GRADO EN INGENIERÍA ELECTRÓNICA
INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

ADQUISICIÓN Y TRATAMIENTO DE SEÑALES EEG: CASO DE APLICACIÓN EN JAVA

Alumno: PÉREZ FRUTOS, MIKEL (22760547T)
Directora (1): ÁLVAREZ RODRÍGUEZ, MARIA LUZ
Directora (2): MARTÍNEZ RODRÍGUEZ, RAQUEL

Curso: 2018-2019
Departamento: DEPARTAMENTO DE SISTEMAS DE INGENIERÍA Y AUTOMÁTICA

Fecha: 11/02/2019

AGRADECIMIENTOS

Dado que este es el único espacio personal de entre todas las páginas que conforman este documento, me gustaría aprovechar estas líneas para expresar mi agradecimiento a las personas que de una u otra manera han participado en mi vida, haciendo que hoy esté aquí. En primer lugar, quiero dar las gracias con mayúsculas a mi madre, ya que es la persona más importante gracias a la cual hoy estoy escribiendo estas líneas; y también a mi hermana, que en estos años se ha convertido en mi mejor amiga. También quiero agradecer a las personas que han estado a mi lado incondicionalmente durante estos años, tanto amigos que traía de fuera, como compañeros de clase que acabaron convirtiéndose en amigos.

Todos vosotros, habéis formado parte de mi presente durante este camino que nunca ha sido fácil. Por otro lado, quiero mostrar también mi agradecimiento a mi tutora, María Luz Álvarez Gutiérrez, quién me ha prestado la ayuda necesaria en todo momento en el cual he necesitado de ella. Gracias por aceptarme como alumno para realizar este trabajo y gracias también por la cercanía y el trato que he recibido. Y, por último, quiero agradecer también a Raquel Martínez Rodríguez por pensar en mí a la hora de ofrecerme esta oportunidad de proyecto cuando todavía estaba cursando tercero de carrera, y por confiar en mí de la manera en la que lo ha hecho.

Mikel Pérez Frutos
Bilbao, 2018

ÍNDICE GENERAL

1. RESUMEN

1.1. DESCRIPCIÓN

En este trabajo de fin de grado se realiza un análisis y procesado de las señales cerebrales con el objetivo de utilizarlas como señales de control en un caso práctico. Con éste, aparte de seguir dando continuidad a las ideas planteadas por otros alumnos cursos anteriores, se pretende seguir dando a conocer los sistemas BCI y las posibles aplicaciones que pueden tener estos en el entorno social. Por ello, se ha implementado un sistema BCI orientado al estudio de la actividad cerebral del usuario.

Las señales EEG se capturan mediante un dispositivo electrónico comercial, de la familia Neurosky, conocido como casco Mindwave Mobile. Éste realiza la función de interface de cerebro ordenador (BCI - Brain Computer Interface) capturando, a través de un conjunto de sensores, las ondas cerebrales. Haciendo uso de MatLab, se ha capturado la señal bruta en tiempo real proporcionada por el casco; y, a partir la citada información se ha definido un algoritmo que detecta los parpadeos, generando una señal de control. Ésta última será la base de una máquina de estados que posteriormente controlará una aplicación práctica.

Partiendo de los resultados obtenidos, se implementa una aplicación Java en el entorno de programación Eclipse IDE, que muestra las señales capturadas por el electrodo del casco, y aplica el citado algoritmo para la obtención de la señal de control. Dentro de la aplicación, también se ha modificado un juego de Arcade, bajo el nombre de NeuroPong, donde el usuario utilizará el parpadeo para desplazar la barra que hará de raqueta en el mítico juego de máquina recreativas. Además, el usuario podrá hacer un seguimiento de su nivel de atención durante el juego, que se podrá analizar posteriormente.

1.2. OBJETIVOS

El objetivo general de este proyecto es contribuir desde la Ingeniería al desarrollo de aplicaciones informáticas orientadas a la mejora de la calidad de vida de personas con movilidad reducida o déficit de atención.

Los objetivos específicos del proyecto son:

- Estudiar el hardware y el software del casco NeuroSky Mindwave.
- Conocer el protocolo de comunicación del casco.
- Capturar las señales captadas por el casco.
- Analizar la información emitida.
- Tratar las señales para conseguir las señales de control
- Análisis desarrollo e implementación de una aplicación.

1.3. ESTUDIOS PREVIOS.

Para poder alcanzar los citados objetivos se ha realizado analizado:

- Cómo transmite el casco la información al puerto USB.
- El tipo y la naturaleza de las ondas cerebrales recogidas por casco.
- El tipo de filtros y métodos a utilizar para el análisis de señales.

También ha sido necesario:

- Profundizar a cerca del entorno de Matlab.
- Aprender a cerca la programación en lenguaje Java en el ID de Eclipse.
- Las librerías Windows Builder o JFreeChart para este entorno.

1.4. MOTIVACIÓN

La principal motivación del proyecto es su conexión con la bioelectrónica. Con el paso del tiempo, la electrónica se ha convertido en el motor del avance de la tecnología posibilitando la creación de dispositivos capaces de facilitar el trabajo de las personas. Combinando dichos avances con los conocimientos de la medicina y de la electrónica para desarrollar instrumentos que mejoren la vida de las personas.

Uno de los estudios más interesantes que realiza esta ciencia es el análisis y tratamiento de las ondas cerebrales; ya que desde hace siglos uno de los mayores intereses del ser humano ha sido descubrir más acerca del cerebro y su funcionamiento. Esto ha sido otro de los puntos que más interés ha generado en mí a la hora de aceptar el proyecto.

Por otro lado, el hecho de poder trabajar tanto en el ámbito de la electrónica como en el ámbito de la informática ha sido otra de las atracciones de este proyecto; ya que, durante la carrera, unos de los aspectos que más interés ha generado en mí ha sido el ser capaz de mezclar la programación con el hardware para poder analizar determinados estímulos captados mediante sensores. Además, el trabajar en Java, programación que no había utilizado nunca, ha supuesto un extra y un reto que tenía ganas de plantearme; ya que considero que la electrónica y la informática trabajan muy en consonancia.

Además, dentro de todas las posibilidades de análisis que se pueden realizar sobre la información que se obtiene del casco Mindwave Mobile, la docente Raquel Martínez Rodríguez, actualmente inmersa en un proyecto junto con un módulo de psicología, citó la posibilidad de orientar el trabajo a una aplicación tipo dentro de algún campo relacionado con la psicología. Es por ello, que el hecho de que el proyecto pueda tener un componente social y una aplicación orientada a las personas por ejemplo con déficit de atención, es la otra de las razones de peso que hizo que decidiera orientar mi trabajo de fin de carrera a un proyecto con esta temática.

2. ANTECEDENTES Y ESTADO DEL ARTE

En las últimas décadas, los avances en el campo de la neurotecnología han supuesto el desarrollo de canales de información directos entre el cerebro y las máquinas. Estos canales que emiten datos en tiempo real de la actividad cerebral son conocidos como sistemas BCI (Brain-Computer Interface).

2.1. INFORMACIÓN CEREBRAL

2.1.1. EL CEREBRO

El cerebro es un órgano complejo que forma parte del sistema nervioso central (SNC), situado en la parte anterior y superior de la cavidad craneal donde flota en un líquido transparente, llamado líquido cefalorraquídeo. Este órgano está formado por millones de neuronas que, interconectadas mediante axones y dendritas, permiten regular todas y cada una de las funciones del cuerpo y la mente.

Cada una de las neuronas, transmite una determinada información a través de actividad electroquímica, creando pequeños campos eléctricos. Cuando cientos de miles de éstas funcionan al mismo tiempo, el campo eléctrico resultante de la suma de la actividad individual de cada una de ellas, es lo suficientemente grande como para ser medido desde fuera del cráneo. Esta medida es conocida como encefalograma (EEG) y el campo eléctrico resultante se conoce como ondas cerebrales.

Además, cada estado mental genera un patrón diferente de actividad neuronal; por tanto, existen unas ondas cerebrales características para cada estado. Esto último ha permitido a la ciencia que, a partir del estudio de las citadas ondas, se pueda interpretar en qué estado mental se encuentra la persona sometida a estudio, pudiendo analizar esta información con el objetivo de vincular un estado mental a una determinada acción o situación. Éste es el concepto básico de un sistema Brain Computer Interface (BCI).

2.1.2. BIOSEÑALES

El término bioseñal engloba el conjunto de señales generadas en todos los organismos vivos. En el cuerpo humano cada sistema genera una cantidad de señales que aportan información sobre si éste se encuentra en un estado normal o está enfermo. Una enfermedad o defecto en un sistema provoca una alteración generando señales diferentes a las señales en estado normal.

Los sucesos biológicos que ocurren en el cuerpo humano producen actividades mecánicas, eléctricas y químicas. Esto supone que estos eventos generen señales útiles para el análisis y posterior diagnóstico médico que pueden ser medidas y monitorizadas. Es hacia las señales bioeléctricas hacia donde está orientado el proyecto. Éstas son las corrientes eléctricas generadas por la diferencia de potencial eléctrico a través de un

tejido, órgano o sistema celular como el sistema nervioso. Su fuente son los potenciales de transmembrana celular, el cual ante ciertas condiciones puede variar para generar un potencial de acción.

Las señales bioeléctricas más típicas son las siguientes:

- ECG (Electrocardiograma).
- ECG (Electromiograma).
- EEG (Electroencefalograma).
- EOG (Electrooculograma).

2.1.3. SINAPSIS

Una sola neurona puede mantener un potencial de reposo o voltaje a través de la membrana, puede disparar impulsos nerviosos, o potenciales de acción, y puede llevar a cabo los procesos metabólicos necesarios para seguir viva. Sin embargo, la actividad en una neurona es mucho más interesante cuando se consideran sus interacciones con otras neuronas. Las neuronas individuales hacen conexiones con neuronas blanco y estimulan o inhiben su actividad, lo que forma circuitos que pueden procesar la información entrante y producir una respuesta.

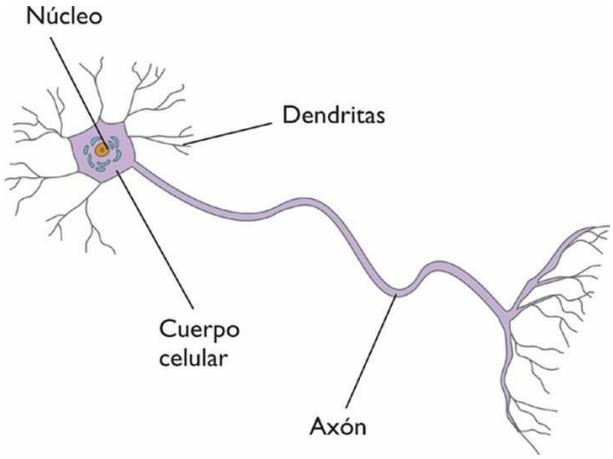

Ilustración 1. Neurona

La citada comunicación sucede en la sinapsis: el punto de comunicación entre dos neuronas o entre una neurona y una célula blanco, como un músculo o una glándula. En la sinapsis, el disparo de un potencial de acción en una neurona provoca la transmisión de una señal a otra neurona. Un potencial de acción viaja por el axón de la célula emisora, y llega al terminal del axón. El terminal del axón es adyacente a la dendrita de la célula receptora. Este punto de estrecha conexión entre axón y dendrita es la sinapsis.

La mayoría de las sinapsis son químicas, las cuales se comunican haciendo uso de los conocidos como mensajeros químicos. Sin embargo, existen situaciones en las que la sinapsis es eléctricas, donde los iones fluyen directamente entre células.

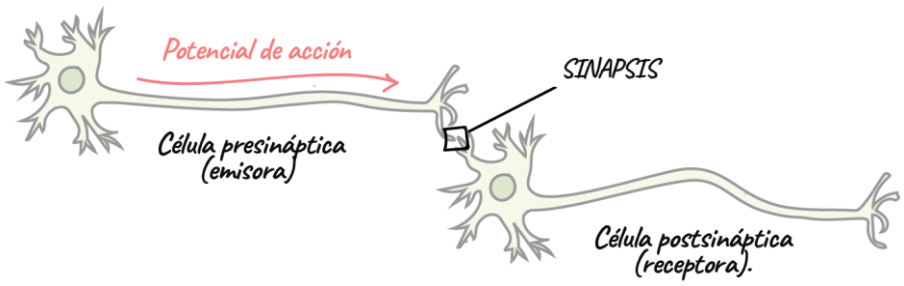

Ilustración 2. Sinapsis

2.1.3.1. SINAPSIS QUÍMICA

Cuando se produce una sinapsis química, se liberan mensajeros químicos conocidos como neurotransmisores. Los neurotransmisores transportan la información desde la neurona emisora, a la célula receptora. Como se ha citado, la sinapsis generalmente se forma entre los terminales nerviosos de la neurona emisora y el cuerpo celular o las dendritas de la neurona receptora.

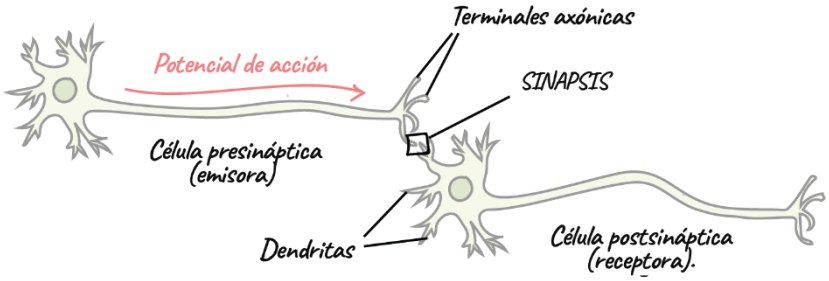

Ilustración 3. Sinapsis química

Donde,

- Un axón puede tener múltiples ramificaciones que le permite hacer sinapsis con varias células receptoras.
- Una neurona puede recibir miles de entradas sinápticas de muchas neuronas emisoras diferentes.

Dentro de la terminal axónica de una célula emisora hay muchas vesículas sinápticas. Éstas son esferas membranosas que contienen las moléculas del neurotransmisor. Además, existe un pequeño espacio entre la terminal axónica de la neurona emisora y la membrana de la célula receptora, este espacio se llama espacio sináptico.

La imagen que se muestra a continuación representa el terminal del axón de la célula presináptica que contiene vesículas sinápticas con neurotransmisores. En la superficie exterior del terminal del axón hay canales de calcio activados por voltaje. En el otro extremo, hay una célula receptora cuya superficie está cubierta de receptores para el neurotransmisor.

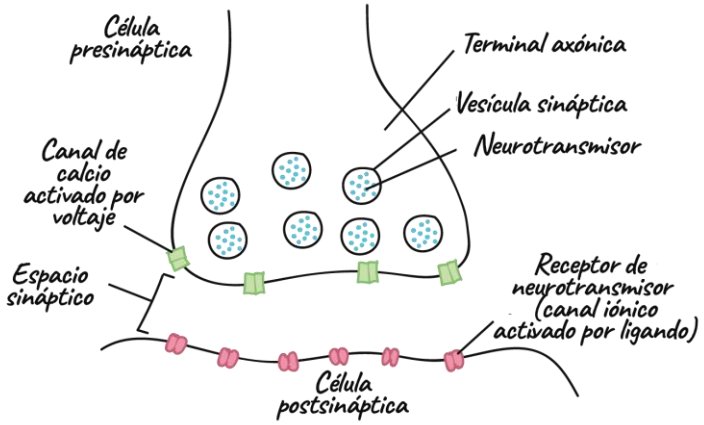

Ilustración 4. Espacio sináptico

Cuando un potencial de acción o impulso nervioso llega al terminal del axón, acciona canales de calcio activados por voltaje en la membrana celular. El Ca2+, que está mucho más concentrado fuera de la neurona que dentro, entra a la célula. El Ca2+ permite que las vesículas sinápticas se fundan con la membrana del terminal del axón, con lo que se liberan los neurotransmisores en el espacio sináptico.

La imagen que se adjunta se representa lo que sucede cuando el potencial de acción llega a la terminal axónica, y se provoca un flujo de iones:

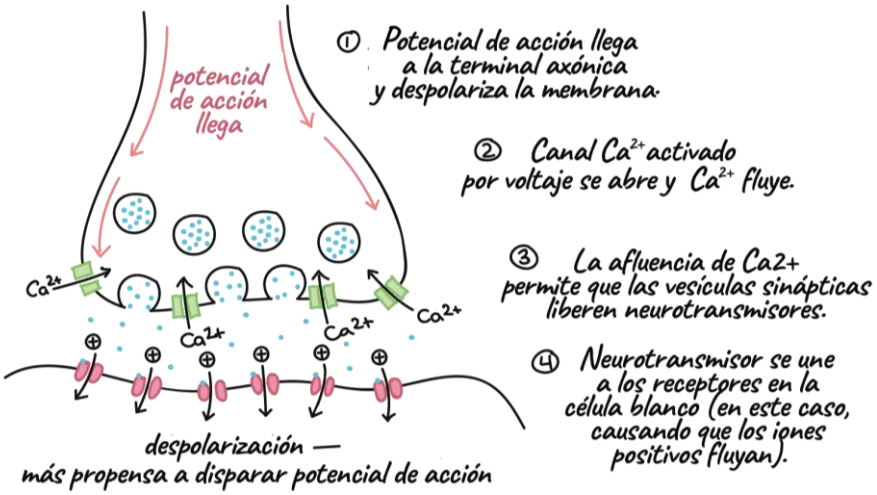

Ilustración 5. Transmisión de neurotransmisores

Donde,

1. El potencial de acción alcanza la terminal axónica y despolariza la membrana.
2. Se abren los canales de calcio activados por voltaje y los iones de calcio entran.
3. El ingreso de iones de calcio hace que las vesículas liberen el neurotransmisor.
4. El neurotransmisor se une a los receptores en la célula (entrada de iones p+).

Las moléculas de neurotransmisor se difunden por el espacio sináptico y se unen a las proteínas receptoras en la célula receptora. La activación de los receptores provoca la apertura o cierre de canales iónicos en la membrana celular.

2.1.3.2. SINAPSIS ELÉCTRICA

En una sinapsis eléctrica, a diferencia de una sinapsis química, existe una conexión física directa entre la neurona emisora y la neurona receptora. Esta conexión toma la forma de un canal llamado unión en hendidura, que permite que la corriente de iones fluya directamente de una célula a otra.

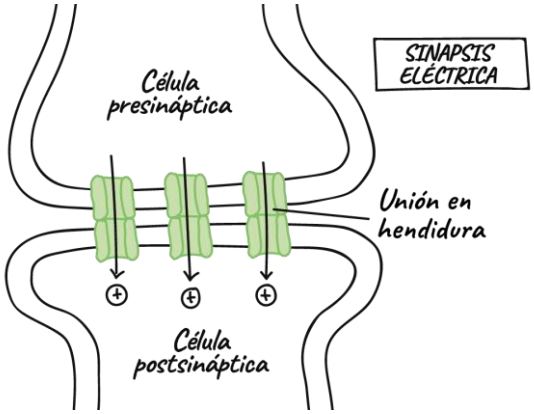

Ilustración 6. Sinapsis eléctrica

Como beneficio con respecto a la sinapsis química, la sinapsis eléctrica es más rápida. Además, ésta permite la actividad sincronizada de grupos de células. En muchos casos, pueden llevar corriente en ambas direcciones, es decir, información en ambos sentidos. Y como desventaja, a diferencia de las sinapsis químicas, las sinapsis eléctricas carecen de la versatilidad, flexibilidad y capacidad de modulación de señales.

2.1.4. ENCEFALOGRAMA

La electroencefalografía consiste en el registro de la actividad eléctrica del cerebro utilizando electrodos aplicados sobre el cuero cabelludo. Estos electrodos se colocan sobre las zonas correspondientes a las diferentes áreas del cerebro para así detectar y registrar patrones de actividad eléctrica y verificar la presencia de anomalías. En la actualidad, y por su facilidad de uso y colocación, se utilizan generalmente gorros con electrodos colocados en las posiciones convencionales del Sistema Internacional 10-20.

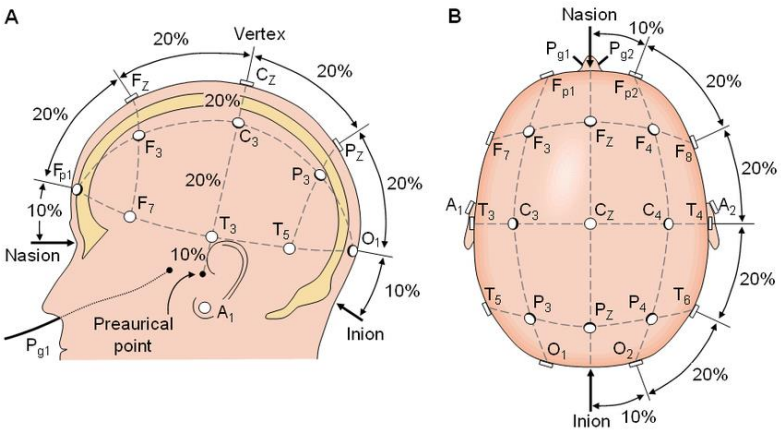

Ilustración 7. Posiciones convencionales del Sistema Internacional 10-20

Sin embargo, es complicado medir la actividad cerebral ya que el cerebro humano se compone de miles de millones de neuronas interconectadas del tamaño de una cabeza de alfiler, por lo que la localización de los electrodos y el paciente siempre serán determinantes para esta actividad.

Hay diferentes métodos de captación:

- **Captación profunda**: insertar electrodos de aguja en el tejido nervioso.
- **Electrocorticograma (ECoG):** los electrodos se colocan en la superficie cerebral.
- **Electroencefalograma (EEG):** utilizar electrodos de superficie sobre el cráneo.

Este último es el que utiliza el casco con el que se está trabajando para la parte encargada de la captación de información en el proyecto. La amplitud de una señal de EEG varía de aproximadamente 1 a 100 µV en un adulto normal, y es más o menos de 10 a 20 mV cuando se mide con electrodos subdurales.

2.1.4.1. RITMOS CEREBRALES

Las señales del EEG son divididas en grupos basadas en su contenido de frecuencia. La manera más común es dividir dichas frecuencias en bandas definidas mediante estudios de los fenómenos psicológicos y fisiológicos asociados a la actividad cerebral. Cada frecuencia, tiene su propio conjunto de características que representa un nivel específico de la actividad cerebral y por lo tanto un estado único de la conciencia.

NOMBRE	FRECUENCIA
RITMO DELTA	0.5 - 3.5 HZ
RITMO THETA	4 - 7.5 HZ
RITMO ALPHA	8 - 13 HZ
RITMO BETA	14 - 30 HZ
RITMO GAMMA	< 30 HZ

Tabla 1. Bandas de frecuencia de los ritmos cerebrales

Donde:

- **RITMO DELTA:** También es conocido como la onda del sueño, ya que este ritmo suele aparecer generalmente en estados de sueño profundo. Su amplitud varía de los 20 a 200 µV, y su rango de frecuencia oscila entre 0,5 y 3,5 Hz.

- **RITMO THETA (Θ):** Se presenta generalmente con la activación del lóbulo temporal, su banda de frecuencia está entre los 4 y 8 Hz y su amplitud varía entre los 20 y 100 µV, y se suele presentar personas que se encuentran sometidos a altos niveles de estrés.

- **RITMO ALFA (A):** Este ritmo corresponde al estado de relajación, donde la persona tiene poca actividad mental o se encuentra con los ojos cerrados. Éste se atenúa cuando la persona inicia alguna clase de actividad mental que requiera concentración. Las señales asociadas a este ritmo presentan una frecuencia comprendida entre los 8 y 14 Hz, una amplitud de entre 20 y 60 µV. La fuente principal de estas señales se encuentra en la región occipital del cerebro.

- **RITMO BETA (B):** Su amplitud se encuentra en el rango entre los 2 y 20 µV y su frecuencia entre los 14 y 26 Hz. Éste está vinculado principalmente con los movimientos de las extremidades y actividades musculares, pero también puede asociarse a un estado mental de concentración por parte de la persona y se puede encontrar principalmente en la región frontal y central del cerebro.

- **RITMO GAMMA:** Son las frecuencias superiores a 30 Hz. Suelen tener la frecuencia más alta y la amplitud más baja, y se relacionan con actividades cerebrales intensas.

Existe también un sexto ritmo cerebral llamado ritmo µ, el cual puede observarse en el rango de frecuencias de 8 a 12 Hz. Aunque es básicamente el mismo rango del ritmo Alpha, el ritmo µ se observa en la corteza sensorio motora, mientras que el Alpha solo se observa en la corteza visual.

2.1.4.2. PREPROCESAMIENTO

Cuando los electrodos reciben la señal bruta consecuencia de los campos eléctricos generados en el córtex es necesario realizar algún tipo de preprocesamiento sobre dicha señal antes de realizar cualquier tipo de análisis matemático.

Para ello:

- Se reorganizan los datos que se obtienen.
- Se extrae algún punto de interés sin realizar alguna otra modificación.
- Se remueven algunos elementos de un electrodo con mala calidad.
- Se filtran o transforman para facilitar el análisis posterior.

La razón detrás del pre-procesado es que toda señal EEG viene acompañada por un ruido que entorpece el procesamiento. Este ruido suele ser parte de la señal, por lo que eliminar parte del ruido puede derivar en eliminar alguna porción de datos correctos. Por ello se ha de balancear la señal y el ruido para obtener un buen procesado. Esto último depende del objetivo del estudio que se vaya a realizar, ya que lo que para una investigación es ruido para otra puede ser una señal de valor.

Todo ello depende de:

- ❖ Los ritmos cerebrales a estudiar.
- ❖ Si se necesitan frecuencias superiores a los 30 Hz o no.
- ❖ Si los ERP son el motivo de estudio, o se buscan eliminarlos.

Por ello el fijar el balance señal-ruido, influye en los protocolos de pre-procesado de cualquier investigación y es un parámetro muy personal en el cual no existe un camino correcto o errado, sino que depende del objetivo a alcanzar.

2.1.4.2. CARTOGRAFÍA CEREBRAL

Con el desarrollo de la informática y de los algoritmos matemáticos necesarios, se ha posibilitado el estudio temporal y espacial cuantitativo de la actividad de una señal EEG. La cartografía cerebral consiste en un conjunto de técnicas que permiten representar la actividad neurobiológica cerebral mediante mapas de actividad. Estos mapas de actividad hacen posible establecer relaciones entre el procesamiento de información que tiene lugar cuando se somete a un sujeto a una tarea cognitiva.

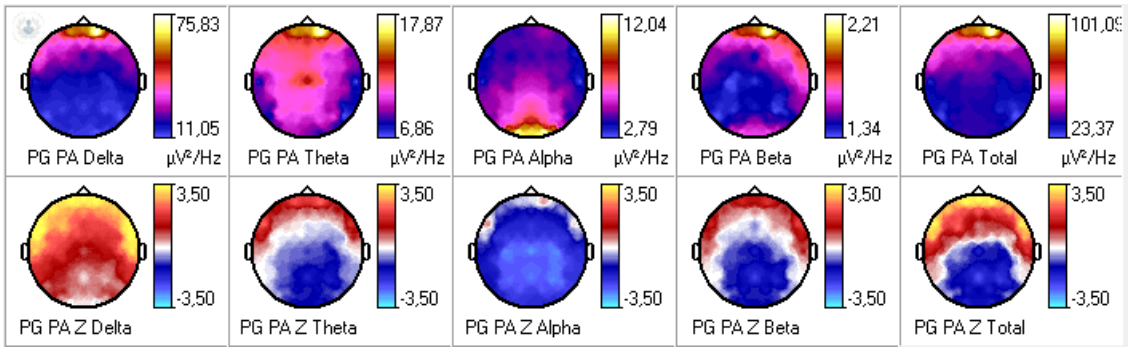

Ilustración 8. Cartografía cerebral

Además, se trata de una técnica de representación que hace que la interpretación de la información sea más sencilla. El interés principal de ésta reside en su capacidad de representar la información en forma de imágenes donde se sintetiza el contenido de todas las señales registradas por los electrodos. Así, las relaciones espacio-temporales entre las diferentes regiones del cerebro se pueden establecer de una forma mucho más fácil. De esta manera, los fenómenos de corta duración difíciles de detectar, aparecen claramente sobre la cartografía.

2.1.4.3. CAPTACIÓN DE LAS ONDAS CEREBRALES

La captación de las ondas cerebrales puede realizarse de diferentes maneras, según la posición de los electrodos:

- ❖ Superficiales.
- ❖ Basales
- ❖ Quirúrgicos.

Por otro lado, según el tipo de electrodo usado la captación de las ondas se denomina:

- Si los electrodos son superficiales se denomina electroencefalograma (EEG).
- Cuando son electrodos quirúrgicos, se le llama electrocorticograma (ECoG).
- Si se usan electrodos quirúrgicos más profundos, se llama Estéreo EEG (E-EEG).

2.1.5. EVOLUCIÓN DE LOS ENCEFALOGRAMAS

En 1870 los médicos Fritsch y Hitzig observaron que al estimular determinadas áreas laterales del cerebro se producían movimientos en el lado opuesto del cuerpo. Años más tarde, el científico Richard Caton confirmó que el cerebro producía corrientes eléctricas, siendo uno de los mayores descubrimientos en este campo.

En 1913, Prawdwick-Neminski registró el electroencefalograma de un perro, siendo el primero en intentar una aproximación de estas observaciones. Más tarde, a principios del Siglo XX, todos los experimentos se realizaban sobre cerebros descubiertos; ya que al tratarse de cambios eléctricos muy pequeños los que se detectaban y de no disponer de procedimientos de amplificación de dichos cambios, era muy difícil la obtención de señales que alcanzaran el exterior del cráneo para poder recogerlas con los electrodos.

En 1928, Hans Berger continuó con las investigaciones de Richard Caton y descubrió lo que se llamaría a partir de ese momento el ritmo de Berger. Sin embargo, fueron necesarios más conocimientos técnicos para conseguir un aparato que obtuviera datos concluyentes sobre la actividad cerebral. Y fue en 1934, cuando en una demostración pública en la reunión de la Sociedad de Fisiología en Cambridge, Adrian y Matthews verificaron por primera vez el ritmo de Berger.

Después, el neurólogo Hans Berger continuó su investigación y observó que cuando una persona abría los ojos o resolvía un problema se alteraba el ritmo de Berger. Esto fue verificado también por Adrian y Matthews quienes, al tener mejores conocimientos técnicos, mejoraron los resultados de Berger e incluso demostraron que el ritmo regular surgía sólo en las áreas visuales de asociación y no en todo el cerebro.

Todos los descubrimientos citados, comenzaron a interesar a los investigadores hasta el punto de empezar a estudiar enfermedades que hasta ese momento eran consideradas como misteriosas, como por ejemplo la epilepsia. Sin embargo, fue imposible aislar las funciones simples en la corteza cerebral, lo que derivó en la necesidad de un estudio del cerebro como un órgano total. Por ello, en la actualidad el electroencefalograma constituye un método imprescindible en el diagnóstico de los fenómenos críticos cerebrales, así como de su control evolutivo y pronóstico.

En el apartado relacionado con la interacción del ser humano con la computadora, las interfaces cerebro computador y la inteligencia artificial son las líneas que han tomado

mayor relevancia. Particularmente, se están llevando a cabo experimentos con usuarios utilizando sistemas BCI con el objetivo de apoyar en mayor proporción a aquellos colectivos con algún tipo de discapacidad, dependencia o trastorno, o incluso integrar algunos dispositivos como un accesorio para, por ejemplo, un videojuego.

Actualmente, se llevan los dispositivos BCI más utilizados son:

- Anticap.
- Mindwave Mobile (Neurosky).
- Nemicomi (refleja el estado de ánimo).
- MindBall.
- MindPlay.
- Algunos otros con un porcentaje alto de efectividad como Intendix-Soci.

2.2. SISTEMAS BCI

Actualmente, el poder interactuar con las máquinas a través de nuestros pensamientos es algo en lo que se está trabajando de manera muy intensa. Se conoce como interfaces cerebro-computador a los sistemas capaces de medir la actividad neurológica del cerebro y traducirlas para procesarlas con un programa de ordenador.

Estos fueron ideados en 1973 por Jacques Vidal en la Universidad de California; pero no fue hasta el inicio de los años 90 cuando se hicieron las primeras pruebas en seres humanos. Más adelante, con el desarrollo de los EEG con métodos no invasivos para adquirir las señales, la mayoría de los trabajos publicados se centraron en este tipo de interfaz ya que no necesitaba cirugía.

Sin embargo, estos sistemas en comparación con los sistemas invasivos poseen menor resolución y son menos efectivos para adquirir señales de alta frecuencia porque el cráneo amortigua las señales. Esto supone que las ondas electromagnéticas creadas por las neuronas se dispersen y que requieran un tiempo de preparación antes de cada uso. Además, éstas últimas dependen también de la concentración del usuario.

Un sistema BCI suele construirse siguiendo los siguientes pasos:

- Adquirir las señales crudas de los electrodos, en tiempo real o para analizar.
- Extraer las características con técnicas matemáticas discretas de procesamiento.
- Aplicar un módulo de clasificación e interpretación de las anteriores.
- Enviar la respuesta a un dispositivo de salida.

Estas etapas son controladas mediante un protocolo de funcionamiento que define el inicio, el tiempo de operación, los detalles del procesamiento de la señal, la naturaleza de los comandos del dispositivo y la supervisión del rendimiento.

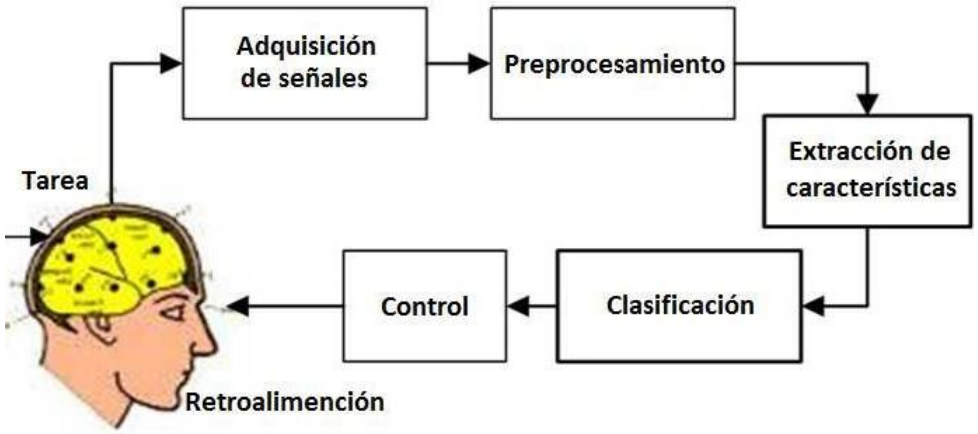

Ilustración 9. Pasos en el funcionamiento de un BCI

Donde, las principales dificultades en el desarrollo y la implementación de los sistemas BCI basadas en la electrografía, son la adquisición y pre procesamiento de las señales, la extracción de las características y la clasificación de los datos de EEG, ya que debe hacerse en tiempo real para generar una respuesta a una velocidad razonable.

2.2.1. HISTORIA DE LOS SISTEMAS BCI

Las tecnologías BCI surgen en 1979 cuando aparecieron los primeros proyectos de investigación debido a la relación entre las señales EEG y los movimientos reales e imaginarios de los usuarios. El objetivo principal de esta tecnología era la creación de una interfaz que permitiera a las personas con graves problemas de discapacidad motora o cerebral poder comunicarse, expresar sentimientos o estados de ánimo, o incluso hasta realizar tareas gracias a dispositivos electrónicos.

En las últimas décadas los avances han supuesto la creación y perfeccionamiento de canales de información directos entre el cerebro y las máquinas. Los siguientes factores han supuesto una expansión en la investigación:

- La disminución en los costos de adquisición de estos sistemas.
- Las mejoras en la tecnología.
- Los nuevos sensores.
- El uso de comunicación inalámbrica.

Los citados avances suponen que, en un periodo no muy extenso de tiempo, los dispositivos BCI estarán al alcance del cualquier usuario y que cambiarán la forma en la que se interactúa con los computadores.

En la actualidad, los sistemas BCI se están utilizando fuera de los laboratorios con fines complementarios a los científicos, como por ejemplo en hogares. Además, en los últimos años y gracias a la disponibilidad del PC cada vez más potente y la disposición de entornos de desarrollo software más versátiles, el número de investigadores en el ámbito de la tecnología BCI ha crecido considerablemente.

2.2.2. PROCESADO DE LA INFORMACIÓN EN UN SISTEMA BCI

El procesado de la señal en los sistemas BCI se divide, generalmente, en cuatro etapas:

- En primer lugar, se realiza un pre-procesado donde se filtran las señales EEG y se eliminan algunos artefactos superpuestos a la señal de interés.
- Después, se extraen determinadas características específicas de la señal EEG.
- A continuación, se aplican métodos de selección de características que escogen las más significativas dentro de lo extraído.
- Finalmente, el algoritmo de clasificación traduce el conjunto de características seleccionado en un comando concreto, relacionado con la intención del usuario.

2.2.3. MODELO FUNCIONAL GENÉRICO

La finalidad de un sistema BCI es el de captar la actividad cerebral, procesarla para extraer los datos que interesan y después utilizarlos para interaccionar con el dispositivo o aplicación al uso. Cabe destacar que ésta se basa en las señales cerebrales, sin incluir la actividad eléctrica producida por los músculos.

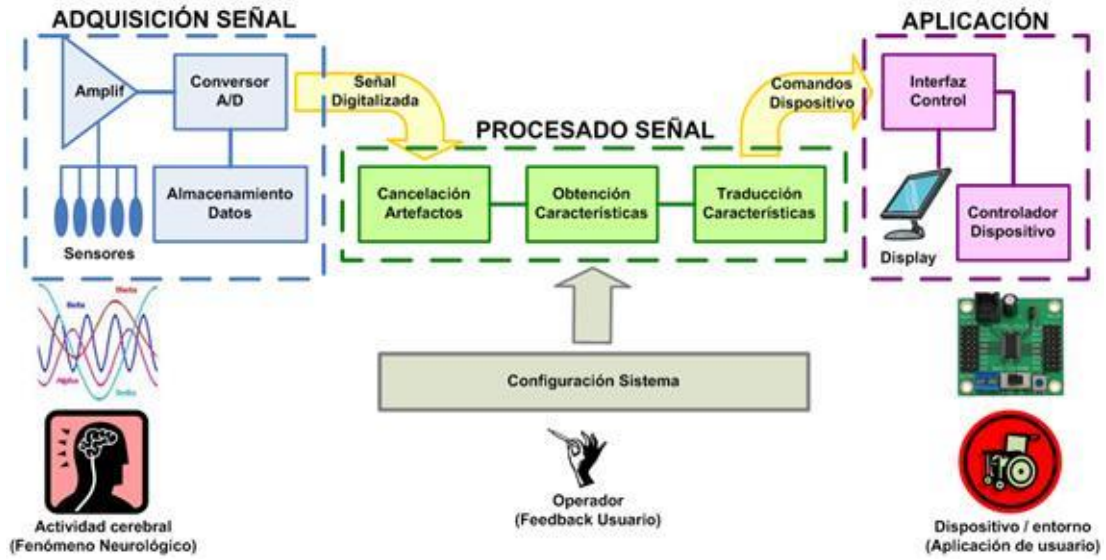

Ilustración 10. Modelo de funcionamiento genérico de un sistema BCI.

Los pasos a seguir que describen el funcionamiento de una BCI, son los siguientes:

1. ADQUISICIÓN DE LA SEÑAL: Adquirir la señal con el sensor utilizado, amplificarla y realizar la conversión analógico digital.

2. PROCESAMIENTO: Se distinguen 3 etapas en este paso:

- **CANCELACIÓN DE ARTEFACTOS:** Se eliminan los ruidos provenientes de actividad eléctrica relacionada con los músculos y que distorsionan la señal.

- **OBTENCIÓN DE LAS CARACTERÍSTICAS.** Se traduce la señal de entrada en un vector de características en relación al fenómeno neurológico asociado.

- **DECODIFICACIÓN:** Se traducen las características obtenidas en las señales de control que se usaran en el dispositivo a controlar.

3. APLICACIÓN: Se reciben las señales de control del anterior y se realizan las accionas configuradas en función de dichas señales

2.2.4. TIPOS DE SISTEMAS BCI

2.2.4.1. INVASIVO

En los sistemas invasivos los sensores se implantan directamente en el cerebro mediante una neurocirugía. Estos producen señales de mayor calidad de todos los dispositivos de BCI; pero son propensos a que con el tiempo se genere una cicatriz de tejido por acumulación, que produce que la señal se debilite o desaparezca por completo, ya que el cuerpo reacciona a los objetos extraños en el cerebro.

2.2.4.2. PARCIALMENTE INVASIVO

En los sistemas parcialmente invasivos, los sensores se implantan dentro del cráneo mediante cirugía sin invadir la materia gris; por lo que las señales siguen siendo de buena calidad, con la ventaja de que el riesgo de que se creen cicatrices es mucho menor que los del caso anterior.

2.2.4.3. NO INVASIVO

La mayoría de las investigaciones y experimentos publicados sobre sistemas BCI en los últimos años se realizan con interfaces de tecnologías no invasivas. Esto es debido a que son sencillos de instalar y remover y no requieren cirugía. Sin embargo, la resolución que alcanzan es mucho menor y no son capaces de detectar señales de alta frecuencia.

2.2.5. CLASIFICACIÓN DE LOS SISTEMAS BCI SEGÚN LA SEÑAL DE ENTRADA

2.2.5.1. ENDÓGENOS

Dependen de la capacidad del usuario para controlar su actividad electrofisiológica, como por ejemplo la amplitud del EEG en una banda de frecuencia específica sobre un área concreta del córtex cerebral. Estos requieren de un período de entrenamiento intensivo y son los siguientes:

A. BASADO EN POTENCIALES CORTICALES LENTOS:

Son cambios lentos de voltaje generados sobre el córtex, con una duración que oscila entre 0.5 y 10 segundos. Los valores negativos se asocian con el movimiento y otras

funciones que implican una activación cortical. Además, se ha demostrado que las personas pueden aprender a controlar estos potenciales.

B. BASADO EN IMÁGENES MOTORAS O RITMOS SENSORIOMOTORES:

Se basa en un paradigma de dos o más clases de imágenes motoras (movimiento de la mano derecha o izquierda, de los pies, de la lengua...) u otras tareas mentales (rotación de un cubo, realización de cálculos aritméticos...). Éstas producen cambios en la amplitud de los ritmos sensoriomotores µ (8-12 Hz) y ß (16-24 Hz), registrados sobre la zona somatosensorial y motora del córtex, y que presentan variaciones en la ejecución de un movimiento real, la imaginación de uno o la preparación para el mismo.

2.2.5.2. EXÓGENOS

Dependen de la actividad electrofisiológica provocada por estímulos externos y no necesitan de una etapa intensiva de entrenamiento y son los siguientes:

A. BASADOS EN POTENCIALES EVOCADOS P300:

El potencial P300 es un pico de amplitud que aparece en el EEG aproximadamente unos 300ms después de haberse producido un estímulo auditivo o visual poco frecuente. Para ello se somete al usuario a un conjunto de estímulos de los que solo un grupo reducido tiene relación con la intención del usuario. Así, los estímulos de interés, al estar mezclados con otros estímulos mucho más comunes, provocan la aparición de un potencial P300 en la actividad cerebral del usuario.

B. BASADOS EN POTENCIALES EVOCADOS VISUALES DE ESTADO ESTABLE:

Se detectan en el EEG sobre la zona visual del córtex cerebral después de aplicar un estímulo visual al usuario. Estos potenciales son estables si la tasa de presentación del estímulo visual está por encima de 6 Hz (6 repeticiones por segundo). Cuando el usuario enfoca su mirada en una imagen que parpadea a una frecuencia determinada, es posible detectar dicha frecuencia analizando el espectro de la señal EEG.

2.2.6. PLATAFORMAS SOFWARE PARA SISTEMAS BCI

Las plataformas de desarrollo de sistemas BCI deben ofrecer funcionalidades como:

- La adquisición de datos.
- La extracción de características.
- La clasificación.
- Los módulos de presentación y realimentación.

Es por esto que muchos fabricantes han desarrollado su propio conjunto de programas en base a diferentes requisitos, lenguajes de programación y usuarios potenciales. Este conjunto de programas, suelen ser de código cerrado y de obtención bajo previo pago;

sin embargo, han surgido varias plataformas que sí están enfocadas a su libre uso disponiendo del código abierto y licencias GNU públicas, facilitando la modificación y la inclusión de aportaciones de terceros.

PLATAFORMA	WINDOWS	MAC OS X	LINUX	LICENCIA
BCI2000	X	X		GPL
OpenVIVE	X		X	LGPL
BCILAB	X	X	X	GPL
BCI++	X			GPL
TOBI	X		X	GPL, LGPL
xBCI	X	X	X	GPL
BF++	X			FREE

Tabla 2. Relación de Plataformas SW para el Desarrollo de Sistemas BCI.

2.2.7. EJEMPLOS DE SISTEMAS BCI

2.2.7.1. EMOTIV INSIGHT

Consiste en un dispositivo de captación de EEG inalámbrico de 5 canales que cubre las localizaciones: frontal, temporal y parieto-occipital del cerebro. Éste está diseñado para el uso cotidiano de personas que buscan entender y mejorar sus propios cerebros.

Ilustración 11. Diadema Emotiv Insight

El dispositivo permite captar las ondas cerebrales e identificar si la persona bajo estudio está nerviosa, estresada o entusiasmada, entre otras opciones, utilizando sensores de polímero patentados hidrófilos (repelen la humedad del aire y la piel). La información que recopila el aparato permite construir un electroencefalograma en apenas segundos

y sin cables, y se puede utilizar para analizar el impacto de diferentes factores externos en la persona y sus emociones. Además, tiene tres sensores EEG y dos de referencia y su precio actual en el mercado ronda los 299 dólares.

2.2.7.2. MUSE

Muse es un potente y compacto sistema de electroencefalografía (EEG) que aprovecha mejoras en la tecnología de sensores secos, de bluetooth y de duración de la batería, así como avances significativos en el procesamiento de señales digitales. Este dispositivo facilita el acceso y el uso de datos de ondas cerebrales, dentro y fuera del laboratorio y en entornos del mundo real.

La diadema MUSE es una herramienta que mide las señales del cerebro como un monitor de ritmo cardíaco detecta su latido del corazón. Dispone de 7 sensores finamente calibrados que detectan y miden la actividad del cerebro, ubicados de la siguiente manera: dos en la frente, dos detrás de las orejas y tres sensores de referencia.

Ilustración 12. Diadema Muse

También es posible utilizar este sistema como una herramienta de gimnasia cerebral que permite al usuario hacer más cosas con su mente además de ayudar a relajarse más efectivamente. La citada actividad se transmite por Bluetooth, así que es posible realizar un seguimiento por Smartphone o tablet. Su precio actual en el mercado ronda los 269€.

2.2.7.3. EMOTIV EPOC +

Se trata de un sistema que dispone de 14 canales para la obtención de una EEG. Está diseñado para la investigación del cerebro humano escalable y contextual, y para aplicaciones avanzadas de la interfaz cerebro-computadora; es decir, sistemas BCI. Además, proporciona acceso a datos cerebrales de grado profesional con un diseño rápido y fácil de usar.

Su sistema de detección analiza unas 30 expresiones, emociones y acciones diferentes. EPOC es capaz de reconocer sensaciones de excitación, meditación, concentración,

tensión o frustración. La lista de expresiones añade: gestos de sonrisa, risa, enfado, miedo, mirada cruzada y entornada, movimiento direccional del ojo, y otras muecas.

Ilustración 13. Diadema Emotiv Epoc +

Su configuración permite acceder a los datos del EEG brutos que son de elevada calidad y también permite realizar investigaciones que aprovechen las citadas detecciones para comandos mentales, métricas de rendimiento o expresiones faciales. Se trata de una versión mejorada del citado sistema EMOTIV INSIGHT dentro de la misma familia de diademas fabricadas por la empresa Emotiv, y su precio ronda sobre los 799 dólares.

2.2.7.4. MINDWAVE MOBILE

El último siglo de investigación en neurociencia ha aumentado enormemente nuestro conocimiento sobre el cerebro. A día de hoy, las señales eléctricas generadas en el córtex se pueden medir colocando un sensor en el cuero cabelludo. Específicamente, este dispositivo de la familia NeuroSky, mide dichas señales analógicas eléctricas, comúnmente conocidas como ondas cerebrales, y las procesa en señales digitales.

Brainwave Type	Frequency range	Mental states and conditions
Delta	0.1Hz to 3Hz	Deep, dreamless sleep, non-REM sleep, unconscious
Theta	4Hz to 7Hz	Intuitive, creative, recall, fantasy, imaginary, dream
Alpha	8Hz to 12Hz	Relaxed (but not drowsy) tranquil, conscious
Low Beta	12Hz to 15Hz	Formerly SMR, relaxed yet focused, integrated
Midrange Beta	16Hz to 20Hz	Thinking, aware of self & surroundings
High Beta	21Hz to 30Hz	Alertness, agitation

Tabla 3. Ondas asociadas a actividades cerebrales

El casco Mindwave Mobile funciona con la tecnología ThinkGear la cual permite a un dispositivo interactuar con las ondas cerebrales. La diadema está formada por el sensor situado en la zona de la frente, los sensores para captar la EEG, los puntos de referencia ubicados en el clip para la oreja, y el chip integrado que procesa todos los datos. Las ondas cerebrales y los umbrales de atención y meditación se calculan en el chip ThinkGear y posteriormente son trasferidos por Bluetooth.

Ilustración 14. Diadema Mindwave Mobile

La tecnología de NeuroSky ThinkGear amplifica la señal de ondas cerebrales sin procesar, y elimina el ruido y el movimiento muscular. La diadema tiene integrado un algoritmo patentado de NeuroSky para caracterizar estados mentales conocido como eSense. Dicho algoritmo se aplica a la señal restante y da como resultado los valores del medidor eSense interpretado. Se ha de tener en cuenta que los valores del medidor eSense no describen un valor exacto, sino que representan los rangos de actividad.

Mindwave Mobile es compatible con Windows, Mac OS X y dispositivos móviles como por ejemplo iPhone, iPad y también dispositivos Android; y su precio es de 100€ aproximadamente.

2.2.8. JUSTIFICACIÓN DE LA ELECCIÓN DEL CASCO MINDWAVE

Actualmente, el casco Mindwave es un dispositivo para el cual existe una gran cantidad de información sobre su funcionamiento, además de extensos documentos explicando proyectos que se han llevado a cabo haciendo uso de éste. También se trata de un dispositivo para el cual se han desarrollado una gran variedad de recursos que son capaces de funcionar en consonancia con la actividad del casco. Por ejemplo, entornos de programación como Java, Matlab, Arduino, Eclipse y OpenVIVE son capaces de interactuar con el entorno de Neurosky.

Además, la diadema Mindwave Mobile es capaz de captar información sobre los ocho tipos de ondas cerebrales que existen (delta, theta, low-alpha, high-alpha, low-beta, high-beta, low-gamma, mid-gamma) y a partir de esta información calcular los niveles de atención y meditación utilizando un algoritmo conocido como eSense. A esto se le añade que se han desarrollado librerías que hacen posible acceder de forma precisa a cada una de estas informaciones, simplificando el análisis de las tramas.

Por último, es posible trabajar con el casco Mindwave tanto en Windows como Android o IOS. Esto amplía el abanico de posibilidades de desarrollo de proyectos; y además incrementa el número de plataformas y recursos disponibles para utilizar, añadiendo versatilidad al código generado y a las aplicaciones desarrolladas.

3. REALIZACIÓN DEL PROYECTO

La elaboración del proyecto se ha dividido en tres fases:

FASE DE CAPTURA DE LA INFORMACIÓN: En esta primera fase se ha realizado una captura de los datos proporcionados por el casco. Para ello se ha utilizado la plataforma de Arduino y un módulo Bluetooth para, mediante un pequeño montaje y un programa simple, tener una primera toma de contacto con la información captada y procesada por el casco Mindwave. A partir de este primer montaje, se ha podido analizar el protocolo de comunicaciones del casco y también la naturaleza de los datos que envía.

FASE DE ANÁLISIS Y PROCESADO DE LA INFORMACIÓN: Una vez conocido el protocolo de comunicaciones y la información enviada en las tramas por el casco, haciendo uso de MatLab se ha extraído toda la información captada por el electrodo del casco. Mediante representaciones graficas en tiempo real se ha analizado la naturaleza de la información y se ha realizado un primer procesado de ésta. Para el caso específico de la información sobre la señal bruta (RAW) se ha definido un algoritmo que permite detectar los parpadeos en consecuencia del ruido que estos añaden a la señal cuando el electrodo los detecta.

FASE DE IMPLEMENTACIÓN: Una vez realizado el procesado de la información, se ha desarrollado una aplicación Java para PC que recoge las tramas enviadas por el casco extrayendo la información de éstas y representándola gráficamente. Además, la aplicación implementa el algoritmo de detección de los parpadeos desarrollado en MatLab con la posibilidad de modificar el umbral de parpadeo (nivel de ruido a detectar como parpadeo). Este algoritmo se ha utilizado posteriormente para controlar el funcionamiento de un pequeño juego.

En los siguientes apartados se detallará el hardware y software utilizado en el desarrollo del proyecto, así como los desarrollos en cada una de las tres fases del proyecto.

3.1. HARDWARE UTILIZADO

3.1.1. ARDUINO UNO

Para la **fase de captura de información** uno de los elementos que se ha utilizado es la gama UNO de Arduino. Arduino es una plataforma basada en hardware y software libre de código abierto, que ofrece recursos para que cualquier persona pueda crear sus propias aplicaciones. El hardware libre engloba el conjunto de dispositivos cuyas especificaciones y diagramas son de acceso público. Esto supone que Arduino ofrece recursos para que cualquier otra persona pueda crear sus propias placas.

Por otro lado, el software libre representa el conjunto de programas informáticos cuyo código es accesible para cualquier persona; es decir, cualquiera puede hacer uso de él y

modificarlo. En este caso Arduino proporciona la plataforma Arduino IDE (Entorno de Desarrollo Integrado) para la creación de nuevas aplicaciones.

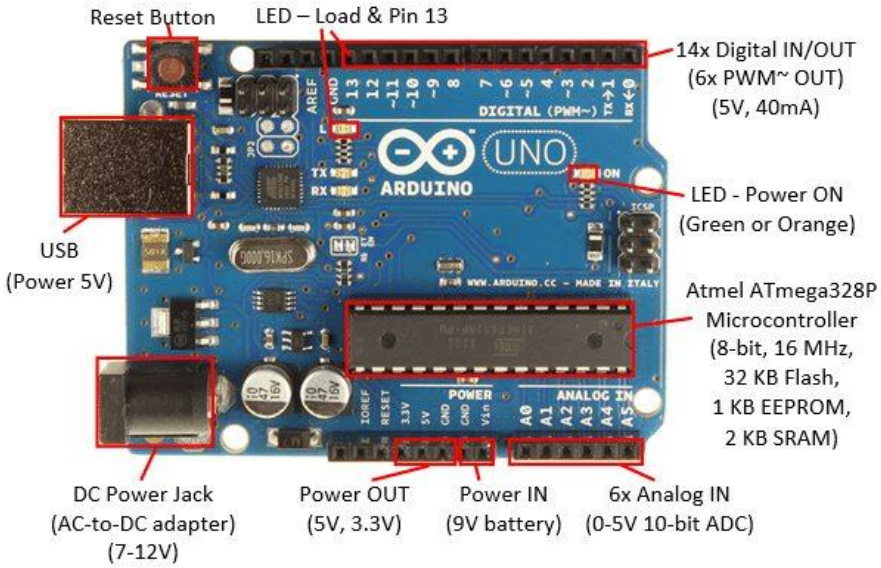

Ilustración 15. Arduino UNO.

El resultado de todo lo anterior supone una placa con todos los elementos necesarios para conectar periféricos a las entradas y salidas de un microcontrolador, y que puede ser programado en Windows, Mac y Linux.

3.1.2. MÓDULO HC08

Para la **fase de captura de información** otro de los elementos que se ha utilizado es un módulo bluetooth, en nuestro caso el HC08. El módulo HC-08 Bluetooth es parte de una nueva generación de dispositivos de transmisión de datos que hace uso del protocolo V4.0 BLE. Mediante este módulo, es como se realizará la conexión serie con el casco Mindwave Mobile.

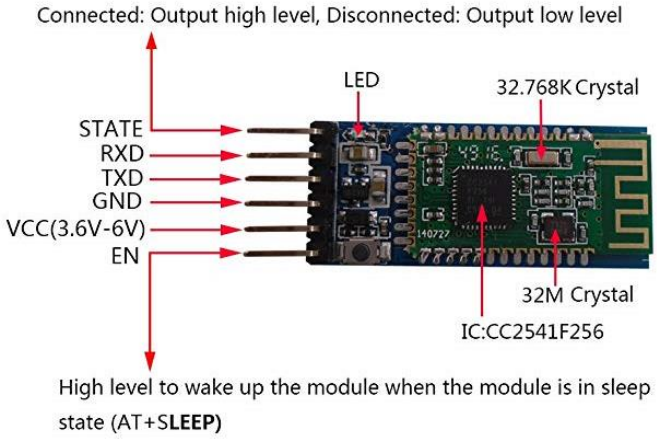

Ilustración 16. Módulo HC08

Datos:

- Frecuencia de trabajo de 2.4GHz ISM (Industrial, Scientific and Medical).
- Modulación GFSK.
- Poder de transmisión de 4dBm.
- Sensibilidad de recepción de -93dBm.
- Radio de comunicación de 80 metros de distancia.
- Tamaño: 26.9mm · 13mm · 2.2mm, fácil de soldar y parchear.
- 256 Kbyte de espacio.

Además, a partir de comandos AT, el usuario puede cambiar los modos, la frecuencia de baud serial, el nombre de equipo y otros parámetros fácilmente.

Especificaciones:

- Chip TI CC2540, V4.0 BLE.
- Voltaje de operación: 2.6v - 4.2v.
- Frecuencia de baud: 9600 (Modificable).
- Modos Master/Slave soportados.

Características:

- En el "MODO 2" el consumo en standby es de solo 0.4uA.
- En el "MODO 3" es de alrededor de 1.2 - 1.6uA.
- En el "MODO 3" puede ser activado inalámbricamente con 1 byte de datos.
- Modos: central, periférico, observador o transmisor.
- Se puede ajustar la distancia de transmisión (2m a 60m).

Es importante recordar, a la hora de realizar las conexiones, que los pines RXD y TXD van cruzados; es decir, el pin de la placa Arduino configurado como RXD irá conectado al pin TXD de la placa SH-HC-08, y análogamente, el pin asignado a TXD de la placa de Arduino se deberá conectar al RXD de la placa SH-HC-08.

3.1.3. CASCO MINDWAVE NEUROSKY

El dispositivo Mindwave Mobile ha sido utilizado en todas y cada una de las fases del proyecto ya que éste es la fuente de información con la que se trabaja. Éste está constituido por una diadema que capta la actividad eléctrica del cerebro y divide la señal recogida atendiendo a la frecuencia en diversos tipos de ondas. Esto es lo que nos permite conocer el estado mental del usuario.

Desafortunadamente, el cuerpo humano produce una gran cantidad ruido eléctrico añadido a la actividad cerebral. Es por ello al hacerse el registro la citada actividad a partir de la configuración de referencia común de un único canal, se incorpore este contacto de referencia, acoplado al lóbulo de la oreja, que ayuda a filtrar la señal no cerebral recibida. Esto que supone que también, la mayoría de las aplicaciones a realizar con este dispositivo sean de uso práctico y poco complejas.

Como se ha citado, la diadema dispone de un único sensor de aleación de acero seco colocado en la sección FP1 (parte izquierda de la frente) y de un electrodo de referencia colocado en la posición A1 (lóbulo de la oreja izquierda). El motivo por el cual éste último es utilizado como referencia, es que experimenta el mismo ruido ambiental que el situado en la parte frontal, pero con un mínimo de actividad neuronal.

Esta diadema mide, principalmente, los estados de:

- Meditación (medido por las ondas alfa/theta).
- Atención/concentración (medido por las ondas beta/gamma).

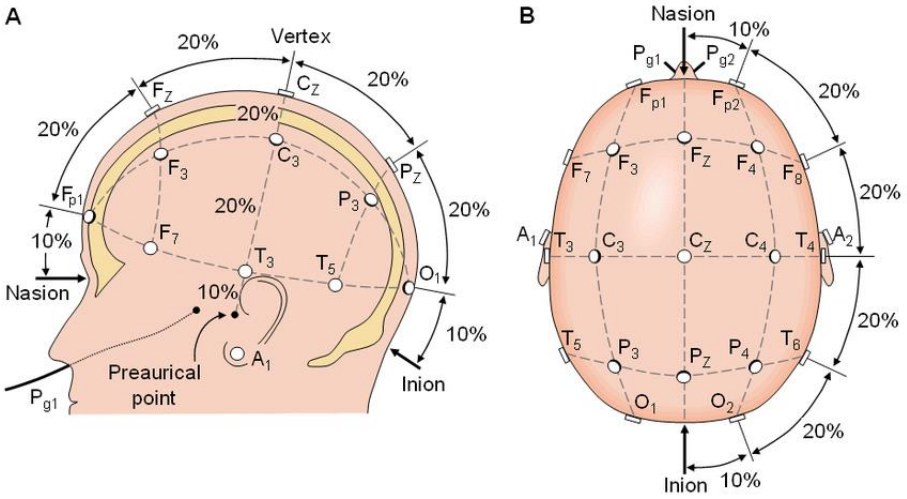

Ilustración 17. División de la corteza cerebral.

Esta información se obtiene gracias a que la compañía Neurosky ha implementado un algoritmo conocido como eSense, que mide los niveles de atención y relajación en una escala de 0 a 100, a partir de los ritmos cerebrales: alfa, beta, gamma… Por otro lado, también es capaz de captar el parpadeo gracias al sensor que se coloca en la frente, añadiendo ruido a la señal bruta captada por el electrodo.

El casco se conecta por radiofrecuencia a través de un módulo USB y funciona con la mayoría de los sistemas operativos. La mayoría de las aplicaciones de esta interfaz son de carácter no-científico, aunque la compañía especifica en su página web, que después de realizar diferentes estudios, la precisión de los datos captados por éste tiene una fiabilidad equivalente a aparatos utilizados en la medicina actual.

Este dispositivo está pensado principalmente para el desarrollo cognitivo en niños; por tanto, el software adjunto se compone de una serie de aplicaciones desarrolladas en un entorno muy sencillo y amigable. Existe una serie de juegos en los cuales se requiere el poder tener un control sobre nuestro estado mental, ya sea atención o relajación, para alcanzar unos objetivos. También dispone de otras aplicaciones como “Meditation Journal” que permite ver de manera significativa los datos de ondas cerebrales en el dominio del tiempo.

3.1.3.1. FUNCIONAMIENTO DEL CASCO

El sensor PF1 es el encargado de obtener la información; sin embargo, es el chip integrado ThinkGear el que la procesa de forma digital. En éste se procesan las ondas cerebrales y, además, a partir del algoritmo eSense, se obtienen también los valores de atención y meditación los cuales son representados en una escala del 0 al 100. Además, el casco Mindwave detecta el parpadeo de la persona que lo lleva. Éste estímulo será importante en este proyecto, ya que a partir de él se generará una máquina de estados.

Los datos obtenidos a partir de este dispositivo son los siguientes:

- Ondas cerebrales.
- Espectro de la frecuencia del EEG (alpha, beta…).
- Procesamiento y salida de los niveles de atención y meditación.
- Parpadeo.

3.1.3.2. CARACTERÍSTICAS

Las características físicas del casco son:

- Peso: 90 gramos.
- Alto: 225 mm.
- Ancho: 155 mm.
- Profundidad: 165 mm.

Las características del bluetooth son siguientes:

- Versión: 2.1.
- Potencia de salida: Clase 2.
- Voltaje mínimo: 1 V.
- Rango: 10 metros.
- Consumo de potencia: 80mA (conectado y transmitiendo).
- Indicador de batería baja: 1.1V
- UART (serie): VCC, GND, TX, RX.
- Tasa UART: 57600 baudios.
- La vida de la batería es de unas 8/10 horas con una sola pila de tipo AAA.

Las medidas que se obtienen a partir de él son las siguientes:

- Ondas cerebrales.
- Espectro de frecuencia de la EEG.
- Procesamiento y salida de los niveles de atención y meditación.
- Señal de la EEG que indica el nivel de batería o la calidad de la señal.
- Parpadeo.

3.3. SOFTWARE UTILIZADO

3.3.1. ARDUINO IDE

Para la **fase de captura de información**, además del Arduino UNO y el módulo HC08, se ha utilizado la plataforma Arduino IDE. El entorno de desarrollo integrado de Arduino, conocido como IDE (del inglés Integrated Development Environment) es un programa informático formado por un conjunto de herramientas de programación. El concepto IDE representa un entorno que ha sido empaquetado como un programa de aplicación; es decir, que consiste en:

- Un editor de código.
- Un compilador y un depurador.
- Un constructor de interfaz gráfica (GUI).

También, en el caso de Arduino incorpora las herramientas para cargar el programa ya compilado en la memoria flash del hardware a través del puerto serie. Los programas generados en Arduino están compuestos por un solo fichero de extensión ".ino", aunque también es posible organizar dicho programa en varios ficheros donde fichero principal siempre debe estar en una carpeta con el mismo nombre que éste.

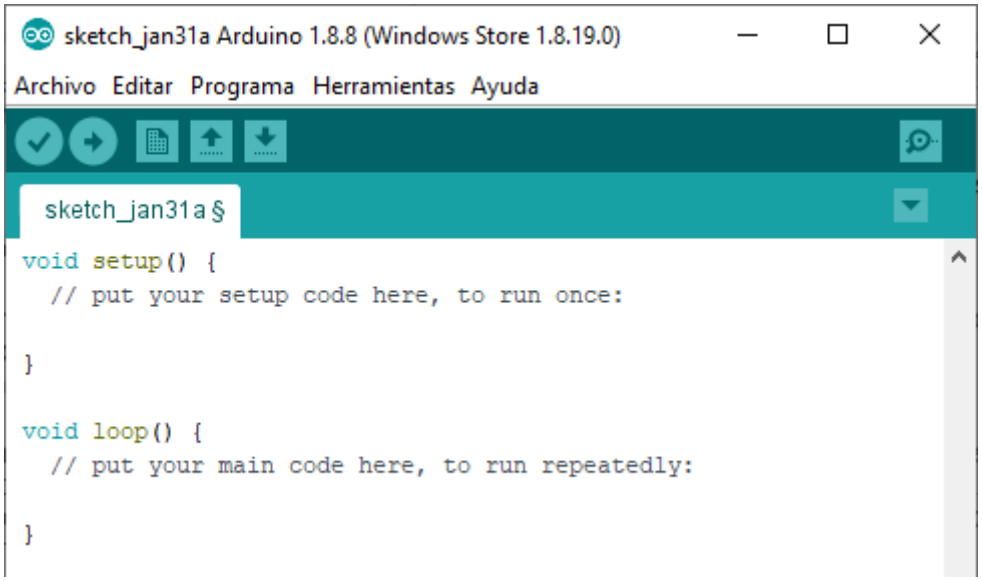

Ilustración 18. Arduino IDE.

Todos los cambios en las versiones pueden verse en:

- https://www.arduino.cc/en/Main/ReleaseNotes

El código fuente del IDE de Arduino está disponible en:

- https://github.com/arduino/Arduino/

Las instrucciones para construir el IDE desde código fuente pueden verse en:

- https://github.com/arduino/Arduino/wiki/Building-Arduino

3.3.2. MATLAB

El entorno de Matlab de MathWorks se ha utilizado en la **fase de análisis y procesado de la información** aprovechando la capacidad de éste para realizar análisis matemáticos con gran cantidad de datos, además de la posibilidad de trabajar con información en tiempo real de manera muy simple y directa.

3.3.2.1. INTRODUCCION AL MATLAB.

MATLAB es un entorno de computación y desarrollo de aplicaciones orientado a la creación de proyectos donde sea necesaria una gran cantidad de cálculos matemáticos y la visualización gráfica de estos.

Éste integra:

- El análisis numérico.
- El cálculo matricial.
- El procesado de señales.
- La visualización gráfica

Además, la plataforma dispone también de un amplio abanico de programas denominados Toolboxes, que extienden significativamente el número de funciones del programa principal. Estos cubren actualmente prácticamente casi todas las áreas principales en el mundo de la ingeniería y la simulación, destacando:

- **Procesado de imágenes y señales.**
- Control robusto.
- Estadística.
- Análisis financiero.
- Matemáticas simbólicas.
- Redes neurales.
- Lógica difusa.
- Identificación de sistemas.
- Simulación de sistemas dinámicos.

Además integra los requisitos claves de un sistema de computación técnico:

- Cálculo numérico.
- **Gráficos.**
- **Herramientas para aplicaciones específicas**
- Capacidad de ejecución en múltiples plataformas.

3.3.2.2. CARACTERÍSTICAS DEL ENTORNO

La plataforma MATLAB presenta las siguientes características:

- Cálculos intensivos desde un punto de vista numérico.
- Gráficos y visualización avanzada.
- Lenguaje de alto nivel basado en vectores, arrays y matrices.
- Colección muy útil de funciones de aplicación.

Que pueden clasificarse de la siguiente manera:

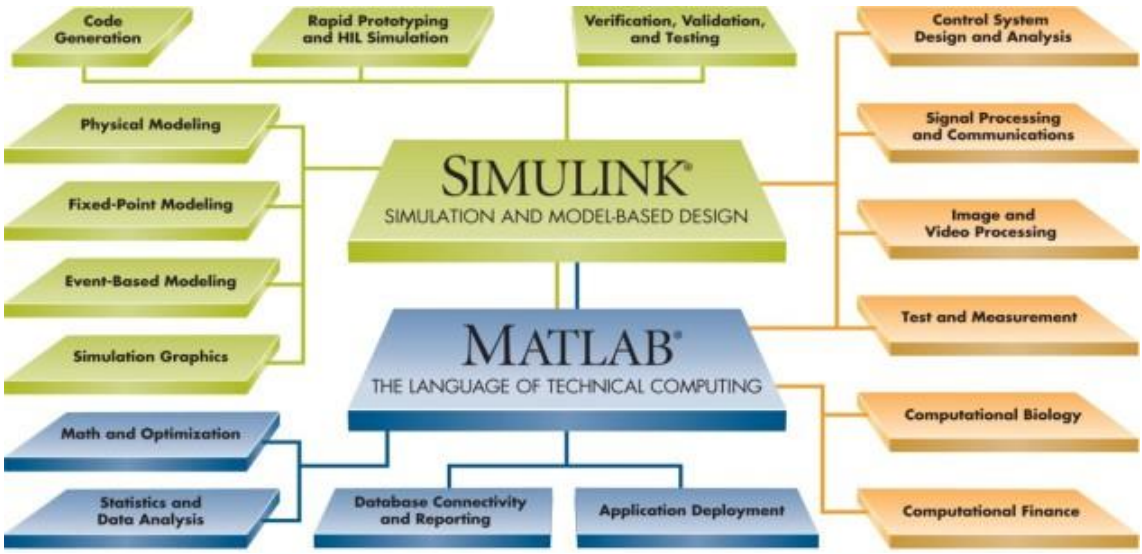

Ilustración 19. Clasificación de las herramientas de Matlab.

Toolbox especiales:

- **SEÑALES Y SISTEMAS: Procesamiento de señales y análisis de sistemas.**
- SYMBOLYC MATH: Herramienta de cálculo simbólico basada en Maple V.

3.3.2.3. SALIDAS O PRESENTACIONES

La plataforma MATLAB provee un acceso rápido a una potente graficación orientada a objetos gráficos que permite representar los resultados de un análisis, incorporar gráficos en modelos de sistemas, representar complejos objetos 3-D, y crear resultados de presentación, destacando:

- Representaciones 2-D y 3-D, incluyendo datos triangulados y reticulados.
- Representaciones 3-D quiver, ribbon, y stem.
- Control de fuentes, letras griegas, símbolos, subíndices y superíndices.
- Gráficos de torta, de barras 3-D y gráficos de barras horizontales.
- Gráficos 3-D y sólido modelado.
- Representación de imágenes y archivos I/O.
- Gráficos comentados.
- Leer/Escribir archivos de datos (Hierarchical Data Format, HDF)

3.3.3. JAVA Y ECLIPSE

El entorno de Eclipse se ha utilizado en la **fase de implementación** como plataforma con la que desarrollar una aplicación Java. Eclipse es una plataforma de desarrollo, diseñada para ser extendida de forma indefinida mediante plugins, y fue desarrollada con el objetivo de convertirse en una plataforma de integración de herramientas de desarrollo.

Este entorno no tiene asignado un lenguaje de programación específico, sino que es un entorno de desarrollo integrado genérico, con mucha popularidad entre la comunidad de desarrolladores del lenguaje Java. Además, proporciona herramientas para la gestión de espacios de trabajo, escribir, desplegar, ejecutar y depurar aplicaciones.

3.3.3.1. PRINCIPALES CARACTERÍSTICAS

PERSPECTIVAS, EDITORES Y VISTAS.

Una perspectiva consiste en una preconfiguración de ventanas y editores, relacionadas entre sí, que permiten trabajar en un determinado entorno de trabajo de forma óptima.

GESTIÓN DE PROYECTOS

Un proyecto engloba el conjunto de recursos relacionados entre sí, como, por ejemplo:

- El código fuente.
- Documentación.
- Ficheros de configuración.
- Árbol de directorios

El entorno de desarrollo integrado proporciona al usuario asistentes y ayudas para la creación de proyectos. Es decir, cuando se crea uno, se abre la perspectiva adecuada al tipo de proyecto creado, con la colección de vistas, editores y ventanas preconfigurada por defecto.

DEPURADOR DE CÓDIGO

La plataforma incluye un potente depurador, y que visualmente ayuda a mejorar el código. Para ello sólo se debe ejecutar el programa en modo depuración (con un simple botón). La perspectiva de depuración, donde se muestra de forma ordenada toda la información necesaria para realizar dicha tarea.

COLECCIÓN DE PLUG-INS:

Existe una gran cantidad de plugins que pueden utilizarse en el entorno de la aplicación, algunos de ellos publicados por los desarrolladores de Eclipse y otros generados por

otros desarrolladores. Los hay de carácter gratuito, de pago, o bajo distintas licencias; pero casi cualquier aplicación tiene el *plug-in* adecuado para desarrollarla.

3.3.3.1.1. PLUG-IN JDT

El plugin JDT aporta un conjunto de complementos que agregan a la plataforma Eclipse las capacidades de un entorno de desarrollo integrado (IDE) de Java con todas las funciones. Los complementos de JDT proporcionan una interfaz de programación de aplicaciones (API) para que otros desarrolladores de herramientas puedan ampliarlos.

Los complementos de JDT se clasifican en:

JDT APT: Agrega al proyecto un soporte de procesamiento de anotaciones y ofrece:

- Soporte para ejecutar procesadores de anotaciones escritos para APT.
- Artefactos basados en anotaciones durante la construcción incremental.
- Marcadores de problemas para problemas basados en anotaciones.

JDT CORE: Define la infraestructura sin interfaz de usuario e incluye:

- Un constructor de Java incremental.
- Un modelo con una API para navegar por el árbol de elementos de Java.
- Un árbol de elementos de Java que define una vista centrada de un proyecto.
- Asistencia de código y ayuda de selección de código.
- Una infraestructura de búsqueda basada en índices.
- Apoyo a la evaluación.

DEPURACIÓN DE JDT: Implementa el soporte de depuración de Java independiente del idioma proporcionado por el depurador de la plataforma y proporciona:

- Lanzamiento de una máquina virtual Java en modo de ejecución o depuración.
- Adjuntar a una máquina virtual Java en ejecución.
- Evaluación de expresiones en el contexto de un marco de pila.
- Páginas de Scrapbook para la evaluación interactiva de fragmentos de código.
- Recarga dinámica de clases donde sea compatible con la máquina virtual Java.

TEXTO JDT: Proporciona al editor de Java las siguientes características:

- Coloración de sintaxis y palabras clave.
- Ayuda de código específico de contexto (Java, Javadoc) y selección de código.
- Método de edición de nivel.
- Anotaciones para problemas, puntos de quiebre o coincidencias de búsqueda.
- Actualización de Outliner a medida que se realiza la edición.
- La ayuda muestra la especificación Javadoc para el elemento Java seleccionado.
- La asistencia de importación organiza automáticamente las declaraciones.

- Formato de código.

JDT UI: Implementa contribuciones de entorno de trabajo específicas de Java:

- Explorador de paquetes.
- Vista de jerarquía de tipos.
- Vista de esquema de Java.
- Asistentes para crear elementos de Java.

Además, los usuarios pueden previsualizar los cambios individuales derivados de una operación de refactorización. También, el soporte de búsqueda implementa búsquedas precisas como encontrar declaraciones y/o referencias a paquetes, tipos, métodos y campos, con ámbito al área de trabajo, un conjunto de trabajo o la selección actual. Por otro lado, el soporte de comparación implementa una comparación estructurada de unidades de compilación de Java que muestra los cambios en los métodos individuales.

3.3.3.2. WINDOWS BUILDER

El entorno de WindowBuilder facilita en gran volumen la creación de interfaces gráficas de usuario (GUI) sin dedicar mucho tiempo al código. A partir del diseñador visual y las herramientas de diseño WYSIWYG (lo que ves es lo que tienes) se pueden configurar de manera simple hasta las ventanas más complejas ya que permite agregar de manera muy simple controles y controladores de eventos, cambiar las propiedades de los controles, etcétera.

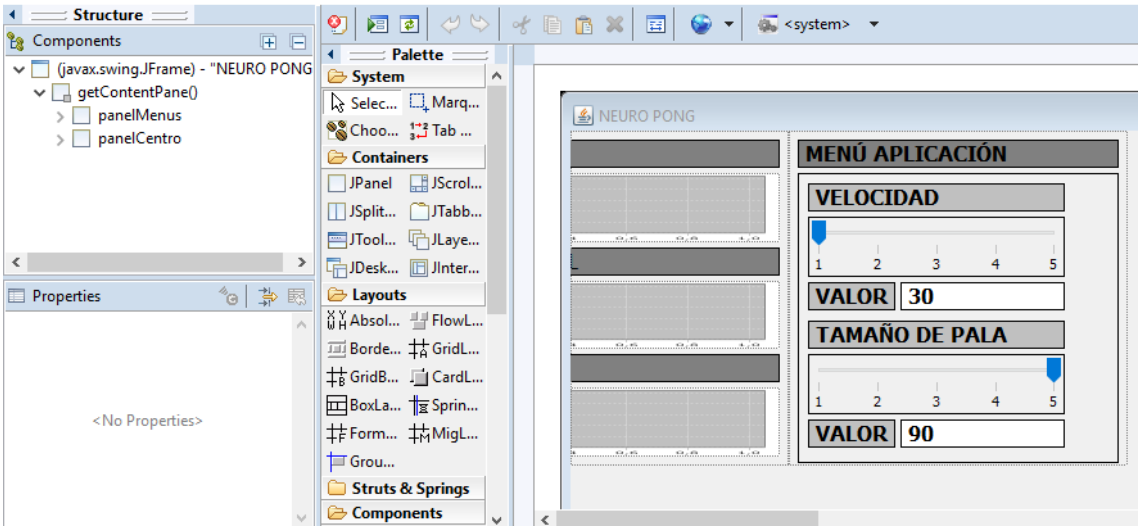

Ilustración 20. WindowBuilder.

Además, código generado no requiere bibliotecas personalizadas adicionales para compilar y ejecutar el programa, ya que el código generado se puede usar sin tener instalada esta expansión. También admite la edición de código de forma libre y la mayoría de las re-factorizaciones del usuario.

3.3.3.3. JFREECHART

JFreeChart es una biblioteca de gráficos 100% Java gratuita que facilita a los desarrolladores mostrar gráficos de calidad profesional en sus aplicaciones.

Características:

- Una interfaz de programación de apps que admite una amplia gama de gráficos.
- Un diseño flexible fácil de ampliar.
- Está orientada a aplicaciones tanto del servidor como del cliente.
- Compatibilidad con muchos tipos de salida.
- Compatibilidad con Swing, JavaFX, imágenes y gráficos vectoriales.

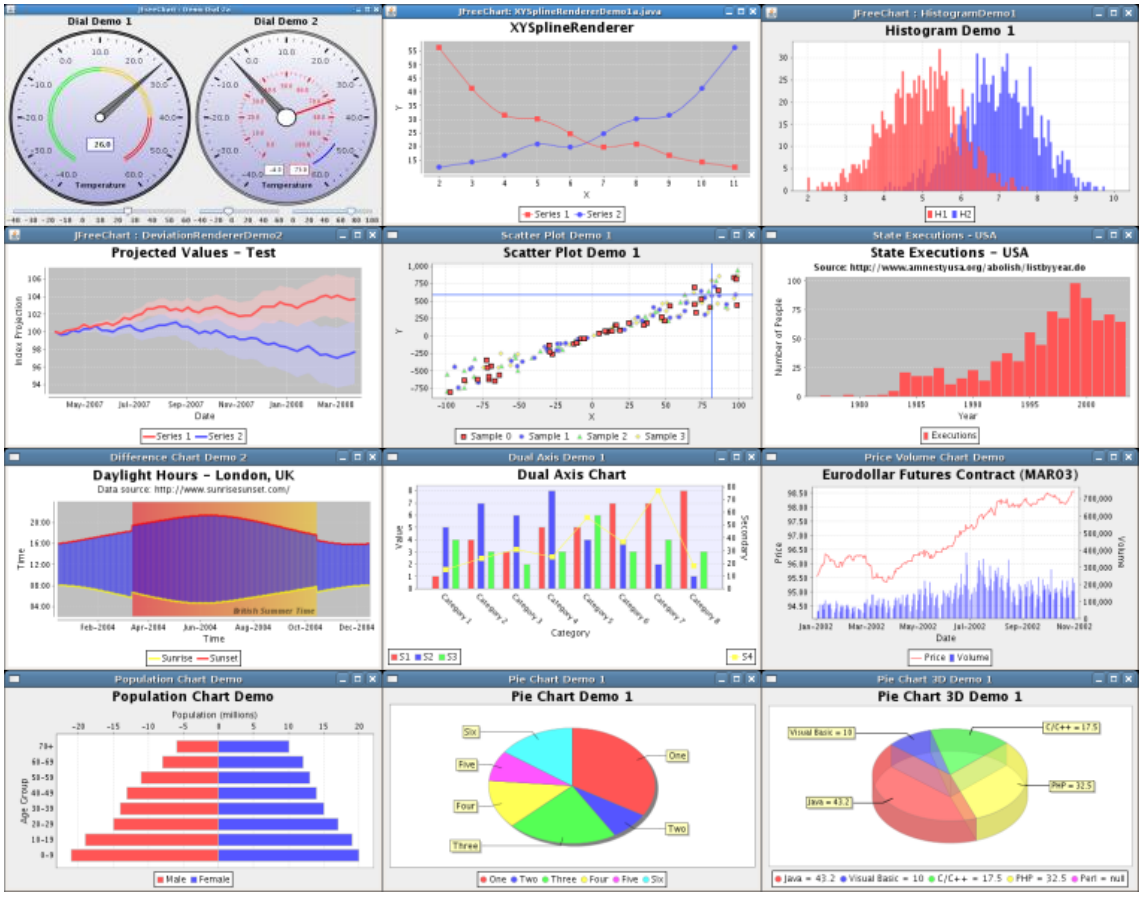

Ilustración 21. Tipos de gráficos en JFreeChart.

3.4. FASES DEL PROYECTO

3.4.1. FASE DE CAPTURA DE LA INFORMACIÓN

3.4.1.1. CONFIGURACIÓN DEL MÓDULO HC-08

El módulo HC-08 es un pequeño módulo que convierte la información que llega vía BT a puerta serie y viceversa. Éste se puede utilizar directamente para enviar y recibir mensajes a través del Bluetooth conectando sus pines serie RXD y TXD a los

correspondientes del Arduino y con el programa adecuado. Para entrar en el modo AT (modo de configuración) basta con arrancarlo mientras se mantiene en estado lógico HIGH el pin ENABLE, con el siguiente montaje:

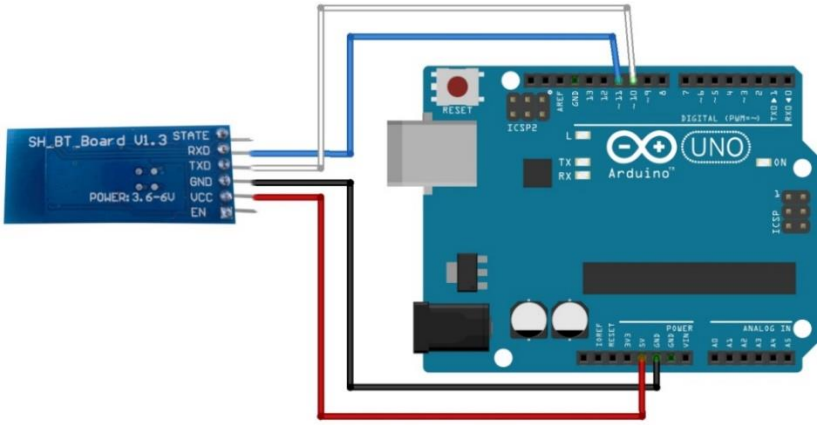

Ilustración 22. Diagrama de conexiones.

Y con el siguiente código se puede modificar la configuración del módulo en cuestión, para parametrizarlo de tal manera que se vincule con el casco Mindwave Mobile, y reciba los datos recogidos.

El citado código es el siguiente:

```
#include <SoftwareSerial.h>

SoftwareSerial BT1(4,2);

void setup ()
{
	Serial.begin(9600);
	Serial.println("Enter AT commands:");
	BT1.begin(9600);
}

void loop ()
{
	if (BT1.available())
	{
		Serial.write(BT1.read());
	}

	if (Serial.available())
	{
		char c = Serial.read();
		Serial.print(c);
		BT1.print(c);
	}
}
```

3.4.1.2. TABLAS DE COMANDOS AT

A continuación, se citan los comandos AT conocidos para el módulo HC-08:

OBTENER LA VERSIÓN DEL SOFTWARE		
Comando	Respuesta	Parámetro
AT	OK	Ninguno
AT+VERSION?	OK:<PARAM>	PARAM: Versión del software

OBTENER LA DIRECCIÓN DEL DISPOSITIVO		
Comando	Respuesta	Parámetro
AT+ADDR?	OK+ADDR:<PARAM>	PARAM: Dirección MAC

OBTENER/ESTABLECER EL ROL		
Comando	Respuesta	Parámetro
AT+ROLE<PARAM>	OK+SET:<PARAM>	0: Esclavo (POR DEFECTO) 1: Maestro 2: Slave loop
AT+ROLE?	OK+GET:<PARAM>	

OBTENER/ESTABLECER SI EL MÓDULO ARRANCA INMEDIATAMENTE		
Comando	Respuesta	Parámetro
AT+IMME<PARAM>	OK+SET:<PARAM>	0: Inmediato (POR DEFECTO) 1: Esperar
AT+IMME?	OK+GET:<PARAM>	

OBTENER/ESTABLECER NOTIFICACIONES		
Comando	Respuesta	Parámetro
AT+NOTI<PARAM>	OK+SET:<PARAM>	0: No (POR DEFECTO) 1: Notificar
AT+NOTI?	OK+GET:<PARAM>	
NOTA: Muestra OK + CONNECTED cuando se conecta un dispositivo.		

OBTENER/ESTABLECER LOS BAUDIOS		
Comando	Respuesta	Parámetro
AT+BAUD<PARAM>	OK+SET:<PARAM>	1: 1200 (No soportado) 2: 2400 (No soportado) 3: 4800 (No soportado) 4: 9600 (Por defecto) 5: 19200 6: 38400 7: 57600 8: 115200 9: 230400 (No soportado)
AT+BAUD?	OK+GET:<PARAM>	

OBTENER/ESTABLECER EL BIT DE PARIDAD		
Comando	Respuesta	Parámetro
AT+PARI<PARAM>	OK+SET:<PARAM>	0: Ninguno (Por defecto) 1: Par 2: Impar
AT+PARI?	OK+GET:<PARAM>	

DESCONECTAR LA CONEXIÓN BLUETOOTH		
Comando	Respuesta	Parámetro
AT+DISCON	Ninguna	Ninguno

OBTENER/ESTABLECER EL NOMBRE DEL DISPOSITIVO		
Comando	Respuesta	Parámetro
AT+NAME<PARAM>	OK+SET:<PARAM>	PARAM: Nombre Cómo máximo 11 caracteres. Por defecto: SH-HC-08
AT+NAME?	OK+NAME:<PARAM>	

OBTENER/ESTABLECER EL PIN		
Comando	Respuesta	Parámetro
AT+PASS<PARAM>	OK+SET:<PARAM>	PARAM: Código PIN Por defecto: 000000
AT+PASS?	OK+GET:<PARAM>	

OBTENER/ESTABLECER EL MODO DE EMPAREJAMIENTO		
Comando	Respuesta	Parámetro
AT+TYPE<PARAM>	OK+SET:<PARAM>	0: No necesita PIN. 1: Autenticación sin PIN. 2: Autenticación con PIN.
AT+TYPE?	OK+GET:<PARAM>	

OBTENER EL VALOR DE RSSI		
Comando	Respuesta	Parámetro
AT+RSSI?	OK+RSSI:<PARAM>	PARAM: Intensidad de señal

ESTABLECER EL MÓDULO EN MODO DURMIENTE		
Comando	Respuesta	Parámetro
AT+SLEEP	OK+SLEEP	Ninguno

OBTENER/ESTABLECER LA POTENCIA DEL MÓDULO		
Comando	Respuesta	Parámetro
AT+TXPW<PARAM>	OK+SET:<PARAM>	0: -23db 1: -6db 2: 0db (POR DEFECTO) 3: 4db
AT+TXPW?	OK+GET:<PARAM>	

OBTENER/ESTABLECER EL MODO DE FUNCIONAMIENTO		
Comando	Respuesta	Parámetro
AT+MODE<PARAM>	OK+SET:<PARAM>	0: Transparente (DEFECTO) 1: Transmisión directa 2: Difusión IBeacon
AT+MODE?	OK+GET:<PARAM>	

BUSCAR DISPOSITIVOS CERCANOS		
Comando	Respuesta	Parámetro
AT+SCAN	OK+SCANS:<PARAM>	PARAM: Nº de dispositivos

CONECTAR A UN DISPOSITIVO ENCONTRADO		
Comando	Respuesta	Parámetro
AT+CONN<PARAM1>	OK+CONN:<PARAM2>	PARAM1: El primer dispositivo será 0. El segundo dispositivo será 1. PARAM2: A: Conectado E: Error de conexión. F: Fallo de conexión

CONECTAR A UN DISPOSITIVO POR MAC		
Comando	Comando	Comando
AT+CONNADD	AT+CONNADD<PARAM>	PARAM: Dirección MAC PARAM2: A: Conectado E: Error de conexión. F: Fallo de conexión

REINICIAR EL DISPOSITIVO		
Comando	Respuesta	Parámetro
AT+RESET	OK+RESET	Ninguno

RESTABLECER VALORES DE FÁBRICA		
Comando	Respuesta	Parámetro
AT+RENEW	OK	Ninguno

CONFIGURAR LA VELOCIDAD DEL TRANSMISIÓN		
Comando	Respuesta	Parámetro
AT+UART<P1,P2,P3>	OK+ UART<P1,P2,P3>	P1: bps, 38400 bps (por defecto). P2: bit de parada. 0: 1 bit. 1: 2 bits. P3: bit de paridad. 0: Sin paridad. 1: Paridad impar. 2: Paridad par.

MODO DE CONEXIÓN		
Comando	Respuesta	Parámetro
AT+CMODE<PARAM>	OK+ CMODE<PARAM>	0: Bluetooth especificado. 1: Bluetooth disponible.

MODO DE TRABAJO		
Comando	Respuesta	Parámetro
AT+ROLE<PARAM>	OK+ROLE<PARAM>	0: Modo esclavo. 1: Modo maestro.

DIRECCIÓN MAC		
Comando	Respuesta	Parámetro
AT+BIND<PARAM>	OK+ BIND<PARAM>	PARAM: Dirección MAC.

3.4.1.3. IDENTIFICADOR ÚNICO DEL CASCO MINDWAVE

Para conocer el **identificador único** del casco y poder vincularlo al módulo de Bluetooth, es necesario acceder en el Panel de control de ordenador al apartado de Dispositivos e impresoras. Previamente, el casco ha tenido que ser vinculado a través de un adaptador Bluetooth USB al ordenador, para que éste pueda reconocerlo como tal en la ventana bajo el nombre de Dispositivos.

Este proceso es simple, ya que una vez conectado el adaptador, haciendo clic en la opción **Agregar Bluetooth u otro dispositivo** (esta opción aparece en Windows 10 una vez se ha entrado en la ventana de Dispositivos en la parte superior de ésta), el propio elemento buscará el casco para posteriormente vincularlo al ordenador, y que éste directamente lo reconozca cuando este visible como el objeto que es.

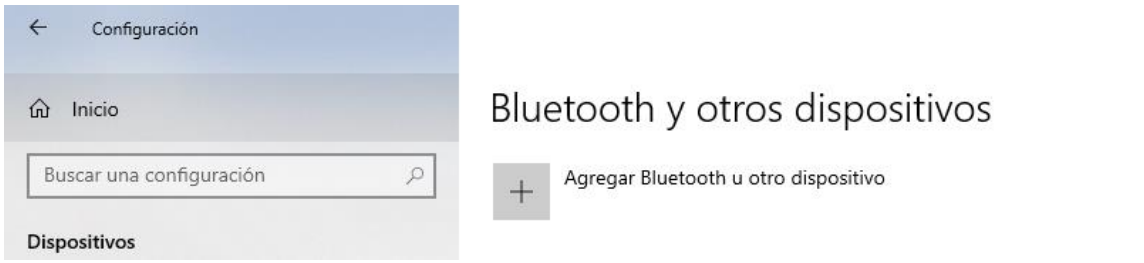

Ilustración 23. Bluetooth y otros dispositivos

Una vez realizado el paso anterior, en la ventana de Dispositivos e impresoras, aparecerá la opción de **Mindwave Mobile** como se muestra en la siguiente ilustración.

Ilustración 24. Dispositivos e impresoras.

A continuación, haciendo clic con el botón derecho sobre el icono de Mindwave Mobile, pinchando en la opción Propiedades, y dentro de la ventana emergente, en la pestaña de Bluetooth, se nos mostrará el **identificador único.**

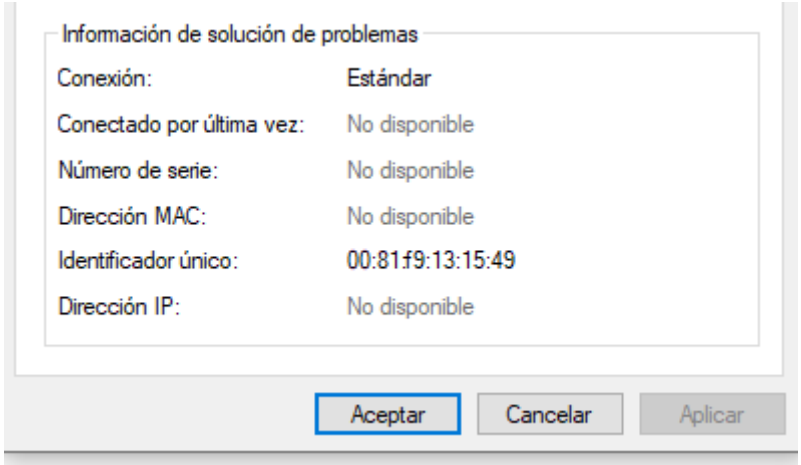

Ilustración 25. Propiedades. Mindwave Mobile.

Donde, para el casco Mindwave Mobile con el que se está trabajando, el **identificador único** tiene el siguiente valor: 00:81:F9:13:15:49. Una vez conseguido esta información, se puede proceder a vincular el casco con el módulo HC08 que se va a utilizar junto con Arduino, para conseguir los objetivos marcados en la **fase de captura de la información.**

3.4.1.4. SECUENCIA DE COMANDOS AT

Para vincular el casco Mindwave con el módulo Bluetooth HC-08 se deben introducir los siguientes comandos en el orden que se indica a continuación.

Para ello, haciendo uso del código citado en el apartado **"3.4.1. CONFIGURACIÓN DEL MÓDULO HC-08"** se escribe lo siguiente en la ventana del Serial Reader:

1. Primero se configura el modo de trabajo del módulo, la contraseña, y la velocidad

   AT + UART = "57600,0,0"

   AT + ROLE = "1"

   AT + PASS = "1234"

   AT + CMODE = "0"

   AT + BIND = "0081,F9, 131549"

2. Después, haciendo uso de unos comandos AT adicionales a los citados en el apartado "**3.4.1. TABLAS DE COMANDOS AT"**, se termina de configurar el módulo de tal manera que solamente se conecte a una dirección MAC específica.

   AT + INIT (Abrir un puerto serie para enviar datos en ambos sentidos).

   AT + IAC = 9E8B33 (Código de acceso del Bluetooth).

   AT + CLASS = 0 (Examinar los dispositivos Bluetooth (por defecto 0)).

   AT + PAIR = 0081,F9,131549,20 (Dirección MAC y tiempo límite de conexión).

   AT + LINK = 0081,F9,131549 (Dirección del dispositivo Bluetooth remoto).

   AT + INQM = 1,9,48

   - Indica la intensidad de señal RSSI.
   - No pregunta si responden más de 9 dispositivos.
   - TIEMPO LIMITADO DE PREGUNTA: 48 · 1,28 = 61,44s

   AT + INQ (Responde con 3 parámetros).

   - Dirección Bluetooth.

- Tipo de dispositivo.
- Intensidad de la señal RSSI.

NOTA: Para usar los comandos AT anteriores, no hay que hacer uso de las comillas; es decir, se han de introducir los datos intermedios después del comando AT. Después del final de cada comando se ha de presionar **enter**.

Configuración del módulo HC-08:

- El módulo HC-08 se configura como dispositivo maestro.
- El casco Mindwave Mobile actuará como dispositivo esclavo.
- El módulo HC-08 será par con sólo un dispositivo específico.
- El casco Mindwave Mobile y el Módulo HC-08 se emparejarán automáticamente, cuando ambos dispositivos sean encendidos.

3.4.1.5. PROTOCOLO DE COMUNICACIÓN DE MINDWAVE MOBILE

El protocolo de comunicación del casco está implementado en el chip ThinkGear, encargado de atenuar el ruido ambiental para poder realizar mediciones en áreas no aisladas eléctricamente. Es el sensor FP1 el que recoge la información y la envía al chip, donde es procesada y convertida en una secuencia de datos digitales. A continuación, una vez filtradas las interferencias, las ondas cerebrales se amplifican y son comparadas con los algoritmos codificados en la memoria del chip ThinkGear.

3.4.1.5.1. POOR_SIGNAL

Se trata de un byte sin signo que describe calidad de la señal medida por el ThinkGear, cuyo rango de valores varía entre 0 y 200.

Cuantitativamente:

- Cualquier valor diferente de cero representa que hay ruido.
- Si el valor es 200, quiere decir que no existe conexión entre el sensor y la piel.

Esta información es enviada con un periodo de un segundo, está activa por defecto y su frecuencia de muestreo es 1 Hz.

Los factores que pueden modificar la calidad de la señal son:

a) Dependen del usuario:

- Los contactos del sensor o la masa no están en contacto con la frente o la oreja.
- Contacto entre el sensor y la piel pobre.
- El movimiento excesivo.
- Nivel de batería del dispositivo.

b) Dependen del ambiente en el que se está trabajando:

- El ruido electroestático ambiental.
- El ruido generado por otras medidas bioeléctricas: EMG, EoG…
- Interferencias de otro tipo de ondas.

Sin embargo, tal y como se ha citado, el chip ThinkGear dispone de una secuencia de filtrado además de un algoritmo para la medición de los valores de atención y meditación, los cuales permiten detectar, corregir, compensar y tolerar muchos errores.

3.4.1.5.2. ATTENTION Y MEDITATION ESENSE.

Cada uno de los datos viene representado por un byte de información. Estos se representan en una escala de 0 a 100. Un valor del medidor de eSense de 0 es un valor especial que indica que ThinkGear no puede calcular un nivel de eSense con una confiabilidad razonable, situación que suele producirse normalmente al principio, ya que el dispositivo se encuentra en fase de calibración debido al ruido excesivo.

Por otro lado, los rangos para cada interpretación son relativamente amplios ya que algunas partes del algoritmo eSense están aprendiendo dinámicamente, y en ocasiones emplean algunos algoritmos de "adaptación lenta" para ajustarse a las fluctuaciones y tendencias naturales de cada usuario. Es por esto último por lo cual los sensores ThinkGear pueden operar ser utilizados en una amplia gama de personas con una gama extremadamente amplia de condiciones personales y ambientales, ofreciendo al mismo tiempo una buena precisión y fiabilidad.

Cuantitativamente:

- Un valor entre 1 y 20 indica niveles "muy bajos" de eSense, mientras que un valor entre 20 y 40 indica niveles “reducidos" de eSense. Estos niveles pueden indicar estados de distracción, agitación o anormalidad.
- Un valor entre 40 y 60 en cualquier momento dado en el tiempo se considera neutral y es similar en noción a las "líneas de base" establecidas en las técnicas de medición de EEG convencionales.
- Un valor de 60 a 80 se considera "ligeramente elevado", y puede interpretarse como niveles posiblemente más altos de lo normal.
- Los valores de 80 a 100 se consideran "elevados", lo que significa que son indicativos de niveles elevados de ese eSense.

Los valores de “attention” y “meditation” se actualizan con una frecuencia de un segundo. Esto es consecuencia de que el algoritmo que realiza estos cálculos cuenta con un aprendizaje dinámico para ajustarse a cada persona.

3.4.1.5.3. RAW WAVE VALUE

Esta información viene representada dos bytes (16 bits) y representa una única muestra de la señal bruta conocida como “RAW”. Éste se trata de un dato con valor entero con signo, dentro de un rango acotado desde -32768 a 32767, donde:

- El primer byte representa la parte alta.
- El segundo byte la parte baja.

Por defecto, al igual que el caso de POOR_SIGNAL, este valor también se encuentra siempre activo y se recoge 512 veces por segundo, lo que supone que la diadema Mindwave capte una muestra cada 2ms.

3.4.1.5.4. ASIC_EEG_POWER

Este dato 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.

BIG ENDIAN:

Este formato ordena los bytes del más significativo al menos significativo, que puede parecer una forma más "natural" de escritura y es utilizado por procesadores usados en máquinas Apple entre otras.

Los datos 13 y "trece" expresados en hexadecimal son 0x3133 y 0x74726563650d0a, pues bien, escritos como datos de 2 bytes en formato Big-endian quedan:

```
0x31 0x33
0x74 0x72 0x65 0x63 0x65 0x0d 0x0a
```

LITTLE ENDIAN:

Este formato es adoptado por la mayoría de procesadores Intel.

Los mismos datos como datos de 2 bytes en formato Little-endian quedan:

```
0x33 0x310x0a 0x0d 0x65 0x63 0x65 0x72 0x74
```

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)

Cabe destacar que, esta información no tiene unidades y que solamente tiene sentido comparándola con los demás datos o entre ellos mismos. Ésta se recoge con una frecuencia de una vez por segundo y están activados por defecto.

3.4.1.5.5. BLINK STRENGTH

Este valor indica la intensidad del parpadeo de ojos. Es un byte cuyo rango es de 1 a 256 y que se recoge cada vez que existe un parpadeo. El dato se actualiza cada segundo.

3.4.1.6. PAQUETES THINKGEAR

La diadema Mindwave Mobile utilizada envía la información a través de un flujo de bytes en formato serie asíncrono. Es por ello que se considera importante y necesario conocer la naturaleza de dicha información, y cómo está codificada. Para ello, se ha de conocer el formato de los conocidos como "ThinkGear Packets", con el fin de extraer e interpretar la información transmitida.

Estos paquetes se utilizan para enviar los datos desde un módulo ThinkGear a un receptor arbitrario (un PC, otro microprocesador o cualquier otro dispositivo que pueda recibir un flujo de bytes en serie).

Además, el formato de estos está diseñado para ser robusto y flexible:

- El encabezado y el checksum del flujo de datos proporcionan una sincronización y una posibilidad de verificación de la integridad de datos.
- El formato de PAYLOAD asegura que se puedan agregar nuevos campos de datos al paquete en el futuro sin volver obsoleto los analizadores de paquetes en las aplicaciones.

Esto último significa que cualquier aplicación que implemente un analizador de paquetes ThinkGear correctamente podrá usar modelos más nuevos de módulos ThinkGear probablemente sin tener que cambiar sus analizadores o aplicaciones. Los citados "ThinkGear Packets" se mandan a través de un medio de transporte UART, puerto serie COM, USB, Bluetooth, a través de ficheros o cualquier otro mecanismo capaz de transportar bytes.

Un paquete ThinkGear es un formato de paquete que consta de 3 partes:

- Encabezado de paquete.
- Paquete de carga útil.
- Suma de comprobación de la carga útil.

```
[SYNC] [SYNC] [PLENGTH] [PAYLOAD ...] [CHECKSUM]
_____________________________ _______________ _____________
^^^^^^^^(Header)^^^^^^^ ^^(Payload)^^ ^(Checksum)^
```

La sección [PAYLOAD] puede ser de hasta 169 bytes de longitud, mientras que las secciones de [SYNC], [PLENGTH] y [CHKSUM] están formados por un único byte. Esto significa que un paquete completo y válido tiene una longitud mínima de 4 bytes (si la carga útil de datos está vacía) y una longitud máxima de 173 bytes (si la carga útil de datos tiene una longitud máxima de 169 bytes).

3.4.1.7. ENCABEZADO DE PAQUETE

El encabezado de un paquete consta de 3 bytes:

- [SYNC] Byte de sincronización (0xAA).
- [SYNC] Byte de sincronización (0xAA).
- [PLENGTH] Un byte que representa la longitud de carga útil del paquete.

```
[SYNC] [SYNC] [PLENGTH]
_____________________
^^^^^^^(Encabezado)^^^^^^
```

Los dos bytes[SYNC] se utilizan para señalar el comienzo de un nuevo paquete y son bytes con el valor 0xAA (en formato decimal 170). La sincronización tiene una longitud de dos bytes en lugar de uno solo, con el objetivo de reducir la posibilidad de que los bytes que se producen dentro del paquete puedan confundirse con el comienzo de un paquete.

Sin embargo, aun así, cabe la posibilidad de que esta situación se dé en una trama; por ello se utilizan los bytes de [PLENGTH] y [CHKSUM] combinados para asegurar que el paquete “mal sincronizado" nunca se interprete accidentalmente como un paquete válido. El byte [PLENGTH] indica la longitud en bytes de la sección de carga útil de datos del paquete [PAYLOAD] la cual está acotada dentro del rango desde 0 hasta 169. Es por ello que cualquier valor superior supone un error “PLENGTH TOO LARGE”. Además, es importante recordar que el valor almacenado en [PLENGTH] representa la longitud de la información útil del paquete, no de todo el paquete. La longitud completa del paquete siempre será [PLENGTH] + 4.

3.4.1.8. SUMA DE COMPROBACIÓN DE LA CARGA ÚTIL (CHECKSUM).

El byte [CHKSUM] debe usarse para verificar la integridad de la carga útil de datos del paquete. La suma de comprobación de la carga útil se calcula de la siguiente manera:

- Sumar todos los bytes de la carga útil de datos del paquete.

- Tomar los 8 bits más bajos de la suma.
- Realizar el inverso de bits (el complemento de uno) en los 8 bits más bajos.

El dispositivo que reciba un paquete debe utilizar esos tres pasos para calcular el Checksum para la carga útil de datos recibida, y luego compararlo con el byte [CHKSUM] recibido con el paquete.

- Si el checksum calculado y los valores de [CHKSUM] recibidos no coinciden, todo el paquete deberá ser descartado y considerarlo como no válido.
- Si el checksum calculado y los valores de [CHKSUM] recibidos coinciden, es cuando el dispositivo puede hacer uso de dicha información, y analizarla.

3.4.1.8. ESTRUCTURA DE CARGA DE DATOS

Una vez que se haya verificado el checksum de un paquete, se pueden analizar los bytes de la carga de datos [PAYLOAD]. La carga de datos está formada por una serie de valores, cada uno contenido en una serie de bytes llamada DATAROW. Cada DATAROW contiene información sobre lo que representa el valor de los datos, la longitud del valor de los datos y los bytes del propio valor de los datos. Por lo tanto, para analizar una carga útil de datos, se debe analizar individualmente cada DATAROW hasta que todos los bytes de la carga útil de datos hayan sido tratados.

3.4.1.9. FORMATO DE UN DATAROW

Un DATAROW consta de bytes en el siguiente formato:

```
([EXCODE]...) [CODE]   ([VLENGTH])   [VALUE...]
_____________________ _____________ ___________
^^^(Tipo de valor)^^^ ^^(longitud)^^ ^^(valor)^^
```

NOTA: Los bytes entre paréntesis son condicionales, lo que significa que solo aparecen en algunos DATAROWS y no en otros.

Un DATAROW puede empezar con cero o más bytes de tipo [EXCODE], que son bytes con el valor hexadecimal de 0x55. Este número indica el nivel de código extendido que se usa junto con el byte [CODE] para determinar la naturaleza del dato que contiene el DATAROW. Por lo tanto, los analizadores deben comenzar a analizar un DATAROW contando el número de [EXCODE] que aparecen para determinar el nivel de código.

- Si el valor del byte [CODE] está acotado entre 0x00 y 0x7F, se supone que el byte [VALUE] tiene una longitud de 1 byte. En este caso, no hay un byte [VLENGTH], por lo que el byte [VALUE] aparecerá inmediatamente después del byte [CODE].

- Si el valor del byte [CODE] es mayor que 0x7F, entonces un byte [VLENGTH] sigue inmediatamente al byte [CODE], y este es el número de bytes en [VALUE]. Estos CÓDIGOS más altos son útiles para transmitir matrices de valores, o valores que no pueden ajustarse en un solo byte.

NOTA: El formato DATAROW se define de esta manera para que el analizador implementado no se descomponga en el futuro si se agregan nuevos valores de [CODE].

3.4.1.10. TABLA DE DEFINICIONES DE [CODE] PARA CÓDIGOS DE UN BYTE

A continuación, se adjunta una tabla que recoge los [CODE] de un solo byte:

```
0 0x02 - Calidad de SEÑAL POBRE (0-255)
0 0x03 - HEART_RATE (0-255) Una vez / s en EGO.
0 0x04 - ATENCIÓN eSense (0 a 100)
0 0x05 - MEDITATION eSense (0 a 100)
0 0x06 - 8BIT_RAW Valor de onda (0-255)
0 0x07 - Inicio de la sección RAW_MARKER (0)
```

3.4.1.11. TABLA DE DEFINICIONES DE [CODE] CÓDIGOS MULTI-BYTE

A continuación, se adjunta una tabla que recoge los [CODE] de un multi-byte:

```
0 0x80 2 RAW Wave Value: un solo big-endian.
```

- Valor firmado de dos bits de 16 bits
- Byte de orden superior seguido de byte de orden inferior (-32768 a 32767).

```
0 0x81 32 EEG_POWER: ocho big-endian de 4 bytes.
```

- Valores de punto flotante representando delta, theta, alpha-low, alfa-high, beta-low, beta-high.
- Banda de EEG de gama baja y media.

```
0 0x83 24 ASIC_EEG_POWER: ocho big-endian.
```

- Valores sin signo de 3 bytes representando delta, theta, alpha-low, alfa-high, beta-low, beta-high.
- Banda de EEG de gama baja y media.

```
0 0x86 2 RRINTERVAL: big-endian de dos bytes sin firma.
```

- Entero que representa los milisegundos dos picos R.

```
Cualquier 0x55 - NUNCA USADO (reservado para [EXCODE]).
Cualquier 0xAA - NUNCA UTILIZADO (reservado para [SYNC]).
```

3.4.1.12. ANALISIS UN PAQUETE DE INFORMACIÓN

A continuación, se cita el proceso a seguir para analizar un paquete de datos:

1. Leer los bytes de la secuencia hasta que se encuentre un byte [SYNC]: (0xAA).
2. Leer el siguiente byte y asegurarse de que también es un byte [SYNC]: (0xAA).

Si no es un byte [SYNC], vuelve al paso 1; de lo contrario, continúe con el paso 3.

3. Leer el siguiente byte de la secuencia como [PLENGTH].

Una vez se tiene el valor de [PLENGTH]:

- ❖ Si [PLENGTH] es 170, se ha de repetir el paso 3.
- ❖ Si [PLENGTH] es mayor que 170, luego vuelva al paso 1 (PLENGTH TOO LARGE).
- ❖ De lo contrario, continuar con el paso 4.

Después:

4. Leer los bytes [PLENGTH] de la secuencia [PAYLOAD],
5. Guardarlos en un área de almacenamiento (matriz unsigned char Payload [256]).
6. Sumar cada byte a medida que se lee (checksum += byte).
7. Coger los 8 bits LSB del acumulador del checksum y se invierten.

```
checksum & = 0xFF;
checksum = ~ checksum & 0xFF;
```

8. Lea el siguiente byte de la secuencia como el byte [CHKSUM].
   - ❖ Si el [CHKSUM] no coincide con el checksum calculado (CHKSUM FAILED).
   - ❖ Si el [CHKSUM] coincide se puede analizar el contenido de la carga útil.
9. En cualquier caso, vuelva al paso 1.

3.4.1.13. ANALISIS DE LOS DATAROWS DE LA CARGA ÚTIL DE UN PAQUETE

A continuación, se cita el proceso a seguir para analizar la carga útil de un paquete:

1. Analizar y contar el número de bytes [EXCODE] al principio del DATAROW actual.
2. Analizar el byte [CODE].
3. Si [CODE] ≥ 0x80, analizar el siguiente byte como el byte [VLENGTH].
4. Analizar el byte [VALUE] basado en el DATAROW: [EXCODE] [CODE] [VLENGTH].
5. Si no se han analizado todos los bytes del PAYLOAD, se volverá al paso 1.

3.4.2. FASE DE ANÁLISIS Y PROCESADO DE LA INFORMACIÓN

3.4.2.1. CAPTURA DE LA SEÑAL RAW EN MATLAB

El casco Mindwave Mobile, proporciona información acerca de la señal cerebral "cruda" (RAW). Esta señal es la que posteriormente se divide en las ondas cerebrales: alpha, theta, gamma, beta. En este caso, el casco es capaz de capturar información sobre éstas individualmente, además de los estados mentales de atención y meditación, y el parpadeo del usuario.

Sin embargo, para el caso en el que se está trabajando, únicamente es necesaria la información acerca del valor de dicha señal cerebral cruda, que a partir de este momento se citará como "RAW". Por tanto, se creará un script en MatLab que sea capaz de aislar dicho dato (RAW) de la trama enviada por el casco, y que además haga una representación temporal de esta información.

Para evitar un exceso de código, se ha hecho uso de la librería "thinkgear.dll", la cual permite extraer de la trama los siguientes valores:

- TG_DATA_POOR_SIGNAL
- **TG_DATA_ATTENTION**
- **TG_DATA_MEDITATION**
- **TG_DATA_RAW**
- TG_DATA_DELTA
- TG_DATA_THETA
- TG_DATA_ALPHA1 y TG_DATA_ALPHA2
- TG_DATA_BETA1 y TG_DATA_BETA2
- TG_DATA_GAMMA1 y TG_DATA_GAMMA2
- TG_DATA_BLINK_STRENGTH
- TG_BAUD_57600
- TG_STREAM_PACKETS

Para ello, a partir del siguiente código se carga la citada librería en el proyecto:

```
%CARGAR THINKGEAR.DLL

    loadlibrary('Thinkgear.dll');
    fprintf('Se ha cargado la librería: ThinkGear.dll\n');
    dllVersion = calllib('Thinkgear', 'TG_GetDriverVersion');
    fprintf('Versión de "Thinkgear.dll": %d\n', dllVersion);
```

También se establecerá el valor de la velocidad de transmisión:

```
%VELOCIDAD

    TG_BAUD_57600 = 57600;
```

Como se ha comentado en el apartado 3.4.3.4. FORMATO DE UN DATAROW:

```
%FORMATO DE DATOS PARA TG_CONNECT () AND TG_SETDATAFORMAT ()

    TG_STREAM_PACKETS = 0;
```

```
%TIPO DE DATOS QUE DEVUELVE 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;
```

Por tanto, partiendo de que el análisis de la trama ya está implementado, se generará un archivo ".m" que se conecte al casco y que grafique la información extraída.

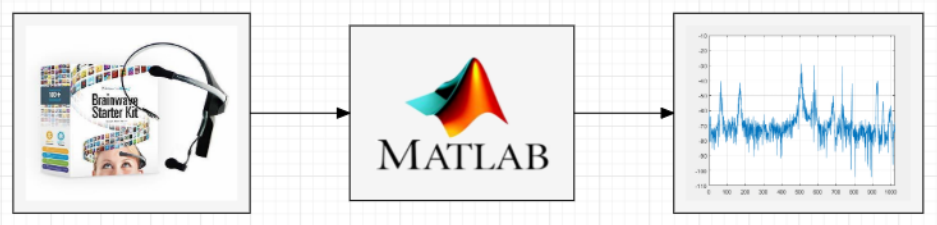

Ilustración 26. Diagrama del proyecto en MatLab.

3.4.2.2. CONEXIÓN DEL CASCO MINDWAVE A MATLAB

En primer lugar, es necesario realizar una conexión entre el casco Mindwave y el entorno de MatLab. Para ello, a partir de la apertura de un puerto serie, se establecerá un medio de comunicación bidireccional entre ambos elementos.

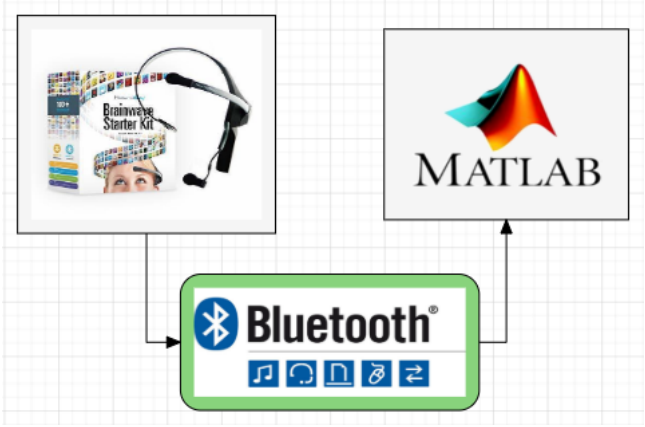

Ilustración 27. Conexión del casco a MatLab.

Para ello, de la misma manera que en apartado "3.4.1.2. IDENTIFICADOR ÚNICO DEL CASCO MINDWAVE" se accederá a las propiedades del casco en el **Panel de control**, y en la pestaña **Servicios**, se obtendrá el puerto COM al que está vinculado el casco. Una vez en la pestaña citada, en el apartado de **Servicios de Bluetooth** aparecerá indicado el puerto serie al que está enchufado el adaptador Bluetooth del casco.

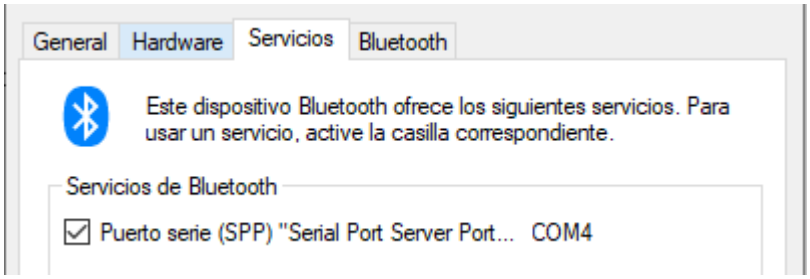

Ilustración 28. Puerto COM del casco.

En el script, para simplificar el trabajo con éste se creará una variable que representará el puerto COM al que está conectado el casco, con el objetivo de únicamente tener que modificar un valor si el casco es conectado a otro puerto en otra sesión.

```
portnum1 = 4;
comPortName1 = sprintf('\\\\.\\COM%d', portnum1);
```

Después, a partir de la función **TG_GetNewConnectionId** se obtiene el identificador de la conexión, necesario para establecer la conexión entre el casco y el programa.

```
connectionId1 = calllib('Thinkgear', 'TG_GetNewConnectionId');

if (connectionId1 < 0)
    ERROR(sprintf('ERROR: TG_GetNewConnectionId %d.\n', connectionId1));
end
```

Una vez conocido el **connectionId1** se procede a configurar el puerto:

```
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
```

Una vez ejecutado estas líneas de código, el puerto quedará configurado de tal manera que MatLab pueda trabajar con la información que se recibe.

3.4.2.3. LECTURA Y REPRESENTACIÓN DE LOS DATOS

Una vez se ha establecido la conexión, haciendo uso de la librería “thinkgear.dll”, se extraen de la trama los valores de **atención**, **meditación** y **raw** en una misma ventana.

```
if(calllib('Thinkgear','TG_GetValueStatus',connectionId1,TG_DATA_RAW)~=0)

    j = j + 1;
    data_att(j,1) =
    calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_ATTENTION);
    data_att(j,2) =
    calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_MEDITATION);
    raw_att(j,1) =
    calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_RAW);
```

```
        if(j < 70) xmin = 0; xmax = 69;
        else xmin = j - 70; xmax = j;
        end

        % VER MEDITACIÓN - ATENCIÓN - ENCEFALOGRAMA:

        subplot(3,1,3), plot(data_att(:,2)), title('Meditación');
        subplot(3,1,2), plot(data_att(:,1)), title('Atención');
        subplot(3,1,1), plot(raw_att(:,1)), title('Encefalograma');
        drawnow;

    end
```

Mediante esta representación se han podido comprobar detalles:

- El casco necesita un proceso de estabilización para poder calcular los valores de atención y meditación a partir de la señal bruta.
- La pinza que representa el punto de referencia, ha de estar bien conectada, ya que si no se añadirá mucho ruido a la señal captada por el casco.
- Los parpadeos añaden ruido a la señal y son fácilmente detectables visualmente, lo que supone que, a la hora de detectarlos, el algoritmo se simplifica.

En cuanto a los valores calculados a partir el algoritmo eSense, el desarrollador cita:

- Los valores de atención se pueden controlar centrándose en una idea singular:
  - Elegir un punto de la pantalla y fijar la mirada.
  - Identificar y mantener un pensamiento concreto.
  - Enfocar la mirada en un objeto concreto.
  - Concentrarte en algo que te gusta.
  - Hacer un cálculo matemático.
  - Dar un discurso.
  - Cantar una canción silenciosamente.
  - Imaginar una acción que estas tratando de que se cumpla.
- Los valores de atención se pueden controlar centrándose en una idea singular:
  - Respirar y exhalar lento y profundo.
  - Relajar todos los músculos.
  - Dejar la mente en blanco.
  - Cerrar los ojos.
  - Imaginar que estás en la cama a punto de irte a dormir.
  - Imaginar que estás flotando sobre el agua caliente.

3.4.2.4. DETECCIÓN DE LOS PARPADEOS

Como se ha citado en el anterior apartado, a partir del código adjuntado, se consiguió registrar en una gráfica el valor de la señal raw, mezclada con el ruido generado por los parpadeos; por ello, con la siguiente modificación de dicho código, se puede representar en una nueva gráfica (dentro de la anterior ventana) una señal que represente con un 1 cuando el valor de la señal RAW supere un valor umbral, y con un 0 cuando el valor de RAW esté por debajo de dicho valor umbral.

```
if (calllib('Thinkgear','TG_GetValueStatus',connectionId1,TG_DATA_RAW)~=0)

    j = j + 1;
    data_att(j,1) =
    calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_ATTENTION);

    data_att(j,2) =
    calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_MEDITATION);

    raw_att(j,1) =
    calllib('Thinkgear','TG_GetValue',connectionId1,TG_DATA_RAW);
    if raw_att(j,1) > consigna cntrl_1 = 1;
    else cntrl_1 = 0;
    end

    if raw_att(j,1) < consigna cntrl_2 = 1;
    else cntrl_2 = 0;
    end

    if(j < 70)

          xmin = 0;
          xmax = 69;

    else

          xmin = j - 70;
          xmax = j;

    end

    control_1(j,1) = cntrl_1; %(ASCENDENTE)
    control_2(j,1) = cntrl_2; %(DESCENDENTE)

    % Ver Meditación - Atención - Encefalograma:

          subplot(2,2,3), plot(control_1(:,1)), title('Tren del pulsos');
          subplot(2,2,4), plot(data_att(:,2)), title('Meditación');
          subplot(2,2,2), plot(data_att(:,1)), title('Atención');
          subplot(2,2,1), plot(raw_att(:,1)), title('Encefalograma');
          drawnow;
end
```

El parpadeo no es un estado mental o una onda cerebral con un umbral de frecuencias característico. Éste es un potencial generado por una acción muscular; es decir, una señal detectada por el electrodo del casco producida por la contracción del músculo frontal. Este potencial se mezcla con la señal raw, produciendo un pico en la señal muy característico; donde cuanto mayor sea la contracción muscular de mayor valor serán los picos en la señal.

De esta manera, se pueden establecer aproximadamente unos umbrales para acotar los valores que alcanza la señal RAW en consecuencia de un parpadeo:

UMBRALES DE PARPADEO EN VALOR ABSOLUTO	
Parpadeo involuntario	0 a 599

Parpadeo forzado	**600 a 1499**
Parpadeo muy forzado	**Valores mayores que 1500**

Tabla 3. Umbrales de parpadeo

Como se puede observar en la tabla, para el caso de aplicación con el que se está trabajando, interesa únicamente hacer uso de aquellos valores que sean superiores a 600 de la señal RAW; es decir, aquellos que representan un parpadeo forzado. Por ello, la consigna a establecer deberá ser de 599.

```
if raw_att(j,1) > consigna cntrl_1 = 1;
else cntrl_1 = 0;
end

if raw_att(j,1) < consigna cntrl_2 = 1;
else cntrl_2 = 0;
end
```

3.4.3. FASE DE IMPLEMENTACIÓN

3.4.3.1. CREACIÓN DE UNA APLICACIÓN EN JAVA

A partir de este momento, en el que se ha analizado la información obtenida del casco, y separado la información de los parpadeos de la señal RAW, se decidió implementar una aplicación Java que fuera capaz de representar dicha información en diversas gráficas; y, además, utilizar la señal de control generada a partir de los parpadeos como máquina de estados de un juego.

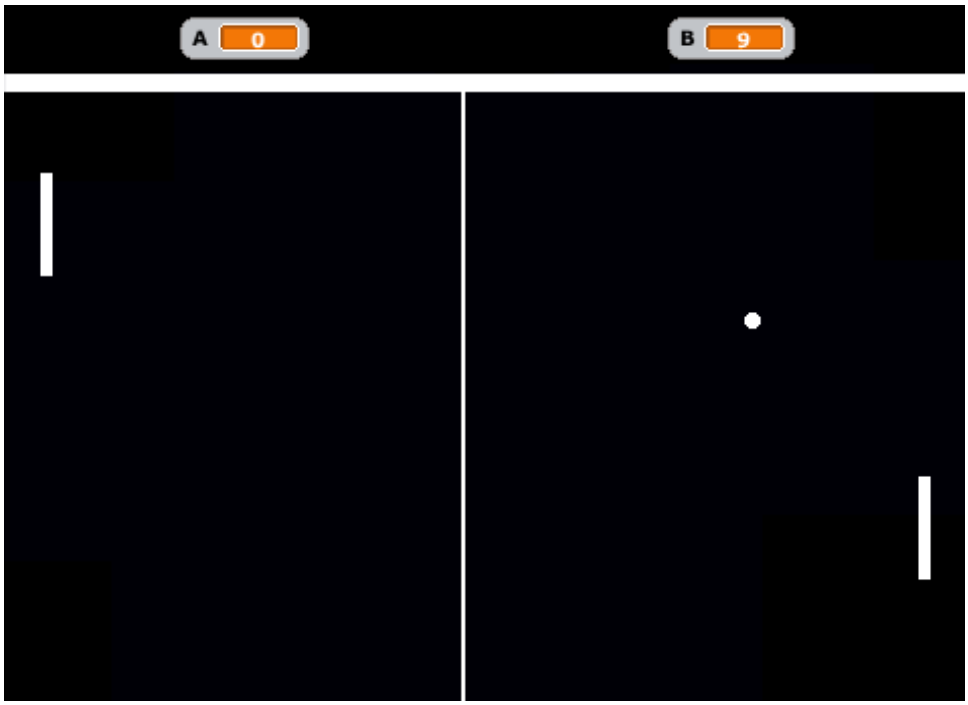

Ilustración 29. Juego a controlar con la señal de control

La intención con respecto al juego es ser capaz de variar la dirección de movimiento de la pala del tablero con los parpadeos, generando la citada máquina de estados: arriba y abajo, y moviendo la barra con la barra espaciadora del teclado.

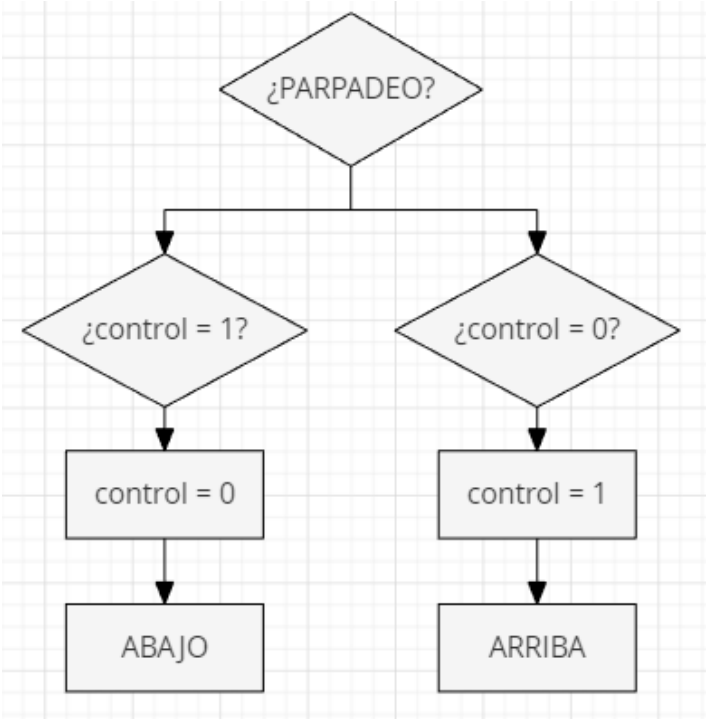

Ilustración 30. Gestión de la señal de control en el juego

3.4.3.2. DESARROLLO DE LA APLICACIÓN

La aplicación desarrollada se ha creado utilizando el lenguaje de programación Java dentro del entorno de desarrollo Eclipse. Como se ha mencionado, esta aplicación recoge y grafica la información recibida del casco Mindwave Mobile y después de procesarla permite el control mediante parpadeos del juego NeuroPong.

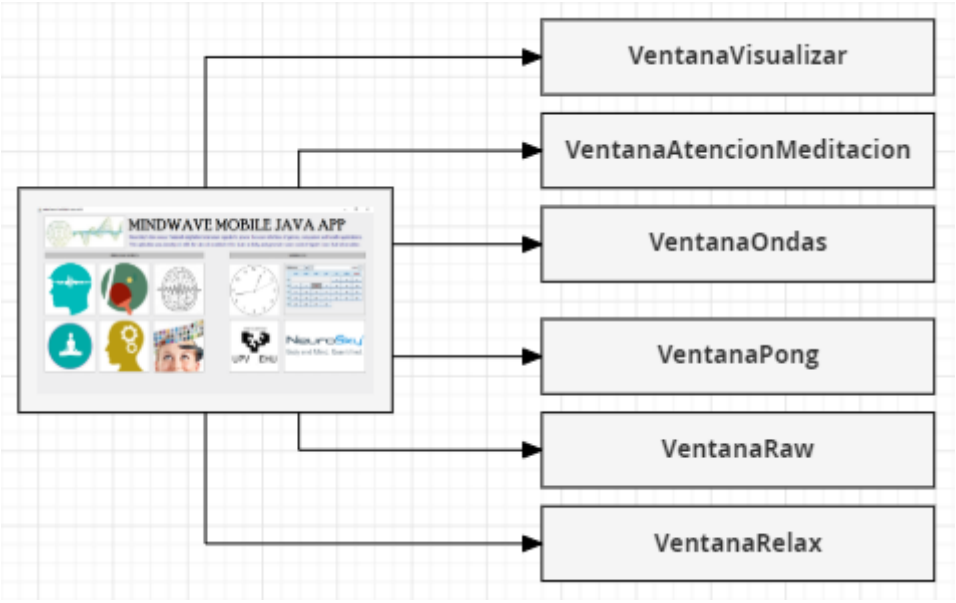

Ilustración 31.Menú principal de la aplicación.

Donde:

- Se ha realizado un programa en Java que permite al usuario obtener una representación en tiempo real de la información cerebral capturada por el electrodo del casco.

- Además, el programa permite al usuario interaccionar con un pequeño juego que utiliza el algoritmo desarrollado en Matlab para la detección de los parpadeos. El objetivo de este juego es hacer uso de un estímulo captado por el casco, y darle una utilidad, como en este caso es el modificar la dirección de movimiento de la pala de un juego.

- También se hace un seguimiento del estado de atención del usuario mientras hace uso del juego, monitorizando los valores obtenidos, además de guardar los resultados en un fichero.

- Por otro lado, la aplicación permite al usuario representar los datos de un fichero gráficamente, para poder hacer un estudio de estos posterior a su recolección, lo que facilita el análisis de la información del casco de manera cualitativa y cuantitativa.

Todo esto supone que la interfaz simplifica todo el proceso de análisis de la información que se ha llevado a cabo durante todo el proyecto, además de dar utilidad en forma de máquina de estados a una de las variables captadas.

3.4.3.3. SOFTWARE Y APARIENCIA DE LA APP

Una vez se arranca el programa, la primera ventana que aparece es la de **Inicio**, en la cual se presenta el logo de la aplicación y la referencia a la Universidad.

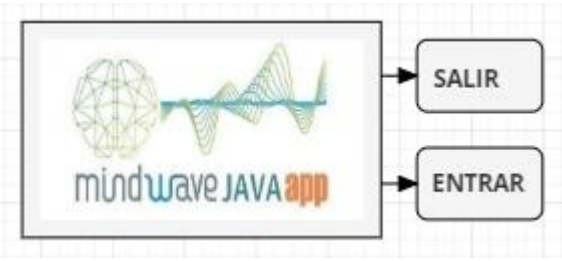

Ilustración 32. Interfaz de inicio.

En ella se adjuntan dos botones:

- **Salir:** Cierra la aplicación.

```
btnSalirApp.addActionListener(new ActionListener ()
{
    public void actionPerformed (ActionEvent e) {System.exit(0);}
});
```

- **Entrar:** Abre la **VentanaAccesoUsuario**.

```
btnEntrarApp.addActionListener(new ActionListener ()
{
    public void actionPerformed (ActionEvent arg0)
    {
       VentanaAccesoUsuario ventanaAccesoUsuario;
       ventanaAccesoUsuario = new VentanaAccesoUsuario ();
       ventanaAccesoUsuario.setLocationRelativeTo(null);
       ventanaAccesoUsuario.setResizable(false);
       ventanaAccesoUsuario.setVisible(true);
    }
});
```

Clicando en el botón **Entrar** se accede a la **VentanaAccesoUsuario** la cual tiene otro pequeño menú que permite acciones como registrarse en el sistema o acceder al menú principal de la aplicación.

NOTA: En la transición entre la ventana **Inicio** y la **VentanaAccesoUsuario** un mensaje recuerda al usuario que se ha de crear una carpeta en el escritorio con el nombre de **Inside_of_Mind** para poder hacer uso del programa. En ella se guardará el registro de los usuarios y las correspondientes contraseñas, y también se almacenarán todos los datos extraídos del casco cuando se solicite en la aplicación que se guarden.

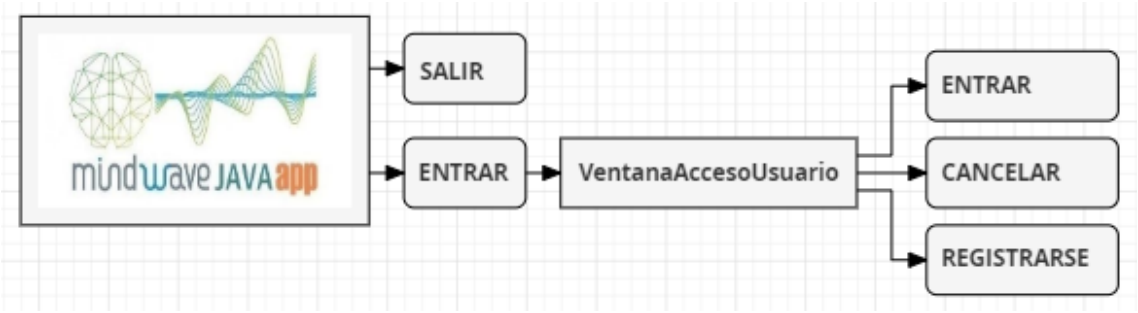

Ilustración 33. Ventana de acceso de usuario.

En ella se adjuntan tres botones:

- **Cancelar:** Volver a la ventana **Inicio.**

```
btnCancelar.addActionListener(new ActionListener ()
{
    public void actionPerformed (ActionEvent e) {dispose ();}
});
```

- **Entrar:** Verifica el usuario y la contraseña y abre la **VentanaInicio.**

```
btnEntrar.addActionListener(new ActionListener ()
{
    @SuppressWarnings("deprecation")

    public void actionPerformed (ActionEvent e)
    {
```

```
        int a = 0;

        // SI ALGÚN CAMPO ESTÁ VACÍO:

        if(VentanaAccesoUsuario.txtFieldUSUARIO.getText().isEmpty()
        && VentanaAccesoUsuario.passwordField.getText().isEmpty())
        {
            JOptionPane.showMessageDialog
            (null,"Por favor, introduzca el usuario y la contraseña");
        }

        else
        {
            if(VentanaAccesoUsuario.txtFieldUSUARIO.getText().isEmpty())
            {
               JOptionPane.showMessageDialog
               (null,"Por favor, introduzca el usuario.");
            }

            else if
            (VentanaAccesoUsuario.passwordField.getText().isEmpty())
            {
               JOptionPane.showMessageDialog
               (null,"Por favor, introduzca la contraseña.");
            }

            else
            {
               usuario = VentanaAccesoUsuario.txtFieldUSUARIO.getText();
               contrasena = VentanaAccesoUsuario.passwordField.getText();
               String CEncriptada = DigestUtils.md5Hex(contrasena);

               // COMPARAR NOMBRE Y CONTRASEÑA:

               try {a = validarUsuarioRegistrado(usuario, CEncriptada);}
               catch (IOException e1) {e1.printStackTrace();}

               if (a == 1) // SI ES CORRECTO:
               {
                   Menu VentanaInicio = new Menu ();
                   VentanaInicio.setExtendedState(MAXIMIZED_BOTH);
                   VentanaInicio.setVisible(true);
                   dispose ();
                   inicio.Autorun.ventanaIntro.setVisible(false);
               }

               else   // SI NO ES CORRECTO:
               {
                   JOptionPane.showMessageDialog
                   (null,"El usuario y/o la contraseña introducidos son
                   incorrectos o no están registrados\n en la base de
                   datos del programa. Por favor introduce correctamente
                   el USUARIO\n y la CONTRASEÑA en caso de estar
                   registrado; o regístrate en caso de no estarlo.");
               }
            }
        }
    }
});
```

Como se cita en las anotaciones adjuntas con el código, antes de acceder a la **VentanaInicio** se comprueba si todos los datos han sido introducidos, o si hay alguno por rellenar. Después, mediante la función **validarUsuarioRegistrado** se comprueba si el usuario y la contraseña están en la base de datos del programa. En caso afirmativo, se accede a **VentanaInicio**.

```
public int validarUsuarioRegistrado(String user, String password)
throws IOException
{
    String linea = null;
    String usuario = null;
    String contraseña = null;
    String fichero = "UsuarioDatos.txt";
    String c = System.getProperty("user.home");

    c = c + "\\"+"Desktop"+"\\"+"Inside_of_Mind"+"\\"+fichero;

    File f = new File (c);
    FileReader fr = new FileReader (f);
    BufferedReader br = new BufferedReader(fr);

    int i = 0; // RECORRER EL FICHERO.

    while ((linea = br.readLine ()) != null)
    {
       i = linea.indexOf (';');
       usuario = linea.substring (0, i);
       contraseña = linea.substring (i + 1, (linea.length() - 1));

       if (usuario.equals (user) && contraseña.equals (password))
       {
           fr.close ();
           return 1;
       }
    }

    fr.close();
    return 0;
}
```

- **Registrarse:** Abre la **VentanaRegistroUsuario**.

```
btnRegistro.addActionListener(new ActionListener ()
{
    public void actionPerformed (ActionEvent e)
    {
       VentanaRegistroUsuario ventRegistroUsuario;
       ventRegistroUsuario = new VentanaRegistroUsuario ();
       ventRegistroUsuario.setLocationRelativeTo(null);
       ventRegistroUsuario.setSize(455, 200);
       ventRegistroUsuario.setVisible(true);
    }
});
```

Clicando en el botón **Registrarse** se accede a la **VentanaRegistroUsuario** la cual tiene otro pequeño menú que permite al usuario registrarse en el sistema.

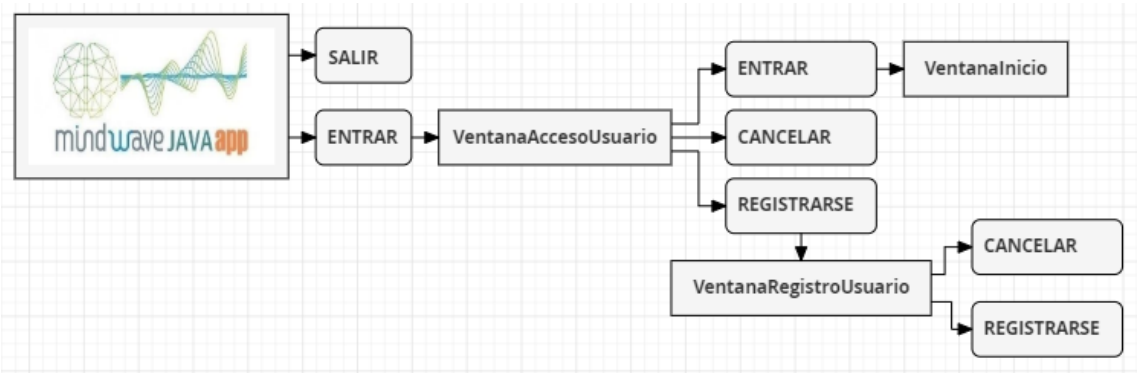

Ilustración 34. Ventana de registro de usuario.

En ella se adjuntan dos botones:

- **Cancelar:** Volver a la **VentanaAccesoUsuario.**

```
btnCancelar.addActionListener(new ActionListener ()
{
    public void actionPerformed (ActionEvent e) {dispose ();}
});
```

- **Registrar:** Registra en la base de datos del programa el usuario y la contraseña.

```
btnRegistrar.addActionListener(new ActionListener ()
{
    @SuppressWarnings("deprecation")
    public void actionPerformed (ActionEvent arg0)
    {
       String UsuarioNuevo = null;
       String NuevaContrasenaUsuario = null;

       if (VentanaRegistroUsuario. txtFieldNewUser.getText().isEmpty()
       && VentanaRegistroUsuario.txtFieldNewPass.getText().isEmpty())
       {
           JOptionPane.showMessageDialog
           (null,"Por favor, introduzca el usuario y la contraseña");
       }

       else
       if(VentanaRegistroUsuario.txtFieldNewUser.getText().isEmpty())
       {
           JOptionPane.showMessageDialog
           (null,"Por favor, introduzca el usuario.");
       }

       else
       if(VentanaRegistroUsuario.txtFieldNewPass.getText().isEmpty())
       {
           JOptionPane.showMessageDialog
           (null,"Por favor, introduzca la contraseña.");
       }

       else
       {
           UsuarioNuevo =
```

```
            VentanaRegistroUsuario.txtFieldNewUser.getText();

            NuevaContrasenaUsuario
            = VentanaRegistroUsuario.txtFieldNewPass.getText();

            String TextoEncriptado =
            DigestUtils.md5Hex(NuevaContrasenaUsuario);

            try {escribirFichero(UsuarioNuevo,TextoEncriptado);}
            catch (IOException e) {e.printStackTrace ();}

            JOptionPane.showMessageDialog
            (null,"Tu USUARIO y CONTRASEÑA se han registrado.");
            dispose ();
        }
    }
});
```

Mediante estas líneas de código, la aplicación coge la información escrita en los **textFields** y la almacena en un fichero dentro de la carpeta **Inside_of_Mind** mediante la función **escribirFichero**. Además, al igual que en el caso anterior, también se comprueba antes de guardar la información, si ambos campos están completos.

```
public void escribirFichero (String userNew, String passwNew)
throws IOException
{
    int z = userNew.length();
    String fichero = "UsuarioDatos.txt";
    String d = System.getProperty("user.home");
    String UsuarioNew = userNew.substring(0, z);
    BufferedWriter bw;

    d = d + "\\"+"Desktop"+"\\"+"Inside_of_Mind"+"\\"+fichero;

    File Escritura = new File (d);

    //PARA NO PISAR EL ARCHIVO:

       bw = new BufferedWriter (new FileWriter (Escritura, true));
       bw.write(UsuarioNew);
       bw.write(";");
       bw.write(passwNew);
       bw.write(";");
       bw.write("\r\n");
       bw.close();
}
```

Clicando en el botón **Entrar** de la **VentanaAccesoUsuario** se accede a **VentanaInicio**, la cual permite al usuario acceder a diferentes tipos de visionado de la información, el **juego de NeuroPong,** y también se ha completado la ventana programando un reloj y un calendario para completar la ventana principal de la aplicación.

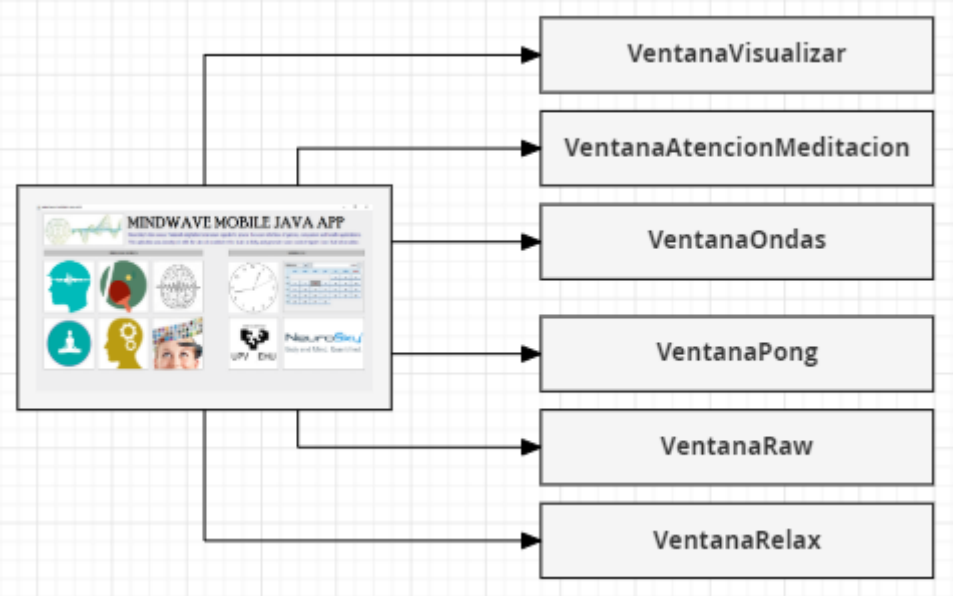

Ilustración 35. Menú principal de la aplicación.

Donde:

- **VentanaVisualizar:** Representa la ventana de **Visualizar señales**, y en ella se ha intentado hacer una representación global de toda la información que se obtiene del casco, de forma similar a cómo lo hace la aplicación que viene con el casco Mindwave Mobile, donde se representan en tiempo real los valores de atención, meditación, y las ondas cerebrales.

- **VentanaAtencionMeditacion**: Representa la ventana de **Atención y Meditación**, y en ella se grafican en tiempo real, los valores calculados por el algoritmo eSense, a partir de las ondas cerebrales, que representan los niveles de atención y meditación del usuario.

- **VentanaOndas:** Representa la ventana de **Ondas**, y en ella es posible visualizar en tiempo real el valor de cada una de las ondas cerebrales: alfa, beta, gamma y delta. La ventana está formada por gráficas, una por cada una de las posibles ondas cerebrales a capturar.

- **VentanaPong:** Representa la ventana de **NeuroPong**, y en ella se puede hacer uso de la máquina de estados generada a partir de los parpadeos detectados con el casco, para cambiar la dirección de la barra del juego. La pala, se desplazará pulsando la barra espaciadora, y modificará a partir de los citados parpadeos. También, en esta ventana se ha implementado un menú con el que modificar el tamaño de la pala y la velocidad de la pelota, con la intención de añadir dificultades al juego. Además, se han añadido tres representaciones gráficas: raw, señal de control y atención, con la intención de monitorizar la señal bruta, la conversión de la señal bruta en una señal de control, y observar cómo se modifican los valores de atención con las diferentes posibilidades del juego.

- **VentanaRaw:** Representa la ventana de **Encefalograma**, y en ella es posible ver en tiempo real, tanto la señal bruta como la señal de control si así se desea. La ventana permite visualizar cada una de ellas por separado, o las dos juntas en una misma ventana.

- **VentanaRelax:** Representa la ventana de **Meditación**, y ésta es una modificación del código base y de la apariencia de la ventana implementada por otro compañero en su trabajo de fin de carrera con el nombre de "Inside_of_Mind". Mediante esta ventana se pretende monitorizar tanto los valores de meditación calculados por el algoritmo eSense como los valores de la onda cerebral Theta, que es la asociada a este estado.

3.4.3.3.1. APARIENCIA DE LAS VENTANAS

Con la intención de mantener un mismo formato en el programa, todas las ventanas que se han implementado, mantienen el siguiente diseño:

Ilustración 36. Formato de las ventanas del programa

Donde el menú principal también tiene un formato estándar:

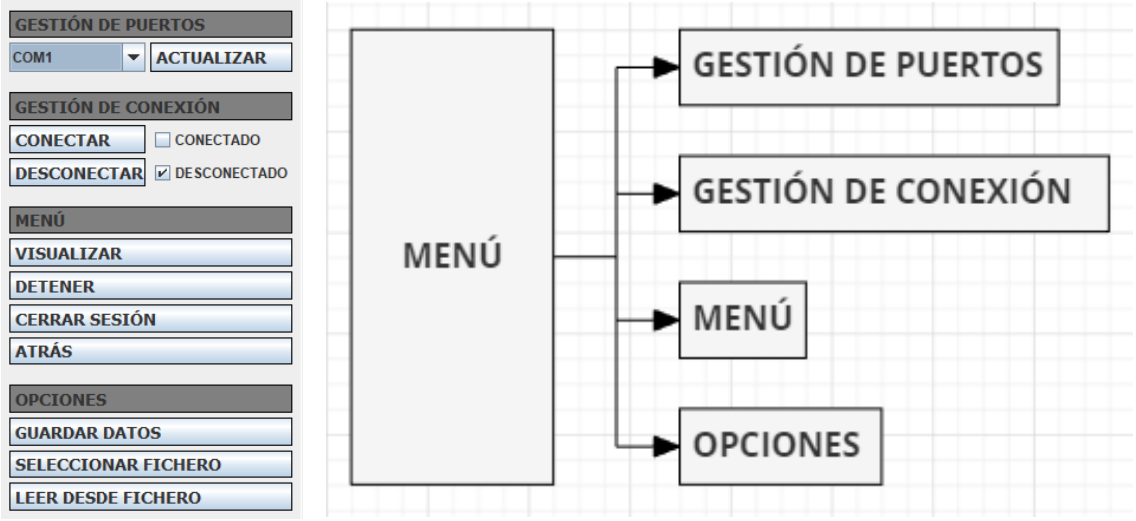

Ilustración 37. Menu de una ventana

En el subapartado de **Gestión de puertos** se ha programado un **JComboBox** donde se listarán los puertos del ordenador disponibles a los que conectarse.

```
comboBoxPuertos = new JComboBox();
comboBoxPuertos.setModel(new DefaultComboBoxModel(puertos));
```

También se ha implementado el botón **Actualizar** cuya función es volver a consultar qué puertos están disponible y elaborar una nueva lista en el **JComboBox.**

```
btnActualizar.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent arg0)
    {
        puertos = PuertoSerie.listarPuertosDisponibles();
        comboBoxPuertos.setModel(new DefaultComboBoxModel(puertos));
    }
});
```

Donde la función **listarPuertosDisponibles** ejecuta las siguientes líneas de código, que como se ha citado, se utilizan para elaborar una lista de puertos.

```
public static String [] listarPuertosDisponibles ()
{
    Enumeration<?> listaPuertos;
    String[] puertos = new String[numeroPuertosActivos()];
    CommPortIdentifier idPuerto;
    listaPuertos = CommPortIdentifier.getPortIdentifiers();

    for (int i = 0; listaPuertos.hasMoreElements(); i++)
    {
        idPuerto = (CommPortIdentifier) listaPuertos.nextElement();
        puertos[i] = (String) idPuerto.getName();
    }

    return puertos; // ARRAY CON LOS PUERTOS ACTIVOS CONECTADOS AL PC
}
```

De tal manera que el primer diagrama se podría completar de la siguiente manera:

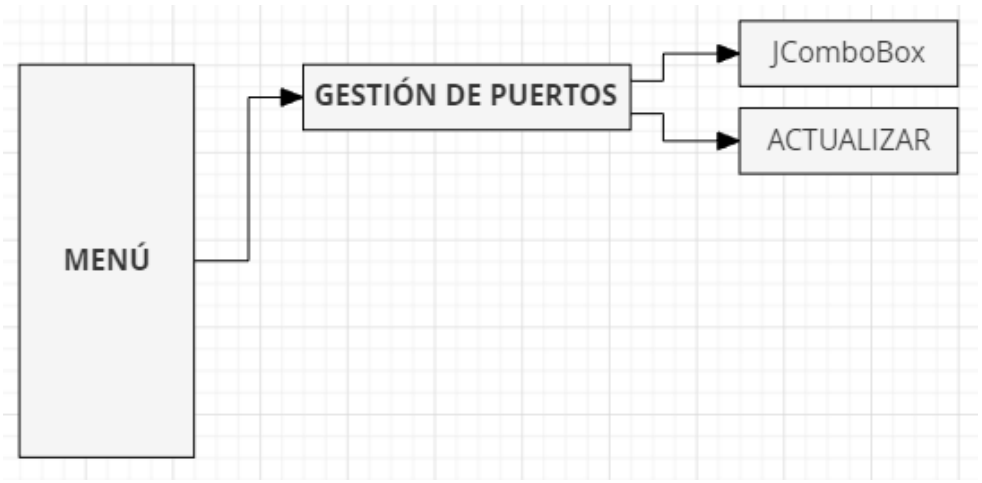

Ilustración 38. Gestión de puertos

En el subapartado de **Gestión de conexiones** se ha programado los botones **conectar** y **desconectar**, cuyo objetivo es vincular o desvincular el casco a la aplicación. En el caso específico del botón **conectar** también establece los parámetros de comunicación para el casco de Mindwave.

```
checkboxConectado = new JCheckBox("CONECTADO");
checkboxConectado.setSelected(false);

btnConectar.addActionListener(new ActionListener ()
{
   public void actionPerformed (ActionEvent arg0)
   {
       actionBotonConectar ();
   }
});
```

Donde la función **actionBotonConectar** coge el puerto seleccionado por el usuario del **comboBoxPuertos**, y obtiene su identificador mediante la función **seleccionarPuerto.**

```
// NOMBRE DEL PUERTO SELECCIONADO.
portName = (String) comboBoxPuertos.getSelectedItem();

// SELECCIONAR EL PUERTO PARA OBTENER SU IDENTIFICADOR
idPuerto = PuertoSerie.seleccionarPuerto(portName);

if (idPuerto == null)
{
   JOptionPane.showMessageDialog(null, "Puerto no seleccionado.");
}
```

La función **seleccionarPuerto**:

```
public static CommPortIdentifier seleccionarPuerto (String NOMBRE)
{
   CommPortIdentifier idPuerto = null;
   Enumeration<?> listaPuertos;
   boolean encontrado = false;
   listaPuertos = CommPortIdentifier.getPortIdentifiers();

   while (listaPuertos.hasMoreElements() && !encontrado)
   {
       idPuerto = (CommPortIdentifier) listaPuertos.nextElement();

       if (idPuerto.getPortType() == CommPortIdentifier.PORT_SERIAL)
       {
          if (idPuerto.getName().equals(NOMBRE_PUERTO))
          {
              encontrado = true;
              System.err.println(idPuerto.getName());
          }
       }
   }

   return idPuerto; // IDENTIFICADOR DEL PUERTO.
}
```

Una vez obtenido el identificador, mediante la función **abrirPuerto** se abre el puerto de comunicación seleccionado por el usuario, de la siguiente manera:

```
// ABRIR EL PUERTO.
puertoSerie = PuertoSerie.abrirPuerto(idPuerto, portName);

if (puertoSerie == null)
{
   JOptionPane.showMessageDialog(null, "No se pudo abrir puerto.");
}
else
{
   System.out.println("Puerto Abierto: " + puertoSerie.getName());
}
```

Donde la función **abrirPuerto**:

```
public static SerialPort abrirPuerto (CommPortIdentifier id, String n)
{
   SerialPort puertoSerie = null;
   try {puertoSerie = (SerialPort) idPuerto.open(name, TIME_OUT);}
   catch (PortInUseException e1)
   {
       System.out.println("Error abriendo el puerto"+ e1.getMessage());
   }
   return puertoSerie; // PUERTO SERIE QUE SE ABRE.
}
```

Por último, con la función **configurarPuerto** se establecen los parámetros característicos de una conexión serie entre dos dispositivos. Para ello, en las variables **BAUD**, **DATOS**, **STOP** y **PARIDAD** se almacena la configuración que se desea para los elementos en cuestión. De esta manera, cuando el programa introduce dichos parámetros en el puerto serie, éste queda configurado de la forma adecuada para su uso.

```
// CONFIGURAR EL PUERTO.
configurado = PuertoSerie.configurarPuerto(puertoSerie);
if (configurado == false)
{
   JOptionPane.showMessageDialog(null,"No pudo configurar puerto");
}
try {salida = puertoSerie.getOutputStream();}
catch (IOException e2) {e2.printStackTrace();}
try {entrada = puertoSerie.getInputStream();}
catch (IOException e1) {e1.printStackTrace();}
```

Donde la función **configurarPuerto**:

```
public static boolean configurarPuerto (SerialPort puertoSerie)
{
   try
   {
       puertoSerie.setSerialPortParams(BAUD, DATOS, STOP, PARIDAD);
       puertoSerie.notifyOnDataAvailable(true);
   }
   catch (UnsupportedCommOperationException e1)
   {
       System.out.println("Error estableciendo los parámetros.");
```

```
            return false; // NO SE HA CONFIGURADO CON ÉXITO.
        }
        return true; // SE HA CONFIGURADO CON ÉXITO.
    }
```

Para el caso del botón **desconectar** se tienen las siguientes sentencias:

```
checkboxDesconectado = new JCheckBox("DESCONECTADO");
checkboxDesconectado.setSelected(true);

btnDesconectar.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent arg0)
    {
        actionBotonDesconectar();
    }
});
```

Donde la función **actionBotonDesconectar** elimina el **EventListener** y cierra el puerto serie abierto para dar por finalizada la conexión entre el casco y el ordenador.

```
if (configurado == true)
{
    puertoSerie.removeEventListener();
    puertoSerie.close();
    checkboxConectado.setSelected(false);
    checkboxDesconectado.setSelected(true);
}

else {JOptionPane.showMessageDialog(null, "Puerto no conectado.");}
```

De tal manera que el primer diagrama sigue completándose de la siguiente manera:

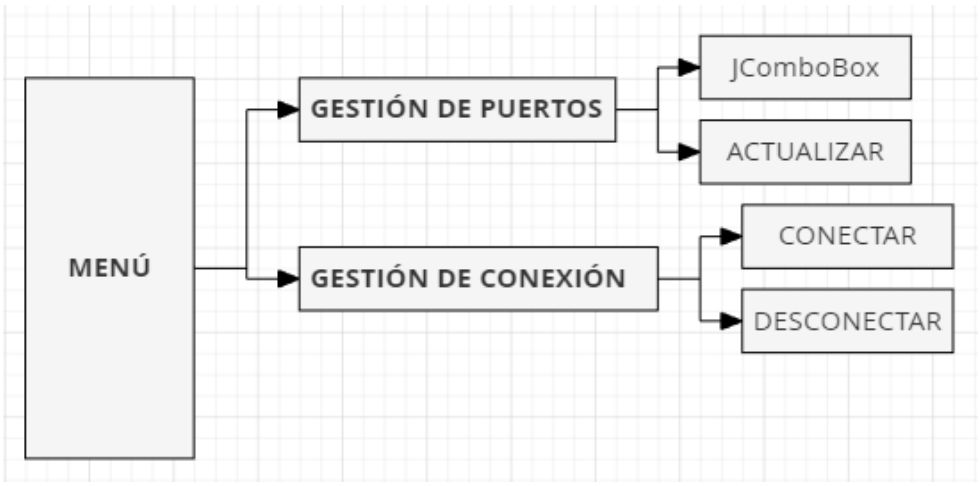

Ilustración 39. Gestión de conexión

En el subapartado de **Menú** se han programado los botones **Visualizar**, **Detener**, **Atrás** y **Cerrar sesión.** El funcionamiento de los dos primeros botones está relacionado con la representación gráfica de la información correspondiente a cada ventana, mientras que los dos siguientes se utilizan para navegar a través del programa.

El botón Visualizar implementa un **EventListener** que será el encargado de detectar que se ha recibido un paquete de información proveniente del casco.

```
btnVisualizar.addActionListener(new ActionListener ()
{
   public void actionPerformed (ActionEvent arg0)
   {
       visualizarVisualizar = true;
       try
       {
          PanelMenuVisualizar.puertoSerie.addEventListener(
          (SerialPortEventListener) new SerialReaderVisualizar
          (PanelMenuVisualizar.entrada));
       }
       catch (TooManyListenersException e) {e.printStackTrace();}
   }
});
```

Cada una de las ventanas tiene su propio **SerialReader**, que posteriormente envía la información recibida a un analizador de la trama para extraer de ella la información necesaria para cada una de las ventanas. Por otro lado, el botón **Detener**, como el nombre representa claramente, detiene la representación gráfica de la información recibida desde el casco, eliminando el **EventListener** haciendo uso de la sentencia **puertoSerie.removeEventListener().**

De esta manera, el primer diagrama se completa de la siguiente manera:

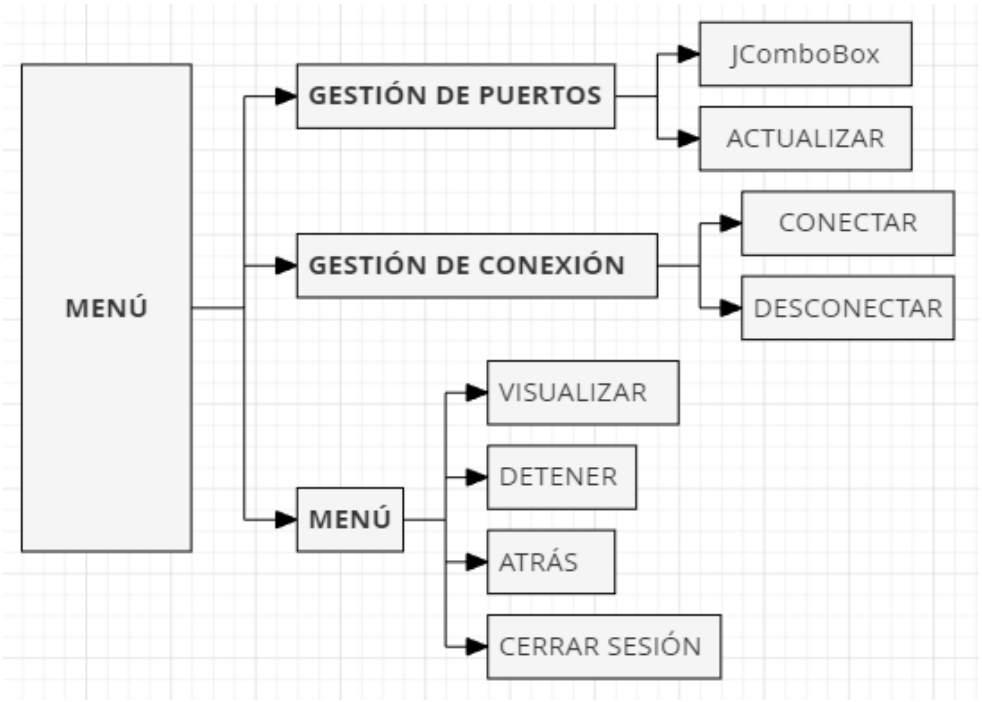

Ilustración 40. Menu

Además, los botones **Atrás** y **Cerrar sesión**, como se ha citado, se utilizan para navegar a través del programa y de las ventanas de éste. En el caso del primero, éste se utiliza

para volver a la ventana anterior a la que se está (Menu principal). En el caso del segundo, se utiliza para cerrar la sesión del usuario con el que se ha accedido, volviendo a la ventana de introducción del programa.

En el subapartado de **Opciones** se han programado los botones **Guardar datos**, **Seleccionar fichero a leer** y **Leer desde fichero.** Todas estas opciones están orientadas al trabajo con ficheros: creación y lectura de estos.

El botón **Guardar datos**, almacena la información graficada desde el casco en un fichero de formato ".txt" desde el momento en el cual se pulsa el citado botón.

NOTA: Es importante recordar que cualquier información extraída de la aplicación será almacenada en la carpeta "Inside_of_Mind", que es necesaria para que la aplicación funcione. En caso de no existir este directorio el programa no avanzará de la **ventana de acceso de usuario**.

El botón **Seleccionar fichero a leer** permite al usuario activo representar gráficamente en el entorno la información de otra sesión de manera muy simple.

```
btnSeleccionarFichero.addActionListener(new ActionListener ()
{
   public void actionPerformed (ActionEvent arg0)
   {
      actionBotonSeleccionarFichero ();
   }
});
```

Donde la función **actionBotonSeleccionarFichero** implementa una nueva ventana que permite al usuario realizar una búsqueda en la memoria del ordenador para seleccionar el fichero que se desea representar.

```
public void actionBotonSeleccionarFichero ()
{
   SeleccionarFichero seleccionarFichero = new SeleccionarFichero ();
   seleccionarFichero.setSize(400, 110);
   seleccionarFichero.setVisible(true);
   ficheroSeleccionado = true;
}
```

En el momento en el cual se pulsa el botón **SeleccionarFichero** aparecerá la ventana:

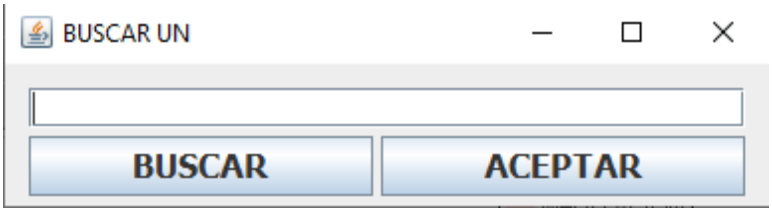

Ilustración 41. Buscar un fichero

Y clicando en el botón **Buscar**:

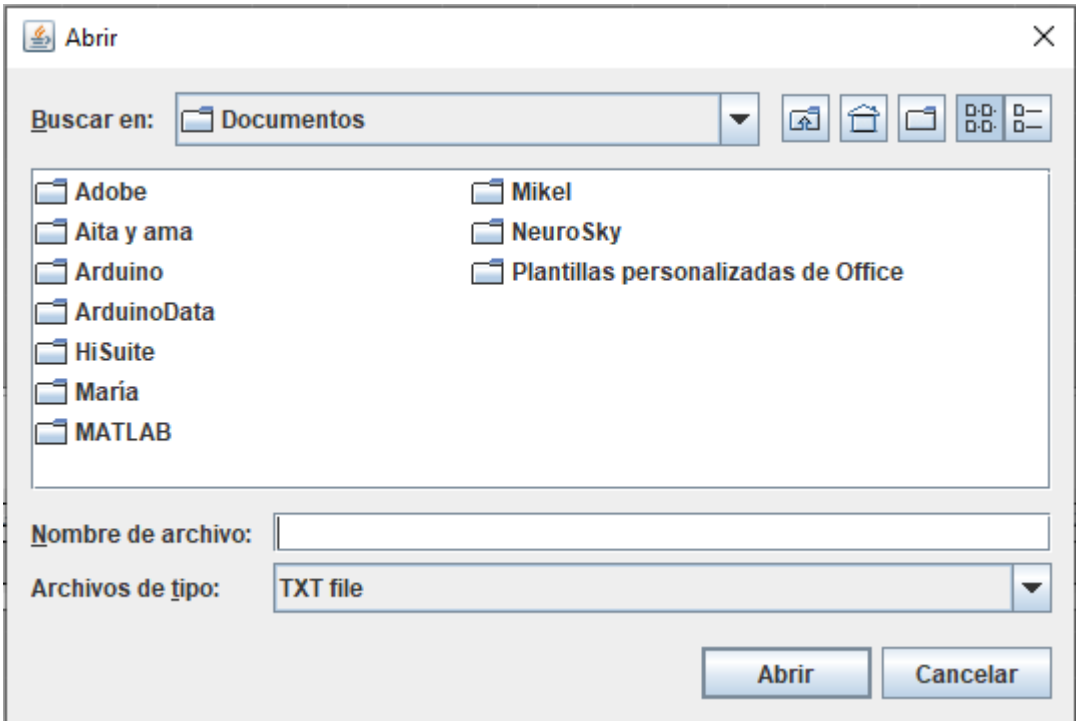

Ilustración 42. Directorios del ordenador

Mediante esta ventana se puede navegar a través de los directorios del sistema, hasta encontrar la carpeta que contenga el archivo que se quiere representar. Una vez encontrado el citado **fichero**, pulsando en el botón **Abrir** y posteriormente en el botón **Aceptar** de la anterior ventana, el programa almacenará en su memoria la dirección de memoria en la que está dicho fichero para poder graficarlo en cuanto el usuario pulse el botón **LeerDesdeFichero**.

Cuando se pulsa el botón **LeerDesdeFichero** dependiendo de la ventana en la cual se quieran representar los datos, el **ActionListener** de botón graficará de una manera determinada la información, o lo hará de otra. Esto depende de la naturaleza de información que se desea representar.

```
btnLeerDesdeFichero.addActionListener(new ActionListener ()
{
   public void actionPerformed (ActionEvent arg0)
   {
       actionBotonLeerDesdeFichero ();}});
   }
}
```

Donde la función **actionBotonLeerDesdeFichero**, que para cada ventana del programa es diferente, de forma general funcionará de la siguiente manera:

1) Leer la información del fichero.
2) Separarla en las variables de interés.
3) Sincronizar cada variable con su variable local en el programa.
4) Graficar la información con un intervalo de 100ms de tiempo.

De esta manera el diagrama genérico del panel de menú se completa así:

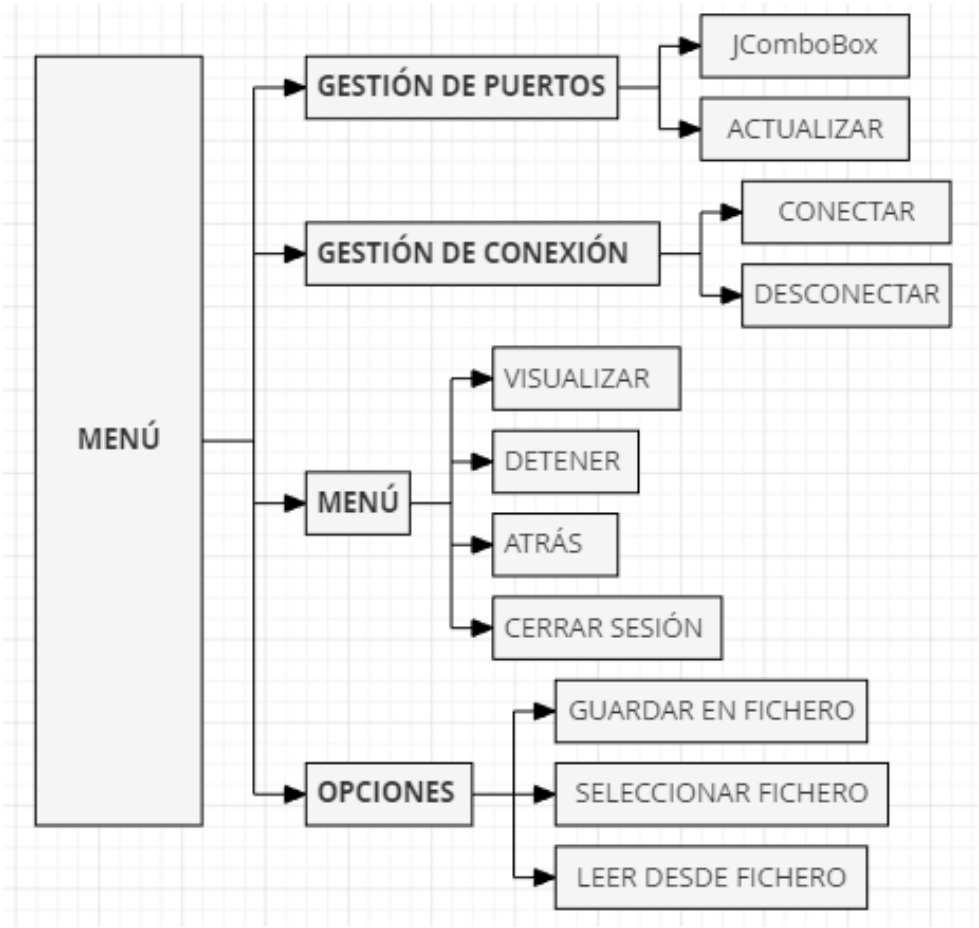

Ilustración 43. Diagrama de bloques del Panel de menú

3.4.3.3.2. MÉTODO DE ANÁLISIS DEL PROTOCOLO

En primer lugar y como se ha citado en el apartado **3.4.1.12. ANALISIS UN PAQUETE DE INFORMACIÓN**, para analizar la trama, se han de realizar varias comprobaciones:

- **1ª COMPROBACIÓN:** Los dos primeros bytes son 0xAA.
- **2ª COMPROBACIÓN:** La longitud de PAYLOAD no es superior a 169.

De tal manera que:

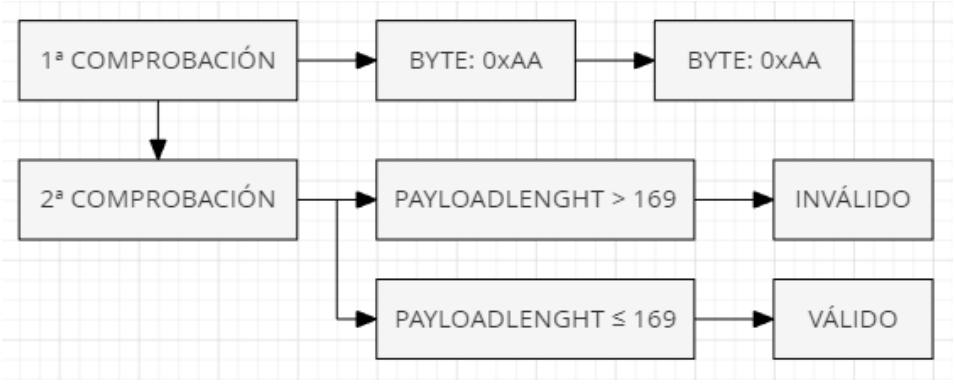

Ilustración 44. Comprobaciones de la trama

Después de validar las dos comprobaciones, se procede a analizar la información tal y como se cita en el apartado **3.4.1.13. ANALISIS DE LOS DATAROWS DE LA CARGA ÚTIL DE UN PAQUETE.** Como se cita, cada valor de información que se envía viene precedido por un [CODE] que representa el tipo de dato que se envía. Es por ello, que mediante un **switch** se almacena la información en la variable correspondiente, poniendo en cada **case** como identificador, cada uno de los [CODE] que pueden darse.

El código sería el siguiente, donde con un **while** se corre todo el PAYLOAD analizando la información que éste contiene de la siguiente manera:

```
case (byte) 0x02:
{
   System.out.print ("[analizarTrama] [INFCODE]: 02 - "); z++;
   poorQuality = payloadData[z];
   informacion[1] = poorQuality;
   System.out.println ("POOR_SIGNAL Quality: " + poorQuality);
   break;
}

case (byte) 0x04:
{
   System.out.print ("[analizarTrama] [INFCODE]: 04 - "); z++;
   nivelAtencion = payloadData[z];
   informacion[2] = nivelAtencion;
   System.out.println("NIVEL DE ATENCION:" + nivelAtencion);
   break;
}

case (byte) 0x05:
{
   System.out.print ("[analizarTrama] [INFCODE]: 05 - "); z++;
   nivelMeditacion = payloadData[z];
   informacion[3] = nivelMeditacion;
   System.out.println("NIVEL DE MEDITACION: " + nivelMeditacion);
   break;
}

case (byte) 0x80:
{
   System.out.print ("[analizarTrama] [INFCODE]: 80 - ");
   raw = (payloadData[2] << 8) + payloadData[3];
   System.out.println("RAW Wave Value (-32768 a 32767): " + raw);
   informacion[4] = raw;
   break;
}

case (byte) 0x83:
{
   System.out.println ("[analizarTrama] [INFCODE]: 83 - "); z++;
   VLENGH = payloadData[z]; z++;

   for (int x = 0; x < VLENGH; x++)
   {
      Ondas[x] = payloadData[z];
      System.out.println ("ASIC_EEG_POWER[" + x + "]: " + Ondas[x]);
      z++;
   }
```

```
        z = z + VLENGH;
        break;
    }

    case (byte) 0x16:
    {
        System.out.print ("[analizarTrama] [INFCODE]: 16 - "); z++;
        blinkStrenght = payloadData[z];
        System.out.println ("Blink Strength (0 a 255): " + blinkStrenght);
        break;
    }
```

Donde en la variable **int informacion []** se almacena la información captada por el casco de la siguiente manera. con el objetivo de representar en un array de enteros todos los valores de información extraídos de la trama:

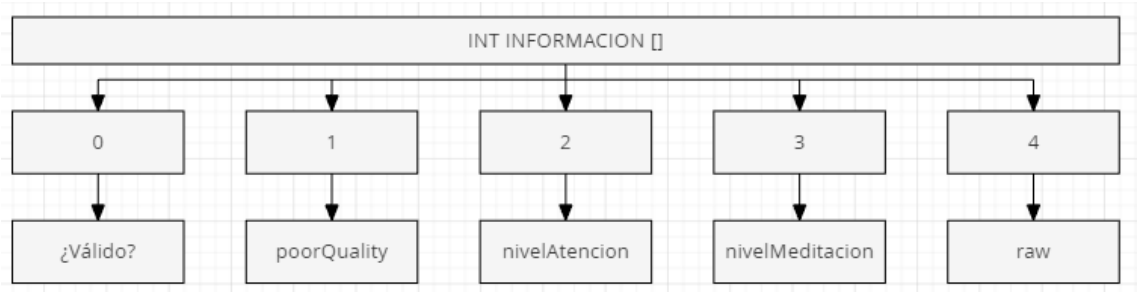

Ilustración 45. Array de enteros con la información de la trama

3.4.3.3.3. SERIAL READER

El Serial Reader obtiene la información del **InputStream** mediante la siguiente función:

```
public SerialReader (InputStream entrada) // LEE EL PUERTO SERIE.
{
    this.entrada = entrada; // GUARDA DATOS ENTRADA EN VARIABLE
}
```

Se implementa también un **serialEvent** que ejecutará las siguientes líneas de código si el evento generado es de tipo **SerialPortEvent.DATA_AVAILABLE;** es decir, siempre que se genere un evento en el puerto que represente que hay información disponible para leer, el programa ejecutará lo siguiente:

```
public void serialEvent (SerialPortEvent evento)
{
    if (evento.getEventType() == SerialPortEvent.DATA_AVAILABLE)
    {
        try
        {
            bytesAvaliables = entrada.available(); // Nº ESTIMADO BYTES
        }
        catch (IOException e2) {e2.printStackTrace();}
        if (bytesAvaliables != 0)
        {
```

```
buffer = new byte[bytesAvaliables];
try
{
    // NÚMERO DE BYTES QUE HAY REALMENTE.
    BytesLeidos = entrada.read(buffer, 0, bytesAvaliables);
}

catch (IOException e1) {e1.printStackTrace();}
```

La función de estas líneas de código es la siguiente:

1. Calcular el número estimado de bytes que se pueden leer.
2. Calcular el número real de bytes que se tienen.
3. Almacenar la información en un buffer.

Una vez almacenada la información que había en el **SerialReader** en el buffer, ésta se envía a la función **analizarTrama** para extraer cada una de las variables de interés en el array de "**informacion"** explicado en **3.4.3.3.2. MÉTODO DE ANÁLISIS DEL PROTOCOLO.** De esta manera, el **SerialReader** obtiene la información separa en variables.

```
informacion = MetodoProtocoloMindWave.analizarTrama(buffer);
```

Posteriormente, comprueba si la trama es válida o no analizando la primera posición de la variable "**informacion".** Es en dicha posición donde la función **analizarTrama** guardará siempre la información acerca de la validez de las tramas.

En caso de que los datos no sean válidos, se establecerá un cero en las variables de interés de cada ventana y no se representarán en los gráficos de la ventana.

```
if (informacion[0] == -1) {raw = 0;}
```

En caso de ser validos los datos, estos se almacenarán en las correspondientes variables y se representarán en las gráficas de la ventana correspondiente. Por ejemplo, en el código que se muestra a continuación, se representa la señal bruta captada por el casco, almacenada en la variable **raw** y en caso de que el usuario lo solicite, se representa también la **señal de control** generada a partir de la onda bruta.

```
else
{
   raw = informacion[4];
   doubleRaw = new Double(raw).doubleValue();
   VentanaRaw.serieColectionRaw.getSeries(0).add
   (new Millisecond(), doubleRaw);
   if (VentanaRaw.visualizarControl == true)
   {
       señalControl = GenerarSeñalControl.generarSeñalControl(raw);
       doubleControl = new Double(señalControl).doubleValue();
       VentanaRaw.serieColectionControl.getSeries(0).add
       (new Millisecond(), doubleControl);
   }
}
```

De forma resumida, el **SerialReader** se podría representar de la siguiente manera:

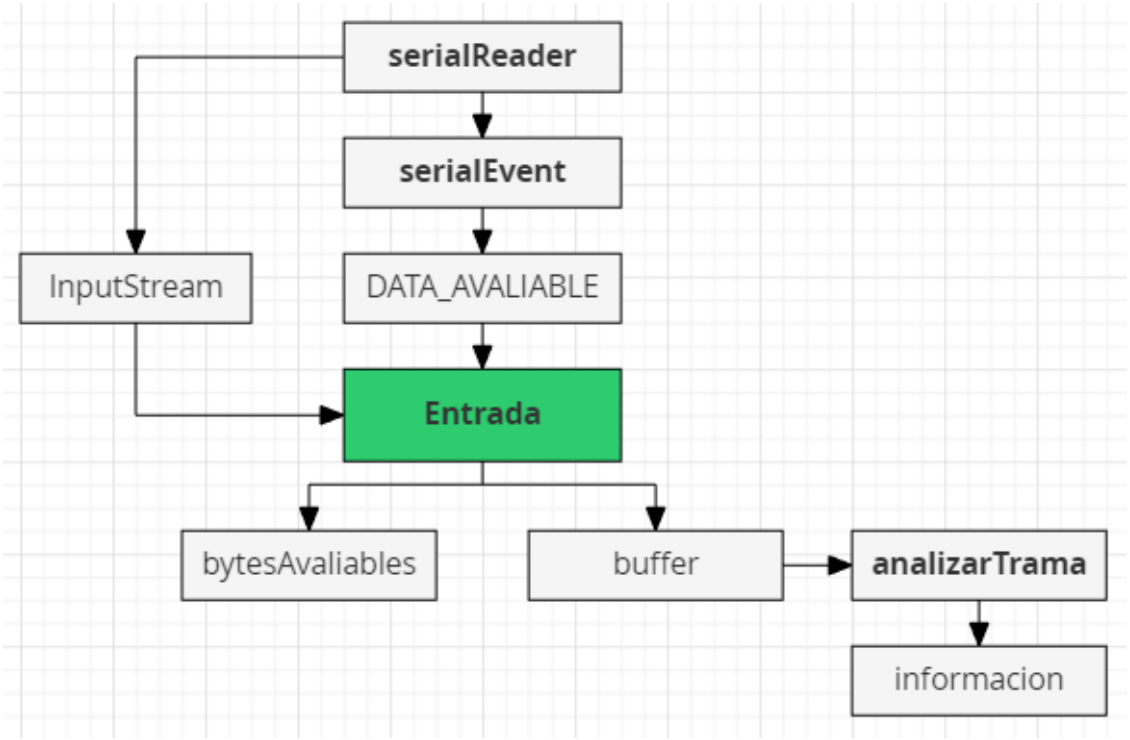

Ilustración 46. SerialReader

3.4.3.3.4. NEURO PONG

La apariencia de la ventana del juego sería la siguiente:

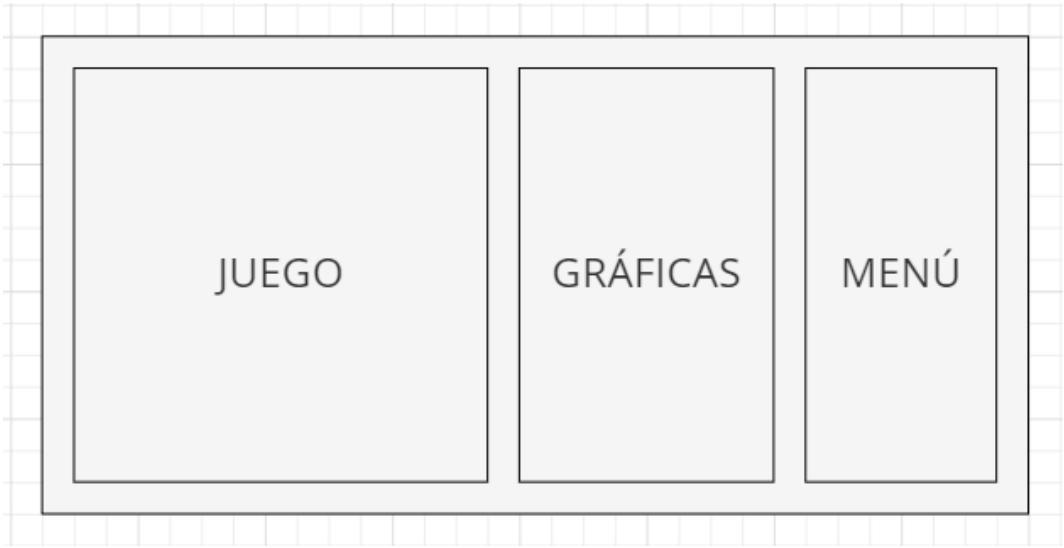

Ilustración 47. Ventana NeuroPong

El **JPanel** que contiene el **juego** que está formado por un conjunto de clases de Java que crean cada uno de los elementos necesarios: una pelota, un tablero, una pared, y una raqueta. La **raqueta** es la que el usuario moverá haciendo uso de la máquina de estados generada a partir de la señal de control construida con los parpadeos.

De manera que el panel del juego queda definido de la siguiente manera:

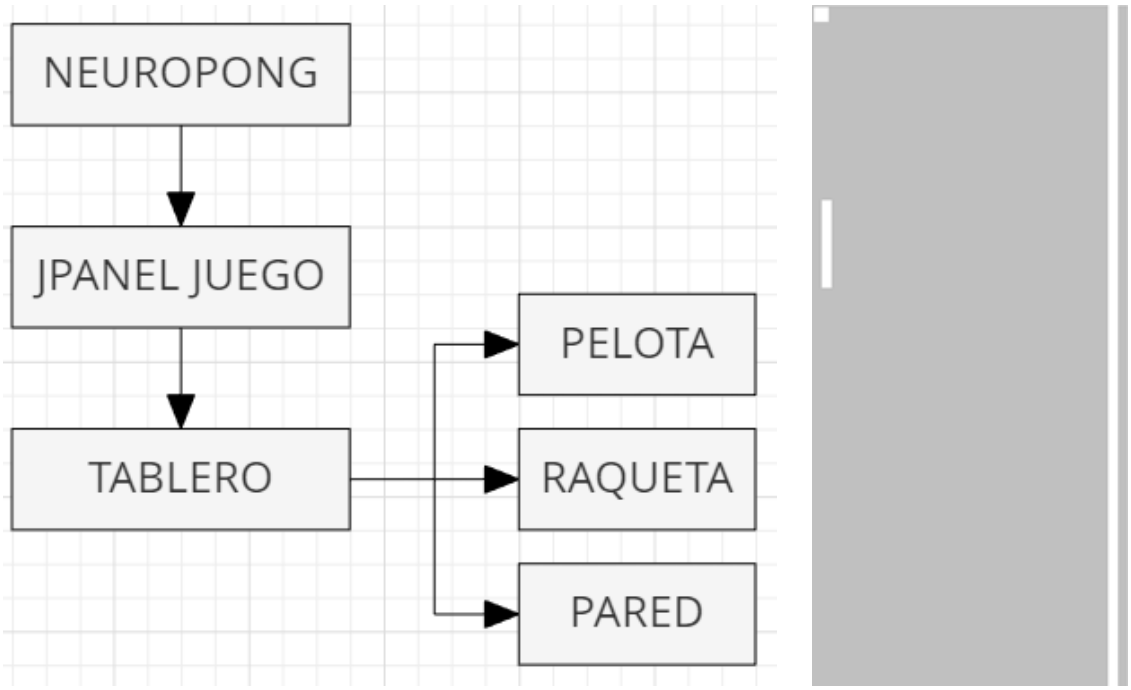

Ilustración 48. JPanel NeuroPong

El JPanel que contiene las **gráficas** está formado por tres **JFreeCharts** cuyo objetivo no otro que realizar una representación en tiempo real de la información objeto de estudio, de la señal de control generada a partir de la primera, y en último lugar una gráfica que representa la evolución del nivel de atención del usuario durante el juego.

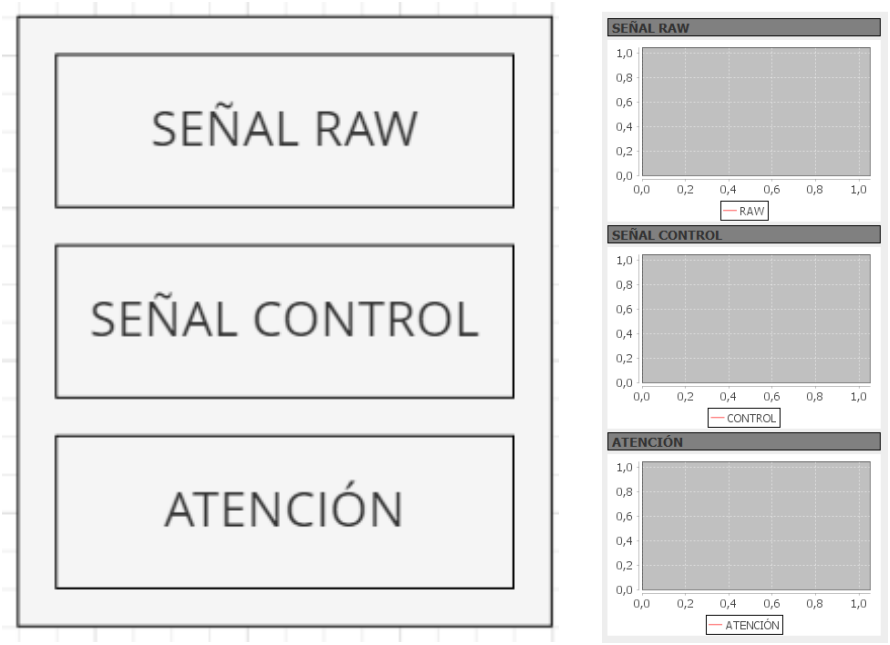

Ilustración 49. JPanel Gráficas

El JPanel que contiene el **menú** está formado por los botones necesarios para vincular el casco a la aplicación (**gestión de puertos** y **gestión de conexión**), con los botones del

menú principal tipo que se ha explicado, y además incluye dos **JSliders** para controlar tanto la velocidad de la pelota como el tamaño de la pala.

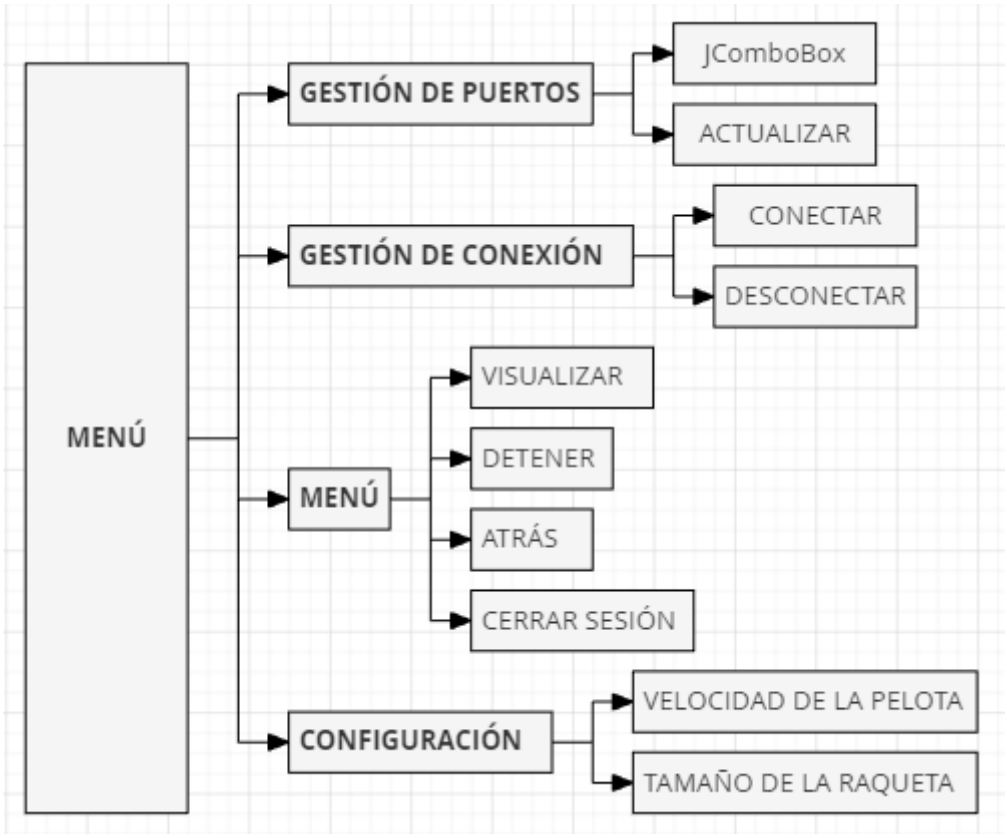

Ilustración 50. JPanel Menú

Donde el submenú **Configuración** queda configurado de la siguiente manera:

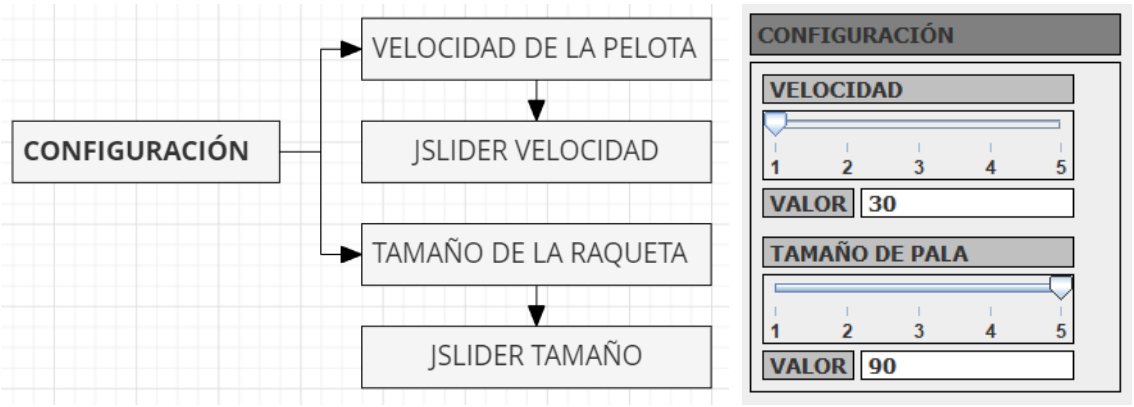

Ilustración 51. Submenú Configuración

El **JSlider** de la velocidad modifica el tiempo (en milisegundos) que utiliza el programa entre dos repaint del tablero, de la siguiente manera:

```
canvas.repaint();
try
{
	Thread.sleep(velocidad); // Modificar la velocidad de la pelota
}
```

Cuando mayor es el número de JSlider, menor es la variable que se introduce en **Thread.sleep** y, por tanto, el tablero se repinta con menor cadencia, lo que hace que la pelota "se desplace más rápido".

El **JSlider** del tamaño de la raqueta modifica el tamaño del rectángulo que en el juego hace las veces de la citada raqueta, de la siguiente manera:

```
public Rectangle2D getRaqueta ()
{
	alto = VentanaNeuroPong.getancho();
	return new Rectangle2D.Double(x, y, ancho, alto);
}
```

Cuando mayor es el número de JSlider, menor es la variable que se introduce en **Rectangle2D.Double** y, por tanto, el tamaño de la raqueta disminuye.

Ambos **JSliders** están diseñados de tal manera que cuando el valor de estos incrementa añadan dificultad al juego; es decir, cuando mayor es el número, mayor es la velocidad de la pelota y menor el tamaño de la pala. En el caso contrario, el juego estaría en el mínimo de dificultad programado.

3.4.3.3. APARIENCIA DE LA APLICACIÓN

Cuando se inicia la aplicación, en primer lugar, aparece la **ventana de inicio** que contiene el logo de la aplicación, en qué universidad se ha desarrollado y los botones de **entrar** y **salir**, y queda configurada de la siguiente manera:

Ilustración 52. Ventana de Inicio

Cuando se pulsa el botón de **entrar,** después de que en un **JOptionPane** mediante un mensaje explique que se ha de crear un directorio en el escritorio con el nombre de "Inside_of_Mind" para poder hacer uso de la aplicación, aparecerá la **ventana de inicio de sesión**, configurada así:

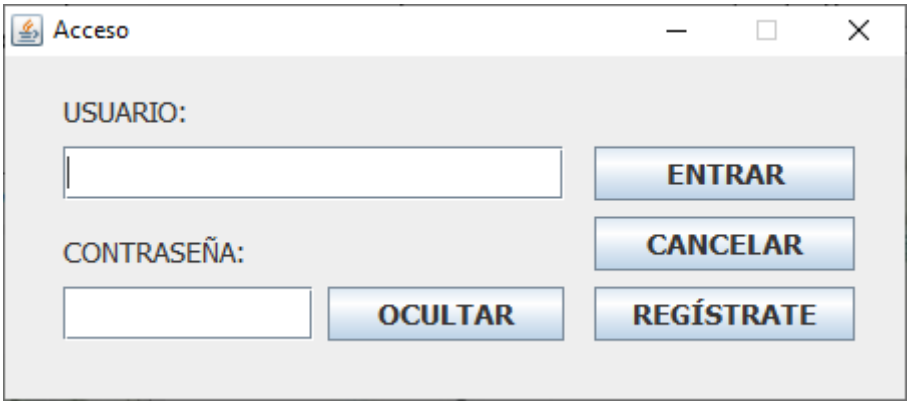

Ilustración 53. Ventana de Acceso de Usuario

Ésta permite al usuario tanto iniciar sesión como registrarse en el sistema. Y es cuando pulsa el botón registrarse cuando aparece la ventana de registro de usuario, con la siguiente apariencia:

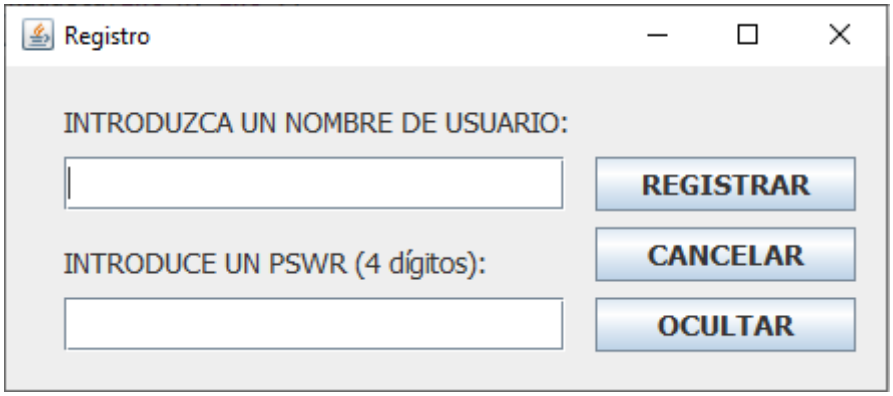

Ilustración 54. Ventana de Registro de Usuario

Sin embargo, si el usuario introduce un usuario y contraseña registrados en la base de datos de la aplicación y pincha en el botón entrar de la anterior ventana, se accede a la **ventana de Menú**, o ventana principal de la aplicación:

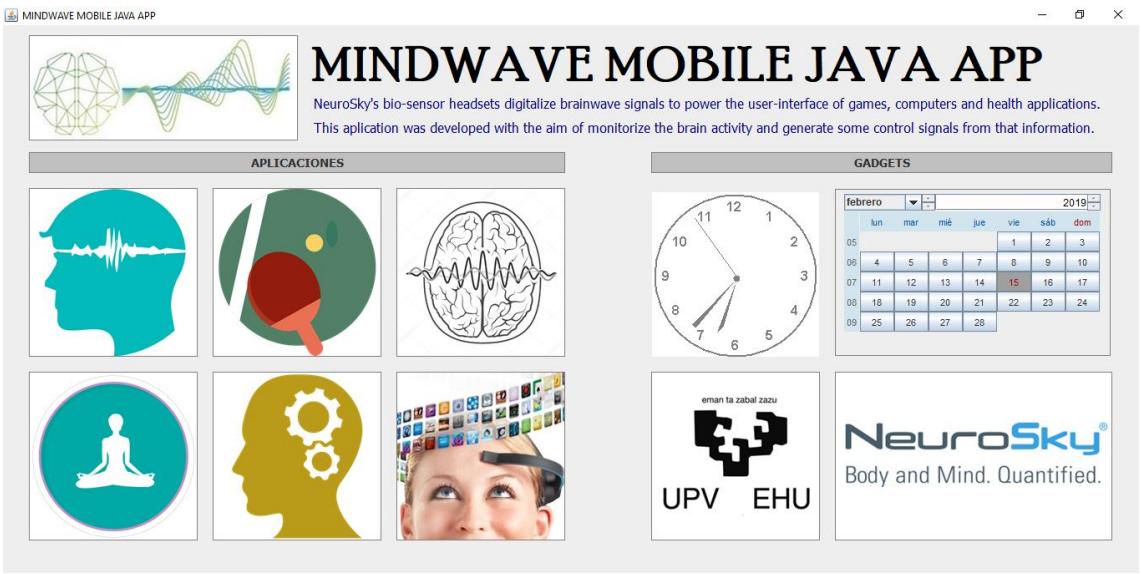

Ilustración 55. Ventana de Menú de la aplicación

En el apartado de aplicaciones se tienen las siguientes opciones:

Ilustración 56. Aplicaciones

Pinchando en **ENCEFALOGRAMA**:

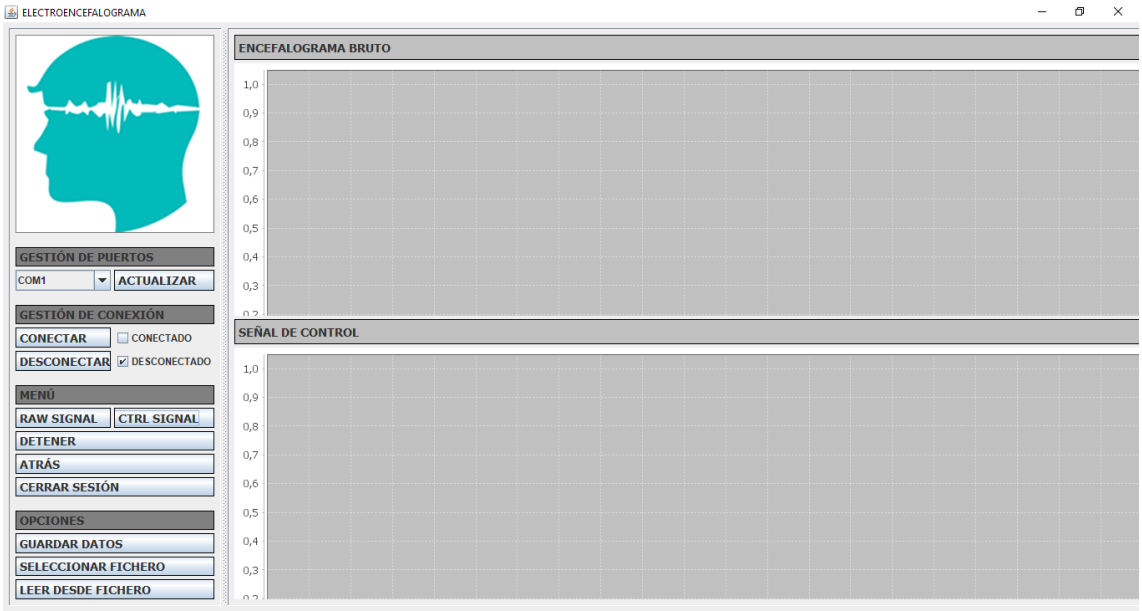

Ilustración 57. Encefalograma

Se tiene:

- Menú con las opciones citadas.
- Pinchando en **RAW SIGNAL** aparecerá la representación gráfica de la señal bruta.
- Pinchando en **CTRL SIGNAL** aparecerá la señal de control creada con parpadeos.

NOTA: Se puede visualizar la señal de control sin necesidad de que la graficación de la señal bruta esté activa; y de la misma manera, se podrá ver únicamente la señal bruta.

Pinchando en **ONDAS**:

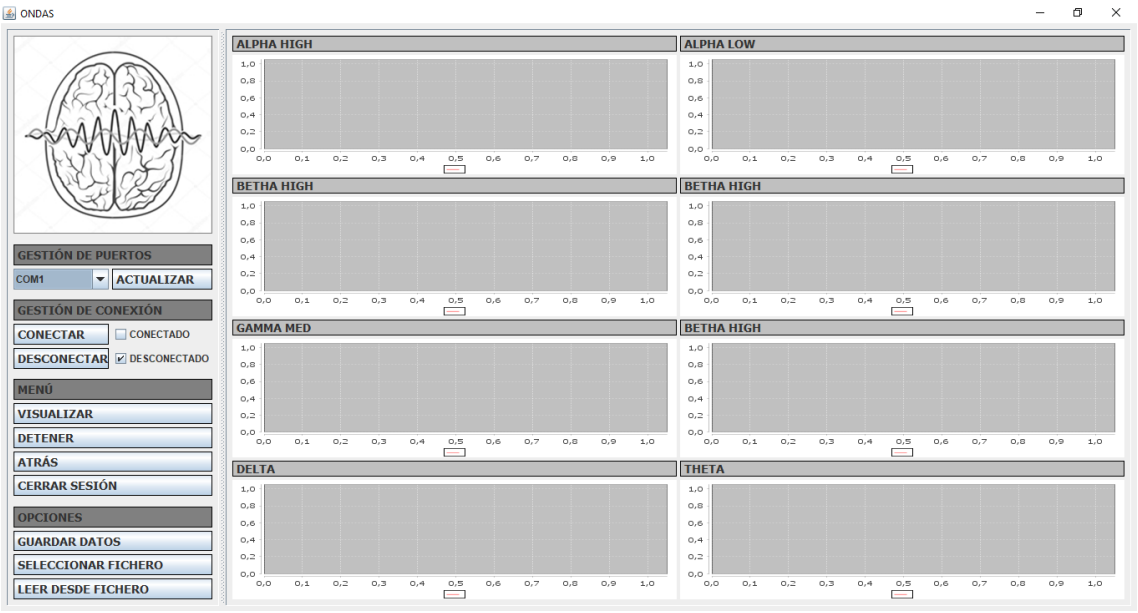

Ilustración 58. Ondas

Se tiene:

- ❖ Menú con las opciones citadas.
- ❖ Pulsando el botón de visualizar se representarían las ondas cerebrales.

Donde:

Ilustración 59. Ondas cerebrales.

NOTA: Es importante recordar que se ha de configurar el casco de tal manera que se envíe esta información en forma de tramas, ya que éste puede trabajar en diferentes modos de transmisión de información.

Pinchando en **RELAXING TIME**:

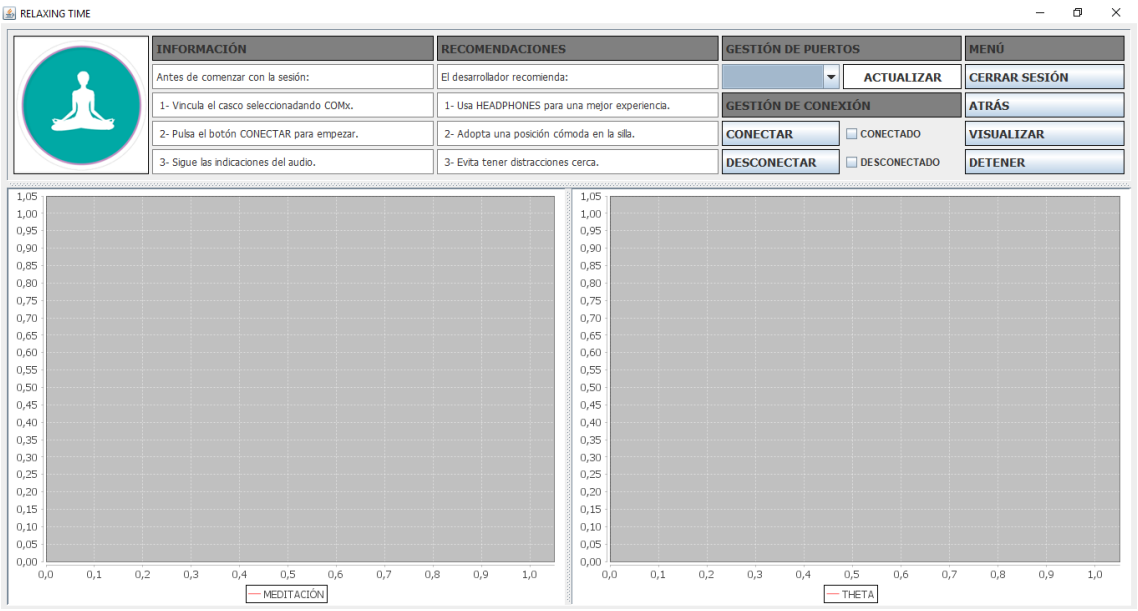

Ilustración 60. Relaxing time

Se tiene:

- Menú sin las opciones de guardar la información en un fichero.
- Pulsando el botón de visualizar se representarían la meditación y theta.
- También se adjuntan unas recomendaciones e información para usar la app.

Pinchando en **ATENCIÓN Y MEDITACIÓN:**

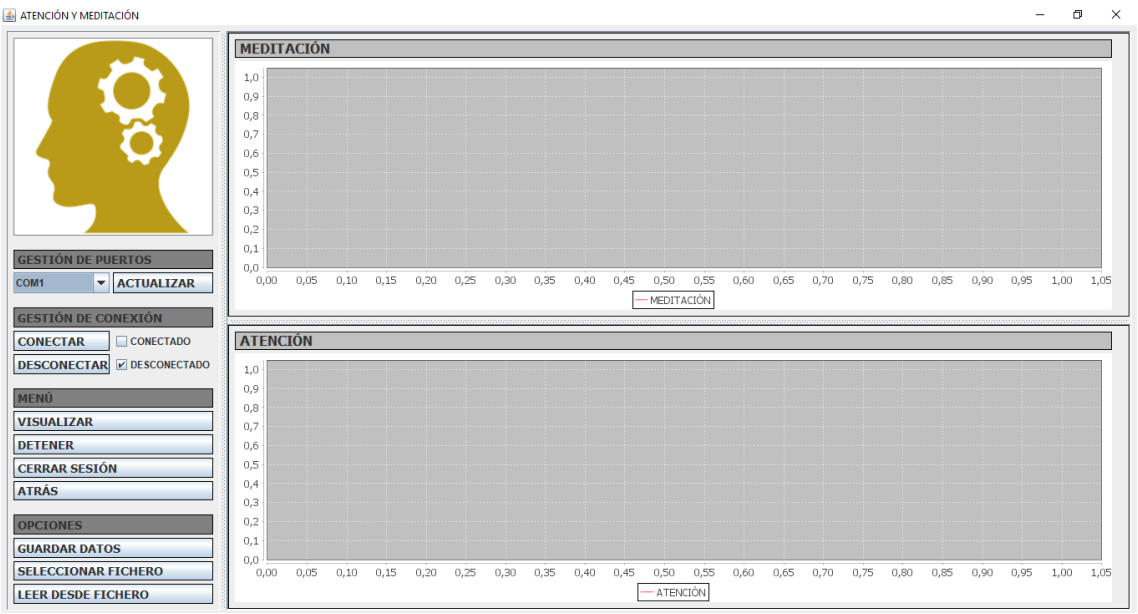

Ilustración 61. Atención y meditación

Se tiene:

- Menú con las opciones citadas.
- Pulsando el botón de visualizar se representaría la atención y la meditación.
- Estos dos valores son calculados con el algoritmo eSense.

Pinchando en **VISUALIZAR SEÑALES:**

Ilustración 62. Visualizar señales

Se tiene:

- Menú con las opciones citadas.
- Pulsando el botón de visualizar se representaría toda la información del casco.

NOTA: Esta ventana representa los valores que se representan en la aplicación desarrollada por Neurosky y que acompaña con el casco en formato CD.

Pinchando en **NEURO PONG:**

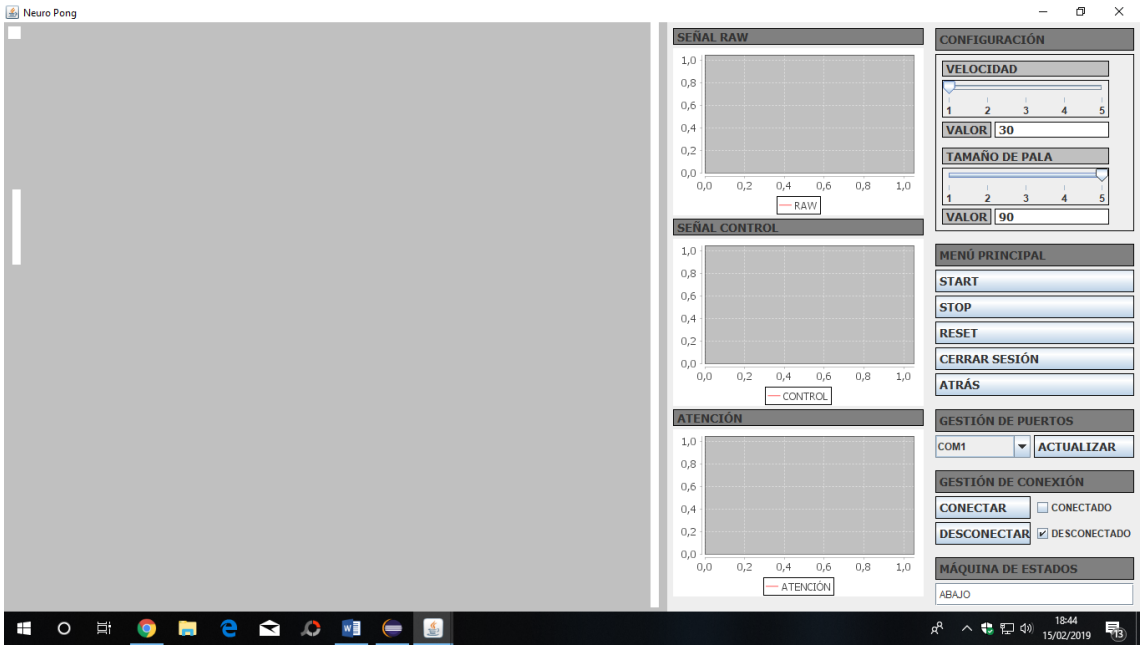

Ilustración 63. Neuro Pong

4. RESULTADOS

4.1. CONCLUSIONES

El objetivo del proyecto era crear un interfaz cerebro computador. Para ello, se ha estudiado el hardware y el software del casco Mindwave Mobile de la familia Neurosky. Esto ha permitido un control sobre la información que éste proporciona y una mayor facilidad a la hora de trabajar con ella. Además, durante el proceso de estudio del software del dispositivo, se ha podido conocer el protocolo de comunicación del casco, vital para la aplicación que se ha desarrollado, así como para generar los algoritmos que extraen la información de las tramas.

También, se ha realizado un análisis de la información obtenida mediante el electrodo del casco y se ha tratado para conseguir las señales de control. Gracias a esto, se ha podido implementar la máquina de estados que controla la dirección del movimiento de la raqueta del juego. Unificando lo citado, se ha conseguido implementar una aplicación Java capaz de representar gráficamente la información extraída del casco y, además, desarrollar un juego donde la máquina de estados generada a partir de los parpadeos, sea utilizada como uno de los controles del juego.

Durante el proceso de implementación, se ha aprendido Java dentro del entorno eclipse, donde el IDE de Eclipse proporciona un gran abanico de posibilidades y facilidades en la programación. Además, el utilizar Java como lenguaje de programación ha permitido conocer los lenguajes de programación orientado a objetos, y reutilizar parte de código permitiendo reducir el trabajo a realizar.

Por último, se ha experimentado que el mundo de los sistemas BCI que permiten que la ingeniería colabore con otras ciencias, como puede ser la biología, la psicología, la medicina. Es una prueba que muestra como con la instrumentación adecuada se puedan hacer uso de los estímulos físicos para simplificar ciertas tareas en la vida de una persona.

4.2. LÍNEAS DE DESARROLLO FUTURAS

A partir del trabajo realizado, se podría orientar el estudio a:

- Mejorar la sensibilidad del programa del NeuroPong en cuanto a la máquina de estados se refiere; es decir, eliminar los rebotes en los parpadeos.

- Buscar otros estímulos captados por el casco Mindwave que puedan ser utilizados para confeccionar otras máquinas de estados complementarias a la actual. Aunque, tal y como se citan en muchos proyectos realizados con dicho casco, sería necesario realizar un entrenamiento de la mente para poder gestionar determinados estímulos.

- ❖ Haciendo uso de herramientas matemáticas como la transformada de Fourier para descomponer la señal bruta captada por el casco en las ondas cerebrales características del cerebro con el objetivo de realizar una comparación entre la información captada por el casco y los resultados del análisis matemático.

- ❖ Utilizar las máquinas de estados en una aplicación física; por ejemplo, ser capaces de gestionar en tiempo real el funcionamiento de un determinado dispositivo, haciendo uso únicamente de las señales cerebrales.

5. BIBLIOGRAFÍA

[1] http://www.fgcsic.es/lychnos/es_ES/articulos/Brain-Computer-Interface-aplicado-al-entrenamiento-cognitivo

[2] https://www.cognifit.com/es/cerebro

[3] http://tugimnasiacerebral.com/gimnasia-cerebral-para-ni%C3%B1os/trastorno-por-deficit-de-atencion-en-ni%C3%B1os-con-sin-hiperactividad-sintomas-tratamiento-tda-tdah

[4] http://www.schalklab.org/research/bci2000

[5] http://openvibe.inria.fr/

[6] http://sccn.ucsd.edu/wiki/BCILAB

[7] http://www.sensibilab.lecco.polimi.it/

[8] http://www.tobi-project.org/

[9] http://sourceforge.net/projects/xbci/

[10] http://www.brainterface.com/

[11] http://bbci.de/pyff/index.html

[12] http://www.asterics.eu/

[13] http://www.eldish.net/hp/automat/toolb_mb.htm

[14] https://www.monografias.com/trabajos5/matlab/matlab.shtml

[15] https://www.genbeta.com/desarrollo/eclipse-ide

[16] http://www.jfree.org/jfreechart/samples.html

[17] https://www.eclipse.org/windowbuilder/