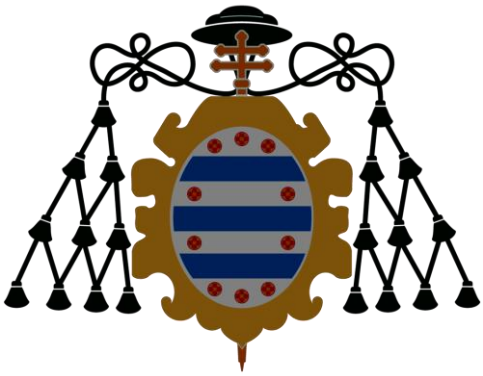

Universidad de Oviedo

Memoria del Trabajo Fin de Máster realizado por

José Ignacio de la Hera Cibrián

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

Desarrollo de algoritmos de análisis inteligente, visualización de datos y procesamiento de señales neurofisiológicas para estudio de enfermedades neurodegenerativas

Dirigido por

Antonio Miguel López Pérez
José Ignacio Serrano

Gijon 2018

Todos somos aprendices en un arte
donde nunca nadie se convierte en maestro

Ernest Hemingway

Resumen

El proyecto consiste en desarrollar un programa que permita procesar de diferentes maneras archivos generados por un sistema de adquisición de señales neurológicas. El sistema de adquisición empleado es el EEG, comúnmente conocida como electroencefalografía. Esta técnica solamente tiene la posibilidad de medir las señales de la corteza cerebral, las cuales se asocian con las capacidades motoras del cerebro humano.

El objetivo es limpiar las ondas medidas al máximo posible, para obtener las partes de la señal producidas por el sujeto de estudio de manera consciente en las tareas solicitadas, descartando en la medida de lo posible los componentes o artefactos producidos por el paciente, ya sea de manera involuntaria o inconsciente.

Una vez obtenida la señal limpia se procede a obtener y estudiar los ERD (*Event related desinchronization*) y ERS (*Event related synchronization*) asociados a los movimientos realizados por el paciente.

Esta sería la finalidad principal del sistema, de manera que los ERD y ERS puedan ser estudiados por personal con conocimiento experto del campo de la neurología.

Para conseguir limpiar la señal se van a aplicar técnicas de procesamiento digital de la señal y de inteligencia artificial, dando al usuario la opción de elegir la que considere más adecuada para en cada momento.

De esta manera, tampoco sería necesaria la aplicación de todas las técnicas implementadas, ni de aplicarlas en un orden concreto, siempre y cuando, se respeten las limitaciones existentes que imponen algunas de las técnicas.

Palabras clave: ERD, ERS, enfermedades neurodegenerativas, Parkinson

Agradecimientos

He de decir que en este documento figuro como autor, pero no soy su único participe. De esta manera, hay diferentes colectivos o personas sin los cuales este trabajo tal y como es no habría sido posible. Quizás hubiese sido algo parecido, realizado por otra persona, o quizás no existiría ninguno, pero en cualquiera de los casos no sería el documento actual.

Al CSIC, al grupo de GNEC, y especialmente a María Dolores del Castillo y José Ignacio Serrano, por enseñarme algo de lo que nunca pensé que sabría.

A la Universidad Complutense, por ayudarme en el largo camino que he recorrido hasta donde he llegado, y especialmente a Marco Antonio Gómez, por ayudar a un antiguo alumno de la facultad cuando más perdido andaba. Aunque dejemos de ser alumnos, no quiere decir que lo sepamos todo.

A la universidad de Oviedo, por lo que he aprendido con ellos. En particular a Antonio Miguel López, por ayudarme con este proyecto.

Y por último, a mi familia y entorno. Por estar ahí a pesar de no comprender las elecciones realizadas. Hasta yo he llegado a dudar de ellas, pero son estas las que me han llevado a ser como soy.

Índice de figuras

Índice General

1 Introducción

Este proyecto es una herramienta para limpiar y procesar digitalmente señales analógicas que provienen de la grabación de las activaciones neuronales de un cerebro humano. El objetivo es crear un sistema capaz de realizar dicho procesamiento, dando la posibilidad de estudiar los resultados a personal especializado, el cual sería el encargado de obtener conclusiones en base al comportamiento de las activaciones neuronales que calcule el programa a partir de la señal procesada.

Para poder hacer todo esto, se ha creado un proyecto en C++, complementando las carencias del lenguaje mediante las bibliotecas de Armadillo y Matlab, los cuales dan a C++ una gran potencia a la hora de utilizar álgebra lineal, y el uso de SigPack como biblioteca para creación de los filtros de frecuencia.

A la hora de completar el proyecto con interfaces gráficas se ha elegido QT para el diseño de las ventanas y el control de eventos.

1.1 Motivación

La temática del proyecto aborda temas de investigación y bioingeniería, de manera que se intenta mediante ellas obtener información acerca de cómo se comportan ciertas enfermedades, pudiendo de esta manera tanto obtener más información para combatirlas, como obtener datos acerca de cómo el tratamiento está afectando al paciente, y, de esta manera evaluar la efectividad del mismo.

No obstante, aunque hay muy buenas investigaciones en algoritmos en esta área, no existen implementaciones de código abierto o gratuitas, de manera que no hay un acceso generalizado a ellas.

Otro problema existente, es que las principales implementaciones están planteadas para utilizarlas como plugins descargables dentro de la toolbox eeglab en Matlab. Es comprensible que esto sea así dado la facilidad que aporta este lenguaje a la hora de trabajar con álgebra lineal, pero aporta ciertos problemas que dificultan el acceso, como son la necesidad de comprender Matlab, la necesidad de conocer cómo gestionar sus toolbox, el uso de plugins en la toolbox, una necesidad de licencia de Matlab o una menor velocidad de computo respecto a otros lenguajes.

La mayoría de estos puntos son críticos en el momento en el que una persona sin experiencia técnica tiene que trabajar con estos métodos, de manera que se pretende simplificar el acceso a ellos de manera que puedan acceder a esos sistemas sin

necesidad de perderse en aspectos técnicos, y evitando el pago de licencias para el acceso a un entorno muy concreto.

Por otra parte, para una investigación es necesario utilizar múltiples pruebas y pacientes. Si tenemos en cuenta que los algoritmos planteados no son inmediatos, y que van a necesitar aplicarse reiteradamente, parece también necesario que el sistema tenga la mayor velocidad de cómputo posible, tanto para reducir esperas por parte del personal, como para permitir obtener más datos en el mismo tiempo.

1.2 Objetivos

El objetivo principal es implementar una herramienta intuitiva y fácil de usar para los investigadores, incluyendo en ella las técnicas de procesado digital de la señal más comunes en el entorno de las señales neurológicas.

En este ámbito de trabajo existen diversas aproximaciones a los datos y diversas técnicas que se pueden aplicar, según los marcadores que se quiera estudiar. En este proyecto, esos marcadores son los ERD y ERS. Estos se explican de manera más detallada en el apartado 2.1.

Para ello, se consideran tres subobjetivos principales:

- Definir una estructura de datos que pueda contener toda la información relevante de las señales almacenadas en los archivos, independientemente de cual sea el formato del archivo del que proceda (GDF, DAT).

- Crear un sistema de carga de archivos binarios que realice correctamente las tareas de lectura de uno de los tipos de archivos propuestos, aportando una interfaz de trabajo para que sea simple ampliarlo a más en el futuro.

- Completar los diferentes algoritmos solicitados, ya sea mediante implementación propia, o mediante el apoyo en bibliotecas especializadas, de manera que se puedan obtener correctamente los ERD y ERS del paciente.

Además, existen también necesidades en el ámbito de la organización, la visualización y los resultados:

- Se considera necesario implementar una interfaz simple e intuitiva, de manera que asista al especialista, simplificando su trabajo y aportando una curva de aprendizaje lo más simple posible.

- Aportar una visualización clara y simple, que permita al usuario comprender los datos del paciente lo más rápidamente posible, ahorrando así tiempo al experto, de manera que este pueda ver claramente el comportamiento que tienen los ERD y ERS del paciente.

1.3 Estructura de la memoria

La estructura del documento es la siguiente:

- Introducción: Se realiza una breve introducción al proyecto, junto con el planteamiento de los objetivos a conseguir a lo largo del mismo.

- Fundamentos de neurociencia: Dado que este documento esta inicialmente enfocado como un proyecto de ingeniería, se ha considerado necesario añadir una breve explicación de los principales aspectos del comportamiento tanto de una neurona humana como del cerebro en general. Asimismo, se comenta el modo de grabación de los experimentos, de manera que se conozcan los diferentes elementos que están grabados en los archivos que se procesan, independientemente del formato con el que estén grabados.

- Metodología: Se comenta la metodología de trabajo utilizada, así como la planificación de los requisitos del proyecto.

- Trabajo previo y tecnologías utilizadas: Se comenta el estado del proyecto a la hora de empezarlo, las diferentes tecnologías que se han utilizado, y proyectos similares a este.

- Estructuras de datos y formatos de los archivos: Se comentan los principales formatos de archivos utilizados para almacenar este tipo de señales, así como la solución propia para mantener los datos de los archivos en el sistema, y como se cargan estos datos.

- Algoritmos y funcionalidades propuestas: Se comentan las diferentes técnicas y algoritmos que se utilizan en este tipo de sistemas y serán incorporadas al programa.

- Manual de usuario: Se adjunta un pequeño manual acerca del funcionamiento de la herramienta y como trabajar con ella.

- Instalación, importación y ampliación del proyecto: Se dan las pautas para poder desplegar la herramienta sobre una nueva máquina partiendo de un ejecutable compilado, como importar el proyecto a Visual Studio para poder trabajar con él, y una explicación acerca de cómo está estructurado el programa para que se pueda ampliar en un futuro.

- Discusión y trabajo futuro: se realizan comentarios acerca del desarrollo del proyecto, el grado de consecución de los objetivos y el trabajo pendiente de realizar para futuras mejoras.

2 Fundamentos de neurociencia

Dado que el proyecto tiene un origen y un fin basado en elementos biológicos, englobados en un proyecto de ingeniería, se considera necesario profundizar en cierta manera, en la base de funcionamiento de dichos elementos. Se consideran dos áreas de explicación diferenciadas. La primera, dirigida a la biología, organización y funcionamiento del cerebro, de manera que se tenga una visión general de su funcionamiento, y la segunda, la rutina de trabajo a la hora de grabar las respuesta neuronales de los pacientes, de manera que se entiendan ciertos elementos en cuya existencia se basan algunas de las aplicaciones de los algoritmos que se van a implementar.

2.1 Introducción al comportamiento del cerebro

El proyecto se centra en los estímulos que se transmiten de neurona a neurona a lo largo del cerebro.

Si nos fijamos a nivel de todo el sistema nervioso, los estímulos se van transmitiendo entre sus diferentes neuronas. Esta transmisión de información se realiza a través de los axones hacia las dendritas de las neuronas con las que tienen conexión. Los métodos de transmisión de la información pueden ser, mediante reacción química en el caso de las neuronas con más distancia, o mediante descargas eléctricas de voltaje reducido en el resto de conexiones. En el caso del cerebro, se utiliza el segundo tipo [1].

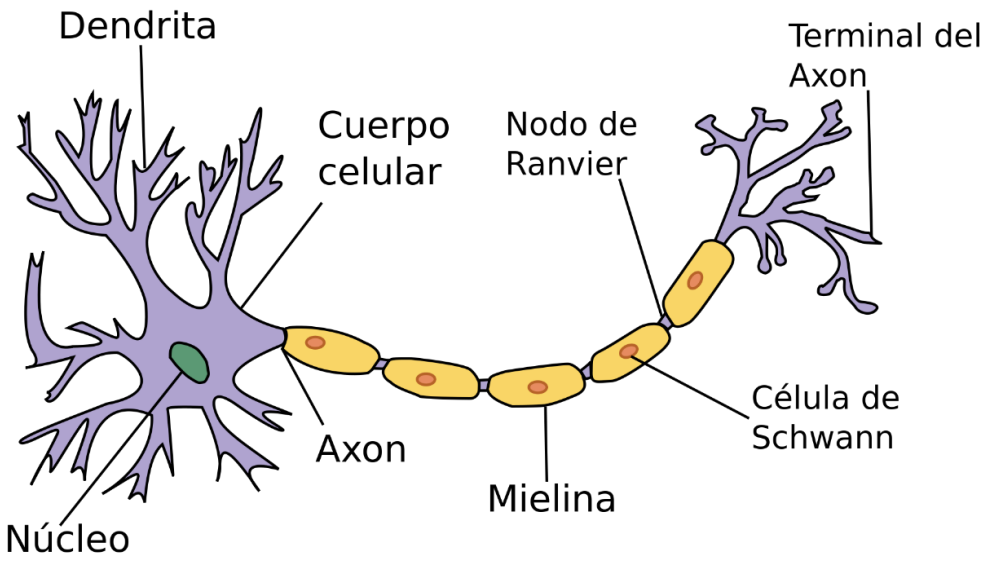

Ilustración 1: Diferentes partes de una neurona

El potencial eléctrico del axón de una neurona, que sería el elemento que define su valor ante un sistema de medida, mantiene un comportamiento ondulatorio mientras están activos sus estímulos de entrada, provenientes de los axones de las neuronas que conectan con sus dendritas. Esto es debido a que la comunicación entre neuronas se basa en un sistema de carga y descarga. Primero la neurona espera a estar cargada al máximo posible y después propaga la electricidad a través del axón, la cual va viendo reducido su potencial a la vez que la neurona se va preparando para una nueva descarga. En caso de necesitar aumentar la frecuencia de descarga, los tiempos de carga, y por consiguiente, los voltajes descargados, serán menores.

Estos estímulos eléctricos, a pesar de tener un voltaje reducido, pueden ser medidos mediante sistemas de adquisición de señales, siendo posteriormente amplificados y grabados mediante sistemas de adquisición EEG.

Ahora bien, una neurona tiene múltiples conexiones de entrada y por ella pasan multitud de estímulos. No son elementos dedicados. Esto implica que hay muchos elementos que afectan al comportamiento de una neurona, y no es un elemento que mantenga su frecuencia fácilmente. Además, las neuronas son de un tamaño reducido, de manera que un electrodo de medida abarca el voltaje existente en una zona, la cual engloba una gran cantidad de neuronas.

En este caso, esto se refleja en el sistema estándar 10-20, el cual establece tanto la localización como el nombre de los diferentes canales, electrodos o puntos de medida que va a tener el sistema de adquisición de señales.

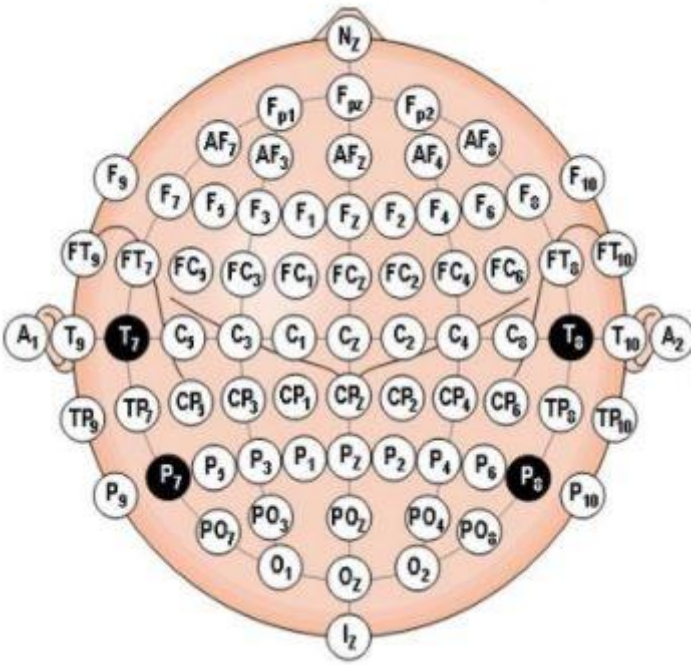

Ilustración 2: Localización y nombre de los electrodos de medida en los experimentos realizados. Estándar 10-20

Una vez comprendido el sistema de comunicación entre neuronas, y como se organizan los electrodos del sistema de medida, existen ciertos elementos a tener en cuenta en el cerebro, relevantes en el momento en el que se pasa a estudiar la comunicación entre neuronas a el comportamiento más general entre grupos de neuronas.

Atendiendo a este sistema de comunicación, la actividad en un canal medido se puede separar en diferentes bandas de actividad: [2]

- delta (0-4 Hz)
- theta (4-8 Hz)
- alfa (8-12 Hz)
- beta (12-30 Hz)
- gamma (30-70 Hz)

Aparte de estas bandas de frecuencia, en el cerebro hay frecuencias superiores, aunque estas no se suelen tener en cuenta en el análisis clínico, y menos en el análisis de la actividad motora, ya que las frecuencias típicas en este tipo de análisis son hasta los 40 Hz. No obstante se listan a continuación.

- ~60~120 Hz, con origen en las retinas.
- ~250 Hz, ondas del hipocampo.
- ~600 Hz, descargas talamocorticales

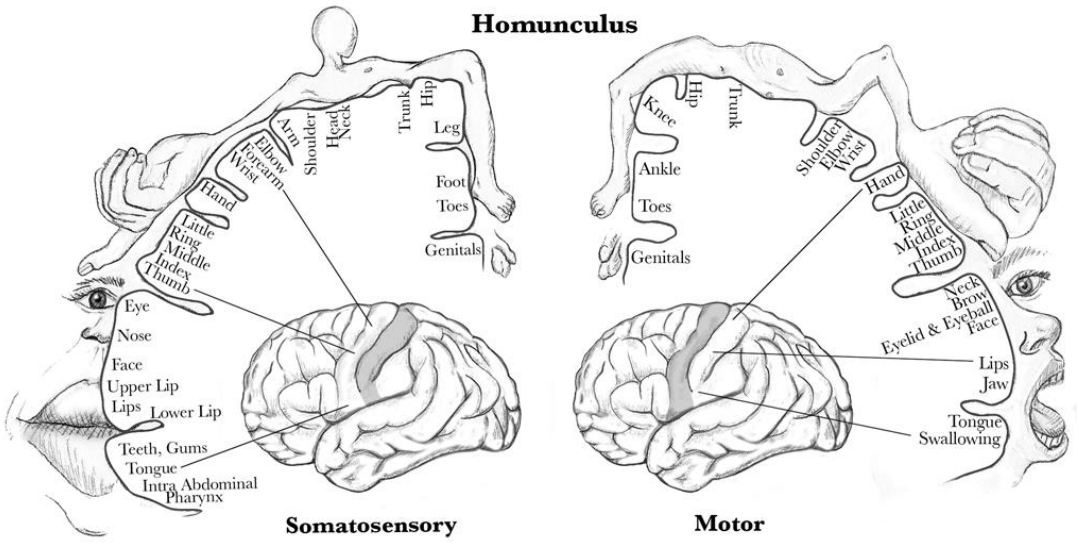

Ilustración 3: Sección del cerebro que relaciona diferentes áreas con las partes del cuerpo que controlan

Otro concepto que resulta conveniente mencionar es la organización del cerebro, aunque solo se tratará la zona motora, que es en la cual se ha centrado el proyecto.

En la imagen se puede ver varios puntos del cerebro que se asocian con las diferentes partes de las que se encargarían de mover. Esto se puede enlazar también con la imagen del estándar 10-20 en la que se plantean los puntos de medida del sistema de adquisición de la señal, de manera que se pueda conocer en que canales se manifestara más un tipo de movimiento según la parte del cuerpo a la que afecte.

No obstante, esto es un esquema general, las partes no están firmemente definidas. Ejemplo de esto son los pacientes que han sufrido un ictus, los cuales pueden perder zonas del cerebro debido a que las neuronas en esa área han muerto por culpa de un infarto cerebral. En estos casos, y gracias a la plasticidad del cerebro, este se reajusta, dando las funciones perdidas a una zona cercana, reorganizando así las funciones de cada área. [3] [4]

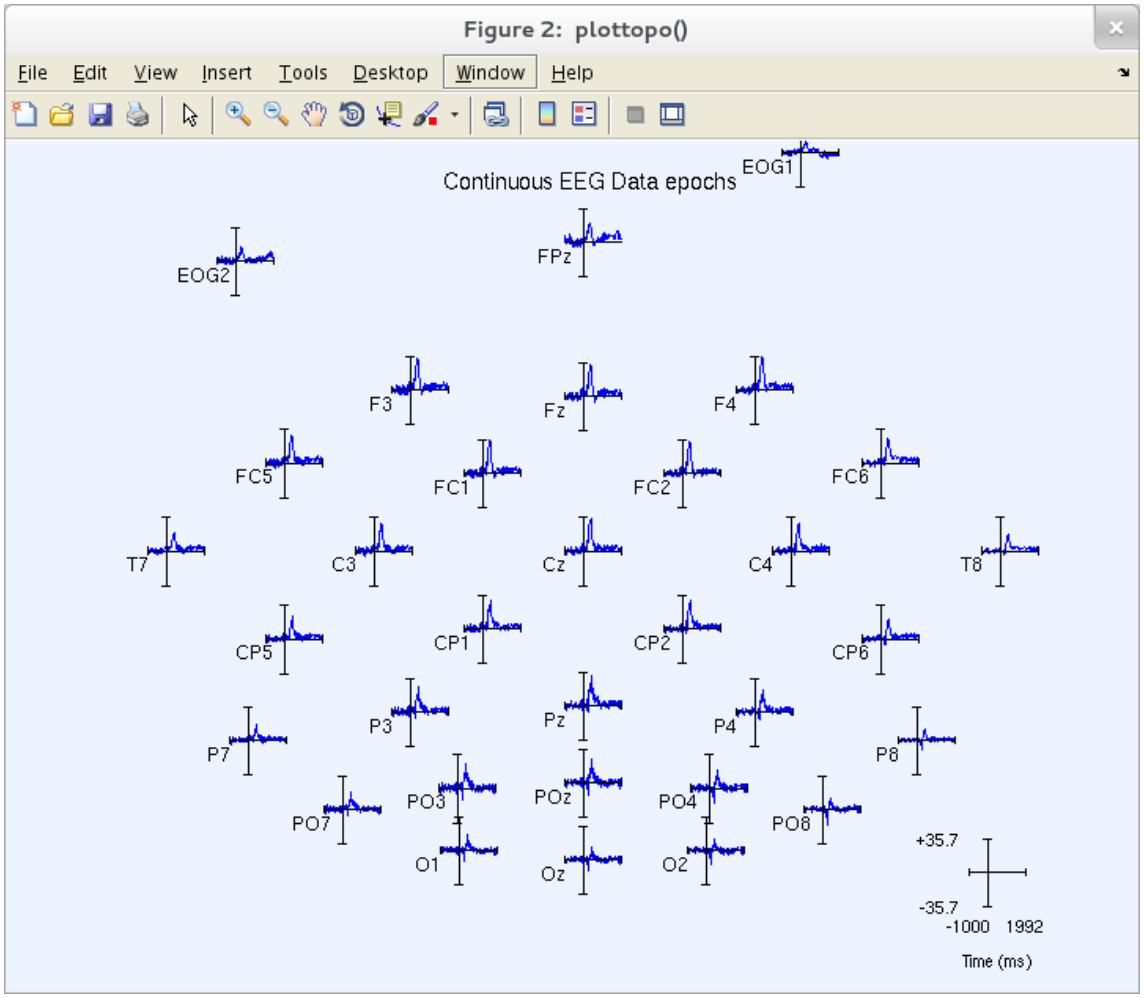

Ilustración 4: Variación de potencial con respecto del tiempo para cada uno de los canales de medida durante un estímulo concreto

Además de por la equivalencia de las zonas con la posición de los electrodos del sistema de adquisición, se explica esto para manifestar que cada parte del cerebro es autónoma, de manera que, aunque una acción se puede manifestar a lo largo de todo el cerebro, el área en el que más se va a manifestar es en el área relativo al músculo que se ha movido.

Otra implicación de esto es, que si se realizan dos movimientos simultáneos, cada uno de ellos puede producir ruido en la medida del otro estimulo, lo que se denominaría un artefacto.

Un ejemplo de esto se puede ver en la figura adjunta, en la cual se aprecia que, aunque el estímulo principal está localizado en la zona central del cerebro, los electrodos más externos también detectan el estímulo, aunque con menos fuerza.

Por último, una vez comprendida la actividad, frecuencias y organización del cerebro, es importante conocer los conceptos de ERD y ERS. [5] [6] [7]

ERD es el acrónimo de *event related desynchronization*, o desincronización debido a un evento.

La clave para localizar este fenómeno está en analizar el comportamiento de las neuronas que están en la banda alfa, la cual está asociada al reposo del cerebro. Esto es así debido a que el ERD está asociado a la preparación y realización de un movimiento o actividad motora, de manera que la banda alfa reduce en gran medida la cantidad de neuronas que oscilan en su banda de frecuencia. Estas neuronas se van a la frecuencia necesaria para que se activen y muevan los músculos necesarios para la acción, de manera que el potencial en la banda alfa se reduce.

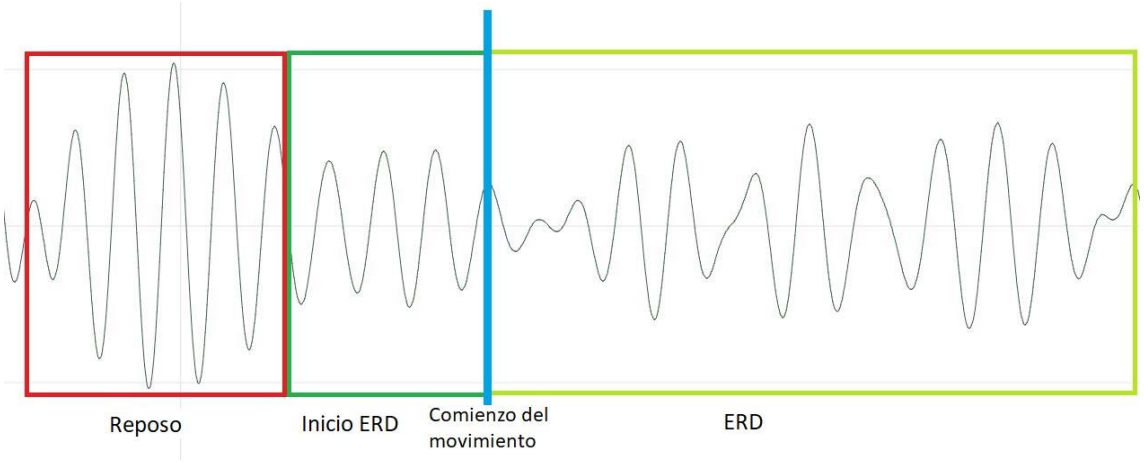

Ilustración 5: Variación del potencial eléctrico en la banda alfa (8-12 Hz) y diferentes eventos asociados al comienzo del movimiento

Este fenómeno es especialmente visible al preparar el movimiento, ya que es cuando hay una mayor ausencia de neuronas en la banda alfa. Al iniciar el movimiento, algunas

de las neuronas que habían cambiado de frecuencia vuelven a la banda alfa, pero la mayoría se mantiene fuera hasta que acabe el movimiento, desencadenando así un ERS.

ERS es el acrónimo de *event related synchronization* o sincronización debido a un evento.

Este fenómeno es posterior a un ERD, comenzando en el momento en el que se acaba el movimiento.

Lo que permite localizar este acontecimiento es encontrar el momento en el que las neuronas van volviendo a la banda alfa del cerebro, marcando así que el movimiento ha finalizado, y recuperando el potencial en la banda alfa perdido por el ERD.

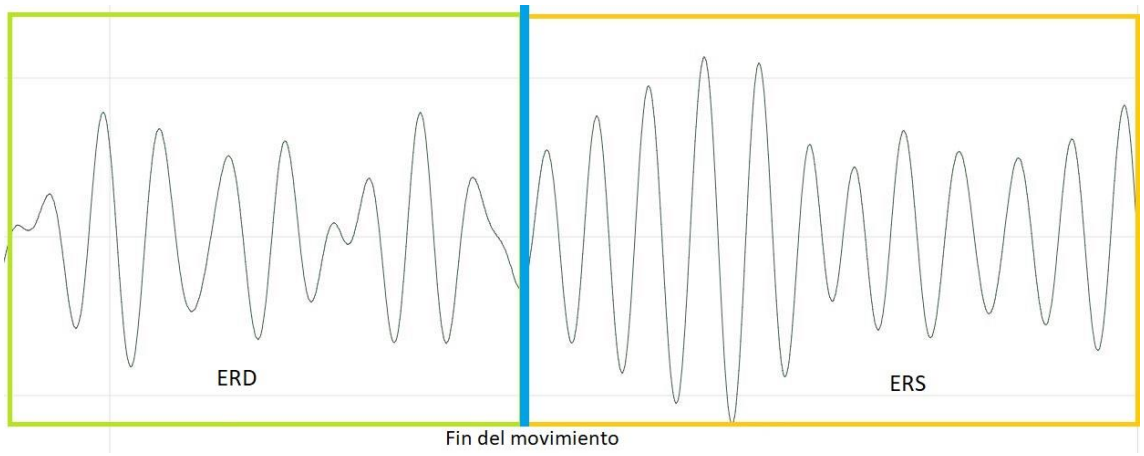

Ilustración 6: Variación del potencial eléctrico en la banda alfa (8-12 Hz) y diferentes

Un ejemplo práctico de esto sería estudiar el área del cerebro encargada de los brazos al dar una palmada.

Inicialmente esa área del cerebro, está en reposo y con mucha actividad en la banda alfa.

Milisegundos antes de que se comience a mover ningún musculo, las neuronas ya se están preparando para cuando tengan que moverlos. Aquí comenzaría un ERD localizado en esta área del cerebro. Este sería el momento en el que mejor se notase, ya que es cuando menos neuronas hay en la banda alfa.

Posteriormente, se comienzan a mover los músculos de los brazos. A lo largo de toda esta acción, el ERD va a seguir activo, pero la banda alfa va a recuperar algunas neuronas al comenzar el movimiento, de manera que se va a notar algo peor que al principio del ERD.

Por último, cuando se dejan de mover los brazos, las neuronas vuelven a sincronizarse en la banda alfa, volviendo a la frecuencia de reposo, identificándose aquí un ERS.

Solamente queda decir, que estos eventos ERD y ERS van juntos, y que son marcadores importantes a la hora de comprobar la manera de reaccionar por parte del cerebro de los pacientes, siendo el comportamiento de estos la información que se quiere encontrar. Esto es así debido a que pueden ser utilizados para asistir a expertos en el diagnóstico, para investigar el alcance de la enfermedad, para estudiar su evolución, o para estudiar el efecto de tratamientos que afecten a la corteza motora del cerebro.

2.2 Rutina de los experimentos

En un experimento, se busca obtener unos datos que permitan un correcto procesamiento, y un estudio posterior de los datos.

Cada experimento consta de múltiples repeticiones o *epoch* de una actividad concreta, y con diferentes eventos dentro del *epoch*. A continuación se explica mediante un ejemplo la rutina de un *epoch* de ejemplo. Nótese que es un ejemplo, cualquier otra grabación puede tener diferentes eventos, diferente movimiento, o incluso diferente estructura de los eventos.

Como ejemplo, se comenta el proceso de grabación de un experimento de *wrist extension*, o extensión de muñeca, lo cual consiste en mantener el brazo en posición horizontal y levantar la mano utilizando la muñeca.

Como condiciones iniciales, para la grabación, es necesario que el paciente este todo lo cómodo posible. El experimento puede durar algunos minutos, y un paciente cansado puede realizar otras acciones debidas a la incomodidad o la impaciencia. Cuanto más quieto este, mejor resultado puede dar la grabación.

En este caso, el paciente está sentado frente a una pantalla, con el brazo extendido sobre la mesa y con el codo en ángulo de 90 grados.

Se inicia la grabación, y el usuario se mantendrá en la posición inicial.

Pasados 5 segundos después de comenzar, en la grabación se marca en los datos el evento 1, en este caso, que comienza el *epoch*.

Pasados 5 segundos se marca en los datos el evento 2, dando paso a un *feedback* en pantalla para el paciente, al cual avisa mediante una cuenta atrás de 3 segundos el tiempo restante para el siguiente evento.

Pasado el tiempo de la cuenta atrás se marca en los datos el evento 3, dando un nuevo *feedback* al usuario para que haga la extensión de muñeca.

Pasados 2 segundos, se marca en los datos el evento 4, y se indica por pantalla al usuario que puede devolver la muñeca a la posición original para descansar.

Pasados 5 segundos para que descanse la muñeca, se marca en los datos el fin del evento.

Una vez acabada la grabación del *epoch*, se esperan 5 segundos para marcar el comienzo del siguiente, pero sin parar de grabar.

De esta manera se graban varios *epoch* seguidos, a lo largo del tiempo que dura el experimento, teniendo todos ellos los eventos comentados, manteniendo la misma numeración, de manera que, a lo largo de todo el experimento, el evento 3 signifique el comienzo de un *wrist extension*.

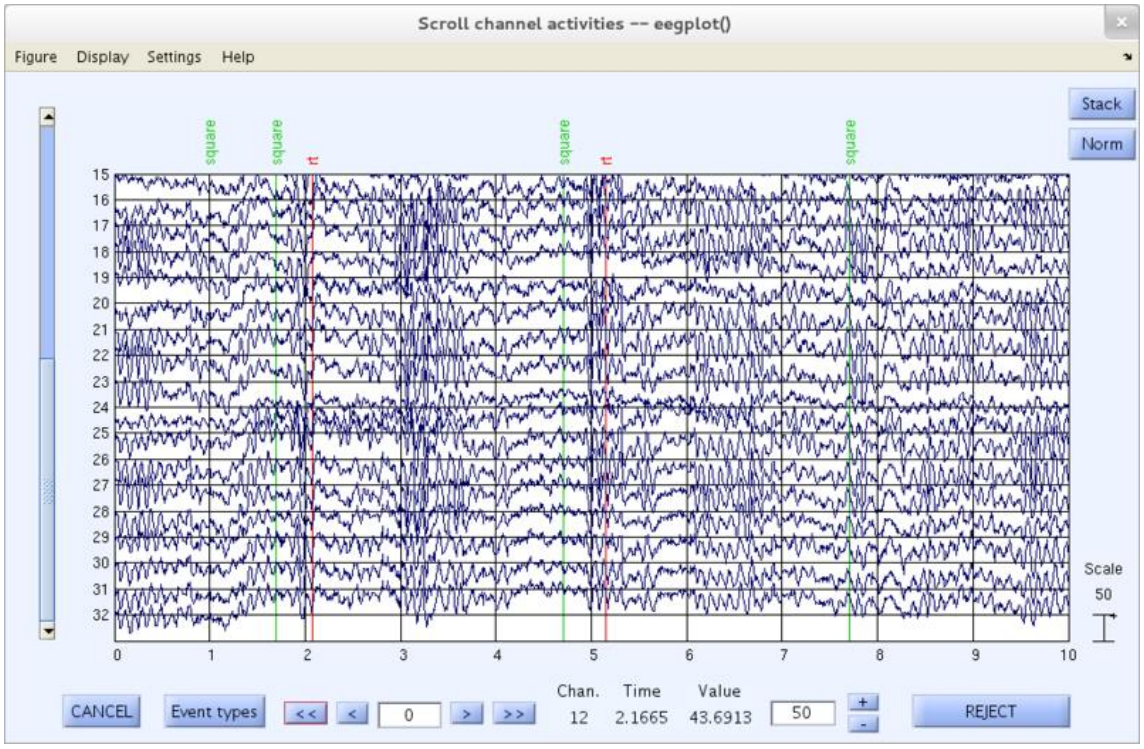

Ilustración 7: Actividad medida en un experimento. Mediante líneas horizontales verdes y rojas se pueden ver los eventos ocurridos durante la grabación

3 Metodología

A continuación se va a hablar acerca de la metodología que se ha seguido, y como se ha adaptado a las necesidades especiales del proyecto.

3.1 Definición y adaptación de la metodología XP

El XP o extreme programming es una metodología ágil de desarrollo de software, en la cual al cliente se le permite ir haciendo modificaciones incrementales en los requisitos del software en cada reunión que se realiza. Así se van definiendo o acotando nuevos requisitos a medida que se van completando los que se han planteado en reuniones anteriores. [7] [8]

El acuerdo al que se llega con el cliente en estas reuniones es el plan de iteración. Partiendo de las características deseadas por el cliente para completar para la próxima reunión, los programadores se encargan de trocear estas características, llegando a un acuerdo con el cliente de cuales se van a implementar para la siguiente versión.

Esto permite a los programadores centrarse en un requisito o unas características en concreto para las que ya se ha llegado a un acuerdo.

No obstante, es necesario que entre reunión y reunión, las características sobre las que se ha realizado el acuerdo hayan avanzado o se hayan finalizado, permitiendo generar una versión estable del proyecto, de manera que el cliente siempre pueda ver avances, y pueda expresar o no su conformidad.

En el caso de estudiar la adaptación de esta metodología al proyecto, tendríamos las siguientes características.

Los roles serían los siguientes:

- Cliente: María Dolores del Castillo
- Definición de requisitos: José Ignacio Serrano
- Programador y jefe de proyecto: José Ignacio de la Hera
- Tester: José Ignacio de la Hera

Por lo general, extreme programming plantea que el tiempo entre cada plan de iteración son dos semanas.

En este caso, dado que los requisitos que han ido surgiendo estaban bien definidos y aislados entre sí, estos han sido el objetivo de cada uno de los planes de iteración, en lugar de trocearlos en diferentes características e implementar el requisito en varias iteraciones. Esto implica que las reuniones para cada iteración se han espaciado en el tiempo mientras que el requisito era completado en lugar de las dos semanas recomendadas.

No obstante, parte de los motivos que plantea extreme programming para realizar reuniones tan frecuentes con el cliente es para mantenerlo informado del avance y que vea los resultados.

Para eso ha habido múltiples reuniones de revisión y dudas dentro del desarrollo del plan de iteración entre las diferentes partes del proyecto.

3.2 Definición de requisitos

Los requisitos que han surgido a lo largo del proyecto, se han trabajado en ellos, y se han mantenido como relevantes en el último plan de iteración son los siguientes:

- Estudio del problema [Req01]

Aunque este no sería un requisito en si del proyecto, el estudio y comprensión del problema es necesario para que el ingeniero conozca las señales con las que se va a trabajar, sus características y la finalidad del proyecto, de manera que se familiarice con las propiedades y restricciones existentes en el trabajo.

- Estructuras de datos [Req02]

A la hora de poder trabajar con el proyecto, una de las primeras cosas que es necesario realizar es la definición e implementación de una estructura de datos que englobe toda la información que se pueda utilizar en él. Esto encajaría con el diseño del modelo de los datos en un diseño MVC.

- ASR [Req03]

Estudio, implementación y pruebas del algoritmo ASR. La funcionalidad de este algoritmo es la corrección y eliminación de picos y descargas inesperadas producidos en la señal.

- Filtros frecuenciales [Req04]

Estudio, implementación y pruebas de filtros frecuenciales. La funcionalidad de esta técnica es la separación de la actividad en diferentes frecuencias para poder estudiarlas por separado.

- Promediado [Req05]

Estudio, implementación y pruebas del algoritmo de promediado basado en la elección de eventos sobre los que promediar.

- MARA [Req06]

Estudio, implementación y pruebas del algoritmo MARA. La funcionalidad de MARA es la localización y eliminación de las componentes de la señal que se han catalogado como ruido.

- CAR [Req07]

Estudio, implementación y pruebas del algoritmo CAR. Este es un algoritmo es normalizar la señal, así como ayudar a detectar fuentes de señales muy pequeñas en entornos ruidosos.

- Desarrollo de Interfaces gráficas [Req08]

Diseño e implementación de la interfaz del proyecto. Esta parte permitiría al personal no especializado trabajar con la herramienta, cumpliendo la parte de vista en la técnica de diseño MVC.

3.3 Planificación de requisitos

En la planificación se aporta una aproximación a como se van a afrontar los requisitos previos, así como una estimación de tiempo de trabajo en cada uno de ellos.

3.3.1 Estudio del problema [Req01]

Se considera necesario que el programador tenga conocimientos en ciertas áreas, de manera que se destinará tiempo para que se forme en diferentes aspectos:

- Neurología

Estudio del funcionamiento del cerebro mediante el análisis de diversos artículos de investigación.

Los temas tratados son los fundamentos de neurología y el funcionamiento de los marcadores ERD y ERS que se van a buscar.

- Procesamiento digital de la señal

Estudio de diversas técnicas de procesamiento digital de la señal.

- Herramientas de trabajo comunes en procesamiento de datos EEG

Estudio de la herramienta EEGLAB, así como pequeñas pruebas de funcionamiento.

- Funcionamiento de los experimentos y del sistema de medida

Explicación por parte de investigadores del procedimiento empleado en los experimentos realizados.

Distribución y funcionamiento de los electrodos de medida.

Tiempo estimado, seis semanas.

3.3.2 Estructuras de datos [Req02]

Creación de una estructura de datos correcta para almacenar todos los datos de un experimento y sus sucesivas modificaciones mediante los algoritmos planteados.

Para cumplirlo, se plantea estudiar los datos comunes entre las diferentes características y formatos de archivos EEG. De esta manera, se puede estructurar correctamente un modelo de datos, abstrayendo y modularizando diversas partes mediante la programación orientada a objetos.

Este requisito se plantea cumplir mediante implementación en C++, junto con bibliotecas de álgebra lineal y tipos abstractos de datos.

Tiempo estimado, dos semanas.

3.3.3 ASR [Req03]

Creación de un algoritmo de limpieza de picos y perturbaciones mediante la comparación de una muestra limpia de los datos del paciente con los datos medidos. En el caso de encontrar una zona con una desviación estándar superior a la existente en los datos de muestra, el sistema debe tomarlos como datos perdidos, reconstruyendo la señal utilizando la muestra limpia.

Este requisito se plantea cumplir mediante el estudio de la documentación disponible, así como el estudio, adaptación y modificación de ejemplos existentes en Matlab, pasando posteriormente a una implementación del código basada en C++ y bibliotecas de álgebra lineal.

Tiempo estimado, diez semanas.

3.3.4 Filtros frecuenciales [Req04]

Implementación de un sistema de diseño de filtros frecuenciales a partir de las frecuencias y el tipo de filtro deseado por parte del usuario.

Este requisito se plantea cumplir mediante desarrollo en C++ junto con el trabajo con bibliotecas de álgebra lineal y de procesado digital de la señal.

Tiempo estimado, dos semanas.

3.3.5 Promediado [Req05]

Implementación de un sistema de promediado en base a la localización de los eventos seleccionados por el usuario, así como el tiempo a utilizar por el algoritmo.

Este requisito se plantea cumplir mediante el desarrollo en C++ junto con el uso de bibliotecas de álgebra lineal.

Tiempo estimado, dos semanas.

3.3.6 MARA [Req06]

Implementación de un sistema que extraiga una serie de características de las componentes independientes de los datos existentes.

En base a estas características el sistema debe clasificar las componentes para mantenerlas o borrarlas, reconstruyendo la señal con las componentes mantenidas.

Este requisito se plantea cumplir mediante el estudio y desarrollo en Matlab y migración a C++, junto con el uso de bibliotecas de álgebra lineal e inteligencia artificial.

Tiempo estimado, ocho semanas.

3.3.7 CAR [Req07]

Algoritmo empleado para normalizar los datos, de manera que sea posible compararlos entre diferentes usuarios, así como descubrir y aumentar pequeñas fuentes de señal escondidas en los datos.

Este requisito se plantea cumplir mediante una implementación el estudio de la documentación del algoritmo junto con una implementación en C++ junto con bibliotecas de álgebra lineal.

Tiempo estimado, tres semanas.

3.3.8 Interfaces gráficas [Req08]

Desarrollo de una interfaz que permita trabajar con los diferentes algoritmos y ver los resultados a nivel gráfico, simplificando así el trabajo del personal especializado.

Este requisito se plantea cumplir mediante desarrollo en C++ con ayuda de bibliotecas gráficas.

Tiempo estimado, cuatro semanas.

3.4 Diagrama de Gantt estimado

El diagrama de Gantt representa una estimación del gasto del tiempo por parte de los programadores en las diferentes tareas.

Si seguimos los tiempos definidos en la planificación, quedaría un diagrama como el siguiente, solapándose ligeramente etapas en las que se cree que puede trabajar simultáneamente.

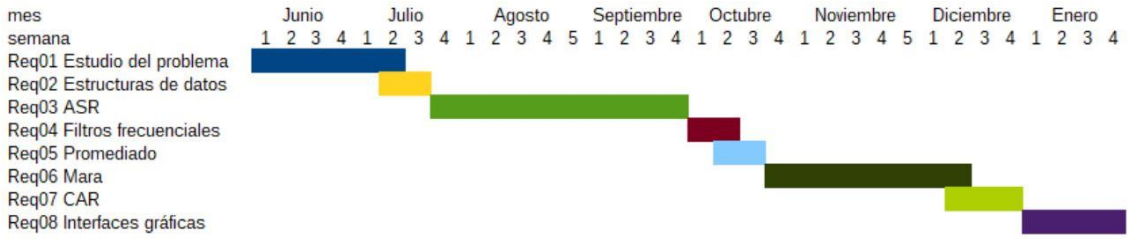

Ilustración 8: Diagrama de Gantt planificado para el proyecto

4 Trabajo previo y tecnologías utilizadas

En el proyecto se integran diversas tecnologías para conseguir un entorno funcional y con la posibilidad de realizar las tareas propuestas como objetivos. Como camino hasta estas, también se ha pasado por el estudio de otras, descartándolas por diversos motivos. Aparte de esto, hay otros elementos que se han utilizado, para comprender mejor donde se quería llegar, o para comprobar la evolución del proyecto.

De esta forma, y para facilitar la lectura y posterior búsqueda, se realiza su descripción atendiendo a varios grupos, siendo estos los lenguajes utilizados, los programas usados durante el proyecto, las bibliotecas y componentes estudiados, el trabajo previo, y por último, otras tecnologías.

4.1 Trabajo previo en el proyecto

El proyecto se engloba en un marco de investigación, en el cual se tenía claro lo que se buscaba con este trabajo, de manera que se aportó una serie de tareas a conseguir, extraídas de las ideas y algoritmos que se planteaban a lo largo de artículos científicos de la correspondiente temática, junto con técnicas comunes en el procesamiento de datos.

Aparte de esto, se aportaron archivos con grabaciones de los sistemas de adquisición de señal con los que probar la corrección de los sistemas creados, así como ciertos programas usados comúnmente para abrir y trabajar con estos archivos.

No obstante, no se disponía de código ni de aportaciones previas. En ese sentido, se ha partido de cero.

4.2 Lenguajes

Actualmente existen múltiples lenguajes de programación, cada uno con sus características y complejidades. No obstante, las necesidades del proyecto delimitan mucho las opciones a elegir, permitiendo solamente lenguajes rápidos, y con una alta orientación a entornos de escritorio.

4.2.1 C++ 11

C++ es un lenguaje de programación de propósito general basado en el lenguaje C. Soporta programación imperativa, orientada a objetos y características generales de programación.

C++ es un lenguaje potente y de alto rendimiento utilizado comúnmente para construir sistemas operativos, motores de videojuegos y aplicaciones de escritorio.

4.2.2 Matlab

Matlab es un lenguaje de programación desarrollado por MathWorks. Esta entre los principales entornos de desarrollo de software para ingenieros y científicos.

Comenzó como un lenguaje de programación de matrices donde el álgebra lineal fuese simple. No obstante, el lenguaje ha ido evolucionando, y las bibliotecas y funcionalidades incluidas en él también lo han hecho, dando soporte a procesamiento de imágenes, procesamiento digital de la señal, visión por computador y muchas otras.

4.2.3 Python

Python es un lenguaje de programación de alto nivel ampliamente usado. Es un lenguaje de propósito general simple y legible con multitud de bibliotecas implementadas, lo cual da al lenguaje valor añadido, entre ellas, multitud de herramientas para proyectos científicos como SciPy.

De entre los 3 lenguajes, se ha decidido basar el proyecto en C++, ya que algunos de los algoritmos propuestos son lentos al ejecutarse, y este es el que mejor rendimiento puede aportar. Además, también aporta buenas herramientas a la hora de trabajar en la interfaz gráfica.

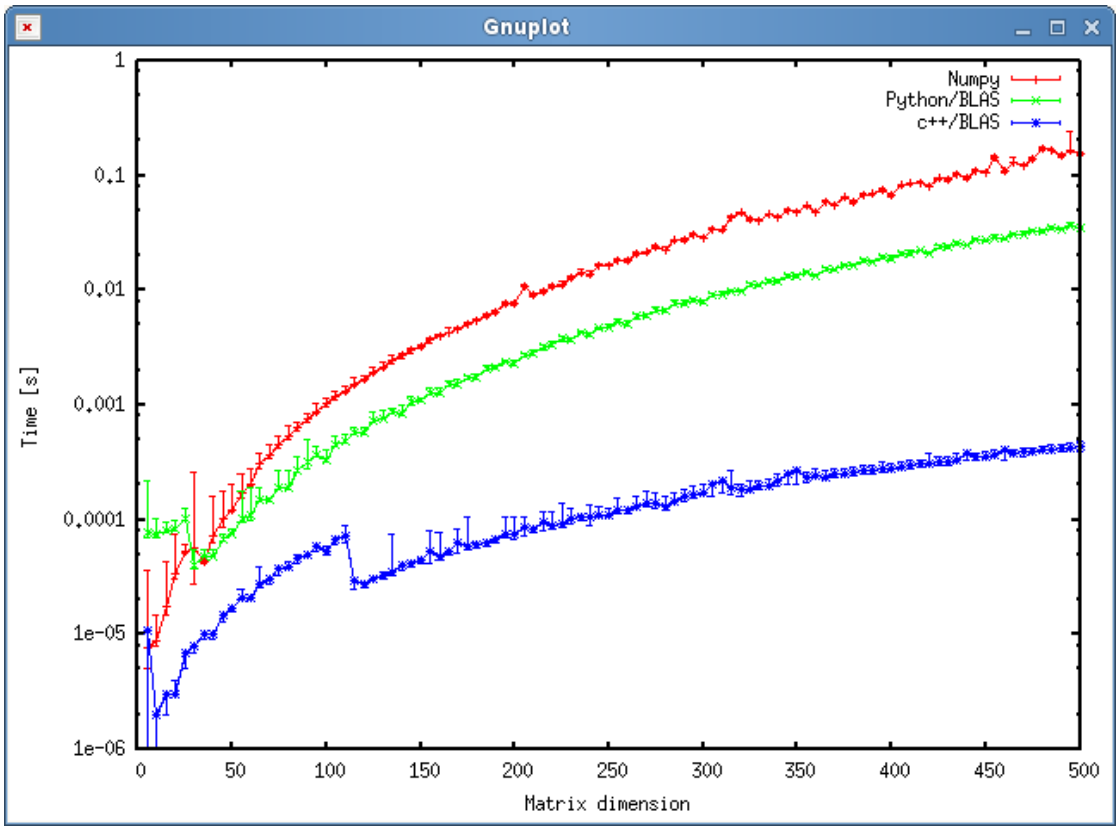

Ilustración 9: Comparativa de rendimiento entre Python, Python con BLAS y C++ con BLAS

No obstante, ciertas partes se han realizado con Matlab, dado el uso intensivo de álgebra lineal y su facilidad para probar partes del código e ir comprobando los resultados, además de su acceso a una mayor variedad de funciones matemáticas que en C++.

4.3 Tecnologías utilizadas

4.3.1 Visual Studio 2017

Visual Studio es un entorno de desarrollo propio de Microsoft que aporta muchas opciones para el ingeniero a la hora de elegir cómo trabajar, ya sea gracias a soportar diversos lenguajes o a los diversos plugins que hay para el programa.

Otro punto que potencia su elección es su popularidad y la cantidad de recursos existentes en Internet, lo que facilita enormemente que exista documentación y ayuda acerca de la mayoría de problemas surgidos a lo largo del proyecto.

Estas opciones junto con la familiaridad previa con el entorno han provocado que se haya utilizado Visual Studio 2017 comunity edition como entorno de programación para C++.

4.3.2 Nuget

Nuget es un plugin para C++ que aporta un repositorio de bibliotecas online, las cuales puedes descargar y el propio Nuget gestiona y añade al proyecto.

Esta manera de gestionar las dependencias externas del proyecto ahorra mucho tiempo y dificultades a la hora de gestionar las dependencias del proyecto y sus respectivas carpetas de descarga, ya que su localización, importación y enlazado es automatizado por parte del programa, y no manualmente.

Además, esto facilita la portabilidad del proyecto entre ordenadores, de manera que solamente necesitas mantener las mismas dependencias.

Ejemplos de software con fines similares serian Maven en el caso de java, o NPM en el caso de node y JavaScript.

4.3.3 Matlab R2016a

Matlab R2016a es una herramienta de software matemático que aporta un entorno de desarrollo integrado para el lenguaje Matlab, así como multitud de herramientas que se basan en este entorno, útiles sobre todo en ambientes docentes y de investigación.

Ejemplos de esto es Symulink, o toolbox de temáticas tan variadas como visión por computador, robótica o procesamiento de señales electrofisiológicas.

4.3.4 HxD

HxD es un visor de archivos en código hexadecimal, de manera que se vean cada uno de los bits del archivo.

Esta funcionalidad es muy útil, ya que permite comprobar si los bytes que se han cargado por parte del sistema se corresponden con los que existen en el archivo, o también comprobar la corrección del archivo acorde al formato de los datos descritos para los archivos *.GDF o *.DAT.

NM24Post_WE_I_2017.03.22_11.33.53.gdf

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   47 44 46 20 31 2E 32 35 58 20 20 20 20 20 20 20  GDF 1.25X
00000010   20 20 20 20 20 20 20 20 20 58 20 20 20 20 20 20            X
00000020   20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000030   20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000040   20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000050   20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000060   20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000070   20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000080   20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000090   20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
000000A0   20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
000000B0   20 20 20 20 20 20 20 20 00 41 00 00 00 00 00 00          .A......
000000C0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000D0   00 00 00 00 00 00 00 00 20 20 20 20 20 20 20 20  ........
000000E0   20 20 20 20 20 20 20 20 20 20 20 20 E0 3B 02 00              à;..
000000F0   00 00 00 00 01 00 00 00 00 02 00 00 40 00 00 00  ............@...
00000100   43 68 61 6E 6E 65 6C 20 31 20 20 20 20 20 20 20  Channel 1
00000110   43 68 61 6E 6E 65 6C 20 32 20 20 20 20 20 20 20  Channel 2
00000120   43 68 61 6E 6E 65 6C 20 33 20 20 20 20 20 20 20  Channel 3
00000130   43 68 61 6E 6E 65 6C 20 34 20 20 20 20 20 20 20  Channel 4
00000140   43 68 61 6E 6E 65 6C 20 35 20 20 20 20 20 20 20  Channel 5
00000150   43 68 61 6E 6E 65 6C 20 36 20 20 20 20 20 20 20  Channel 6
00000160   43 68 61 6E 6E 65 6C 20 37 20 20 20 20 20 20 20  Channel 7
00000170   43 68 61 6E 6E 65 6C 20 38 20 20 20 20 20 20 20  Channel 8
00000180   43 68 61 6E 6E 65 6C 20 39 20 20 20 20 20 20 20  Channel 9
00000190   43 68 61 6E 6E 65 6C 20 31 30 20 20 20 20 20 20  Channel 10
000001A0   43 68 61 6E 6E 65 6C 20 31 31 20 20 20 20 20 20  Channel 11
```

Ilustración 10: Visor hexadecimal y datos interpretados

4.3.5 Bci2000viewer

Bci2000 es un programa que permite cargar los archivos generados mediante interfaces BCI, de manera que muestra los resultados grabados, aportando una función por cada canal grabado, y marcando los momentos en los que ocurren cada uno de los eventos.

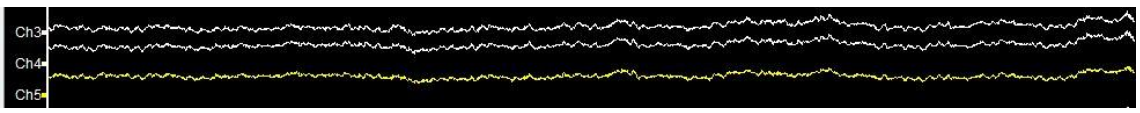

Ilustración 11: Visualización de diferentes canales en Bci2000viewer

El resultado es similar al que se obtendría visualizando los datos en EEGLAB, pero como su funcionalidad se reduce a la visualización, es más simple y más rápido para consultas puntuales.

4.4 Bibliotecas y componentes

Dentro de los programas y lenguajes previos, se ha utilizado los siguientes componentes

4.4.1 Matlab

- Eeglab [9]

Eeglab es una toolbox de Matlab para trabajar con señales electrofisiológicas.

Esta biblioteca aporta métodos para cargar en Matlab archivos de señales, y ciertos métodos para trabajar con ellas, como son su visualización y la aplicación de filtros en frecuencia, así como ciertos plugins que permiten ampliar su funcionalidad.

El uso de eeglab ha sido necesario, ya que aporta una referencia visual con la que comparar la corrección de ciertas técnicas implementadas, como la carga de los datos en C++.

También ha sido muy útil a la hora de estudiar el funcionamiento de algunas técnicas, ya que aporta documentación o implementación de varias de las que se han añadido al proyecto.

4.4.2 C++

- Python36 engine. [10]

Motor de Python 3.6 para C++ proporcionado por Nuget, que permite procesar archivos *.py. De esta manera se pueden integrar en el mismo proyecto ambos lenguajes, permitiendo así pasar variables de C++ al entorno Python, y aplicar las bibliotecas y funciones de este último sobre los datos.

- Matlab engine. [11]

Motor de Matlab para C++ incluido en el entorno de programación de Matlab R2016a. Esto permite interpretar scripts de Matlab dentro de C++ y compartir datos entre ambos lenguajes, lo que aporta la funcionalidad de su core al entorno de C++.

- Armadillo [12]

Armadillo es una biblioteca de álgebra lineal para C++, diseñada para minimizar el cambio entre Matlab y C++, de manera que se acelere la conversión de código de investigación a código de producción.

Esta implementación aporta clases eficientes para vectores, matrices y cubos, así como funciones asociadas a ellos.

Esta eficiencia y la aceleración en trabajos de descomposición de matrices son gracias a su integración con bibliotecas como Blas o Lapack.

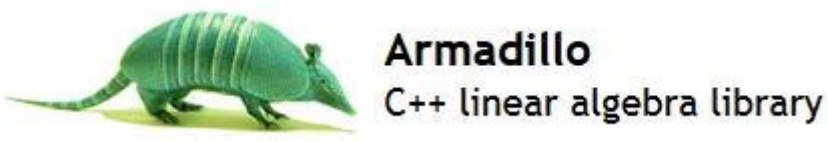

Ilustración 12: Logo de armadillo

- SigPack [13]

Biblioteca básica de procesamiento digital de la señal creada a partir de la biblioteca armadillo. Soporta elementos como las transformadas de Fourier, y los filtros finitos.

- Mlpack (descartada) [14]

Biblioteca con implementaciones de algoritmos de machine learning escalable con posibilidad de añadirlos a proyectos de grandes dimensiones. Toda la biblioteca está basada en las clases de armadillo.

Se planteó utilizarla para tener el cálculo de los componentes de ICA, pero hubo que descartarla debido a su bajo rendimiento con la alta dimensionalidad de los datos.

- BLAS

Biblioteca necesaria para armadillo que le permite acelerar los cálculos de operaciones de álgebra lineal debido a su alta optimización.

Existen diferentes versiones e implementaciones como cuBLAS de Nvidia e Intel MLK. Se optó por openBLAS, debido a su facilidad de integración.

- QT

QT es una biblioteca que permite crear una GUI de manera fácil y rápida, ayudando a gestionar los eventos que se producen en ella, como pulsaciones de botón.

También aporta una amplia variedad de opciones a la hora de crear y representar gráficas para mostrar resultados.

Además de QT, se estudió el uso de Wxwidget, pero esta fue descartada por problemas con gnuPlot, debido a los cuales no permitía trabajar con gráficas.

4.4.3 Python

- Scikit

Biblioteca de Python especialmente dedicada al uso de funciones científicas, con una amplia variedad de funciones implementadas.

4.5 Proyectos similares

Como proyecto similar a este existe la toolbox para Matlab eeglab. [9]

Esta toolbox es bastante útil a la hora de cargar y visualizar datos de los archivos en los formatos comunes cuando se miden componentes neurológicas.

No obstante, la funcionalidad base es reducida y necesita que se añadan diversos plugins para los métodos de filtrado y procesamiento más pesados y potentes.

Estas razones, junto con la necesidad de una licencia de Matlab en activo y de conocer el lenguaje de programación para ciertas tareas, dificultan enormemente su uso por parte de personal sin conocimientos en ambas áreas.

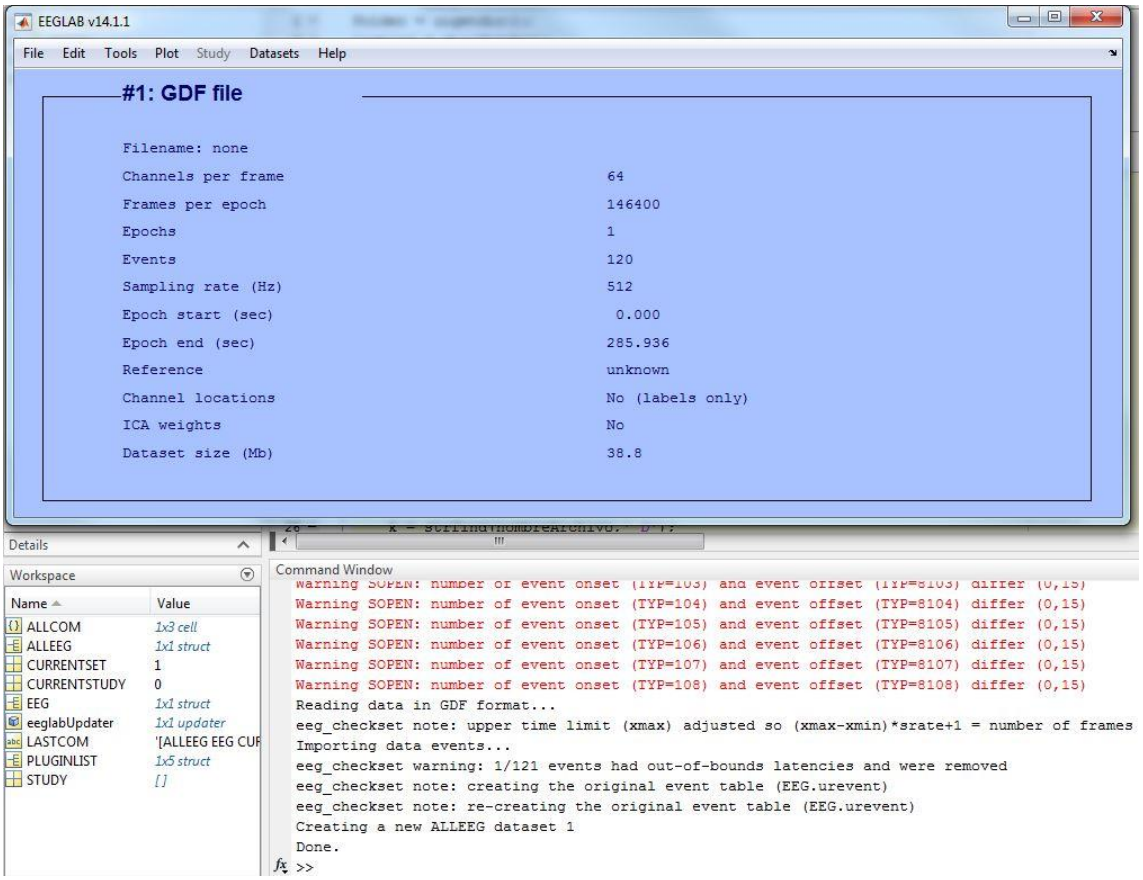

Ilustración 13: Interfaz EEGLAB y entorno Matlab R2016a

5 Estructuras de datos y formatos de los archivos

5.1 Elección del formato de archivos de EEG

La base del proyecto es realizar una herramienta que pueda procesar los archivos generados por el sistema de adquisición de señales neurológicas del que se dispone. Este sistema de medida actual es capaz de generar archivos binarios en dos formatos, los cuales son DAT y GDF, dando en este caso la flexibilidad de poder elegir uno u otro.

De esta manera, se procedió a estudiar el formato DAT. Este es el formato más genérico de los dos, ya que su uso no queda reducido solamente a estímulos neuronales.

No obstante, aunque para este formato se encontró información acerca de cómo se estructuran la cantidad de canales que hay o información concreta de los canales, se descartó debido a la ausencia de datos acerca de cómo se estructuran las señales grabadas.

No obstante, se ha planteado una interfaz para que se pueda ampliar el sistema y dar soporte a este formato de archivos en un futuro.

En cuanto al formato GDF, una primera búsqueda dio resultados acerca de la organización de este formato. No obstante, al comenzar a trabajar se ha visto que no coincidía con el archivo. [15]

Comprobando información acerca del sistema y de la documentación encontrada, se vio que el sistema de adquisición de la señal aporta archivos en formato GDF 1,25, mientras que la versión estudiada era la 2.

Al no tener un archivo en este formato con el que validar el trabajo realizado y estando este en una fase muy temprana, se planteó también como interfaz a implementar en un futuro. Se ha tomado esta decisión debido a que existe la posibilidad de que la actualización del sistema de medida conlleve una actualización de la versión de archivo GDF que genera.

Por último, se exploró la documentación acerca del formato GDF 1, siendo esta la versión implementada y funcional.

5.2 Archivos CED

El archivo CED es un archivo que contiene información acerca de los diferentes electrodos del sistema de adquisición de señales.

Si se recuerda la información contenida en el formato GDF, este tiene un texto asociado al canal, que se podría interpretar como nombre, pero en los archivos procesados por el sistema de medida actual, la única información existente en este apartado es *"Channel N"*, siendo N el número del canal.

De esta manera, el archivo CED da más información acerca del canal y de las propiedades del electrodo.

En cuanto al formato del archivo, es similar a un CSV, pero separando cada campo por un tabulador en lugar de por una coma. De esta manera, el formato CED es fácilmente comprensible por parte de una persona al abrirlo con un procesador de texto plano.

Un ejemplo del archivo es el siguiente:

Number	labels	type	theta	radius	X	Y	Z	sph_theta	sph_phi	sph_radius
1	Fp1		-17.9	0.515	80.8	26.1	-4	17.9	-2.7	85

Sería conveniente también, fijarse en que no hay dato en el campo type. Esto se ve en el archivo en los puntos en los que hay dos tabuladores seguidos. Pudiendo llevar a error el estudio del archivo el no conocer esto.

5.3 Estructura de datos desarrollada [Req02]

Lo primero a comprender de la estructura, es que se ha desarrollado de manera que se pueda adaptar a los posibles formatos, ya que existe la posibilidad de que se quiera ampliar la herramienta en un futuro. En este caso, estos son GDF y DAT.

Por otro lado, dado que existe la posibilidad de que añadan nuevas técnicas a la herramienta, se ha intentado mantener la mayoría de los datos posibles en el sistema. No obstante, hay múltiples atributos que no se han utilizado.

EEGdata es el objeto que contendría todos los datos cargados desde el archivo binario, ya sea directamente, como pasa con el nombre del paciente, o mediante otro objeto, como en el caso de la información de los canales de medida.

Las variables más relevantes de este objeto son:

- Epoch y numEpoch

Almacena la matriz de los diferentes experimentos que mantiene el archivo. Se mantiene como matriz, dado que en el formato DAT puede haber varios experimentos en el archivo. No obstante, es algo que no existe en el formato GDF, de manera que todo lo que se ha trabajado con el programa, no se ha dado el caso de que haya dos elementos en dicho array. No obstante, es una posibilidad.

Dentro de cada elemento del array hay un puntero a una matriz con los datos medidos. Esta matriz será de tamaño numCanales*numMuestras.

- Canales y numCanales

Número de canales y lista que contiene los punteros a los objetos Canal.

- tabla

Puntero al objeto TablaEventos, en el cual se mantiene la información de todos los eventos del objeto.

- frecuenciaMuestreo

Frecuencia a la que el sistema de adquisición de señales ha grabado las muestras.

- Xmin y Xmax

Tiempo en segundos de la grabación. Xmin sería el instante de comienzo y Xmax el de fin. En el caso de GDF, estos campos no existen, así que son calculados en base a la frecuencia de muestreo y a la cantidad de muestras existentes.

Aparte de estos, tenemos otros campos menos relevantes:

- procesamientos

En este campo se van mencionando las diferentes técnicas que se han aplicado al objeto. Su única finalidad es aportar información en la interfaz acerca del objeto.

- idPaciente, idGrabacion, fechaGrabacion, idEquipo, idLab, idTecnicoLab o numSerie

Si nos fijamos en el resto de atributos de este objeto, se ven variables como idPaciente, idGrabacion, fechaGrabacion, idEquipo, idLab o idTecnicoLab. Estos atributos se han planteado como atributos informativos con intención de utilizarlos de manera identificativa en la interfaz de la aplicación, o por si son necesarios en un futuro. No obstante, a día de hoy son campos sin utilidad, dado que son campos que el

sistema de adquisición de señales deja en blanco. Esto es comprensible, sobre todo en el caso de la identificación del paciente, ya que la revelación de la identidad de un paciente supone una violación de la ley orgánica de protección de datos.

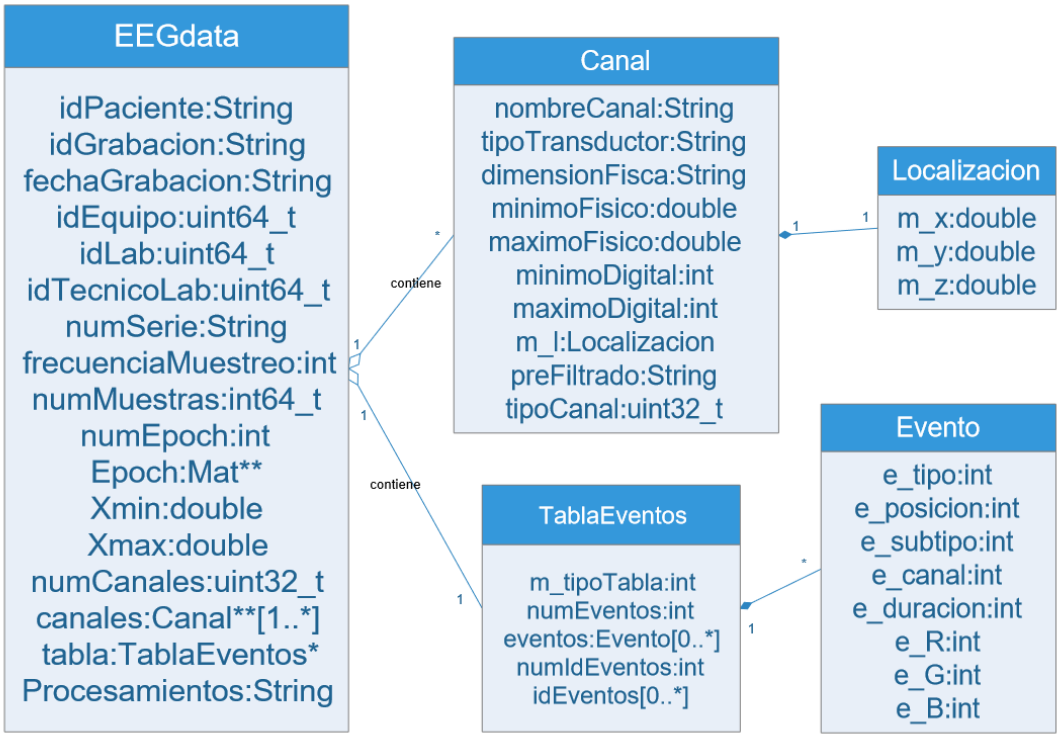

Ilustración 14: Diagrama de clases de los objetos y atributos del modelo de datos desarrollado

Para el objeto Canal, tenemos los siguientes datos:

- nombreCanal

Nombre del canal según el estándar 10-20 internacional.

- m_l

Posición del electrodo del canal. En el objeto Localizacion asociado se almacenan las coordenadas x, y, z.

- TipoCanal

En esta variable se va a guardar el formato en el que se guardan los datos del canal. Más información acerca de esto se puede ver en el anexo de GDF. En la primera tabla del anexo hay un campo code, que indica los diferentes valores del campo y su significado. No obstante, independientemente del valor de este campo, al cargar los datos en la matriz de datos son pasados por un proceso de conversión a formato double.

- PreFiltrado, maximoDigital, minimoDigital, maximoFisico, minimoFisico, dimensionFisica y tipoTransductor

Estos son datos acerca del canal cargados desde el formato GDF. Es información acerca de las propiedades físicas del canal o información acerca de procesamientos previos que aplique el sistema de adquisición de señales a los datos. Este grupo de datos no es utilizado en la aplicación.

Para el objeto TablaEventos:

- m_tipoTabla

Esta variable marca el tipo de tabla que es. Si se sigue el formato GDF se ve que hay dos formas de organizar los eventos. La primera es dándole a cada evento la información necesaria para trabajar con ellos (tabla tipo 1).

La segunda es dándole a los eventos también información de su duración y del canal en el que se ha aplicado (tipo 3).

Dado que existe este formato, se ha contemplado su implementación. No obstante, para el caso de trabajar con ERD y ERS, se graba con el primer tipo de tabla.

- eventos, y numEventos

En estos campos se mantienen la cantidad y la lista de eventos del experimento.

- idEventos y numIdEventos

En estos campos se mantienen la cantidad y la lista de los diferentes tipos de evento existentes en el experimento.

En cada una de las listas de TablaEventos se almacenarían eventos, los cuales tienen el siguiente formato:

- e_tipo y e_subtipo

Aquí se almacenan los id del evento. Se ha planteado con dos campos, ya que el formato DAT puede soportar el trabajar con tipo y subtipo. No obstante, el formato GDF solamente contempla el uso del campo tipo. Este tipo seria el id que relacione a todos los eventos iguales a lo largo de toda la duración del experimento.

- e_posicion

Posición en la que se ha producido el evento. Esto se marca mediante el número de la muestra en la que se ha medido el evento. Con esto y la frecuencia de grabación se puede saber el tiempo exacto en el que se ha producido el evento.

- e_duracion y e_canal

Campos asociados a la duración del evento y el canal en el que ocurre. Suelen estar vacíos.

- e_R, e_G y e_B

Color con el que dibujar el evento en una representación gráfica. Todos los eventos con el mismo tipo mantienen el mismo color, de manera que todos los eventos iguales a lo largo del tiempo tengan la misma manera de representarlos.

5.3.1 Variaciones respecto a la planificación

Para el cumplimiento con el requisito de la estructura de datos, se planteó un sistema en C++, junto con tipos abstractos de datos.

Esto se considera cumplido, ya que se han considerado las matrices de armadillo como una clase con posibilidades para trabajar con datos abstractos.

Respecto al tiempo, dado las dificultades a la hora de afrontar los formatos de archivos, no se ha cumplido la estimación, pasando de las dos semanas esperadas a las cuatro semanas resultantes.

5.4 Carga de los datos

Como se ha explicado al principio del capítulo, el formato seleccionado para su implementación y uso es el GDF versión 1.

De manera, que se ha realizado una sistema que realice una carga de los datos alojados en un archivo binario con el formato mencionado, a la estructura de datos realizada para mantenerlos en la aplicación.

En esta carga binaria se cargan todos los datos del documento, independientemente de que se utilicen o no. Esto se ha decidido así debido a que la modificación de la carga desde el formato GDF puede ser complicada, de manera que se ha preferido añadir al sistema toda la información posible, por si es necesaria en ampliaciones futuras.

El formato GDF consta de 4 partes diferenciadas.

- Cabecera estática

Esta parte almacena los datos generales del documento. Los más relevantes de esta parte es la identificación del paciente, datos acerca de la fecha de grabación, duración del experimento, y el número de señales que se han grabado o canales que tiene el sistema.

Este último campo es el campo NS, el cual se utiliza para calcular el tamaño de la cabecera variable.

- Cabecera variable

En la cabecera variable están los datos de cada uno de los canales de medida, de manera que cada campo en la cabecera variable se repite NS veces. Ejemplo de los campos de esta parte es una etiqueta para el canal, las dimensiones físicas del electrodo, y si se ha aplicado algún tipo de filtro analógico por parte del sistema de adquisición.

También mantiene un campo NR para cada uno de los canales, el cual indica el número de medidas realizadas por el canal. Dado que todos los valores de los campos NR son el mismo, este campo no se contempla como dato propio del electrodo, sino del EEG en general, de manera que no se convierta en un dato duplicado.

Por último, el campo Type contiene el formato en el que el sistema ha guardado los datos. Este puede ser en entero, entero sin signo, decimal o punto flotante, cada uno también con diferentes longitudes como puede ser 8, 16, 32 o 64 bits.

- Datos grabados

En esta parte se almacenan las diferentes medidas que el sistema ha grabado. Siendo NS*NR medidas.

Los datos se organizan de manera que se vayan avanzando los canales del 1 al NS, para la medida 1, repitiéndose una medida para cada uno de los canales hasta llegar a la medida NR.

- Tabla de eventos

Por último, está el apartado de eventos.

Si se revisa la documentación de GDF se ve que existe el campo modo de tabla de eventos, teniendo este campo dos posibles valores. Esto implica que existen dos formatos de la tabla. Se han implementado ambos. No obstante, la versión más simple de las dos cumple adecuadamente con el cometido, siendo esta la explicada y utilizada por el sistema de grabación.

En esta versión más básica, la tabla de eventos almacena el campo N, que indica la cantidad de eventos en el sistema.

Para cada uno de los eventos, se cargan el tipo de evento que es, codificado en formato numérico, y la medida concreta en la que se marcó ese evento.

Aquí se han comentado las partes más relevantes del archivo, no obstante, en este documento existe un anexo en el cual se detalla ampliamente la estructura del archivo binario. Por otro lado, también se puede consultar información adicional acerca de este formato en la bibliografía.

Una vez completada la carga de los datos EEG, se procede a cargar la información acerca del sistema de medida, alojada en un archivo CED.

De este archivo, actualmente solo se almacena en el sistema el nombre del canal, dado que en este archivo sí que se sigue el estándar de medida 10-20, así como la localización X Y Z del electrodo. Los datos restantes no se añaden al sistema, y de los que se cargan, actualmente solo se trabaja con el nombre de los canales según el estándar 10-20.

6 Algoritmos y funcionalidades propuestas

Las funcionalidades propuestas y los algoritmos que se quieren utilizar en el programa son los siguientes:

6.1 ASR [Req03]

6.1.1 Técnica

El algoritmo ASR (Artifact Subspace Reconstruction) es un algoritmo para eliminar desviaciones y picos de descargas por métodos estadísticos. [16] [17]

El sistema consta de tres partes principales, las cuales son filtrado, entrenamiento y aplicación.

Para la parte de filtrado, se utiliza un filtro finito hacia delante y hacia atrás, para eliminar desviaciones en los datos. La aplicación del filtro en este punto, se centra en aplicar el filtro con frecuencias muy bajas, de manera que se eliminen posibles perturbaciones provenientes de la electrónica de los aparatos de medida.

Para el entrenamiento, esta técnica suele contar con un minuto de los datos del paciente en reposo, de manera que se tenga una muestra limpia del comportamiento neurológico del paciente. En el caso de no tener esta muestra, como es el caso de este proyecto, es necesario buscar las partes más limpias de la grabación para poder entrenar el algoritmo.

Para la aplicación del ASR, partiendo de datos entrenados, se busca ventanas con una desviación típica con un valor varias veces superior a la desviación que muestran los datos de entrenamiento [18]. Los tramos encontrados, se reconstruyen basándose en los datos obtenidos en el entrenamiento.

6.1.2 Funcionalidad implementada

6.1.2.1 Filtrado

Para la aplicación, se ha utilizado el acceso a los métodos de Matlab, de manera que se accede a los métodos de diseño de ventanas. En este caso, se utilizan ventanas de káiser.

En cuanto al diseño del filtro, recae también en Matlab, partiendo de las ventanas y las frecuencias previas, así como la frecuencia de muestreo de los datos.

Una vez limpiadas estas perturbaciones, se pasa al cálculo del ASR en sí.

6.1.2.2 Limpieza

Este algoritmo, funciona muy bien a la hora de reducir desviaciones estadísticas, incluso en tiempo real. No obstante, para poder hacer eso, el algoritmo se basa en cierto tiempo de grabación del paciente con datos en reposo. En este caso, eso no es posible, ya que por una parte, sería muy molesto para el paciente alargar el experimento solamente para estar parado sin hacer nada, y por otra, el objetivo de la investigación, es aplicarlo a pacientes con enfermedades motoras, de manera que no es posible grabar a esos pacientes durante un minuto sin que sufran espasmos o perturbaciones. Debido a esto, es necesario buscar los datos de calibración del ASR en los datos que se obtienen en el experimento, y no en una grabación alternativa con el paciente en reposo.

Para poder hacer esto, se aplican varios procesos a los datos.

Primero, se separan cada uno de los canales. Después, se calcula el tamaño de ventana, teniendo estas un solapamiento de 2/3, y una cantidad de ventanas igual a la frecuencia de muestreo.

Para cada una de las ventanas, se calcula el valor RMS. Una vez obtenidos los datos de todas las ventanas del canal, se procede a calcular los valores mu (estimación media de los datos sin ruido) y sigma (estimación de la desviación típica para los datos sin ruido) para el canal, en base a la distribución que tienen los datos.

Con estos valores se saca la estimación de si esa ventana se debe eliminar o no, mediante la fórmula ((RMS-mu)/sigma).

Una vez obtenidos los valores para todas las ventanas y los canales, se procede a ordenar los valores obtenidos para cada una de las ventanas, de manera que se tenga para cada ventana de tiempo los valores que ha obtenido en cada uno de los canales ordenados de menor a mayor.

Aplicando la tolerancia de canales malos, establecida en 5 para ASR, se omiten las 5 primeras estimaciones (mínimos valores estimados para esa ventana) y las 5 últimas (máximos valores estimados para esa ventana). Con los valores restantes, se estudia que valores están por debajo o por encima de los umbrales de ASR, situados entre -3.5 y 5.

En caso de que alguna de las estimaciones restantes para esa ventana este por encima o por debajo del margen, esa ventana se considera ruidosa y elimina para todos los canales.

6.1.2.3 *Entrenamiento*

Partiendo de los datos limpiados en el tramo anterior, se procede a entrenar el algoritmo ASR.

Lo primero que aplica en esta fase es un filtro Yulewalk, de manera que se amplifique la señal y el ruido en las frecuencias superiores y se suavice en las frecuencias inferiores.

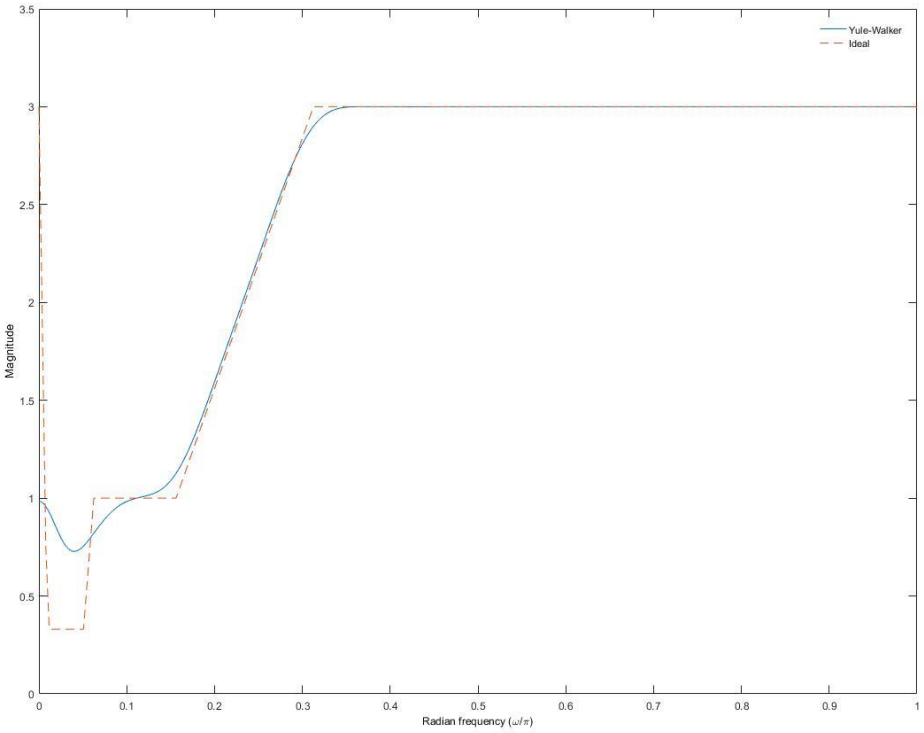

Ilustración 15: Diseño del filtro Yulewalk aplicado.

Con los datos obtenidos tras aplicar el filtro, y un tamaño de bloque de 10 elementos, se procede a calcular la matriz de covarianza de las muestras, promediada en bloques con muestras sucesivas del tamaño de bloque establecido. Esta es la matriz U.

Obtenida la matriz U se procede a calcular su media geométrica mediante el algoritmo de Weiszfeld [19].

Haciendo un reshape del resultado para obtener una matriz cuadrada y aplicando una raíz cuadrada sobre el resultado, se obtiene la matriz M o matriz de mezcla.

Para obtener la matriz restante, se aplica una descomposición EIG sobre la matriz M, obteniendo las matrices V y D.

Utilizando los datos iniciales multiplicados por la matriz V devuelta por la descomposición, y sacando el valor absoluto del resultado, tendríamos la matriz X.

Con esta matriz se va a aplicar un proceso similar al empleado en la limpieza. Partiendo de ventanas con un tamaño de la mitad de la frecuencia, y solapamiento de 2/3, se procede a calcular el RMS de cada una de las ventanas, y con estos valores RMS, los valores mu y sigma del canal. De esta manera, aplicando para cada uno de los canales, se obtienen los datos para toda la matriz.

Obtenidos todos estos datos, es posible calcular la matriz T con la siguiente formula, dando así por concluido el entrenamiento del sistema.

$$T = diag(mu + cutoff * sig) * V'$$

Ilustración 16: Fórmula para calcular la matriz T

6.1.2.4 Aplicación

En este punto, y partiendo de los datos obtenidos después de aplicar el filtro descrito en el punto 6.1.2.1, junto con las matrices de calibración M y T obtenidas en el apartado anterior, se procede a aplicar el algoritmo en pequeños.

El tamaño del tramo está establecido en un cuarto de segundo, de manera que el número de muestras varía dependiendo de la frecuencia de la grabación. Para el caso de una grabación a una frecuencia de 512 Hz, la aplicación sea mediante ventanas de 128 muestras.

El primer paso del algoritmo, es añadir una extrapolación de la señal del tamaño del tramo, de manera que se añada un tramo más tanto al comenzar como al finalizar la grabación.

Esta extrapolación se realiza tomando para cada canal la primera o ultima muestra de la grabación, según sea el caso, duplicando su valor, y restando las siguientes n, anteriores o posteriores, siendo n la cantidad de muestras que se quieren añadir.

Posteriormente, se aplica el mismo filtro Yulewalk que se ha aplicado en las fases previas del algoritmo, de manera que se amplifiquen las frecuencias superiores.

Después se procede a calcular la covarianza por medias móviles. Para ello, se procede a calcular la multiplicación de cada uno de los canales por sí mismo traspuesto a lo largo de una tercera dimensión, generando así un cubo de C*C*N, siendo N el número de muestras en la matriz.

Sobre estos datos, se aplicaría un reshape, pasando el cubo a una matriz de tamaño C^2 * el número de muestras. Sobre este resultado, se aplicaría un filtro de media móvil a lo largo de su segunda dimensión.

Por último, se mantendrían solamente las columnas que son múltiplos del tamaño de tramo, y se aplicaría un nuevo reshape, dando a los elementos restantes forma de cubo, en el cual cada slice es de lado C, y prolongando el cubo todo lo necesario a lo largo de la tercera dimensión.

Para cada slice de la matriz de covarianzas, se aplica PCA para buscar las regiones en las que hay desviación mediante EIG. Se ordena D según los valores de la diagonal, y se ordenan las componentes en V siguiendo el mismo orden generado.

Se calculan los elementos a eliminar mediante la fórmula siguiente. C es la cantidad de canales existentes en la grabación, D es la diagonal producida al aplicar la descomposición EIG, V las componentes calculadas mediante EIG, y T la matriz calculada en el entrenamiento del algoritmo, dando así el vector keep que se utiliza para la reconstrucción de la señal.

$$keep = D < \sum \left((T * V)^2 \right) \vee 1 : C < C/3$$

Ilustración 17: Obtención de los índices de las componentes a eliminar

Una vez analizadas las partes a eliminar, se procede a calcular la matriz R. En caso de que no haya ningún elemento que borrar, esta será una matriz identidad de tamaño C.

En caso de tener elementos para borrar, R se calcularía de la siguiente manera:

- V'*M.
- En los índices en los que el vector keep es cero, esa fila de la matriz resultante se pone completamente a ceros.
- Se aplica la pseudoinversa del resultado
- Y por último, R seria el resultado de multiplicar M * la pseudoinversa calculada * V'

Siendo V la matriz de eigenvectores, y M la matriz de mezcla calculada previamente en el entrenamiento.

En caso de que en esta iteración o en la anterior, se marcase algún elemento a eliminar, se procede a reconstruir el tramo defectuoso de la señal utilizando la R calculada y la técnica de mezcla de coseno alzado.

Siendo n la muestra en la que acaba el tramo de estudio actual, y n-1 el valor de n en la iteración previa, de manera que indica la muestra en la que comienza el tramo.

$$blend = \frac{1 - \cos\left(\frac{\Pi * (1:(n - n_{-1}))}{n - n_{-1}}\right)}{2}$$

Ilustración 18: Cálculo del filtro de coseno alzado

Típicamente, la diferencia entre n y n-1 va a ser el tamaño de tramo. No obstante, el primer índice para n no es 0, sino 1, y la segunda el valor del tramo en sí, de manera que existe la posibilidad de que exista algún tramo con un valor ligeramente inferior, especialmente al comienzo o al final de los datos.

De esta manera, la señal se reconstruiría multiplicando la matriz R por los datos que se quieren reconstruir, y aplicando a cada canal el filtro de coseno alzado aplicado. A todo esto, habría que sumar el resultado de la reconstrucción de los datos aplicando R-1 y la inversa del filtro de coseno alzado.

Por último, se eliminan los datos que se habían extrapolado tanto delante como detrás de los datos originales.

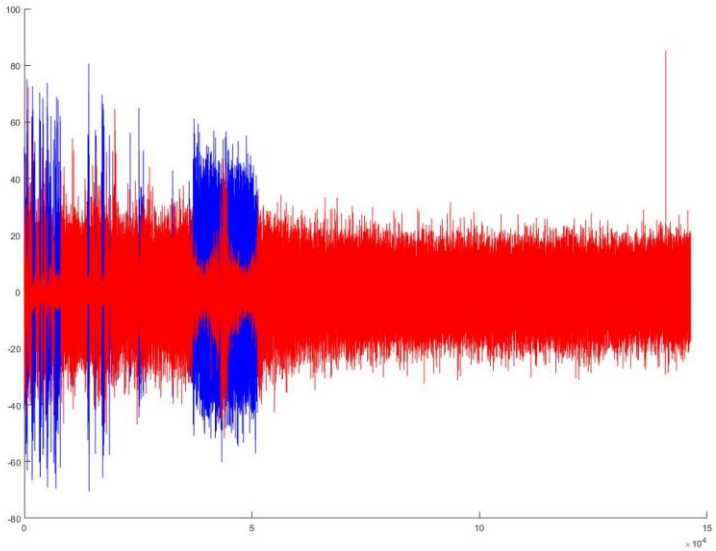

Ilustración 19: Datos originales en azul, y resultados tras aplicar ASR a esos datos en rojo, correspondientes a un canal de medida

6.1.3 Variaciones respecto a la planificación

El algoritmo ASR se planificó como un requisito a implementar en C++, basándose en implementaciones realizadas en Matlab, y estimando una duración de diez semanas.

Dada la escasez de documentación encontrada, el trabajo con este algoritmo se ha centrado en comprender la implementación de Matlab, y realizar una migración del código a C++. No obstante, dada la alta dependencia con filtros y funciones no encontradas en C++, hay parte que no se ha podido migrar, manteniéndolo en Matlab, o incluso cuando este ha fallado, en Python.

Respecto a la planificación, al poder mantener parte del algoritmo en otros lenguajes con mayor acceso a funcionalidades científicas, se pudo mantener su desarrollo en las diez semanas previstas.

6.2 Filtros frecuenciales [Req04]

6.2.1 Técnica

Para el estudio que ocupa este documento, las frecuencias relevantes no son todas las existentes, de manera que es necesario que se puedan diferenciar entre la totalidad de las frecuencias grabadas y las que son necesarias trabajar con ellas.

Esta separación de frecuencias se realiza mediante filtros frecuenciales. [2]

El trabajo de estos filtros es diferenciar y separar los elementos de una señal que funcionan a un determinado rango de frecuencia, de los restantes elementos.

Dentro de estos tipos, hay dos técnicas de aplicación:

- Filtros infinitos

Esta manera de aplicar filtros produce filtros más rápidos y pequeños, pero también filtros más inestables, ya que la aplicación del filtro en cada punto es dependiente del resultado del filtro en todos los elementos anteriores, de manera que por mucho tiempo que pase, incluso el primer estímulo afecta al resultado

- Filtros finitos

Estos filtros son más grandes, y por lo tanto más lentos, pero la aplicación del filtro es directa sobre los datos, no necesita datos previos del filtro, de manera que no depende de los resultados que se han obtenido previamente.

Y 4 tipos de filtro:

- Pasabanda

Este tipo de filtro elimina las frecuencias que no están dentro de las frecuencias marcadas

- Notch o parabanda

Este tipo de filtro elimina las frecuencias que están dentro del rango marcado

- Pasoalto

Este tipo de filtro elimina las frecuencias inferiores al valor del filtro

- Pasobajo

Este tipo de filtro elimina las frecuencias superiores al valor del filtro

6.2.2 Funcionalidad implementada

Para este punto de la aplicación solamente se han introducido los filtros finitos.

A pesar de ser más lentos, estos filtros no tienen riesgo de interferencias de los datos iniciales según el filtro va avanzando, y en este caso, la cantidad de elementos a filtrar suele ser elevada.

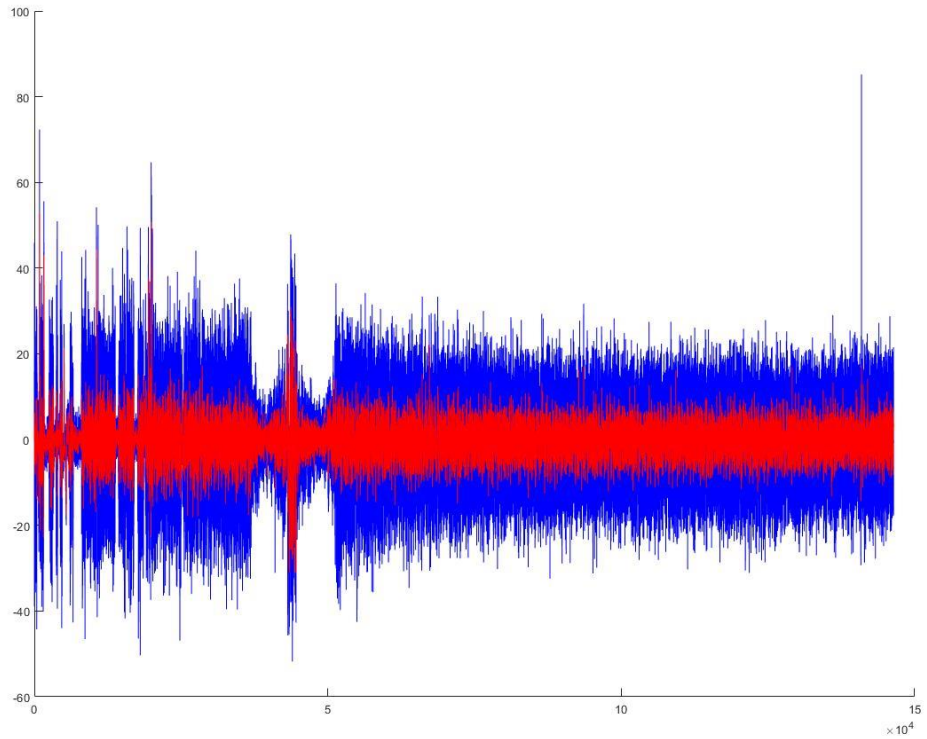

Ilustración 20: Antes (azul) y después (rojo) de aplicar a un canal de datos un filtro pasabanda de 1 a 40 HZ

Como se ha comentado más arriba, una selección correcta del tipo de filtro y de las frecuencias de corte, produciría la eliminación de determinadas frecuencias. La eliminación de esas frecuencias puede quitar las componentes neuronales que no se consideran relevantes, y que en este caso afectan solamente al sistema como ruido, pero también puede producir la reducción de ruido del entorno, como el proveniente de cables eléctricos de corriente alterna, los cuales producen interferencias en la señal en las bandas de los 60HZ.

En cuanto a la distribución del filtro, es necesario seguir el estándar establecido en la literatura, el cual en todas las publicaciones revisadas es Hamming.

6.2.3 Variaciones respecto a la planificación

El cumplimiento de este requisito se planteó como una tarea a implementar mediante C++ y bibliotecas de álgebra lineal y procesamiento digital de la señal, estimando la duración del trabajo en dos semanas.

Esta estimación de las tecnologías se ha cumplido lo planificado, utilizando C++, Armadillo y SigPack respectivamente.

En cuanto al tiempo, no se ha cumplido la estimación. Esto se ha debido a un intento de implementación de las técnicas sin apoyo de la biblioteca de SigPack, de manera que al incluir la biblioteca hubo que replantear el modo de funcionamiento, llevando el tiempo de trabajo a las cuatro semanas.

6.3 Promediado [Req05]

6.3.1 Técnica

La técnica del promediado se basa en seleccionar un ancho de ventana, y unos puntos sobre los que extraer esa ventana, procediendo más tarde a aplicar la media de los tramos extraídos.

Esta técnica permite una aproximación a los datos, comunes entre cada uno de los tramos extraídos, mitigando el ruido propio de cada uno de los tramos.

6.3.2 Funcionalidad implementada

Como se ha explicado en el apartado Rutina de los experimentos, la forma de grabarlos consiste en repetir varias veces un *epoch*, a la vez que se van marcando los eventos que suceden.

De esta manera, en los datos que llegan al método de promediado, se tienen todos los datos del experimento y puntos marcados donde ocurren las mismas acciones.

En este caso, el usuario no va a ser capaz de elegir puntos al azar según considere, sino que la elección se basara en elegir un tipo de evento de los existentes en los datos, de

manera que el promediado se aplique sobre los puntos en los que existe ese tipo de evento.

Lo que sí que va a poder elegir con total libertad es el tiempo que quiere incluir en el promediado, indicando los milisegundos anteriores y posteriores que tendrá la ventana. Aplicando esta ventana de tiempo a todas las apariciones del evento seleccionado, se obtendrían todos los comportamientos que el usuario ha tenido con respecto a ese evento. Calculando la media entre los comportamientos, se obtendría un comportamiento que se aproximaría mucho más al resultado general entre los diferentes *epoch*.

Esto es muy útil, ya que cada uno de los *epoch* de un EEG va acompañado de su ruido propio, como pueden ser artefactos. Ejemplos de estos artefactos son parpadeos, o algún tipo de actividad muscular ajena a la acción solicitada (mover el brazo por un espasmo muscular inconsciente, realizar un movimiento consciente como mover el brazo para rascarse…).

En cambio, al aplicar el promediado, el movimiento realizado por el paciente entre *epoch* es constante, de manera que no se verá afectado en gran medida por el promediado, mientras que los artefactos, que en su mayoría son movimientos o acciones puntuales que ha realizado el usuario, se verán afectados, tendiendo a suavizarse o incluso borrarse según aumente la cantidad de muestras utilizadas en el promediado.

6.3.3 Variaciones respecto a la planificación

El cumplimiento de este requisito se planteó como una tarea a implementar mediante C++ y bibliotecas de álgebra lineal, estimando la duración del trabajo en dos semanas.

Ambas estimaciones se han cumplido.

6.4 ICA

6.4.1 Técnica

ICA [20] [21] es un algoritmo de análisis de componentes individuales, típicamente utilizado en sistemas de procesamiento de la señal, en los cuales se tengan mezcladas varios subcomponentes que sean estadísticamente independientes, de manera que, partiendo de una serie de puntos de grabación, y un tramo de grabación completo, se puedan extraer las señales existentes en la grabación.

El ejemplo típico en la explicación de ICA es el problema de la sala de cocktail y grabaciones con micrófonos [22]. Cada uno de los micrófonos sería un punto de grabación, al cual llegaría sonido de toda la sala, mezclando así la información de diferentes fuentes independientes. La aplicación de ICA sobre estas grabaciones con

los datos mezclados es capaz de aislar las componentes independientes existentes en los datos medidos. En este caso, sería equivalente a extraer el sonido de las diferentes conversaciones grabadas, eliminando así la mezcla de diferentes conversaciones grabada por cada uno de los micrófonos utilizados.

En este caso, la explicación encaja razonablemente bien, ya que cada componente neurológica se propaga en mayor o menor medida a lo largo de bastante parte del cerebro, siendo este similar al sonido en una sala de cocktail, y las activaciones neuronales se graban en diversos puntos, siendo los electrodos los sustitutos de los micrófonos.

6.4.2 Funcionalidad implementada

Comúnmente, la implementación utilizada para realizar el análisis es la técnica fastICA, la cual se basa en técnicas binarias.

Típicamente, las versiones binarias de ICA se centran en conseguir la mejor componente posible y avanzar una a una. Esto suele dar una mayor definición de las componentes iniciales, pero siendo las siguientes dependientes del orden y definición de las componentes encontradas.

Por otro lado, existe runICA, la cual se basa en infomax, o estimación de máxima probabilidad [20] [23] [24].

El uso de infomax implica ciertas características relevantes. En contraposición a las técnicas binarias, la versión infomax se centra en conseguir todas las componentes a la vez, de manera que las últimas componentes no sufren dependencia de las primeras que se han encontrado.

La búsqueda de las componentes en paralelo tiene ciertas ventajas. Dada la alta dimensionalidad de los datos, este método permite realizar el análisis en un tiempo adecuado, frente a la lentitud de fastICA.

Además, dado que el cálculo de las componentes es a la vez, las componentes encontradas no tienen dependencia con respecto a las encontradas previamente, de manera que mantienen una calidad estable.

En cuanto a runICA, la principal característica de este algoritmo es que da las matrices de pesos y esfericidad, quedando en el usuario el cálculo de la matriz de desmezclado.

De manera general, esto puede aparentar ser un problema, pero el cálculo restante sería simplemente la multiplicación de ambas matrices.

Por otro lado, en el entorno actual esto es una ventaja, ya que esta descomposición va conjunta con la aplicación de MARA, y en dicho algoritmo es necesario el uso de las 3 variables.

6.5 MARA [Req06]

6.5.1 Técnica

MARA (Multiple Artifact Rejection Algorithm) es un algoritmo de eliminación de artefactos ocultos en la señal. [25] [26]

Para hacer eso, se basa en extraer las componentes independientes que existen en la señal, que son con las que va a proceder a trabajar, las cuales las consigue a partir del algoritmo ICA.

Una vez obtenidas las componentes, se puede comenzar a trabajar con el algoritmo.

El trabajo de esta técnica cuenta con varios puntos.

Primero, filtra las componentes independientes, según el canal al que están asociadas, de manera que solamente los canales que tienen coincidencia con los que se contemplan en MARA son utilizados en el algoritmo.

Después, consiste calcular una serie de características de las componentes, y en base a estas, mediante un clasificador lineal, decidir si se consideran ruido o señal.

Por último, queda reconstruir los datos con las componentes independientes que son consideradas señal, borrando así las componentes que se han considerado artefactos.

6.5.2 Funcionalidad implementada

6.5.2.1 Carga de datos de MARA

En el apartado de introducción se adjunta la bibliografía de MARA. Aparte de esta bibliografía, se adjunta aquí otra entrada haciendo referencia a la web GitHub, en la cual, además de haber información acerca de la técnica, existen archivos con datos fundamentales para el correcto funcionamiento del algoritmo. [27]

- Nombres de los diferentes canales de medida del sistema de adquisición usado.
- Características de las diferentes componentes calculadas para entrenar el clasificador de su experimento.
- Clasificación de las componentes como artefacto o señal para entrenar el clasificador.
- Matriz de compensación de profundidad Lambda.

En estos archivos esta la información que se recopilo en la investigación realizada para calibrar MARA, y en base a la cual se entrena el sistema.

6.5.2.2 Correlación de canales

Lo primero que hay que tener en cuenta para MARA, es que ICA extrae componentes independientes según el tiempo, pero no mantiene correlaciones en el espacio. Inicialmente, MARA contemplaba esta correlación en el espacio mediante el uso de TDSEP en lugar de ICA. No obstante, en versiones posteriores se ha recuperado el uso de ICA.

Para cubrir esta carencia, MARA aplica una comparación entre los nombres de los diferentes canales grabados en el experimento y los que se grabaron para crear los datos del algoritmo (archivo adjunto al artículo de MARA), de manera que solo se tengan en cuenta los canales que existen en ambos sistemas, eliminando así la evaluación de los canales no conocidos por parte de MARA.

Si volvemos al capítulo Fundamentos de neurociencia, se puede ver la distribución de los canales en el sistema de medida del estándar internacional 10-20, el usado para grabar los experimentos. En el caso de MARA, se usó un sistema más completo con 104 electrodos.

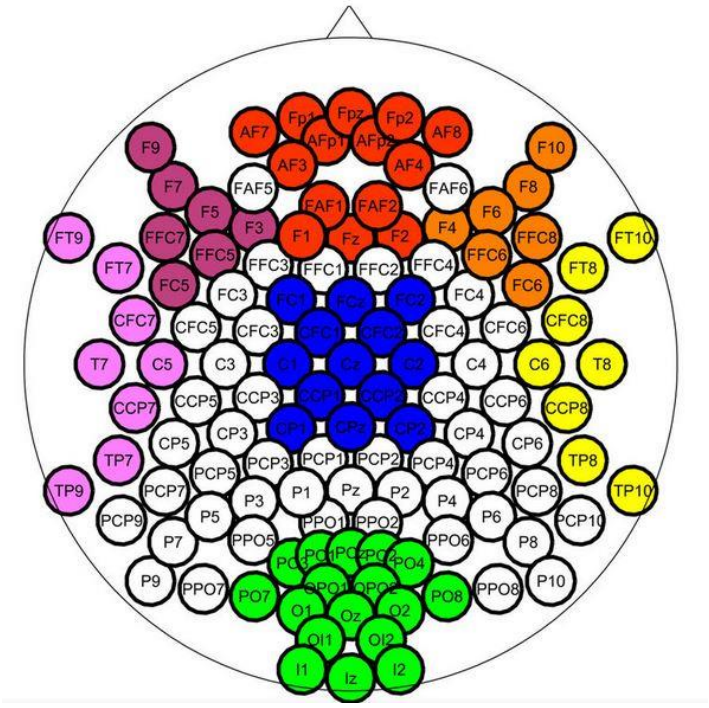

Ilustración 21: Distribución y nombre de los canales utilizados en la investigación del algoritmo MARA

De todos estos se van a mantener los canales que tengan una correlación con los diferentes canales contemplados en MARA. En este caso, entre el sistema de medida de MARA y el del estándar 10-20 hay 55 coincidencias.

6.5.2.3 Extracción de los datos de clasificación

Una vez conseguidos los datos con los que trabajar, se procede a extraer las diferentes características que va a utilizar el sistema para clasificar para cada una de las componentes para las que hay coincidencia.

Si se consulta el artículo de investigación de MARA, [25] se ve que en él se estudian múltiples características, tanto temporales como frecuenciales, no obstante, en las conclusiones del articulo solo se tienen en cuenta 6 para la construcción del clasificador. A continuación, se mencionan las empleadas en el clasificador, adjuntando junto al nombre la columna equivalente en los archivos de entrenamiento del clasificador lineal.

- Current density norm, primera característica

Para esta característica se parte de la matriz de pesos de las componentes independientes calculadas. Esta matriz se filtra, de manera que se mantengan solamente las filas cuyo nombre de canal coincida con los existentes en MARA. Para estandarizar estos datos, cada uno de los valores se divide entre la desviación estándar existente con respecto a los valores de ICA para ese instante. Esta matriz seria la variable patterns.

Además de esta variable, es necesaria la matriz M100.

Para calcularla, lo primero es necesario trabajar con los datos cargados desde los archivos de MARA, concretamente con los datos de la matriz de compensación de profundidad Lambda.

Partiendo de estos datos, lo primero es eliminar los canales que no existen en el sistema de medida estándar 10-20, de manera que se mantengan los mismos canales que los que se van a clasificar. Esta sería la matriz F.

Después, partiendo también de la matriz completa de compensación de profundidad Lambda, se calcula la inversa de la matriz de pesos s-LORETA [28] [29]. Esta sería la matriz W.

La matriz de pesos H es una matriz cuadrada. El tamaño de lado es la cantidad de canales coincidentes entre MARA y el estándar 10-20. Los valores serian la diagonal principal de unos, menos uno entre el tamaño de lado.

Con estas matrices se puede pasar a calcular la matriz de compensación de profundidad Lambda adecuada para los datos del paciente.

$$L = H * F * W$$

Obtenida la matriz Lambda apropiada para los datos del paciente, se puede proceder a calcular la matriz M100 según la próxima formula.

$$M100 = L' * inv1 * H;$$

Siendo inv1 la inversa de la multiplicación de la matriz L por L transpuesta más 100 en la diagonal principal.

$$Inv1 = inv(L*L' + 100 *eye(size(L*L'))$$

Una vez obtenidas las matrices M100 y patterns, se multiplica M100*patterns, y se aplica la normalización de Frobenius a cada una de las columnas. Posteriormente se aplicaría el logaritmo neperiano a los datos para finalizar la normalización.

$$\|X\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

Ilustración 22: Normalización de Frobenius

- Spatial range, segunda característica

Esta característica parte de la existencia de la matriz patterns, calculada para la característica anterior.

Partiendo de esta matriz, la característica para cada una de las componentes va a ser aplicar el logaritmo neperiano sobre la diferencia entre la máxima y la mínima activación existente en cada una de las columnas de la matriz.

- Average local skewness, tercera característica

Para cada componente independiente se calcula el logaritmo del skewness en ventanas de 15 segundos. En caso de haber más de 30 segundos, de manera que se puedan aplicar más de dos tramos, se haría el logaritmo de la media para calcular el valor de la componente.

En caso de no llegar a los 15 segundos, se reduciría poco a poco el tiempo de intervalo mínimo, para obtener el skewness más grande posible.

En cuanto al cálculo del skewness, una vez encontrada la ventana con la que mejor se puede trabajar, se aplicarían las fórmulas de las ilustraciones.

$$Z = x - \sum_{0}^{n} \frac{x_i}{n}$$

Ilustración 23: vec Z = values - mean(values);

$$Skewness = \frac{\sum_{0}^{n} \frac{Z_i^3}{n}}{\left(\sum_{0}^{n} \frac{Z_i^2}{n}\right)^{1.5}}$$

Ilustración 24: double skewness = mean(Z^3)/(mean(Z^2)^1.5);

- Band Power 8-13 Hz, quinta característica

Partiendo de la componente independiente con la que se está calculando, se aplica una estimación de la densidad de potencia espectral o PSD. En este caso, mediante la aproximación de Welch. [30] [31]

Partiendo de la estimación, esta característica se calcularía como la media entre los valores dados para las frecuencias entre 8 y 13 Hz, lo que se correspondería con la banda alfa del cerebro.

- Lambda y fit error, cuarta y sexta característica

Partiendo también de la estimación de la densidad de poder espectral, se sacan 6 puntos.

PSD a 2 Hz, a 3 Hz, el mínimo PSD en las frecuencias entre 5 y 13 Hz, el PSD en la frecuencia anterior al mínimo PSD entre 5 y 13 Hz, el mínimo PSD en la banda de 33 a 39 Hz, y el PSD en la frecuencia siguiente a la mínima de la banda 33 a 39 Hz, los cuales serían las componentes Y. Por otro lado, la componente X seria la frecuencia correspondiente a cada PSD.

Este grupo de puntos se utiliza para el ajuste de los 3 valores (a, b y c) de una función mediante regresión.

Para el cálculo del mejor valor posible, se aplican varios descensos del gradiente con inicialización aleatoria de sus variables, quedándonos con el resultado que menor error aporte.

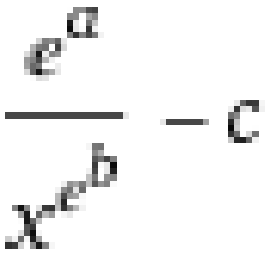

Ilustración 25: Función sobre la que se aplica regresión en MARA

De este resultado, la variable sería lambda, y el error de ajuste de la función sería la variable fit error.

6.5.2.4 Datos de entrenamiento

Después de obtener los datos sobre los que se va a clasificar, se procede a preparar los datos de entrenamiento aportados por MARA.

Estos datos contienen la clasificación de las componentes como artefacto o no, las características de la tercera a la sexta y matriz patterns. Con esto y la matriz M100 calculada previamente con los datos del paciente, se calculan las dos características ausentes que se van a aplicar para clasificar, de manera que se tienen las seis características y la clasificación de cada una de ellas.

Una vez calculadas las diferentes características para cada una de las componentes encontradas, se procede a su clasificación.

6.5.2.5 Clasificación

En esta parte, entraría en juego un clasificador lineal por análisis de discriminantes lineales (LDA) [32] [33], el cual se entrenaría a partir de estos datos y del archivo en el que se tiene su clasificación.

Aplicado el clasificador sobre los datos del paciente, se obtienen los dos valores que da el algoritmo, el primero valorando los datos como posible señal, y el segundo valorando lo mismo como posible artefacto.

Partiendo de esta clasificación, de los datos EEG iniciales, la matriz de pesos y la de esfericidad se procede a reconstruir la señal, eliminando todas las señales que se han clasificado como ruido.

6.5.2.6 Modificaciones aplicadas al algoritmo

Por último, queda decir que en el planteamiento del artículo original, cuentan con que ciertas componentes pueden tener tanto ruido como señal, y que una aproximación

que elimine todos los elementos que se considere artefactos, podría eliminar también datos importantes, por eso, comúnmente se establece un threshold (0.8), de manera que solamente las componentes que sean consideradas como artefactos y lo superen sean eliminadas.

No obstante, la aplicación de un threshold se basa en el uso de MARA tanto en grabaciones en tiempo real, como en su uso como sistema de limpieza principal. Para sistemas sin tiempo real, y con procesamientos previos de la información como este, ya se ha eliminado la mayoría del ruido al llegar a este punto, de manera que los elementos que se consideran artefactos se reducen, tanto en cantidad como en el valor que obtienen del clasificador lineal. Teniendo en cuenta esto, se ha decidido obviar la aplicación del threshold, eliminando los componentes que se clasifiquen como artefacto.

Por último, en el artículo también se plantea la aplicación de una PCA, para reducir la cantidad de componentes existentes, eliminando así pequeñas variaciones de la señal antes de aplicar ICA. No obstante, se ha descartado su aplicación por motivos similares al threshold.

6.5.3 Variaciones respecto a la planificación

El cumplimiento de este requisito se planteó como una tarea a implementar mediante C++ y bibliotecas de álgebra lineal e inteligencia artificial, estimando la duración del trabajo en ocho semanas.

La estimación de las tecnologías involucradas ha sido correcta, con la excepción del uso de bibliotecas de inteligencia artificial, las cuales resultaron no ser adecuadas al problema.

En cuanto al tiempo estimado, no se pudo cumplir, pasando de las ocho semanas estimadas a las once necesarias.

6.6 CAR [Req07]

6.6.1 Técnica

CAR es el acrónimo de Common Average Reference. Esta es una técnica que permite encontrar fuentes de pequeñas señales en grabaciones muy ruidosas, así como una normalización de los datos del paciente, de manera que se puedan comparar mejor los resultados con los de otros pacientes. [34]

Es una técnica basada en el cálculo de la media de voltaje entre todos los electrodos y todos los puntos de la grabación. De esta manera, solamente las señales y ruidos

comunes a todos los canales se quedan en la media calculada, no teniendo en cuenta las señales aisladas.

Las principales formas de aplicación difieren en cómo trabaja el algoritmo. La aproximación más básica, es el cálculo de la media de todas las demás muestras en ese instante de tiempo y aplicarla sobre la señal con la que se está trabajando.

La segunda aproximación consiste en analizar lo ruidoso que es el canal antes de añadirlo a la media.

Esto se realiza calculando el RMS de una ventana de los datos y comparándolo con la media de los RMS en esa ventana para cada uno de los diferentes canales, de manera que si el valor del RMS del canal que se calcula es entre 0,3 y 2 veces el valor de la media de los RMS de los diferentes canales, el canal se considera bueno, y se utiliza para la media del CAR.

6.6.2 Funcionalidad implementada

Para la implementación de la funcionalidad, se estudiaron las diversas formas de aplicar la técnica, junto con las ventajas e inconvenientes de cada una, llegando a la conclusión de que se debía implementar la primera versión del algoritmo.

Esto es debido a que las desventajas del primer sistema son solamente un peor funcionamiento en sistemas con mucho ruido y en tiempo real.

No obstante, para llegar a este último algoritmo, se habrán aplicado alguno de los métodos previos, como el promediado, de manera que no será un sistema excesivamente ruidoso.

La otra desventaja reside en la aplicación del algoritmo sobre sistemas en tiempo real. Ya que el sistema no lo es, no se considera relevante.

6.6.3 Variaciones respecto a la planificación

El cumplimiento de este requisito se planteó como una tarea a implementar mediante C++ y bibliotecas de álgebra lineal, estimando la duración del trabajo en tres semanas.

La estimación de las tecnologías involucradas se ha cumplido correctamente. La estimación de tiempo resulto ser elevada, reduciéndose el tiempo empleado a dos semanas.

7 Manual de usuario

Dado que la misión principal de la herramienta es simplificar el trabajo a personal no especializado en informática, manteniendo las prestaciones que se han mencionado a lo largo del proyecto, se ha intentado tener una interfaz simple y correctamente estructurada.

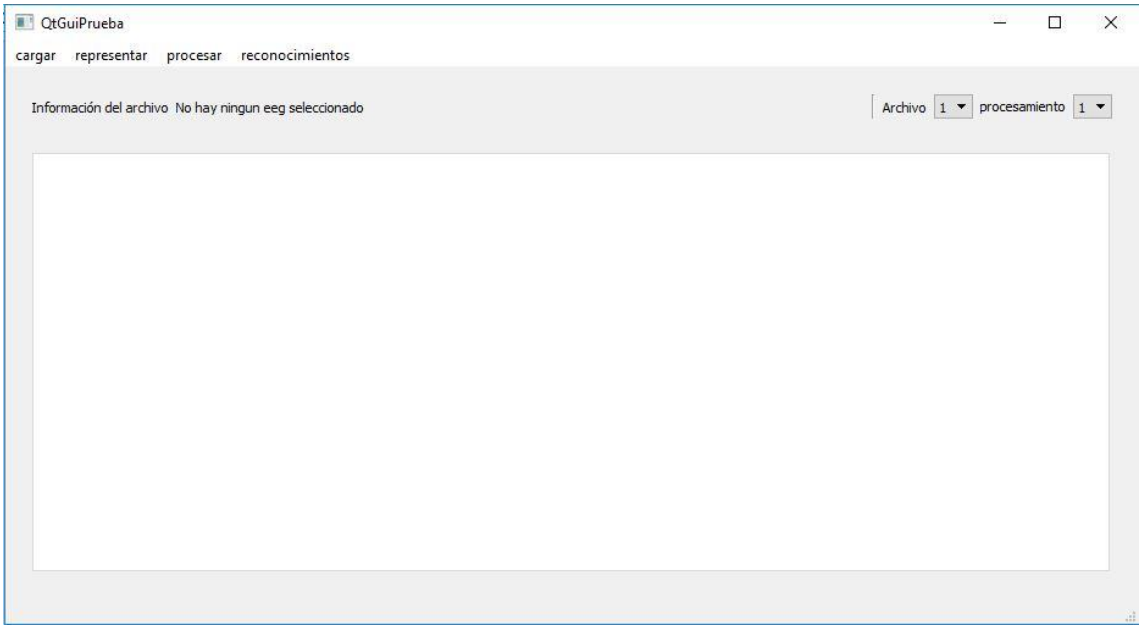

Ilustración 26: Interfaz al abrir el programa

Para ello, hay diversas partes que se comentaran a continuación.

7.1 Interfaz de información

Aporta datos respecto al EEG seleccionado y permite seleccionar los datos con los que se quiere trabajar.

El apartado de información del EEG muestra el EEG seleccionado actualmente, así como información acerca del procesamiento que se le ha aplicado.

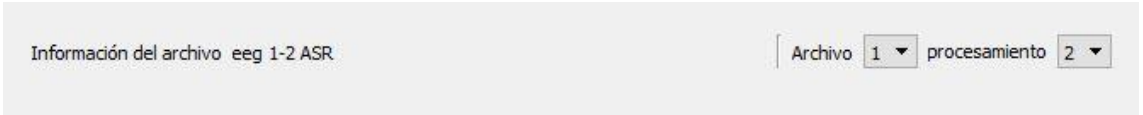

Ilustración 27: Interfaz de información y selección de archivo

El combobox de archivo permite seleccionar el archivo con el que se quiere trabajar de entre todos los que se han cargado en el sistema.

El combobox de procesamiento permite elegir entre todos los procesamientos aplicados, de manera que se pueda recuperar de cualquiera de los procesamientos por los que se ha pasado previamente el sistema.

7.2 Menús y funcionalidades

En esta parte hay diferentes funcionalidades agrupadas, de manera que se han realizado varios grupos independientes. Además, se han planteado ciertas funcionalidades que se consideran apropiadas, pero que no se han podido implementar.

7.2.1 Cargar

El primero de estos menús es el de cargar, conteniendo dos funcionalidades:

- Cargar archivo

Funcionalidad para importar datos al sistema. Esta opción abre una ventana del sistema para la selección de archivos, restringiendo las posibles elecciones a archivos con formato DAT y GDF.

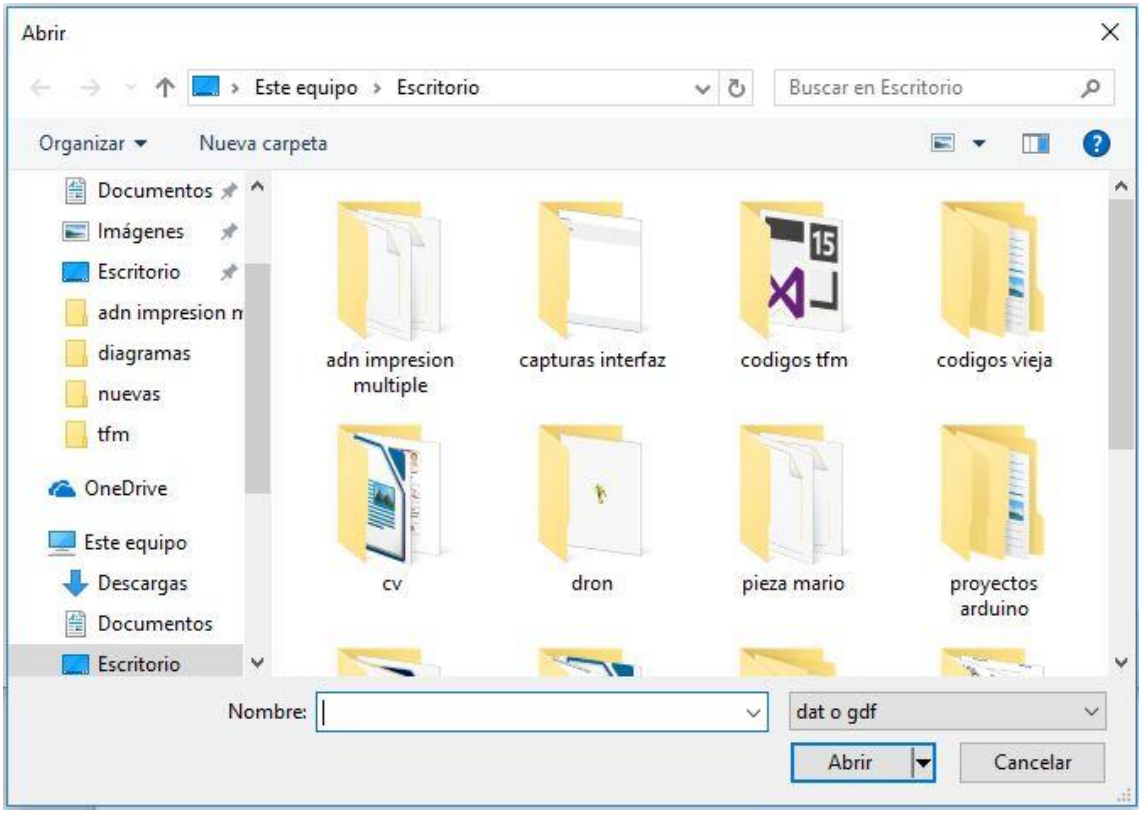

Ilustración 28: Selección de archivo de EEG

Una vez seleccionado un archivo, procede a realizar su carga en el sistema. Una vez completada la tarea, repite la operación para la selección del archivo CED.

La carga de ambos archivos implica la existencia de datos con los que comenzar a trabajar.

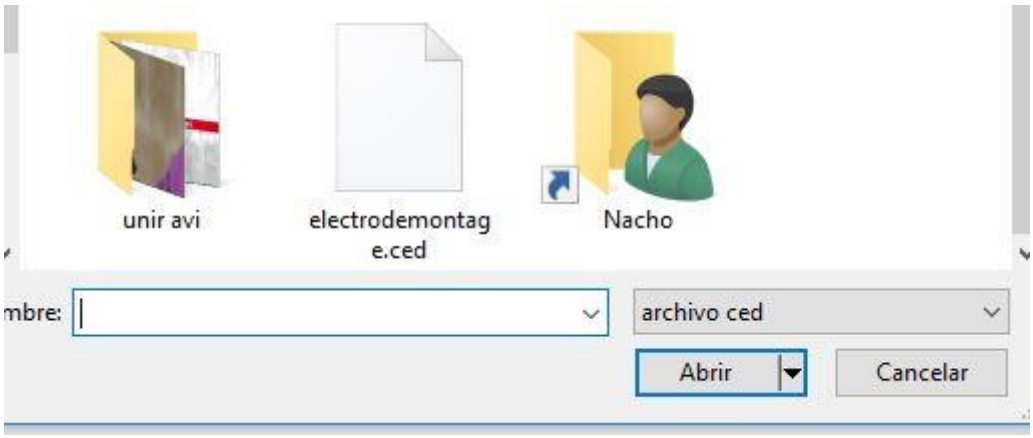

Ilustración 29: Selección de archivo de formato CED

- Borrar

Esta es una funcionalidad no implementada. La intención es que se pueda borrar el procesamiento seleccionado, de manera que se permita aliviar la carga de memoria que utiliza el sistema.

7.2.2 Representar

Este menú solamente tiene la opción de elegir los canales a representar.

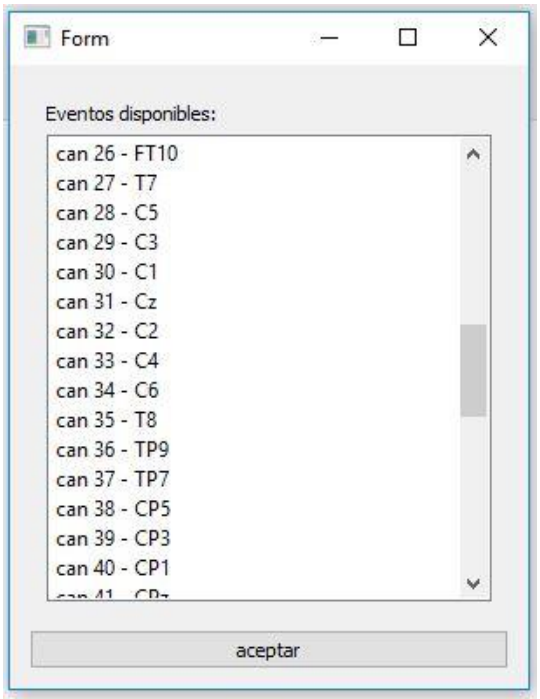

Ilustración 30: Interfaz para la elección de los canales a representar

Partiendo del EEG seleccionado, se ofrece la lista de los canales y su nombre según el estándar 10-20 para que el usuario elija aquellos que desea representar.

Una vez seleccionados, procede a representarlos mediante un Qchart en la parte inferior de la interfaz, añadiendo una nueva pestaña, de manera que se pueda volver a los diferentes gráficos mostrados a lo largo de la interacción con la herramienta.

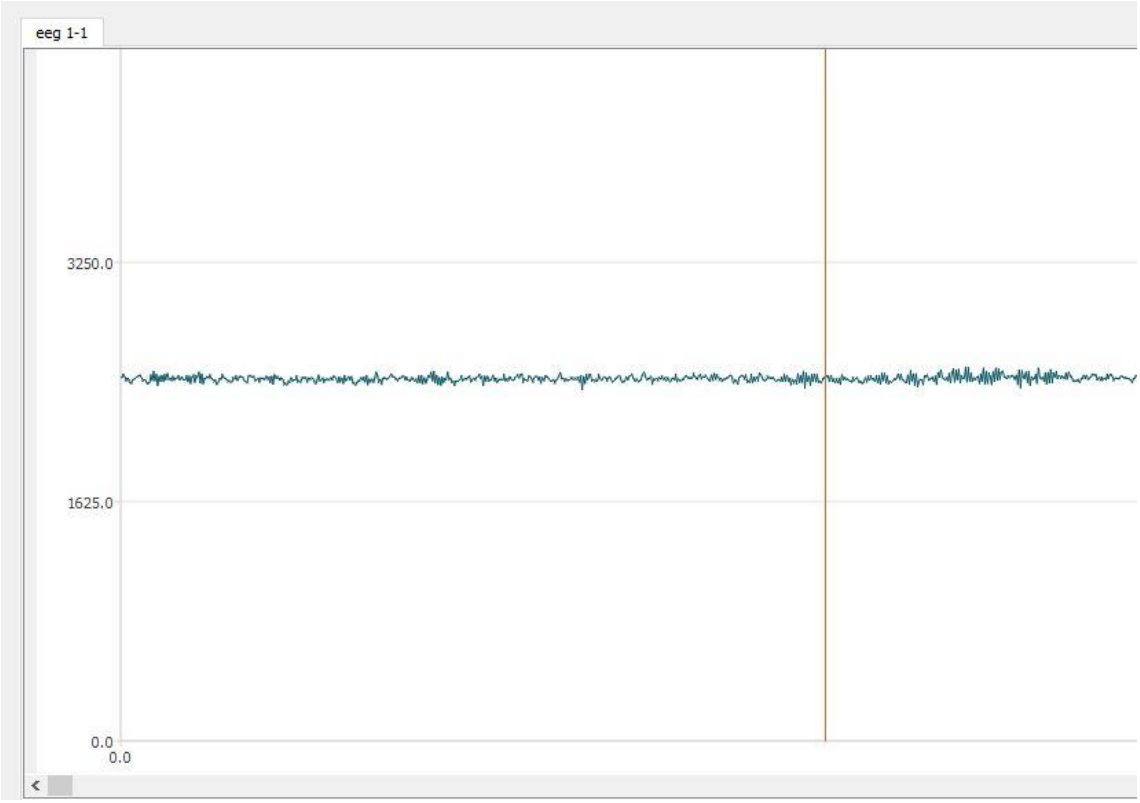

Ilustración 31: Representación de uno de los canales

7.2.3 Procesar

En este menú se agrupan las diferentes técnicas implementadas en el proyecto, las cuales van a añadir nuevos procesamientos al sistema y a modificar los datos existentes.

- Aplicar ASR

Este algoritmo se utiliza para encontrar y corregir picos y partes del archivo en las que hay una desviación de los datos con respecto a una muestra poco ruidosa de los datos.

La aplicación de este algoritmo está condicionada a no haber aplicado previamente la técnica de promediado, ya que de ser así no habría datos suficientes para la aplicación de la técnica.

- Aplicar filtro frecuencial

Mediante la opción de filtro frecuencial se da la opción de aplicar un filtro a los datos.

Estos pueden elegir el tipo de filtro que quieren aplicar, y las frecuencias sobre las que va a trabajar.

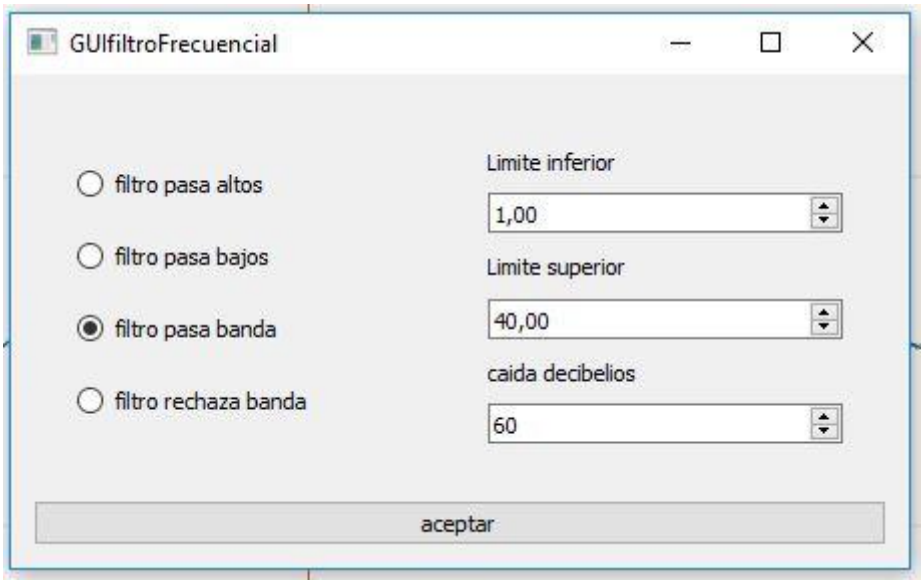

Ilustración 32: Interfaz de configuración de los filtros frecuenciales

Esta técnica puede ser usada sobre los datos cuando y como se quiera, sin ninguna restricción de orden.

- Aplicar promediado

Con la opción de promediado se permite elegir el tipo de evento sobre el que promediar, así como la cantidad de milisegundos que se van a utilizar en el promediado.

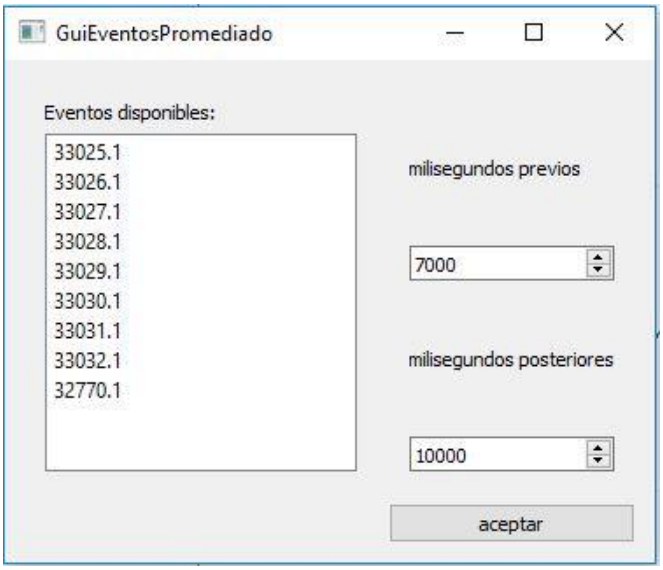

Ilustración 33: Interfaz para la elección del tipo de evento sobre el que promediar

Al acabar de promediar, en la estructura de datos resultantes es la misma, pero la cantidad de datos existentes es menor, ya que mantiene como datos solamente los resultantes tras aplicar el promediado, pasando, por ejemplo, de 285 segundos a 17, en el caso mostrado en la imagen.

Otro cambio relevante, es que se borra la tabla de eventos, ya que al cambiar los datos, los eventos no se van a corresponder con los datos existentes. Esto implica que solamente se va a poder realizar un promediado a los datos.

- Aplicar MARA

La aplicación de MARA debe quedar condicionada a la aplicación del promediado. Los datos sin promediar contienen tal cantidad de datos a lo largo de tanto tiempo que este algoritmo no funcionaría de manera apropiada, además de trabajar con una lentitud exagerada.

Dado que previamente se aplican ciertos procesamientos, es raro que quede un artefacto en los datos. No obstante, de ser así, este artefacto debería tener una fuerza relevante, además de estar en todos los *epoch*. Mediante la aplicación de MARA, este tipo de artefactos deberían ser detectados y eliminados fácilmente.

- Aplicar CAR

La aplicación de CAR queda relegada a la normalización de los datos para que puedan ser comparados con los de otros experimentos. Esta debería ser la última técnica aplicable a los datos.

- Aplicar análisis frecuencial

Por último, se planteó un posible estudio de los datos resultantes en un formato frecuencial en lugar de temporal, de manera que mediante la combinación de representaciones de los datos temporales en gráficas, y de los datos frecuenciales en mapas de calor, se pueda alcanzar un mejor estudio del estado del paciente.

No obstante, esta funcionalidad no se pudo completar.

7.2.4 Reconocimientos

Este apartado se ha planteado para mostrar todos los reconocimientos de los diferentes sistemas que se utilizan, así como las licencias que necesitan. No obstante, esta funcionalidad esta sin implementar.

8 Instalación, importación y ampliación del proyecto

8.1 Despliegue y requisitos del programa

A la hora de llevar el programa a un nuevo ordenador, hay que tener en cuenta que no bastaría con mover el ejecutable de sitio. El proyecto contiene varias dependencias, así como archivos que son necesarios para el correcto funcionamiento del sistema.

Ejemplo de esto son los archivos binarios que contienen los datos base para aplicar mara, o las dependencias que el proyecto tiene con QT.

De esta manera, sería necesario copiar la carpeta completa del programa, duplicando así junto con el ejecutable sus múltiples dependencias.

No obstante, el motor de Matlab es un recurso que no se puede enlazar e incluir en la compilación, de manera que se debe buscar dinámicamente en el sistema sobre el que se ejecuta el programa.

Para tal fin, es necesario tener instalado Matlab en el sistema. Dentro del entorno se debe buscar la carpeta que contiene la interfaz para que C++ pueda trabajar con Matlab. Esta tendrá una ruta similar a "*.\MATLAB\R2016a\bin\win64*" en la cual está alojado el archivo libEGL.dll necesario.

Para que la herramienta pueda encontrarlo, es necesario que la ruta absoluta, incluyendo la carpeta que contiene al archivo, sea añadida al PATH del sistema.

Mediante este proceso, el sistema es portable entre diferentes entornos Windows sin necesidad de instalarse en él. En este caso, se han comprobado y validado las versiones 7 y 10 de este sistema operativo, ambas con Matlab versión R2016a.

8.2 Manual de importación del proyecto

A continuación se explican los requisitos para poder importar el proyecto en el estado actual sobre otra máquina, de manera que se pueda seguir trabajando y ampliando sus funcionalidades.

El primer paso es descargar e instalar Visual Studio 2017 community edition. Esta es la versión con la que se ha trabajado en diferentes ordenadores, de manera que se recomienda que sea esta versión. No obstante es posible que el proyecto funcione correctamente sobre otras versiones del programa.

Después es necesario instalar en el sistema la biblioteca QT 5.10 a través del instalador de escritorio. A lo largo del asistente de instalación se preguntara cuáles son los complementos que se quieren añadir a la selección por defecto. Es importante marcar en este paso las opciones QtCharts y msvc2017.

Otro elemento necesario en el sistema es Matlab, siendo Matlab R2016a la versión que se ha utilizado. Otras versiones de Matlab no han sido probadas.

Una vez instalados todos los elementos del entorno hay que proceder a configurar Visual Studio para que pueda trabajar correctamente con QT.

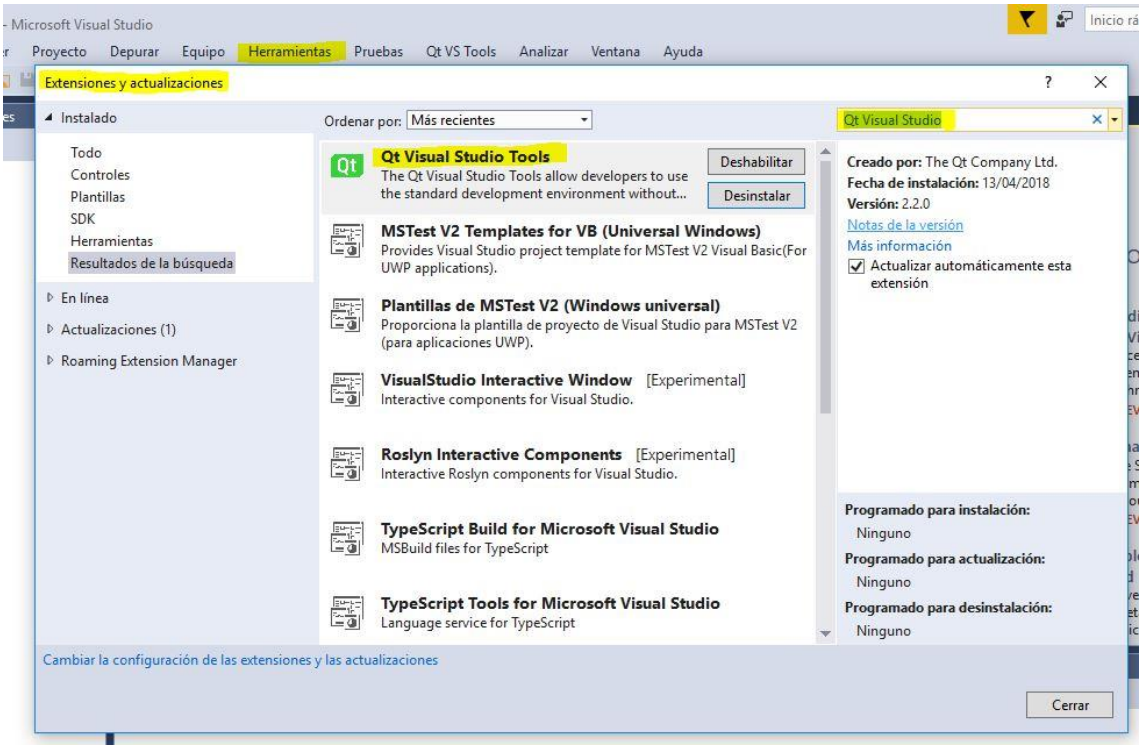

Ilustración 34: Herramientas -> extensiones y actualizaciones -> en línea -> Qt Visual Studio Tools

Hay que abrir el programa e instalar el plugin de QT llamado "*Qt Visual Studio Tools*". Para esto se sigue la siguiente lista de menús y opciones.

Una vez instalado el plugin, hay que reiniciar Visual Studio, de manera que al hacerlo aparece un nuevo ítem en la barra de menú, llamado "*Qt vs tools*".

Se selecciona este menú, entrando en la opción "*Qt options*", y dentro de esta, la opción "*añadir un nuevo path*".

Este path que añade es tiene que enlazar con la carpeta en la que se ha instalado msvc2017, que es una de las opciones que se ha instalado en QT con el asistente.

Si no se ha cambiado durante la instalación, QT se instala en la raíz del disco duro. El resultado es la ruta "*C:\Qt\Qt5.10.0\5.10.0\msvc2017_64*". En caso de haber

cambiado la carpeta de instalación, se debería buscar esta ruta en la carpeta que se ha instalado.

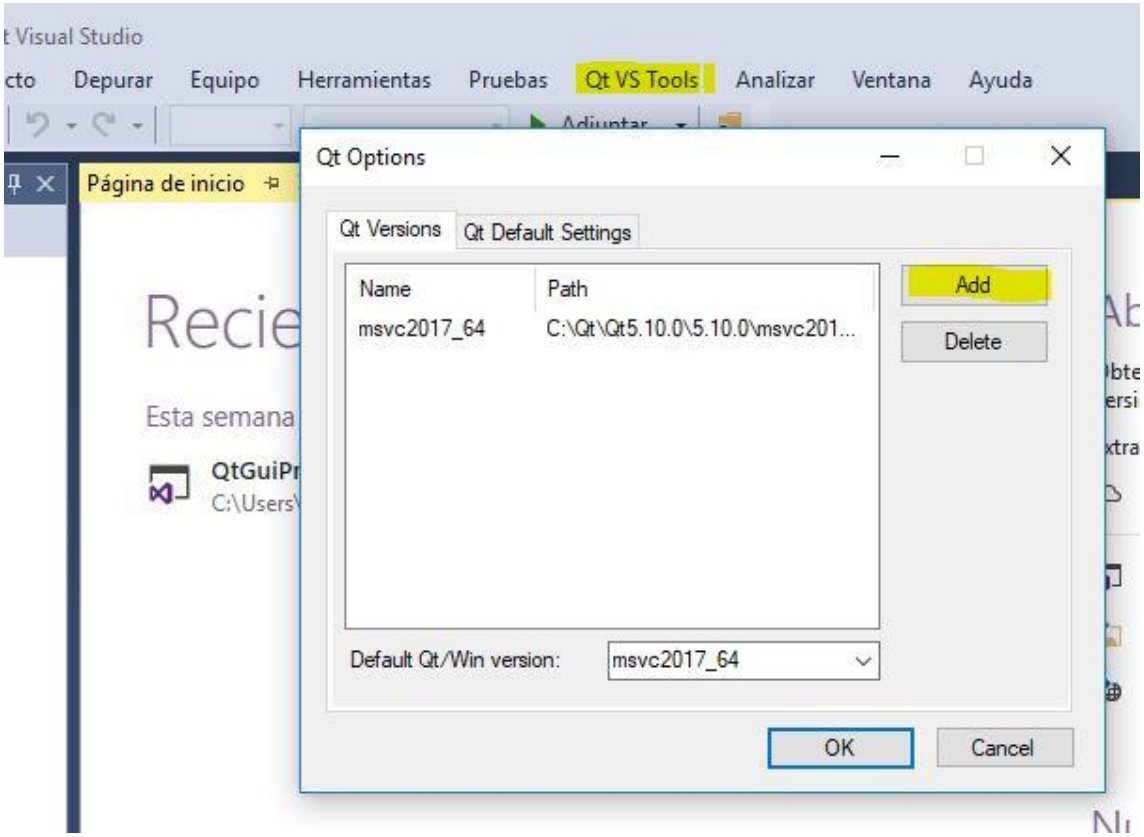

Ilustración 35: Configuración del path de QT en Visual Studio

Completada la configuración, se procede a abrir el proyecto en Visual Studio.

Si se compila nada más abrir el proyecto, este da fallo por ausencia del archivo python36_d.lib. Esto es debido a que la biblioteca de Python no soporta que el proyecto esté en modo debug, de manera que se debería cambiar a modo release.

Junto al código del proyecto, se ha aportado la carpeta de la biblioteca SigPack. Esta carpeta se debería copiar junto con el proyecto, y se deberían actualizar las rutas de inclusión y dependencia para que el proyecto pueda encontrar la carpeta. Las nuevas rutas son la ruta absoluta hasta la carpeta que contiene el archivo sigpack.h.

En caso de que Matlab o QT se hayan instalado en rutas diferentes, se deberían también actualizar sus rutas de inclusión y dependencia.

Por otro lado, se debería revisar la versión SDK de Windows sobre la que se está desarrollando. Las nuevas instalaciones traen por defecto la última disponible, pero al importar un proyecto, este mantiene como opción a elegir el SDK previo sobre el que se trabajaba, con lo cual es tan simple como elegir la opción que no está por defecto. La ruta para revisar esto sería la siguiente:

proyecto -> propiedades -> general -> version sdk

No se ha probado con nuevas versiones, pero no deberían dar problemas.

Por último, la compilación del código no hace que QT añada al proyecto las dependencias que utilizan, de manera que aunque el proyecto compile no va a arrancar. Esto ocurre cada vez que se quiere utilizar un nuevo componente de QT, como por ejemplo, su core o las gráficas.

Para solucionar esto, es necesario:

- Compilar el código.
- Abrir una consola de Windows.
- Ir en la consola a la carpeta del ejecutable.
- Ejecutar el comando "*windeployqt programa.exe*", sustituyendo programa por el nombre del archivo ejecutable. [35]

De esta manera, QT se encarga de añadir a la carpeta las dependencias del proyecto.

8.3 Organización y estructura de la aplicación [Req08]

8.3.1 Datos y acceso

El modelo de datos básico es el objeto EEGdata. Este junto con sus diferentes dependencias, aportan toda la información acerca de los datos de un paciente con un procesamiento concreto.

Dado que los diferentes procesamientos dan diferentes modificaciones en los datos, y no interesa perder el estado de los datos, estos se almacenan en una estructura de datos almacenada dentro del objeto Main_window.

Para tal fin, se ha planteado una lista de listas.

En la lista de listas, cada lista contenida se va a asociar con un nuevo archivo cargado al sistema, de manera que va a almacenar los datos puntuales de un paciente.

Dentro de esta lista, se van a ir almacenando las instancias de EEGdata que almacenan los diferentes procesamientos de los datos del usuario.

8.3.2 Gestión e interacción de los gráficos

El trabajo con las interfaces gráficas a nivel de diseño se ha realizado mediante la herramienta de interacción de QT.

A la hora de interconectar estos elementos con el código, se ha utilizado SLOTS, de manera que la interacción con los elementos gráficos sirvan de trigger de diferentes funciones del código. Esta asociación de funciones con triggers, se realiza en el constructor de la clase.

Cada uno de los SLOT del sistema necesitan una declaración dentro del apartado public Q_SLOT del archivo .h. En este apartado se puede encontrar la lista completa de ellos.

Caso especial son las diferentes ventanas emergentes utilizadas a lo largo del sistema. Dado que se utilizan para el paso de información, se las ha dado cierta autonomía, bloqueando el acceso a la ventana principal mientras estas están activas, y dándolas acceso a funciones de Main_window para comunicar la información que sea relevante. Los SLOT asociados a una ventana emergente son fácilmente reconocibles, ya que llevan la palabra gráfico al inicio del nombre de la función.

Para controlar el fin de la ventana hay dos formas. La primera, cerrarla, de manera que libera el foco de la aplicación y vuelve a la ventana principal sin hacer nada más.

La segunda seria configurar las variables y darle a aceptar. Este botón lleva asociado una función SLOT que se encargaría de recuperar la configuración realizada, cerrar la ventana y pasarle la información a Main_window.

Una vez recuperada la información, Main_window es el encargado de redireccionar a la función que deba llamar.

Hay que decir que no todos los SLOT de Main_window están asociados a una ventana gráfica. Cuando no es así, están asociados directamente a la función que aplica la tarea que se desea realizar.

Por otro lado, en la parte superior existen dos combobox. Cualquier cambio en ellos está asociado a un trigger diferente.

Un cambio en el primero, de ellos provocaría la actualización del segundo. Partiendo del índice seleccionado, se extrae la lista correspondiente de la lista de listas. El tamaño de la lista extraída es el límite de opciones que se le da a este segundo combobox, de manera que nunca se pueda elegir un índice que no está en el sistema. Por defecto, se establece en la lista el 1, ya que es un valor que va a estar seguro en la lista.

Un cambio en el segundo de ellos hace que se actualice la información mostrada en la interfaz acerca del EEGdata seleccionado, mostrando un mensaje con el procesamiento que este archivo ha sufrido.

Por ultimo hay un QtabWidget. Este es el área más grande de la pantalla es donde se representaran las diferentes gráficas generadas mediante pestañas, pudiendo trabajar con diferentes gráficas a la vez.

8.3.3 Procesamientos y técnicas

Desde la clase Main_window hay acceso a todas las técnicas de procesamiento de la información disponibles en el sistema. El acceso a ellas es a través de los elementos comentados en el apartado previo. Estas funciones se pueden identificar fácilmente, ya que llevan la palabra “lanzar” al principio del nombre de la función.

Acerca del funcionamiento, el comportamiento es bastante genérico.

- Primero, se obtienen los índices almacenados en los combobox de la interfaz.
- Segundo, a través de estos índices se recupera el EEGdata seleccionado.
- Tercero, se aplica la técnica elegida.
- Cuarto, se pasa el objeto por el constructor por copia. Este paso no es necesario, pero si recomendable.
- Y por último, se almacena el objeto en la lista necesaria. El índice de esta lista es el mismo que el almacenado en el primer combobox, el cual ya se ha recuperado previamente. La función “addNewEEGtoColumn” se encargaría de todo el trabajo de añadir y disparar eventos para que se actualicen los índices de los combobox.

8.3.4 Carga de un nuevo archivo

Para el caso de cargar un nuevo archivo, se procedería a crear un objeto EEGdata a partir de los archivos seleccionados.

El método de selección del archivo es la ventana de archivos del sistema. Esta ventana se lanza en el método “elegirArchivo”. El único parámetro que necesita esta función es el filtro del tipo de archivos que se desea mostrar. El resultado es la ruta completa hasta el archivo.

El método “selectFileReader” se encargara de decidir cuál de los de archivos del programa es el más adecuado para la carga del archivo (GdfFileReader o DatFileReader, ambos herederos de la clase FileReader). Utilizando el método “loadFile” de esta clase se crea un EEGdata.

Una vez creado el EEGdata, es necesario repetir el proceso de selección del archivo para el formato CED. La llamada al método “loadChannelLocations” junto con el

EEGdata provisional y la ruta al archivo CED, realizará el trabajo restante para completar los datos.

Una vez completado el proceso, es necesario aplicar el constructor por copia de EEGdata para aumentar el alcance de la instancia y evitar el borrado de alguna variable local utilizada en el proceso.

Por último, quedaría añadir una nueva lista, en la cual se almacenaría este EEGdata, añadirla a la lista de listas, y actualizar los índices de los combobox.

8.3.5 Representación gráfica

Por último, queda la interacción que lleva a una representación gráfica.

Este trabajo se realiza en la función “dibujarCanalesSeleccionados”, la cual recibe la lista de canales que se desean dibujar por parte de la ventana gráfica emergente.

Partiendo de esa lista de canales y del EEGdata seleccionado, se procede a crear un nuevo Qchart.

Progresivamente, se extraen de EEGdata los canales seleccionados uno a uno de la matriz de datos, añadiendo estos puntos a un objeto Qline mediante la función “generarQline”. Una vez obtenido el objeto Qline, se le añade al objeto Qchart.

En caso de que la tabla de eventos del EEGdata no sea nula, es necesario añadir líneas verticales en el gráfico en cada uno de los momentos en los que ocurre un evento. El color de la línea, está almacenado dentro del evento a dibujar, y la función encargada de crear el Qline correspondiente es “generarDibujoEvento”.

Ahora que tenemos el Qchart, queda añadirlo en un QChartView, y este en un QScrollArea.

El QScrollArea es el objeto que se añade al QTabWidget, de manera que se puedan mantener diferentes dibujos, en diferentes pestañas, y con scroll en la imagen en caso de ser esto necesario.

8.4 Diagrama de Gantt final

En el diagrama de Gantt final se adjuntan los tiempos de desarrollo finales en cada uno de los requisitos del proyecto, así como algún otro elemento que haya consumido un tiempo relevante.

Ilustración 36: Diagrama de gantt final

8.5 Diagrama de casos de uso

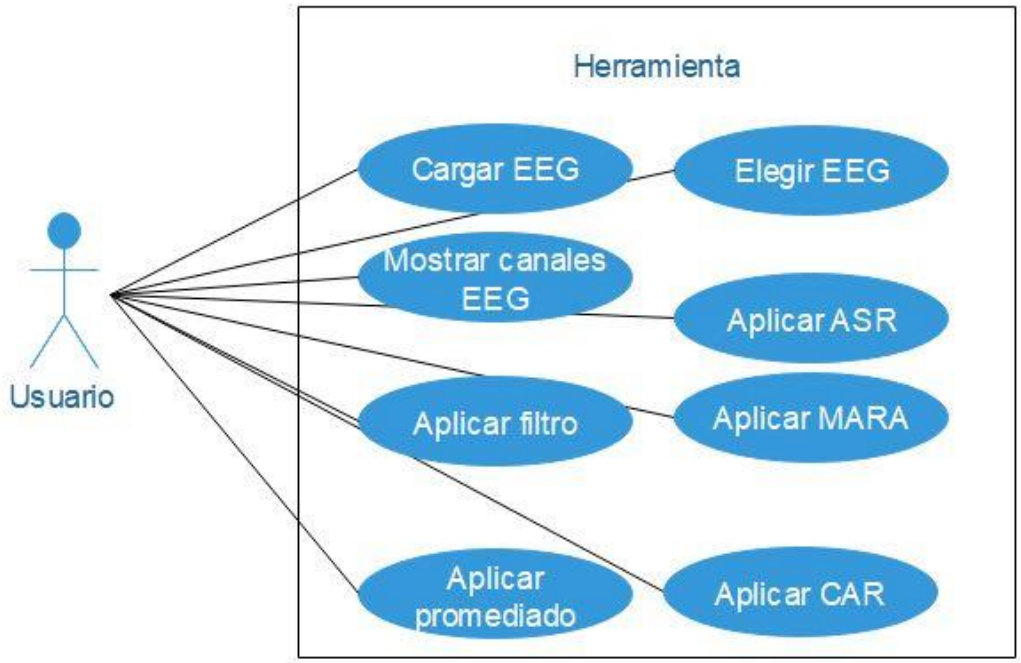

Ilustración 37: Diagrama de casos de uso

8.6 Diagrama de clases

Para que el diagrama de clases quede de manera comprensible y con datos de un tamaño adecuado, se ha decidido plantearlo aquí en diferentes trozos. No obstante, en caso de querer consultar la versión completa, existe un anexo para tal fin.

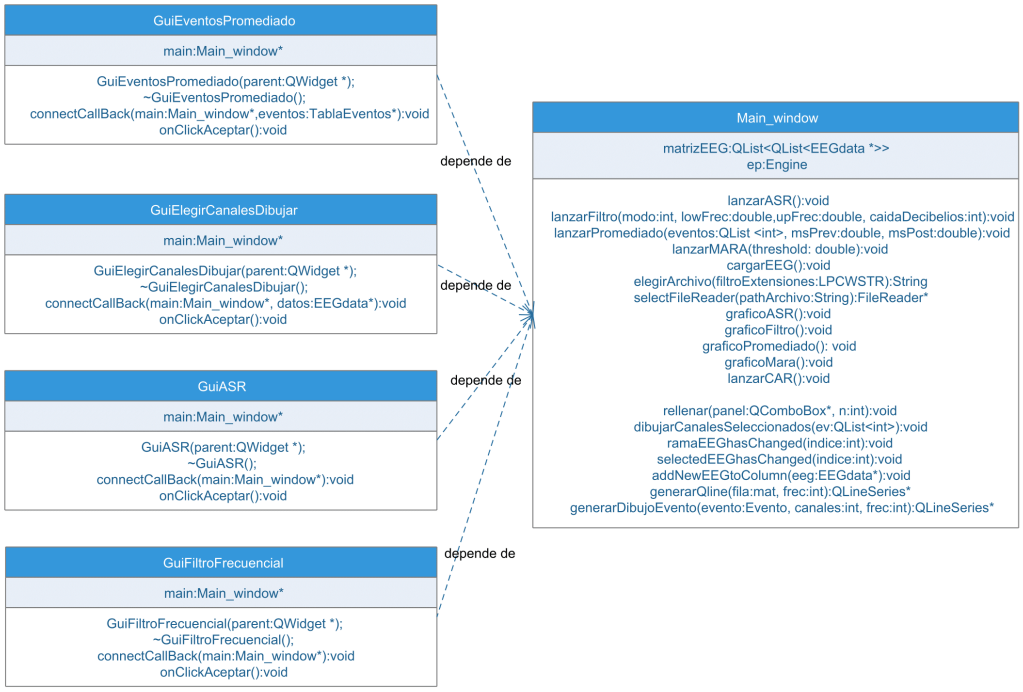

Ilustración 38: Diagrama de estructuración de las clases relativas a los elementos gráficos

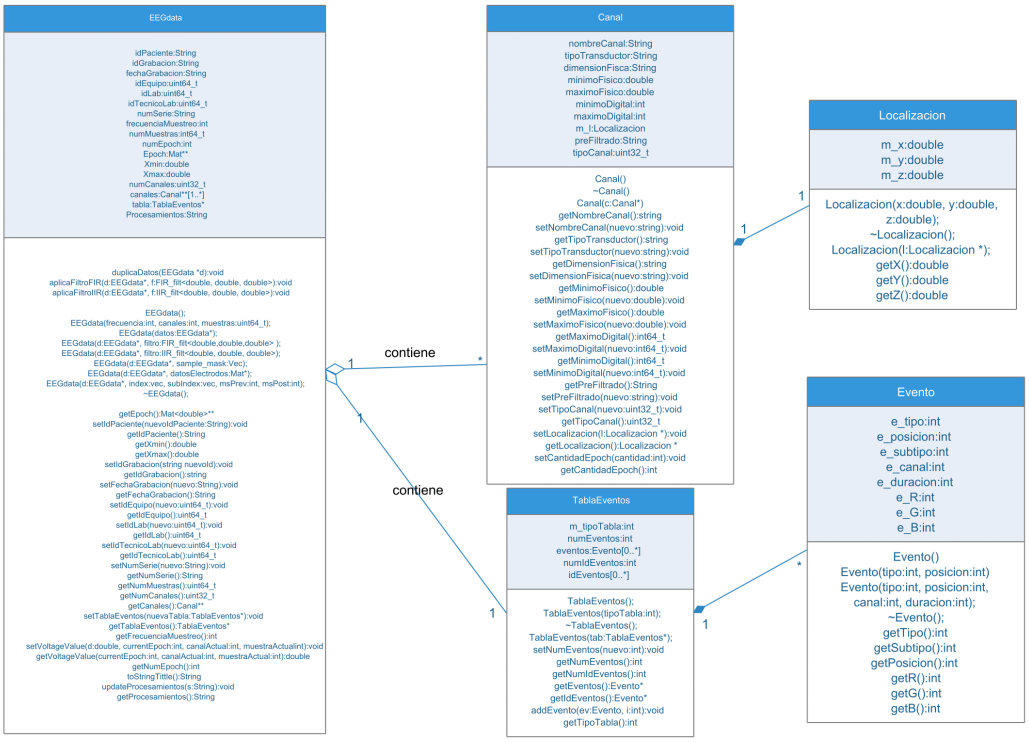

Ilustración 39: Diagrama acerca de la estructuración del modelo de datos utilizado en la aplicación

Ilustración 40: Relación entre la ventana principal, el modelo de datos y los algoritmos ASR y MARA

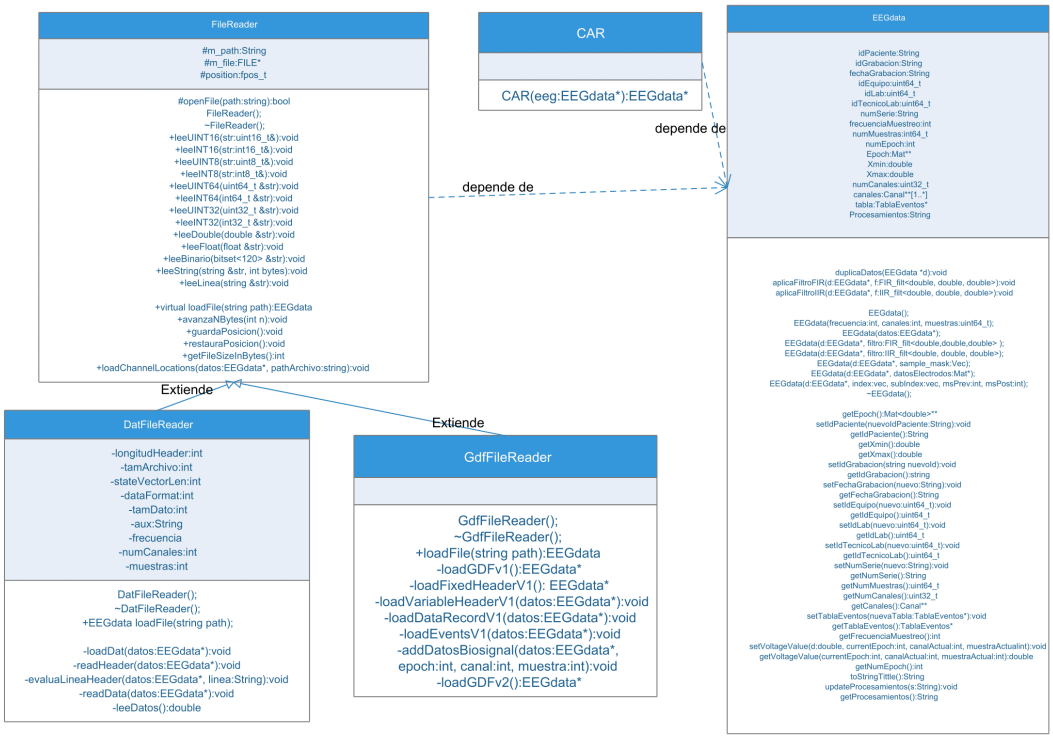

Ilustración 41: Relación entre los sistemas de carga de archivos y el modelo de datos

8.7 Diagramas de secuencia

Debido a la misma problemática de tamaño con los diagramas de clases, los diagramas de secuencia están disponibles en un anexo al final del presente documento.

9 Discusión y trabajo futuro

9.1 Discusión

El proyecto ha avanzado hasta llegar a un estado maduro y funcional, no obstante, a lo largo del trabajo ha habido multitud de retos que superar y problemas con los que lidiar.

9.1.1 Problemas con el formato de los archivos

Partiendo de las propuestas originales, se planteaban dos posibles formatos, entre los que constan los formatos GDF y DAT.

El formato DAT es un formato más general para guardar datos, no solamente estímulos neuronales. No obstante, aunque el formato estaba bien documentado para muchos elementos, no se encontró documentación adecuada acerca de la organización de los valores medidos por los sensores. Así que, por falta de documentación se tuvo que descartar este tipo de archivo.

En cuanto al GDF, a la hora de empezar con este formato se partió de la última documentación encontrada. No obstante, para este formato de archivo se completó solamente una interfaz, sin llegar aportar una funcionalidad completa. Esto se debe a que al ver que la parte construida no funcionaba bien, se observó que el formato GDF constaba de diferentes versiones. La versión que se comenzó a implementar fue la versión basada en GDF 2, de manera que no era compatible con los archivos recibidos, los cuales eran GDF 1.

De esta manera, aunque se ha preparado la aplicación para una posible implementación de los demás formatos, el que soporta actualmente es el GDF 1. Si bien es cierto que lo ideal sería la implementación de los tres formatos, el sistema de grabación, como se ha comentado previamente, permite generar tanto GDF 1 como DAT, de manera que mediante la implementación del formato GDF 1 se permite conseguir que la carga se realice de forma correcta, consiguiendo así una interfaz que carga los datos del archivo al programa.

9.1.2 Problemas con ASR

Respecto a la parte de algoritmos y su funcionamiento, a la hora de conseguir que estos funcionasen correctamente, ha sido necesario combinarlos con una variedad de métodos que hacen cálculos matemáticos o elementos de álgebra lineal, además de probar su funcionamiento.

Concretamente, la técnica ASR se creó en Matlab partiendo de ejemplos de implementaciones del algoritmo ya existentes integradas en técnicas más complejas. No obstante, para el proyecto no eran necesarias estas funcionalidades extras, de manera que se avanzó conservando la funcionalidad del ASR y eliminando las partes que no eran necesarias para el proyecto.

Ejemplos de esta funcionalidad eliminada es la supresión de canales que se consideran excesivamente ruidosos o irrelevantes. De esta manera se llegó a tener una versión funcionando en Matlab, con la que se trabajó antes de comenzar a migrar el código a C++.

No obstante, este algoritmo necesita el cálculo de algunos datos mediante funciones matemáticas concretas. En este caso, el problema lo presentó la función de la inversa incompleta de la distribución gamma, para la cual no se encontraron implementaciones en C++.

Esto, junto con la necesidad del uso de filtros infinitos de yulewalk para ampliar el ruido, dentro del algoritmo, hizo necesario añadir Matlab como biblioteca dentro de C++, de manera que se puedan utilizar dentro del programa código escrito en Matlab, sin necesidad de convertir todo el algoritmo, impidiendo así poder migrar completamente el código a C++.

Aun así, el uso de Matlab para el cálculo de la inversa incompleta de gamma devolvía error en tiempo de ejecución por símbolo desconocido, con lo que para evitar este fallo se procedió a calcular este valor mediante una función de Python llamada desde C++.

9.1.3 Problemas con los filtros

Respecto a los filtros frecuenciales, la primera implementación de un algoritmo para el cálculo de los valores de un sistema de filtrado se hizo mientras se estaba aprendiendo el uso de la biblioteca de álgebra lineal Armadillo. La implementación se hizo partiendo del código de Matlab para la creación de un filtro pasobajo, aplicando una distribución de Hamming.

Posteriormente se investigaron las bibliotecas que se listaban como dependientes de Armadillo, encontrando SigPack, la cual contenía métodos de generación de filtro pasobajo y métodos que aplicaban el filtro. A la biblioteca se le encontraron carencias, ya que solamente mantenía implementación para los filtros pasobajo, aunque gracias a un email de consulta se amplió rápidamente la biblioteca y se incluyeron los filtros restantes.

De esta manera, al tener una biblioteca que proporcionaba métodos de creación y uso de los filtros necesarios, y apoyándose en los sistemas de álgebra lineal ya disponibles en el entorno, se decidió aplicarla en lugar de crear o mantener métodos propios.

9.1.4 Problemas con ICA

Para ICA se probaron diversas bibliotecas de análisis de componentes independientes. Ejemplo de esto es Mlpack, junto con métodos independientes en C. No obstante, las pruebas de rendimiento sobre estos métodos no eran buenas, llegando a estar el algoritmo funcionando durante 5 horas sin dar resultados, de manera que se descartó el uso de estas librerías, pasando a estudiar la versión runica de Matlab.

Dado que las pruebas realizadas con el daban resultados en un tiempo aceptable, se migró esta versión a C++, dejándolo como implementación definitiva al ver que mantenía buenos resultados.

9.1.5 Problemas con las interfaces

Por otro lado, la parte gráfica ha sido necesaria, de manera que las personas que vayan a utilizar el programa no tengan que aprender a interactuar con una ventana de comandos, simplificando al máximo su trabajo.

Además el uso de una interfaz gráfica aporta la posibilidad de ver gráficas con los resultados.

Para tal fin se creó un nuevo proyecto en el cual se diseñó la interfaz, y al cual se le iban añadiendo los diferentes algoritmos desarrollados.

No obstante, esta parte también ha producido fallos al unir el sistema gráfico con el proyecto en el que se creaban y probaban los algoritmos, siendo necesario solucionarlos para su correcta ejecución. Ejemplo de esto son la volatilidad de los punteros al mezclar memoria estática y dinámica, hecho que puede provocar comportamientos incoherentes para alguien que no sea experto, amplificándose las incoherencias al mezclarlos con estructuras de datos abstractas. Estos problemas han causado varias reestructuraciones en la organización del código.

9.1.6 Problemas de enlazado de bibliotecas

A nivel más general, otro fallo común que se ha sufrido, es la facilidad de encontrar errores de enlazado a la hora de añadir una nueva biblioteca, o de mover partes de un proyecto a la hora de unirlo con otro. Afortunadamente, la mayoría de estos se han resulto gracias a que se ha encontrado una biblioteca con un enlazado más simple, o de manera posterior, enlazando a través de repositorios con Nuget.

El descubrimiento de Nuget para el enlazado de bibliotecas desde repositorio fue hacia la mitad del proyecto. No obstante, como el problema de enlazado de las bibliotecas

es de los más difíciles de resolver, se eliminaron los enlaces a todas las bibliotecas que se pudo, y se enlazaron mediante Nuget.

Las veces que no se ha podido realizar un enlazado con Nuget y ha habido algún problema, este se ha corregido revisando a fondo las dependencias y adaptando el enlazado de la biblioteca.

9.1.7 Problemas en la visualización

Además de estos fallos, hay comportamientos que se han tenido que cambiar.

Inicialmente se quería que según se procesase el archivo, el programa mostrase gráficamente el estado de los datos. Este comportamiento se llegó a implementar, y mantener durante algunas pruebas, no obstante, el consumo de memoria RAM se disparaba, llegando a consumir 4 GB para mostrar un solo procesamiento, y ralentizando la ejecución del programa.

Dado que dicho consumo se relaciona con la gran cantidad de puntos que se querían mostrar en las gráficas realizadas, y como no todos los datos son necesarios, se pasó a un sistema en el cual sea el usuario el que selecciona los canales que desea visualizar, y cuando desea hacerlo.

Como orientación acerca de la gran cantidad de puntos, queda decir que en los archivos con los que se ha probado el sistema, esta se encuentra alrededor de los 9 millones de datos (64 canales, y 146400 muestras por canal, aproximadamente).

9.2 Conclusiones

Como se planteó en la introducción, se partían de 5 objetivos, los cuales se separaron en dos grupos.

El primer grupo afecta a los datos y su procesamiento, teniendo así tres subobjetivos.

El primero consta de diseñar una estructura de datos genérica para almacenar los datos de cualquiera de los archivos planteados.

Si partimos de los formatos de los archivos, en su mayoría son similares, teniendo pequeñas variaciones en la organización de los eventos. Ejemplo de esto es que los diferentes eventos en los archivos DAT constan de índice y subíndice, mientras que en GDF solamente mantienen índice.

Esto se ha solucionado, generando un subíndice 1 para todos los eventos existentes en los archivos GDF, de manera que los datos en la aplicación se mantengan en el mismo formato, independientemente del origen de estos.

Si se vuelve al apartado Estructura de datos desarrollada [Req02], se puede revisar que esto está correctamente resuelto, consiguiendo de esta manera cumplir el objetivo.

El segundo consta de crear un sistema que realice correctamente la carga de los archivos binarios a datos del sistema.

Se considera que se ha conseguido cumplir el objetivo, ya que es posible cargar los datos desde el formato GDF 1, de manera que se pueden limpiar y procesar los datos aportados. Más información acerca de cómo se ha resuelto puede ser encontrada en el apartado Carga de los datos.

Problemas obtenidos acerca de la organización e implementación de los diversos sistemas de carga binaria se pueden leer en el apartado previo.

El tercero consta de conseguir una versión funcional de los diferentes algoritmos propuestos.

La información acerca de estos métodos se puede encontrar en el capítulo Algoritmos y funcionalidades propuestas. Los principales problemas sufridos a lo largo de la implementación o adaptación de las diferentes técnicas se han comentado en el apartado anterior.

Si bien algunos puntos de los comentados se consideran mejorables, el grado de satisfacción con los resultados es alto, considerando que el objetivo se ha cumplido.

En cuanto al segundo grupo, se centra más en la organización visual, tanto de la aplicación como de los resultados.

El primero consta de crear una interfaz simple e intuitiva que permita al usuario conocer las posibilidades del programa fácilmente, y saber lo que está haciendo en todo momento. Resultados acerca de esto se pueden ver en el Manual de usuario.

La elección del método a aplicar, es simple y rápida, teniendo todos los métodos agrupados en la misma pestaña, de manera que se pueda ver todo lo que se puede hacer con un simple vistazo, de manera que se considera que el objetivo se ha cumplido.

Ahora bien, cada uno de los métodos mantiene el nombre con el cual los expertos en el área se refieren a ellos, de manera que, si bien para ver las opciones del programa y utilizarlas no hace falta ningún conocimiento previo, sí que es necesario un conocimiento experto en la materia para elegir cual sería el más adecuado aplicar, o el orden en el cual deberían ser aplicados.

Por último, se planteaba la organización de los resultados de manera que se comprendan fácilmente.

Viendo que las gráficas de cada uno de los canales se muestran solo bajo petición del experto, de manera que no se muestren una ingente cantidad de datos innecesarios de canales irrelevantes, se considera que esa parte cumple con el objetivo.

9.3 Trabajo futuro

Como puntos pendientes, han quedado diversos elementos.

- Implementación de los formatos de archivo pendientes

Como se ha comentado a lo largo del proyecto, el formato implementado es el GDF1, no obstante, existen también los formatos GDF2 y DAT.

Con el hardware de grabación actual, la implementación del formato GDF2 no tiene mucho sentido, ya que este no es capaz de grabar en dicho formato. No obstante, la implementación del mismo implicaría que el programa pueda procesar archivos generados con sistemas más modernos.

Asimismo, dado que el sistema de grabación actual soporta tanto GDF1 como DAT, se considera apropiado también la implementación del formato DAT.

No obstante, tampoco se considera imperativo, ya que en Matlab existen implementaciones para convertir los archivos DAT en GDF1.

- Analizar y mejorar el consumo de memoria en el algoritmo ASR

El algoritmo ASR realiza cálculos de manera intensiva, y al aplicarlo existe un gran consumo de memoria. No obstante, queda pendiente por estudiar si el aumento del consumo de memoria RAM del algoritmo es debido a la necesidad de mantener abierto el motor de Matlab o de alguna posible fuga de memoria, de manera que sabiendo el origen del consumo, puedan aplicarse las medidas correspondientes para reducirlo.

- Análisis frecuencial de los resultados

Los resultados calculados quedan, tras acabar el procesamiento en formato temporal, al igual que como se cargan desde el archivo. Si bien en este formato se pueden observar resultados, su combinación con un análisis en frecuencia mediante transformadas de Fourier, así como una representación mediante mapas de calor de los datos puede dar otro punto de vista interesante, de manera que viendo los

resultados de ambas representaciones se pueda obtener una idea más exacta del estado del paciente.

- Eliminar dependencias de Matlab

El proyecto actual tiene integradas dependencias con un motor de Matlab para C++. La idea es reducir al máximo esa dependencia, de manera que se optimice el código y se ejecute todo en C++, sin necesidad de tener conversión de datos de un entorno a otro, y por consiguiente, reduciendo el tiempo de ejecución.

Otro punto a tener en cuenta es que la dependencia con Matlab implica tener el programa en el sistema sobre el que se trabaja, de manera que necesita una licencia activa. Una alternativa para cubrir este punto seria la migración de Matlab a Octave o algún lenguaje similar a Matlab pero de licencia abierta.

- Análisis de rendimiento de openBLAS contra Intel MLK y cuBLAS

El proyecto, actualmente hace uso de la implementación openBLAS de la biblioteca BLAS para optimizar el tiempo que tarda armadillo en realizar ciertas operaciones. No obstante, la elección de openBLAS fue por simplicidad a la hora de integrarlo con el proyecto y por ser esta de código abierto.

De esta manera, queda pendiente la integración del proyecto con cuBLAS (versión de la biblioteca con Nvidia Cuda integrado para hacer uso de la GPU) y con Intel MLK, para realizar una comparación de rendimiento. No obstante, ambas alternativas son código propietario de sus respectivas empresas, de manera que quedaría pendiente también la revisión de la licencia bajo la que se distribuirían dichas bibliotecas.

- Mayor adaptación al modelo MVC

En el sistema no se ha separado y estructurado correctamente como un MVC por falta de tiempo.

No se considera una tarea compleja, ya que las partes están claramente definidas, y el único trabajo restante sería llevar el modelo y sus funciones a un objeto aparte que no esté contenido en la vista.

- Mejoras en la visualización

Actualmente, la visualización de los datos aporta una figura, en la cual se muestran los datos de los canales seleccionados, pero todos ellos tienen que ser del mismo objeto. Esto implica, que para ver dos estados de un mismo canal, por ejemplo, antes y después de un determinado procesamiento, es necesario crear dos figuras diferentes y compararlas.

No obstante, se contempla como posible mejora, implementar la posibilidad de ver en el mismo gráfico datos que provienen de diferentes objetos, de manera que se puedan comparar en un mismo gráfico datos de diferentes procesamientos, o incluso comparar resultados entre diferentes pacientes.

Anexo 1: Formato GDF versión 1

channel type	size per sample [bytes]	code
char	1	0
int8	1	1
uint8	1	2
int16	2	3
uint16	2	4
int32	4	5
uint32	4	6
int64	8	7
float32	4	16
float64	8	17
bitN	N/8	255+N
ubitN	N/8	511+N
examples:		
Boolean 0=false, 1=true	1/8	256
ubit1	1/8	512
bit12	3/2	267
bit24	3	279

Ilustración 42: Diferentes formatos de datos utilizados en los archivos GDF

FIXED HEADER	remark	Position Start:End [bytes]	Bytes	Type
Version identification (GDF 0.12)		0	8	char[8]
Patient identification (P-id)		8	80	char[80]
Recording identification (R-id)		88	80	char[80]
Startdate and -time of recording (YYYYMMDDhhmmsscc)		168	16	char[16]
number of bytes in header record		184	8	int64
Equipment Provider identification (EP-id)	NEW	192	8	uint64
Laboratory Identification (Lab id)	NEW	200	8	uint64
Technician Identification (T-id)	NEW	208	8	uint64
reserved / serial number		216	20	char
number of data records (-1 if unknown)		236	8	int64
Duration of a data record, as a rational number in seconds (first the numerator, secondly the denominator.		244	8	uint32[2]
NS: number of signals (channels)		252	4	uint32
VARIABLE HEADER		256		
Label		256	NS*16	char[16]*NS
Transducer type		256+16*NS	NS*80	char[80]*NS
Physical dimension		256+96*NS	NS*8	char[8]*NS
Physical minimum		256+104*NS	NS*8	float64*NS
Physical maximum		"	NS*8	float64*NS
digital minimum		"	NS*8	int64
digital maximum		"	NS*8	int64
Pre-filtering		"	NS*80	char[80]*NS
nr: number of samples in each record	size only 4*NS	"	NS*4	uint32*NS
Channel TYPE	NEW	"	NS*4	uint32*NS
reserved		"	NS*32	char[32]*NS
DATA RECORD		256*(NS+1)		
nr samples from channel [1] of TYPE[1]				Type[1]
nr samples from channel [2] of TYPE[2]				Type[2]
nr samples from channel [3] of TYPE[3]				Type[3]
nr samples from channel [NS] of TYPE[NS]				Type[NS]
EVENT TABLE	NEW	ETP		
mode of event table can be 1 or 3.		ETP + 0	1	uint8
Samplerate associated with Event positions.		ETP + 1	3	uint8[3]
Number of events N		ETP + 4	4	uint32
Position [in samples]		ETP + 8	N*4	uint32
Type		ETP + 8+N*4	N*2	uint16
Channel [optional]	Only if mode is 3	ETP + 8+N*6	N*2	uint16
Duration [in samples, optional]	Only if mode is 3	ETP + 8+N*8	N*4	uint32

Ilustración 43: Tabla de organización de un archivo GDF

Anexo 2: Diagrama de clases completo

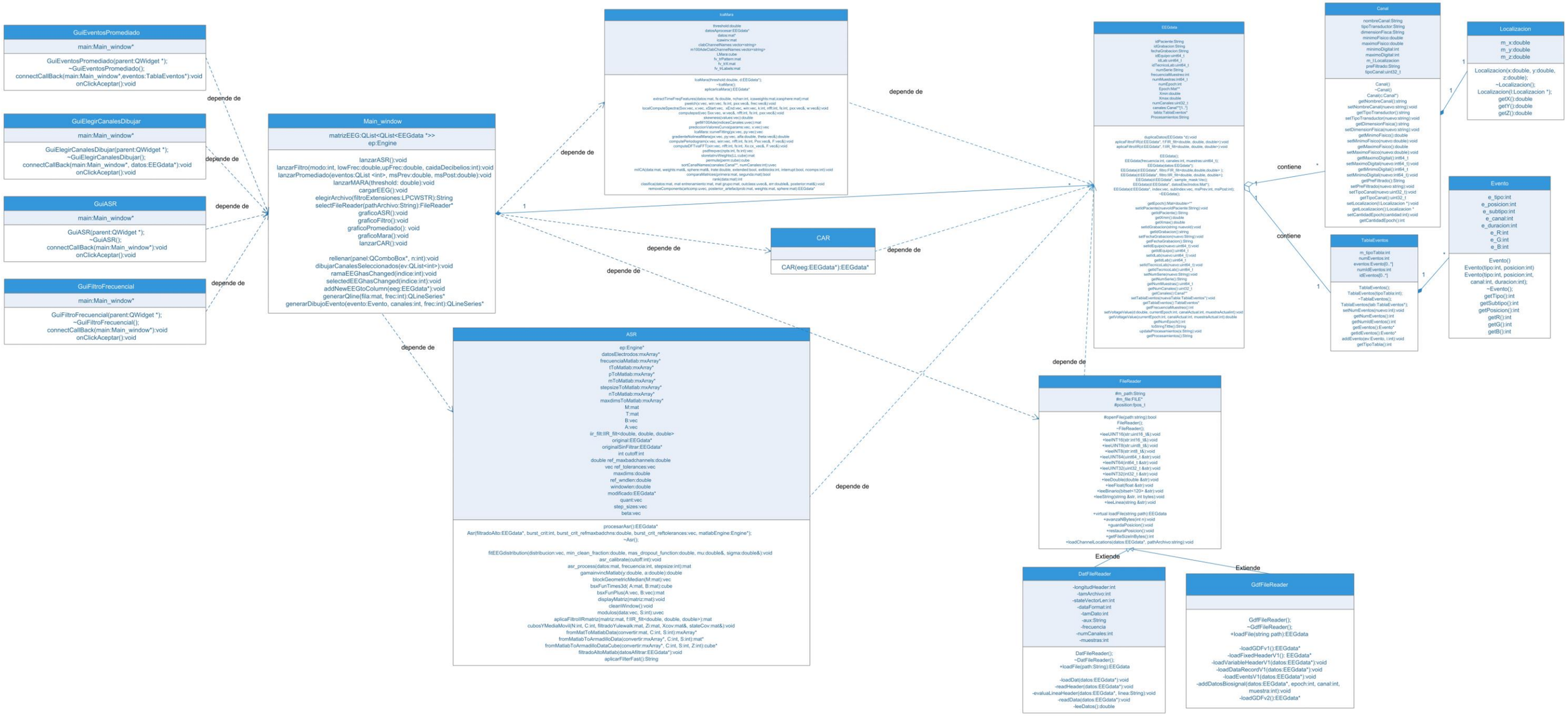

Ilustración 44: Diagrama de clases completo de la aplicación

Anexo 3: Diagramas de secuencia

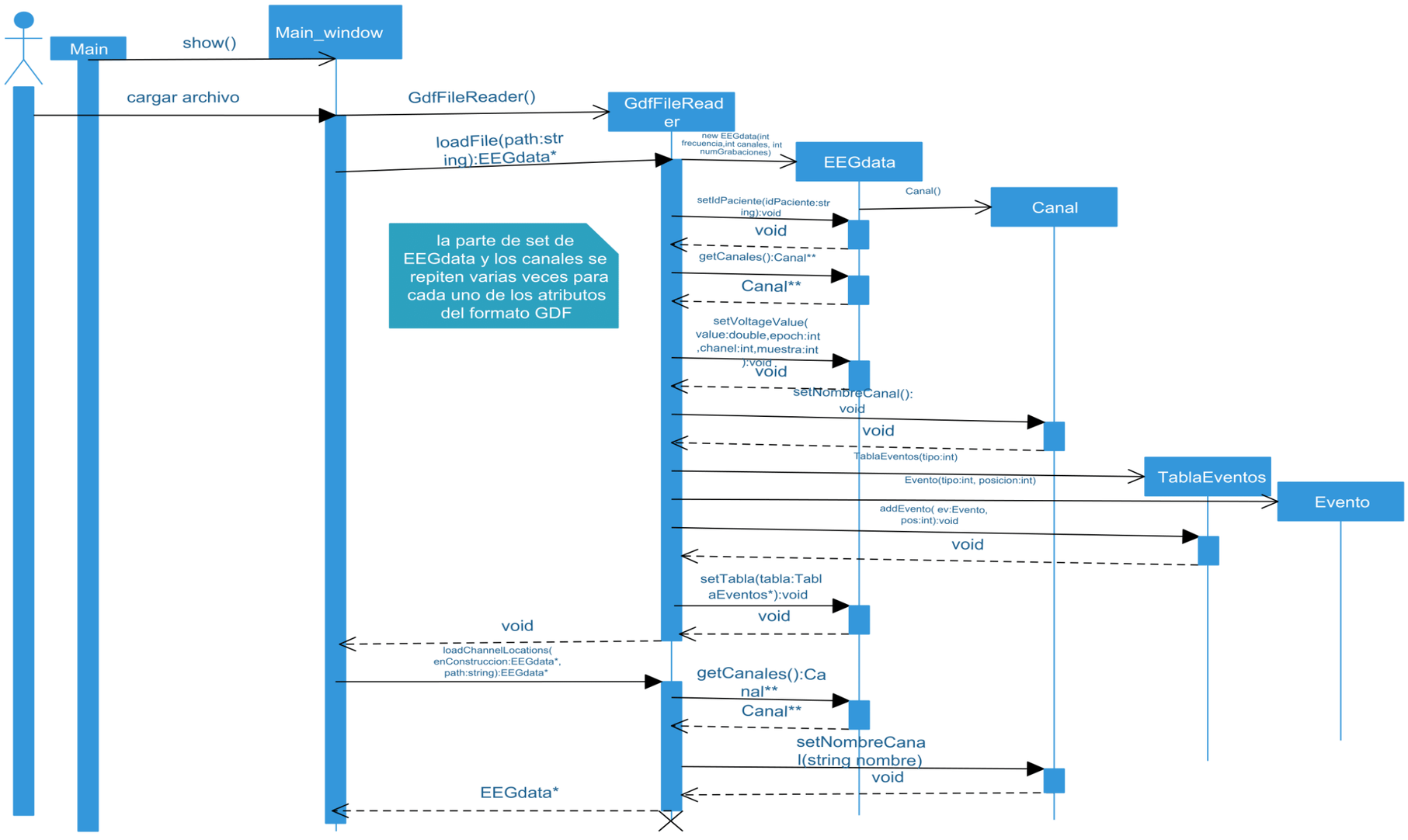

Ilustración 45: Diagrama de secuencia para la inicialización del programa y carga de un archivo GDF

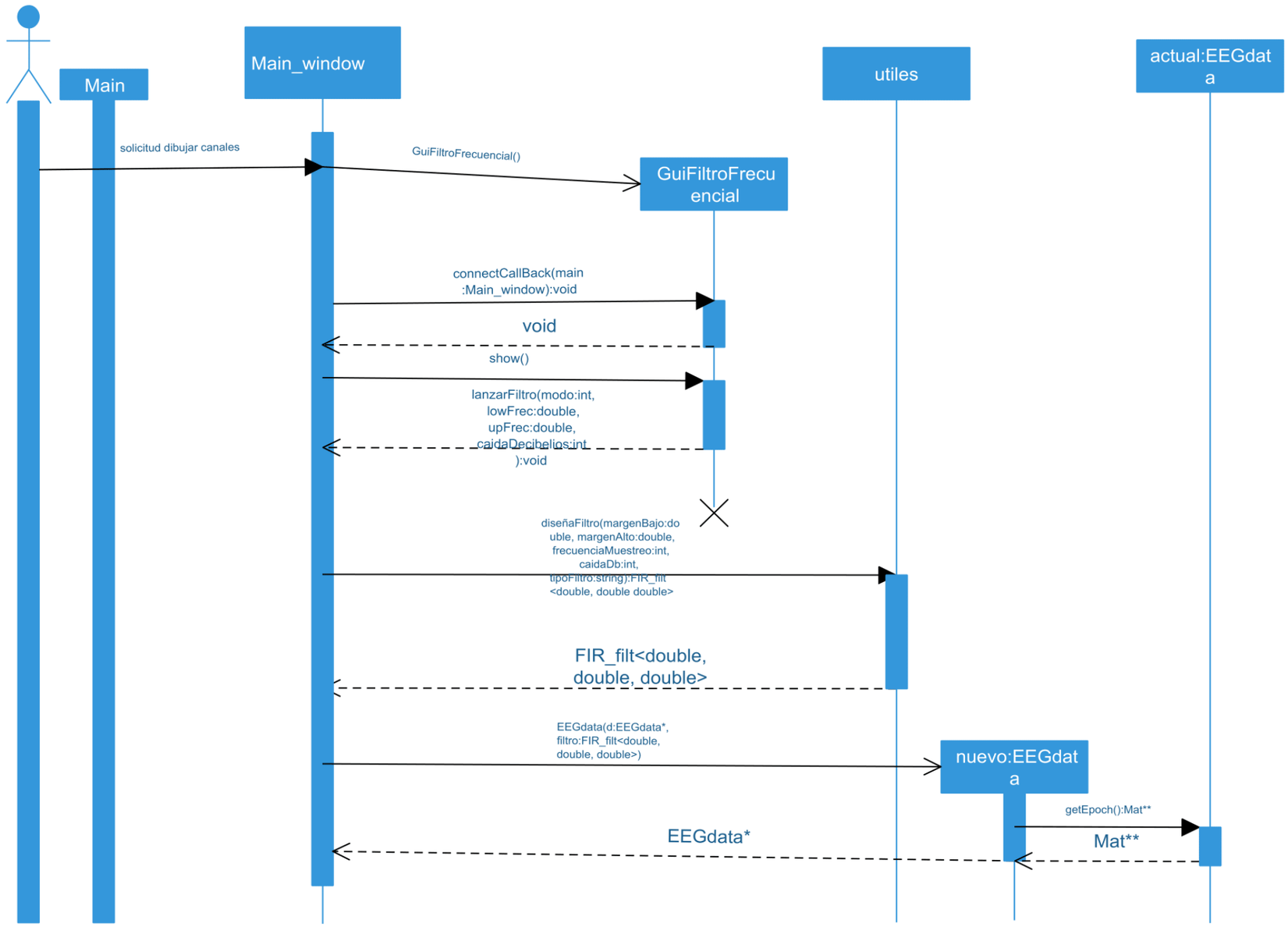

Ilustración 46: Diagrama de secuencia para la aplicación de un filtro

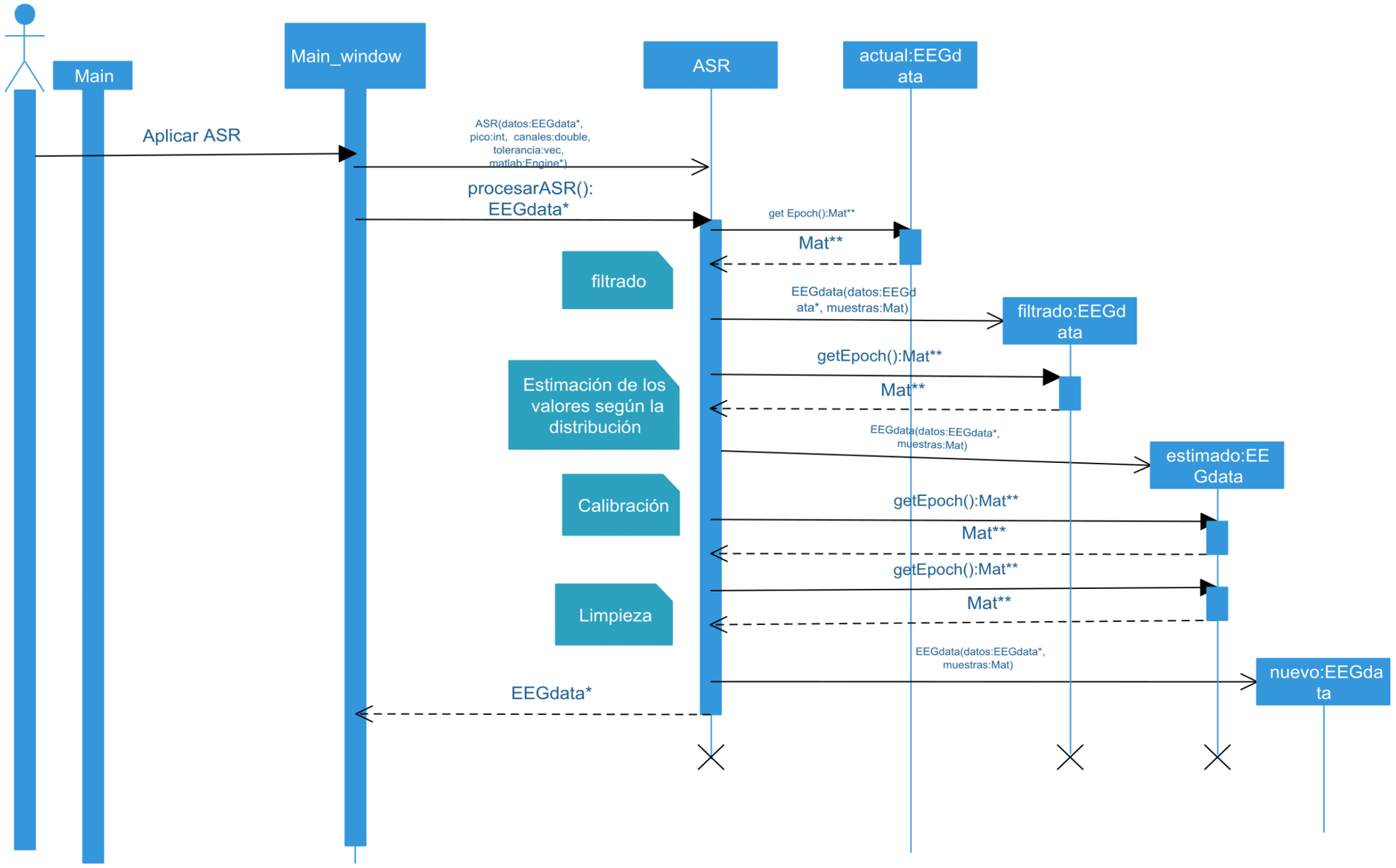

Ilustración 47: Diagrama de secuencia para la aplicación del algoritmo ASR

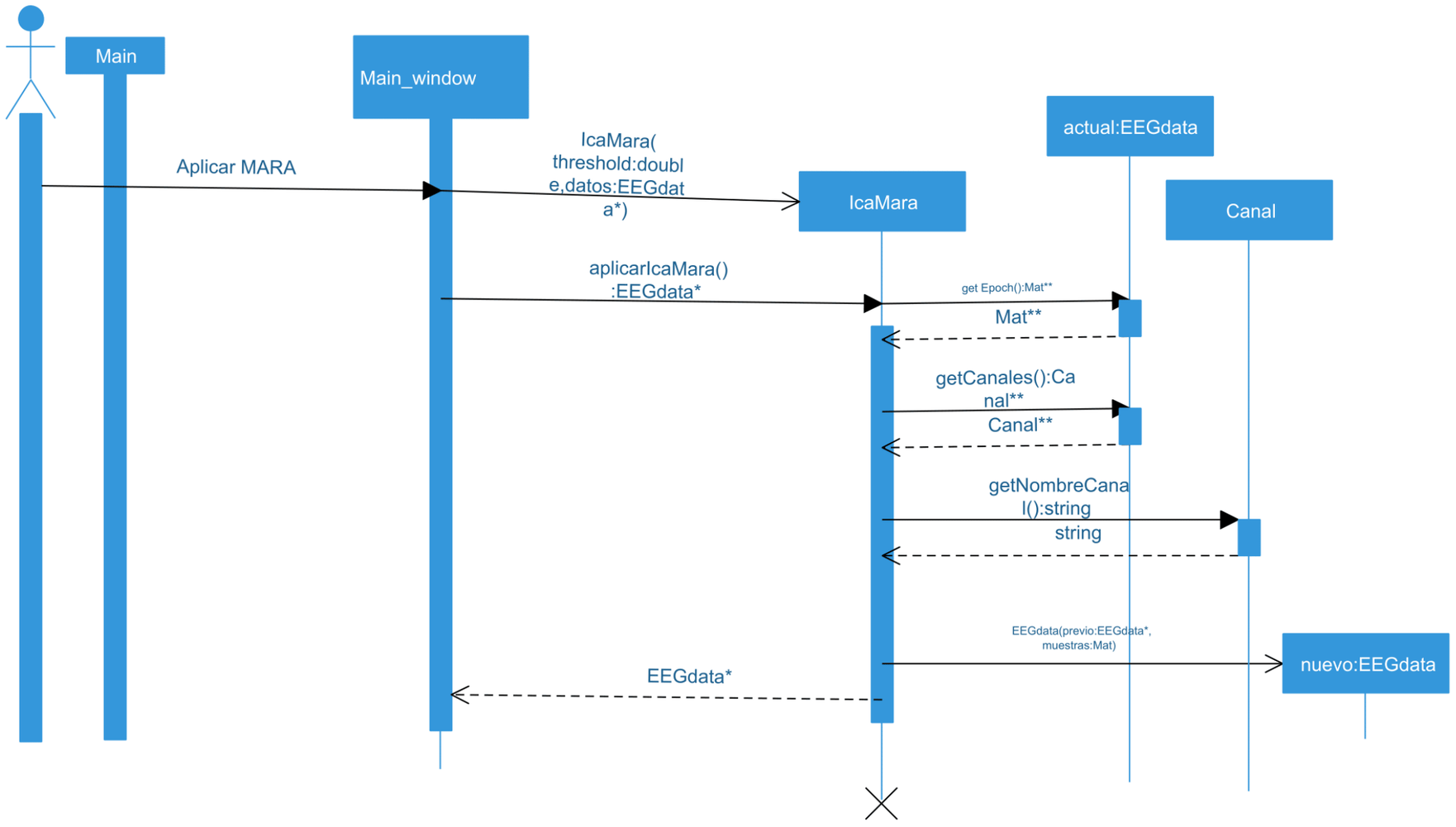

Ilustración 48: Diagrama de secuencia para la aplicación del algoritmo MARA

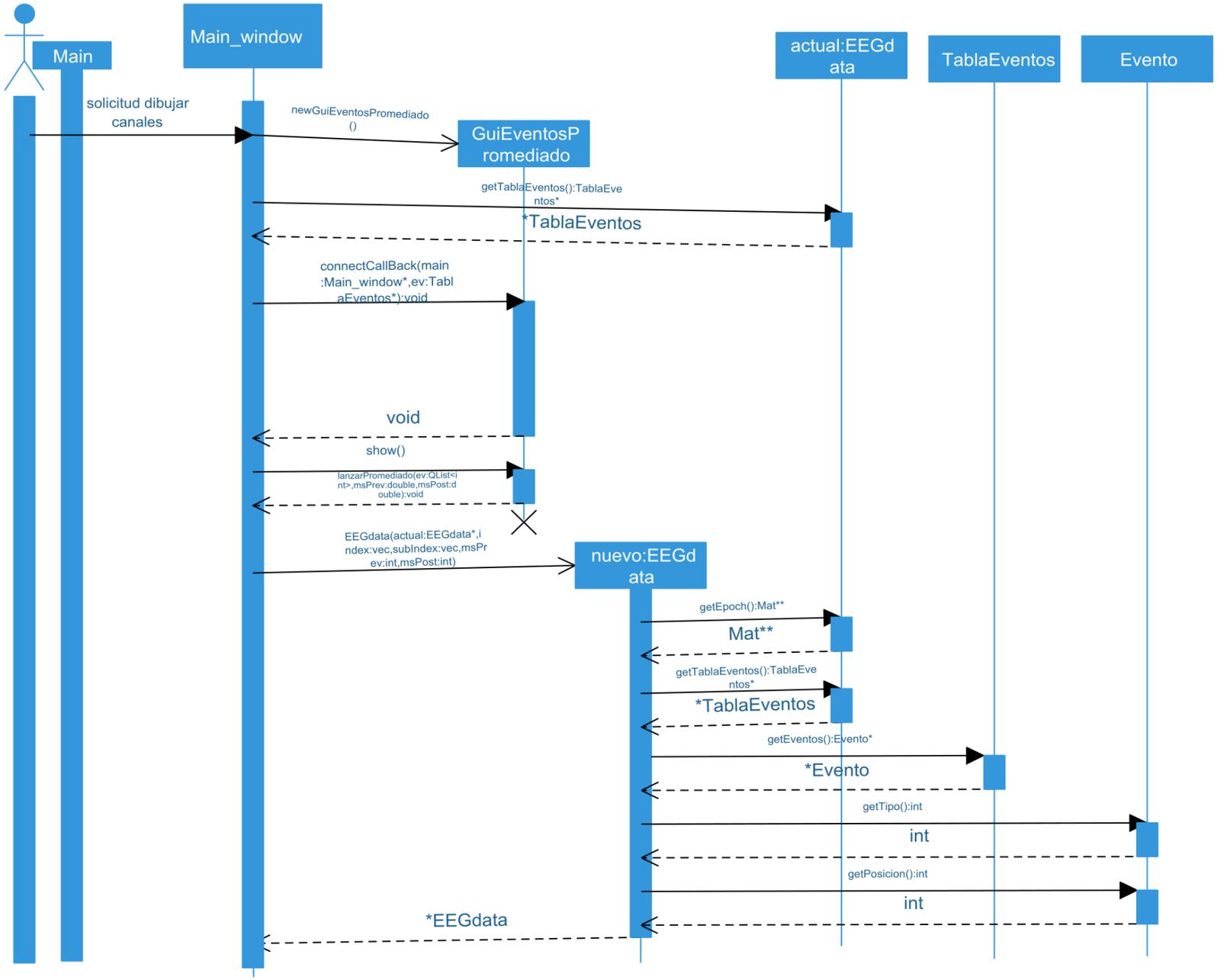

Ilustración 49: Diagrama de secuencia para realizar el promediado

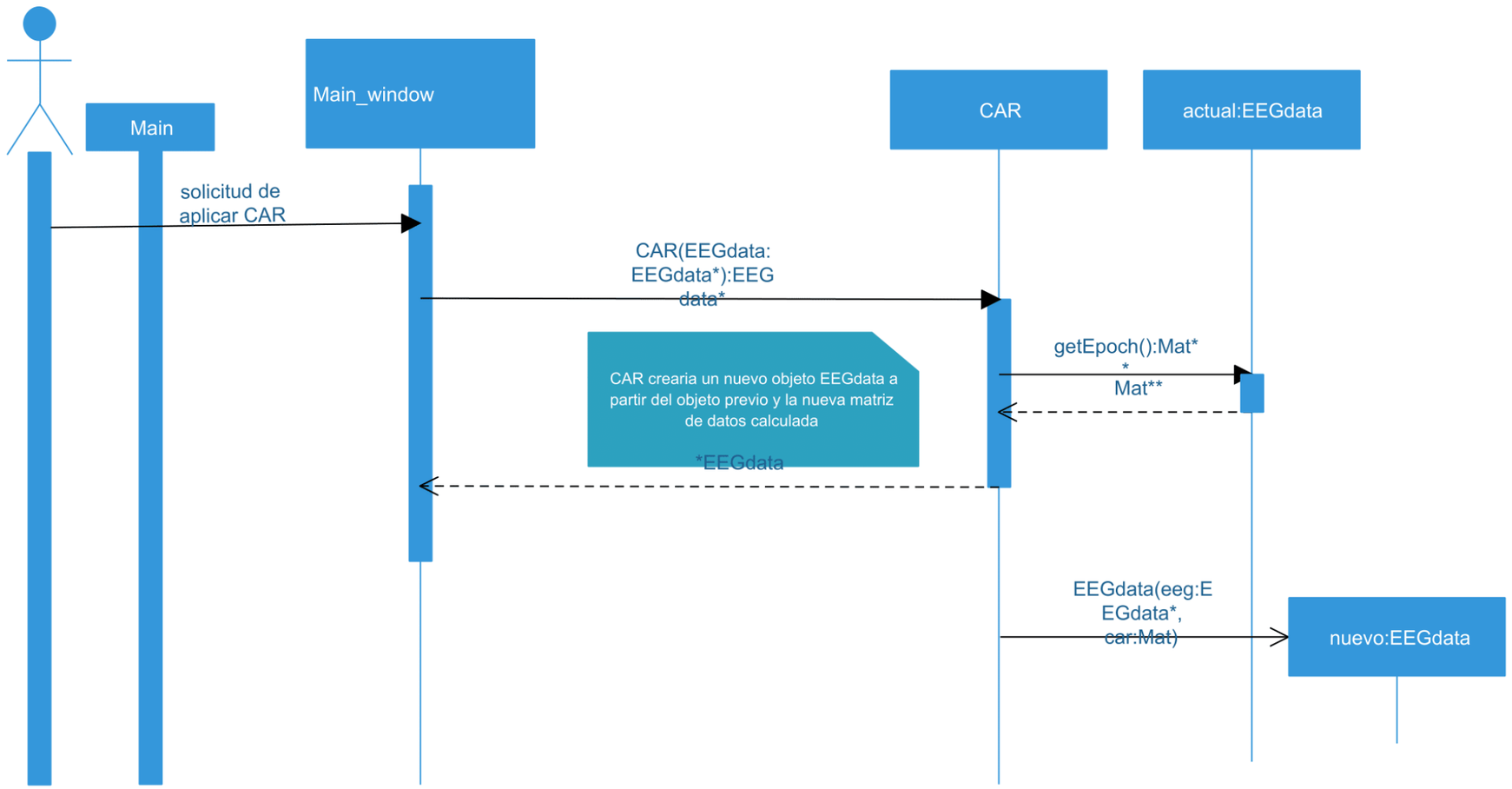

Ilustración 50: Diagrama de secuencia para la aplicación del método CAR

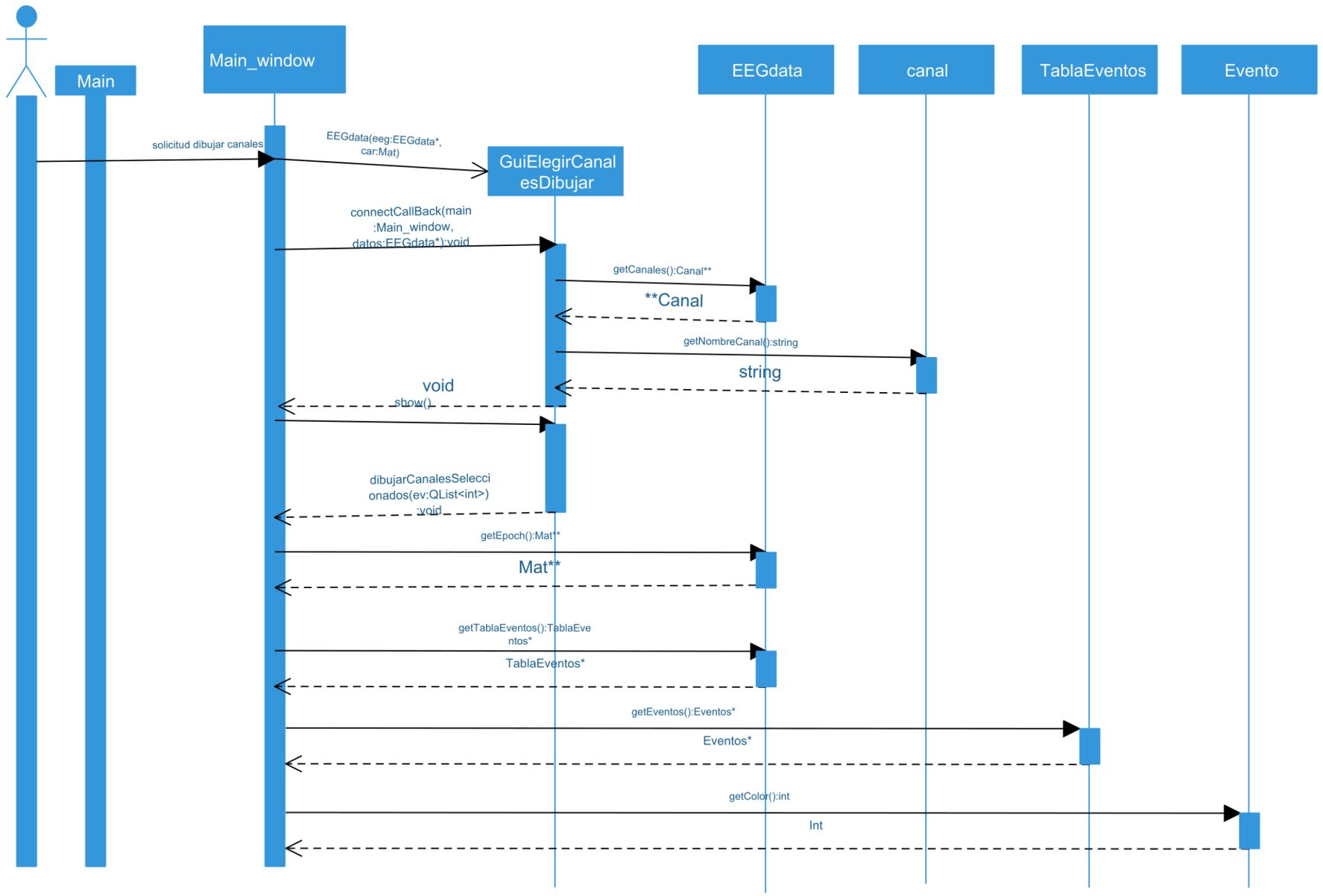

Ilustración 51: Diagrama de secuencia para la visualización de los canales y sus eventos

10 Bibliografía

[1] X. Mariño, Neurociencia para Julia, Pamplona: Editorial Laetoli, 2012.

[2] W. Van Drongelen, Signal Processing for Neuroscientist, Amsterdam: Elsevier, 2007.

[3] H. Shibasaki y M. Hallett, «What is the Bereitschaftspotential?,» *Clinical Neurophysiology,* vol. 117, nº 1, pp. 2341-2356, 2006.

[4] R. Thibault, M. Lifshitz y A. Raz, «The self-regulating brain and neurofeedback: Experimental science and clinical promise,» *Cortex,* vol. 74, nº 1, pp. 247-261, 2016.

[5] G. Pfurtscheller y F. Lopes Da Silva, «Event-related EEG/MEG synchronization and desynchronization: basic principles,» *Clinical Neurophysiology,* vol. 110, nº 11, pp. 1842-1857, 1999.

[6] G. Pfurtscheller, «Event-related synchronization (ERS): an electrophysiological correlate of cortical areas at rest,» *Electroencephalography and Clinical Neurophysiology,* vol. 83, nº 1, pp. 62-69, 1992.

[7] «selectbs.com,» [En línea]. Available: http://www.selectbs.com/process-maturity/what-is-extreme-programming. [Último acceso: 12 Julio 2017].

[8] S. Raman, «infoq.com,» 2 Abril 2014. [En línea]. [Último acceso: 11 Julio 2017].

[9] Swart Center for computational neuroscience, «Swartz Center for computational neuroscience,» [En línea]. Available: https://sccn.ucsd.edu/eeglab/index.php. [Último acceso: 13 Junio 2017].

[10] Nuget, «Nuget,» [En línea]. Available: https://www.nuget.org/packages/python/. [Último acceso: 4 Septiembre 2017].

[11] Mathworks, «Introducing Matlab api for C++,» [En línea]. Available: https://es.mathworks.com/help/matlab/matlab_external/introducing-matlab-

engine.html. [Último acceso: 29 Agosto 2017].

[12] «Sourceforge.net,» [En línea]. Available: http://arma.sourceforge.net/download.html. [Último acceso: 3 Julio 2017].

[13] SigPack, «sourceforge,» [En línea]. Available: http://sigpack.sourceforge.net/. [Último acceso: 8 Septiembre 2017].

[14] R. Curtin, «Mlpack,» [En línea]. Available: http://mlpack.org/about.html. [Último acceso: 19 Julio 2017].

[15] G. Pfurtschteller, «GDF - A GENERAL DATAFORMAT FOR BIOSIGNALS,» [En línea]. Available: http://pub.ist.ac.at/~schloegl/matlab/eeg/gdf4/TR_GDF.pdf. [Último acceso: 21 Julio 2017].

[16] C. Kothe, «SCCN - The Artifact Subspace Reconstruction Method,» Enero 2013. [En línea]. Available: http://sccn.ucsd.edu/eeglab/plugins/ASR.pdf. [Último acceso: 27 Julio 2017].

[17] T. Mullen y C. Kothe, «Youtube - Artifact Removal, Developed by NCTU BRC and UCSD,» 14 Enero 2014. [En línea]. Available: https://www.youtube.com/watch?v=wESFVDKHRBE. [Último acceso: 27 Julio 2017].

[18] C. Kothe, «DTU - Artifact subspace reconstruction,» 9 Septiembre 2015. [En línea]. Available: http://neuro.imm.dtu.dk/wiki/Artifact_subspace_reconstruction. [Último acceso: 6 Septiembre 2017].

[19] A. Beck y S. Sabach, «Israel Institute of Technology,» [En línea]. Available: https://web.iem.technion.ac.il/images/user-files/becka/papers/44.pdf.

[20] A. Hyvärinen, «Independent Component Analysis: Algorithms and Applications,» [En línea]. Available: https://www.cs.helsinki.fi/u/ahyvarin/papers/NN00new.pdf. [Último acceso: 13 Octubre 2017].

[21] A. Delorme, «ICA for dummies,» [En línea]. Available: http://arnauddelorme.com/ica_for_dummies/. [Último acceso: 11 Octubre 2017].

[22] «Wikipedia - Independent component analysis,» [En línea]. Available: https://en.wikipedia.org/wiki/Independent_component_analysis. [Último acceso: 11 Octubre 2017].

[23] «http://cognitrn.psych.indiana.edu - Runica,» [En línea]. Available: http://cognitrn.psych.indiana.edu/busey/temp/eeglabtutorial4.301/allfunctions/runica.html. [Último acceso: 14 Octubre 2017].

[24] «ICA decomposition EEGLAB tutorial,» [En línea]. Available: http://cognitrn.psych.indiana.edu/busey/temp/eeglabtutorial4.301/maintut/ICA_decomposition.html. [Último acceso: 15 Octubre 2017].

[25] I. Wincler, S. Haufe y M. Tangermann, «Automatic Classification of Artifactual ICA-Components for Artifact Removal in EEG Signals,» *behavioral and brain functions,* pp. 1-15, 2011.

[26] I. Downdig, «GitHub,» [En línea]. Available: https://irenne.github.io/artifacts/. [Último acceso: 8 Agosto 2017].

[27] I. Dowding, «GitHub,» [En línea]. Available: https://github.com/irenne/MARA. [Último acceso: 8 Agosto 2017].

[28] P. Marqui, «Standardized low resolution brain electromagnetic tomography (sLORETA): technical details,» *Methods & Findings in Experimental & Clinical Pharmacology,* pp. 1-16, 2002.

[29] F. Esposito, «EEG and MEG Inverse Modeling,» [En línea]. Available: http://www.brainvoyager.com/bvqx/doc/UsersGuide/EMEGSuite/EEGAndMEGInverseModeling.html.

[30] P. D. Welch, «The use of Fast Fourier Transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms,» *IEEE Transactions on Audio and Electroacoustics,* vol. 15, nº 2, pp. 70-73, 1967.

[31] Wikipedia, «Welch's method,» 11 Agosto 2017. [En línea]. Available: https://en.wikipedia.org/wiki/Welch%27s_method. [Último acceso: 17 Octubre 2017].

[32] J. M. Marin, «uc3m.es,» [En línea]. Available: http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema1dm.pdf. [Último acceso: 28 Noviembre 2017].

[33] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Linear_discriminant_analysis. [Último acceso: 28 Noviembre 2017].

[34] K. Ludwig, R. Miriani y N. Langhals, «Using a Common Average Reference to Improve Cortical Neuron Recordings From Microelectrode Arrays,» *Journal of Neurophysiology,* vol. 101, pp. 1679-1789, 2009.

[35] QT, «Qt for Windows - Deployment,» [En línea]. Available: https://doc.qt.io/qt-5/windows-deployment.html. [Último acceso: 12 Enero 2018].