Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Electrónica y Automática Industrial

Trabajo Fin de Grado

"Control de la mano robot Inmoov-SR mediante casco NeuroSky Mindset"

Autor: Antonio Hernández Martínez

Tutor/es: Rafael Barea Navarro

2017

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

"Control de la mano robot Inmoov-SR mediante casco NeuroSky Mindset"

Autor: Antonio Hernández Martínez
Tutor: Rafael Barea Navarro

Tribunal:

Presidente: J. Antonio Jiménez Calvo

Vocal 1.º: Manuel Ocaña Miguel

Vocal 2.º: Rafael Barea Navarro

Fecha de Depósito:

AGRADECIMIENTOS

A mi familia, en especial a mi madre, por haberme dado la oportunidad de llegar hasta aquí

A mi pareja, por haberme apoyado en este periodo de mi vida

A mis compañeros de universidad con los que he compartido estos años

Índice

Índice de figuras

Índice de tablas

I. Resumen

En este trabajo el objetivo es conseguir controlar los movimientos de apertura y cierre de la mano robot InMoov-SR conectada al brazo IRB120 de ABB mediante señales EEG, recogidas por medio del casco NeuroSky Mindset.

Las señales son recogidas cuando el sujeto está en estado basal y cuando realiza movimiento con su mano y son procesadas con la ayuda de Matlab para de esta manera conseguir establecer las señales de control necesarias para activar la apertura o el cierre de la mano.

Finalmente, se diseñará una interfaz gráfica en Matlab que permitirá conectar este programa con el software de ABB RobotStudio, para poder así controlar el brazo robot.

Palabras Clave: NeuroSky Mindset, Matlab, IRB120, InMoov, RobotStudio.

II. Abstract

The aim of this project is to be able to control the opening and closing movements of robot InMoov-SR's hand which is connected to the arm IRB120 of ABB through EEG signals, collected through NeuroSky Mindset helmet.

The signals are received when the subject is in baseline state and whenever they make a movement with their hand, these movements are processed with the help of MATLAB in order to set the control signals required to activate the opening or closing of the hand.

Finally, a graphical interface will be designed in Matlab which will allow to connect this program with the software of ABB RobotStudio, to be able to control the robot arm.

Keywords: NeuroSky Mindset, Matlab, IRB120, InMoov, RobotStudio.

III. Memoria

1. Introducción

La electroencefalografía (EEG) es el registro y evaluación de los potenciales eléctricos generados por el cerebro y obtenidos por medio de electrodos situados sobre la superficie del cuero cabelludo. El estudio de las ondas cerebrales mediante electroencefalografía se remonta al año 1870 cuando los médicos militares Fritsch y Hitzig observaron que al estimular mediante corriente galvánica determinadas áreas laterales de cerebros descubiertos (de algunas de las bajas de la batalla de Sedán) se producían movimientos en el lado opuesto del cuerpo. En 1913, Prawdwicz-Neminski registró lo que llamó «electrocerebrograma» de un perro, siendo el primero en intentar clasificar semejantes observaciones. [1]

Gracias al EEG se ha podido observar que existe una relación entre los movimientos realizados por un usuario y la respuesta producida en las señales cerebrales y, de esta manera, se han podido desarrollar los interfaces cerebro computadora (BCI por sus siglas en inglés) que permiten a un sujeto comunicarse con un dispositivo electrónico sin necesidad de emplear sus manos u ordenes de voz. Esto supone un avance en el ámbito de la ingeniería biomédica, permitiendo mejorar la calidad de vida de las personas que sufren algún tipo de discapacidad.

1.1. Estado del arte

En este apartado se habla de los proyectos más relevantes que utilizan BCI en la actualidad, centrándose en aquellos en los que el objetivo de esa comunicación entre el cerebro y el dispositivo electrónico sea que personas que hayan sufrido la perdida de algún miembro de su cuerpo, o con movilidad reducida, puedan controlar una prótesis mediante estos sistemas:

I) Proyecto "*Devalhand*". Desarrollado por la Universitat Jaume I. En este proyecto se analizan diferentes diseños de manos antropomórficas, se estudian los modos de agarre más habituales de la mano humana en la vida diaria y se proponen diseños de manos antropomórficas incluyendo sus actuadores y sensores, así como el sistema de control, proponiendo métodos poco invasivos que permitan al paciente controlar la prótesis con facilidad tanto en la fase de entrenamiento como en la de uso, mejorando si es posible las prestaciones de manipulabilidad. [7]

II) Proyecto "Control de una silla de ruedas a través de un sistema BCI basado en la discriminación de dos tareas mentales". Desarrollado por el departamento de electrónica de la universidad de Málaga. En este trabajo se presenta una propuesta de control de una silla de ruedas a través de una interfaz cerebro computadora mediante la discriminación de sólo dos tareas mentales. Con el paradigma de control propuesto se pretende minimizar los porcentajes de error en la clasificación de estados mentales, proporcionando a su vez un conjunto suficiente de comandos que permita dirigir la silla de ruedas en todas direcciones (avanzar, retroceder, girar a la derecha, girar a la izquierda y parar). [8]

III) Prótesis robótica "LUKE" desarrollada por DARPA (Agencia de Proyectos de Investigación Avanzados de Defensa de Estados Unidos). Funciona mediante unos electrodos situados en el miembro amputado que recogen las señales eléctricas que envía el cerebro a los músculos del usuario. De esta manera, es capaz de distinguir si el usuario ha doblado o tensado el brazo y, por tanto, es capaz de adaptar su posición. Todo ello sin necesidad de botones, mandos o cualquier otro ajuste manual que sí requieren las prótesis convencionales. [9] [10]

IV) Proyecto "Interfaz cerebro computador basado en señales EEG para el control de movimiento de una prótesis de mano usando ANFIS". Proyecto llevado a cabo por el grupo de investigación GI2B del instituto tecnológico metropolitano y por el grupo de investigación GEA de la institución universitaria Salazar y Herrera. El objetivo de este proyecto es plantear una metodología de clasificación de señales recogidas mediante EEG para el control del movimiento de una prótesis basándose en el sistema de inferencia neuro-difuso adaptativa o sistema ANFIS. [11]

V) Prótesis Youbionic creada por Federico Ciccarese en 2017. La prótesis Youbionic es una mano biónica que puede ser impresa completamente en 3D, está controlada por un microprocesador Arduino y, al sentir el musculo en movimiento, permite al usuario realizar movimientos de agarre mejorando su calidad de vida. [12] [13]

Además de estos ejemplos, DARPA ha creado prótesis conectadas al cerebro que son capaces de hacer que el paciente sienta la extremidad como si fuese la suya mediante sensores de presión, lo que le permite también una mayor sensibilidad a la hora de coger objetos. También se ha conseguido, mediante chips implantados en el cerebro, que un joven tetrapléjico vuelva a tener movilidad en las manos y los dedos a partir de la lectura de los impulsos eléctricos que emite su encéfalo. [14]

1.2. Objetivos del TFG y campos de aplicación

El objetivo principal de este TFG es controlar en tiempo real los movimientos de la mano del robot Inmoov-SR, conectada al extremo del brazo robot IRB120 de ABB, mediante un sistema BCI "*Brain Computer Interface"* (interfaz cerebro computadora). El sistema se inicia con la adquisición de los datos procedentes del casco NeuroSky y el procesamiento de estos en Matlab para su posterior uso en el brazo robot IRB120 de la empresa ABB, que utiliza el software RobotStudio. Es en el IRB120 donde se tiene conectada la mano robot del InMoov SR. La comunicación entre Matlab y RobotStudio se lleva a cabo mediante un socket TCP/IP. Se ha creado también una interfaz gráfica en Matlab que facilita al usuario el uso de este sistema.

En la figura 1.1 se puede observar de forma esquematizada el objetivo a lograr en este proyecto:

Figura 1.1: Diagrama de bloques del sistema creado en este proyecto.

Para poder controlar los movimientos de la mano robot, se ha realizado un estudio en un grupo de sujetos para detectar cambios en la señal adquirida con el casco NeuroSky debidos a cambios producidos en la banda gamma de las ondas cerebrales como consecuencia de la realización de movimiento con la mano derecha. Esta GBA inducida (gamma-band activity) puede ser utilizada como una señal binaria de control, permitiéndonos abrir y cerrar la mano robot InMoov.

En la realización de este proyecto han colaborado las siguientes disciplinas: ingeniería biomédica, ingeniería electrónica, robótica, biología, neurología, biotecnología.

2. Fundamentos Fisiológicos

El encéfalo humano ha sido descrito como el sistema más complejo del universo conocido. Está compuesto por neuronas, mielina y vías nerviosas, y constituye el núcleo del sistema nervioso central, además de ser el órgano que nos permite realizar las actividades mentales más complejas y de ser capaces de sentir el yo, es decir, de tener consciencia. Dentro de nuestro cerebro hay un conjunto de estructuras trabajando a gran velocidad y, hoy en día, este órgano sigue siendo un misterio en la mayoría de sus aspectos. [15]

El sistema nervioso central se encuentra protegido por tres membranas llamadas duramadre (membrana externa), aracnoides (membrana intermedia) y piamadre (membrana interna), conocidas como meninges. Tanto la médula espinal como el encéfalo están protegidos por una envoltura ósea, la columna vertebral y el cráneo, respectivamente.

2.1. Anatomía del encéfalo

El encéfalo humano de un adulto completamente desarrollado está formado por tres estructuras (figura 2.1) que son el cerebro, el cerebelo y el tronco encefálico.

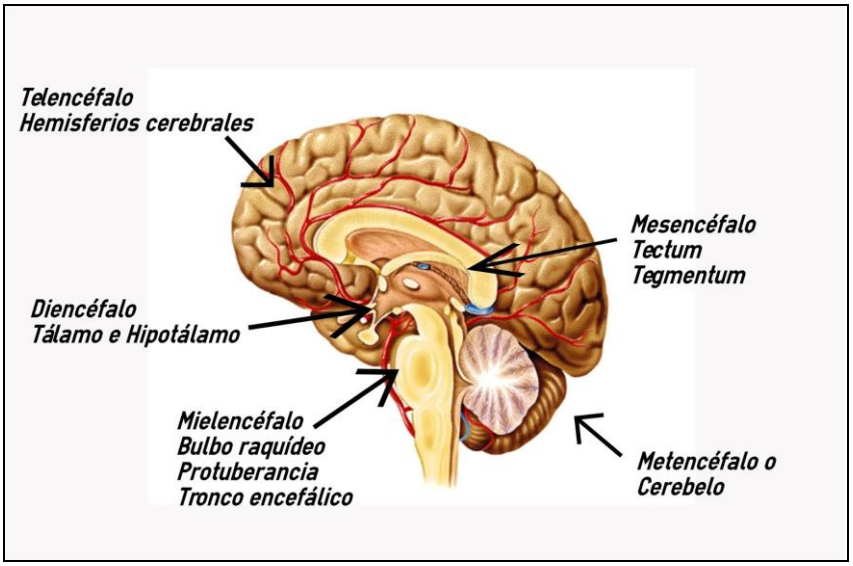

Figura 2.1: Anatomía del encéfalo

El cerebro a su vez se encuentra formado por el telencéfalo y el diencéfalo y está dividido en dos mitades, hemisferio izquierdo y derecho, que son en su mayor parte simétricas. El tronco encefálico está formado por el mesencéfalo, la protuberancia o puente de Varolio y el bulbo raquídeo (figura 2.1).

2.1.1. Estructura del cerebro

El cerebro humano está dividido principalmente en dos hemisferios, izquierdo y derecho. Aunque a lo largo de los años se han atribuido funciones bien diferenciadas a cada uno de ellos, la realidad es mucho más compleja. De manera resumida se puede atribuir al hemisferio izquierdo un pensamiento racional, es el encargado de aprender el lenguaje, de hacer operaciones matemáticas complejas y de extraer recuerdos de la

memoria, por eso es conocido como el hemisferio racional. El hemisferio derecho es el encargado de reconocer imágenes o rostros, de procesar la música y de dar sentido a lo que vemos. También juega un papel en el lenguaje, interpretando el contexto y el tono de voz de una persona. Este hemisferio es conocido como el hemisferio creativo o emocional. El hemisferio derecho e izquierdo están constantemente compartiendo información a través del cuerpo calloso que los conecta (figura 2.2).

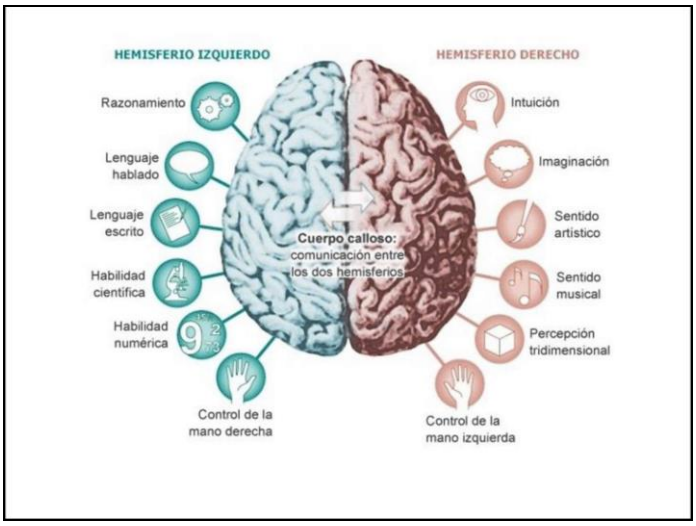

Figura 2.2: Hemisferios derecho e izquierdo del cerebro.

Cada uno de los dos hemisferios está formado por los lóbulos frontal, occipital, parietal y temporal (figura 2.3).

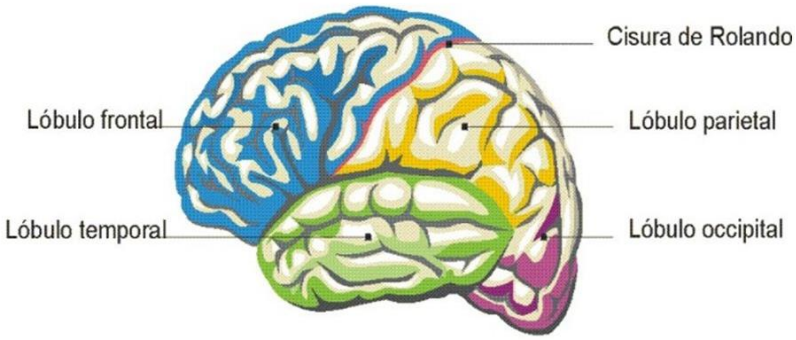

Figura 2.3: Lóbulos cerebrales

El **lóbulo frontal** es el más grande y el más estudiado del encéfalo humano. Sin él seríamos seres puramente emocionales, guiados por impulsos. Se dice que el lóbulo frontal es el que nos diferencia del resto de animales, ya que este nos permite pensar de manera abstracta, retener información transitoria relativa a un problema que debe ser resuelto en tiempo real e imaginar planes y estrategias futuras y sus posibles consecuencias. En la parte del lóbulo frontal más cercana al surco de Rolando, que lo separa del lóbulo parietal, existe un área denominada córtex motor que al ser estimulada mediante descargas eléctricas, producía movimiento en los pacientes, relacionando así el lóbulo frontal con el movimiento voluntario de las distintas partes del cuerpo. [20] [24]

El **lóbulo parietal** situado bajo el hueso craneal que le da nombre y entre el lóbulo frontal y el occipital. De modo resumido se puede decir que juega un papel importante, especialmente la integración y el procesamiento de la información sensorial, el procesamiento de la información simbólica (en la que se incluyen los procesos relacionados con el lenguaje y su utilización) y el procesamiento de la información numérica. [21]

El **lóbulo occipital** ubicado en la parte del cerebro más cercana a la nuca. El proceso que define mejor la utilidad del lóbulo occipital es el procesamiento de la información visual. [22]

El **lóbulo temporal** situado en el lateral inferior del encéfalo, aproximadamente a la altura de los oídos en estrecho contacto con el lóbulo occipital. Se trata del lóbulo con mayor conexión con el sistema límbico teniendo gran influencia en las emociones y estados de ánimo. El lóbulo temporal es fundamental para el desarrollo de habilidades como el habla o la percepción auditiva, además de estar muy vinculado a la afectividad, la memoria y el reconocimiento. [23]

Existe un quinto lóbulo denominado **ínsula** que no es visible desde fuera del cerebro. Es el centro de conexión entre el sistema límbico y el neocórtex, encargado del razonamiento humano.

A su vez, en el cerebro se pueden encontrar las estructuras mencionadas en el apartado anterior, de una manera mayormente simétrica, cuyas funciones se explican a continuación.

El **telencéfalo** es la parte del cerebro que es más fácil de ver a simple vista, ya que ocupa la mayor parte de la superficie del encéfalo. Sus componentes son la corteza cerebral, los ganglios basales, el sistema límbico, el hipocampo y la amígdala.

- La **corteza cerebral** o **córtex** es la parte del cerebro rugosa y llena de pliegues que cubre el resto del encéfalo y es el área en la que se integra la información necesaria para llevar a cabo los procesos mentales más complejos. En el córtex es donde se encuentran la mayoría de las neuronas del cerebro humano.

- Los **ganglios basales** son un grupo de estructuras situadas por debajo de la corteza cerebral y distribuidos de forma simétrica bajo cada uno de los hemisferios. Es la parte del cerebro que nos permite ser capaces de realizar movimientos relativamente complejos y precisos de manera fácil y casi automática. Además, nos permite perfeccionar cadenas de movimientos que hemos realizado previamente hasta llegar a dominarlas.

- El **sistema límbico** es un conjunto de estructuras cuyos límites son bastante difusos. Es el encargado de regular nuestras emociones y es lo que se considera “cerebro emocional” en contraposición al “cerebro racional” que correspondería a las zonas ocupadas por el córtex; sin embargo, el córtex y el sistema límbico no pueden funcionar de manera correcta por separado.

- El **hipocampo** es una estructura alargada situada en la parte interna de los lóbulos temporales que se encuentra presente en las formas de mamíferos más antiguas. Se encarga de la recuperación de recuerdos y el aprendizaje.

- La **amígdala** es un conjunto de neuronas que se agrupan en la cara interna del lóbulo temporal de cada uno de los hemisferios. Forma parte del sistema límbico y es la encargada de relacionar estados emocionales con situaciones vividas, de esta manera es la que nos permite saber cómo reaccionar ante una situación con una emoción u otra.

El **diencéfalo** o **cerebro medio** está formado por las estructuras tálamo, hipotálamo subtálamo y epitálamo y es el encargado de procesar las emociones como la tristeza o la alegría.

- El **tálamo** es la parte más grande del diencéfalo, y en él se integra por primera vez toda la información que nos llega a través de los sentidos a excepción del olfato. También es el encargado de hacer que el sistema nervioso autónomo reaccione rápidamente ante estímulos que puedan suponer la presencia de peligro.

- El **hipotálamo,** situado debajo del encéfalo. Su misión es regular nuestro organismo encargándose de mantener la temperatura corporal, el ritmo respiratorio, las hormonas en la sangre, etc. También es responsable de la aparición de la sed y el hambre.

2.1.2. Estructura del tronco encefálico

En la figura 2.4 se pueden observar las partes que componen el tronco encefálico y las funciones de cada una de ellas.

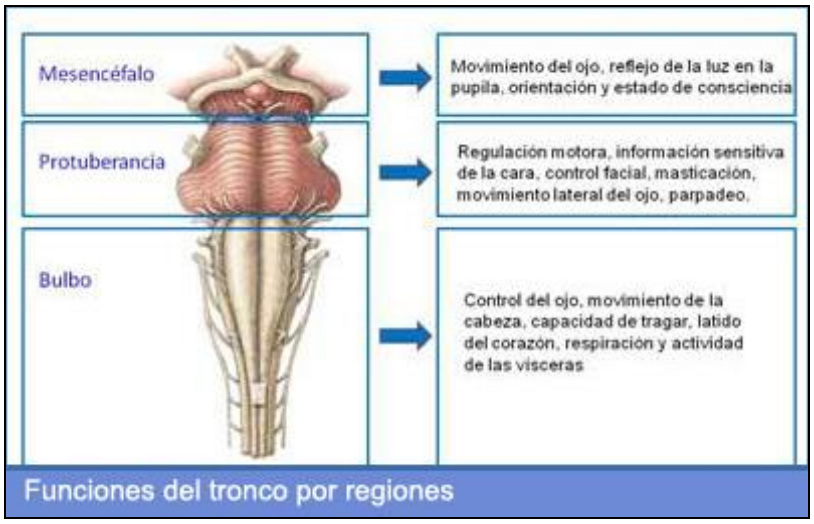

Figura 2.4: Anatomía del tronco encefálico

El **mesencéfalo** es la parte del tronco del encéfalo que queda justo debajo del diencéfalo. Comunica el tallo cerebral con las estructuras cerebrales e interviene en

el mantenimiento de los procesos automáticos que nos permiten sobrevivir, está dividido en el tectum y el tegmentum

La **protuberancia** o **puente de Varolio**, es una de las estructuras más importantes en el cerebro. Un pequeño golpe en esta estructura podría suponer entrar en coma o incluso la muerte, ya que es la parte más grande del tronco encefálico y este a su vez, es el encargado de mantener procesos automáticos que nos mantienen con vida.

El **bulbo raquídeo** es la parte inferior del tronco encefálico cuyas funciones son similares a las del mesencéfalo y el puente de Varolio. Además, es el enlace entre el encéfalo y la médula espinal. En esta parte se encuentra la decusación de las pirámides, que es donde los fajos de fibras nerviosas de los dos se entrecruzan para pasar de un lado a otro explicando de esta manera que la información del lado izquierdo del cuerpo llegue al hemisferio derecho del cerebro y viceversa.

2.1.3. Cerebelo

El **cerebelo** (figura 2.5) es una de las partes del cerebro con una mayor concentración de neuronas y entre sus muchas funciones se incluyen la regulación y monitorización de movimientos complejos que requieren una cierta coordinación o el mantenimiento del equilibrio al estar de pie y caminar.

Figura 2.5: Localización del cerebelo dentro del cráneo.

Como se puede observar muchas de las funciones del cerebro no se encuentran situadas en una sola zona, sino que están repartidas en muchas de ellas y es la coordinación entre todas las partes que forman el encéfalo lo que da lugar a su correcto funcionamiento.

2.2. La neurona

Las neuronas (figura 2.6) son las principales células del sistema nervioso, cuya función es encaminar y procesar la información en el organismo. Las neuronas comparten las mismas características y cuentan con la misma composición que otras células, pero el aspecto electroquímico les permite transmitir las señales a través de grandes distancias.

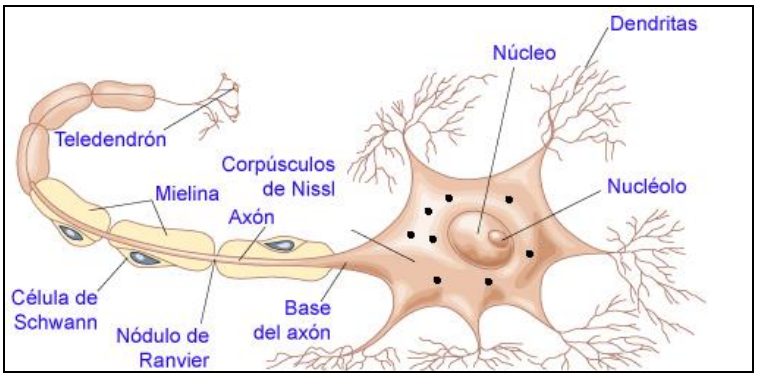

Figura 2.6: Anatomía de una neurona

Las neuronas están formadas por 3 partes principales:

El cuerpo celular o soma. Es considerada la parte principal de la neurona y en ella se encuentran todos los componentes necesarios de la célula. Si la célula del cuerpo muere, también fallece la neurona completa.

El axón es la parte más larga de la neurona. Comienza en la célula y transmite el mensaje electroquímico (potencial de acción) que contiene la información. Dependiendo del tipo de neurona, los axones pueden estar cubiertos por una delgada capa de mielina. La mielina está formada por grasa y proteína. Su función es ayudar a que la transmisión del impulso nervioso sea veloz a lo largo del axón**.**

Las dendritas, encargadas de crear conexiones, permiten que la neurona se comunique con otras células o reconozca su ambiente. Las dendritas pueden estar localizadas en uno o a ambos lados de la neurona.

En la figura 2.6 podemos observar también los nódulos de Ranvier, encargados de trasladar el impulso nervioso entre neuronas a mayor velocidad, las células de Schwann, encargadas de crear la vaina de mielina del axón o los teledendrones, que conectan con las dendritas de la siguiente neurona.

Para ser funcional, una neurona necesita compartir su información con otras neuronas u otras células y para compartir esta información las neuronas están conectadas entre sí. La conexión que forma una neurona con la siguiente se denomina sinapsis neuronal, un espacio muy activo que existe entre una neurona y otra célula (figura 2.7).

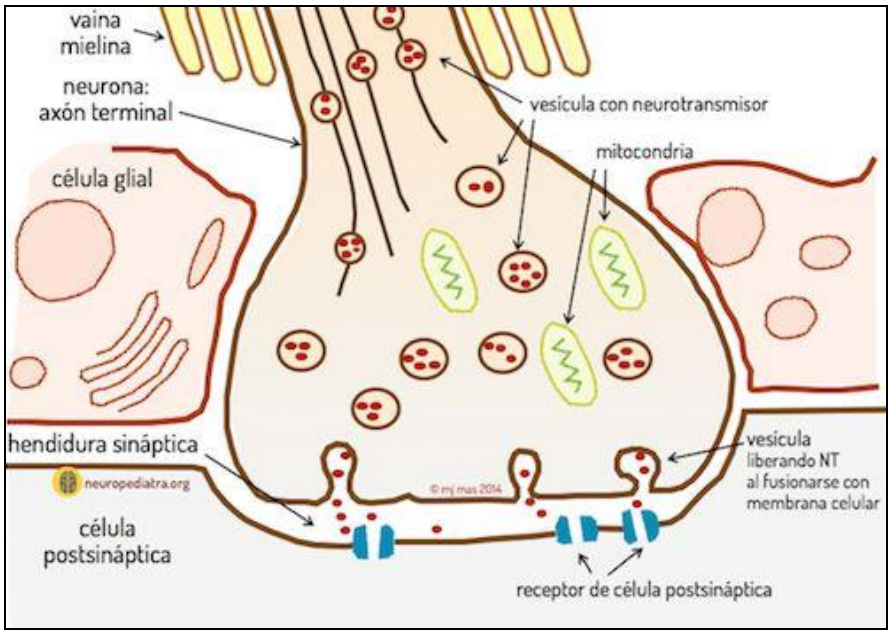

Figura 2.7: Sinapsis neuronal

Las neuronas reciben la información en forma de un impulso electroquímico o potencial de acción a través de las dendritas, esta es transmitida hasta el cuerpo celular en forma de señales eléctricas denominadas impulsos nerviosos y es aquí donde dicha información es procesada. La señal de respuesta obtenida es transmitida por el axón hasta las dendritas de la siguiente neurona, formando así las redes neuronales que están presentes en todo el sistema nervioso. Las señales de respuesta son conocidas como los procesos de excitación o inhibición de las neuronas. Además, estos impulsos eléctricos también conocidos como potencial de acción son de carácter unidireccional, llegando siempre a la neurona a través de sus dendritas y saliendo de ellas a través del axón (figura 2.8). De esta manera una neurona puede, por ejemplo, mandar un impulso eléctrico que haya procesado previamente, hasta un componente motor que realizará movimiento, como puede ser el movimiento voluntario de la mano.

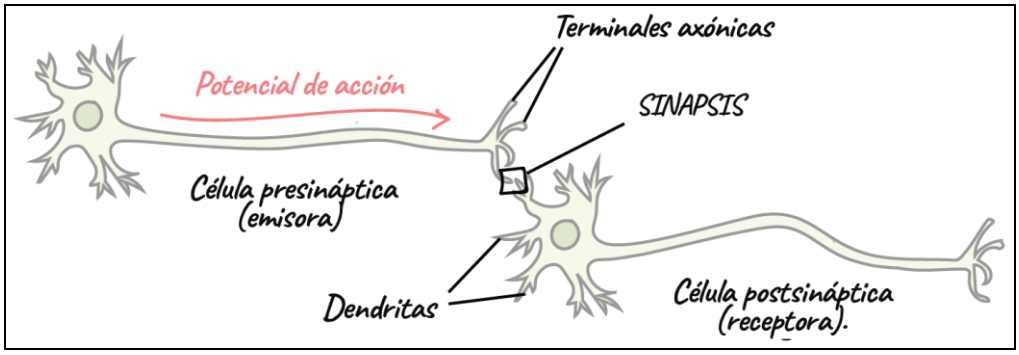

Figura 2.8: Sinapsis entre dos neuronas

Las ondas eléctricas que transmiten las neuronas son originadas como consecuencia de un cambio transitorio de la permeabilidad en la membrana plasmática de su célula. Su propagación se debe a la existencia de una diferencia de potencial o potencial de membrana. Esta diferencia de potencial es debida a que la concentración de iones en el

interior de las células es distinta a la del medio en el que se encuentran. La membrana celular separa el interior del exterior de la célula y posibilita que las concentraciones de sales sean distintas. Si se hace permeable las concentraciones tenderán a igualarse, los iones más abundantes a un lado pasarán como un torrente de agua hacia el otro lado produciendo una corriente eléctrica.

Las neuronas son células especializadas en transmitir electricidad y para ello modifican la permeabilidad de su membrana en el axón, permitiendo la entrada y salida de sales y con ello la transmisión del impulso eléctrico. Al llegar el impulso eléctrico al final del axón, estimula la liberación a la hendidura sináptica de las sustancias químicas elaboradas en el interior de la neurona, llamadas neurotransmisores, que son las que contienen la información (figura 2.7). Existen diferentes tipos de neurotransmisores y cada neurona está especializada en sintetizar un determinado tipo.

La carga de una célula inactiva se mantiene en valores negativos (el interior respecto al exterior) y varía dentro de unos estrechos márgenes. Cuando el potencial de membrana de una célula excitable se despolariza más allá de un cierto umbral (de -65 mV a -55 mV) la célula genera un potencial de acción. Un potencial de acción (figura 2.9) es un cambio muy rápido en la polaridad de la membrana que pasa de negativo a positivo ("despolarización") y vuelta a negativo ("repolarización") en un ciclo que dura unos milisegundos.

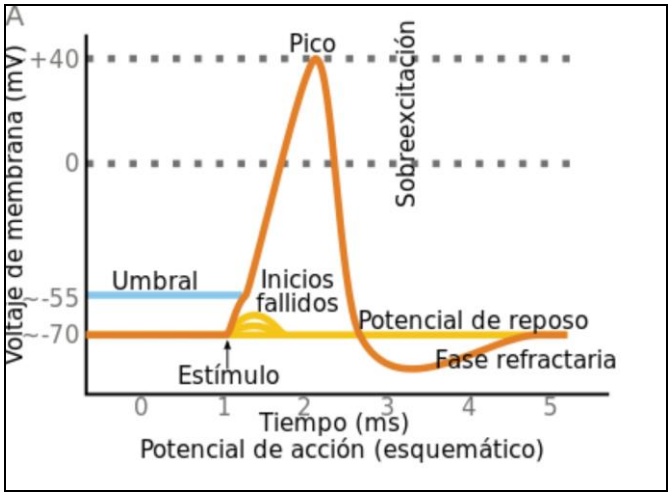

Figura 2.9: Potencial de acción

El potencial de acción se mide mediante técnicas de registro de electrofisiología, que estudian las propiedades eléctricas de los tejidos y las células, como puede ser el cambio de potencial o de corriente eléctrica en su interior. En concreto, para medir la actividad eléctrica producida en el cerebro se emplea la técnica conocida como electroencefalografía.

3. Electroencefalograma (EEG)

La Electroencefalografía es el registro y evaluación de los potenciales eléctricos generados por el cerebro y obtenidos por medio de electrodos situados sobre la superficie del cuero cabelludo. El electroencefalograma (EEG) es el registro de la actividad eléctrica de las neuronas del encéfalo. Dicho registro posee formas muy complejas que varían mucho con la localización de los electrodos y entre individuos. Esto es debido al gran número de interconexiones que presentan las neuronas y por la estructura no uniforme del encéfalo. [1]

3.1. Ondas cerebrales

La actividad de las neuronas del cerebro no es absolutamente caótica, sino que sigue una lógica muy compleja en la que puede notarse cómo diferentes neuronas disparan señales eléctricas casi al mismo tiempo de un modo continuado. Esta frecuencia constituida por la actividad de varias neuronas forma lo que se conoce como ondas cerebrales, patrones de activación que, a diferencia de lo que ocurre con la frecuencia de activación de una sola neurona, son lo suficientemente potentes y claros como para poder ser registrados mediante la electroencefalografía. [30]

Las ondas cerebrales (figura3.1) pueden ser clasificadas en diferentes tipos según su frecuencia y amplitud en ondas delta, theta, alfa, beta y gamma.

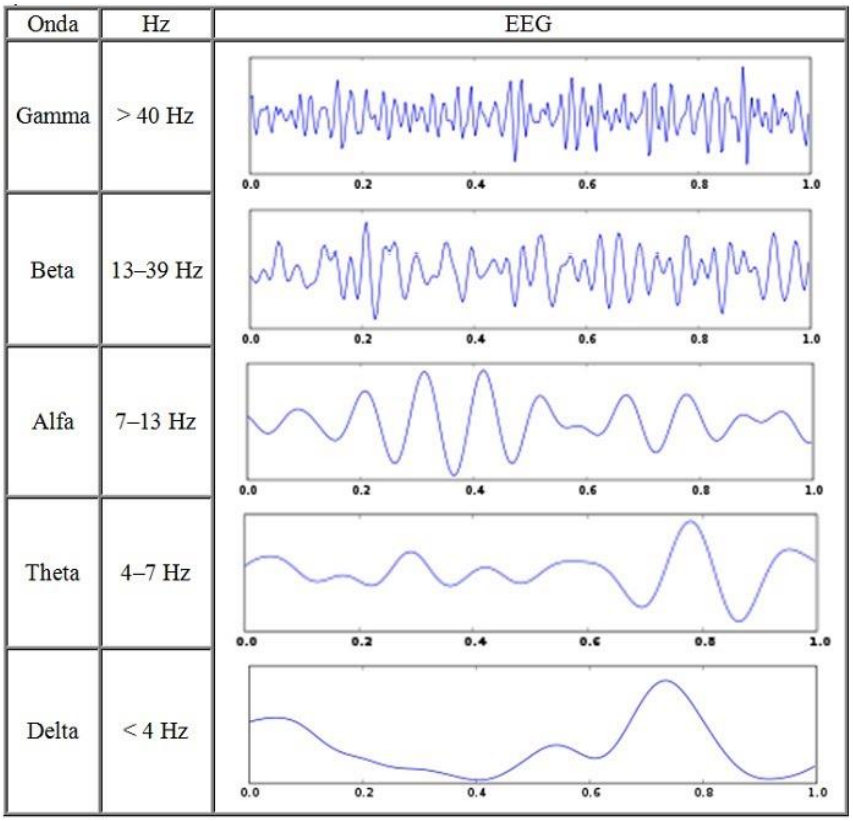

Figura 3.1: Ondas cerebrales

Las **ondas delta** son las de mayor amplitud, entre 20 y 200 µV, y por tanto menor frecuencia, de 0.1 a 4 Hz. Normalmente están asociadas con etapas de sueño profundo sin soñar y en casos de daño cerebral o coma.

Las **ondas theta** están presentes en niños y en adultos en un estado de sueño ligero o en periodos de mucho estrés emocional y ansiedad. Su frecuencia varía entre 3.5 y 7.5 Hz y su amplitud entre 20 y 100 µV siendo las de mayor amplitud tras las ondas delta. Están asociadas a los estados de calma profunda, relajación e inmersión en los recuerdos y las fantasías, y también con la etapa de sueño REM, que es aquella en la que soñamos. Por consiguiente, cuando aparecen estas ondas se estima que sí hay consciencia o que es muy probable que la haya.

Las **ondas alfa** que presentan más frecuencia que las ondas theta, de 8 a 13 Hz y con una amplitud que oscila entre los 20 y los 60 µV. Siguen estando relacionadas con los estados de relajación o calma profunda, sin estar relacionadas con el sueño profundo como ocurre con las ondas theta.

Las **ondas beta,** con una frecuencia que va de los 12 a 33 Hz y una amplitud de entre 2 y 20 µV Están relacionadas con etapas de sueño nulo. Estas ondas aparecen cuando se están realizando acciones que requieren permanecer en un cierto estado de alerta y de gestión ágil de la atención o en momentos de gran concentración

Las **ondas gamma,** son el tipo de ondas cerebrales con una mayor frecuencia, de 30 a 100 Hz, aunque su presentación más habitual es por encima de los 40 Hz y, a su vez, son las ondas de menor amplitud, inferior a 2 µV. Aparecen en estados de vigilia y se cree que su presencia está relacionada con la aparición de la consciencia. Estas ondas se asocian también a momentos de gran actividad mental, de extrema atención y concentración y, además, son las ondas que mejor registran los movimientos motores y por tanto es en estas ondas en las que se centra este proyecto, tratando de detectar cambios en la banda gamma como consecuencia del movimiento de la mano del sujeto.

3.1.1. Banda Gamma

La banda Gamma es la banda de frecuencias que abarcan las ondas cerebrales gamma desde 30 hasta los 100 Hz. En esta banda se ha observado una actividad inducida (GBA inducida) como respuesta ante estímulos sensoriales externos o durante la realización de tareas motoras que no se encontraba al inicio del movimiento. [31]

En este proyecto se ha extraído la actividad de la banda gamma baja, de 30 a 60 Hz, durante la realización de movimiento por parte del sujeto, y se ha comparado con un EEG realizado cuando el sujeto se encontraba en estado basal. La GBA se ha extraído como densidad de potencia espectral, utilizando métodos no lineales como la transformada de Fourier. La manera de calcular la GBA inducida es explicada de manera más detallada en el capítulo 8 de este proyecto.

Además, se ha demostrado en un estudio con un grupo de pacientes que cuando estos realizan movimiento con su mano, consiguen aumentar la actividad en la banda gamma, pero pasado un tiempo realizando movimiento, estos pacientes no necesitan realizar movimiento para activar la banda gamma, únicamente necesitan reproducir en su cerebro el movimiento realizado. [2]

3.2. Tipos de EEG

Dependiendo del objetivo de la prueba, se pueden realizar varios tipos de electroencefalograma en el paciente, entre los cuales se encuentran los siguientes.

- **EEG estándar**: Para esta prueba no es necesaria ninguna preparación previa, la única observación es la limpieza del cuero cabelludo el día anterior y no usar ni tintes, ni acondicionadores, ni laca. Esta prueba se hace en el más absoluto reposo, ya que cualquier movimiento puede alterar los resultados y dura 30 minutos. Se registra la actividad cerebral de la persona en vigilia, es decir, estando completamente despierta y se pueden realizar activaciones del cerebro mediante fotoestimulación o hiperventilación. Este examen se realiza en pacientes en los que se esté estudiando una enfermedad diferente a la epilepsia.

- **EEG en privación de sueño**: Para este EEG es necesario que los pacientes adultos permanezcan toda una noche sin dormir. No pueden tomar ningún medicamento, ni consumir café, bebidas cola, energéticas, cigarros o alcohol. En el caso de los niños pequeños, se hacen privaciones parciales de sueño, duermen sólo cuatro horas y luego se mantienen despiertos hasta el examen. Una vez comienza la prueba se recomienda al paciente la máxima relajación, propiciando el sueño, para que de esta forma aparezcan los trazados fisiológicos de las distintas fases del sueño, así como anomalías que no se puedan observar en un EEG estándar.

- **EEG prolongado**: La duración de este examen es superior a 30 minutos. Habitualmente va desde 1 hora hasta la cantidad de horas que el médico solicitante estime conveniente. Está indicado para el diagnóstico de Epilepsias, para identificar el tipo de Epilepsia y cuando existe la sospecha de un estado epiléptico. [33]

- **EEG continuo**: Habitualmente para realizar este examen el paciente debe estar hospitalizado por varios días (de 4 a 5 días) y al mismo tiempo se hace un registro de video simultáneo. El objetivo es afinar el diagnóstico del tipo de Epilepsia o realizar el diagnóstico diferencial con otras enfermedades que pudiesen “simular” crisis epilépticas. También se utiliza para realizar ajustes de terapia farmacológica y como evaluación preoperatoria en el caso de pacientes candidatos a cirugía de la Epilepsia. [33]

- **Holter EEG 24 o 48 horas**: Los electrodos se instalan en la clínica, pero posteriormente el paciente vuelve a su domicilio y realiza una vida normal. El paciente debe tomar nota de cualquier hecho relevante que le suceda durante el

tiempo que dure la prueba. Se usa principalmente para confirmar el diagnóstico de la epilepsia.

El EEG también puede ser utilizado como prueba para determinar la muerte cerebral (cese completo e irreversible de la actividad cerebral) en aquellos pacientes que no sigan tratamientos depresores del sistema nervioso central, ya que estos interfieren con la lectura del EEG. Esta prueba es muy importante debido a que, para poder extraer los órganos para cuyo trasplante se precisa viabilidad de los mismos en aquellos pacientes donadores, es necesario previamente declarar la muerte cerebral.

3.3. Tipos de electrodos y montajes de EEG

Para poder captar las señales eléctricas procedentes de nuestro cerebro existen una variedad de sensores que dependiendo de su colocación pueden ser clasificados en superficiales, basales o quirúrgicos. Los electrodos superficiales se colocan sobe el cuero cabelludo, los basales sobre la base del cráneo y los quirúrgicos sobre la superficie de la corteza o en localizaciones cerebrales profundas.

El registro de la actividad cerebral recibe distintos nombres dependiendo del tipo de electrodos utilizados, siendo un electrocorticograma (ECoG) si los datos han sido recogidos utilizando electrodos quirúrgicos situados en la corteza cerebral, estéreo electroencefalograma (E-EEG) si los electrodos utilizados son de aplicación profunda y electroencefalograma (EEG) si son electrodos superficiales o basales. [41]

Dentro de los electrodos superficiales podemos encontrar los siguientes:

- **Electrodos adheridos** (figura 3.2), pequeños discos metálicos que se adhieren con pasta conductora sobre el cuero cabelludo.

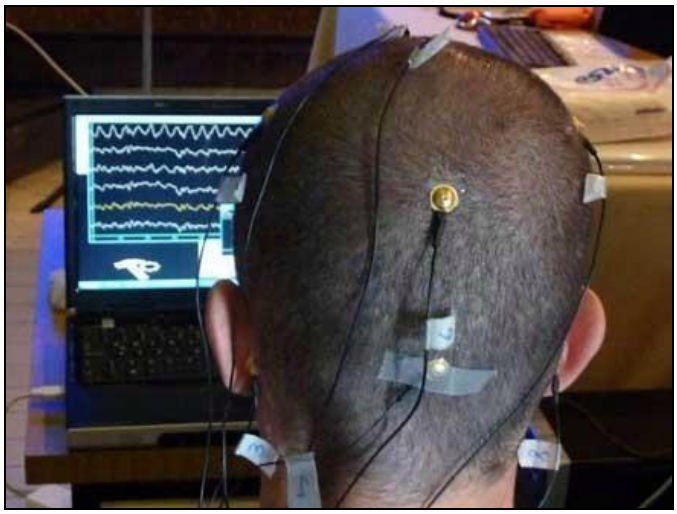

Figura 3.2: Electrodos adheridos

- **Electrodos de contacto** (figura 3.3), pequeños tubos de plata clorurada roscados a soportes de plástico que en su extremo de contacto se colocan una almohadilla que se humedece con solución conductora.

Figura 3.3: Electrodos de contacto

- **Electrodos en casco de malla** (figura 3.4), que consisten en un casco elástico en el que los electrodos están colocados, siendo este último el que permite una mayor comodidad para el paciente y el que presenta una mayor precisión en la colocación de los electrodos, puesto que estos ya están colocados en el casco previamente.

Figura 3.4: Electrodos en casco de malla

Además, para realizar un electroencefalograma existen diferentes tipos de montaje. La colocación de los electrodos en dichos montajes está sujeta a un sistema internacional o sistema 10-20 (figura 3.5), denominado así porque los electrodos están espaciados entre el 10 % y el 20 % de la distancia total entre puntos reconocibles del cráneo (Inion, Nasion y Vertex). [42]

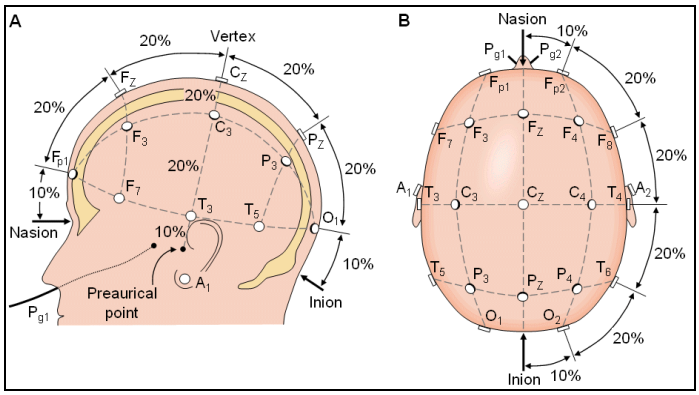

Figura 3.5: Montaje de electrodos según el sistema internacional 10-20

Las señales se registran a partir de una serie de electrodos colocados cada uno en una de estas posiciones. Cada canal EEG mide la diferencia de potencial eléctrico entre dos electrodos que se conoce como derivación. Un estudio electroencefalográfico adecuado requiere un mínimo de 10 canales. Dentro de este montaje existen dos tipos de registros, bipolar y monopolar.

Los registros bipolares utilizan parejas de electrodos. Tanto el situado en la posición uno como el electrodo situado en la posición dos, registran actividad cerebral y la diferencia entre los dos puntos es lo que va al amplificador para su registro (figura 3.6).

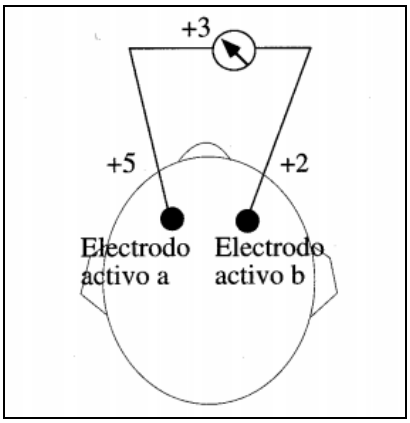

Figura 3.6: Registro bipolar

En los registros monopolares se toma la señal de cada uno de los electrodos independientemente de la de los demás. En esta situación el electrodo de registro se llama electrodo activo y el segundo cable de entrada al equipo se toma de un electrodo llamado de referencia (figura 3.7). [41]

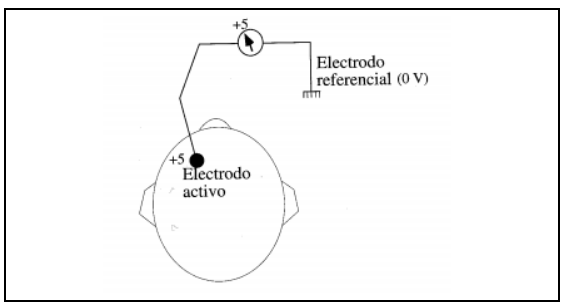

Figura 3.7: Registro monopolar

4. Sistemas *brain-computer interface* (BCI)

Este proyecto, aunque hace uso de un sistema BCI, no se centra tanto en la creación y uso de este tipo de sistemas, sino en la parte relacionada con el procesado de la señal y su transformación en una señal de control, necesaria para mandar comandos a la mano del robot InMoov, por tanto, en este apartado se explicará de manera resumida en que consisten este tipo de sistemas.

4.1. Tipos de sistema

Los sistemas BCI (figura 4.1) son los encargados de realizar la conexión del cerebro con los distintos dispositivos electrónicos. Estos sistemas captan las señales procedentes del cerebro mediante sensores, procesan esta señal, y nos permiten abrir un canal de comunicación entre el dispositivo electrónico y nuestro cerebro, de manera que se puede interactuar con él únicamente haciendo uso de nuestra actividad cerebral, sin necesidad de utilizar el sistema motor. En otras palabras, un sistema BCI es aquel capaz de traducir las intenciones del usuario, generadas por nuestro cerebro, en órdenes que son interpretadas y ejecutadas por una máquina u ordenador.

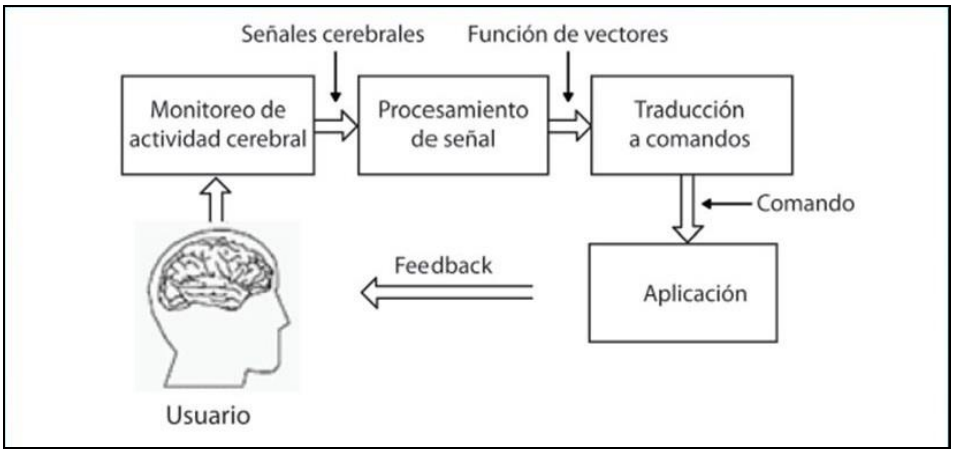

Figura 4.1: Sistema BCI

Los BCI se pueden clasificar en exógenos o endógenos, dependiendo de si es necesaria o no una estimulación externa para producir la actividad cerebral.

En los sistemas exógenos es necesaria estimulación externa (destellos de luz, señales acústicas, etc.) para conseguir actividad cerebral. Por ejemplo, la prueba del potencial evocado es de carácter exógeno, ya que mide la velocidad de respuesta de nuestro cerebro ante un estímulo externo que puede ser visual, auditivo, somatosensorial, o motor.

Por otro lado, están los sistemas endógenos, en los que es el propio sujeto el que modifica su actividad cerebral, y por tanto dependen únicamente de su capacidad de control. En este caso el sujeto debe someterse a un entrenamiento previo para mejorar dicho control.

Dentro de los sistemas endógenos existen los que están basados en potenciales corticales lentos (SCP) y los que se basan en imágenes motoras o ritmos sensoriomotores.

Los SCP son cambios lentos de voltaje generados sobre el córtex cerebral, con una duración variable entre 0.5 y 10 segundos. Los SCP negativos se asocian típicamente con el movimiento y otras funciones que implican una activación cortical. Se ha demostrado que las personas pueden aprender a controlar estos potenciales. [44]

Los ritmos sensoriomotores son aquellos que van entre 8 y 12 Hz y entre 16 y 24 Hz, registrados sobre la zona somatosensorial y motora del córtex cerebral. Estos ritmos presentan variaciones tanto para la ejecución de un movimiento real como para la imaginación de un movimiento o la preparación al mismo. Al igual que con los SCP, estos pueden llegar a ser controlados con entrenamiento.

4.2. Tipos de cascos neuronales

A continuación, se van a explicar distintos tipos de cascos neuronales que existen en el mercado.

Hoy en día existen en el mercado una gran variedad de cascos neuronales que permiten realizar EEG. Dentro de este tipo de cascos se pueden diferenciar dos mercados diferentes, los de ámbito médico y los de uso particular.

En este apartado se hablará de los cascos de uso particular, ya que los de uso médico están destinados a aplicaciones que sufran alguna patología neuronal y son cascos de mayor coste y mayor número de canales de EEG. Como ejemplo de un casco de carácter médico podemos hablar del GES 400 (figura 4.2) utilizado para la realización de pruebas cerebrales en adultos.

Figura 4.2: Casco Neuronal GES 400

Hablando de cascos de uso particular, se pueden diferenciar dos empresas, NeuroSky y Emotiv Systems.

El primero del que se va a hablar es el casco de la empresa Emotiv Systems que se conoce como Emotiv EPOC (figura 4.3). Es un casco que cuenta con 14 electrodos y un

sensor giroscópico. Asimismo, este casco cuenta algoritmos que analizan el cerebro del usuario y trabajan en función de las características propias del mismo. Además, este casco es capaz de detectar expresiones faciales como sonrisas, risas, arqueamiento de cejas y guiños.

Figura 4.3: Casco Emotiv EPOC

Por otra parte, está el casco NeuroSky Mindwave o NeuroSky Mindset dependiendo del modelo. En concreto el casco utilizado en la realización de este proyecto es el NeuroSky Mindset del que se hablará en el capítulo 5 de esta memoria. Se ha optado por elegir este casco porque, aunque tanto el Emotiv EPOC como el Mindset cumplen con las especificaciones necesarias, el Mindset tiene un precio de unos 170€ mientras frente a los casi 400€ que cuesta el EPOC actualmente.

5. Casco NeuroSky Mindset

En este apartado se va a presentar el casco NeuroSky Mindset, utilizado en este proyecto como sistema de adquisición de datos (figura 5.1)

Figura 5.1: Casco NeuroSky Mindset

A diferencia de otros cascos de la marca NeuroSky, el casco Mindset transmite las señales recogidas mediante *bluetooth* y no mediante radiofrecuencia, y además incluye auriculares y micrófono, lo que permite que pueda ser usado como un dispositivo manos libres. Los auriculares también permiten estudiar los cambios en la señal captada por el sensor producidos por estímulos auditivos.

5.1. Características

El casco Mindset creado por la empresa NeuroSky es un preciso instrumento de observación cuya función es medir la frecuencia de nuestras ondas cerebrales para monitorizarla y mostrarla en la pantalla de nuestro ordenador. En pocos segundos es capaz de medir nuestros niveles de Atención, Relajación y Meditación. La medición se realiza mediante un sensor superficial seco colocado en la zona del cráneo denominada FP1 y tres electrodos referenciales (figura 5.2) colocados en la almohadilla izquierda del casco que hacen contacto con el lóbulo de la oreja izquierda (zona A1) realizando de esta manera un registro monopolar (figura 5.3).

Figura 5.2: Electrodos referenciales del casco NeuroSky Mindset

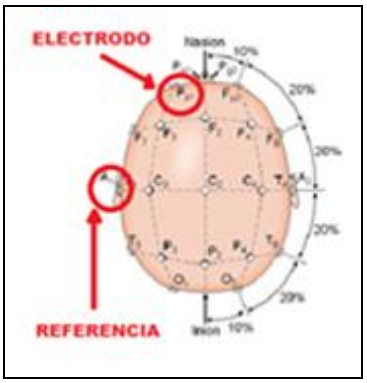

Figura 5.3: Registro monopolar del casco Mindset

El sensor frontopolar obtiene la señal procedente de nuestro cerebro y la divide en las distintas ondas cerebrales (alfa, beta, gamma, delta y theta)

5.2. Protocolo de comunicación ThinkGear

El protocolo de comunicación de cualquier producto de la marca Neurosky está instaurado en el chip ThinkGear. El sensor frontopolar recoge las señales, las envía al chip, y este las procesa y las convierte en una secuencia de datos digitales. Una vez filtradas las interferencias, las ondas cerebrales se amplifican y son cotejadas con los algoritmos que se encuentran codificados en la memoria del chip.

Del chip ThinkGear se puede extraer la siguiente información:

El dato **poor_signal_quality** se trata de un byte sin signo que describe lo "pobre" que es la señal medida por el ThinkGear. El rango de valores es de 0 a 200; cualquier valor distinto de cero indica que hay ruido y si el valor es 200 quiere decir que no existe conexión entre el sensor y la piel.

Este valor se envía cada segundo, está activo por defecto y su frecuencia de muestreo es 1 Hz. La calidad de la señal puede verse modificada por varias causas:

- Los contactos del sensor o la masa no están en contacto con la frente o la oreja respectivamente.
- Existe un contacto pobre entre el sensor y la piel.
- Excesivo movimiento del usuario.
- Ruido electroestático ambiental.
- Ruido generado por otras medidas bioeléctricas.

El dato **eSense metters**, que cuenta con dos valores, attention y meditation, los cuales se recogen en una escala de 0 a 100 y se actualizan cada segundo. Cerca de cero tanto la atención como la meditación son bajas, y cerca de 80 a 100 son altas. El valor cero significa que el sensor no es capaz de calcular un valor

El valor de la señal raw o **RAW_Wave_Value**. está formado por dos bytes y representa una única muestra de la señal RAW. Se trata de un entero con signo, dentro del rango -32768 a 32767. Este dato es sobre el que se trabaja en este proyecto y es recogido por el casco a una velocidad de 512 muestras por segundo, es decir, 512Hz de frecuencia de muestreo

El **ASIC_EEG_POWER**, que representa el valor de 8 tipos de ondas cerebrales. Esta información va recogida en 8 grupos de 3 bytes, enteros con signo y en formato little-endian. Los 8 tipos de ondas cerebrales se recogen en el siguiente orden:

- Delta (0.5 – 2.75 Hz)
- Theta (3.5 – 6.75 Hz)
- Low-alpha (7.5 – 9.25 Hz)
- High-alpha (10 – 11.75 Hz)
- Low-beta (13 – 16.75 Hz)
- High-beta (18 – 29.75 Hz)
- Low-gamma (31 – 39.75 Hz)
- High-gamma (41 – 49.75 Hz)

Estos valores se recogen una vez cada segundo.

El dato **BLINK_STRENGTH,** que indica la intensidad del parpadeo de ojos del usuario. Es un byte cuyo rango va de 1 a 255. Se recoge cada vez que existe un parpadeo. El dato se actualiza cada segundo. Este parpadeo se encuentra a una frecuencia de unos 17 Hz, y en este proyecto es considerado como ruido.

Los componentes ThinkGear envían todos estos datos mediante un flujo de bytes en serie asíncrono. El medio de transporte puede ser la UART, puerto serie COM, USB, Bluetooth, ficheros o cualquier otro mecanismo capaz de transportar bytes.

6. ABB

ABB (Asea Brown Boveri) es una empresa creada en 1988 cuando se fusionaron las empresas Asea, fundada en 1883 y Brown Boveri en 1891.

Esta empresa se dedica a la ingeniería eléctrica y a la automatización y es líder mundial en robótica, área en la que se centra este proyecto. ABB cuenta en su catálogo con 6 robots de pintura, 27 robots articulados y un robot paralelo, que pueden mover desde los 3 kg hasta los 800 kg. Este proyecto se centra en el área de robótica de ABB, haciendo uso del brazo robot IRB120 y del software RobotStudio para el diseño y control de dicho brazo robot.

6.1. Brazo Robot IRB120

En este proyecto se utilizará el brazo robot IRB120 de ABB (figura 6.1). Este brazo es el más pequeño del catálogo de ABB, pesa 25 kg, puede soportar una carga de 3 kg y su área de trabajo alcanza los 580 mm. Cuenta con 6 grados de libertad, 3 de orientación y 3 de posición, permitiéndole alcanzar cualquier posición dentro de su área de trabajo. Estas especificaciones, junto a su bajo coste en comparación con otros robots articulados de ABB permiten hacer uso de este robot de manera didáctica.

Figura 6.1: Brazo robot IRB120

De este brazo robot existe el modelo IRB120T, que alcanza velocidades mayores en sus ejes, permitiendo mejorar los tiempos de ciclo en la producción en un 25 %.

En la figura 6.2 se puede observar el área de trabajo de este robot, mencionada anteriormente.

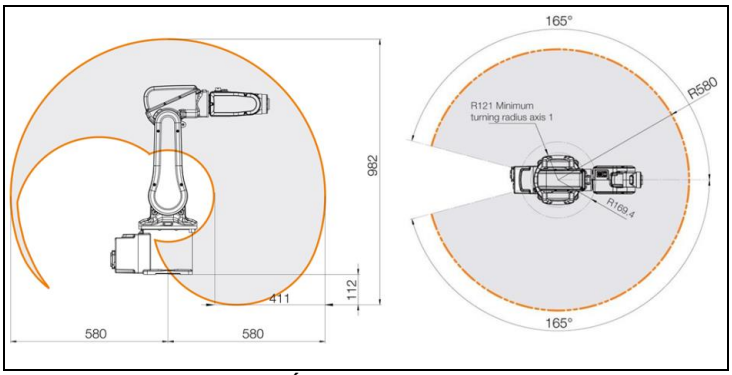

Figura 6.2: Área de trabajo del IRB120

6.2. RobotStudio

RobotStudio es el software creado por ABB que permite crear, programar y simular estaciones de trabajo de robots industriales ABB en 3D. Al ser un software creado por la propia empresa, permite simular cualquiera de los robots reales de ABB y nos permite exportar el programa creado a una estación real disminuyendo los riesgos y tiempos de arranque, aumentando así la productividad. En la figura 6.3 se puede observar la estación utilizada en la realización de este proyecto, con el robot IRB120 y la mano del robot InMoov acoplada a su extremo.

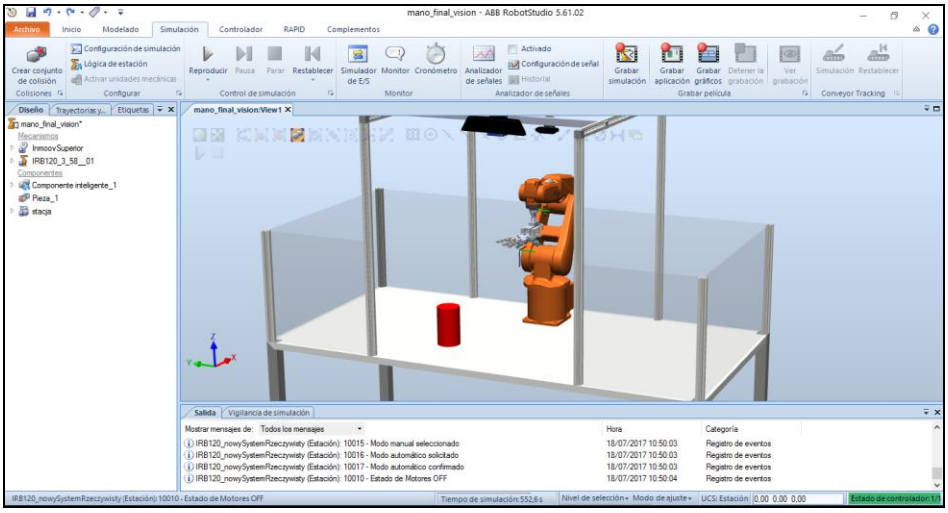

Figura 6.3: Estación de RobotStudio

También se ha colocado un cilindro sólido en las coordenadas XYZ 500, -250, 170 respectivamente. El robot moverá este cilindro cuando se active la señal de control desde esta posición al punto Y=250 conservando las otras dos coordenadas y pasando por el punto intermedio 300, 0, 400 para evitar colisiones con la mesa en el sistema real. Además, se ha diseñado en RobotStudio un componente inteligente que permite detectar el cilindro, agarrarlo, y soltarlo en la simulación.

El simulador RobotStudio funciona sobre RobotWare un software controlador flexible que contiene un conjunto de archivos necesarios para implementar todas las funciones, configuraciones, datos y programas necesarios para el control de los robots de ABB. [38]
Además, en la programación con RobotStudio se utiliza el lenguaje RAPID (Robotics Application Programming Interactive Dialogue) un lenguaje de alto nivel, muy similar al lenguaje C, creado por ABB para controlar sus robots. Los programas en RAPID constan de una rutina principal (main) donde se inicia la ejecución, un conjunto de subrutinas o funciones que permiten dividir el programa en módulos y los datos necesarios para el funcionamiento del programa. [40]
Una vez se ha desarrollado el código RAPID y se ha comprobado su correcto funcionamiento en simulación, este se carga en el controlador real del robot, en este caso el IRC5.

6.3. Controlador IRC5

Los controladores del robot industrial de ABB ofrecen un control de movimiento superior y permiten una rápida integración de hardware opcional. El IRC5 es la quinta generación de controladores de robot de ABB (figura 6.4). Su tecnología de control de movimiento, que combina TrueMove y QuickMove, es esencial para el rendimiento del robot en términos de precisión, velocidad, tiempos de ciclos, programación y sincronización con dispositivos externos. [46]

Figura 6.4: Controlador IRC5

Entre otros elementos se incluyen el FlexPendant (figura 6.5) que consiste en una unidad de programación con pantalla táctil y movimiento del robot con joystick, lenguaje RAPID flexible y potentes opciones de comunicación. Todas las operaciones y tareas de programación pueden realizarse con el FlexPendant portátil sin ser necesaria experiencia previa en programación. [47]

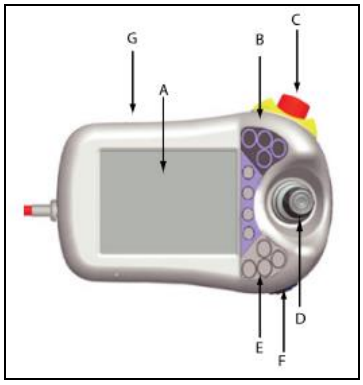

Figura 6.5: FlexPendant de ABB

El FlexPendant cuenta además con un pulsador de parada de emergencia (Botón C en la figura 6.5) y con un sistema de seguridad conocido como “pulsador de hombre muerto”. Este sistema de seguridad es un botón que se encuentra en la parte trasera del FlexPendant y que debe estar en todo momento pulsado por el operario, de no ser así el sistema robot se detendrá.

7. Robot Inmoov

El robot Inmoov (figura 7.1) es un robot humanoide diseñado por el escultor francés Gael Langevin en septiembre de 2011.

Figura 7.1: Robot Inmoov

Este robot está a disposición de todo el mundo ya que ha sido desarrollado con software de código abierto, puede ser impreso completamente en 3D con una impresora 3D de 12 cm^3 y puede ser controlado por microcontroladores Arduino, una tecnología también basada en código abierto, por lo que cualquier persona puede descargar sus piezas y construirlo, ascendiendo el coste de la impresión a unos 800 €. [4]

Inmoov es capaz de percibir los sonidos, ver, hablar y moverse independientemente. El robot es capaz de percibir su entorno a través de webcams y responder a órdenes facilitadas por su propietario. Éste incorpora diferentes sensores de presión, infrarrojos y de 3 dimensiones, además del sensor Kinect, que permite al robot ver y analizar el espacio tridimensional del entorno del robot. [5]

Al ser una tecnología de código muchos desarrolladores han modificado el código para incluir tecnología de visión artificial, reconocimiento de entornos, conexión a ordenadores, etc. Todas las piezas necesarias para construir este robot se pueden descargar de la propia página del robot InMoov. [25]

En esta página se pueden encontrar las siguientes piezas imprimibles para poder construir el robot Inmoov: 28 piezas de espalda, 23 piezas de bíceps, 14 piezas de pecho, 12 piezas de ojos, 9 piezas de rostro, 10 piezas de antebrazo, 32 piezas de manos, 44 piezas de abdomen (bajo, medio y alto), 14 piezas de cuello, 21 piezas de hombros,11 piezas de cráneo y orejas y 8 piezas de muñeca. En total, el número de piezas que forman este robot es de 226.

Inmoov cuenta además con 5 DOF (grados de libertad) en cada brazo, 16 DOF en cada una de sus dos manos y 6 DOF para su cabeza, y tiene 10 dedos motorizados independientes, todo ello le permite una gran movilidad e independencia. [6]

7.1. Mano robot Inmoov

Este trabajo está centrado exclusivamente en el control de la mano del robot Inmoov (figura 7.2), tratando de conseguir que se abra y se cierre a voluntad del usuario mediante la aplicación de una señal de control binaria generada a partir de la actividad cerebral del sujeto. Una vez hecho esto se intentará que, haciendo uso de la interfaz gráfica de Matlab el usuario sea capaz de introducir una pelota en un recipiente vacío utilizando la misma señal de control.

Figura 7.2: Mano del robot Inmoov

La mano del robot Inmoov cuenta con 16 grados de libertad, lo que le otorga, como se dijo en el apartado anterior, una gran movilidad. Cada uno de sus dedos puede ser controlado independientemente, asemejándose al comportamiento de la mano humana, permitiendo una gran variedad de movimientos y gestos. Además, esta mano puede ser controlada fácilmente con distintos tipos de softwares, como puede ser Arduino. Todas estas cualidades le dan la capacidad de, en un futuro, poder comportarse como una prótesis robótica que se pueda controlar mediante una BCI con total libertad.

En este proyecto únicamente se controlan los movimientos de apertura y cierre, pero en un futuro no muy lejano, con los avances conseguidos en el campo de la neurociencia, podrá implementarse un sistema que permita el control total e individual de cada una de las partes de una prótesis semejante a la mano aquí mostrada, como ya se ha visto con las prótesis que está fabricando DARPA en la actualidad.

8. Actividad inducida en la banda gamma

El objetivo de este trabajo es poder controlar la mano del robot Inmoov-SR con el casco NeuroSky MindSet. Para poder realizar esto lo primero que se necesita es ser capaces de detectar actividad motriz en nuestro cerebro con el NeuroSky y, para ello, se ha evaluado la actividad en la banda gamma baja del cerebro. El casco NeuroSky ha permitido realizar una electroencefalografía del cerebro de 11 sujetos sanos, tanto en reposo como realizando tareas motoras.

8.1. Detección de GBA inducida

Para recoger los datos se ha realizado en cada sujeto un experimento dividido en dos partes, el experimento basal y el experimento motriz. En ambos casos el sujeto se encuentra sentado en una silla en frente de un ordenador con la mano derecha colocada encima de la mesa con la palma hacia abajo. El casco NeuroSky Mindset es colocado en el sujeto como se muestra en la figura 8.1.

Figura 8.1: Posición de los sensores del casco Neurosky

El diagrama de bloques correspondiente a esta sección es el mostrado en la figura 8.2.

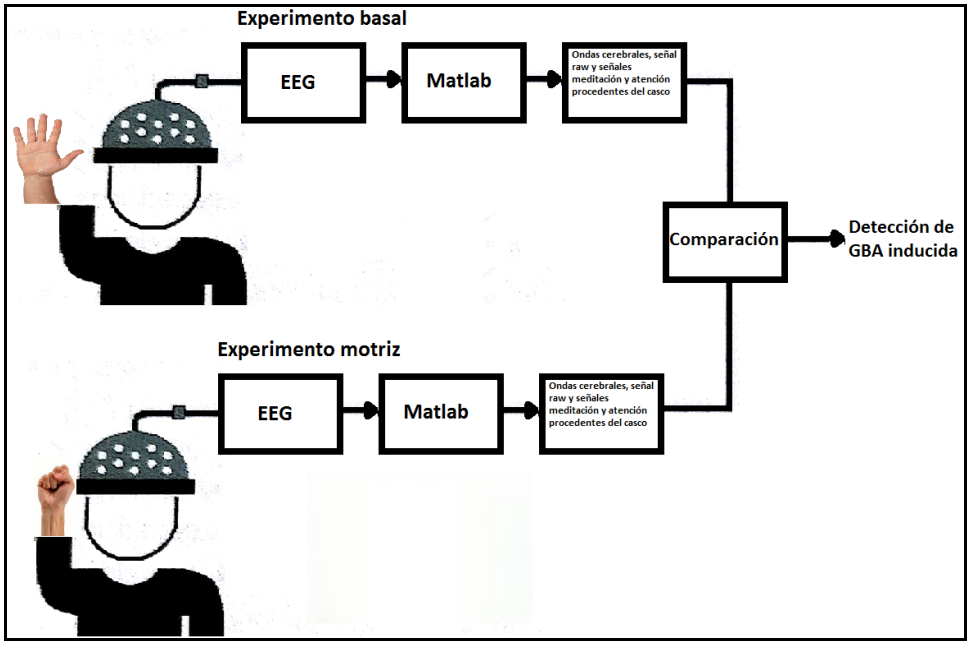

Figura 8 2: Diagrama de bloques para detectar GBA inducida

8.1.1. Experimento basal

Durante el experimento basal, el sujeto debe permanecer inmóvil, de la manera más relajada posible, observando un punto blanco (figura 8.3) en la pantalla del ordenador durante 200 segundos.

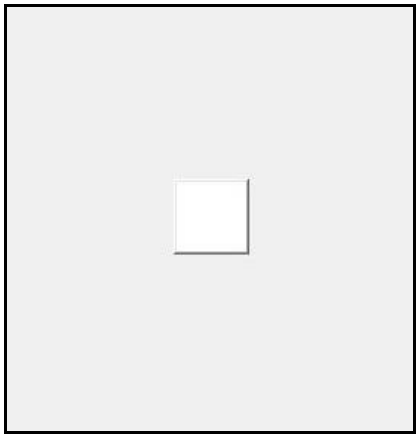

Figura 8.3: Punto blanco del experimento basal

Los datos de este experimento son recogidos con el código "Experimento1.m" mostrado en la sección de planos de este documento. Los datos proporcionados por el software NeuroSky son, como se comentó anteriormente, todas las ondas cerebrales más las señales de atención y meditación. Además, NeuroSky también proporciona la señal raw, señal que será estudiada con el objetivo de detectar la GBA inducida.

En total las señales recogidas en este experimento son 8 (figura 8.4), y son guardadas en Matlab como un array de una longitud aproximada de 102400 muestras ya que la frecuencia de muestreo del casco según la empresa es de 512Hz.

Figura 8.4: Señales obtenidas en un experimento basal

Como se puede observar en la figura 8.4, la frecuencia de muestreo puede ser mayor ya que el dato de 512 Hz puede variar. En cualquier caso, la única señal que el casco recoge realmente a 512Hz es la señal raw, siendo la frecuencia de muestreo de las demás de 1 Hz (figura 8.5) y, por tanto, es la señal raw la que se estudia en el desarrollo de este proyecto.

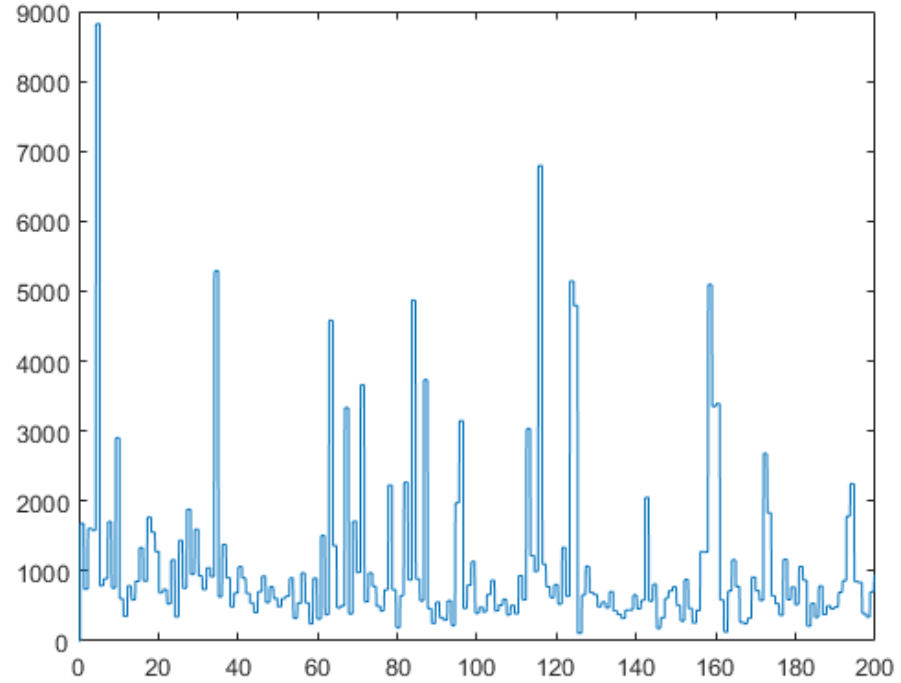

Figura 8.5: Ejemplo de la señal gamma muestreada a 1Hz por el casco NeuroSky durante el experimento basal

Una vez finalizados los 200 segundos del experimento, se obtienen señales raw como la mostrada en la figura 8.6

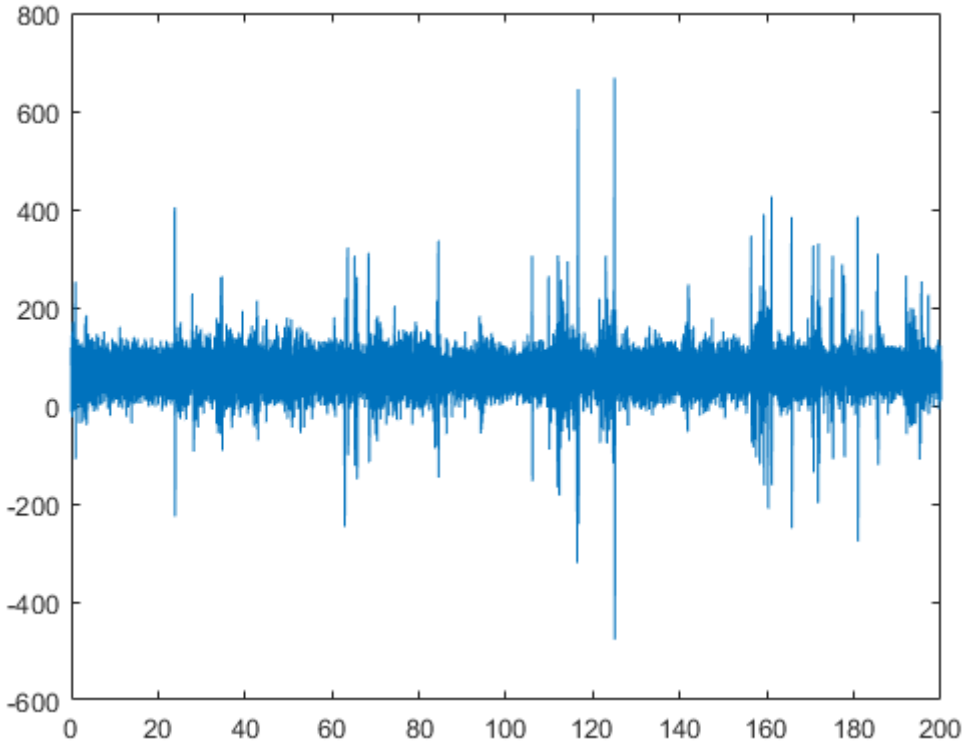

Figura 8.6: Señal Raw obtenida en el experimento basal de un sujeto.

A continuación, se va a explicar de manera más detallada en que consiste la segunda parte del experimento.

8.1.2. Experimento Motriz

En este experimento el sujeto debe permanecer igual a como se encontraba durante la realización del experimento basal, además, este experimento debe realizarse a continuación del anterior puesto que si se realizasen en momentos distintos del día no se podría determinar que el cambio producido en las ondas cerebrales fuera necesariamente debido a la realización de movimiento. Por ejemplo, el haber ingerido un café, o el estado de ánimo cambiante del sujeto puede influir en los resultados del experimento. En el caso de este experimento el código ejecutable es el nombrado "Experimento2.m".

Las señales recogidas son las mismas que las del experimento anterior y al igual que en el caso basal la señal estudiada es la señal raw.

En este experimento el sujeto debe cerrar la mano derecha, que permanecía en reposo sobre la mesa, cada dos segundos, coincidiendo con un parpadeo en rojo del punto del experimento anterior (figura 8.7)

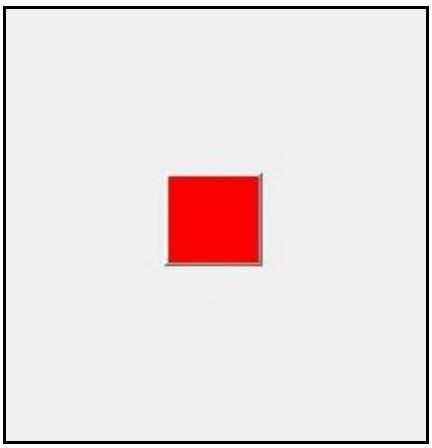

Figura 8.7: Parpadeo en rojo durante la realización del experimento motriz

En la figura 8.8 se puede observar en un diagrama temporal como ha de realizarse este experimento, permaneciendo la mano abierta en todo momento salvo en los parpadeos.

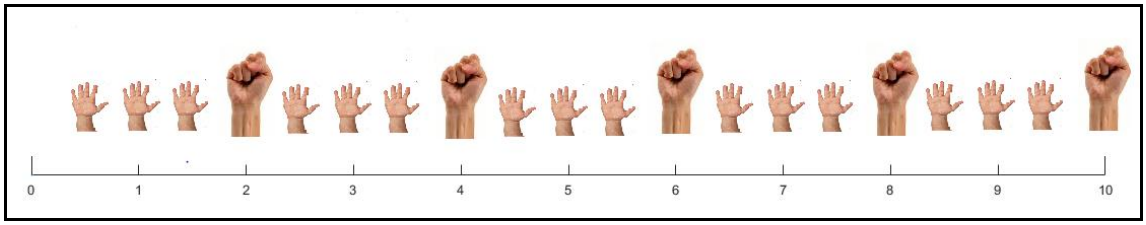

Figura 8.8: Diagrama temporal del experimento 2

Las señales raw obtenidas en este experimento son similares a la mostrada en la figura 8.9.

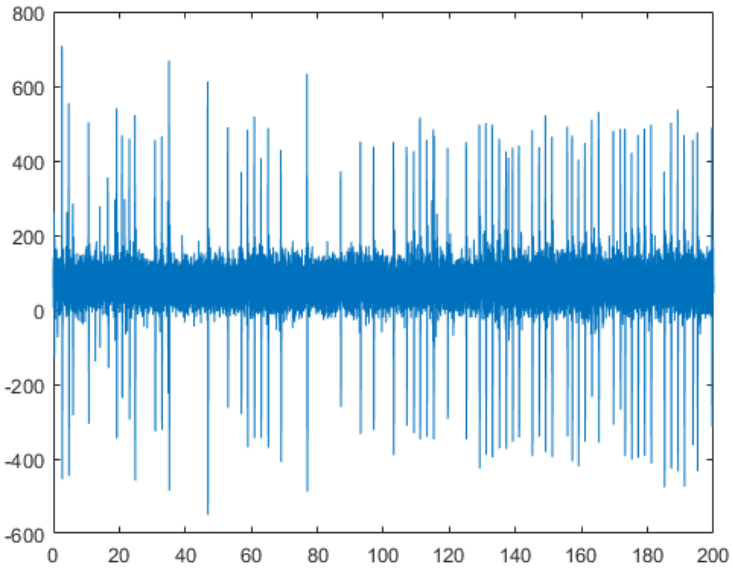

Figura 8.9: Señal raw obtenida en el experimento motriz de un sujeto.

A simple vista se puede observar que en el caso motriz la señal tiene unos picos de mayor amplitud que en el caso basal.

Tras recoger las señales de los 11 sujetos, estas son procesadas para poder justificar la detección de GBA inducida con el casco NeuroSky. La manera de procesar y estudiar estas señales se explica de manera detallada a continuación.

8.2. Procesamiento de las señales obtenidas

En este apartado se explicará el tipo de procesamiento que reciben las señales raw obtenidas y que son consideradas como válidas y el posterior análisis realizado sobre los resultados obtenidos.

8.2.1. Procesado

De los 11 sujetos sobre los que se realiza el experimento se ha tenido que descartar a 3, debido a que durante la realización de sus experimentos el sensor del casco había perdido el contacto con la piel del sujeto y las señales recogidas no eran válidas (figuras 8.10 y 8.11).

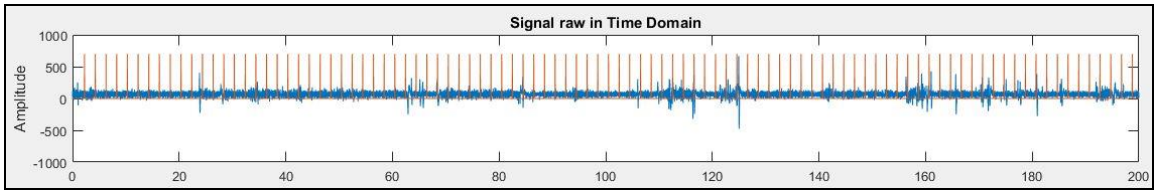

Figura 8.10: Señal válida

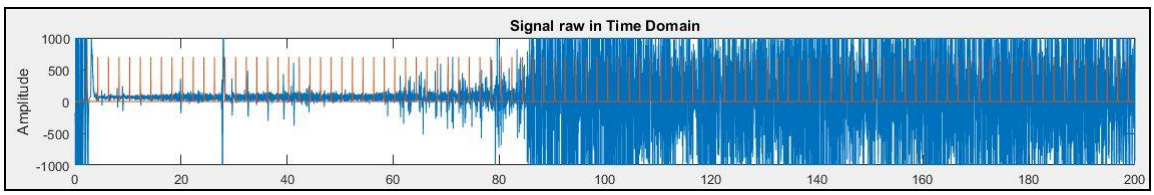

Figura 8.11: Señal no válida

Todo el procesado que se va a explicar a continuación se puede observar de manera más intuitiva en el diagrama de bloques de la figura 8.12

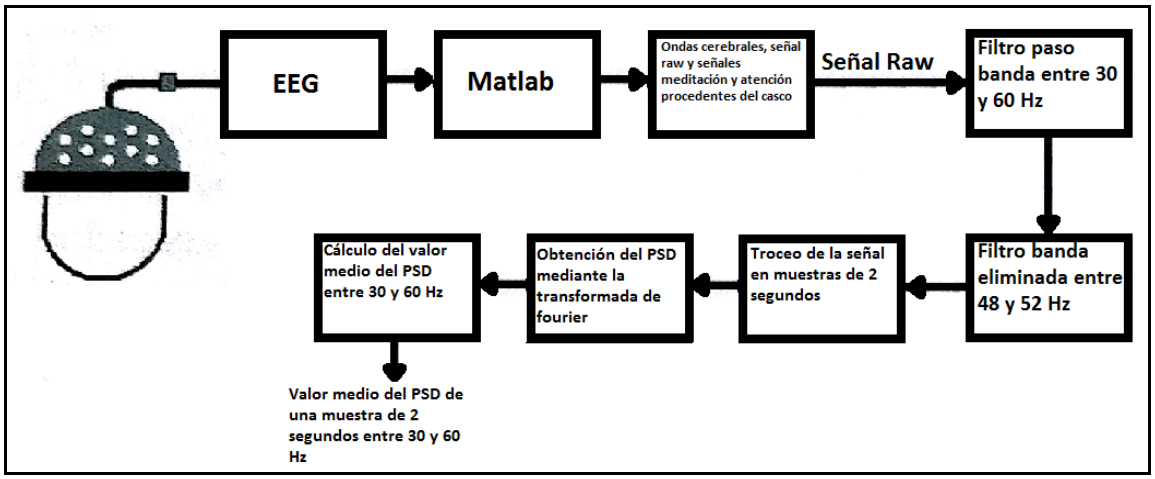

Figura 8.12: Diagrama de bloques del procesamiento de las señales obtenidas

Lo primero que se hace con las señales raw obtenidas es aplicarles un filtro paso-banda a la entre 30 y 60 hercios, lo que sería la banda gamma baja, en la cual según el estudio "Oscillatory gamma activity in humans and its role in object representation" [31] se puede observar una respuesta ante un estímulo sensorial o ante tareas motrices, como puede ser el movimiento de la mano. Además, se elimina así el parpadeo que se encuentra a una frecuencia de 17Hz.

También se aplica un filtro banda-eliminada entre 48 y 52 Hz para quitar las posibles interferencias a 50Hz de la red. Para realizar estos filtros se ha utilizado la toolbox de Matlab "FieldTrip".

Una vez hecho esto, esa señal de 200 segundos es dividida en 100 muestras de 2 segundos para separar de esta manera cada una de las veces que el sujeto ha abierto y cerrado la mano. Este procedimiento es llevado a cabo ejecutando el código "Trocear.m" que se puede encontrar en los planos de este TFG. En las figuras 8.10 y 8.11 se pueden observar en rojo las divisiones que se hacen sobre la señal raw. Además, todas las muestras de 2 segundos se normalizan a 1000 datos por muestra, es decir, todas las muestras se normalizan a una frecuencia de muestreo de 500Hz. Después, se utiliza un interfaz gráfico en Matlab que nos permite visualizar cada una de las muestras por separado y decidir si la muestra es aceptada o rechazada (figura 8.13).

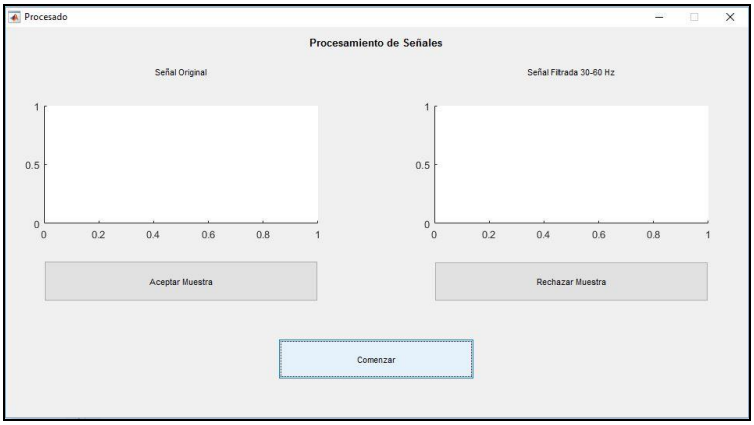

Figura 8.13: Interfaz de selección de muestras

Para cada sujeto, se han cogido todas las muestras de dos segundos (1000 datos), tanto del caso basal como del caso motriz y se han recogido en 2 matrices distintas donde cada fila es una muestra de 1000 datos. Estas muestras de dos segundos se han añadido a un vector fila para cada caso (vector basal y vector motriz) de una en una y cada vez que se ha añadido una muestra nueva, del vector resultante se ha obtenido el espectro en potencia, se ha calculado la envolvente del espectro, y se ha obtenido su valor máximo para N muestras, pudiendo hacer así una gráfica de la evolución del pico máximo del espectro en potencia de una señal desde 1 muestra hasta el número de muestras total de nuestra señal después de haber seleccionado las válidas y rechazado las malas.

Como resultado de este proceso se han obtenido las gráficas mostradas en la figura 8.14 en las que se puede observar que en el caso motriz el PSD de la banda gamma aumenta considerablemente según aumentamos el tiempo del experimento.

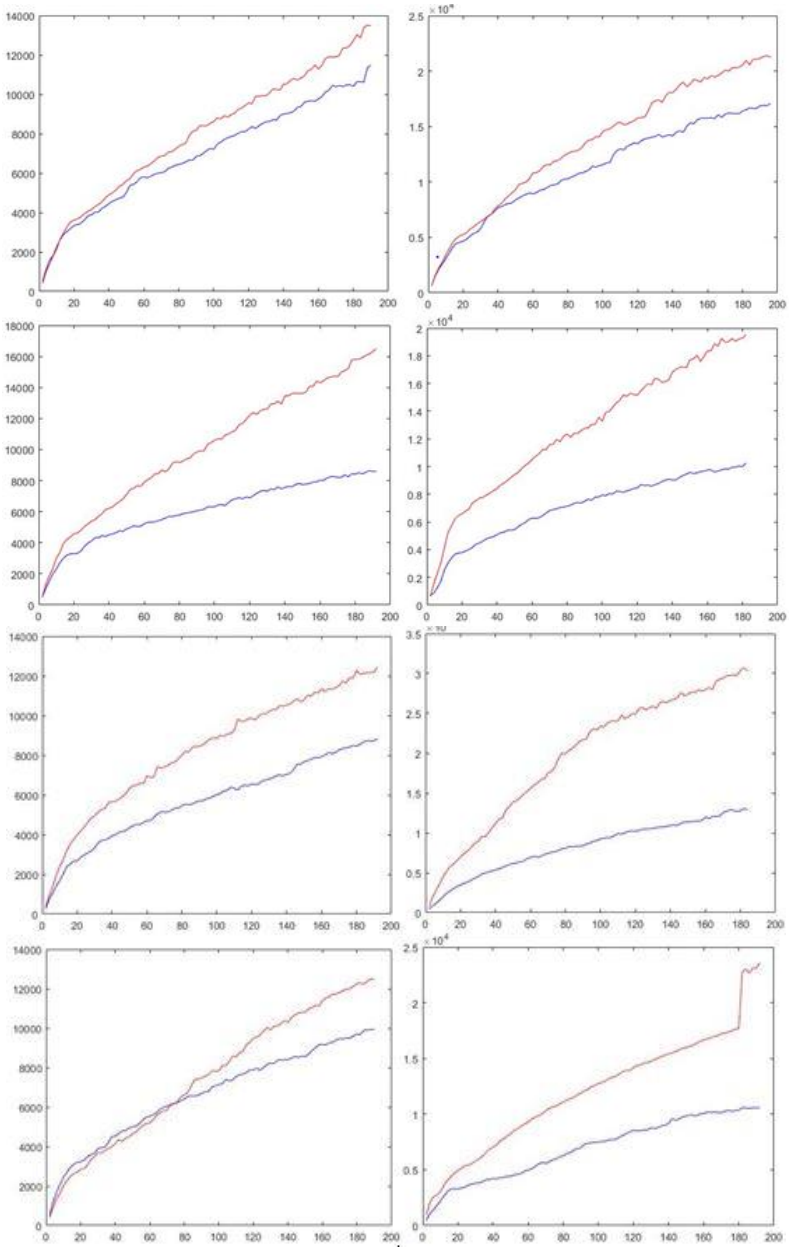

Figura 8.14: Evolución del PSD para los 8 sujetos válidos, a lo largo de los 200 segundos (en rojo el caso activo y en azul el caso basal)

En la figura 8.14 se puede observar como, a medida que aumenta el tiempo que una persona está abriendo y cerrando la mano (experimento motriz), el PSD aumenta considerablemente con respecto al caso basal, permitiendo diferenciar de manera clara cuando una persona ha realizado movimiento a partir de los 20 segundos, o 10 aperturas y cierres de mano. En la gráfica superior derecha esta diferencia no llega hasta los 40 segundos, debido a que el tiempo requerido para activar la banda gamma depende de la persona y de la práctica que tenga para hacerlo.

Tras pasar estas pruebas de detección de banda gamma, se continúa estudiando el PSD de las muestras. A continuación, se explica de manera más detallada el procesamiento aplicado a cada muestra individual de 2 segundos.

En la figura 8.15 tenemos una muestra de 2 segundos de una persona que ha cerrado la mano y la ha vuelto a abrir. Cada persona ha sido monitorizada durante 200 segundos en estado basal y durante otros 200 segundos realizando movimiento cada 2 segundos.

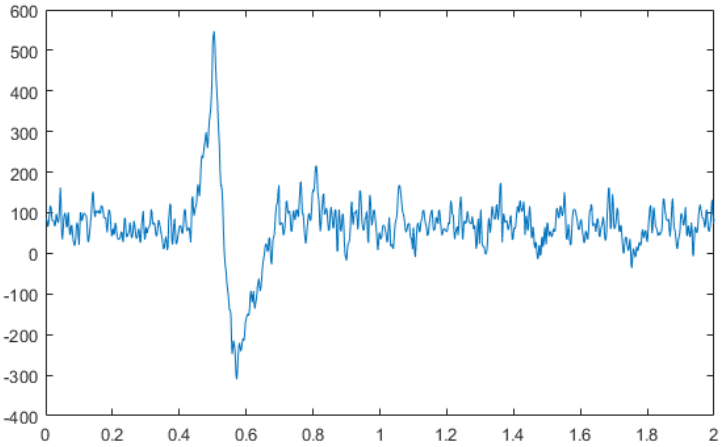

Figura 8.15: Muestra de 2 segundos de una persona realizando movimiento

Sobre esta muestra, es aplicado un filtro entre 30 y 60 Hz, lo que sería la banda gamma baja, en la que se quiere ver la actividad inducida, dando como resultado la señal de la figura 8.16.

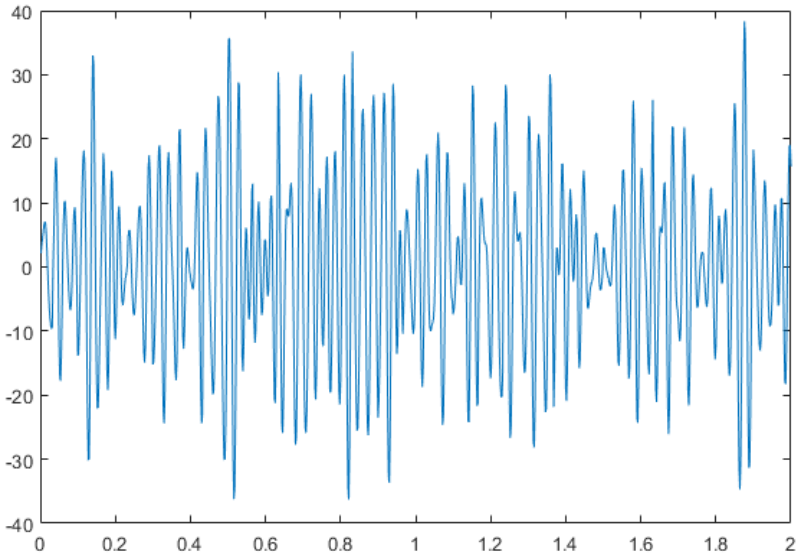

Figura 8.16: Muestra de dos segundos de una persona que ha realizado movimiento, filtrada entre 30 y 60Hz

De esta manera, se puede observar el siguiente espectro en potencia de la muestra de 2 segundos, una vez aplicados el filtro paso banda entre 30 y 60 Hz y el filtro banda eliminada entre 48 y 52 Hz (figura 8.17)

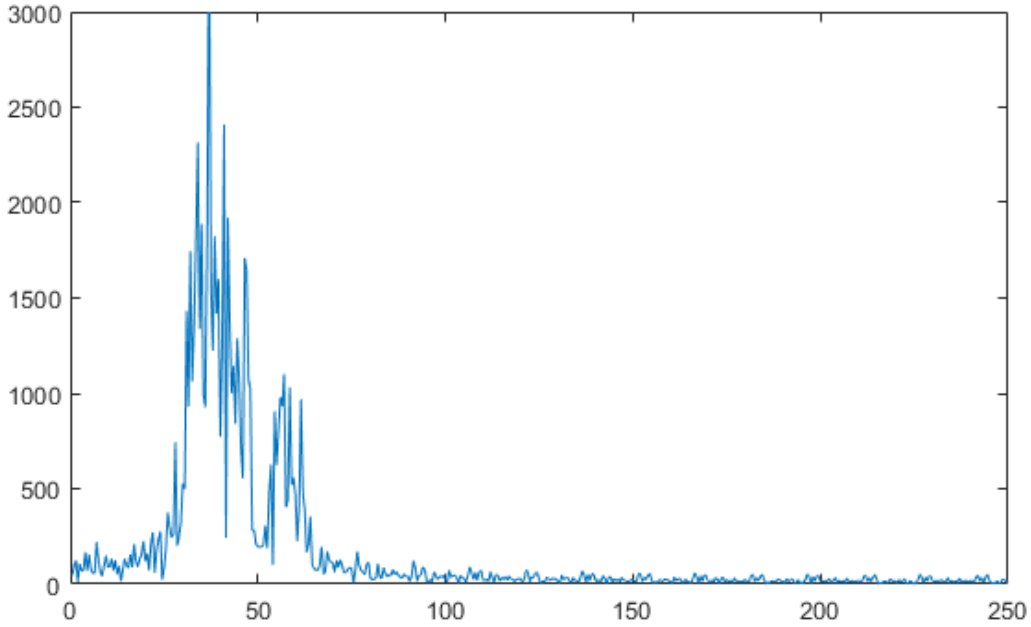

Figura 8.17: PSD de la muestra de 2 segundos de la figura 8.16

De esta señal, se obtiene el valor medio de la banda gamma, entre 30 y 60 Hz (figura 8.19).

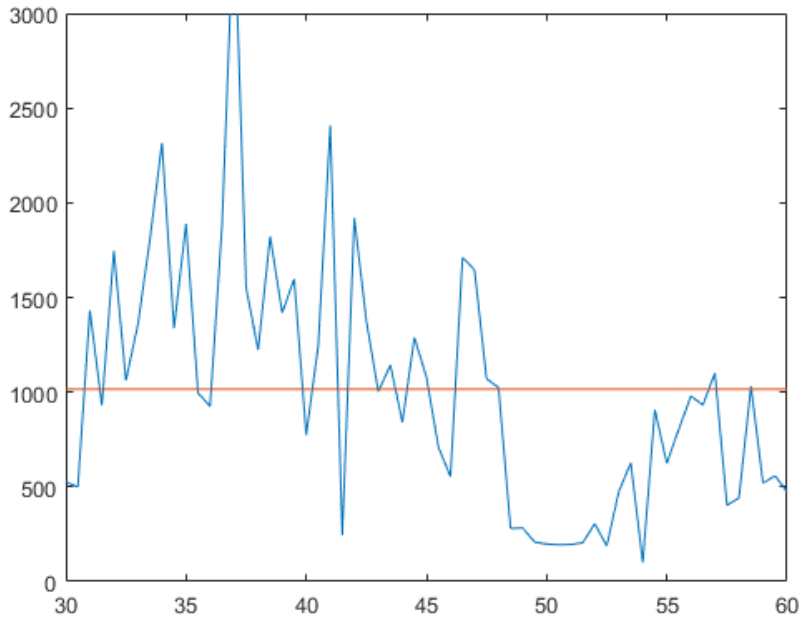

Figura 8.18: PSD entre 30 y 60 Hz de la muestra de 2 segundos de la figura 8.16, y su valor medio

Del espectro en potencia se obtiene su envolvente utilizando un filtro paso bajo a 10 Hz, y obtenemos el valor medio de la banda gamma de la envolvente (figura 8.19)

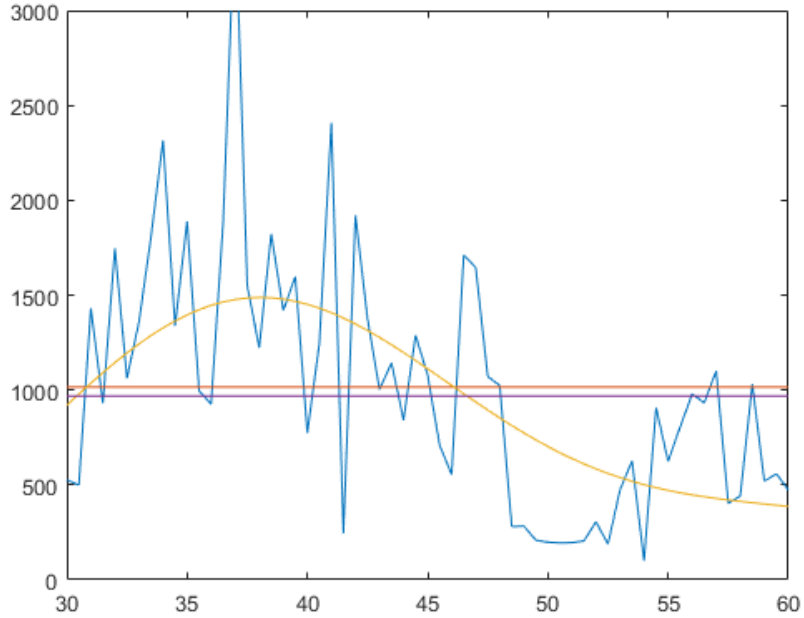

Figura 8.19: Envolvente del PSD de la figura 8.18 y valor medio de dicha envolvente en superposición con los valores originales.

En este caso se muestra en la figura 8.18 el espectro en potencia y su valor medio entre 30 y 60 Hz, de valor 1015,8 y en la figura 8.19 se muestra en adición a la figura 8.18, la envolvente de dicho espectro en potencia y el valor medio de la envolvente, de 968.2970. Quitando la señal original y su valor medio se obtiene lo mostrado en la figura 8.21.

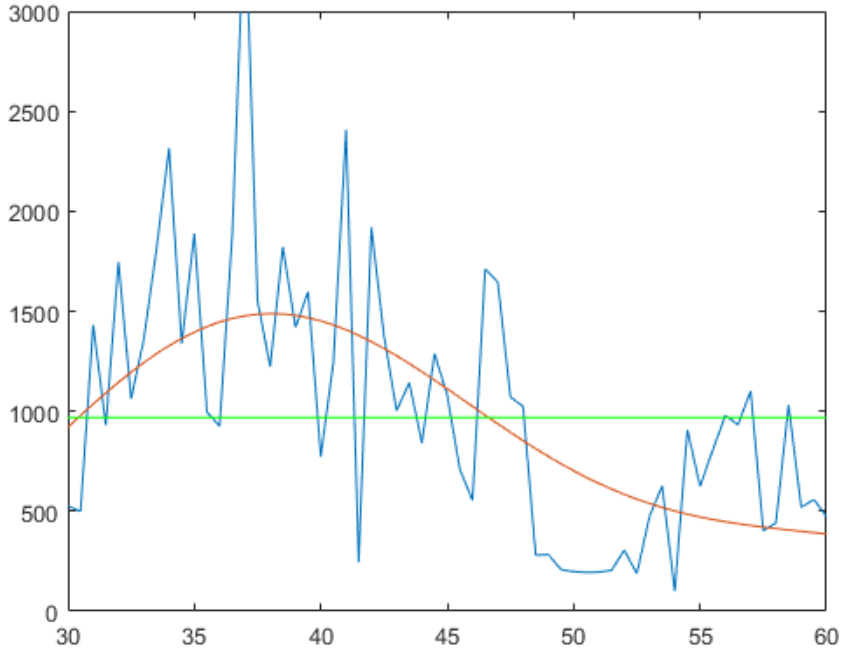

Figura 8.20: Envolvente del PSD de la figura 8.18 y valor medio de dicha envolvente en superposición con el PSD original

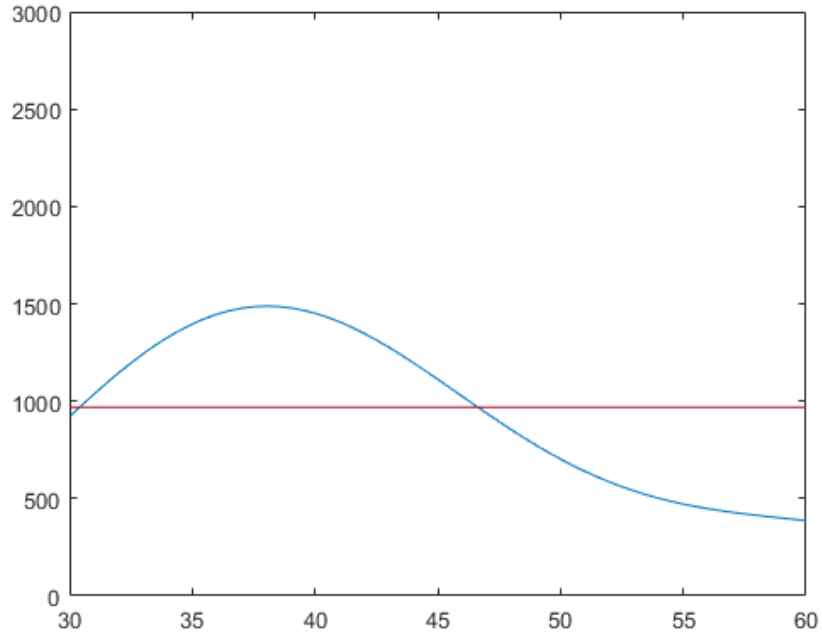

Figura 8.21: Envolvente del PSD de la figura 8.18 y valor medio de la envolvente entre 30 y 60 Hz

Con cada uno de los valores medios del espectro en potencia para cada una de las 100 muestras de un sujeto, se puede calcular una media global para cada sujeto, que podemos superponer en la gráfica anterior, quedando de la siguiente manera. (figura 8.22)

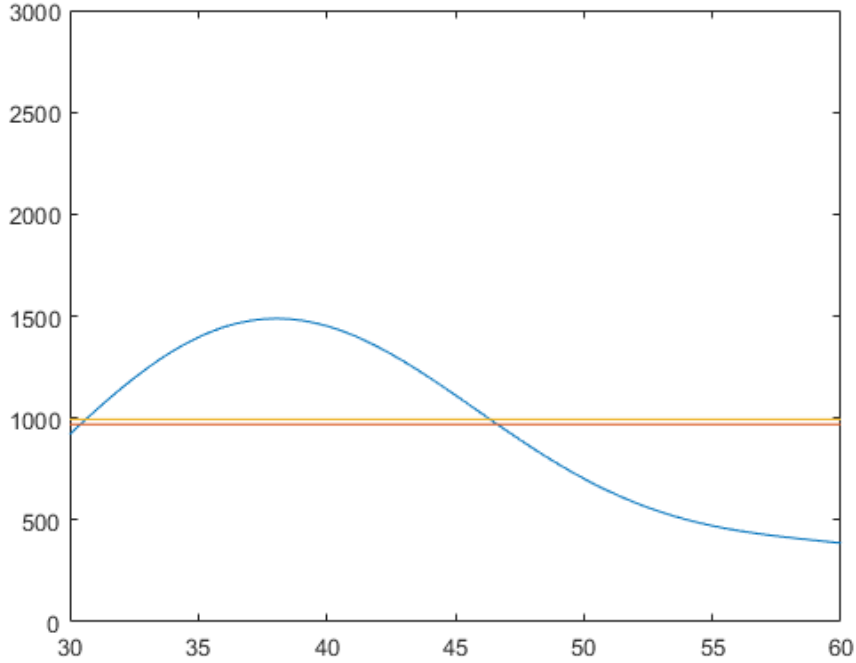

Figura 8.22: Valor medio de todos los valores medios del PSD de cada una de las 100 muestras en movimiento de un sujeto, en superposición con el valor medio de la gráfica 8.20

En la figura 8.22, las líneas que muestran el valor medio del PSD (espectro en potencia) entre 30 y 60 Hz de una muestra, de valor 968.2970 y el valor de la media global del sujeto con un valor igual a 993.7229, están prácticamente solapadas.

De la misma manera, se han obtenido estos valores para una muestra del mismo sujeto en estado basal, como se muestra en las figuras 8.23 y 8.24.

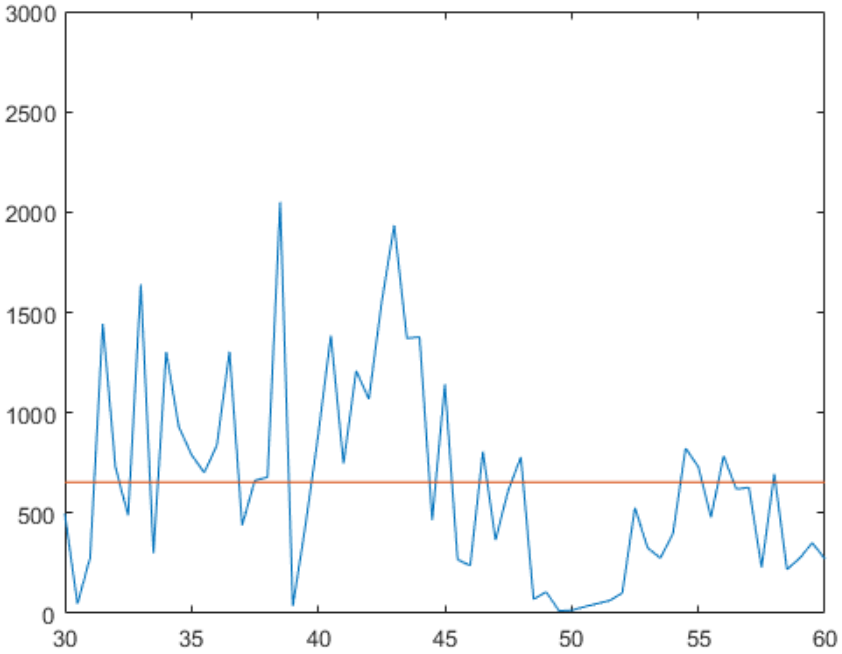

Figura 8.23: PSD entre 30 y 60 Hz de una muestra de 2 segundos en estado basal, y su valor medio

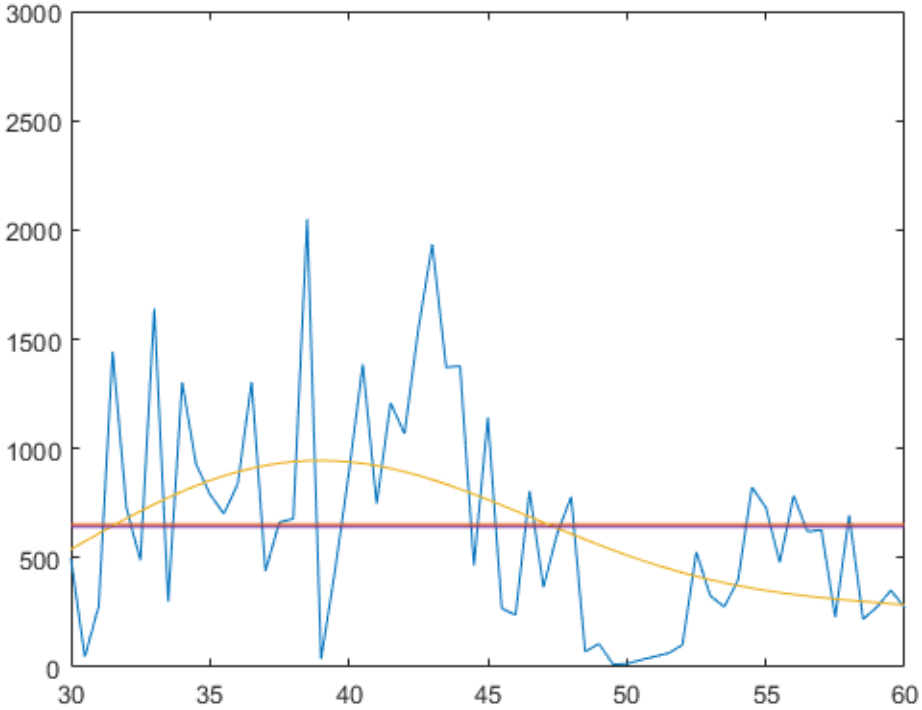

Figura 8.24: Envolvente del PSD de la figura 8.23 y valor medio de dicha envolvente en superposición con los valores originales.

En este caso, el valor medio del PSD original es 653.7069 y el valor medio de su envolvente es 640.7124.

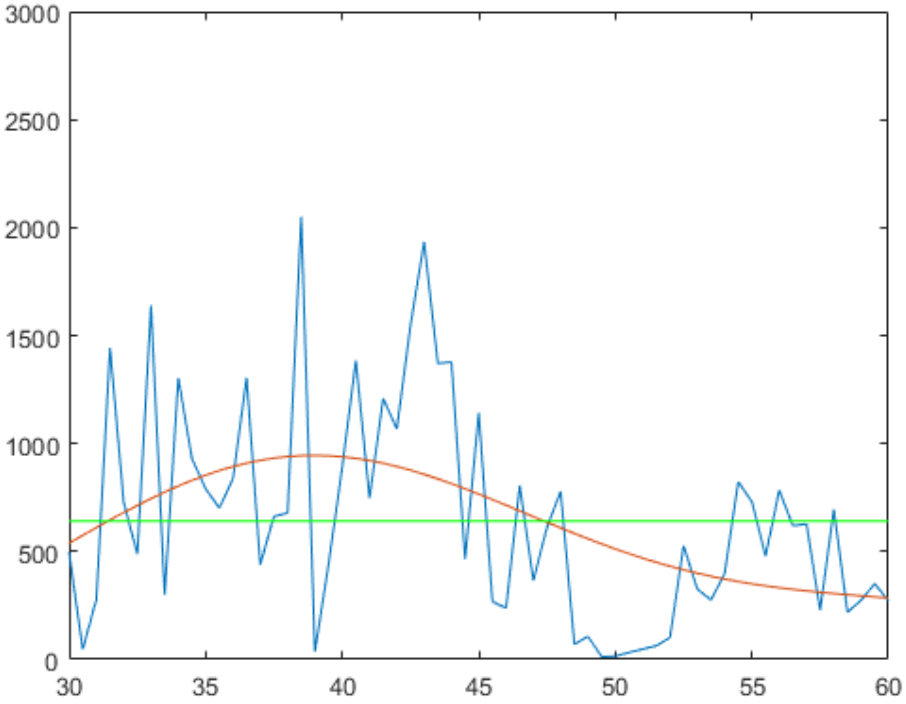

Figura 8.25: Envolvente del PSD de la figura 8.23 y valor medio de dicha envolvente en superposición con el PSD original.

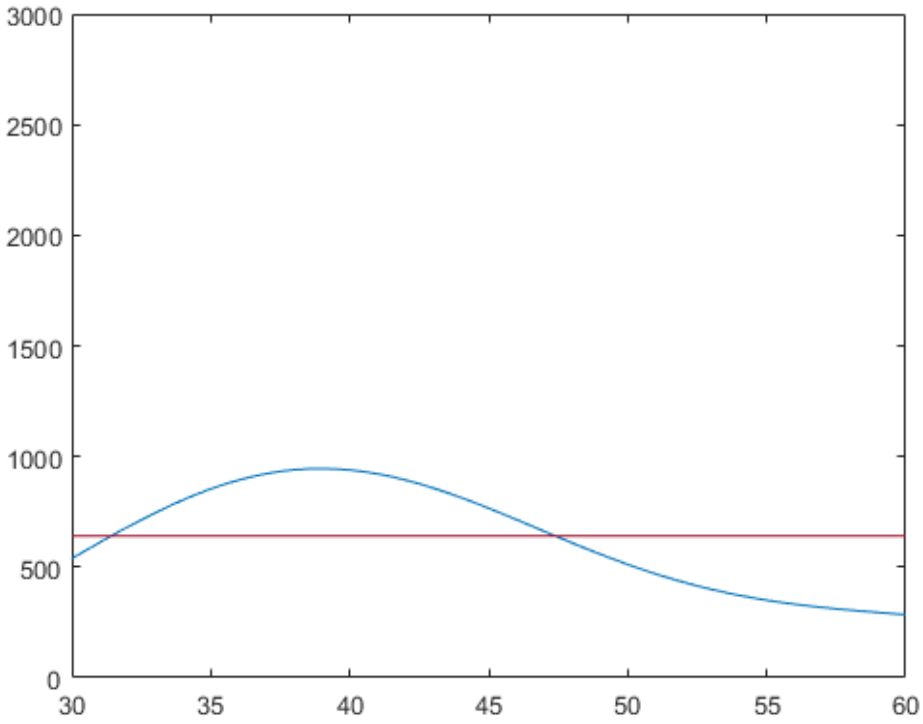

Figura 8.26: Envolvente del PSD de la figura 8.23 y valor medio de la envolvente entre 30 y 60 Hz.

Como se puede observar en las figuras 8.25 y 8.26, en el caso basal el pico máximo del espectro en frecuencia es considerablemente inferior al caso motriz, además en este caso el valor medio es igual a 640.7124.

Tras esto, se ha realizado sobre estas muestras un análisis estadístico

8.2.2. Análisis estadístico.

Las muestras de 2 segundos, ya procesadas como se ha explicado en el apartado anterior, han sido guardadas en 8 tablas (1 por sujeto) individualmente.

Estas tablas son como la siguiente:

Número de muestra	Valor medio del PSD basal	Valor medio del PSD motriz
1	552	1030
2	767	1070
3	690	1130
4	689	1090
5	962	1340
6	725	1360
7	666	1010
8	694	947
9	637	1150
10	609	936
11	655	1010
12	561	1090
13	602	1220
14	607	1010
15	637	1070
16	624	1180
17	634	1030
18	593	1090
19	664	972
20	628	1010
21	619	1050
22	581	920
23	626	859
24	671	942
25	587	1050
26	702	1140
27	626	1190
28	568	1120
29	713	1120
30	666	1120
31	675	1060
32	669	990
33	622	1070
34	649	987
35	674	1270
36	686	1010
37	697	1080
38	697	1050
39	623	1060
40	574	1230
41	656	906

42	700	1040
43	626	1030
44	553	1060
45	694	1270
46	640	1250
47	653	1120
48	661	1090
49	636	959
50	641	1270
51	660	1010
52	759	1280
53	594	1340
54	606	1010
55	603	1250
56	642	1170
57	735	1100
58	680	1050
59	553	1320
60	620	1190
61	647	1130
62	591	1130
63	636	1170
64	619	1180
65	606	1130
66	601	1110
67	660	1130
68	659	1030
69	673	1050
70	621	1130
71	572	1290
72	577	1030
73	605	1070
74	589	1130
75	567	1090
76	520	1340
77	695	1360
78	637	1010
79	509	947
80	669	1150
81	661	936
82	516	1010
83	694	1090
84	613	1220
85	606	1010
86	519	1070
87	620	1180
88	650	1030
89	547	1090
90	607	972
91	581	1010

Tabla 8.1 Valor medio del PSD de todas las muestras de un sujeto

La tabla 8.1 muestra el valor medio de la banda gamma del PSD de cada muestra individual para el sujeto 8. El número de muestras es menor de 100 debido a que aquí solo aparecen las muestras válidas.

		Valores medios		
Sujeto	Lateralidad	Media basal	Media motriz	p
1	Diestro	547	994	7,46E-51
2	Diestro	752	1620	3,20E-31
3	Diestro	102	1190	5,78E-14
4	Diestro	692	801	4,99E-20
5	Diestro	560	727	1,44E-25
6	Diestro	597	760	1,57E-13
7	Diestro	623	774	7,01E-08
8	Diestro	631	1100	1,28E-51

Tabla 8.2 Medias basal, motriz y valor p de la función t de student de cada uno de los 8 sujetos.

En la tabla 8.2 se muestra para cada sujeto, el valor medio de la población total de muestras tanto para el caso basal como para el caso motriz, es decir, el valor medio de las columnas de la tabla 8.1 para cada uno de los sujetos. Además, el valor p obtenido realizando una prueba t de student, tomando las dos columnas de la tabla 8.1 como las dos poblaciones, es significativamente inferior a 0.0001 y, por tanto, según la prueba t de student se puede determinar que las dos poblaciones de muestras presentan diferencias significativas.

También se han obtenido diagramas de barras como el mostrado en la figura 8.27, donde se muestra el valor de la media, en notación científica, para el experimento basal y para el experimento motriz del sujeto 8.

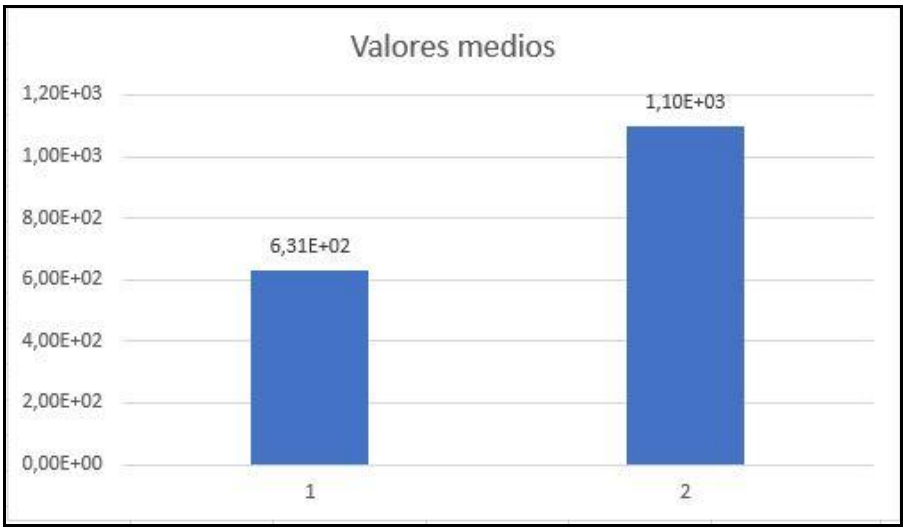

Figura 8.27: Diagrama de barras que muestra el valor medio de las dos poblaciones de muestras del sujeto 8 (1 la media del experimento basal y 2 la media del experimento motriz)

En la figura 8.27 también se ve claramente la diferencia entre poblaciones

En otra gráfica (figura 8.28), se ha pintado el diagrama de caja y bigotes para el sujeto 1 en este caso, donde también se puede observar por otro método que las poblaciones son distintas.

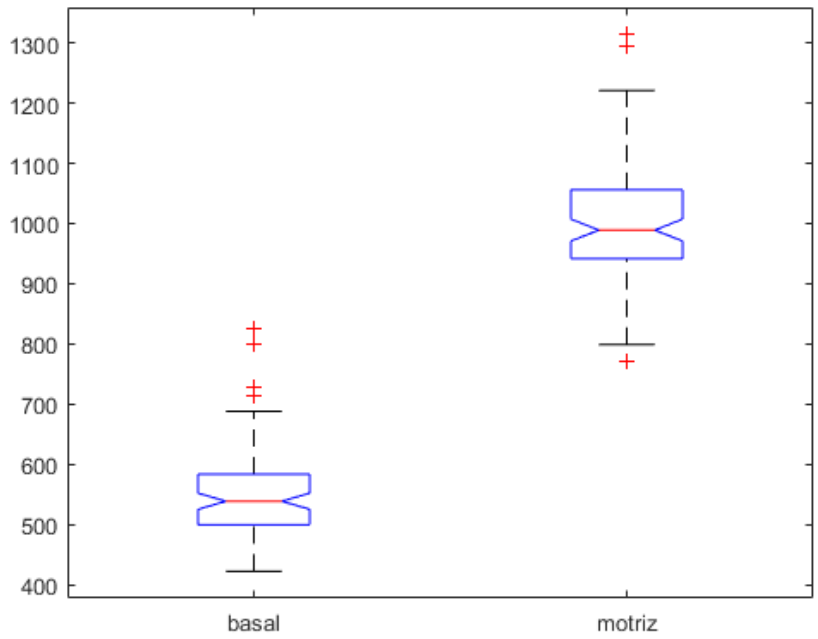

Figura 8.28: Diagrama de caja y bigotes de las dos poblaciones de muestras (basales y motrices).

Por último, se han evaluado las curvas ROC (Característica Operativa del Receptor) de los 8 sujetos, mostradas en la figura 8.29.

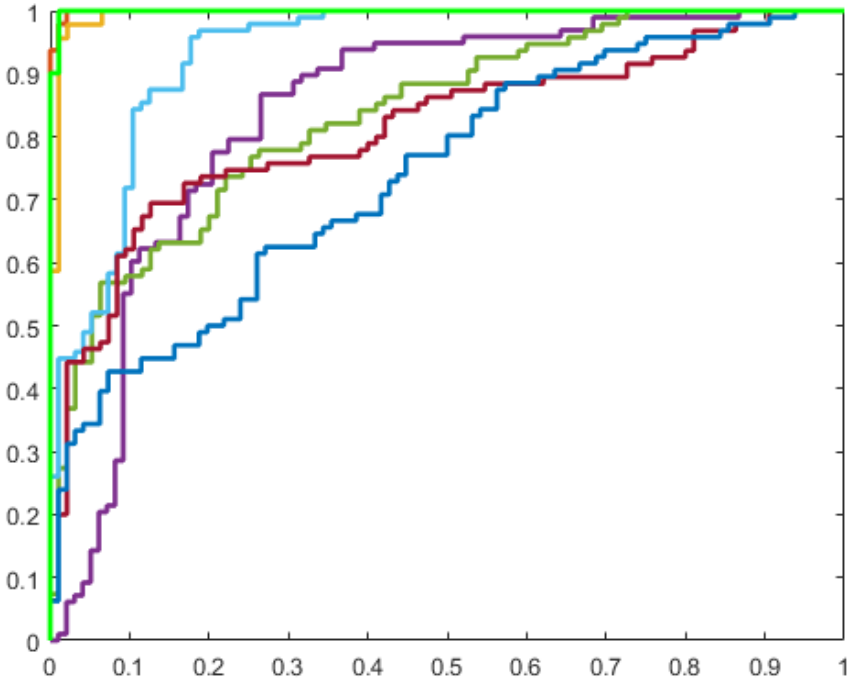

Figura 8.29: Curvas ROC de los 8 sujetos aceptados.

Las curvas ROC se utilizan como un método de clasificación binario, que permite conocer si dos poblaciones distintas son significativamente diferentes.

De estas curvas ROC podemos sacar los 8 valores de AUC (Área bajo la curva) para cada sujeto, quedando la siguiente tabla:

Sujeto	Lateralidad	AUC
1	Diestro	0.9991
2	Diestro	0.9941
3	Diestro	0.8390
4	Diestro	0.8349
5	Diestro	0.9353
6	Diestro	0.8121
7	Diestro	0.7399
8	Diestro	0.9991

Tabla 8.3 Valor del área bajo la curva de cada uno de los 8 sujetos

Cuanto más cercano este el valor de AUC (Área bajo la curva) a 1 más fácil será distinguir las dos poblaciones en un sujeto (basal y motriz) y cuanto más cerca este de 0.5, más parecidas serán ambas poblaciones. El valor medio de AUC de los 8 sujetos es 0.8942.

Por otra parte, se ha obtenido la siguiente curva ROC media de los 8 sujetos (figura 8.30):

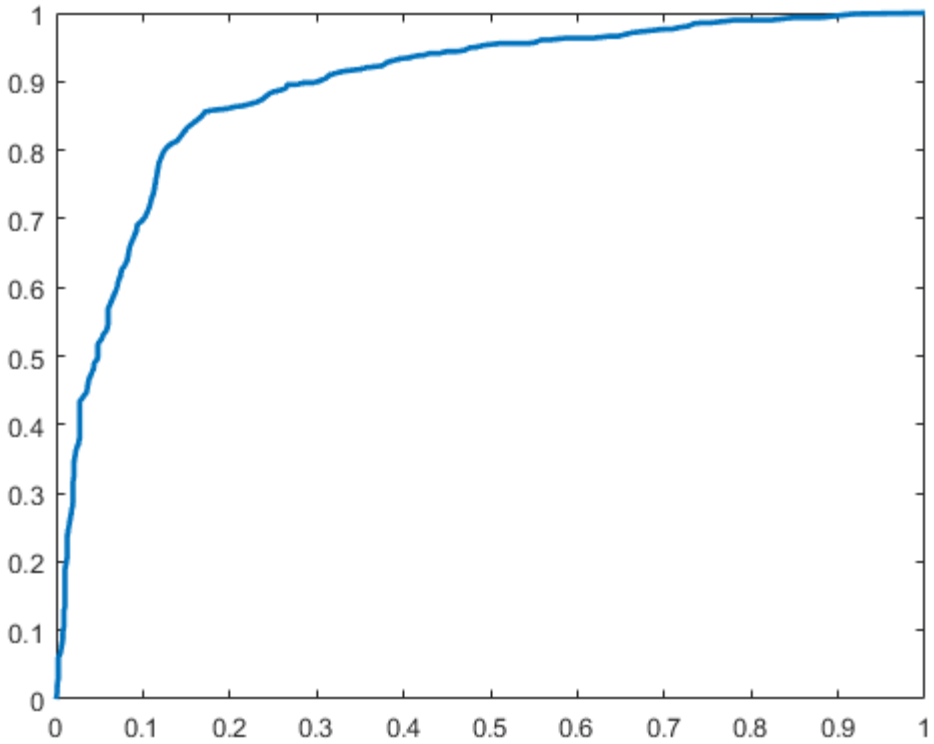

Figura 8.30: Curva ROC media

En este caso el AUC de esta curva ROC media es 0.8935, muy parecido al valor obtenido haciendo la media de los valores de AUC de cada una de las curvas.

8.3. Análisis comparativo

Por último, y para justificar el uso del casco Neurosky, se va a proceder a comparar los datos obtenidos con los obtenidos por el sistema de EEG "Handy EEG SD32", un sistema que presenta una frecuencia de muestreo de 2048Hz (4 veces mayor a la del casco usado en este proyecto) y que cuenta con 32 canales frente al único canal con el que cuenta el Neurosky. Los datos obtenidos por el sistema Handy EEG SD32 se encuentran en el artículo "*Induced Gamma-Band Activity During Voluntary Movement: EEG Analysis for Clinical Purposes*" [2] y han sido facilitados por uno de sus autores, Carlos Amo, con el objetivo de determinar si los resultados del casco NeuroSky son aceptables. En concreto, este apartado se centra en las curvas de evolución de GBA para el caso basal y para el caso motriz (figura 8.14), ya que no se puede hacer una comparación dato a dato debido a la diferencia en unidades de salida de cada uno de los sistemas de EEG.

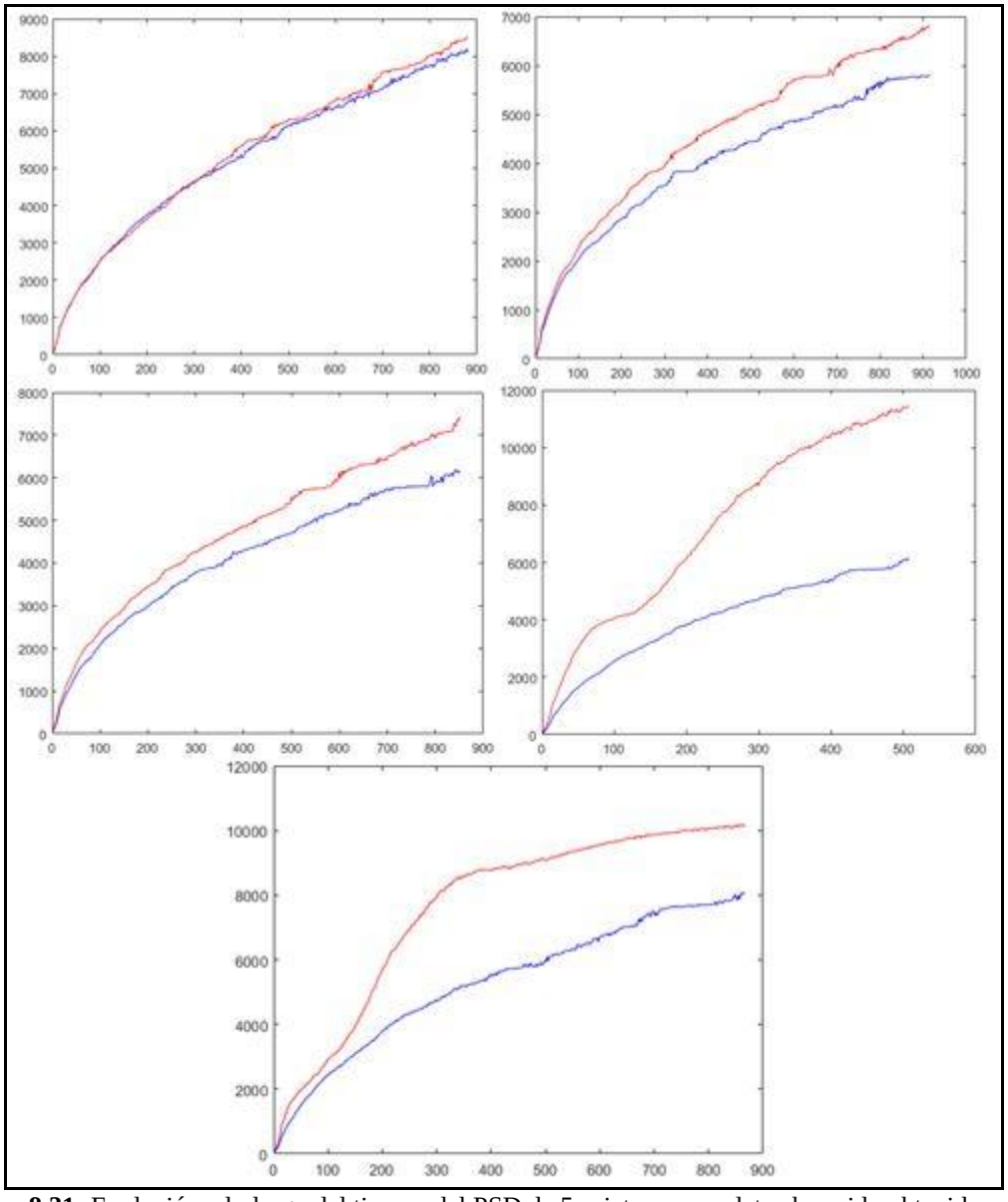

Figura 8.31: Evolución a lo largo del tiempo del PSD de 5 sujetos cuyos datos han sido obtenidos con el sistema EEG *"Handy EEG SD32"*

En este caso la duración de los experimentos es de 18 minutos separados en 3 intervalos de 6 minutos cada uno.

En la figura 8.31 se puede ver que las formas de las curvas obtenidas con este sistema y las obtenidas con el casco Neurosky son similares, teniendo en cuenta la diferencia de frecuencias de muestreo y de tiempo de los experimentos. Este hecho permite justificar el uso del casco Neurosky como sistema de adquisición de datos en el desarrollo de este proyecto.

9. Detección de GBA inducida en tiempo real

En este apartado se explicarán las pruebas que se han llevado a cabo para intentar detectar GBA inducida en tiempo real. La duración de cada una de estas pruebas ha sido de 120 segundos.

9.1. Procedimiento de la prueba de detección de GBA

Estas pruebas duran un total de 120 segundos y consisten en partir de un estado basal, realizar movimiento hasta activar la GBA inducida, y volver al estado basal tras haber realizado movimiento.

En estas pruebas se ha tratado de determinar tres variables que serían, el número de movimientos a realizar por el sujeto, el tiempo de inicio de movimiento y la ventana temporal de observación de la señal sobre la que se calcula el PSD. Todo esto se muestra en la figura 9.1

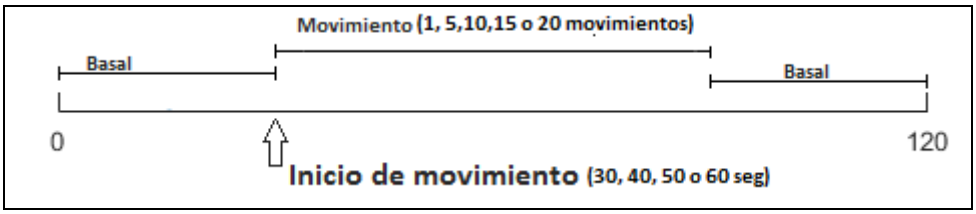

Figura 9.1: Esquema gráfico del procedimiento seguido para realizar las pruebas con las variables "inicio de movimiento" y "movimientos realizados"

Estas pruebas combinan los dos experimentos anteriores y el inicio del movimiento es indicado al sujeto mediante el parpadeo en rojo del punto en la pantalla que parpadea cada 2 segundos y deja de parpadear cuando el sujeto debe volver al estado basal. La posición del sujeto frente al ordenador debe ser la misma que en los experimentos anteriores.

Las primeras pruebas se centrarán en el número de aperturas y cierres de mano y en el tiempo en reposo anterior al movimiento. Combinando estas dos variables se obtiene la tabla 9.1, con todas las pruebas que se deben realizar.

Nº de Aperturas / Tiempo	1	5	10	15	20
30	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
40	Prueba 6	Prueba 7	Prueba 8	Prueba 9	Prueba 10
50	Prueba 11	Prueba 12	Prueba 13	Prueba 14	Prueba 15
60	Prueba 16	Prueba 17	Prueba 18	Prueba 19	Prueba 20

Tabla 9.1: Tabla de pruebas a realizar para determinar el tiempo en reposo y el número de movimientos óptimos para la detección de GBA inducida

9.2 Procesamiento de las señales obtenidas con el casco.

El procesado llevado a cabo en estas pruebas consiste en coger una ventana de observación temporal de 10 segundos que se va desplazando en incrementos de 0.2 segundos y sobe la cual se calcula el valor medio del PSD y se guarda en un vector, proceso que se repite hasta llegar al final de la señal raw recogida con el casco como se muestra en el diagrama de bloques de la figura 9.2

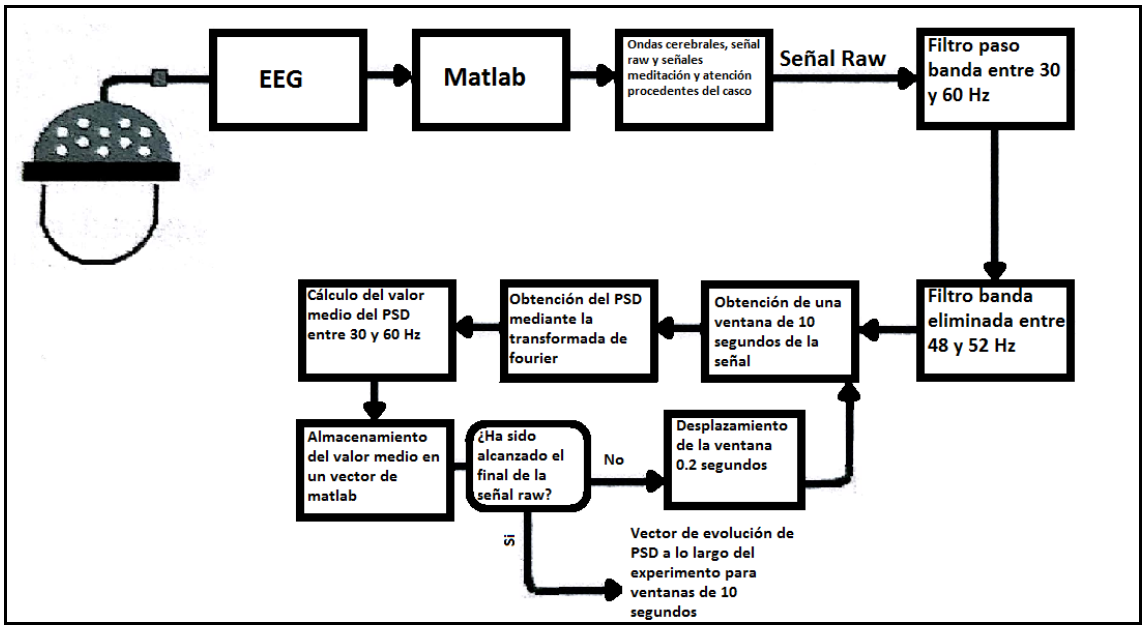

Figura 9.2: Diagrama de bloques del proceso seguido para calcular la GBA inducida en tiempo real

Tras todo este procesado, al vector obtenido se le ha aplicado un filtro paso bajo a 10Hz para eliminar los cambios bruscos y un filtro paso alto a 1Hz para eliminar la componente continua.

9.2. Elección del número de movimientos a realizar

Para este apartado se hablará de las pruebas de la tabla 9.1 refiriéndose a ellas por su número.

Para la elección del número de movimientos se han realizado pruebas moviendo la mano 1, 5, 10, 15 y 20 veces obteniendo como resultado las siguientes gráficas.

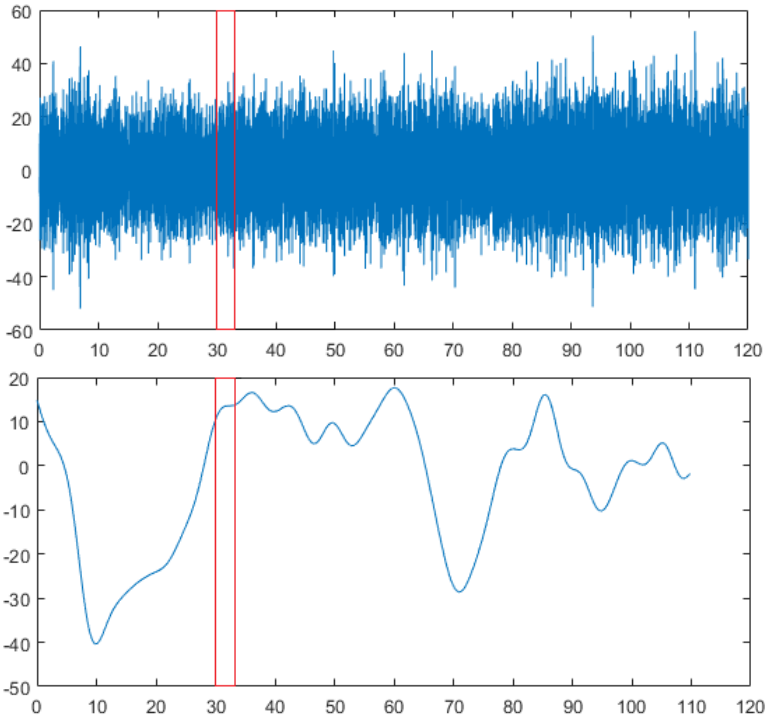

Figura 9.3: Gráfica de un sujeto que ha realizado un único movimiento. El movimiento ha sido realizado en la zona marcada por el rectángulo rojo. La gráfica de arriba muestra la señal raw original y la de abajo muestra la evolución del valor medio del PSD

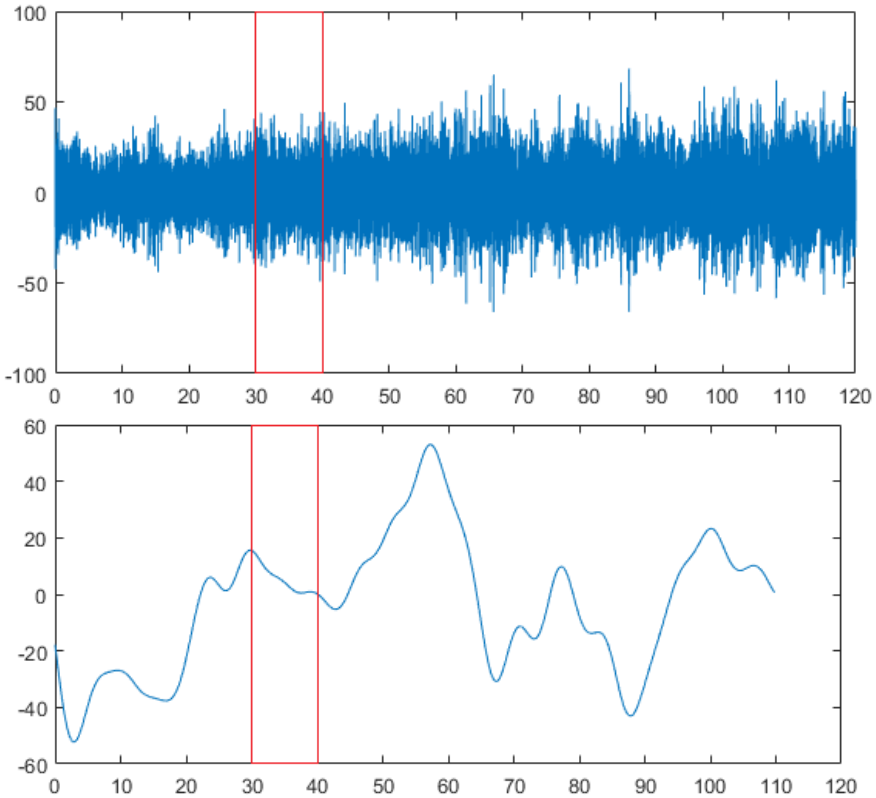

Figura 9.4: Gráfica de un sujeto que ha realizado 5 movimientos. El movimiento ha sido realizado en la zona marcada por el rectángulo rojo. La gráfica de arriba muestra la señal raw original y la de abajo muestra la evolución del valor medio del PSD

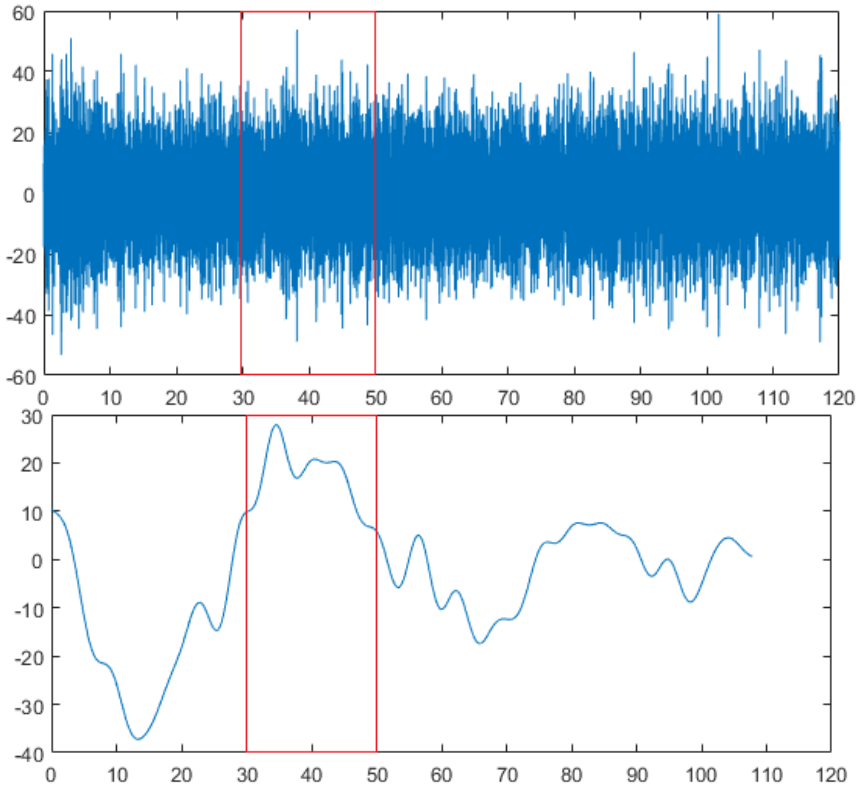

Figura 9.5: Gráfica de un sujeto que ha realizado 10 movimientos. El movimiento ha sido realizado en la zona marcada por el rectángulo rojo. La gráfica de arriba muestra la señal raw original y la de abajo muestra la evolución del valor medio del PSD

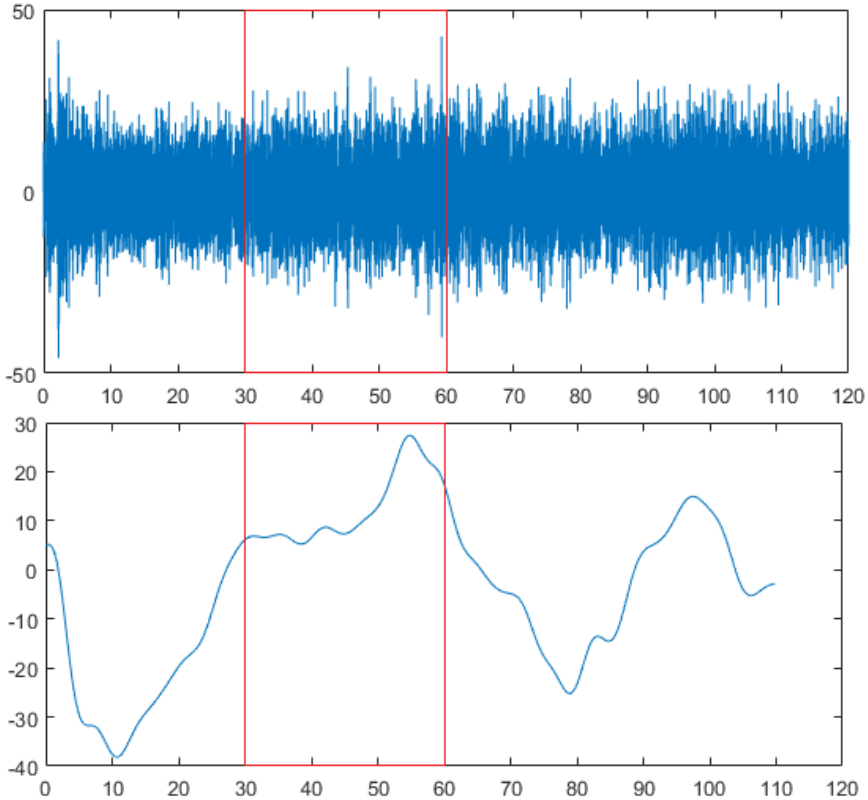

Figura 9.6: Gráfica de un sujeto que ha realizado 15 movimientos. El movimiento ha sido realizado en la zona marcada por el rectángulo rojo. La gráfica de arriba muestra la señal raw original y la de abajo muestra la evolución del valor medio del PSD

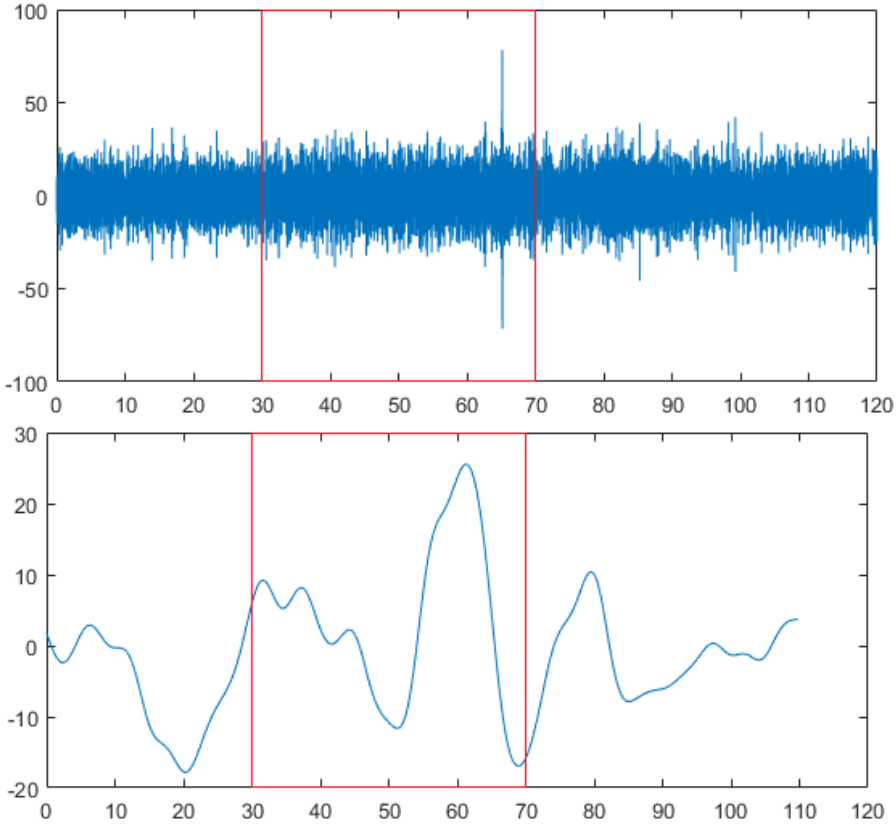

Figura 9.7: Gráfica de un sujeto que ha realizado 20 movimientos. El movimiento ha sido realizado en la zona marcada por el rectángulo rojo. La gráfica de arriba muestra la señal raw original y la de abajo muestra la evolución del valor medio del PSD

Observando estas gráficas se puede ver que a partir de 10 movimientos hay un claro pico en la señal y, por tanto, permite justificar que, cuando un sujeto realiza 10 movimientos o más, este ha activado la GBA inducida, aunque el número final de movimientos que debe realizar el sujeto para activar la GBA depende exclusivamente de la capacidad del propio sujeto. Se puede apreciar que el tiempo en movimiento es el doble del número de movimientos, pues se realiza un movimiento cada 2 segundos.

Para la elección del tiempo de inicio de movimiento se han realizado pruebas comenzando a mover la mano a los 40, 50 y 60 segundos, obteniendo resultados similares a los obtenidos para las pruebas realizadas iniciando el movimiento a los 30 segundos, por tanto, se ha escogido un tiempo de inicio de movimiento de 30 segundos, ya que es suficiente para obtener los resultados deseados.

9.3. Elección de la ventana de observación

Para elegir la ventana temporal de observación, se ha fijado el tiempo de inicio de movimiento a los 30 segundos y el número de movimientos a realizar por el sujeto en 10, como se justificó en el apartado anterior. En la figura

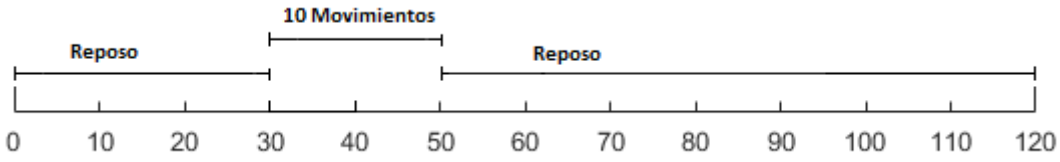

Figura 9.8: Esquema del procedimiento para realizar la segunda parte de las pruebas

En esta segunda fase de pruebas, el diagrama de bloques es el mismo que el mostrado en la figura 9.2, salvo que en este caso la ventana no está fija en 10 segundos. El resto del procesado es exactamente el mismo.

En este caso se van a hacer pruebas con ventanas de 2, 4, 6, 8, 10 y 20 segundos, dando como resultado las siguientes gráficas (figuras 9.9 a 9.14).

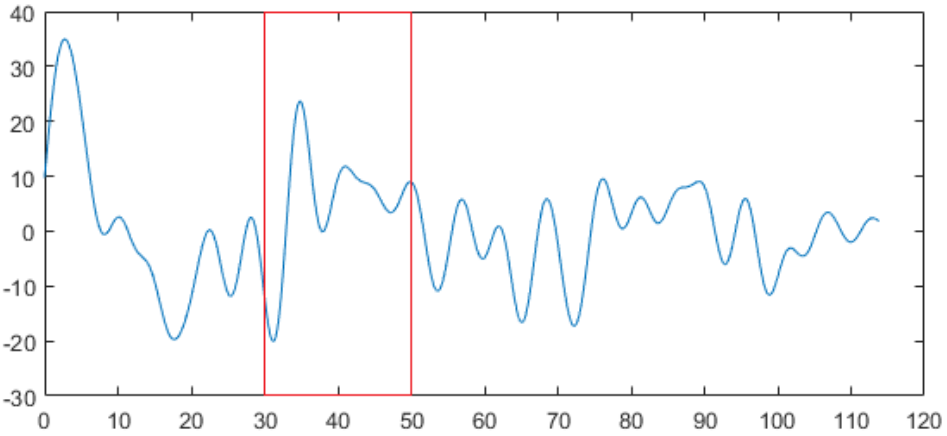

Figura 9.9: Ventana de 2 segundos. En el recuadro rojo está marcado el tiempo durante el que el sujeto realiza movimiento

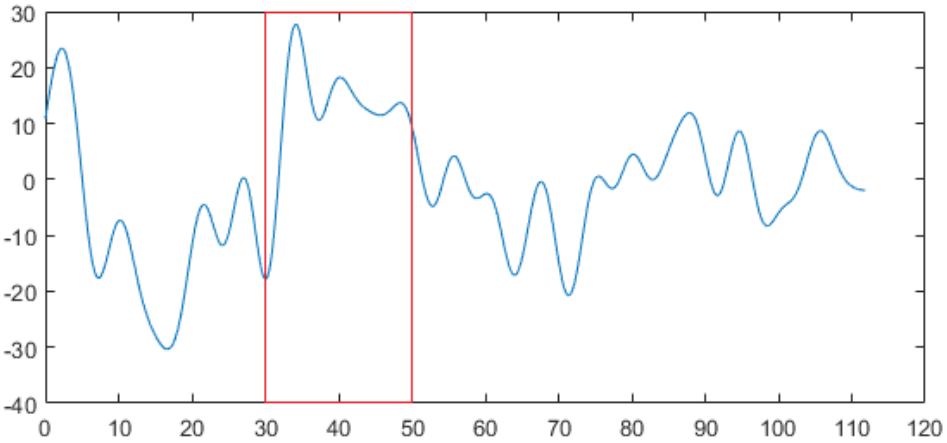

Figura 9.10: Ventana de 4 segundos. En el recuadro rojo está marcado el tiempo durante el que el sujeto realiza movimiento

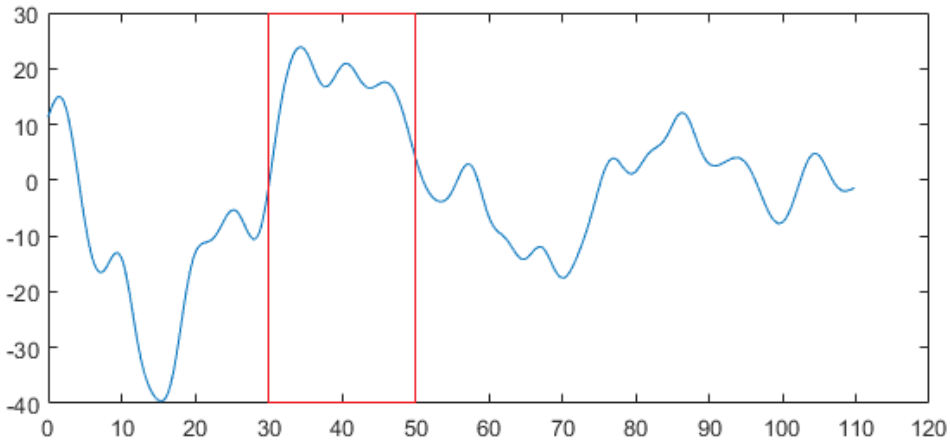

Figura 9.11: Ventana de 6 segundos. En el recuadro rojo está marcado el tiempo durante el que el sujeto realiza movimiento

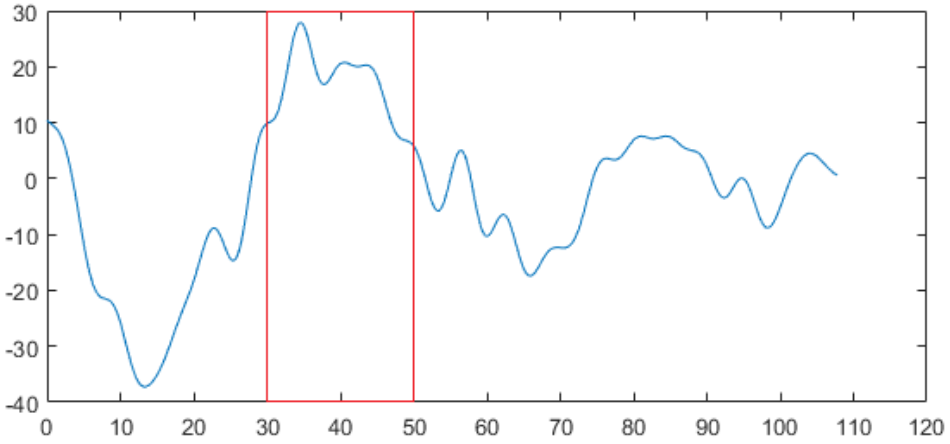

Figura 9.12: Ventana de 8 segundos. En el recuadro rojo está marcado el tiempo durante el que el sujeto realiza movimiento

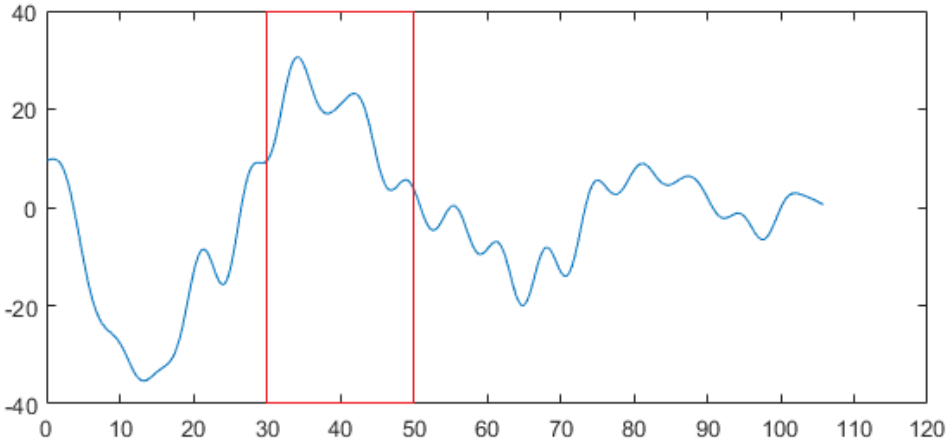

Figura 9.13: Ventana de 10 segundos. En el recuadro rojo está marcado el tiempo durante el que el sujeto realiza movimiento

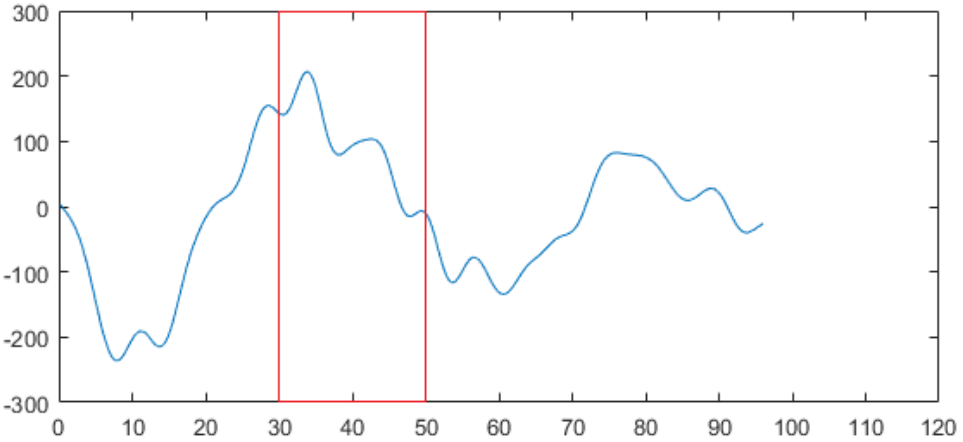

Figura 9.14: Ventana de 20 segundos. En el recuadro rojo está marcado el tiempo durante el que el sujeto realiza movimiento

De estas gráficas se puede determinar que la ventana de observación a partir de 8 segundos presenta un claro pico con respecto al resto de la señal, pero si se amplía hasta 20, la diferencia de este pico con respecto al resto queda más clara. Por tanto, la ventana de observación elegida ha sido de 20 segundos para proceder al estudio en tiempo real.

Por tanto y tras este estudio se han establecido las condiciones siguientes para el procesamiento final de las señales de control:

- Tiempo de inicio de movimiento: 30 segundos.
- Número de movimientos realizados con la mano: 10 movimientos.
- Ventana temporal de observación del PSD: 20 segundos.

9.4. Selección de señal de activación y desactivación

Por último, una vez se han obtenido los vectores de evolución de la GBA, estos son convolucionados con el semiciclo positivo de un seno (figura 9.15), obteniendo de esta manera una señal en la que se pueden diferenciar de manera clara el estado basal del estado en movimiento, obteniendo así la señal de control binaria cuando la GBA supera cierto umbral.

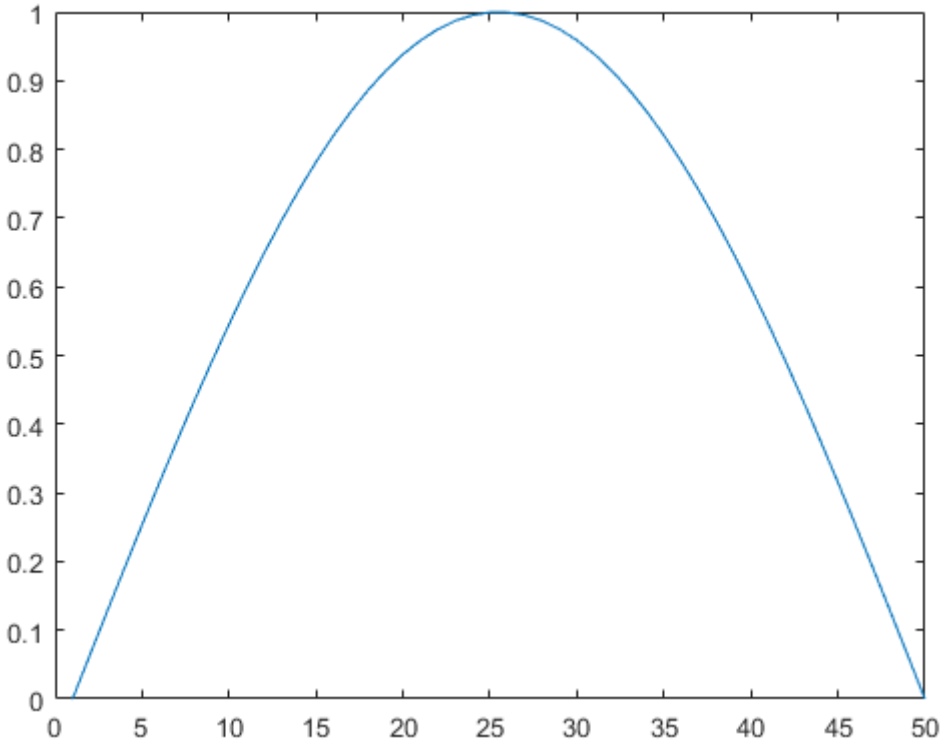

Figura 9.15: Semiciclo positivo de un seno

En la figura 9.16 se puede observar la evolución de la GBA a lo largo del experimento, y el resultado de la convolución de esta señal con un seno.

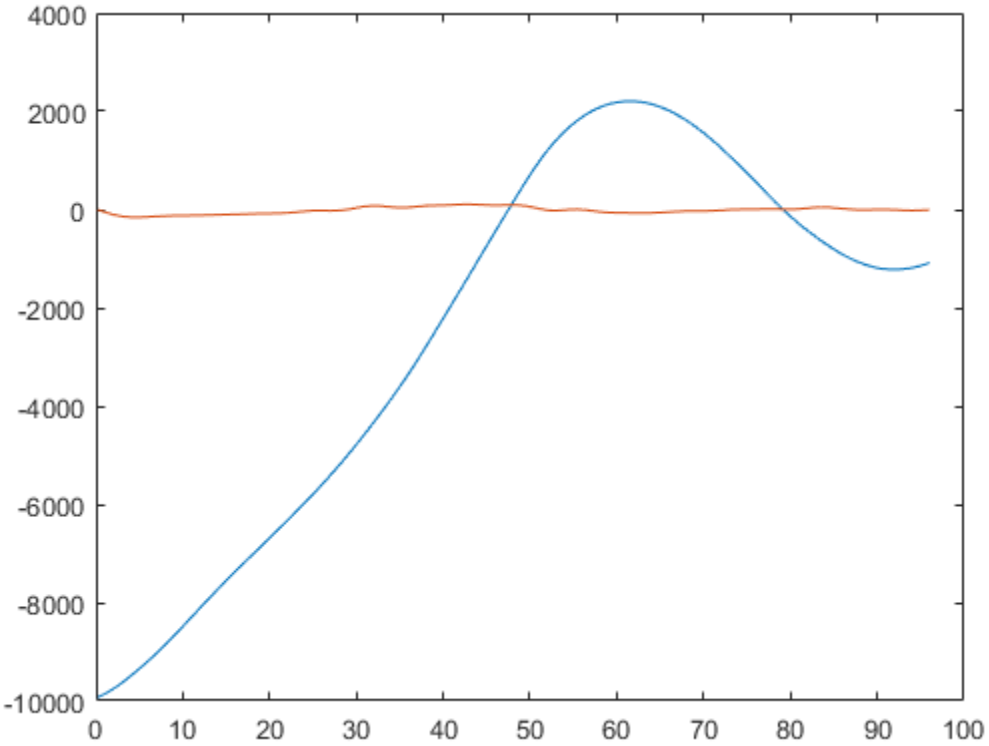

Figura 9.16: Evolución del PSD a lo largo del experimento (rojo) y señal resultante de la convolución de este PSD con el semiciclo positivo de un seno (azul).

En la figura 9.17 podemos observar en un estudio más minucioso las gráficas obtenidas de hacer esta convolución con los resultados de 4 pruebas distintas.

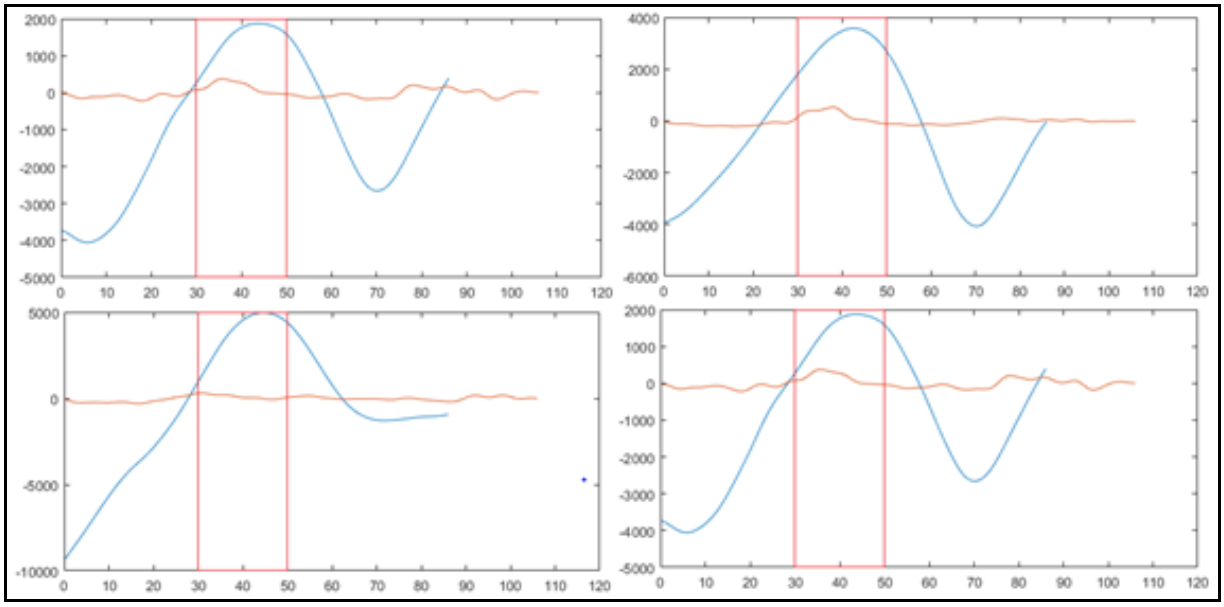

Figura 9.17: Ejemplos de la señal obtenida como resultado de la convolución de 4 pruebas distintas. En el recuadro rojo se marca la zona en la que el sujeto ha realizado los 10 movimientos.

Como se puede observar en la figura 9.17, la detección de movimiento tiene un breve retraso, de unos 10 segundos. Este retraso de 10 segundos es aceptable debido a que Matlab es un programa que funciona de manera secuencial y, por tanto, necesita un tiempo para procesar toda la señal.

Por otro lado, el umbral de activación de la GBA inducida se ha fijado en 1000 a priori, ya que como se puede observar en las figuras el valor de la señal resultante de la convolución supera en todos los casos este valor. Aun así, deberán realizarse algunas pruebas para fijar de nuevo este umbral cuando el sujeto cambie, pues cada persona es distinta y puede tener un umbral de activación y desactivación distinto.

En la figura 9.18 se puede ver el momento exacto en el que la mano se activa y se desactiva.

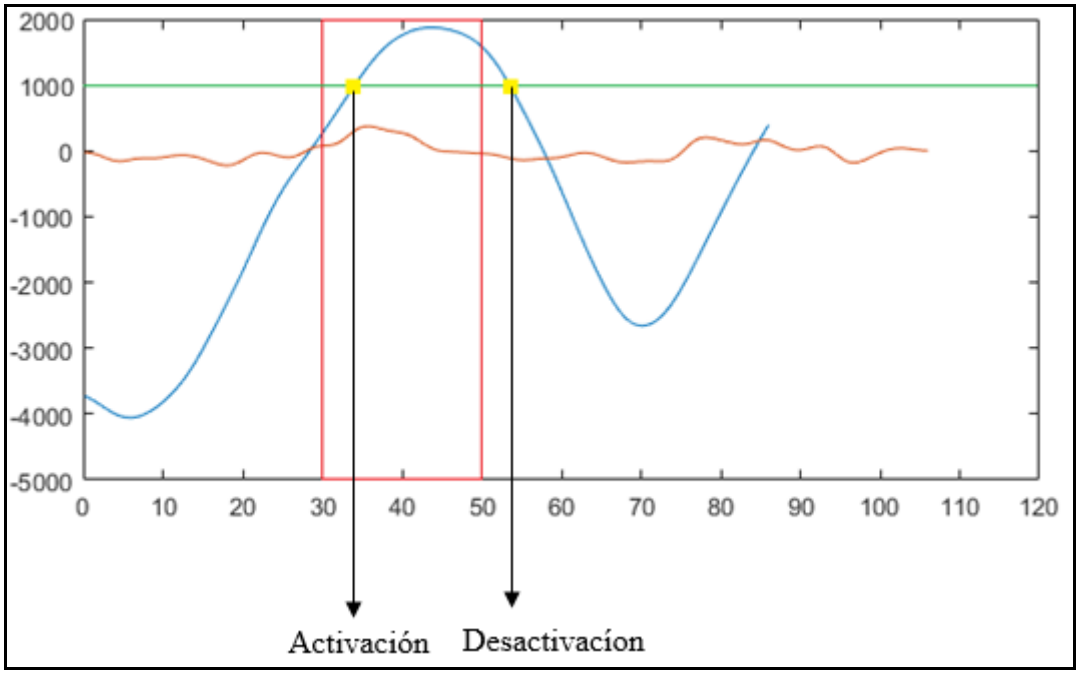

Figura 9.18: Umbral de activación y desactivación de la GBA inducida

10. Control de la mano robot Inmoov

El objetivo de este proyecto como se ha dicho anteriormente es conseguir controlar la mano del robot InMoov en tiempo real haciendo uso del casco NeuroSky y una interfaz gráfica de Matlab. Para conseguir esto el usuario debe ser capaz de inducir actividad en la banda gamma abriendo y cerrando su mano derecha. Una vez conseguido esto la señal de control será enviada a través del socket TCP/IP a RobotStudio donde la mano robot ejecutará el movimiento deseado.

10.1. Sistema de adquisición de datos

El sistema de adquisición de datos es el casco NeuroSky Mindset del que se habló anteriormente. En la página de NeuroSky contamos con una serie de aplicaciones que nos ayudan, mediante entrenamiento, a mejorar el control de nuestras ondas cerebrales. Estas actividades están pensadas para que el usuario sea capaz de modificar a voluntad su nivel de atención y meditación.

Entre las aplicaciones gratuitas que ofrece NeuroSky, en el desarrollo de este proyecto se han utilizado las siguientes para familiarizarse con el funcionamiento del casco:

- Brainwave visualizer
- Adventures of Neuroboy

10.1.1. Aplicaciones Neurosky

Brainwave Visualizer

Brainwave visualizer es una aplicación que muestra una representación gráfica de las ondas cerebrales del sujeto (figura 10.1), tanto la señal original obtenida de su cerebro, como el nivel de cada una de las distintas ondas cerebrales que la forman (alfa, beta, gamma, delta, theta). Esta aplicación también muestra el nivel de atención y meditación, y permite escuchar música y ver cómo afecta a las ondas cerebrales.

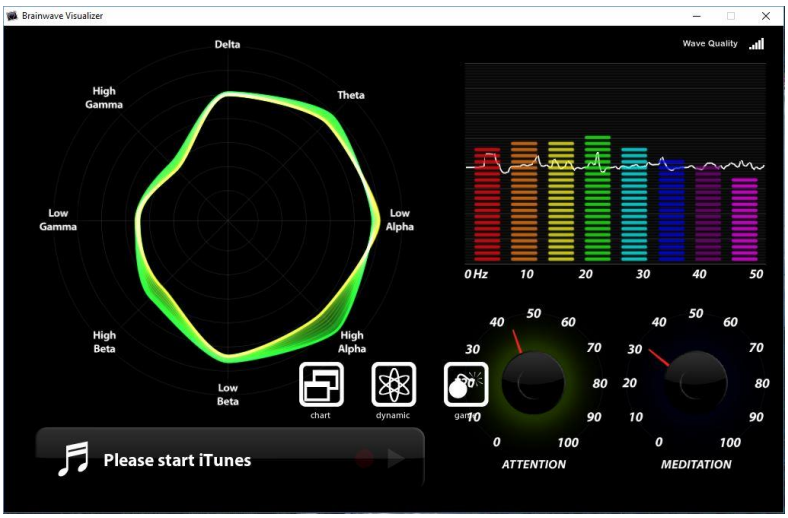

Figura 10.1: Interfaz del programa Brainwave visualizer

Dentro de la aplicación hay dos juegos que permiten al usuario controlar su nivel de atención y meditación. El primero (figura 10.2) consiste en hacer explotar un barril mediante la atención y el segundo (figura 10.3) permite hacer levitar una pelota cuanto más tiempo permanezca el sujeto en un estado de meditación.

Figura 10.2: Juego de entrenamiento de atención.

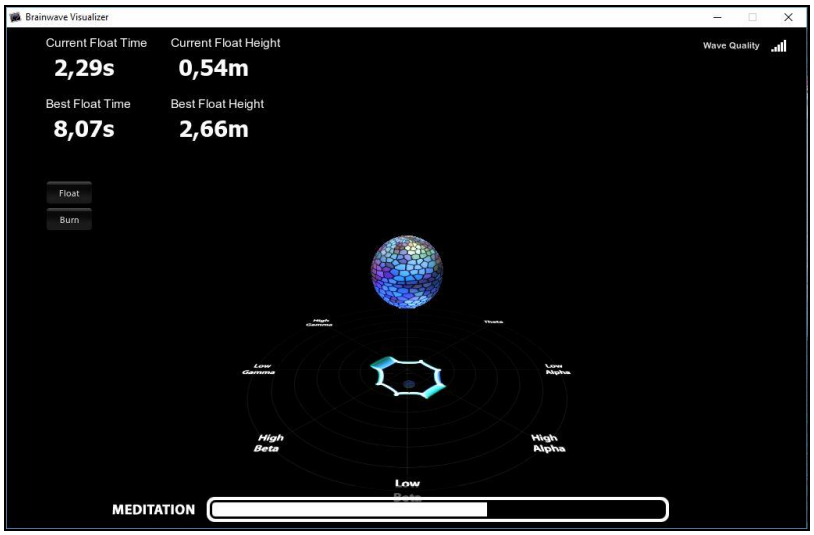

Figura 10.3: Juego de entrenamiento de meditación.

Estas aplicaciones sirven de entrenamiento para el sujeto, que con práctica podrá aumentar su nivel de atención a voluntad o permanecer en un estado de meditación durante más tiempo.

Adventures of NeuroBoy

Adventures of NeuroBoy es un pequeño juego en el que el personaje está en un parque rodeado de objetos como coches o cajas y tiene cuatro poderes que puede usar seleccionándolos con el ratón y activarlos mediante sus niveles de atención o meditación, dependiendo del poder elegido.

Por ejemplo, se puede hacer explotar un coche mediante la concentración (figura 10.4) o hacer levitar una pelota mediante la meditación (figura 10.5).

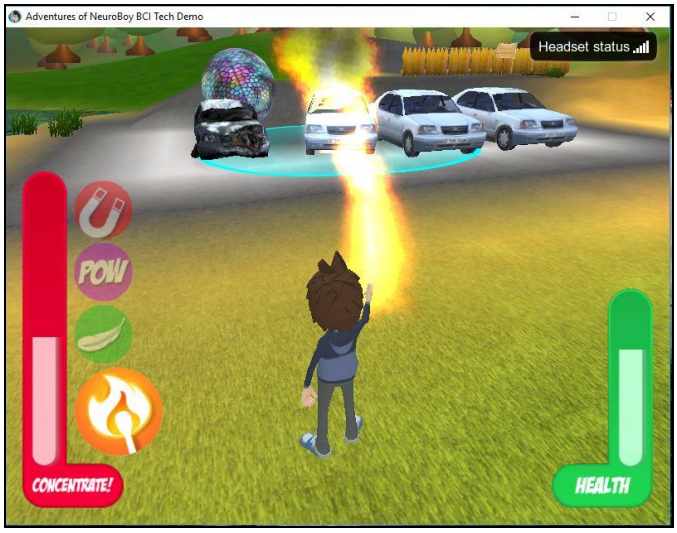

Figura 10.4: Juego de atención de NeuroBoy

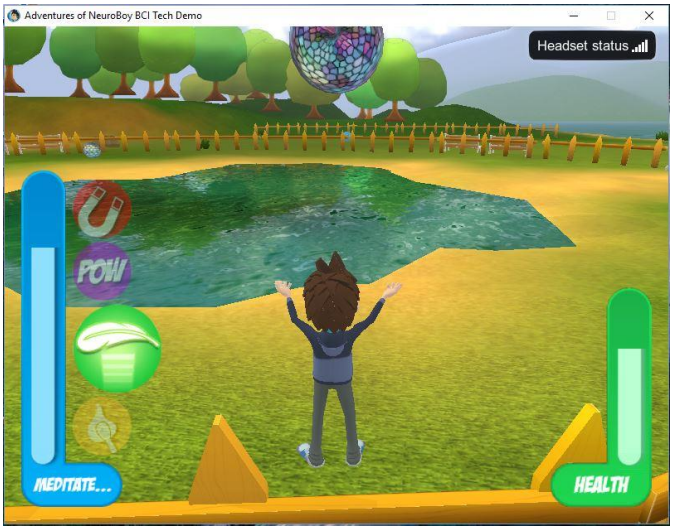

Figura 10.5: Juego de meditación de NeuroBoy

El objetivo de este juego, la igual que el de la aplicación Brainwave no es otro que el de entrenarse en el control de los niveles de atención y meditación pues, como ya se explicó, el casco Mindset es un sistema BCI endógeno y, por tanto, la actividad cerebral registrada depende de la práctica que tenga el sujeto a la hora de controlarla.

En el desarrollo de este proyecto, aunque no se utilicen los niveles de atención y meditación como señal de control, hacer uso de estas aplicaciones puede ser útil para poder controlar con mayor facilidad nuestras ondas cerebrales.

10.2. Protocolo de comunicación Matlab/RobotStudio

En este apartado se va a explicar de manera general todo lo relacionado con el control del robot IRB120. Normalmente, para controlar el brazo robot se escribe un programa en código RAPID y este se simula en RobotStudio para posteriormente cargarlo en el controlador real del robot, el IRC5. Sin embargo, en este caso el control se llevará a cabo desde el software de Matlab.

Matlab permite realizar todo el procesado de las señales que se obtienen mediante el casco Neurosky y generar a partir de ellas las señales de control que se enviarán a RobotStudio. Para realizar esta comunicación se usará un socket para comunicarnos con el servidor utilizando el protocolo TCP/IP (figura 10.6).

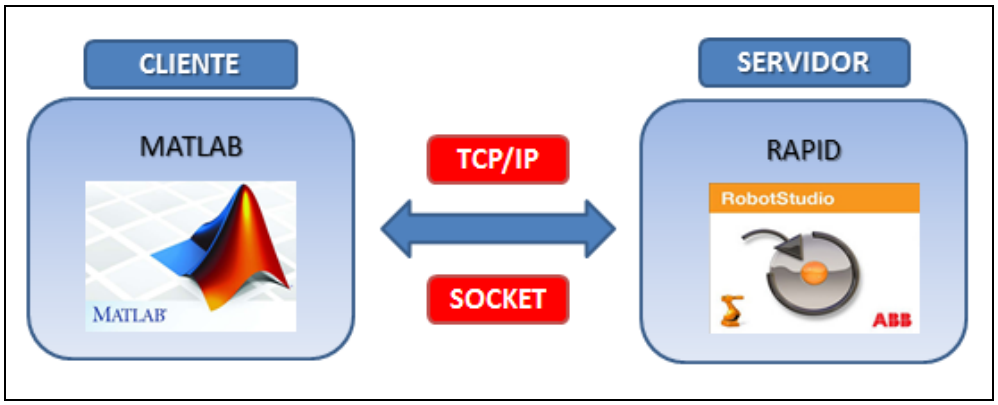

Figura 10.6: Comunicación Matlab/RobotStudio

El socket de comunicación que se ha utilizado en este proyecto es el desarrollado y expuesto en el TFG *"Desarrollo de una interfaz para el control del robot IRB120 desde MATLAB",* por lo que en este apartado se abordará el tema de manera general.

En este proyecto todo el control se ha realizado desde Matlab, haciendo uso de las funciones escritas en este programa, sin entrar en detalle en el código RAPID. En resumen, el código RAPID recibe unas cadenas de datos llamadas datagramas que pueden ser enviados a través de la red y es en estos datagramas donde se encuentra la información tanto de la posición del efector final del robot como de la acción que tiene que realizar la herramienta.

En el código de Matlab podemos encontrar las siguientes funciones que nos permitirán enviar esta información de manera más intuitiva:

Lo primero es conectar Matlab con RobotStudio. Para realizar esta conexión se usarán las clases creadas en el TFG *"Desarrollo de una interfaz para el control del robot IRB120 desde MATLAB",* en el que se observa que para que se produzca la conexión hay que indicarle la **dirección IP** y el **puerto**.

```
 %Conectar con el simulador
robot=irb120('127.0.0.1',1024);

 %Conectar con el robot real
robot=irb120('172.29.28.185',1024);

 %Conectar con el robot
robot.connect;
```

En este proyecto solo se utilizará la conexión con el simulador, no con el robot real.

Además, también podemos encontrar en dicho TFG la función **TCProtandpos** para enviar la posición y orientación del efector final del robot.

```
 %Enviarle una posición de destino
robot. TCProtandpos([180 0 180 X Y Z])
```

Los **tres primeros elementos (180 0 180)** indican la orientación del efector final, a partir de la rotación sobre el eje X, Y y Z respectivamente, es decir, una rotación de 180 sobre el eje X, 0 sobre el eje Y y 180 sobre el eje Z, en grados. Los **tres últimos elementos del vector (X Y Z)** indican la posición del efector final respecto a los ejes de coordenadas X, Y, y Z respectivamente, en milímetros.

Por último, se utiliza la función **robot.TCPtool** para controlar la función realizada por la mano robot:

```
 %Control de herramienta
robot.TCPtool(1)   %Abrir mano
robot.TCPtool(0)   %Cerrar mano
```

Haciendo uso de estas funciones y una interfaz gráfica, se puede llegar a controlar la mano robot haciendo uso de las señales recogidas mediante el casco Mindset una vez estas son procesadas.

10.3. Obtención de las señales de control

Para obtener las señales de control en tiempo real, es necesario que el programa evalúe en cada momento la señal obtenida y la convolucione con el semiciclo positivo de un seno. Para esto, en lugar de convolucionar toda la evolución de la GBA con la señal seno como se hacía en el apartado 9.2 , se convolucionan intervalos de 10 segundos con un seno, y se va desplazando esta ventana de 10 segundos en incrementos de 200 milisegundos a lo largo de toda la señal de evolución de la GBA. En las figuras 10.7 y 10.8 se pueden observar los resultados de esta convolución en un caso basal y en un caso motriz.

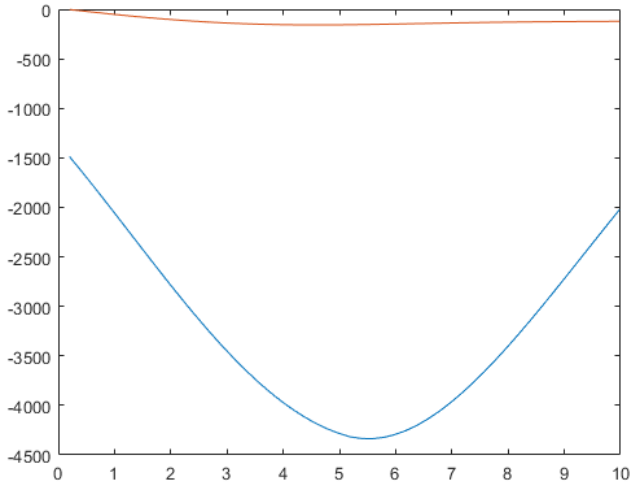

Figura 10.7: Resultado de la convolución cuando no hay movimiento

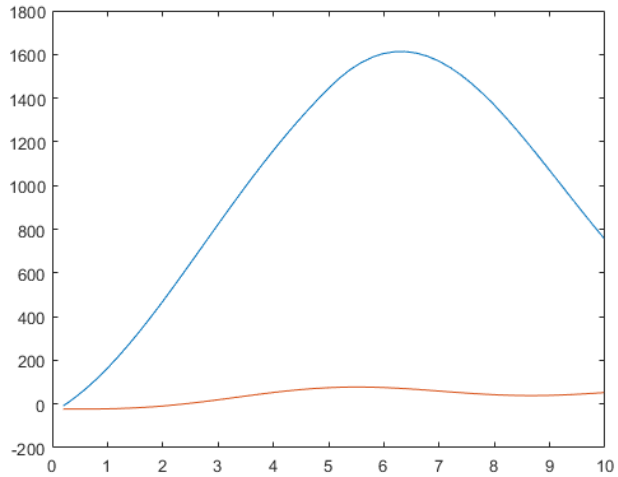

Figura 10.8: Resultado de la convolución cuando el sujeto realiza movimiento

Las señales obtenidas son ligeramente diferentes a las que se conseguían haciendo la convolución con todo el vector, pero se puede observar que de esta manera también se puede conseguir una señal de control binaria cuando la señal supere un umbral.

En cualquier caso, este umbral debe ser ajustado tras realizar las pruebas con el sujeto, ya que las señales obtenidas, aunque se parecen, no son iguales para todos los sujetos y de la misma manera, no son iguales para el mismo sujeto en dos momentos diferentes del día.

Una vez realizadas estas pruebas, se ha escrito un programa para que el proyecto funcione en tiempo real, que sigue los siguientes pasos. Primero, los datos son recogidos durante 40 segundos para que se pueda formar el inicio del vector de evolución de la GBA. Durante este tiempo el usuario puede mirar el cuadrado blanco que se utilizaba en experimentos anteriores para relajarse y permanecer en estado basal. Cuando han acabado esos 40 segundos, el cuadrado parpadea en rojo una vez, avisando al usuario de que ya puede comenzar a realizar movimiento, o continuar en estado basal. Los datos son recogidos del sujeto mientras realiza acciones y son procesados cada 200 milisegundos, utilizando una ventana de 20 segundos y continuando con la formación del vector de evolución de GBA. El resultado de la convolución es mostrado detrás del cuadrado blanco para que el usuario pueda saber en todo momento lo cerca que esta de su umbral de activación y desactivación de la señal de control (figura 10.9).

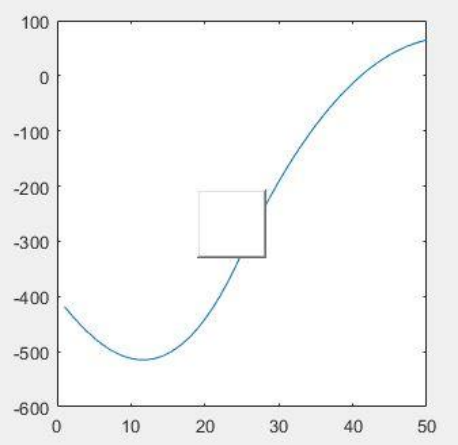

Figura 10. 9: Señal de cierre de la mano

Este vector es procesado como se ha explicado anteriormente, utilizando un filtro paso bajo a 10 Hz y un filtro paso alto a 1 Hz, con la diferencia de que en este caso el filtro se aplica sobre un intervalo de 20 segundos del vector y posteriormente se descartan los 10 últimos segundos. Esto es debido a que, al no tener valores posteriores de la señal hasta que pasen los 200 milisegundos, cuando se aplica el filtro paso bajo, el final de la señal filtrada cambia bruscamente de forma según se van añadiendo nuevos datos. Si solo se aceptan los 10 primeros segundos, se evitan estos cambios bruscos que pueden dar lugar a un error en la señal de control.

Por último, cada vez que los datos son procesados, se busca en el vector de 10 segundos resultante algún valor que sobrepase el umbral y de ser así, el cuadrado blanco pasa a rojo, y se envía la señal a RobotStudio para que el robot cierre la mano. En este punto, si el usuario se vuelve a relajar, el cuadrado vuelve a blanco y la mano se abre (figura 10.10 y 10.11)

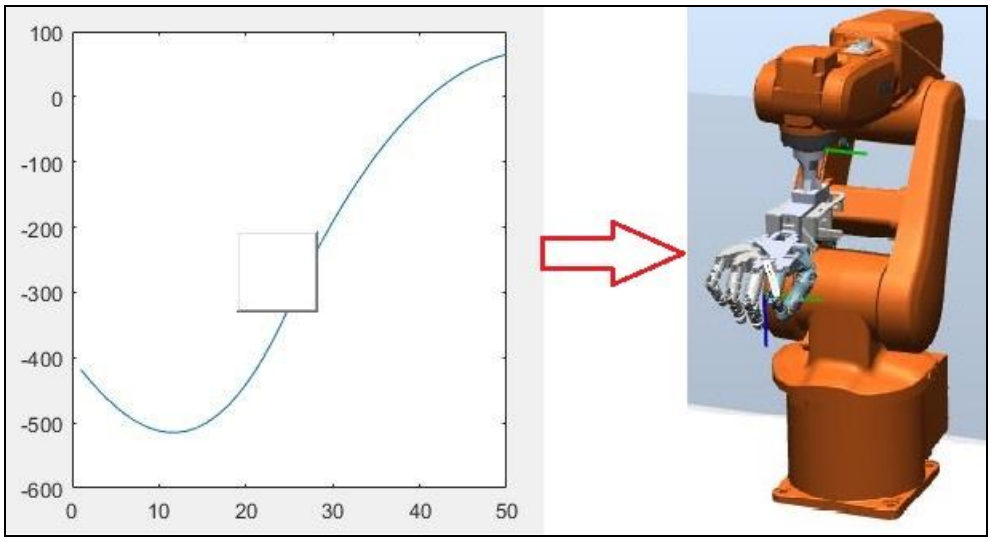

Figura 10.10: Cierre de la mano en RobotStudio

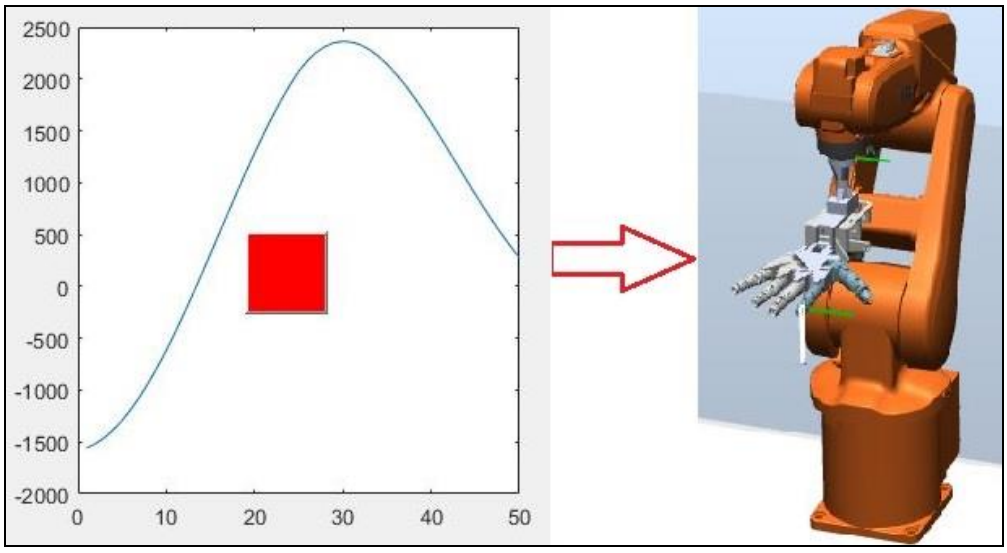

Figura 10.11: Apertura de la mano en RobotStudio

Para facilitar al usuario el uso de este programa, se ha desarrollado una interfaz gráfica en Matlab.

10.4. Interfaz Gráfica

Para crear las interfaces gráficas necesarias para este proyecto se ha hecho uso de la herramienta de Matlab GUIDE. Esta herramienta nos permite crear de manera sencilla una interfaz gráfica de usuario y, cuando salvamos la aplicación, automáticamente nos crea dos archivos, el archivo.m y el archivo.fig.

El archivo.fig contiene la descripción gráfica de la interfaz. En él se guarda la figura con los objetos gráficos (uicontrol o user interface control) insertados en ella, para su posterior modificación o uso.

El archivo.m contiene todo el código necesario para controlar y ejecutar la GUI. En este código están las callbacks de cada uno de los objetos gráficos de la figura y es el programador el que debe escribir código dentro de las callbacks para que los objetos lo ejecuten cuando sean pulsados. Al ejecutar este archivo aparece la figura en pantalla con los objetos gráficos que ya pueden ser utilizados.

En este proyecto se hace uso de dos interfaces gráficas, las cuales son muy parecidas, constan de los mismos botones y solo se diferencian en el tipo de acción que realiza el robot cuando recibe la señal de control generada por el usuario.

La primera interfaz gráfica es la mostrada en la figura 10.12

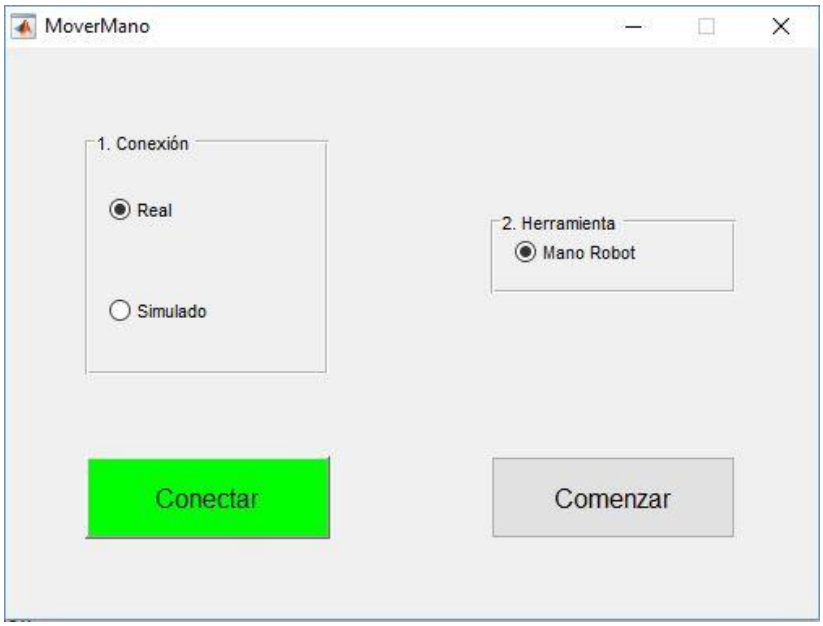

Figura 10.12: Interfaz gráfica de apertura y cierre de la mano.

Esta interfaz es muy sencilla, facilitando así su uso por parte del usuario. Tiene 2 cajas de selección, la primera nos permite seleccionar el tipo de conexión, pudiendo elegir si se quiere controlar el robot de la simulación o el robot real. La segunda selección es la de él tipo de herramienta utilizada. En este proyecto solo se ha incluido la mano robot, puesto que es la única herramienta que se va a controlar. En cualquier caso, la interfaz

se puede modificar de cara a un futuro poder controlar más herramientas con este método. La interfaz cuenta con dos botones, el botón “conectar” y el botón “comenzar”. El botón “conectar” inicia la comunicación entre Matlab y RobotStudio, realizando el primer movimiento de apertura y cierre de la mano y llevando el robot a una posición intermedia. El botón “comenzar” es el que inicia la recogida y procesado de los datos y debe ser el último en ser pulsado, para que los movimientos realizados sean mandados al robot y éste abra y cierre la mano.

La segunda interfaz es la mostrada en la figura 10.13.

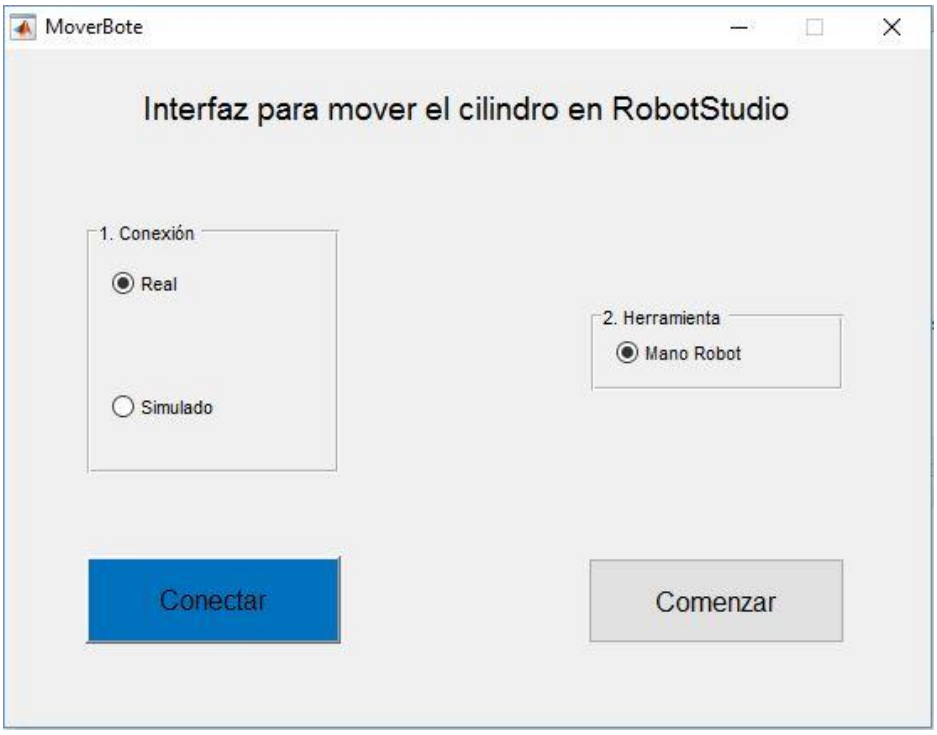

Figura 10.13: Interfaz gráfica de movimiento de cilindro solido en RobotStudio

Como se puede observar, los botones son los mismos que en la interfaz anterior. El funcionamiento es exactamente el mismo, con la diferencia de que, en este caso, al activar la señal de control el robot realiza un movimiento con el bote, moviéndolo de su posición inicial al otro lado del robot y, cuando se desactiva la señal de control, el robot devuelve el bote a su posición inicial.

10.5. Resultados

En este proyecto el programa solo es ejecutado en simulación, debido a las complicaciones de uso de la mano robot real, que, al estar impresa completamente en 3D y sin ningún tipo de sensor de fuerza o presión, hace que sea más complejo poder agarrar objetos correctamente.

10.5.1. Simulación

En este apartado se explicará paso a paso como debe ejecutarse el código para que el programa funcione correctamente en Matlab.

1er PASO: Conectar mediante bluetooth el casco Mindset al ordenador.

2º PASO: Arrancar RobotStudio y cargar la estación "*mano_final_vision.rspag*" y a su vez arrancar Matlab R2015b (32-bit).

3er PASO: En la pestaña simulación de RobotStudio, pulsar reproducir.

4º PASO: En Matlab, ejecutar el programa "fieldtrip_comenzar.m". Es necesario tener descargada la toolbox de Matlab "fieldtrip" y guardarla en la ruta del ordenador que lee ese programa.

5º PASO: En Matlab, ejecutar el programa "MoverMano.m" si se quieren realizar movimientos de apertura o cierre de la mano robot o el programa "MoverBote.m" si lo que se desea es mover el cilindro de la estación de lugar.

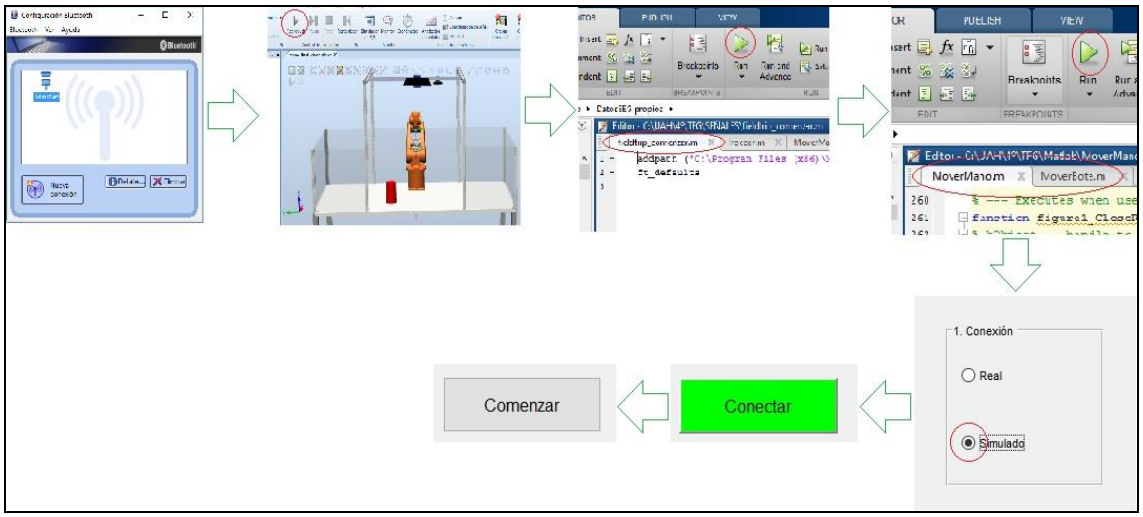

Figura 10.14: Diagrama de bloques del funcionamiento de la aplicación

6º PASO: Marcar la opción "*Simulación*" y pulsar el botón "*CONECTAR*".

7º PASO: Comprobar que se ha conectado el robot en RobotStudio (el robot realizará un movimiento de apertura y cierre de la mano y la mano permanecerá abierta en una posición intermedia).

8º PASO: Pulsar el botón comenzar, el casco comenzará a enviar la información al ordenador durante 40 segundos.

9º PASO: Pasado este tiempo, el usuario podrá realizar movimientos con su mano o bien permanecer en estado de reposo y el IRB120 actuará según haya mayor o menor GBA inducida.

11. Conclusiones y futuras líneas de trabajo

El objetivo de este proyecto era ser capaces de controlar mediante el casco Mindset la mano del robot InMoov, pudiendo abrirla y cerrarla haciendo uso de las ondas cerebrales.

Como se ha podido demostrar, se puede detectar la GBA inducida haciendo uso del casco NeuroSky Mindset, y, gracias a esto, el casco Mindset se puede utilizar para generar señales de control binarias aprovechando dicha GBA como es el caso de este proyecto. Esto puede significar una ventaja, pues con un sistema de bajo coste podemos generar una señal de control de dos estados, pero a su vez, el solo poder controlar dos estados nos limita.

Además, al ser un único sensor el que monitoriza la actividad cerebral, la señal que recibe el ordenador fluctúa de manera significativa, pudiendo dar falsos positivos o negativos.

En este caso, para poder controlar la mano robot del InMoov mediante el casco NeuroSky Mindset, es necesario que el sujeto practique varias veces, pues el sistema BCI utilizado es endógeno y como se explicó en el apartado 4, en la mayoría de los casos, el usuario necesita un entrenamiento previo para aprender a controlar su actividad cerebral, por esto es recomendable que primero se familiarice con el casco haciendo uso de las aplicaciones ofrecidas por NeuroSky.

Por otra parte, aunque Matlab permite controlar la mano robot de manera precisa, al ser un programa que funciona de manera secuencial nunca va a funcionar en tiempo real, habiendo siempre un retraso de unos segundos entre el inicio del movimiento del usuario y la recepción y procesado de los datos. De la misma manera, cuando se envía una instrucción desde Matlab a RobotStudio, Matlab tiene que esperar hasta que RobotStudio ejecute los movimientos en el robot para continuar con la ejecución de su código, perdiendo una cantidad importante de datos procedentes del casco. Esto resultó un inconveniente pues al tener que manejar una cantidad tan grande de datos el ordenador se ralentiza con el tiempo, lo cual supone una pérdida de datos añadida a la comentada anteriormente e incluso un bloqueo de la ejecución del código de Matlab, obligando a reducir el tiempo de funcionamiento del programa a 5 minutos.

Para mejorar este proyecto sería necesario hacer uso de un entorno de programación diferente, en el que las instrucciones puedan mandarse en paralelo y, mientras RobotStudio ejecuta los movimientos en el robot, se puedan seguir procesando los datos obtenidos con el casco. De la misma manera, aunque Matlab pierde muy pocos datos mientras se están procesando los que se han obtenido con anterioridad, esta pérdida podría evitarse si en el entorno de programación utilizado se pudiesen procesar los datos obtenidos mientras se siguen obteniendo nuevos datos, consiguiendo así un programa más rápido y preciso.

Actualmente en el campo de la ingeniería biomédica se están consiguiendo una gran cantidad de avances en el campo de la implantación y control de prótesis biónicas que incluso pueden hacer que el usuario tenga sensibilidad con la ayuda de unos sensores de presión, lo que le permite poder agarrar objetos o tocar a personas controlando la fuerza

como lo haría con su miembro real. Para controlar estas prótesis se utilizan cascos con un número más elevado de electrodos que proporcionan una señal más fiable, e incluso electrodos colocados quirúrgicamente.

Por otra parte, puede hacerse uso de algún tipo de sistema de aprendizaje, como puede ser una red neuronal, que permita personalizar de manera automática el umbral de activación y desactivación de cada usuario.

IV. Manual de usuario

12. Manual de usuario

En este apartado se va a redactar un pequeño manual de usuario, explicando los pasos a seguir para utilizar la aplicación, extendiendo un poco lo redactado en el apartado 10.5.1 donde ya se realizó un resumen que permite poner la aplicación en funcionamiento.

Lo primero es conectar el casco con el ordenador mediante bluetooth. Para conectarlo tan solo es necesario seguir las instrucciones que vienen con el propio casco, conectando primero el adaptador bluetooth que viene con el propio casco en caso de que el ordenador en el que se vaya a utilizar la aplicación no cuente con bluetooth de serie. Una vez hecho esto, con la aplicación "Configurador Bluetooth" que nos facilita NeuroSky lo único que tendremos que hacer será emparejar el casco con el ordenador, usando la opción de puerto serie y eligiendo el puerto COM de menor valor (figuras 12.1 y 12.2).

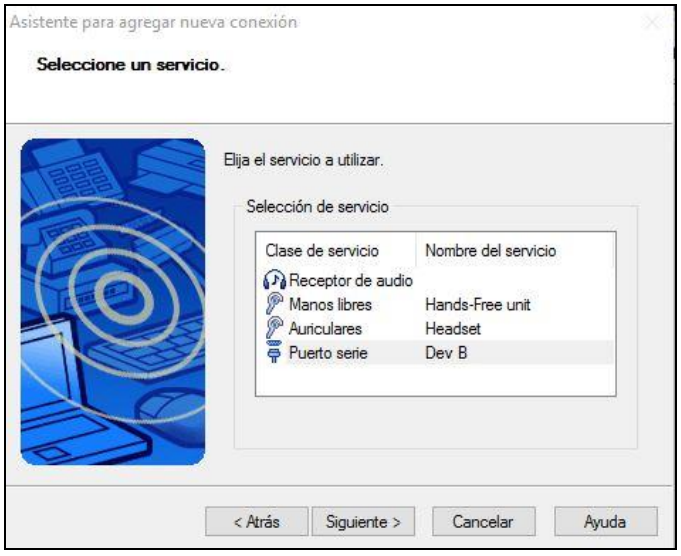

Figura 12.1: Emparejamiento del casco como puerto serie.

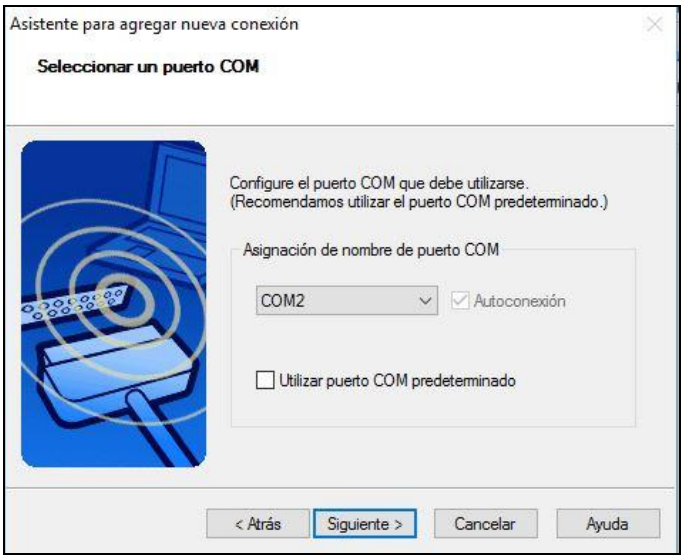

Figura 12.2: Elección del Puerto COM con el número más bajo.

Tras esto se debe abrir Matlab y RobotStudio. En RobotStudio se debe abrir con la opción unpack&work (figura 12.3) la estación "*mano_final_vision.rspag*" y pulsar el botón reproducir para que el programa se quede esperando instrucciones de Matlab (figura 12.4).

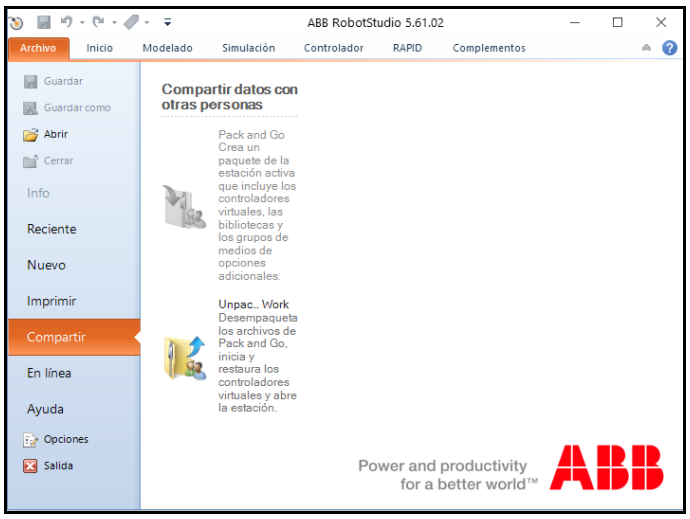

Figura 12.3: Unpack & Work en RobotStudio

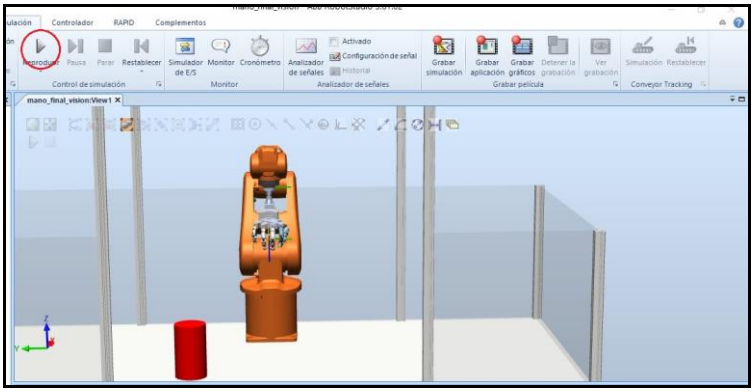

Figura 12.4: Estación final de RobotStudio.

Una vez hecho esto, en Matlab se carga el programa "*fieldtrip_comenzar.m"* (figura 12.5) que carga la toolbox necesaria para poder filtrar las señales obtenidas con el casco.

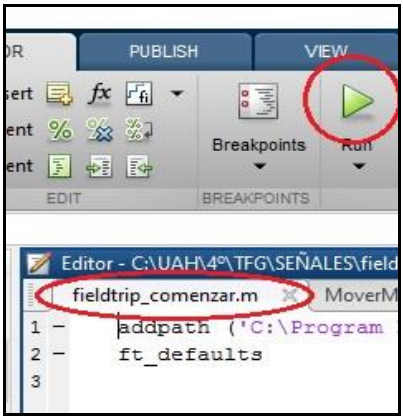

Figura 12.5: Toolbox de Matlab *"fieldtrip"*.

Tras esto, ya se puede empezar a utilizar la interfaz gráfica *"MoverMano.m"* o *"MoverBote.m"*. Al ser las dos interfaces gráficas iguales, salvo por la acción que realiza el robot cuando recibe la señal de control, se explicará únicamente la interfaz *"MoverMano.m"* (figura 12.6).

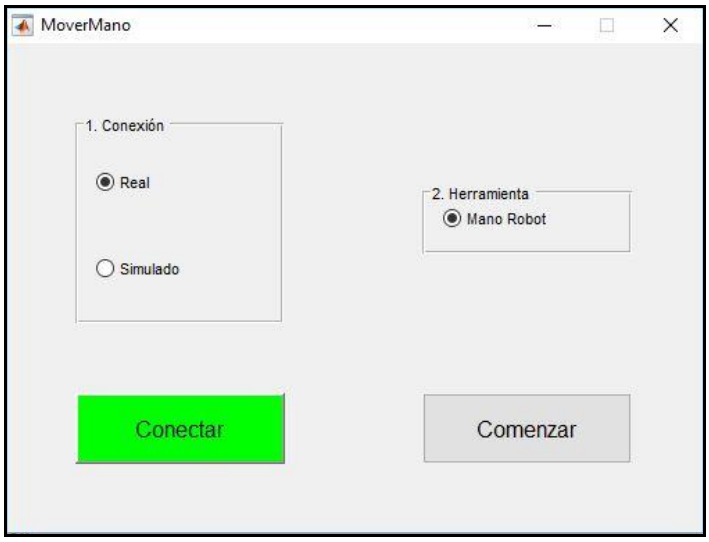

Figura 12.6: Interfaz "*MoverMano.m"*

Esta interfaz es muy sencilla de utilizar, lo primero que se debe hacer es seleccionar en la caja "1.conexión" la opción Simulado, pues este programa solo está preparado para funcionar en simulación. La segunda caja de botones (2.conexión) solo contiene la opción "Mano Robot", de manera que no hay que elegir nada. Esta segunda caja está pensada para añadir más herramientas en un futuro. Después de realizar estas comprobaciones, se debe pulsar el botón "Conectar" que aparece en verde en la interfaz *"MoverMano.m"* y en azul en la interfaz *"MoverBote.m"*.

Una vez hecho esto, se debería observar en RobotStudio como el robot simulado realiza una serie de movimientos hasta llegar a la posicion de reposo con la mano abierta. Cuando haya acabado esta secuencia, se pulsa el botón comenzar, que hará aparecer un cuadrado blanco en la pantalla (figura 12.7).

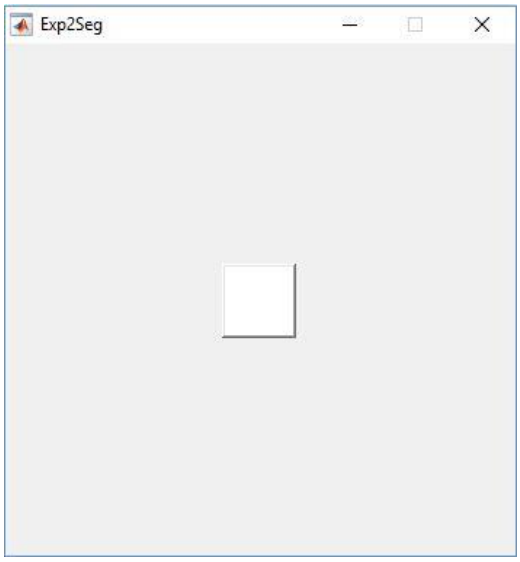

Figura 12.7: Punto blanco para facilitar al usuario entrar en estado de reposo.

Este cuadrado blanco permanecerá así durante 40 segundos, tiempo necesario para que la aplicación recoja una muestra de la señal lo suficientemente grande como para poder determinar si el usuario ha movido la mano o no. Las ventanas en las que se mueve la aplicación son de 20 segundos, por lo que el movimiento de una mano puede ser detectado hasta 20 segundos después de haber sido realizado. Por otra parte, si el sujeto está nervioso, habla, realiza movimientos oculares, etc.… puede activar la banda gamma aún sin mover la mano, produciendo un falso positivo.

El programa permanece en funcionamiento durante 5 minutos, pasado este tiempo el casco se desconecta y el programa deja de analizar la señal.

Si en algún momento al pulsar el botón comenzar Matlab devuelve un valor -2, esto quiere decir que el puerto serie que se desea utilizar está siendo usado por otro programa y se deben comprobar todas las aplicaciones que se tengan abiertas en ese momento.

V. Planos

13. Planos

En este capítulo se recoge todo el código escrito en este proyecto

Conectar.m

```
 % %CODIGO DE CONEXION CON BRAZO IRB120
global robot Simulado comenzar
 %Obtenemos los datos sobre el modo de conexión
comenzar=0;
modo=get(handles.simulado,'Value');
if modo==1
    %Conectar con el simulador
   robot=irb120('127.0.0.1',1024);
else
    %Conectar con el robot real
   robot=irb120('172.29.28.185',1024);
end

 %Conectar con el robot
robot.connect;
pause(2);
robot.TCPtool(1);    %Activar Ventosa
pause (2);
robot.TCPtool(0);    %Desactivar Ventosa
pause (2);
 %Envío de coordenadas para mover el brazo a posición intermedia
robot.TCProtandpos([180 0 180 300 0 300]);
comenzar=1;
```

Trocear.m

```
clear all
close all
global posicion;
global datos_raw;
global datos_atencion;
global datos_meditacion;
global trozos;
global trozos_raw;
global trozos_med;
global trozos_att;
global z;
global w;
w=1;
global raw_proc;
global med_proc;
global att_proc;
global raw_aceptados;
global raw_procesada;
global med_procesada;
global att_procesada;
raw_procesada=zeros(1,1000);
med_procesada=zeros(1,1000);
att_procesada=zeros(1,1000);
z=1;
 %Hasta aquí se inicializan todas las variables que se utilizaran en adelante
load ('Prueba10_50seg_B11.mat');
opcion=input('Introduce 1 para las pruebas o 2 para los experimentos:')
switch (opcion) %120 para el experimento basal y 200 para el experimento abriendo y
cerrando la mano
    case 1
        Tiempo=120;
    case 2
        Tiempo = 200;
end
```

```
datos_raw=data_raw;
datos_atencion=data_att;
datos_meditacion=data_med;

Nsamps = length(datos_raw);
Fs=Nsamps/Tiempo;
t = (1/Fs)*(1:Nsamps);           %Prepare time data for plot

 %Do Fourier Transform
F_datos_raw = abs(fft(datos_raw)); %Retain Magnitude
F_datos_med = abs(fft(datos_meditacion));
F_datos_att = abs(fft(datos_atencion));
f = Fs*(1:Nsamps)/Nsamps;    %Prepare freq data for plot

 %Plot Sound File in Time Domain
figure

 %Señal raw en el dominio temporal
subplot(3,1,1)
plot(t, datos_raw)
hold on
plot(t,trigger*700)
xlabel('Time (s)')
ylabel('Amplitude')
title('Signal raw in Time Domain')
axis([0 Tiempo -1000 1000])

 %datos de meditacion en el dominio temporal
subplot(3,1,2)
plot(t, datos_meditacion)
hold on
plot(t,trigger*700)
xlabel('Time (s)')
ylabel('Amplitude')
title('Signal meditaion in Time Domain')
axis([0 Tiempo 0 100])

 %datos de atención en el dominio temporal
subplot(3,1,3)
plot(t, datos_atencion)
hold on
plot(t,trigger*700)
xlabel('Time (s)')
ylabel('Amplitude')
title('Signal attention in Time Domain')
axis([0 Tiempo 0 100])

 %Plot Sound File in Frequency Domain
figure
subplot(3,1,1)
plot(f, F_datos_raw)
xlim([0 f(length(f))/2])
xlabel('Frequency (Hz)')
ylabel('Amplitude')
title('Frequency Response of Signal Raw')

subplot(3,1,2)
plot(f, F_datos_att)
xlim([0 f(length(f))/2])
xlabel('Frequency (Hz)')
ylabel('Amplitude')
title('Frequency Response of Signal meditation')

subplot(3,1,3)
plot(f, F_datos_med)
xlim([0 f(length(f))/2])
xlabel('Frequency (Hz)')
ylabel('Amplitude')
title('Frequency Response of Signal attention')

 %Fs=200;
raw=datos_raw;
med=datos_meditacion;
att=datos_atencion;
```

```
[filt] = ft_preproc_bandpassfilter(raw, Fs, [30 60]);  % el filtro devuelve la señal
filtrada en tiempo
 %[filt2] = ft_preproc_bandpassfilter(raw, Fs, [30 90]);
[filt_med] = ft_preproc_bandpassfilter(med, Fs, [30 60]);
[filt_att] = ft_preproc_bandpassfilter(att, Fs, [30 60]);
 %Nos quedamos con el filtro entre 30 y 60 hercios
 %Do Fourier Transform
F_filt = abs(fft(filt));             %Retain Magnitude
F_filt_med = abs(fft(filt_med));
F_filt_att = abs(fft(filt_att));
 %F_filt2 = abs(fft(filt2));       %Discard Half of Points
f = Fs*(1:Nsamps)/Nsamps;    %Prepare freq data for plot
 %Plot Sound File in Time Domain
figure
 %señal raw filtrada entre 30 y 60 Hz
subplot(3,1,1)
plot(t, filt)
hold on
plot(t,trigger*700)
xlabel('Time (s)')
ylabel('Amplitude')
title('Bandpass filter 30-60 Hz. Signal raw')
hold on
axis([0 Tiempo -300 300])
 %señal meditacion filtrada entre 30 y 60Hz
subplot(3,1,2)
plot(t, filt_med,'r')
hold on
plot(t,trigger*100)
xlabel('Time (s)')
ylabel('Amplitude')
title('Bandpass filter 30-60 Hz. Signal meditation')
hold on
axis([0 Tiempo -15 15])
 %señal atención filtrada entre 30 y 60 Hz
subplot(3,1,3)
plot(t, filt_att,'g')
hold on
plot(t,trigger*100)
xlabel('Time (s)')
ylabel('Amplitude')
title('Bandpass filter 30-60 Hz. Signal attention')
axis([0 Tiempo -15 15])
 %Plot Sound File in Frequency Domain
figure
subplot(3,1,1)
plot(f, F_filt)
xlim([0 f(length(f))/2])
xlabel('Frequency (Hz)')
ylabel('Amplitude')
title('Frequency Response Bandpass filter 30-60Hz.')
hold on
subplot(3,1,2)
plot(f, F_filt_med)
xlim([0 f(length(f))/2])
xlabel('Frequency (Hz)')
ylabel('Amplitude')
title('Frequency Response Bandpass filter 30-60Hz. Signal meditation')
subplot(3,1,3)
plot(f, F_filt_att)
xlim([0 f(length(f))/2])
xlabel('Frequency (Hz)')
ylabel('Amplitude')
title('Frequency Response Bandpass filter 30-60Hz. Signal attention')
 %posiciones del trigger
i=1;
j=1;
for i=1:1:length(trigger)
if(trigger(i) == 1)
posicion(j)=i;
j=j+1;
end
end
```

```
 %troceo

i=1;
j=1;
for j=1:1:length(posicion)-1    %normaliza a 1000
while(i<1001)
 %Esta matriz es la señal original raw troceada
trozos(j,i)= datos_raw(round((posicion(j)+(i*(posicion(j+1)-posicion(j))/1000))));
 %estos tres son señales filtradas
trozos_raw(j,i)= filt(round((posicion(j)+(i*(posicion(j+1)-posicion(j))/1000))));
trozos_med(j,i)= filt_med(round((posicion(j)+(i*(posicion(j+1)-posicion(j))/1000))));
trozos_att(j,i)= filt_att(round((posicion(j)+(i*(posicion(j+1)-posicion(j))/1000))));
i=i+1;
end
i=1;
end
longitud=length(trozos(:,1));
raw_aceptados=zeros(longitud,1000);
raw_proc=zeros(longitud,1000);
med_proc=zeros(longitud,1000);
att_proc=zeros(longitud,1000);  %estas variables las inicializo al final del código
                                %se utilizarán en el GUI y para reservar el
                                %espacio necesito saber cuántos trozos tengo
                                %antes de crearlas

 %testeo de muestras manual
 %k=1;
 %figure;
 %while(k<length(trozos(:,1))+1)
 %plot(trozos(k,:))
 %k=k+1
 %pause;
 %end

Procesado;
```

Procesado.m

```
 %Este programa nos permite decidir que muestras aceptamos y que muestras
 %rechazamos de la señal raw obtenida con el casco.

function varargout = Procesado(varargin)
 % PROCESADO MATLAB code for Procesado.fig
 %      PROCESADO, by itself, creates a new PROCESADO or raises the existing
 %      singleton*.
 %
 %      H = PROCESADO returns the handle to a new PROCESADO or the handle to
 %      the existing singleton*.
 %
 %      PROCESADO('CALLBACK',hObject,eventData,handles,...) calls the local
 %      function named CALLBACK in PROCESADO.M with the given input arguments.
 %
 %      PROCESADO('Property','Value',...) creates a new PROCESADO or raises the
 %      existing singleton*.  Starting from the left, property value pairs are
 %      applied to the GUI before Procesado_OpeningFcn gets called.  An
 %      unrecognized property name or invalid value makes property application
 %      stop.  All inputs are passed to Procesado_OpeningFcn via varargin.
 %
 %      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
 %      instance to run (singleton)".
 %
 % See also: GUIDE, GUIDATA, GUIHANDLES

 % Edit the above text to modify the response to help Procesado
```

```
 % Last Modified by GUIDE v2.5 16-Mar-2017 16:38:58

 % Begin initialization code - DO NOT EDIT

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Procesado_OpeningFcn, ...
                   'gui_OutputFcn',  @Procesado_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
 % End initialization code - DO NOT EDIT

 % --- Executes just before Procesado is made visible.
function Procesado_OpeningFcn(hObject, eventdata, handles, varargin)
 % This function has no output args, see OutputFcn.
 % hObject    handle to figure
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)
 % varargin   command line arguments to Procesado (see VARARGIN)

 % Choose default command line output for Procesado
handles.output = hObject;

 % Update handles structure
guidata(hObject, handles);
global z;
 % UIWAIT makes Procesado wait for user response (see UIRESUME)
 % uiwait(handles.figure1);

 % --- Outputs from this function are returned to the command line.
function varargout = Procesado_OutputFcn(hObject, eventdata, handles)
 % varargout  cell array for returning output args (see VARARGOUT);
 % hObject    handle to figure
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)

 % Get default command line output from handles structure
varargout{1} = handles.output;

 % --- Executes on button press in Aceptar_Muestra.
 function Aceptar_Muestra_Callback(hObject, eventdata, handles)
 % hObject    handle to Aceptar_Muestra (see GCBO)
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)
global z
global w
global trozos
global trozos_raw
global trozos_att
global trozos_med
global posicion
global raw_procesada
global raw_proc
global med_procesada
global med_proc
global att_procesada
global att_proc
global raw_aceptados
raw_proc(w,:)=trozos_raw(z,:);  %filtrados entre 30 y 60 Hz
med_proc(w,:)=trozos_med(z,:);  %filtrados entre 30 y 60 Hz
att_proc(w,:)=trozos_att(z,:);  %filtrados entre 30 y 60 Hz
raw_aceptados(w,:)= trozos(z,:);  %señal original
z=z+1;
w=w+1;
```

```
if (z < length(posicion))
    axes(handles.axes1);
     %plot(datos_raw(posicion(z+1):posicion(z+2)));
    plot(trozos(z,:));
    axes(handles.axes2);
    plot(trozos_raw(z,:));
end
 % --- Executes on button press in Rechazar_Muestra.
function Rechazar_Muestra_Callback(hObject, eventdata, handles)
 % hObject    handle to Rechazar_Muestra (see GCBO)
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)
global z
global w
global trozos
global trozos_raw
global posicion
global raw_aceptados
global raw_procesada
global raw_proc
global med_procesada
global med_proc
global att_procesada
global att_proc
z=z+1;
if(z >= length(posicion))
    raw_procesada=mean(raw_proc(1:w-1,:));
    med_procesada=mean(med_proc(1:w-1,:));
    att_procesada=mean(att_proc(1:w-1,:));  %cojo solo las filas en las que hay datos
para hacer la media
    save Prueba10_50seg_B11_procesado
    close all
end
if (z < length(posicion))
    axes(handles.axes1);
     %plot(datos_raw(posicion(z+1):posicion(z+2)));
    plot(trozos(z,:));
    axes(handles.axes2);
    plot(trozos_raw(z,:));
end
 % --- Executes on button press in Comenzar.
function Comenzar_Callback(hObject, eventdata, handles)
 % hObject    handle to Comenzar (see GCBO)
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)
 %global posicion;
 %global datos_raw;
global trozos;
global trozos_raw;
global z;
global w;
w=1;
z=1;
axes(handles.axes1);
 %plot(datos_raw(posicion(z+1):posicion(z+2)));

plot(trozos(z,:)); %señal original
axes(handles.axes2);
plot(trozos_raw(z,:)); %señal filtrada entre 30 y 60 Hz
 % --- Executes during object creation, after setting all properties.

function axes1_CreateFcn(hObject, eventdata, handles)
 % hObject    handle to axes1 (see GCBO)
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    empty - handles not created until after all CreateFcns called
 % Hint: place code in OpeningFcn to populate axes1
 % --- Executes during object creation, after setting all properties.

function axes2_CreateFcn(hObject, eventdata, handles)
 % hObject    handle to axes2 (see GCBO)
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    empty - handles not created until after all CreateFcns called
 % Hint: place code in OpeningFcn to populate axes2
```

Experimento1.m

```
function readRAW
 %run this function to connect and plot raw EEG data
 %make sure to change portnum1 to the appropriate COM port
clear all
close all
data = zeros(1,256);     %preallocate buffer
portnum1 = 2;    %COM Port #
comPortName1 = sprintf('\\\\.\\COM %d', portnum1);
 % Baud rate for use with TG_Connect() and TG_SetBaudrate().
TG_BAUD_57600 =      57600;
 % Data format for use with TG_Connect() and TG_SetDataFormat().
TG_STREAM_PACKETS =     0;
 % Data type that can be requested from TG_GetValue().
TG_DATA_POOR_SIGNAL =     1;
TG_DATA_ATTENTION =       2;
TG_DATA_MEDITATION =      3;
TG_DATA_RAW =             4;
TG_DATA_DELTA =           5;
TG_DATA_THETA =           6;
TG_DATA_ALPHA1 =          7;
TG_DATA_ALPHA2 =          8;
TG_DATA_BETA1 =           9;
TG_DATA_BETA2 =          10;
TG_DATA_GAMMA1 =         11;
TG_DATA_GAMMA2 =         12;
TG_DATA_BLINK_STRENGTH = 37;
 %load ThinkGear dll
loadlibrary('Thinkgear.dll');
fprintf('Thinkgear.dll loaded\n');
 %get dll version
dllVersion = calllib('Thinkgear', 'TG_GetDriverVersion');
fprintf('ThinkGear DLL version:  %d\n', dllVersion );
 % Get a connection ID handle to ThinkGear
connectionId1 = calllib('Thinkgear', 'TG_GetNewConnectionId');
if ( connectionId1 < 0 )
    error( sprintf( 'ERROR: TG_GetNewConnectionId() returned  %d.\n', connectionId1 )
);
end;
 % Set/open stream (raw bytes) log file for connection
errCode = calllib('Thinkgear', 'TG_SetStreamLog', connectionId1, 'streamLog.txt' );
if( errCode < 0 )
    error( sprintf( 'ERROR: TG_SetStreamLog() returned  %d.\n', errCode ) );
end;
 % Set/open data (ThinkGear values) log file for connection
errCode = calllib('Thinkgear', 'TG_SetDataLog', connectionId1, 'dataLog.txt' );
if( errCode < 0 )
    error( sprintf( 'ERROR: TG_SetDataLog() returned  %d.\n', errCode ) );
end;
 % Attempt to connect the connection ID handle to serial port "COM3"
errCode = calllib('Thinkgear', 'TG_Connect',
connectionId1,comPortName1,TG_BAUD_57600,TG_STREAM_PACKETS );
if ( errCode < 0 )
    error( sprintf( 'ERROR: TG_Connect() returned  %d.\n', errCode ) );
end
fprintf( 'Connected.  Reading Packets...\n' );
 %record data
j = 0;
i = 0;
h = 0;
t1=tic;
t2=t1;
global a;
a='w';
Exp2Seg;
while (toc(t1) < 200)    %loop for 200 seconds
if (calllib('Thinkgear','TG_ReadPackets',connectionId1,1) == 1) %if a packet wasread...
          if (calllib('Thinkgear','TG_GetValueStatus',connectionId1,TG_DATA_RAW) ~= 0)
%if RAW has been updated
              j = j + 1;
              h = h + 1;
              if(toc(t2)>2)  %cada 2 segundos
                    t2=tic;
                    trigger(h) = 1;
              else
```

```
trigger(h) = 0;
              end
              data_gamma1(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_GAMMA1);
              data_att(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_ATTENTION);
              data_med(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_MEDITATION);
              data_raw(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_RAW);
              data_alpha1(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_ALPHA1);
              data_theta(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_THETA);
              data_delta(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_DELTA);
              data_beta1(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_BETA1);
          end
    end
end
 %disconnect
    % % csvwrite('C:\UAH\4°\TFG\Matlab\data.csv',data);
     % %dlmwrite('C:\UAH\4°\TFG\Matlab\data.csv',data,' precision', ' %.6f');
     %To display in Command Window
    disp('Loop Completed')
     % %plot(data_gamma1)
calllib('Thinkgear', 'TG_FreeConnection', connectionId1 );

save basal
```

Experimento2.m

```
function readRAW
 %run this function to connect and plot raw EEG data
 %make sure to change portnum1 to the appropriate COM port
clear all
close all
data = zeros(1,256);     %preallocate buffer
portnum1 = 2;    %COM Port #
comPortName1 = sprintf('\\\\.\\COM %d', portnum1);
 % Baud rate for use with TG_Connect() and TG_SetBaudrate().
TG_BAUD_57600 =      57600;
 % Data format for use with TG_Connect() and TG_SetDataFormat().
TG_STREAM_PACKETS =     0;
 % Data type that can be requested from TG_GetValue().
TG_DATA_POOR_SIGNAL =     1;
TG_DATA_ATTENTION =       2;
TG_DATA_MEDITATION =      3;
TG_DATA_RAW =             4;
TG_DATA_DELTA =           5;
TG_DATA_THETA =           6;
TG_DATA_ALPHA1 =          7;
TG_DATA_ALPHA2 =          8;
TG_DATA_BETA1 =           9;
TG_DATA_BETA2 =          10;
TG_DATA_GAMMA1 =         11;
TG_DATA_GAMMA2 =         12;
TG_DATA_BLINK_STRENGTH = 37;
 %load thinkgear dll
loadlibrary('Thinkgear.dll');
fprintf('Thinkgear.dll loaded\n');
 %get dll version
dllVersion = calllib('Thinkgear', 'TG_GetDriverVersion');
fprintf('ThinkGear DLL version:  %d\n', dllVersion );
% Get a connection ID handle to ThinkGear
connectionId1 = calllib('Thinkgear', 'TG_GetNewConnectionId');
if ( connectionId1 < 0 )
    error( sprintf( 'ERROR: TG_GetNewConnectionId() returned  %d.\n', connectionId1 )
);
end;
```

```
 % Set/open stream (raw bytes) log file for connection
errCode = calllib('Thinkgear', 'TG_SetStreamLog', connectionId1, 'streamLog.txt' );
if( errCode < 0 )
    error( sprintf( 'ERROR: TG_SetStreamLog() returned  %d.\n', errCode ) );
end;
 % Set/open data (ThinkGear values) log file for connection
errCode = calllib('Thinkgear', 'TG_SetDataLog', connectionId1, 'dataLog.txt' );
if( errCode < 0 )
    error( sprintf( 'ERROR: TG_SetDataLog() returned  %d.\n', errCode ) );
end;
 % Attempt to connect the connection ID handle to serial port "COM3"
errCode = calllib('Thinkgear', 'TG_Connect',
connectionId1,comPortName1,TG_BAUD_57600,TG_STREAM_PACKETS );
if ( errCode < 0 )
    error( sprintf( 'ERROR: TG_Connect() returned  %d.\n', errCode ) );
end
fprintf( 'Connected.  Reading Packets...\n' );
 % %
 %record data
j = 0;
i = 0;
h = 0;
t1=tic;
t2=t1;
global a;
a='w';
Exp2Seg;
while (toc(t1) < 200)    %loop for 200 seconds
    if (calllib('Thinkgear','TG_ReadPackets',connectionId1,1) == 1)    %if a packet was
read...
        if (calllib('Thinkgear','TG_GetValueStatus',connectionId1,TG_DATA_RAW) ~= 0)
%if RAW has been updated
                h = h + 1;
              if(toc(t2)>2)  %cada 2 segundos parpadea en rojo el cuadrado en la
pantalla
                 t2=tic;
                 a='r';
                 Exp2Seg;
                 a='w';
                 Exp2Seg;
                 trigger(h)= 1;
              else
                 trigger(h)=0;
              end
              j = j + 1;
              data_gamma1(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_GAMMA1);
              data_att(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_ATTENTION);
              data_med(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_MEDITATION);
              data_raw(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_RAW);
              data_alpha1(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_ALPHA1);
              data_theta(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_THETA);
              data_delta(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_DELTA);
              data_beta1(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_BETA1);
          end
    end
end
 %disconnect
    % % csvwrite('C:\UAH\4°\TFG\Matlab\data.csv',data);
     % %dlmwrite('C:\UAH\4°\TFG\Matlab\data.csv',data,'precision', ' %.6f');
     %To display in Command Window
    disp('Loop Completed')
     % %plot(data_gamma1)
calllib('Thinkgear', 'TG_FreeConnection', connectionId1 );

save activo
```

Prueba120seg.m

```
function readRAW
 %run this function to connect and plot raw EEG data
 %make sure to change portnum1 to the appropriate COM port
clear all
close all
data = zeros(1,256);     %preallocate buffer
portnum1 = 2;    %COM Port #
comPortName1 = sprintf('\\\\.\\COM %d', portnum1);
 % Baud rate for use with TG_Connect() and TG_SetBaudrate().
TG_BAUD_57600 =      57600;
 % Data format for use with TG_Connect() and TG_SetDataFormat().
TG_STREAM_PACKETS =     0;
 % Data type that can be requested from TG_GetValue().
TG_DATA_POOR_SIGNAL =     1;
TG_DATA_ATTENTION =       2;
TG_DATA_MEDITATION =      3;
TG_DATA_RAW =             4;
TG_DATA_DELTA =           5;
TG_DATA_THETA =           6;
TG_DATA_ALPHA1 =          7;
TG_DATA_ALPHA2 =          8;
TG_DATA_BETA1 =           9;
TG_DATA_BETA2 =          10;
TG_DATA_GAMMA1 =         11;
TG_DATA_GAMMA2 =         12;
TG_DATA_BLINK_STRENGTH = 37;
 %load thinkgear dll
loadlibrary('Thinkgear.dll');
fprintf('Thinkgear.dll loaded\n');
 %get dll version
dllVersion = calllib('Thinkgear', 'TG_GetDriverVersion');
fprintf('ThinkGear DLL version:  %d\n', dllVersion );
 % %
 % Get a connection ID handle to ThinkGear
connectionId1 = calllib('Thinkgear', 'TG_GetNewConnectionId');
if ( connectionId1 < 0 )
    error( sprintf( 'ERROR: TG_GetNewConnectionId() returned  %d.\n', connectionId1 )
);
end;
 % Set/open stream (raw bytes) log file for connection
errCode = calllib('Thinkgear', 'TG_SetStreamLog', connectionId1, 'streamLog.txt' );
if( errCode < 0 )
    error( sprintf( 'ERROR: TG_SetStreamLog() returned  %d.\n', errCode ) );
end;
 % Set/open data (ThinkGear values) log file for connection
errCode = calllib('Thinkgear', 'TG_SetDataLog', connectionId1, 'dataLog.txt' );
if( errCode < 0 )
    error( sprintf( 'ERROR: TG_SetDataLog() returned  %d.\n', errCode ) );
end;
 % Attempt to connect the connection ID handle to serial port "COM3"
errCode = calllib('Thinkgear', 'TG_Connect',
connectionId1,comPortName1,TG_BAUD_57600,TG_STREAM_PACKETS );
if ( errCode < 0 )
    error( sprintf( 'ERROR: TG_Connect() returned  %d.\n', errCode ) );
end
fprintf( 'Connected.  Reading Packets...\n' );
 % %
 %record data
j = 0;
i = 0;
h = 0;
t1=tic;
t2=t1;
intentos=10; %modificar esta línea para cambiar el número de veces que se va a realizar
actividad motriz en la mano
global a;
a='w';
Exp2Seg;
while (toc(t1) < 120)    %bucle de 120 segundos
    if (calllib('Thinkgear','TG_ReadPackets',connectionId1,1) == 1)    %if a packet was
read...
          if (calllib('Thinkgear','TG_GetValueStatus',connectionId1,TG_DATA_RAW) ~= 0)
%if RAW has been updated
```

```
h = h + 1;
                if(toc(t1)<50) %modificar esta línea para cambiar el segundo de
inicio
                if(toc(t2)>2)  %cada 2 segundos
                  t2=tic;
                  trigger(h) = 1;
                else
                  trigger(h) = 0;
                  end
                end
               if(toc(t1)>50) %modificar esta línea para cambiar el segundo de inicio
             if(toc(t2)>2 && intentos~=0)  %cada 2 segundos parpadea en rojo el
cuadrado en la pantalla
                t2=tic;
                a='r';
                Exp2Seg;
                a='w';
                Exp2Seg;
                trigger(h)= 1;
                intentos=intentos-1;
             else if (toc(t2)>2 && intentos==0)
                    t2=tic;
                    trigger(h)=1;
                 else
                trigger(h)=0;
             end
             end
                end
             j = j + 1;
             data_gamma1(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_GAMMA1);
             data_att(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_ATTENTION);
             data_med(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_MEDITATION);
             data_raw(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_RAW);
             data_alpha1(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_ALPHA1);
             data_theta(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_THETA);
             data_delta(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_DELTA);
             data_beta1(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_BETA1);
          end
    end
end
 %disconnect
    % % csvwrite('C:\UAH\4°\TFG\Matlab\data.csv',data);
     % %dlmwrite('C:\UAH\4°\TFG\Matlab\data.csv',data,'precision', ' %.6f');
     %To display in Command Window
    disp('Loop Completed')
     % %plot(data_gamma1)
calllib('Thinkgear', 'TG_FreeConnection', connectionId1 );
```

MoverMano.m

```
function varargout = MoverMano(varargin)
 % MOVERMANO MATLAB code for MoverMano.fig
 %      MOVERMANO, by itself, creates a new MOVERMANO or raises the existing
 %      singleton*.
 %
 %      H = MOVERMANO returns the handle to a new MOVERMANO or the handle to
 %      the existing singleton*.
 %
 %      MOVERMANO('CALLBACK',hObject,eventData,handles,...) calls the local
 %      function named CALLBACK in MOVERMANO.M with the given input arguments.
 %
 %      MOVERMANO('Property','Value',...) creates a new MOVERMANO or raises the
 %      existing singleton*.  Starting from the left, property value pairs are
 %      applied to the GUI before MoverMano_OpeningFcn gets called.  An
```

```
unrecognized property name or invalid value makes property application
 %      stop.  All inputs are passed to MoverMano_OpeningFcn via varargin.
 %
 %      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
 %      instance to run (singleton)".
 %
 % See also: GUIDE, GUIDATA, GUIHANDLES

 % Edit the above text to modify the response to help MoverMano

 % Last Modified by GUIDE v2.5 20-Jul-2017 21:04:37

 % Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MoverMano_OpeningFcn, ...
                   'gui_OutputFcn',  @MoverMano_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
 % End initialization code - DO NOT EDIT
 % --- Executes just before MoverMano is made visible.
function MoverMano_OpeningFcn(hObject, eventdata, handles, varargin)
 % This function has no output args, see OutputFcn.
 % hObject    handle to figure
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)
 % varargin   command line arguments to MoverMano (see VARARGIN)
 % Choose default command line output for MoverMano
handles.output = hObject;
 % Update handles structure
guidata(hObject, handles);
 % UIWAIT makes MoverMano wait for user response (see UIRESUME)
 % uiwait(handles.figure1);

 % --- Outputs from this function are returned to the command line.
function varargout = MoverMano_OutputFcn(hObject, eventdata, handles)
 % varargout  cell array for returning output args (see VARARGOUT);
 % hObject    handle to figure
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)
 % Get default command line output from handles structure
varargout{1} = handles.output;
 % --- Executes on button press in Conectar.

function Conectar_Callback(hObject, eventdata, handles)
 % hObject    handle to Conectar (see GCBO)
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)
 % % %Control de apariencia de botones en interfaz
set(handles.Conectar,'Enable','off');
set(handles.Real,'Enable','off');
set(handles.Simulado,'Enable','off');
set(handles.ManoRobot,'Enable','off');
Conectar;  %Conectar con robot
 % --- Executes on button press in Comenzar.
function Comenzar_Callback(hObject, eventdata, handles)
 % hObject    handle to Comenzar (see GCBO)
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)
 %run this function to connect and plot raw EEG data
 %make sure to change portnum1 to the appropriate COM port
portnum1 = 2;    %COM Port #
comPortName1 = sprintf('\\\\.\\COM %d', portnum1);
 % Baud rate for use with TG_Connect() and TG_SetBaudrate().
TG_BAUD_57600 =      57600;
 % Data format for use with TG_Connect() and TG_SetDataFormat().
```

```
TG_STREAM_PACKETS =     0;
 % Data type that can be requested from TG_GetValue().
TG_DATA_POOR_SIGNAL =     1;
TG_DATA_ATTENTION =       2;
TG_DATA_MEDITATION =      3;
TG_DATA_RAW =             4;
TG_DATA_DELTA =           5;
TG_DATA_THETA =           6;
TG_DATA_ALPHA1 =          7;
TG_DATA_ALPHA2 =          8;
TG_DATA_BETA1 =           9;
TG_DATA_BETA2 =          10;
TG_DATA_GAMMA1 =         11;
TG_DATA_GAMMA2 =         12;
TG_DATA_BLINK_STRENGTH = 37;
 %load thinkgear dll
loadlibrary('Thinkgear.dll');
fprintf('Thinkgear.dll loaded\n');
 %get dll version
dllVersion = calllib('Thinkgear', 'TG_GetDriverVersion');
fprintf('ThinkGear DLL version:  %d\n', dllVersion );
 % %
 % Get a connection ID handle to ThinkGear
connectionId1 = calllib('Thinkgear', 'TG_GetNewConnectionId');
if ( connectionId1 < 0 )
    error( sprintf( 'ERROR: TG_GetNewConnectionId() returned  %d.\n', connectionId1 )
);
end;
 % Set/open stream (raw bytes) log file for connection
errCode = calllib('Thinkgear', 'TG_SetStreamLog', connectionId1, 'streamLog.txt' );
if( errCode < 0 )
    error( sprintf( 'ERROR: TG_SetStreamLog() returned  %d.\n', errCode ) );
end;
 % Set/open data (ThinkGear values) log file for connection
errCode = calllib('Thinkgear', 'TG_SetDataLog', connectionId1, 'dataLog.txt' );
if( errCode < 0 )
    error( sprintf( 'ERROR: TG_SetDataLog() returned  %d.\n', errCode ) );
end;
 % Attempt to connect the connection ID handle to serial port "COM2"
errCode = calllib('Thinkgear', 'TG_Connect',
connectionId1,comPortName1,TG_BAUD_57600,TG_STREAM_PACKETS );
if ( errCode < 0 )
    error( sprintf( 'ERROR: TG_Connect() returned  %d.\n', errCode ) );
end
fprintf( 'Connected.  Reading Packets...\n' );
 % %
 %record data
global j;
j = 0;
i = 0;
h = 0;
k = 0;
p = 0;
r = 0;
reverseMov=1;
cerrado=0;
patronTR=sin(0:pi/49:pi);
global a;
a='w';
Exp2Seg;
t2=tic;
while (toc(t2)<300)    %Bucle 5 minutos

    if (calllib('Thinkgear','TG_ReadPackets',connectionId1,1) == 1)    %if a packet was
read...
if (calllib('Thinkgear','TG_GetValueStatus',connectionId1,TG_DATA_RAW) ~= 0)    %if RAW
has been updated
              if(j>=11000)
                 data_raw=data_raw(2:end);
  data_raw(end+1) = calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_RAW);
                 i=i+1;
              end
              if(j<11000)
              j=j+1;
 data_raw(j) = calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_RAW);
                if(j==11000)
                    i=100;
                end
```

```
              % % %PROCESAMIENTO DE LA SEÑAL % %
            if(i==100)
                p=p+1;
                t1=tic;
                i=0;
                data_raw_filt=ft_preproc_bandpassfilter(data_raw(1,1:10000),500,[30
60]);
                data_raw_filt=ft_preproc_bandstopfilter(data_raw_filt,500,[48 52]);
                data_raw_frec=abs(fft(data_raw_filt));
                data_raw_frec_envol=ft_preproc_lowpassfilter(data_raw_frec,500,10);
                data_raw_frec_envol_mean=mean(data_raw_frec_envol(600:1200));
                Vector_mov(p)=data_raw_frec_envol_mean;
                if(length(Vector_mov)>=100)
                   if(h==0)
                        a='r';
                        Exp2Seg;
                        a='w';
                        Exp2Seg;
                        h=1;
                   end
                     %Vector_mov_mean=mean(Vector_mov(1+k:100+k));
                     %Vector_mov_filt=Vector_mov(1+k:100+k)-Vector_mov_mean;
Vector_mov_filt=ft_preproc_lowpassfilter(Vector_mov(1:100+k),500,10);
Vector_mov_filt=ft_preproc_highpassfilter(Vector_mov_filt(1:75+k),500,1);
                    convolucion=conv(patronTR,Vector_mov_filt(1+k:50+k),'same');
                    plot(convolucion)
                    k=k+1;
                if(find(convolucion>1000))
                    a='r';
                    Exp2Seg;
                     if(cerrado==0)
                     Mano_Cerrar
                     cerrado=1;
                     end
                else
                    a='w';
                    Exp2Seg;
                     if(cerrado==1)
                        Mano_Abrir
                         cerrado=0;
                     end
                end
                    end
                toc(t1)
            end
          end
    end
end
calllib('Thinkgear', 'TG_FreeConnection', connectionId1 );

 % --- Executes on button press in Simulado.
function Simulado_Callback(hObject, eventdata, handles)
 % hObject    handle to Simulado (see GCBO)
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)
 % Hint: get(hObject,'Value') returns toggle state of Simulado
 % --- Executes on button press in Real.
function Real_Callback(hObject, eventdata, handles)
 % hObject    handle to Real (see GCBO)
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)
 % Hint: get(hObject,'Value') returns toggle state of Real
 % --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
 % hObject    handle to figure1 (see GCBO)
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)
global j

if(j>1)
            calllib('Thinkgear', 'TG_FreeConnection', connectionId1 );
end
 % Hint: delete(hObject) closes the figure
delete(hObject);
```

MoverBote.m

```
function varargout = MoverBote(varargin)
% MOVERBOTE MATLAB code for MoverBote.fig
%      MOVERBOTE, by itself, creates a new MOVERBOTE or raises the existing
%      singleton*.
%
%      H = MOVERBOTE returns the handle to a new MOVERBOTE or the handle to
%      the existing singleton*.
%
%      MOVERBOTE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MOVERBOTE.M with the given input arguments.
%
%      MOVERBOTE('Property','Value',...) creates a new MOVERBOTE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before MoverBote_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to MoverBote_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help MoverBote

% Last Modified by GUIDE v2.5 20-Jul-2017 21:23:01

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MoverBote_OpeningFcn, ...
                   'gui_OutputFcn',  @MoverBote_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before MoverBote is made visible.
function MoverBote_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MoverBote (see VARARGIN)

% Choose default command line output for MoverBote
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes MoverBote wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = MoverBote_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
 % --- Executes on button press in Conectar.
function Conectar_Callback(hObject, eventdata, handles)
 % hObject    handle to Conectar (see GCBO)
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)

 % % %Control de apariencia de botones en interfaz
set(handles.Conectar,'Enable','off');
set(handles.Real,'Enable','off');
set(handles.Simulado,'Enable','off');
set(handles.ManoRobot,'Enable','off');
Conectar;  %Conectar con robot

 % --- Executes on button press in Comenzar.
function Comenzar_Callback(hObject, eventdata, handles)
 % hObject    handle to Comenzar (see GCBO)
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)
 %run this function to connect and plot raw EEG data
 %make sure to change portnum1 to the appropriate COM port

portnum1 = 2;    %COM Port #
comPortName1 = sprintf('\\\\.\\COM %d', portnum1);
 % Baud rate for use with TG_Connect() and TG_SetBaudrate().
TG_BAUD_57600 =      57600;
 % Data format for use with TG_Connect() and TG_SetDataFormat().
TG_STREAM_PACKETS =     0;
 % Data type that can be requested from TG_GetValue().
TG_DATA_POOR_SIGNAL =     1;
TG_DATA_ATTENTION =       2;
TG_DATA_MEDITATION =      3;
TG_DATA_RAW =             4;
TG_DATA_DELTA =           5;
TG_DATA_THETA =           6;
TG_DATA_ALPHA1 =          7;
TG_DATA_ALPHA2 =          8;
TG_DATA_BETA1 =           9;
TG_DATA_BETA2 =          10;
TG_DATA_GAMMA1 =         11;
TG_DATA_GAMMA2 =         12;
TG_DATA_BLINK_STRENGTH = 37;
 %load thinkgear dll
loadlibrary('Thinkgear.dll');
fprintf('Thinkgear.dll loaded\n');
 %get dll version
dllVersion = calllib('Thinkgear', 'TG_GetDriverVersion');
fprintf('ThinkGear DLL version:  %d\n', dllVersion );
 % %
 % Get a connection ID handle to ThinkGear
connectionId1 = calllib('Thinkgear', 'TG_GetNewConnectionId');
if ( connectionId1 < 0 )
    error( sprintf( 'ERROR: TG_GetNewConnectionId() returned  %d.\n', connectionId1 )
);
end;
 % Set/open stream (raw bytes) log file for connection
errCode = calllib('Thinkgear', 'TG_SetStreamLog', connectionId1, 'streamLog.txt' );
if( errCode < 0 )
    error( sprintf( 'ERROR: TG_SetStreamLog() returned  %d.\n', errCode ) );
end;
 % Set/open data (ThinkGear values) log file for connection
errCode = calllib('Thinkgear', 'TG_SetDataLog', connectionId1, 'dataLog.txt' );
if( errCode < 0 )
    error( sprintf( 'ERROR: TG_SetDataLog() returned  %d.\n', errCode ) );
end;
 % Attempt to connect the connection ID handle to serial port "COM2"
errCode = calllib('Thinkgear', 'TG_Connect',
connectionId1,comPortName1,TG_BAUD_57600,TG_STREAM_PACKETS );
if ( errCode < 0 )
error( sprintf( 'ERROR: TG_Connect() returned  %d.\n', errCode ) );
end
fprintf( 'Connected.  Reading Packets...\n' );
 % %
 %record data
j = 0;
i = 0;
h = 0;
```

```
k = 0;
p = 0;
r = 0;
reverseMov=1;
cerrado=0;
patronTR=sin(0:pi/49:pi);
global a;
a='w';
Exp2Seg;
t2=tic;
while (toc(t2)<300)    %Bucle 5 minutos

    if (calllib('Thinkgear','TG_ReadPackets',connectionId1,1) == 1)    %if a packet was
read...
          if (calllib('Thinkgear','TG_GetValueStatus',connectionId1,TG_DATA_RAW) ~= 0)
%if RAW has been updated

              if(j>=11000)
                 data_raw=data_raw(2:end);
                 data_raw(end+1) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_RAW);
                 i=i+1;
              end

              if(j<11000)
              j=j+1;
              data_raw(j) =
calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_RAW);
                if(j==11000)
                    i=100;
                end
              end


               % % %PROCESAMIENTO DE LA SEÑAL % %
            if(i==100)
                p=p+1;
                t1=tic;
                i=0;

                data_raw_filt=ft_preproc_bandpassfilter(data_raw(1,1:10000),500,[30
60]);
                data_raw_filt=ft_preproc_bandstopfilter(data_raw_filt,500,[48 52]);
                data_raw_frec=abs(fft(data_raw_filt));
                data_raw_frec_envol=ft_preproc_lowpassfilter(data_raw_frec,500,10);
                data_raw_frec_envol_mean=mean(data_raw_frec_envol(600:1200));


                Vector_mov(p)=data_raw_frec_envol_mean;


                if(length(Vector_mov)>=100)

                   if(h==0)
                        a='r';
                        Exp2Seg;
                        a='w';
                        Exp2Seg;
                        h=1;
                   end

Vector_mov_filt=ft_preproc_lowpassfilter(Vector_mov(1:100+k),500,10);
Vector_mov_filt=ft_preproc_highpassfilter(Vector_mov_filt(1:75+k),500,1);
                    convolucion=conv(patronTR,Vector_mov_filt(1+k:50+k),'same');
                    plot(convolucion)
                    k=k+1;
                if(find(convolucion>1000))
                    a='r';
                    Exp2Seg;
                     if(reverseMov==1)
                        reverseMov=0;
robot.TCPtool(1);
                        pause(1);
                        robot.TCProtandpos([180 0 180 500 -250 150]);
                        robot.TCPtool(0);
                        pause(1);
```

```
                        robot.TCProtandpos([180 0 180 400 0 200]);
                        robot.TCProtandpos([180 0 180 500 250 150]);
                        robot.TCPtool(1);
                        pause(1);
                        robot.TCProtandpos([180 0 180 400 0 200]);

                    end

                else

                    a='w';
                    Exp2Seg;


                    if(reverseMov==0)
                            reverseMove=1;

                        robot.TCPtool(1);
                        pause(1);
                        robot.TCProtandpos([180 0 180 500 250 150]);
                        robot.TCPtool(0);
                        pause(1);

                        robot.TCProtandpos([180 0 180 400 0 200]);
                        robot.TCProtandpos([180 0 180 500 -250 150]);
                        robot.TCPtool(1);
                        pause(1);
                        robot.TCProtandpos([180 0 180 400 0 200]);

                    end
                end
                end
                toc(t1)
            end
        end
    end
end
calllib('Thinkgear', 'TG_FreeConnection', connectionId1 );
 % --- Executes on button press in Real.
function Real_Callback(hObject, eventdata, handles)
 % hObject    handle to Real (see GCBO)
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)

 % Hint: get(hObject,'Value') returns toggle state of Real


 % --- Executes on button press in Simulado.
function Simulado_Callback(hObject, eventdata, handles)
 % hObject    handle to Simulado (see GCBO)
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)

 % Hint: get(hObject,'Value') returns toggle state of Simulado

 % --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
 % hObject    handle to figure1 (see GCBO)
 % eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)
global j

if(j>1)
            calllib('Thinkgear', 'TG_FreeConnection', connectionId1 );
end
 % Hint: delete(hObject) closes the figure
delete(hObject);
```

Dentro de estos programas, todo el código relacionado con la conexión del casco y las librerias ThinkGear ha sido facilitado por la página Mathworks.

VI.Pliego de condiciones

14. Pliego de condiciones

En este capítulo se detallan las herramientas utilizadas en el proyecto, con la finalidad de poder elaborar posteriormente el presupuesto final del proyecto.

14.1. Hardware

1. Portátil HP Envy 15
    - Procesador: Intel® Core™ i7-5500U
    - RAM: 8 GB.
    - Tarjeta gráfica: NVIDIA GeForce GTX.

2. Robot industrial ABB-IRB120
    - Peso: 25 kg.
    - Altura: 580 mm.
    - Carga soportada: 3 kg (4kg con muñeca vertical).
    - Controlador IRC5 Compact. Tarjeta DeviceNet (Lean), con 16 entradas y 16 salidas digitales (0-1) de 24V.

3. Casco Mindset de Neurosky
    - Peso: 90 gramos
    - Medidas: alto 225mm x ancho 155mm x profundidad 110 mm (con sensor desplegado)
    - Consumo de potencia: 80 mA (conectado y transmitiendo)
    - UART (serie): VCC, GND, TX, RX
    - Tasa UART: 57600 baudios

4. Periféricos
    - Ratón GXT25 trust.

14.2. Software

1. Sistema operativo
    - Windows 10 © Microsoft Corporation.
2. Software de desarrollo
    - MATLAB R2015b (32-bits)
    - ABB RobotStudio versión 5.61
    - RobotWare versión 5.15
3. Procesador de textos y edición de imágenes
    - Microsoft Office 365 ProPlus 2016
4. Software de edición de imágenes
    - Microsoft Paint 2010

-

VII. Presupuesto

15. Presupuesto

Este capítulo proporciona la información detallada de los costes teóricos, entre los que se incluyen los **costes materiales** y las **tasas profesionales**.

15.1. Costes materiales

En la tabla 14.1 se han desglosado todos los costes materiales de este proyecto.

Concepto		Cantidad	Precio unidad (€)	Precio final (€)
Hardware	HP Envy 15	1	900	900
	IRB120	1	10900	10900
	Periféricos	1	30	30
	NeuroSky Mindset	1	170	170
Software	Licencia Matlab R2015b	1	500	500
	RobotStudio 5.61	1	1100	1100
	Microsoft Office 365 ProPlus 2016	6	12.90 €/mes	77.40
TOTAL				13677.40 €

Tabla 14.1. Costes materiales sin IVA

15.2. Costes profesionales

En la tabla 14.2 se muestra una estimación del coste profesional asociado al proyecto

Concepto	Horas	Precio por hora (€)	Precio final (€)
Ingeniero Industrial	300	35	10500
TOTAL			10500 €

Tabla 14.2. Costes profesionales sin IVA

15.3. Costes totales

Los costes totales del proyecto se recogen en la tabla 11.3. Estos se obtienen sumando los costes materiales y profesionales y aplicando el IVA.

Concepto	Precio final
Coste material	13677.40 €
Coste profesional	10500 €
SUBTOTAL	24177.40 €
IVA (21 %)	5077.25 €
TOTAL	29254.65 €

Tabla 14.3. Costes totales con IVA

VIII. Bibliografía

Bibliografía

[1]Rafael Barea Navarro "Electroencefalografía" http://www.bioingenieria.edu.ar/academica/catedras/bioingenieria2/archivos/apuntes/tema %205 %20- %20electroencefalografia.pdf

[2] Amo, C., Ortiz del Castillo, M., Barea, R., de Santiago, L., Martínez-Arribas, A., Amo-López, P., & Boquete, L. (2016) "*Induced Gamma-Band Activity During Voluntary Movement: EEG Analysis for Clinical Purposes*". Human kinetics, Volumen 20, Tema 4, pág. 409-428.

[3] https://en.wikipedia.org/wiki/InMoov Última consulta 11/07/2017

[4] https://www.muyinteresante.es/tecnologia/articulo/inmoov-un-robot-de-codigo-abierto-impreso-en-3d-301448376039 Última consulta 11/07/2017

[5] http://www.edumakers.es/index.php/construye/robot-inmoov Última consulta 11/07/2017

[6] https://descubrearduino.com/inmoov-el-robot-impreso-en-3d-de-codigo-abierto-y-con-arduino/ Última consulta 11/07/2017

[7] http://be.uji.es/proyectos/proyecto_devalhand.htm Última consulta 11/07/2017

[8] https://riuma.uma.es/xmlui/bitstream/handle/10630/7879/articulo_CEA_Ron_Angevin_2014.pdf?sequence=1 Última consulta 11/07/2017

[9] https://hipertextual.com/2016/07/writing-personal-statement Última consulta 11/07/2017

[10] https://www.xataka.com/robotica-e-ia/luke-el-increible-brazo-bionico-del-creador-de-segway-finalmente-saldra-a-la-venta-este-2016 Última consulta 11/07/2017

[11] Bedoya-Rojas, A., Giraldo-Leiva, J., Durley Torres-Pardo, I. & Albero Becerra-Botero, M. "*Interfaz cerebro computador basado en señales EEG para el control de movimiento de una prótesis de mano usando ANFIS*"

[12] http://www.3dnatives.com/es/youbionic-protesis-bionica-06012017/ Última consulta 12/07/2017

[13] http://www.youbionic.com/#youbionic-hand Última consulta 12/07/2017

[14] http://www.dinero.com/internacional/articulo/protesis-roboticas-e-implantes-cerebrales-para-personas-con-discapacidad-fisica/240434 Última consulta 12/07/2017

[15] https://psicologiaymente.net/neurociencias/partes-cerebro-humano# Última consulta 12/07/2017

[16] https://psicologiaymente.net/neurociencias/mielina Última consulta 12/07/2017

[17] http://www.onmeda.es/anatomia/anatomia_cerebro-estructura-del-cerebro-1478-2.html Ultima consulta 12/07/2017

[18] García Martín, L. "*Control del robot IRB120 mediante el casco de electroencefalografía Neurosky Mindwave*".

[19]http://www.muyinteresante.com.mx/preguntas-y-respuestas/15/09/11/diferencias-cerebro-izq-der/ Última consulta 12/07/2017

[20]https://psicologiaymente.net/neurociencias/lobulo-frontal-cerebro Última consulta 12/07/2017

[21]https://psicologiaymente.net/neurociencias/lobulo-parietal Última consulta 12/07/2017

[22]https://psicologiaymente.net/neurociencias/lobulo-occipital Última consulta 12/07/2017

[23]https://psicologiaymente.net/neurociencias/lobulo-temporal Última consulta 12/07/2017

[24]http://webspace.ship.edu/cgboer/genesp/corteza_cerebral.html Última consulta 12/07/2017

[25]http://inmoov.fr/inmoov-stl-parts-viewer/ Última consulta 12/07/2017

[26]http://comofuncionaque.com/funciones-de-la-neurona/ Última consulta 12/07/2017

[27] https://neuropediatra.org/2014/06/04/sinapsis-neuronal/ Última consulta 12/07/2017

[28] https://es.wikipedia.org/wiki/Potencial_de_membrana Última consulta 13/07/2017

[29] https://es.wikipedia.org/wiki/Potencial_de_acci %C3 %B3n Última consulta 13/07/2017

[30] https://psicologiaymente.net/neurociencias/tipos-ondas-cerebrales Última consulta 13/07/2017

[31] Tallon-Baudry C. & Bertrand O. (1999) "Oscillatory gamma activity in humans and its role in object representation". Trends in cognitive science, volumen 3, tema 4, pág. 151-162.

[32] http://www.clinicasantamaria.cl/noticias/noticia_muestra.asp?new=854 Última consulta 14/07/2017

[33] https://www.clinicalascondes.cl/BLOG/Listado/Neurologia-Adultos/cuatro-tipos-de-electroencefalograma Última consulta 14/07/2017

[34] https://lacofa.fundaciontelefonica.com/2008/12/15/introduccion-a-los-sistemas-brain-computer-interface/ Última consulta 14/07/2017

[35] https://es.wikipedia.org/wiki/ABB Última consulta 18/07/2017

[36] http://new.abb.com/products/robotics/es/robots-industriales/irb-120 Última consulta 18/07/2017

[37] http://new.abb.com/es/abb-in-spain/quienes-somos Última consulta 18/07/2017

[38] http://new.abb.com/products/robotics/es/software-para-aplicaciones/software-alimentacion-de-maquinas/robotware-machine-tending Última consulta 18/07/2017

[39] http://egela.oteitzalp.org/pluginfile.php/5828/mod_resource/content/1/RAPID%20Manual %20operador.pdf Última consulta 18/07/2017

[40] http://www.infoplc.net/descargas/46-abb-robotica/606-introduccion-a-lenguaje-rapid-programacion-irb-140 Última consulta 18/07/2017

[41] http://www.pardell.es/electroencefalografo.html Última consulta 19/07/2017

[42] http://neurofic.com/servicios/responsive/ Última consulta 19/07/2017

[43] http://www.unobrain.com/cascos-mindwave Última consulta 19/07/2017

[44] http://www.fgcsic.es/lychnos/es_es/articulos/Brain-Computer-Interface-aplicado-al-entrenamiento-cognitivo Última consulta 31/07/2017

[45] Azahara Gutiérrez Corbacho, Trabajo Fin de Grado "Desarrollo de una interfaz para el control del robot IRB120 desde MATLAB",2014.

[46] http://new.abb.com/products/robotics/es/controladores/irc5 Última consulta 01/08/2017

[47] https://library.e.abb.com/public/2b5b950d68a0503cc1257c0c003cb703/3HAC041344-es.pdf Última consulta 01/08/2017

[48] https://es.mathworks.com/ Última consulta 03/08/2017

[49] https://www.emotiv.com/epoc/ Última consulta 07/09/2017

[50] www.medicalexpo.es/fabricante-medical/casco-eeg-2788.html Última consulta 07/09/2017

IX. Apéndices

Apéndice A. Siglas y abreviaturas

EEG: Electroencefalografía
BCI: Brain computer interface
ABB: Asea Brown Boveri (empresa)
GBA: Gamma-Band activity (actividad en la banda gamma)
DOF: Degrees of freedom (Grados de libertad)
ANFIS: sistema de inferencia neuro-difuso adaptativa
Hz: Hercios (unidad de medida de la frecuencia)
mV: milivoltios (0.001 voltios, unidad de medida de la tensión eléctrica)
µV: microvoltios (0.001 milivoltios)
RAPID: Robotics Application Programming Interactive Dialogue
ROC: Receiver Operating Characteristic (Característica Operativa del Receptor)
AUC: Area Under Curve (área bajo la curva)
PSD: Power Spectral Density (densidad espectral de potencia)

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá