014 IEEE Healthcare Innovation Conference (HIC)*, IEEE, oct. de 2014. DOI: 10.1109/hic.2014.7038904. dirección: https://doi.org/10.1109/hic.2014.7038904.

[45] S. A. Imtiaz y E. Rodriguez-Villegas, «An open-source toolbox for standardized use of PhysioNet Sleep EDF Expanded Database», en *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, ago. de 2015. DOI: 10.1109/embc.2015.7319762. dirección: https://doi.org/10.1109/embc.2015.7319762.

[46] E. R. Widasari, K. Tanno y H. Tamura, «Automatic Sleep Stage Detection Based on HRV Spectrum Analysis», en *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, oct. de 2018. DOI: 10.1109/smc.2018.00155. dirección: https://doi.org/10.1109/smc.2018.00155.

CAPÍTULO 16

Anexos

16.1. Vídeos de funcionamiento

16.1.1. Vídeo 1

Este vídeo es una primera simulación que se realiza en las primeras etapas del desarrollo del proyecto. Luego de haber podido tener una lectura de los datos en crudo se quiso combinar dicho código con la reproducción de los pulsos binaurales, simulando como funcionaría un sistema final luego de desarrollarse más fases. Esta simulación se puede encontrar al presionar el siguiente enlace o al ingresar a https://www.youtube.com/watch?v=voNAscUr9ps.

16.1.2. Vídeo 2

Este vídeo fue una grabación de muestra en donde se midieron siete canales del casco Ultracortex Mark IV. Este vídeo sirvió como medio de comparación para varias pruebas para el proceso de filtrado en tiempo real. El vídeo se puede encontrar al presionar el siguiente enlace o al ingresar a https://www.youtube.com/watch?v=JHVHqCIwsuY.

16.1.3. Vídeo 3

En este vídeo se hace un filtrado de datos en tiempo real del archivo CSV que se obtuvo del vídeo16.1.2. El objetivo de este proceso era recrear las mismas señales a partir de los datos crudos para verificar que la nueva metodología para filtrar los datos fuera efectiva. El funcionamiento de esta parte del trabajo se puede ver al presionar el siguiente enlace o al ingresar a https://www.youtube.com/watch?v=JVBF7jA2eLM.

16.1.4. Vídeo 4

En este vídeo se hace una simulación que toma señales de las grabaciones de una base de datos pública para validar la etapa de extracción de características, entrenamiento, clasificación y reproducción de los pulsos. Cabe mencionar que como se contaban con las etiquetas de todas las grabaciones, sí era posible determinar el porcentaje de acierto de las predicciones. Dicho rendimiento se muestra en un mensaje dentro de la simulación. Para poder verla a mayor detalle se recomienda ingresar al vídeo al presionar este enlace o al ingresar a https://youtu.be/G5P5746L_bo.

16.1.5. Vídeo 5

Por otro lado en este último vídeo se muestra el resultado de los experimentos realizados para validar las etapas de adquisición y procesamiento de de señales así como todo el funcionamiento de todo el sistema en tiempo real. A diferencia del Vídeo 4, en esta simulación los datos se adquieren de un sujeto de prueba. Para poder observar la simulación presione el siguiente enlace o ingrese a https://youtu.be/9Y1aTghbY3g.

CAPÍTULO 17

Glosario

Cyton Board: Placa de 8 canales de la empresa OpenBCI capaz de muestrear la actividad bioeléctrica del cerebro, muscular y el ritmo cardíaco, entre otros. 34, 36, 37, 39, 42, 53–55, 57, 61

Electro-Cap: Gorro de tela con 19 electrodos adheridos conforme el Sistema Internacional 10-20. 36, 37, 39–42

Electroencefalograma: Registro gráfico de la actividad bioeléctrica del cerebro. 36

Filtros digitales: Tipo de filtros que opera sobre señales discretas con el objetivo de atenuar o resaltar características determinadas de la señal. 61

Hipnograma: Representación gráfica de las fases de sueño determinadas por una polisomnografía description. 65, 70–72

Máquina de Vectores de Soporte: Algoritmo de aprendizaje supervisado que se puede emplear para resolver problemas de clasificación o regresión. 74

Ondas cerebrales: Actividad eléctrica sincronizada producida en el cerebro producto de la interacción entre neuronas. 34, 62

Polisomnografía: Registro de múltiples canales como la actividad cerebral, muscular, cardíaca, ocular, entre otros, mientras una persona duerme. 70, 71, 80

Pulsos binaurales: Ilusión auditiva que se genera cuando el cerebro se ve sometido a 2 tonos con frecuencias ligeramente distintas . 33, 43, 44, 48, 50, 52, 54, 58

Red Neuronal Artificial: Modelo computacional inspirado en las redes neuronales biológicas comunmente utilizado para resolver problemas de clasificación. 74