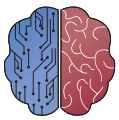
UNIVERSIDAD DE BURGOS

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de la Salud



INGENIERÍA DE LA SALUD

TFG del Grado en Ingeniería de la Salud

Preprocesado y procesado de datos crudos de Electroencefalograma (EEG) Documentación Técnica

Presentado por Raúl Ortega Renuncio

en Universidad de Burgos

11 de febrero de 2026

Tutores: Dra. María Consuelo Saiz Manzanares – Dr. Raúl Marticorena Sánchez

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

El Plan de Proyecto Software establece la estrategia general para el desarrollo, implementación y gestión del proyecto a lo largo de su ciclo de vida. Este documento define los objetivos, el alcance del trabajo, la distribución temporal de las actividades, los recursos necesarios, tanto económicos como tecnológicos, y analiza la viabilidad legal de la implementación. El objetivo principal es proporcionar una hoja de ruta clara que garantice la ejecución eficiente del proyecto y la consecución de los hitos establecidos dentro de los plazos y presupuestos previstos.

A.2. Planificación temporal

La planificación temporal del Trabajo Final de Grado se ha estructurado en tres fases principales que abarcan desde la identificación de datos hasta la documentación final. Esta organización permite un seguimiento ordenado de las actividades y garantiza el cumplimiento de los objetivos previstos.

Las fases se organizan de la siguiente manera:

1. Identificación e investigación de datos: caracterización de archivos del electroencefalograma (EEG) y estructura de datos, investigación bibliográfica exhaustiva, análisis exploratorio de datos disponibles y definición de parámetros de análisis para desarrollo del trabajo.
2. Desarrollo e implementación: preprocesamiento de datos EEG, procesamiento y análisis, desarrollo de *Jupyter Notebooks* para reproducibili-

dad, mejoras iterativas del código y desarrollo de aplicación interactiva con *Streamlit*.

3. Documentación y finalización: redacción de la memoria y anexos, mejoras finales de documentación, creación de presentación para defensa y empaquetamiento del Trabajo Fin de Grado para entrega.

Planificación de *sprints* y tareas por fase

El Trabajo de Fin de Grado ha sido subdividido en *sprints* de una duración variable según las necesidades de cada fase. A continuación se detalla la planificación cronológica de cada *sprint* con las tareas realizadas.

Sprint 1 - Identificación e investigación de datos (13 de enero - 25 de febrero de 2025)

Corresponde a la Fase 1, durante este *sprint* se llevó a cabo la caracterización completa de los archivos de datos adquiridos del EEG. Se identificó la estructura interna de cada archivo, las variables disponibles, el formato de almacenamiento de datos y los metadatos asociados. Simultáneamente, se realizó una investigación bibliográfica exhaustiva para fundamentar el trabajo teóricamente.

Sprint 2 - Desarrollo de un *Jupyter Notebook* (12 de marzo - 25 de abril de 2025)

Corresponde a la Fase 2, en este *sprint* se desarrolló un *Jupyter Notebook* que constituyó la base del análisis computacional del proyecto. El *notebook* fue estructurado de manera modular, permitiendo la ejecución paso a paso del flujo completo de procesamiento de datos. Se documentó cada sección del *notebook* con explicaciones detalladas y comentarios de código, garantizando la reproducibilidad y la comprensión completa del análisis realizado.

Sprint 3 - Mejoras en el código (23 de mayo - 12 de junio de 2025)

Corresponde a la Fase 2, durante este *sprint* se realizaron mejoras sustanciales en la calidad, eficiencia y robustez del código desarrollado. Se refactorizó el código para mejorar su legibilidad y rendimiento computacional, y se implementaron algoritmos optimizados de preprocesamiento de señal de EEG. Además, se incorporó la automatización de la ejecución para el conjunto de participantes y el guardado de los resultados obtenidos. Este

sprint fue cerrado y reabierto más adelante para incluir más mejoras que se fueron identificando.

***Sprint* 4 - Memoria (12 de junio - 9 de septiembre de 2025)**

Corresponde a la Fase 3, durante este *sprint* se inició la redacción de la memoria. Este *sprint* fue más prolongado de lo inicialmente estimado debido a pausas por circunstancias personales. Sin embargo, permitió una reflexión más profunda sobre los resultados y una redacción más cuidadosa de los contenidos técnicos y académicos.

***Sprint* 5 - APP *Streamlit* (18 de diciembre 2025 - 3 de enero de 2026)**

Corresponde a la Fase 2, en este *sprint* se desarrolló una aplicación web interactiva utilizando *Streamlit*, permitiendo la visualización y exploración de los datos EEG de manera más intuitiva y accesible.

***Sprint* 6 - Anexos (3 de enero - 11 de enero de 2026)**

Corresponde a la Fase 3, durante este *sprint* se realizó la redacción de los anexos técnicos necesarios para acompañar la memoria.

***Sprint* 7 - Mejora memoria (9 de enero - 11 de enero de 2026)**

Corresponde a la Fase 3, en este *sprint* se realizaron las últimas mejoras sobre la documentación de la memoria.

***Sprint* 8 - Mejoras (12 de enero - 2 de febrero de 2026)**

Corresponde a la Fase 3, durante este *sprint* se realizaron mejoras finales tanto en los componentes técnicos como en la documentación de todo el proyecto. Se verificó de forma exhaustiva la validación de toda la documentación, asegurando que fuera completa, precisa y coherente en todos sus aspectos. Se realizaron las correcciones en los anexos y se ajustaron pequeños detalles técnicos que pudieron haber sido detectados durante las revisiones, corrigiendo errores tipográficos o inconsistencias menores.

Sprint 9 - Presentación y entrega (3 de febrero - 10 de febrero de 2026)

Correspondiente a la Fase 3, en este *sprint* se preparó la presentación del Trabajo de Fin de Grado para su defensa ante el tribunal académico. Se creó una presentación visual profesional que comunica los aspectos clave del trabajo realizado, incluyendo objetivos claros, metodología implementada, resultados obtenidos y conclusiones derivadas. Se realizaron ensayos de presentación oral para mejorar la comunicación efectiva y gestionar correctamente el tiempo disponible. Y se prepararon todos los materiales necesarios para la defensa.

Seguimiento del proyecto mediante reuniones

Además, para garantizar un seguimiento eficiente del proyecto durante su ejecución, se han realizado reuniones de progreso vía *Microsoft Teams* con los tutores del proyecto con el objetivo de revisar las tareas completadas, evaluar el cumplimiento de hitos y planificar los siguientes pasos a seguir. Durante la fase de documentación se sustituyeron algunas de estas reuniones por comunicación vía correo electrónico.

Durante la Fase 1 (Identificación e investigación de datos), se realizaron tres reuniones iniciales. La primera reunión el 13 de enero de 2025 marcó la presentación de la propuesta del Trabajo de Fin de Grado, donde se compartieron carpetas con información sobre *Bitbrain* y grabaciones formativas sobre el proceso de obtención de datos. La segunda reunión el 12 de febrero de 2025, permitió revisar los datos compartidos y nueva información aportada por *Bitbrain*, despues de una pausa debida a circunstancias personales. En la tercera reunión el 25 de febrero de 2025 se concretaron los principales objetivos del trabajo tras la revisión de los contenidos de los archivos e investigación sobre los datos.

Durante la Fase 2 (Desarrollo e implementación), se llevaron a cabo 7 reuniones de seguimiento distribuidas entre marzo y junio de 2025. Estas reuniones permitieron revisar el progreso en el desarrollo del *Jupyter Notebook* y validar las mejoras implementadas en el código. Se realizaron reuniones el 12 y 26 de marzo, 25 de abril, 7, 23 y 28 de mayo y 12 de junio de 2025. En esta última reunión se comentaron las modificaciones realizadas, los últimos detalles a mejorar del código y se definió inicialmente la estructura a seguir para la redacción de la memoria.

Durante la Fase 3 (Documentación y finalización), se realizaron 4 reuniones finales de seguimiento. Se realizó una reunión el 4 de julio de 2025 para

validar los últimos aspectos del desarrollo, debido al periodo vacacional. Se reanudaron las reuniones el 9 y 23 de septiembre de 2025 para revisar el progreso de la documentación. Posteriormente, entre octubre y diciembre no se avanzó en la documentación, motivado por situaciones sobrevenidas del alumno. En diciembre se reanudo el trabajo, pero en este caso la comunicación fue mayormente vía correo electrónico, hasta el 21 de enero de 2026. En esta se comentaron dudas sobre la estructura y enfoque más adecuado de determinados apartados de los anexos técnicos, garantizando que la documentación complementaria cumpliera con los estándares académicos requeridos antes de la defensa del proyecto.

Seguimiento del proyecto mediante herramientas de gestión

Adicionalmente, el progreso del proyecto ha sido monitoreado mediante *GitHub* y *Zube*. La siguiente gráfica muestra el *throughput* mensual (número de tareas completadas) a lo largo del proyecto, permitiendo una visualización clara de las tendencias generales de productividad:

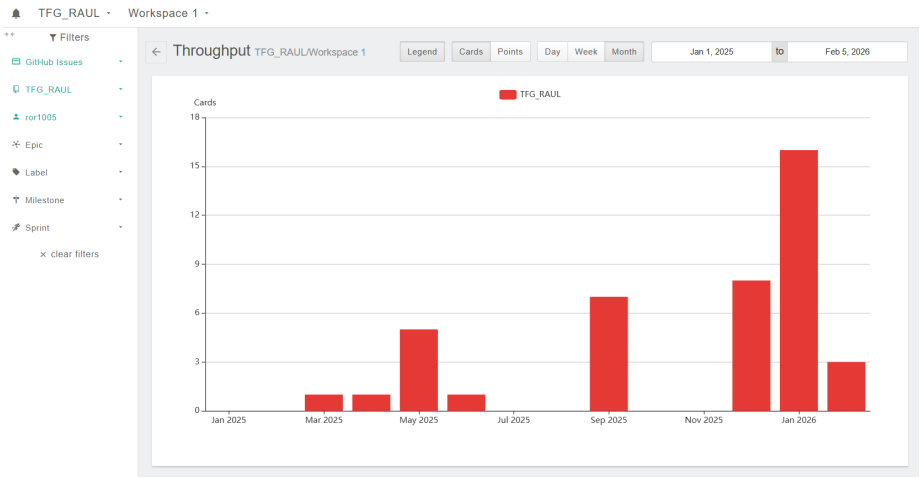


Figura A.1: *Throughput* mensual del proyecto (enero 2025 - febrero 2026). Fuente propia

Se observa que los meses de mayor productividad fueron mayo (5 tareas), septiembre (7 tareas), diciembre (8 tareas) y enero (16 tareas).

Los períodos de menor actividad corresponden a los meses donde se realizaron pausas por circunstancias personales (agosto, octubre-noviembre),

Concepto	Coste
Sueldo bruto anual	25.000 €
Sueldo bruto mensual	2.083,33 €
Sueldo bruto por hora	13,02 €
Total para 375 horas	4.882,50 €

Tabla A.1: Costes personales de la contratación de un ingeniero. Fuente propia

lo que causó interrupciones en el flujo de trabajo pero no afectó significativamente a los objetivos finales del proyecto.

Estas métricas reflejan fielmente el patrón temporal documentado en las reuniones de seguimiento y en la planificación de sprints, validando que la ejecución del proyecto se alineó con los objetivos establecidos en cada fase.

A.3. Estudio de viabilidad

Planificación económica

La planificación económica del proyecto incluye la adquisición de equipamiento especializado, software profesional, costes de personal y gastos de logística. En las siguientes tablas se detalla el desglose de inversiones requeridas para la implementación completa del proyecto.

Costes de personal

Los costes de personal se calculan de acuerdo con el salario medio de un ingeniero biomédico recién graduado y el número de horas dedicadas indicadas por la guía docente de la asignatura. El sueldo medio de un ingeniero biomédico recién graduado es de alrededor de 25.000 euros brutos al año [de Vitoria, 2026], lo que resulta en un coste bruto mensual de 2.083,33 euros. Considerando 160 horas de trabajo mensual, el coste por hora equivale a 13,02 euros y siendo 375 las horas de trabajo según la guía docente de la asignatura, da como resultado un coste bruto de 4.882,50 euros para costes de personal , como se indica en la tabla A.1.

Costes de equipamiento y *software*

Los costes de la adquisición de equipamiento especializado y software profesional requieren consideración especial mediante amortización. El Trabajo

de Fin de Grado tiene una duración de aproximadamente 6 meses (normalmente, enero-junio), durante los cuales se requiere el equipamiento. Sin embargo, estos equipos poseen una vida útil superior y podrán ser utilizados en proyectos y trabajos posteriores. Por lo tanto, resulta más apropiado presupuestar desde este TFG únicamente la parte proporcional del coste según la duración del proyecto respecto a la vida útil del equipamiento.

Dado que la documentación técnica del fabricante (*Bitbrain*) no especifica explícitamente la vida útil, se ha estimado una vida útil de 5 años (60 meses) como valor orientativo razonable.

El cálculo de la amortización se realiza mediante la siguiente fórmula:

$$\text{Coste imputable} = \text{Coste total del equipamiento} \times \frac{\text{Duración del proyecto}}{\text{Vida útil estimada}}$$

$$\text{Coste imputable} = \text{Coste total} \times \frac{6}{60} = \text{Coste total} \times 0,10$$

Aplicando esta metodología de amortización, los costes de la adquisición de equipamiento especializado, software profesional y gastos de logística se detallan en la tabla A.2.

La inversión total amortizada en equipamiento y software requeridos asciende a 3.222,46 euros, lo que representa el 10 % del coste total del equipamiento, correspondiente a los 6 meses de duración del proyecto dentro de su vida útil estimada de 5 años.

Coste total

Finalmente, la inversión total requerida para la ejecución completa del proyecto se calcula mediante la suma de los costes ajustados para el personal, equipamiento amortizado y software, como se muestra en la tabla A.3. Como se muestra en la tabla A.3, la inversión total requerida asciende a 8.104,96 euros. Este coste representa únicamente la parte proporcional del equipamiento y software que se imputa al proyecto, reconociendo que dichos recursos poseen vida útil posterior al TFG y pueden ser utilizados en investigaciones y trabajos futuros. El coste de personal constituye el principal componente del presupuesto (60,2 %), mientras que el equipamiento amortizado representa el 39,8 % restante.

Este coste total es razonable y viable para un Trabajo de Fin de Grado en el campo de la Ingeniería de la Salud. Esta inversión permite acceder a

Instrumentos	**Descripción**	**Coste total**	**Coste inputable (10 %)**
Equipo Portátil Falkon i7	Pantalla 15.6"– Procesador Intel core I7-9750H – Disco duro 512Gb SSD – Memoria RAM 16Gb -Geforce RTX 2070 MAX-Q GDDR6 8GB – Windows 10 Professional - Conexiones: HDMI / USB-C / USB 3.1 / RJ45 x 1 / HDMI x 1 / Jack 3.5 mm / USB-A 3.1 x 4 / Thunderbolt x 1 – Bluetooth / WiFi / Gigabit Ethernet	1.064,60 €	106,46 €
Professional Lab	Laboratorio para el análisis avanzado de emociones y cognición. Este laboratorio incluye los dispositivos *Ring* y *Diadem*, los programas *SennsLab* y *SennsMetrics*, con soporte técnico básico, acceso a análisis en la nube durante el primer año y curso de formación en línea para los clientes	17.085,00 €	1.708,50 €
Suscripción *SennsLab* y *SennsMetrics* (1 año)	Acceso a soporte técnico básico, actualizaciones de software y a una plataforma de análisis de datos en la nube durante un período de 1 año	2.800 €	1.400 €
Envío y manipulación	Costes de preparación y envío	75,00 €	7,50 €
Total instrumentos		**21.024,60 €**	**3.222,46 €**

Tabla A.2: Desglose de inversiones en equipamiento y software con amortización. Fuente propia

Concepto	**Coste**
Coste personal	4.882,50 €
Coste de equipamiento y *software*	3.222,46 €
Total	**8.104,96** €

Tabla A.3: Costes personales de la contratación de un ingeniero. Fuente propia

equipamiento especializado de calidad profesional que de otro modo sería inaccesible para estudiantes de grado. La amortización del equipamiento sobre su vida útil reconoce que esta es una inversión a largo plazo de la institución, no un coste exclusivo de este proyecto.

Viabilidad legal

El desarrollo del estudio se enmarca en el proyecto de I+D+i *"Análisis multidimensional de la carga cognitiva en estudiantes universitarios: un estudio multicanal integrado (MACLUS-IMS)"*, dirigido por la Dra. María Consuelo Sáiz Manzanares. Dicho proyecto cuenta con informe positivo de la Comisión de Bioética de la Universidad de Burgos (UBU N.º IO 5/2024, fecha de expedición 29/01/2025). Además, todos los participantes en el estudio han firmado el consentimiento informado, aceptando su participación en la recolección de datos bajo los estándares de protección de datos personales.

Licencias de *software* utilizado

El proyecto utiliza exclusivamente *software* de código abierto con licencias que garantizan transparencia, reproducibilidad y compatibilidad. Las bibliotecas empleadas y sus respectivas licencias son las siguientes:

- *matplotlib*: Licencia del equipo de desarrollo *Matplotlib* (*Matplotlib Development Team License*, PSF) [Team, 2016].
- *mne*: Licencia BSD de 3 Cláusulas "Nueva" o "Revisada" (*BSD 3-Clause "New" or "Revised" License*) [authors, 2025].
- *mne-icalabel*: Licencia BSD de 3 Cláusulas "Nueva" o "Revisada" (*BSD 3-Clause "New" or "Revised" License*) [MNE, 2022].
- *numpy*: Licencia BSD de 3 Cláusulas "Nueva" o "Revisada" (*BSD 3-Clause "New" or "Revised" License*) [Developers, 2024].

- *pandas*: Licencia BSD de 3 Cláusulas "Nueva" o "Revisada" (*BSD 3-Clause "New" or "Revised" License*) [Team, 2026].
- *scipy*: Licencia BSD de 3 Cláusulas "Nueva" o "Revisada" (*BSD 3-Clause "New" or "Revised" License*) [Developers, 2025d].
- *onnxruntime*: Licencia MIT (*MIT License*) [Corporation, 2021].
- *Streamlit*: Licencia Apache 2.0 (*Apache License 2.0*) [Developers, 2019].
- *Python*: Licencia de la Fundación de *Software Python* versión 2 (*Python Software Foundation License version 2*, PSF) [Foundation, 2024].
- *Jupyter Notebook*: Licencia BSD de 3 Cláusulas "Nueva" o "Revisada" (*BSD 3-Clause "New" or "Revised" License*) [Team and Team, 2023].

Las licencias empleadas son compatibles entre sí y se clasifican como licencias permisivas de código abierto, permitiendo el uso libre en contextos académicos y comerciales sin restricciones especiales más allá de la atribución de autoría.

Licencia del código desarrollado

El código desarrollado en este Trabajo de Fin de Grado se libera bajo la Licencia BSD de 3 Cláusulas "Nueva" o "Revisada" (*BSD 3-Clause "New" or "Revised" License*) [Initiative, sf]. Esta licencia permisiva es la más ampliamente utilizada en el ecosistema de investigación [Library, 2023], coincidiendo con la licencia de las librerías principales empleadas. Esta licencia garantiza que el código permanecerá disponible de forma libre para uso académico y comercial, mientras exige únicamente la atribución de autoría, promoviendo así la difusión y reutilización del *software* en investigación.

Apéndice B

Documentación de usuario

B.1. Introducción

Este Apéndice B presenta una guía completa sobre los requisitos y procedimientos para la instalación, configuración y uso del software empleado en este trabajo. Su objetivo es facilitar la replicación del trabajo o su continuación en futuras investigaciones, garantizando un proceso eficiente y sin inconvenientes para otros usuarios.

B.2. Requisitos software y hardware para ejecutar el proyecto.

Requisitos hardware

Para la evaluación y procesamiento de datos es necesario disponer de un ordenador convencional que cumpla con las siguientes especificaciones técnicas mínimas:

- Procesador: CPU de 64 bits con 2 núcleos.
- Memoria RAM: 8 GB.
- Almacenamiento: 10 GB de espacio libre en disco.

Requisitos software

Para la ejecución y replicación del trabajo, el principal requisito software es la instalación de *Python 3.10 o superior*, ya que ha sido el lenguaje de

programación principal utilizado para desarrollar la aplicación web y siendo plenamente compatible con *Windows 10/11*,*macOS 11+* o *Linux Ubuntu 20.04 LTS* equivalente. En concreto, la aplicación ha sido desarrollada y probada con *Python 3.13.3*.

El desarrollo del trabajo se realizó en *Visual Studio Code*, utilizando *Streamlit 1.52.2* como *framework* principal.

B.3. Instalación / Puesta en marcha

A continuación se detallan los pasos secuenciales necesarios para realizar la instalación y configuración correcta del entorno de desarrollo requerido:

1. Instalación de *Python*: acceder al sitio web oficial y proceder a descargar el instalador más reciente de Python 3.10 o superior compatible con su sistema operativo. Ejecutar el instalador y asegurarse de marcar la casilla *"Add Python to PATH"* antes de finalizar la instalación. Para verificar la correcta instalación, abra una terminal o símbolo del sistema y ejecute el comando `python -version` (o `python3 -version` en *Linux* o *macOS*), que debe devolver la versión instalada.

2. Descargar el repositorio del proyecto: este repositorio es privado, por lo que solo usuarios con acceso concedido pueden descargarlo o clonarlo. Para solicitar acceso, es necesario contactar con el propietario del repositorio. Una vez se conceda el acceso, existen dos opciones para obtener los archivos del proyecto.

   - Descargar como ZIP: acceder al repositorio en *Github* y hacer clic en el botón *"Code"* y *"Download ZIP"*. Extraer los archivos descargados en una carpeta accesible y claramente identificada en el ordenador.
   - Utilizar *Git*: esta opción permite clonar el repositorio de forma directa utilizando el sistema de control de versiones de *Git*. Para ello es necesario proceder a la instalación previa de *Git* en el sistema operativo (descarga *Git*), descargando la versión más reciente e instalando las opciones predeterminadas recomendadas. Una vez instalado, abrir una terminal, navegar hasta la carpeta donde se quiere almacenar el proyecto (por ejemplo, mediante el comando `cd C:\Documentos`) y ejecutar el siguiente comando:

     ```
     git clone <URL-del-repositorio>
     ```

donde `<URL-del-repositorio>` corresponde a la dirección URL del repositorio de *Github* disponible en el botón *"Code"* del repositorio. *Git* procederá a descargar automáticamente todos los archivos del repositorio en una carpeta con el nombre del proyecto.

Para proceder a la instalación de las dependencias necesarias y la ejecución de la aplicación, consultar la sección C.3, donde se detallan los procedimientos técnicos requeridos.

B.4. Manuales y/o demostraciones prácticas

La aplicación desarrollada presenta un diseño de interfaz intuitivo y accesible, ofreciendo una interfaz web interactiva que facilita su uso. Asimismo, cuenta con explicaciones desplegables sobre el flujo de análisis y los modos de ejecución disponibles. A continuación se presenta una demostración visual de la interfaz gráfica, detallando cada una de sus secciones y funcionalidades.

Pantalla principal

La aplicación se presenta con una estructura clara y modular, dividida en dos áreas principales (véase la figura B.1):

- La zona central muestra el título de la aplicación con su identificador visual, una descripción breve de su funcionalidad, y dos secciones expandibles: "Flujo de análisis detallado" y "Modos de carga de datos" (véanse las figuras B.2 y B.3, respectivamente).
- La barra lateral izquierda contiene la sección "Configuración de análisis", que permite la selección del modo de análisis a realizar y la visualización de los valores de los parámetros utilizados en el análisis.

Modo participante individual

Carga de datos

En el modo de participante individual, la interfaz proporciona dos campos de carga de archivos. Una vez cargados los archivos (véase la figura B.4), se muestra una confirmación de carga y se activa el botón "Procesar participante", que al pulsarlo inicia el procesamiento.

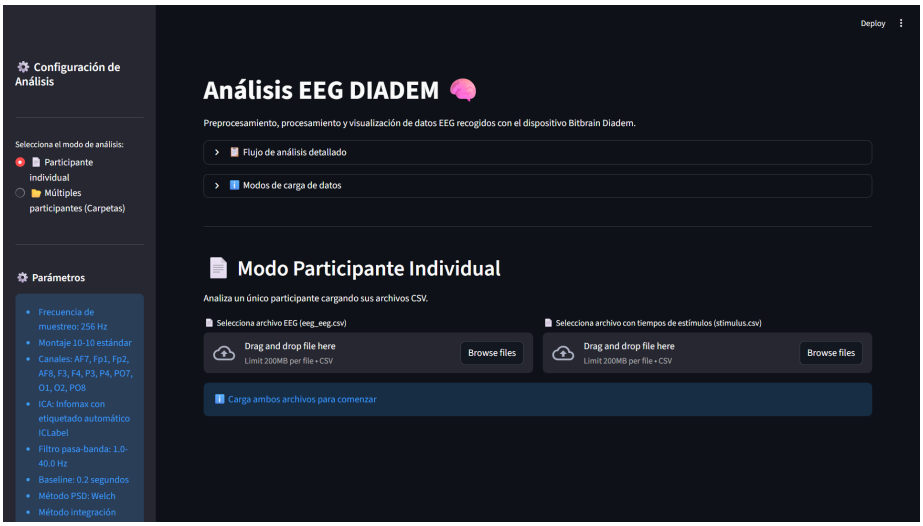


Figura B.1: Pantalla principal de la aplicación "Análisis EEG *DIADEM*". Fuente propia

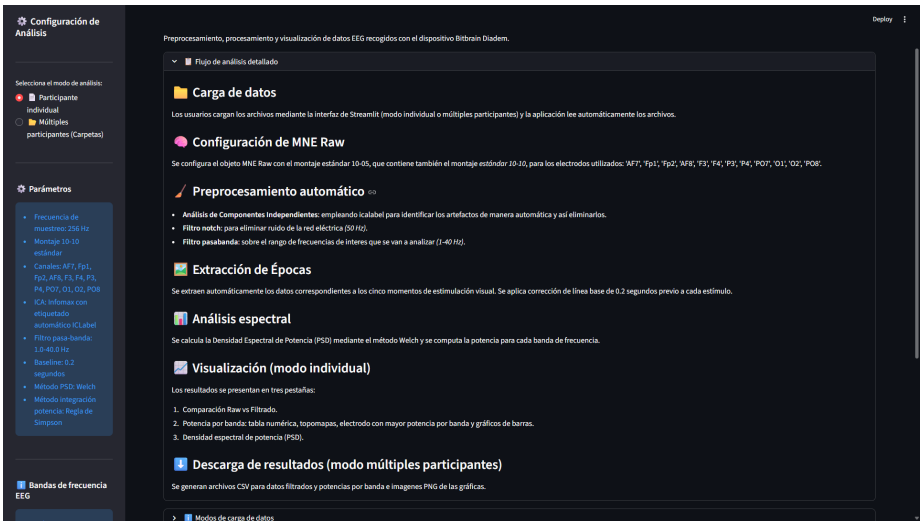


Figura B.2: Sección de flujo de análisis detallado mostrando cada etapa del procesamiento. Fuente propia

Procesamiento

El progreso del procesamiento se visualiza mediante barras de estado que indican cada etapa del flujo de análisis (véase la figura B.5). Una vez completado, se habilita el acceso a las pestañas de resultados.

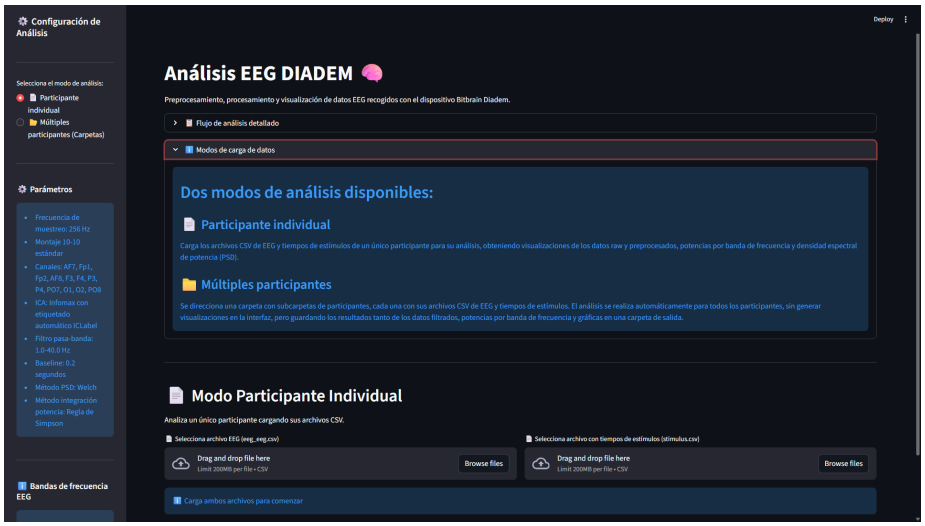


Figura B.3: Sección informativa de los modos de análisis. Fuente propia

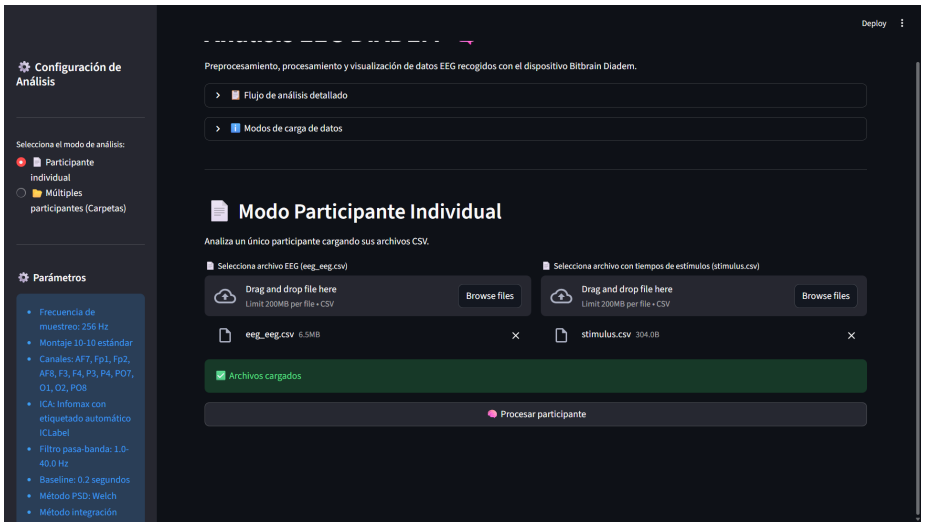


Figura B.4: Confirmación de carga de archivos. Fuente propia

Resultados

Los resultados del análisis se organizan en varias pestañas dentro de la interfaz. En cada pestaña se generan, para cada una de las imágenes correspondientes a los eventos de interés, las representaciones gráficas y numéricas asociadas, de forma que el usuario pueda explorar de manera individualizada la respuesta EEG a cada estímulo.

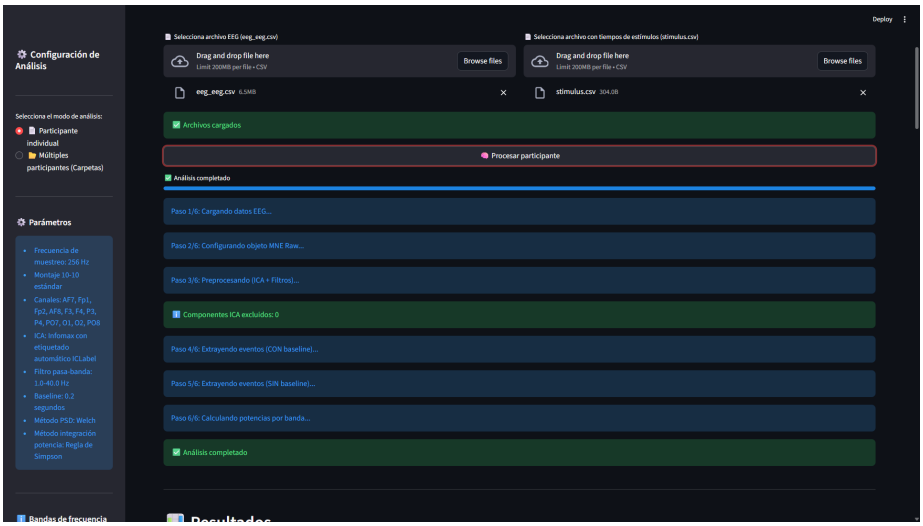


Figura B.5: Progreso del procesamiento en modo participante individual. Fuente propia

- Pestaña "Comparación *Raw* vs Filtrado": muestra gráficos superpuestos de los datos sin filtrar y preprocesados para todos los canales, permitiendo visualizar el efecto del preprocesamiento (véase la figura B.6).

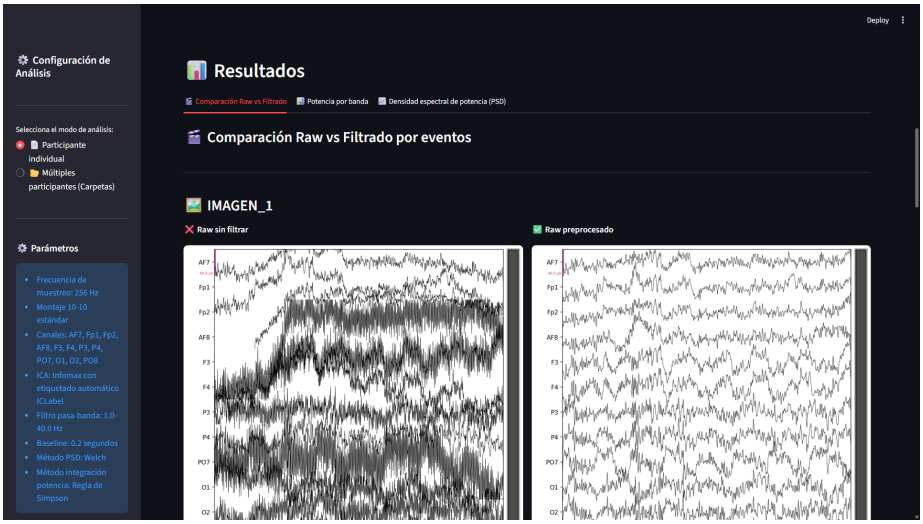


Figura B.6: Pestaña de comparación *Raw* vs Filtrado. Fuente propia

- Pestaña “Potencia por banda de frecuencia”: reúne de forma integrada todos los resultados relacionados con la potencia espectral por banda de frecuencia, incluyendo los valores numéricos por canal y banda, su distribución espacial sobre el cuero cabelludo y una representación gráfica comparativa entre electrodos (véanse las figuras B.7 y B.8).

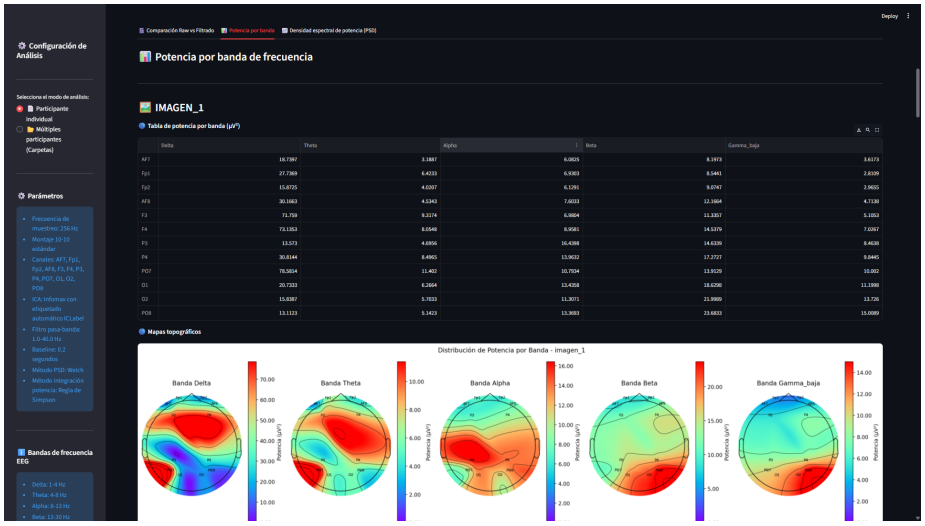


Figura B.7: Pestaña de potencia por banda de frecuencia con tabla y mapas topográficos. Fuente propia

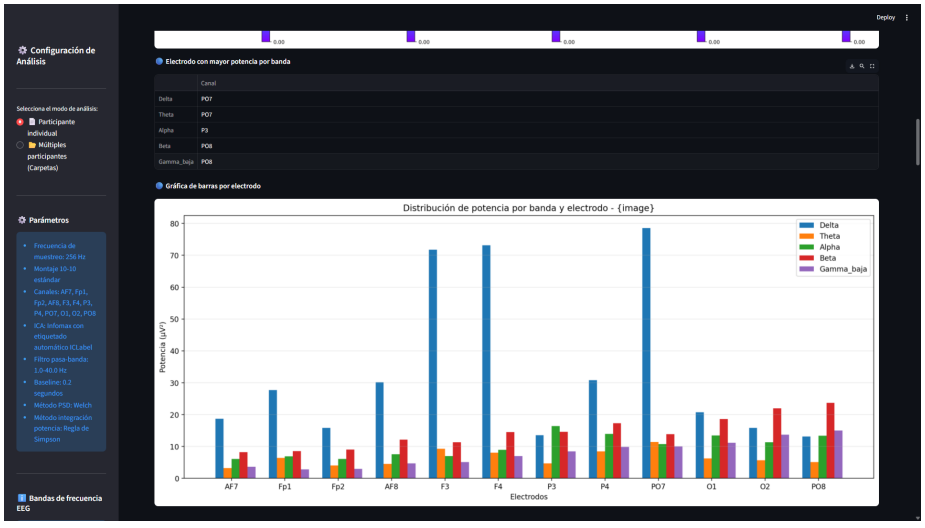


Figura B.8: Gráfica de barras por electrodo en la pestaña de potencia por banda de frecuencia. Fuente propia

- Pestaña “Densidad espectral de potencia (PSD)”: visualiza gráficos de PSD en función de la frecuencia (Hz), mostrando el espectro completo y permitiendo identificar patrones característicos de actividad cerebral. Incluye un diagrama de ubicación de electrodos en el cuero cabelludo (véase la figura B.9).

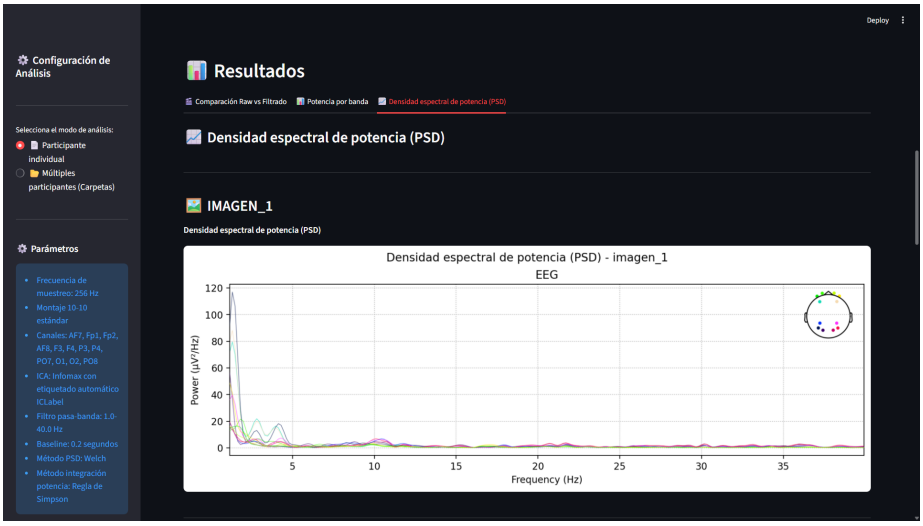


Figura B.9: Pestaña de densidad espectral de potencia. Fuente propia

Modo múltiples participantes

Carga de datos

En el modo de múltiples participantes, la interfaz explica la estructura esperada de carpetas de entrada y proporciona campos para seleccionar tanto la carpeta de entrada (**data/INPUT**) como la carpeta de salida (**data/OUTPUT**), donde se almacenan automáticamente los resultados procesados (véase la figura B.10).

Procesamiento

El progreso de procesamiento se visualiza mediante barras de estado que indican el número de participante en proceso, el estado del análisis completado y el número de componentes ICA excluidos por artefactos (véase la figura B.11).

Una vez finalizado el procesamiento de todos los participantes (véase la figura B.12), se indica el número total de participantes procesados y se

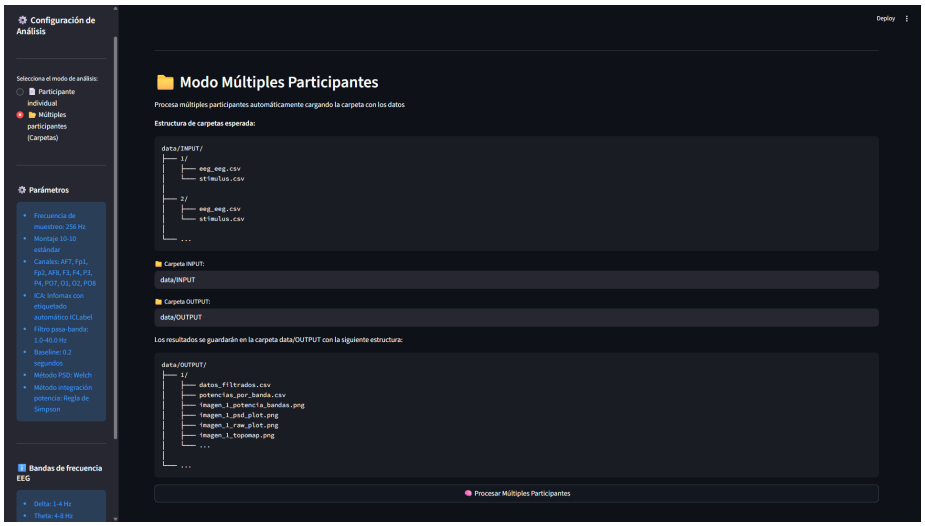


Figura B.10: Interfaz del modo múltiples participantes. Fuente propia

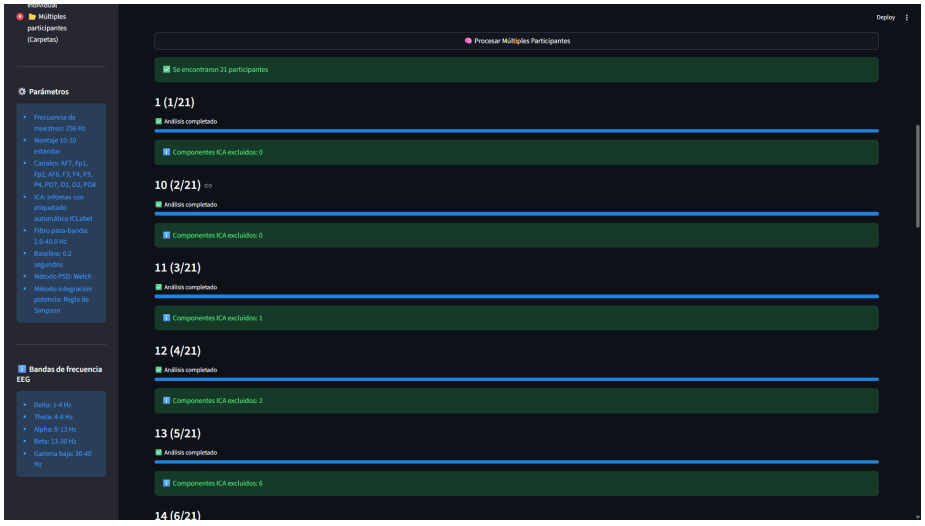


Figura B.11: Visualización del progreso de procesamiento de múltiples participantes. Fuente propia

confirma que los resultados se encuentran guardados en **data/OUTPUT** con todos los archivos generados (datos filtrados, potencias por banda e imágenes de los gráficos).

Figura B.12: Estado final del análisis y confirmación de guardado de resultados. Fuente propia

Apéndice C

Manual del programador

C.1. Introducción

El presente apéndice proporciona información técnica detallada e indispensable para la correcta comprensión, gestión y desarrollo futuro del trabajo realizado. En él se abordan aspectos fundamentales incluyendo la estructura organizativa del repositorio, los procedimientos requeridos para la compilación, instalación y ejecución del código, así como recomendaciones estratégicas para la modificación y mejora continua de la aplicación.

C.2. Estructura de directorios

El repositorio del proyecto contiene la siguiente organización jerárquica de archivos y directorios:

Archivos principales

- **eeg_processing.py**: módulo que contiene las funciones especializadas de preprocesamiento y procesamiento de señales EEG.
- **app_eeg.py**: archivo principal de la aplicación *Streamlit*. Implementa la interfaz de usuario, gestiona la interacción con el usuario y orquesta el flujo de análisis.
- **requirements.txt**: archivo de configuración que especifica el conjunto de bibliotecas de *Python* requeridas para la ejecución del proyecto, incluyendo las versiones de cada dependencia.

- **README.md**: archivo de documentación general y presentación del proyecto en formato *Markdown*.
- **LICENSE**: archivo que contiene la licencia *BSD 3-Clause "New" or "Revised" License* bajo la cual se libera el código del proyecto.
- **memoria.tex**: archivo fuente *LaTeX* que contiene el documento principal del trabajo. Incluye todas las secciones de la memoria.
- **memoria.pdf**: versión compilada en formato PDF del documento de memoria.
- **bibliografia.bib**: archivo de base de datos bibliográfica en formato *BibTeX* que contiene todas las referencias citadas en la memoria.
- **anexos.tex**: archivo fuente *LaTeX* que contiene los anexos del trabajo. Complementa la memoria con información adicional y documentación técnica detallada.
- **anexos.pdf**: versión compilada en formato PDF del documento de anexos.
- **bibliografiaAnexos.bib**: archivo de base de datos bibliográfica en formato *BibTeX* que contiene todas las referencias citadas en el documento de anexos.

Estructura de directorios

- **tex/**: directorio que contiene todos los archivos fuente *LaTeX* necesarios para la generación de la documentación del proyecto.
    1. **1_introduccion.tex**: documento *LaTeX* con la introducción del trabajo.
    2. **2_objetivos.tex**: documento *LaTeX* que especifica los objetivos que se buscan lograr en el trabajo.
    3. **3_teoricos.tex**: documento *LaTeX* que contiene los conceptos teóricos necesarios para la compresión del trabajo.
    4. **4_metodologia.tex**: documento *LaTeX* que describe la metodología empleada, incluyendo descripció de datos, herramientas *hardware* y *software* utilizadas y técnicas metodológicas de programación.

5. **5_resultados.tex**: documento *LaTeX* que contiene el resumen de resultados y discusión de estos.
6. **6_conclusiones.tex**: documento *LaTeX* que contiene las conclusiones y aspectos relevantes del trabajo.
7. **7_lineas_futuras.tex**: documento *LaTeX* que describe las líneas futuras de trabajo y mejoras potenciales.
8. **A_planificacion.tex**: documento *LaTeX* con planificación temporal, económica y viabilidad legal del proyecto.
9. **B_manual_usuario.tex**: documento *LaTeX* con guía exhaustiva de requisitos e instalación para usuarios.
10. **C_manual_programador.tex**: documento *LaTeX* con información técnica para programadores, incluyendo estructura del repositorio y procedimientos de ejecución.
11. **D_datos.tex**: documento *LaTeX* con la descripción de adquisición y tratamiento de datos.En él se realiza una descripción formal y clínica de los datos.
12. **E_diseño.tex**: documento *LaTeX* que contiene el manual de especificaciones del diseño arquitectónico modular del sistema.
13. **F_requisitos.tex**: documento *LaTeX* que contiene la especificación de requisitos funcionales del proyecto.
14. **G_experimental.tex**: documento *LaTeX* con descripción del estudio experimental, parametrización y resultados.
15. **H_ODS.tex**: documento *LaTeX* con reflexión personal sobre los aspectos de la sostenibilidad del proyecto.
16. **readme.txt**: archivo de texto que describe el contenido de la carpeta **tex/**.

- **img/**: directorio reservado para la inclusión de imágenes, gráficos y elementos visuales utilizados en la documentación.
- **data/**: directorio principal designado para almacenar archivos de datos relacionados con el proyecto. Contiene dos subdirectorios:

  1. **INPUT**: subdirectorio que contiene los datos de entrada del trabajo. Organizado por participante, cada directorio de participante contiene dos archivos:
     - **eeg_eeg.csv**: archivo con los datos EEG registrados del participante.

- **stimulus.csv**: archivo con la información temporal de los estímulos presentados durante el registro EEG.

2. **OUTPUT**: subdirectorio destinado al almacenamiento automático de resultados procesados. Organizado por participante, cada directorio de participante contiene los diferentes archivos generados tras la ejecución de la aplicación. Estos son: **datos_filtrados.csv**, **potencias_por_banda.csv**, **[imagen]_raw_plot.png, [imagen]_topomap.png, [imagen]_potencia_bandas.png e [imagen]_psd_plot.png**, siendo [imagen] el nombre de la etiqueta de cada imagen de interés mostrada durante la grabación de datos, que en este caso son cinco imágenes de interes, por lo tanto cada subdirectorio de cada participante contiene un total de 22 archivos (2 CSV y 20 PNG).

- **jupyter_notebooks/**: directorio que contiene tanto el archivo de código *Python* como el *notebook* que contiene la implementación anterior a la adopción de *Streamlit*.

C.3. Compilación, instalación y ejecución del proyecto

Para ejecutar el proyecto es necesario instalar las dependencias y ejecutar la aplicación siguiendo los procedimientos detallados a continuación.

Creación entorno virtual

Antes de la instalación de las dependencias requeridas es recomendable crear un entorno virtual aislado específicamente para este proyecto para evitar conflictos con dependencias.

Para crear el entorno virtual se deben seguir los siguientes pasos (en *Windows*):

Paso 1: abrir una terminal en el directorio raíz del proyecto

Necesita acceder a una terminal (línea de comandos) posicionada en el directorio raíz donde están guardados los archivos de la aplicación (donde están **app.py** y **requirements.txt**). Puede hacerlo de dos formas:

- Opción A: usando clic derecho.

  1. Abra el Explorador de archivos (icono de carpeta en la barra de tareas).
  2. Navegue hasta el directorio raíz del proyecto.
  3. Haga clic derecho en un espacio vacío dentro del directorio (no en un archivo, en el fondo).
  4. En el menú que aparece, busque y seleccione una de las siguientes opciones:
     - "Abrir en Terminal".
     - "Abrir PowerShell aquí".
     - "Abrir símbolo del sistema aquí"

  Se abrirá una ventana de terminal automáticamente posicionada en el directorio de proyecto.

- Opción B: usando terminal y navegando en ella.

  1. Abra el Símbolo del sistema o *PowerShell* desde el menú Inicio.
  2. Navegue hasta el directorio raíz del proyecto utilizando el comando:

     ```
     cd C:\ruta\completa\a\su\directorio
     ```

  3. Pulse Enter.

Paso 2: verificar que está en la ubicación correcta

Una vez que la terminal está abierta en el directorio correcta, debería ver:

```
C:\ruta\completa\a\su\directorio>
```

Para verificar que está en el directorio correcto, escriba el siguiente comando y pulse Enter:

```
ls
```

Debería ver listados los archivos de su proyecto, incluyendo **app_eeg.py** y **requirements.txt**.

Paso 3: crear el entorno virtual

Ahora que está dentro del directorio, cree el entorno virtual escribiendo el siguiente comando y pulsando Enter:

```
python -m venv .venv
```

Este comando genera un directorio denominada **.venv** dentro del directorio del proyecto, conteniendo una copia del interprete de *Python* y sus herramientas de gestión de paquetes.

Paso 4: activar el entorno virtual

Una vez creado, es necesario activarlo antes de instalar las dependencias:

```
.\.venv\Scripts\Activate.ps1
```

Si el comando se ejecuta correctamente, debería ver que el inicio de la línea de comandos cambió y ahora muestra `(.venv)` al principio:

```
(.venv) C:\ruta\completa\a\su\directorio>
```

Si aparece un mensaje de error indicando que las directivas de ejecución están restringidas, ejecutar el siguiente comando dependiendo de si quiere conceder los permisos de forma permanente o unicamente de forma temporal (mientras esa ventana está abierta):

- Permanente:

  ```
  Set-ExecutionPolicy -ExecutionPolicy ‘
  RemoteSigned -Scope CurrentUser
  ```

- Temporal:

  ```
  Set-ExecutionPolicy -Scope Process ‘
  -ExecutionPolicy RemoteSigned
  ```

Posteriormente, reintentar la activación del entorno virtual.

Para usuarios de *Linux* o *macOS*

Si utiliza *Linux* o *macOS* en lugar de *Windows*, ejecute los siguientes comandos en su terminal:

1. Abra la terminal y navegue hasta el directorio raíz del proyecto.

   ```
   cd /ruta/completa/a/su/directorio
   ```

2. Crear el entorno virtual.

   ```
   python3 -m venv .venv
   ```

3. Activar el entorno virtual.

   ```
   source .venv/bin/activate
   ```

   Debería ver que el inicio de la línea de comandos cambió y ahora muestra `(.venv)` al principio.

Instalación de las bibliotecas de *Python*

El archivo *requirements.txt* contiene la especificación detallada de las bibliotecas de *Python* necesarias para la ejecución del trabajo, indicando explícitamente las versiones específicas de cada dependencia. Para proceder a la instalación automática de todas las dependencias especificadas, debe ejecutarse el siguiente comando en la terminal desde el directorio raíz del proyecto:

```
pip install -r requirements.txt
```

Ejecución de la aplicación

Tras completar la instalación de las bibliotecas requeridas, proceda a ejecutar la aplicación en la terminal desde el directorio raíz del proyecto mediante una de las siguientes opciones:

- Opción 1: terminal tradicional. Emplear la terminal como en los pasos anteriores.

  ```
  streamlit run app_eeg.py
  ```

- Opción 2: *Visual Studio Code* (*VS Code*). Emplear *Visual Studio Code* como entorno de desarrollo integrado que facilita la edición, depuración y ejecución del proyecto.

  Para su instalación se debe:

  1. Acceder al sitio web oficial de *Visual Studio Code* y proceder a descargar e instalar el editor de código. Es compatible con *Windows (10 y 11)*, *Linux* principales distribuciones y *macOS (11.0+)*, tal como se muestra en la figura C.1.

Figura C.1: Página web descarga *Visual Studio Code*. [Corporation, sf]

  2. Una vez instalado *Visual Studio Code*, abrir la aplicación para proceder a la instalación de la extensión de *Python*. Para ello se debe acceder a la sección de extensiones dentro de *Visual Studio Code* (icono de bloques en la barra lateral izquierda o `Ctrl+Shift+X`) y buscar la extensión oficial de *Python* desarrollada por *Microsoft* (como se muestra en la figura C.2). Descargar e instalar dicha extensión, que incluye automáticamente la integración de *Python* en el editor.

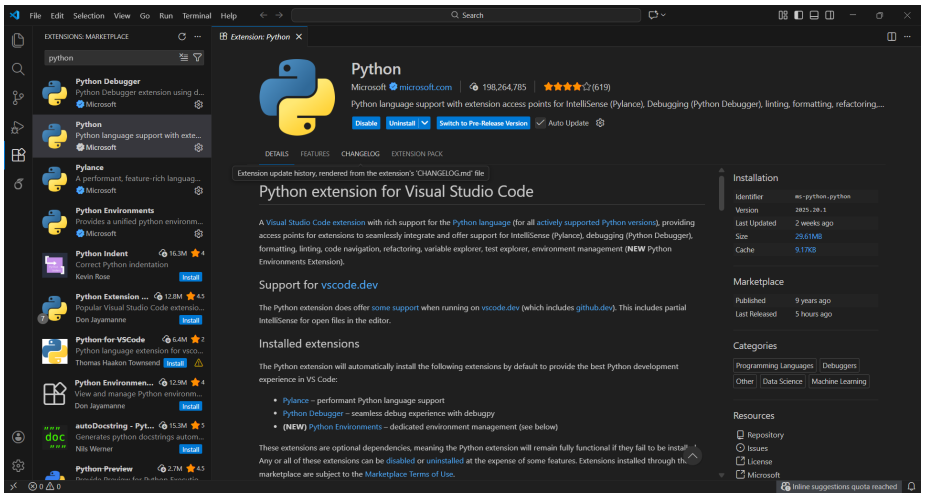


Figura C.2: Instalación *Python*. Fuente propia

Para ejecutar la aplicación desarrollada se debe:

1. Abrir el directorio del proyecto: `"File > Open Folder"`.
2. Seleccionar intérprete *Python*: abrir la paleta de comandos con `Ctrl+Shift+P`, en ella buscar `Python: Select Environment` y seleccionar el entorno anteriormente creado `.venv`. Si aparece un mensaje de error indicando que las directivas de ejecución están restringidas, proceder cómo se indica en la subsección C.3 ante la aparición de este error, pero en este caso se puede realizar en una terminal integrada en *Visual Studio Code* (`"Terminal > New Terminal"`).
3. Abrir la terminal integrada en *Visual Studio Code*: `"Terminal > New Terminal"` y hay que asegurarse de que se encuentra posicionada en el directorio raíz del proyecto y el entorno está activo.
4. Ejecutar:

```
streamlit run app_eeg.py
```

La aplicación de *Streamlit* abrirá automáticamente una ventana en el navegador web predeterminado del sistema, dirigiéndose a la dirección local `http://localhost:8501`, donde será presentada la interfaz interactiva. Esta cuenta con *widgets* desplegables con las instrucciones para su uso y explicación del flujo de trabajo.

C.4. Instrucciones para la modificación o mejora del proyecto

Para editar o mejorar el código se recomienda *Visual Studio Code* con la extensión oficial de *Python*, configurado con un entorno virtual para el proyecto.

Las instrucciones y recomendaciones para la mejora del trabajo en futuras ediciones se describen detalladamente en el capítulo "líneas futuras" de la memoria.

Apéndice D

Descripción de adquisición y tratamiento de datos

D.1. Introducción

El presente apéndice proporciona una descripción detallada del origen y la naturaleza de los datos utilizados en este trabajo.

D.2. Descripción formal de los datos

Datos de entrada

Los datos electroencefalográficos fueron exportados en formato CSV-Lite mediante el *software SennsLab*. Este formato proporciona datos completos sin filtrado digital adicional, preservando todo el contenido espectral registrado.

1. El archivo principal **eeg_eeg.csv** contiene los registros de potencial eléctrico crudos adquiridos de la grabación del experimento para cada participante. La estructura del archivo comprende un total de 15 columnas organizadas de la siguiente manera [Bitbrain, sfb, Bitbrain, 2024]:

   - Canales de EEG (12 columnas): los primeros 12 campos corresponden a los canales EEG individuales, denominados según la nomenclatura estándar 10-10 internacional:

     a) **AF7**
     b) **Fp1**
     c) **Fp2**

d) **AF8**
e) **F3**
f) **F4**
g) **P3**
h) **P4**
i) **PO7**
j) **O1**
k) **O2**
l) **PO8**

Cada una de estas columnas contiene valores de amplitud de voltaje en unidades de microvoltios (μV).

- **sequenceNumber**: Número secuencial entero que identifica las muestras de datos adquiridas por bloques, funciona como contador.
- **timestampReception**: marca temporal en formato numérico que registra el momento en que el software de adquisición recibió el bloque de datos. Se expresa en unidades de microsegundos (μs) desde una época de referencia.
- **timestamp**: marca de tiempo corregida calculadas automáticamente utilizando el valor t0 (*timestamp* inicial de la grabación). Esta corrección asume una transmisión uniforme de datos según:

$$\text{timestamp}[i] = t0 + (i * T\_s) \tag{D.1}$$

donde i es el índice de la muestra y $T\_s = 1/f\_s$ es el periodo de muestreo. Esta columna proporciona una escala temporal uniforme que asume una transmisión de datos sin pérdidas, facilitando la sincronización correcta con eventos externos y la alineación de grabaciones simultáneas de múltiples dispositivos. Se expresa en μs.

La frecuencia de muestreo del dispositivo EEG es de 256 Hz, lo que significa que se adquieren aproximadamente 256 muestras por segundo en cada canal. Esto proporciona una resolución temporal de aproximadamente 3.9 milisegundos entre muestras consecutivas.

2. El archivo **stimulus.csv** complementa los datos de EEG con información sobre los eventos experimentales. Contiene marcas temporales que indican los instantes en los cuales ocurrieron los eventos del protocolo

experimental, tales como instantes de transición entre fases del protocolo y la presentación de estímulos visuales incluidos en el protocolo experimental.

Estas marcas temporales vienen dadas en tres columnas:

a) **timestamp**: marca temporal en microsegundos que indica el instante de transición entre fases experimentales, permitiendo sincronizar los cambios de condición experimental con la actividad EEG registrada.

b) **eventType**: identificador categórico del tipo de fase que comienza tras el evento, codificando numéricamente según el protocolo experimental:

- **0**: corresponde a fase de atención.
- **1**: corresponde a presentación de estímulo visual.
- **2**: corresponde a fase de descanso.

c) **eventIndex**: identificador numérico asignado según el tipo de evento experimental. Este valor sigue el siguiente esquema:

- **0**: corresponde a fase de atención.
- **2**: corresponde a fase de descanso.
- Valores progresivos, excepto el número **2** (**1**, **3**, **4**, **5**, **6**): corresponden a la presentación de estímulos visuales (imágenes) con valores progresivos asignados progresivamente a cada imagen presentada a cada imagen presentada durante el protocolo, a excepción del **2** que corresponde con la fase de descanso.

El protocolo experimental estructura las sesiones en 16 eventos de transición que definen 15 intervalos consecutivos siguiendo el patrón: atención, imagen, descanso. Repitiéndose este ciclo el número de imágenes añadidas al protocolo experimental y comenzando por otro evento de atención antes de comenzar los ciclos del patrón. Cada fila en **stimulus.csv** marca la transición entre intervalos consecutivos, permitiendo segmentar el registro EEG en fases experimentales identificadas para posterior análisis.

Datos de salida

Se exportan los datos preprocesados y procesados en archivos CSV delimitado por coma, **datos_filtrados.csv** y **potencias_por_banda.csv**

respectivamente. **datos_filtrados.csv** contiene todos los datos de correspondientes al archivo de entrada **eeg_eeg.csv**, pero tras ICA y el filtrado digital, sin la aplicación de línea base y con una primera fila que se corresponde con los nombres de los electrodos para poder conocer con que se corresponde cada columna. Y el resto de filas se corresponden con los registros de potencial eléctrico preprocesados en unidades de μV. **potencias_por_banda.csv** contiene las potencias obtenidas tras el análisis. Está compuesto por 13 filas, la primera fila indica las etiquetas correspondientes de las potencias por columna, estas etiquetas son 25 que siguen la siguiente estructura: **[imagen]_Delta, [imagen]_Theta, [imagen]_Alpha, [imagen]_Beta, [imagen]_Gamma_baja**, siendo [imagen] el nombre de la etiqueta de esa imagen. Las 12 filas restantes contiene las potencias para cada etiqueta en cada electrodo en μV^2, estas 12 filas siguen el orden de electrodos: **AF7**, **Fp1**, **Fp2**, **AF8**, **F3**, **F4**, **P3**, **P4**, **PO7**, **O1**, **O2** y **PO8**.

Además, se generaron archivos PNG de las gráficas realizadas en el análisis. Son cuatro gráficas de las diferentes visualizaciones realizadas para el análisis, explicadas en la sección G.2 del apéndice "Estudio experimental", por cada imagen de interés que en este caso son cinco por lo que hay 20 imágenes con las diferentes gráficas. Estas siguen la siguiente estructura: **[imagen]_raw_plot.png, [imagen]_topomap.png, [imagen]_potencia_bandas.png e [imagen]_psd_plot.png**, siendo [imagen] el nombre de la etiqueta de esa imagen.

D.3. Descripción clínica de los datos

Datos de entrada

Como se describe en los "Conceptos teóricos" de la memoria, la señal EEG se origina en las neuronas piramidales de la corteza cerebral y se capturan mediante electrodos colocados en la superficie craneal, que miden la diferencia de potencial eléctrico.

Los datos de **eeg_eeg.csv** contienen las muestras de esta actividad cerebral adquiridas a 256 Hz. Como se menciona en los "Conceptos teóricos" de la memoria, la señal es no lineal y no estacionaria, por lo que sus características varían con el tiempo. También se describe que los datos no solo contienen actividad cerebral sino también señales electrofisiológicas del cuerpo humano y señales no fisiológicas, siendo necesario el preprocesamiento para obtener una señal útil.

Además, como se describe en los "Conceptos teóricos" de la memoria, las ondas cerebrales son oscilaciones rítmicas de la actividad eléctrica cerebral que surgen de la activación sincronizada de grupos de neuronas, generando fluctuaciones de voltaje capturadas por los electrodos. Estas oscilaciones se clasifican según su frecuencia en diferentes tipos de ondas, cada una asociada a diversos estados cognitivos y fisiológicos. El sistema *Minimal EEG Diadem* permite registrar actividad eléctrica en el rango de adquisición DC - 40 Hz (filtro pasa-bajo 3er orden) [Bitbrain, sfa], donde DC (corriente directa) representa el componente de baja frecuencia sin filtrado, incluyendo variaciones lentas desde 0 Hz hasta 40 Hz. Este rango permite capturar la siguiente clasificación de bandas de frecuencia que están presentes en los datos:

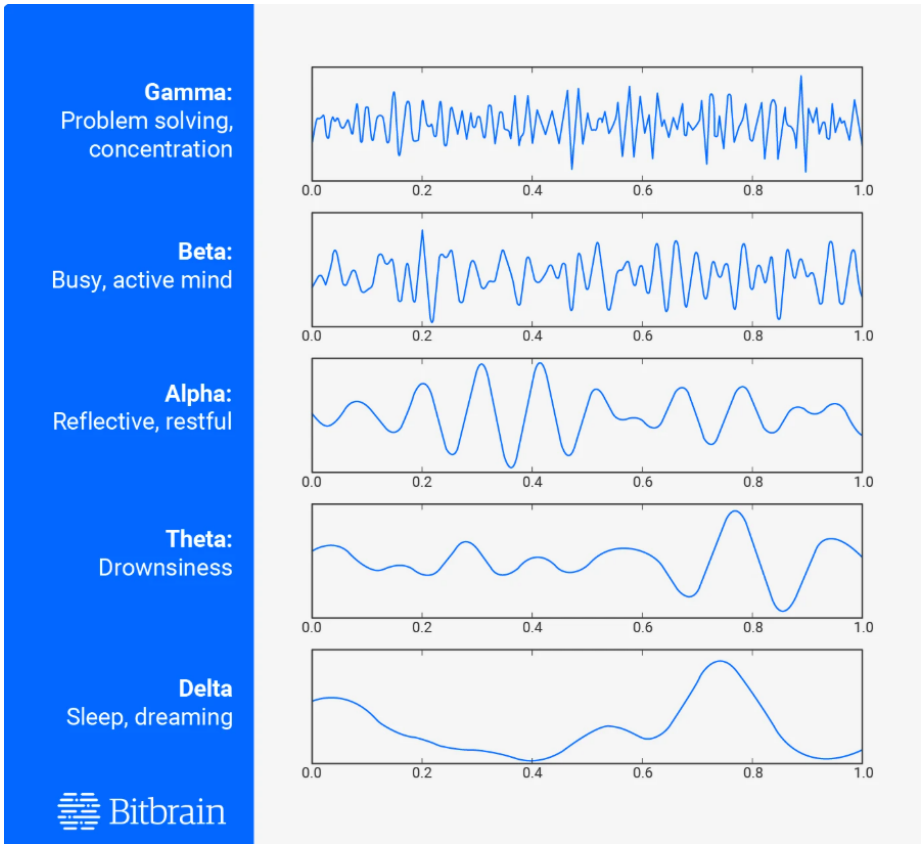


Figura D.1: Representación gráfica de las ondas cerebrales (delta, theta, alpha, beta y gamma) [Shure and Mínguez, 2024].

El archivo **stimulus.csv** contiene los eventos experimentales presentados durante la adquisición de datos EEG, posibilitando la sincronización

Delta	0.5-4 Hz
Theta	4-8 Hz
Alpha	8-13 Hz
Beta	13-30 Hz
Gamma baja	30-40 Hz

Tabla D.1: Clasificación de rangos de bandas de frecuencias. [Webster and Nimunkar, 2020]

temporal con los registros de actividad cerebral. Permitiendo de esta manera alinear la actividad cerebral registrada con los eventos de específicos del protocolo experimental.

Datos de salida

datos_filtrados.csv contiene la señal preprocesada, se han eliminado los artefactos y se ha filtrado, por lo que es útil para comparar con otros métodos de preprocesamiento de la señal y también permite usar esta señal "limpia" para otro tipo de análisis posterior diferente al realizado en este trabajo.

En **potencias_por_banda.csv**, los valores de potencia espectral en cada banda de frecuencia permiten caracterizar el patrón de actividad cerebral presente durante el registro EEG y pueden indicar diferentes condiciones fisiológicas o patológicas. Pero se necesita conocimientos específicos para poder realizar su interpretación, por eso es que se genera este fichero, para que profesionales puedan llevar a cabo su interpretación sin necesidad de realizar la ejecución del programa.

Así mismo, las imágenes generadas facilitan la interpretación, aunque se sigue requiriendo conocimientos específicos para poder realizar su interpretación a nivel de identificar condiciones fisiológicas o patológicas. Pero si que se puede interpretar de forma sencilla tanto la ubicación de la actividad cerebral, como la potencia de esta actividad cerebral a nivel general.

Apéndice E

Manual de especificación de diseño

E.1. Diseño arquitectónico

Arquitectura modular

El sistema de preprocesado y procesado de datos crudos de EEG se ha estructurado siguiendo el patrón de separación de responsabilidades, dividiendo la lógica de preprocesamiento y procesamiento de datos de la interfaz de usuario. Esta arquitectura modular facilita la reutilización del código, el mantenimiento y la escalabilidad del sistema.

La aplicación está diseñada en dos módulos principales:

- **eeg_preprocess.py**: módulo de preprocesamiento y procesamiento de datos EEG.
  - Contiene todas las funciones de carga, preprocesamiento, análisis espectral y exportación de resultados.
  - No tiene dependencias de *Streamlit*, permitiendo su uso independiente en otros contextos.
  - Funciones principales:
    - `load_data()`: carga archivos CSV de EEG y tiempos de estímulo.
    - `setup_mne_raw()`: configura objeto `mne.io.Raw` con montaje estándar 10-10.

- `preprocess_raw()`: aplica filtros, referencia promedio e ICA con etiquetado automático.
- `get_events_baseline()`: extrae épocas con corrección de línea base.
- `get_events_no_baseline`: extrae épocas sin corrección de línea base.
- `get_band_power()`: calcula la potencia por banda de frecuencia y la PSD.
- `save_plots()`: genera y guarda gráficas de análisis.
- `save_results()`: exporta datos filtrados y potencias a CSV.

- **app_eeg.py**: módulo de interfaz de usuario basado en *Streamlit*.
  - Importa y utiliza las funciones del módulo `eeg_preprocess.py`.
  - Gestiona la interacción del usuario mediante la interfaz web.
  - Define los dos modos de análisis: participante individual y múltiples participantes.
  - Componentes principales de la interfaz:
    - Secciones informativas iniciales: desplegables con la descripción del flujo de análisis y de los modos de uso disponibles.
    - Barra lateral de configuración: selector del modo de análisis, campos informativos de los parámetros técnicos y las bandas de frecuencia EEG.
    - Área de carga:
      - En modo individual, se emplean *widgets* para seleccionar **eeg_eeg.csv** y **stimulus.csv**.
      - En modo múltiples participantes, campos de texto para las rutas de **data/INPUT** y **data/OUTPUT**.
    - Botones de ejecución y *feedback*: botones para iniciar el análisis, junto con barras de progreso y mensajes de estado en cada fase del *pipeline*.
    - Pestañas de resultados (participante individual): tres pestañas para la comparación de la señal *raw vs.* preprocesada, análisis de potencias por banda de frecuencia y visualización de la densidad espectral de potencia.

Diagrama de paquetes

La figura E.1 muestra la organización modular del sistema en forma de diagrama de paquetes de lenguaje unificado de modelado (Unified Mo-

deling Language o UML), diferenciando la capa de interfaz de usuario (**app_eeg.py**), la capa de procesamiento EEG (**eeg_processing.py**) y las dependencias externas.

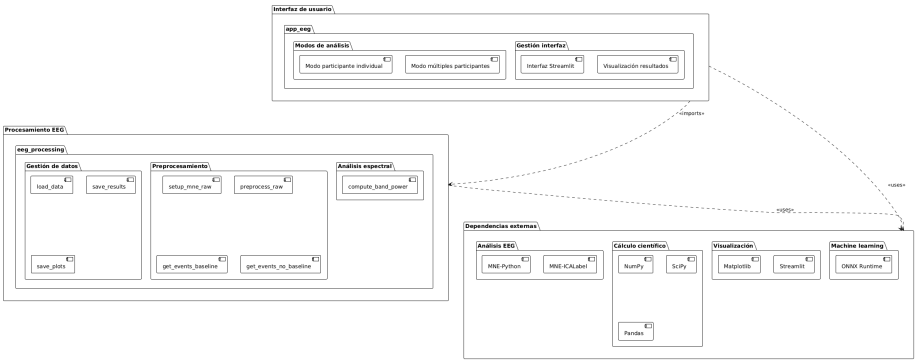


Figura E.1: Diagrama de paquetes del sistema de preprocesamiento y procesamiento de EEG. Fuente propia

Diagrama de despliegue

En la figura E.2 se muestra el diagrama de despliegue del sistema, indicando los componentes de software instalados en el cliente y las dependencias externas.

Flujo de datos

De forma general, el flujo de datos entre módulos sigue el siguiente patrón:

1. El usuario interactúa con la interfaz de *Streamlit* definida en **app_eeg.py** (selección del modo de análisis y carga de los archivos CSV).
2. El *script* **app_eeg.py** invoca la función `load_data()` del módulo **eeg_processing.py**, obteniendo como resultado un *DataFrame* con los datos EEG y un vector de tiempos de estímulo en segundos.
3. A partir de estos datos, **app_eeg.py** llama secuencialmente a las funciones de procesamiento del módulo **eeg_processing.py**:
   - `setup_mne_raw()`: configura el objeto *Raw* de *MNE*.
   - `preprocess_raw()`: aplica el preprocesamiento.

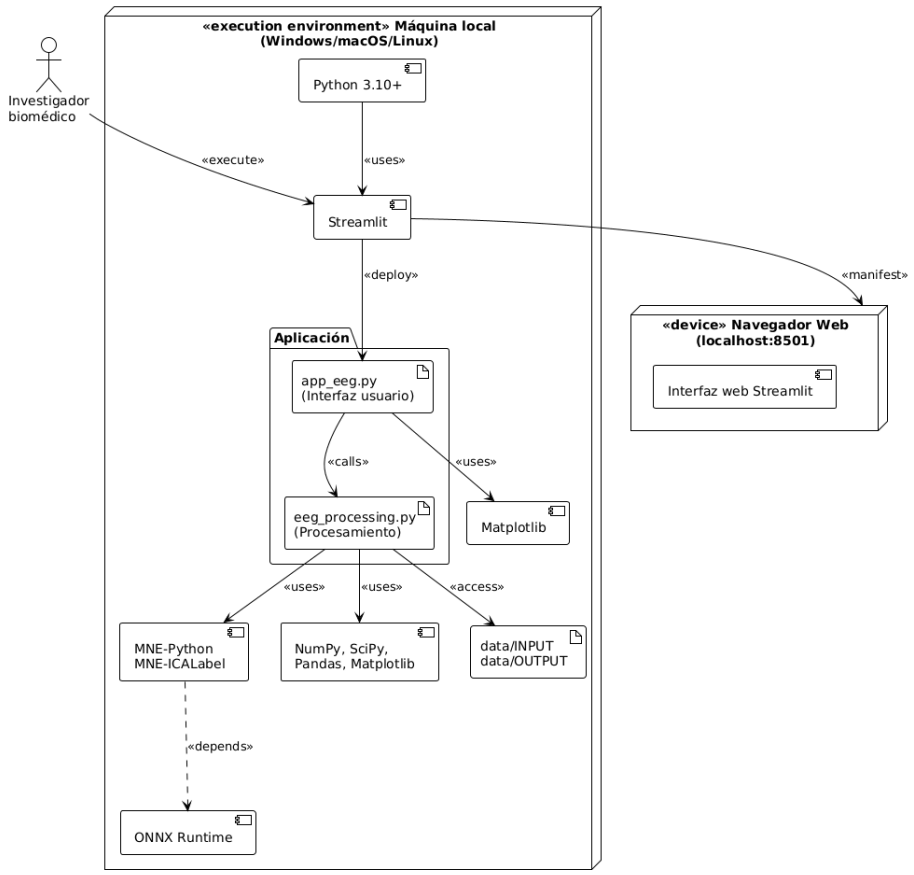


Figura E.2: Diagrama de despliegue del sistema de preprocesamiento y procesamiento de EEG en máquina local. Fuente propia

- `get_events_baseline()` y `get_events_no_baseline`: extraen las épocas asociadas a los estímulos visuales, con y sin corrección de línea base.
- `get_band_power()`: calcula la potencia por banda de frecuencia y la densidad de potencia espectral (PSD) sobre las épocas resultantes.

Las interacciones entre componentes se modelan mediante diagramas de secuencia UML (figuras E.3 y E.4), que representan el orden temporal de los mensajes intercambiados entre módulos.

Flujo en modo participante individual

En modo "Participante individual", el flujo de datos sigue las 16 interacciones temporales entre la interfaz Streamlit, el módulo de procesamiento EEG y las visualizaciones generadas, que se muestran en la figura E.3.

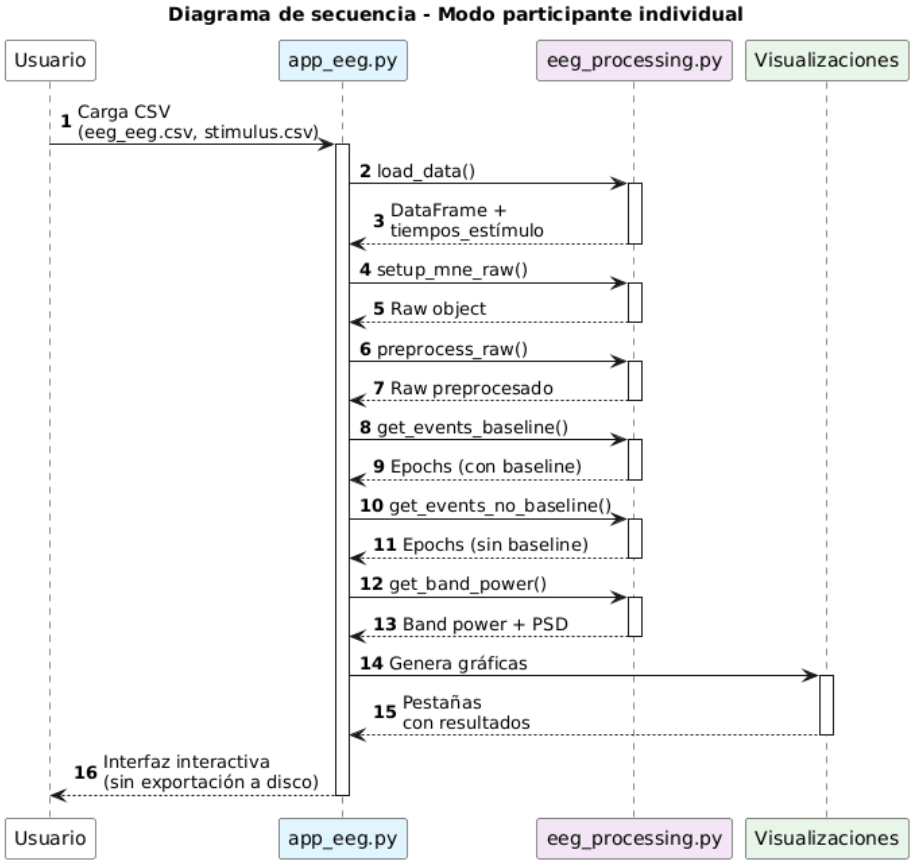


Figura E.3: Diagrama de secuencia UML - Modo participante individual. Fuente propia

De manera sintetizada, el flujo comprende los siguientes pasos:

1. El usuario carga los archivos **eeg_eeg.csv** y **stimulus.csv** desde la interfaz de **app_eeg.py**.
2. **app_eeg.py** ejecuta el *pipeline* descrito previamente, llamando a las funciones de **eeg_processing.py**.
3. Los resultados se utilizan para generar visualizaciones que se muestran en las diferentes pestañas de la interfaz.

4. En este modo no se realiza exportación automática a disco; el objetivo principal es la inspección visual de un participante concreto.

Flujo en modo múltiples participantes

En el modo "Múltiples participantes (Carpetas)", el flujo de datos se adapta al procesamiento por lotes (20 pasos) con un bucle iterativo que aplica el *pipeline* completo a cada participante (figura E.4).

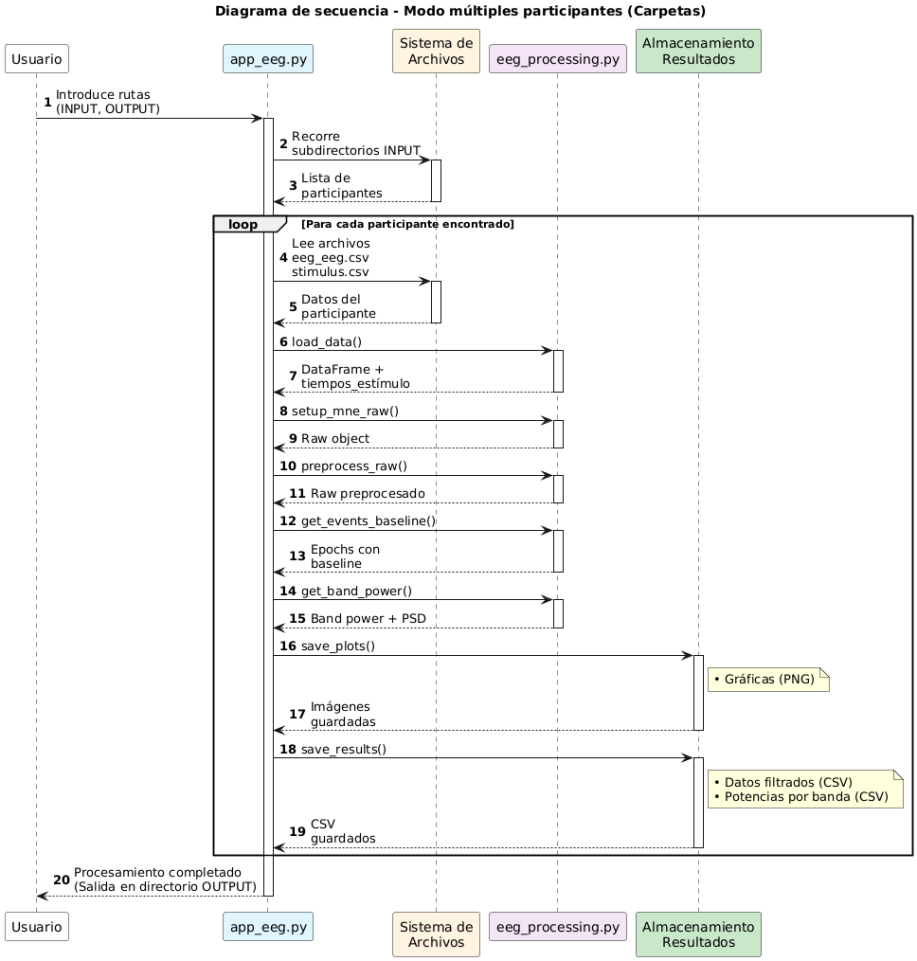


Figura E.4: Diagrama de secuencia UML - Modo múltiples participantes. Fuente propia

De manera sintetizada, el flujo comprende los siguientes pasos:

1. El usuario introduce en **app_eeg.py** las rutas de los directorios `INPUT` y `OUTPUT`.

2. **app_eeg.py** recorre los subdirectorios de `INPUT` y, para cada participante, llama al *pipeline* de funciones de `eeg_processing.py`, guardando en el directorio `OUTPUT` los datos filtrados, las potencias por banda y las gráficas generadas.

3. En este modo no se generan visualizaciones de los resultados en la interfaz; toda la salida se produce en forma de archivos CSV e imágenes para su análisis posterior.

Apéndice F

Especificación de Requisitos

El sistema debe cumplir los siguientes requisitos funcionales:

F.1. Diagrama de casos de uso

En la figura F.1 se muestra el diagrama de casos de usos del sistema de preprocesamiento y procesamiento de EEG desarrollado, donde el actor "Investigador biomédico" interactúa con la aplicación web para cargar datos, ejecutar el preprocesamiento y análisis, visualizar resultados y procesar múltiples participantes en modo por lotes.

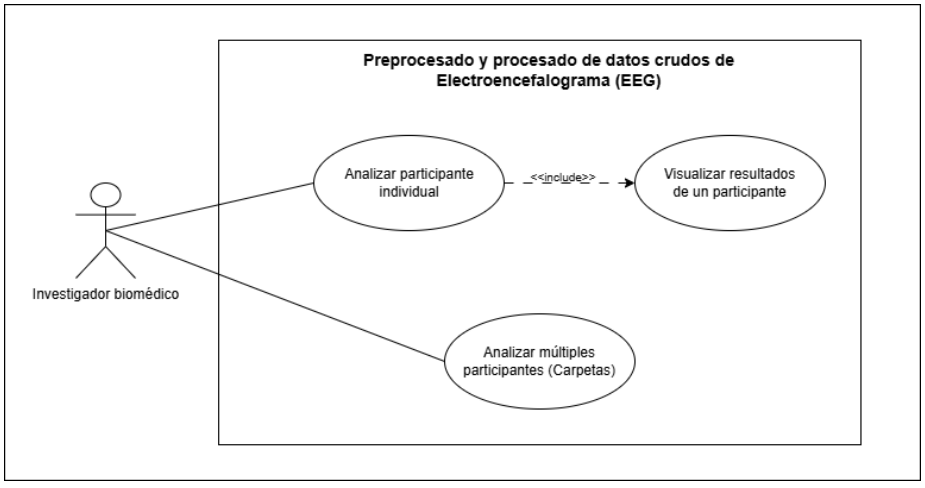


Figura F.1: Diagrama de casos de uso del sistema de preprocesamiento y procesamiento de EEG. Fuente propia

ID	Descripción
RF-1	El sistema debe permitir seleccionar entre dos modos de operación: "Participante individual" y "Múltiples participantes (Carpetas)".
RF-2	En modo individual, el sistema debe permitir cargar archivos CSV (EEG e tiempos de estímulo), validar su formato y ejecutar automáticamente el *pipeline* de análisis.
RF-3	El sistema debe implementar un preprocesamiento automático que incluya: filtros paso-banda (1-40 Hz) y notch (50 Hz), referencia promedio, ICA con etiquetado automático de artefactos, y corrección de línea base.
RF-4	El sistema debe extraer épocas de señal asociadas a los cinco estímulos visuales del experimento, tanto con como sin corrección de línea base.
RF-5	El sistema debe calcular la potencia por banda de frecuencia (Delta, Theta, Alpha, Beta, Gamma baja) y la densidad espectral de potencia (PSD) mediante Welch para cada época.
RF-6	El sistema debe visualizar los resultados en tres pestañas: (1) comparación raw vs filtrado, (2) potencia por banda con tablas, mapas topográficos y gráficas de barras, y (3) PSD.
RF-7	En modo múltiples participantes, el sistema debe procesar automáticamente todos los sujetos de un directorio de entrada, ejecutando el *pipeline* completo para cada uno.
RF-8	El sistema debe exportar los resultados del procesamiento por lotes en formato CSV (datos filtrados y potencias) e imágenes PNG (gráficas), organizados en subdirectorios por participante.
RF-9	El sistema debe mostrar barras de progreso y mensajes de estado durante el procesamiento para informar al usuario del avance.

Tabla F.1: Requisitos Funcionales del sistema. Fuente propia

El actor principal es el investigador biomédico, que interactúa con la aplicación web (implementada en *Streamlit*) para:

- Cargar los datos EEG y los tiempos de estímulos de un participante.

- Ejecutar el preprocesamiento automático de las señales (filtros e ICA con etiquetado automático).
- Realizar el análisis espectral y el cálculo de potencias por banda de frecuencia.
- Visualizar los resultados.
- Procesar por lotes múltiples participantes y exportar los resultados a ficheros CSV e imágenes PNG.

F.2. Explicación casos de uso

A continuación se describen los casos de uso principales del sistema mediante tablas estructuradas.

CU-1	**Analizar participante individual**
Versión	1.0
Autor	Alumno
Requisitos asociados	RF-1, RF-2, RF-3, RF-4, RF-5
Descripción	El investigador selecciona el modo individual, carga los archivos EEG y estimulos de un sujeto, y la aplicación ejecuta automáticamente el flujo de preprocesamiento, extracción de épocas y visualización de resultados.
Precondición	▪ La aplicación está desplegada y accesible vía navegador. ▪ Los archivos **eeg__eeg.csv** y **stimulus.csv** existen y siguen el formato definido por *Bitbrain*. ▪ Las dependencias *software* y el entorno *Python* están correctamente configurados.
Acciones	1. El usuario selecciona el modo "Participante individual" en la barra lateral. 2. El usuario sube sube los archivos **eeg__eeg.csv** y **stimulus.csv** mediante los selectores de archivo. 3. La aplicación confirma la carga correcta de ambos archivos. 4. El usuario pulsa el botón "Procesar participante". 5. La aplicación ejecuta automáticamente el *pipeline* de análisis: preprocesamiento, extracción de épocas y cálculo de potencias espectrales. 6. La aplicación actualiza la interfaz mostrando las pestañas de resultados para su exploración.
Postcondición	Los datos del participante quedan cargados y preprocesados en memoria. Se disponen de las épocas por imagen, las potencias por banda y los espectros de potencia listos para su consulta en la interfaz.
Excepciones	Formato de archivo no válido. *MemoryError* en caso de insuficiente memoria RAM, o errores de convergencia en ICA (*RuntimeError*).
Importancia	Alta.

Tabla F.2: CU-1 Analizar participante individual. Fuente propia

CU-2	**Visualizar resultados de un participante**
Versión	1.0
Autor	Alumno
Requisitos asociados	RF-6
Descripción	El investigador explora los resultados del análisis de EEG de un participante a través de las pestañas de la interfaz: comparación de la señal *raw vs.* preprocesada, potencias por banda de frecuencia y densidad espectral de potencia (PSD).
Precondición	El caso de uso CU-1 se ha completado correctamente.
Acciones	1. El usuario accede a la sección "Resultados". 2. Explora las pestañas de visualización: comparación *raw vs.* filtrado, potencias por banda (tablas, mapas topográficos y gráficas de barras) y densidad espectral de potencia.
Postcondición	El usuario ha inspeccionado visualmente la calidad del preprocesamiento, la distribución espacial de la potencia por bandas y las características espectrales de las señales EEG del participante.
Excepciones	Fallos en la generación de figuras de *MNE/Matplotlib* o errores de refresco de la interfaz si se interrumpe la sesión de *Streamlit*.
Importancia	Media.

Tabla F.3: CU-2 Visualizar resultados de un participante. Fuente propia

CU-3	**Procesar múltiples participantes**
Versión	1.0
Autor	Alumno
Requisitos asociados	RF-1, RF-7, RF-8, RF-9
Descripción	El investigador selecciona el modo "Múltiples participantes (Carpetas)" para procesar automáticamente todos los sujetos de un directorio, generando datos filtrados, potencias por banda y gráficas de resultados organizadas por participante.
Precondición	▪ Existe un directorio **INPUT** con subdirectorios por participante conteniendo **eeg_eeg.csv** y **stimulus.csv**. ▪ El directorio **OUTPUT** es accesible y tiene permisos de escritura. ▪ Las dependencias *software* y el entorno *Python* están correctamente configurados.
Acciones	1. El usuario selecciona el modo "Múltiples participantes (Carpetas)" en la barra lateral. 2. Introduce las rutas de los directorios **INPUT** y **OUTPUT**. 3. Pulsa el botón "Procesar Múltiples Participantes". 4. La aplicación recorre los subdirectorios de **data/INPUT** identificando los participantes disponibles. 5. Para cada participante, la aplicación ejecuta automáticamente el *pipeline* de análisis y guarda los resultados en **OUTPUT** 6. La aplicación actualiza la barra de progreso mostrando el avance del procesamiento.
Postcondición	En el directorio **OUTPUT** se dispone de un subdirectorio por participante con los datos EEG filtrados, las potencias por banda y las figuras asociadas.
Excepciones	Directorio inexistente, archivos ausentes, errores de escritura o fallos de procesamiento.
Importancia	Alta.

Tabla F.4: CU-3 Procesar múltiples participantes. Fuente propia

Apéndice G

Estudio experimental

G.1. Introducción

Este apéndice documenta el proceso experimental del desarrollo del protocolo de preprocesamiento y procesamiento de EEG. Se describen los métodos probados y decisiones técnicas adoptadas, así como la configuración y parametrización de los métodos empleados. De esta forma, el presente apéndice garantiza la transparencia y reproducibilidad necesarias al documentar de manera detallada la metodología y parámetros seleccionados, así como los resultados obtenidos.

G.2. Cuaderno de trabajo

Esta sección documenta los pasos que se han llevado a cabo para cumplir los objetivos fijados inicialmente. Esta documentación va a permitir que otros investigadores puedan replicar el protocolo, validar la metodología o adaptar el *pipeline* a otros contextos.

Métodos probados y decisiones adoptadas

Selección del lenguaje de programación

Se investigó sobre diferentes lenguajes de programación para trabajar con datos EEG. Entre ellos destacaban *MATLAB* con *EEGLAB* y *Python* con *MNE-Python*.

- *MATLAB* con *EEGLAB*: *EEGLAB* es una herramienta estándar muy empleada en neurociencia computacional, con interfaz gráfica intuitiva y con gran volumen de documentación [contributors, 2026, Delorme, 2026]. Sin embargo, tiene un incoveniente considerable y es que *MATLAB* requiere de licencia de pago.

- *Python* con *MNE-Python*: *MNE-Python* es una biblioteca especializada también en EEG, cuenta con una comunidad neurocientífica activa con extensa documentación y es de código abierto [contributors, 2025].

Se adoptó *Python* con *MNE-Python* como entorno de programación porque proporciona la combinación óptima de especialización en EEG, comunidad científica activa, facilidad de prototipado y código abierto sin barreras económicas. Esta elección fundamentó todas las decisiones posteriores.

Enfoque 1: Filtrado directo sin análisis de componentes independientes (ICA)

Primeramente se probó un preprocesamiento basado exclusivamente en filtrado digital sin ICA, ya que el ICA clásico se necesita visualizar para poder identificar los artefactos y eliminarlos. Además, que se requieren conocimientos para poder identificar los artefactos de forma correcta.

Este filtrado digital consistía en aplicar:

- Filtro pasa-banda 1-40 Hz.

- Y se comprobó el uso del filtro *notch* 50 Hz.

Con esta primera implementación se consiguió reducir el ruido de baja frecuencia y la interferencia de la red eléctrica. Sin embargo, los artefactos fisiológicos permanecían en los datos y por lo tanto no permitían que los datos pudiesen ser analizados de forma correcta.

Enfoque 2: *ICLabel*

Se aplicó *ICLabel*, que como se explico en el capitulo de "metodología" de la memoria, es un clasificador automático basado en aprendizaje supervisado [Pion-Tonachini et al., 2019a]. Para su implementación se uso un ejemplo de la documentación *mne.icalabel* (00_iclabel.py), en este ejemplo también se encontraban documentadas los requisitos para el uso correcto de *ICLabel*. En el documento se especifica que *ICLabel* está diseñado

para clasificar componentes independientes obtenidos mediante un algoritmo de descomposición ICA *extended infomax* aplicado a conjuntos de datos de EEG referenciados a un promedio común y filtrados entre 1-100 Hz [Li et al., 2022, Pion-Tonachini et al., 2019b]. Aunque también comenta que es posible ejecutar *ICLabel* en conjuntos de datos que no cumplan esas especificaciones, pero que el rendimiento de la clasificación podría verse afectado negativamente.

Dadas estas recomendaciones, se procedió a que el entrenamiento de ICA cumpliese con esos requisitos. Aunque también se investigó un rango filtrado de frecuencia alternativo, el rango límite del *hardware* de adquisición, con un filtro 1-40 Hz, pero pese a que sí que detectaba artefactos, al entrenar ICA con rango de frecuencia limitado a 40 Hz, el algoritmo *Infomax* no accedía al registro espectral completo de estos artefactos limitando la efectividad de la separación estadística. Y en el caso del filtrado 1-100 Hz, aunque el *hardware* de adquisición está limitado DC-40 Hz, proporciona información espectral ampliada para el entrenamiento. La banda de frecuencia 40-100 Hz contiene ruido (no datos reales adquiridos), pero este ruido preserva la identidad estadística de los componentes originales. Esta información estadística completa permite al algoritmo *Infomax* realizar la separación de componentes de forma más óptima.

Observando la señal con `raw.plot()` se comparó el efecto de las diferentes rangos de frecuencia en el filtrado previo y se observó que la señal obtenida empleando el filtro previo de rango 1-100 Hz generó una señal más limpia.

Otros parámetros empleados en ICA fueron `n_components` que se estableció a `len(ELECTRODES)-1` basandose en la recomendación de la documentación de *MNE-Pyhton* [Developers, 2025c] y de EEGLAB [sccn, sf], estos describen que para datos de rango deficiente, como los datos de EEG después de aplicar una referencia promedio, se recomienda reducir la dimensionalidad en 1 para conseguir un rendimiento óptimo de ICA. La causa de esto es que al calcular la referencia promedio en datos con n canales, el rango de los datos se reduce a *n-1* porque la suma de los potenciales es cero en todos los puntos temporales, de modo que la actividad del último canal es igual a la negativa de la suma de los demás. Por lo que de esta forma se garantiza que ICA extrae toda la varianza disponible en los datos sin sobreajuste, optimizando el rendimiento del algoritmo.

El parámetro `random_state = 97` fija la semilla del generador de números aleatorios de *NumPy* [Developers, 2025c]. La fijación de la semilla permite la repetibilidad de los resultados, es decir, garantiza que múltiples

ejecuciones con los mismos datos de entrada produzcan la misma matriz de componentes.

Una vez completado el entrenamiento de ICA, la matriz de componentes se clasificó automáticamente mediante *ICLabel* en siete categorías: cerebral, muscular, ocular, cardiaco, ruido de línea, ruido de canal y otro. Y se aplicó un criterio de exclusión conservardor, porque además de los componentes etiquetados como cerebrales también se conservaron los etiquetados como otro para no eliminar posible información cerebral que no se clasificó como tal pero tiene elevado potencial [Li et al., 2022, Pion-Tonachini et al., 2019b].

Posteriormente se reconstruyen los datos, pero en vez de sobre los datos filtrados empleados para el entrenamiento, sobre los datos crudos. Y a continuación se realiza el filtrado digital como en el enfoque anterior sin ICA (filtro *notch* y pasabanda).

Obtención de épocas con corrección de línea base

Esta técnica se fundamenta en el principio de que la actividad cerebral evocada en respuesta a un estímulo debe ser cuantificada relativamente a la actividad espontánea precedente, que sirve como referencia o línea base [Community, sf].

En el presente trabajo se aplicó corrección de línea base mediante la sustracción de la media de un intervalo temporal de `T_BASELINE = 0.2` segundos previos al inicio de cada época. Este valor se escogió en base a evidencia científica [Sasatake and Matsushita, 2025] y pruebas con diferentes valores e inspección visual de la señal obtenida.

La aplicación de corrección de línea base beneficia al reducir significativamente la varianza en amplitudes, clarifica los componentes evocados y mejora en detección de respuestas cerebrales auténticas. Aunque no se realiza inspección visual de la señal correspondiente al intervalo de tiempo que se emplea como línea base, dado que se busca una implementación automática del preprocesamiento, es posible que esta contenga artefactos y por lo tanto su uso pueda afectar a la señal correspondiente al evento. A pesar de esto se consideró que su empleo es adecuado dados sus posibles beneficios.

Método de estimación de la densidad espectral de potencia (PSD)

La estimación de la PSD constituye el núcleo del análisis de frecuencia en señales EEG, proporcionando una representación de la distribución de potencia en diferentes bandas de frecuencia, permitiendo caracterizar la actividad oscilatoria cerebral asociada a procesos neurofisiológicos específicos.

La herramienta *MNE-Python* proporciona dos métodos nativos para la estimación de PSD mediante la función `raw.compute_psd()` según su documentación: Welch y Multitaper [Welch, 1967, Slepian, 1978]. Ambos representan enfoques no paramétricos validados en la literatura neurocientífica.

1. Welch: es el método por defecto de *MNE-Python* y proporciona una solución robusta basada en segmentación, ventaneado y promediado de periodogramas. Ventajas:

   - Método por defecto: por lo que es ampliamente utilizado.
   - Robustez estadística: la varianza es reducida al promediar múltiples periodogramas.
   - Simplicidad de implementación: parámetros intuitivos y bien documentados.
   - Precisión suficiente: resolución adecuada para análisis de bandas de frecuencia amplias.

   Como limitación, la resolución frecuencial es ligeramente inferior a multitaper en ciertos contextos.

2. Multitaper: utiliza múltiples funciones de ventana ortogonales (DPSS *tapers*) para calcular periodogramas independientes que luego se promedian. Ventajas:

   - Superior concentración espectral: mejor localización de componentes espectrales localizados.
   - Menor sesgo espectral: reduce artefactos espectrales en comparación con Welch.

   Limitaciones:

   - Complejidad computacional mayor.
   - Parámetros adicionales requieren optimización.
   - Presenta menor documentación que Welch en contextos clínicos.

Se seleccionó el método de Welch por las siguientes razones:

- Contexto de aplicación: el análisis requiere estimaciones de potencia en frecuencia amplias (*delta, theta, alpha, beta, gamma*), donde la concentración espectral ultrafina de multitaper es innecesaria.

- Estándar científico: ampliamente utilizado en estudios de EEG.
- Balance óptimo: representa el mejor compromiso entre precisión, robustez, eficiencia computacional, compatibilidad técnica e implementación práctica para este contexto.

Este método constituye un refinamiento significativo del método de periodograma que aborda la limitación fundamental de alta varianza presente en estimaciones de PSD basadas en transformadas discretas de Fourier simples (o FFT, explicadas en el capítulo "Metodología"de la memoria), proporcionando un balance óptimo entre simplicidad computacional y robustez estadística. La razón fundamental por la que el método de Welch mejora el periodograma clásico es que los datos de EEG son siempre variables en el tiempo. Si se analiza una señal EEG de 30 segundos mediante un periodograma clásico, es altamente improbable que la señal se parezca a una suma perfecta de ondas sinusoidales puras. En realidad, el contenido espectral del EEG cambia constantemente, modificado continuamente por la actividad neuronal subyacente. El método Welch resuelve este problema promediando la transformada de Fourier consecutiva de pequeñas ventanas de la señal, con o sin superposición [Vallat, 2018]. Se requiere que las ventanas sean lo suficientemente largas para lograr resolución razonable en frecuencia, pero lo suficientemente cortas para permitir múltiples segmentos que se promedien, reduciendo así la varianza de manera efectiva. Además, la segmentación con solapamiento aumenta la cantidad de muestras estadísticas disponibles para promediado, mejorando significativamente la estabilidad de la estimación. Por eso se escogieron los siguientes parámetros:

- `n_per_seg = raw.n_times//3`: se usa un tercio de la longitud de los datos para cada segmento.
- `n_overlap = n_per_seg // 2`: se aplica un solapamiento del 50 %, que constituye el estándar en análisis espectral de EEG.

Obteniendo la PSD del promedio de cinco segmentos de esta forma en vez de un único segmento.

Con esta función y método usado, los pasos que se sigue para la obtención de PSD son los siguientes:

1. Segmentación con solapamiento.

2. Ventaneado de los segmentos: después de segmentar los datos, cada segmento individual se multiplica por una función de ventana (*MNE-Python* utiliza por defecto una ventana de *Hamming*), mitigando el fenómeno de dispersión espectral y minimizando pérdida de información mediante el solapamiento entre segmentos.

3. Cálculo de peridiograma y promediado:

   - FFT: se aplica FFT a cada segmento ventaneado, descomponiendo la señal temporal en componentes de frecuencia.
   - Cálculo de magnitud al cuadrado: calcula la magnitud al cuadrado del resultado de la FFT, obteniendo estimaciones de espectro de potencia para cada segmento expresadas en V^2/Hz.
   - Promediado de estimaciones individuales: se promedian todas las estimaciones de potencia espectral obtenidas para cada segmento, reduciendo la varianza respecto a un periodograma clásico e incrementando la estabilidad de la estimación final.

Integración de potencia por banda de frecuencia

La cuantificación de potencia dentro de bandas de frecuencia específicas requiere integración numérica de la densidad espectral de potencia sobre los rangos de frecuencia de interés. El área bajo la curva de PSD en una banda específica representa la potencia total en esa banda de frecuencia.

En el presente trabajo se implementó la regla de Simpson que a diferencia de la regla del trapecio que aproxima la función mediante segmentos lineales (rectas), la regla de Simpson aproxima la función mediante polinomios de segundo grado (parábolas), proporcionando estimaciones significativamente más precisas [Wikipedia, 2025, Vallat, 2018]. La idea es muy simple, se divide el área bajo la curva en varios intervalos y se aproxima cada intervalo mediante polinomios de segundo grado (parábolas), luego se suma el área de todas estas parábolas. En la figura G.1 se puede observar como es su funcionamiento.

Formula matemática [Wikipedia, 2025]: Para n subintervalos equiespaciados (con n par), donde $x_i = a + ih$ y $h = \frac{b-a}{n}$ para $i = 0, 1, \ldots, n$:

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left[f(x_0) + 2\sum_{k=1}^{(n/2)-1} f(x_{2k}) + 4\sum_{k=1}^{n/2} f(x_{2k-1}) + f(x_n)\right] \quad \text{(G.1)}$$

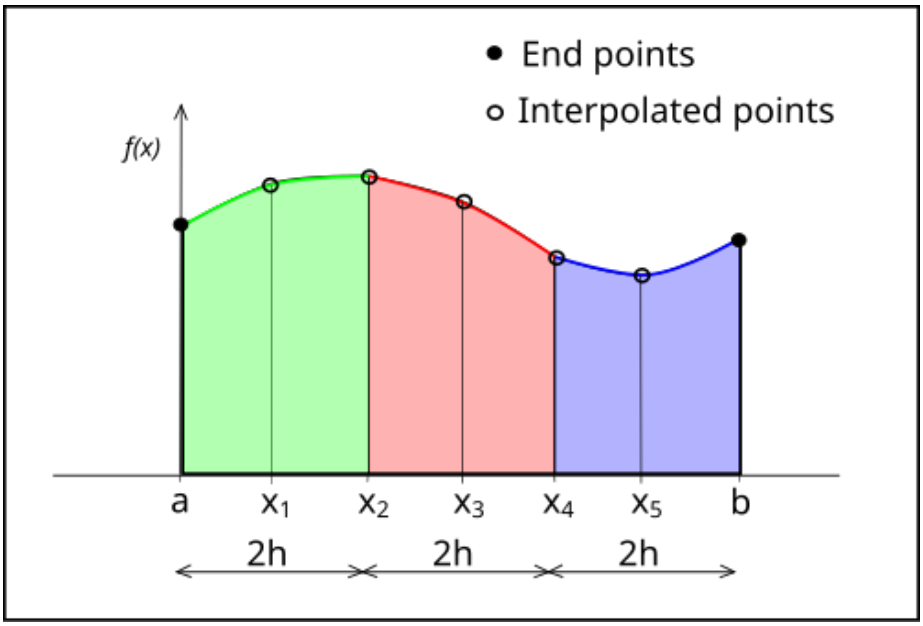


Figura G.1: Ilustración de la regla de Simpson compuesta [Wikipedia, 2025].

El error global en la aproximación está acotado por:

$$E_{\text{máx}} = (b-a) \cdot \frac{h^4}{180} \cdot \max_{a \le \xi \le b} \left| f^{(4)}(\xi) \right| \tag{G.2}$$

En el contexto de este trabajo, la integral de PSD sobre una banda de frecuencia específica se aproxima como:

$$P_{\text{band}} \approx \frac{\Delta f}{3} \left[\text{PSD}(f_0) + 4 \sum_{i \in \text{impar}} \text{PSD}(f_i) + 2 \sum_{i \in \text{par}} \text{PSD}(f_i) + \text{PSD}(f_n) \right] \tag{G.3}$$

donde:

- $\Delta f = f_{\text{freqs}}[1] - f_{\text{freqs}}[0]$ es la resolución frecuencial.
- f_i son los puntos de frecuencia dentro de la banda especificada.
- El coeficiente **4** se aplica a los puntos en posiciones impares: $x_1, x_3, x_5, \ldots, x_{n-1}$.

- El coeficiente **2** se aplica a los puntos en posiciones pares interiores: $x_2, x_4, x_6, \ldots, x_{n-2}$.
- El coeficiente **1** se aplica únicamente a los extremos: x_0 (primer punto) y x_n (último punto).

También se probó el uso de potencia relativa en lugar de reportar la potencia absoluta de la banda, como porcentaje de la potencia total de la señal [Vallat, 2018]. Pero debido a las grandes diferencias de potencias entre bandas, que podía deberse a que no se ha conseguido una limpieza completa, se hacía muy complicado la interpretación de los valores por lo que se descartó esa idea y se usó la potencia absoluta de la banda.

Métodos visualización señal

Para la interpretación y validación de los resultados obtenidos durante el procesamiento de la señal EEG, se generaron cuatro tipos de visualizaciones complementarias. Estas permiten examinar la señal en diferentes dominios (temporal y frecuencial) y perspectivas (global y espacial), facilitando la identificación de artefactos, la evaluación de la calidad de los datos y la comprensión del análisis espectral realizado.

- Gráfica *Raw* EEG: permite visualizar la señal EEG bruta sin procesar en el dominio temporal [Developers, 2025a]. Presenta las siguientes especificaciones técnicas: en el eje X el tiempo en segundos, en el eje Y la amplitud en microvoltios, se representa una traza por electrodo y se escala automáticamente según la amplitud máxima de la señal.
- Mapa topográfico de distribución de potencia [Developers, 2025b]: representa espacialmente la distribución de potencia espectral en cada de frecuencia sobre el cuero cabelludo. Presenta las siguientes especificaciones técnicas: presenta proyección de la vista superior del cuero cabelludo, integra interpolación espacial entre electrodos adyacentes, la escala de color escogida fue “rainbow” que varía azul (potencia baja) al rojo (potencia alta) representada por banda y las unidades son μV^2.
- Gráfico de barras de potencia por banda: permite comparar cuantitativamente la potencia en cada banda de frecuencia para todos los electrodos simultáneamente. Presenta las siguientes especificaciones técnicas: en el eje X los electrodos, en el eje Y la potencia en μV^2, agrupación de 5 barras por electrodo correspondientes a las bandas de frecuencia.

- Gráfica PSD: permite visualizar la densidad espectral de potencia en el dominio frecuencial para todos los electrodos. Presenta las siguientes especificaciones técnicas: en el eje X la frecuencia en Hz, en el eje Y la PSD en V^2/Hz, la escala en X se escogió lineal, la escala Y lineal sin realizar conversión a decibelios (dB) y cada línea en la gráfica corresponde a un electrodo.

Selección de herramienta de visualización: *Streamlit*

En una fase inicial del estudio, el análisis se realizaba mediante un *Jupyter notebook* que importaba las funciones de un archivo *Python*. En dicho *notebook* se había implementado código específico para recorrer todos los participantes, configurar manualmente la ruta de los datos y llamar a las funciones de procesamiento. Aunque este enfoque resultaba funcional, presentaba varias limitaciones prácticas:

- Requería editar el *notebook* para ajustar rutas, lo que hacía el flujo de trabajo más propenso a errores y menos reproducible.
- La salida gráfica se generaba de forma secuencial en una única celda de resultados para todos los participantes, lo que dificultaba la inspección comparada y la localización rápida de una figura concreta.
- La interacción con los datos quedaba restringida al entorno de *Jupyter*, exigiendo cierta familiaridad técnica y limitando su uso por parte de usuarios no expertos.

A la vista de estas limitaciones, se decidió migrar la interacción con el *pipeline* de análisis a una aplicación desarrollada con *Streamlit*. Además, su elección se vio favorecida por la familiaridad previa adquirida en asignaturas del máster, lo que redujo la curva de aprendizaje y facilitó su integración en el flujo de trabajo del estudio. Esta decisión aporta varias ventajas:

- Encapsula el flujo completo en una interfaz web sencilla y accesible desde cualquier navegador sin necesidad de modificar el código.
- Facilita su usabilidad.
- Permite una inspección iterativa y más ordenada de los resultados.

En consecuencia, el uso de *Streamlit* se considera una decisión de diseño adoptada a partir de la experiencia con el enfoque inicial basado en *Jupyter*,

con el objetivo de mejorar la usabilidad, reducir errores de manipulación y facilitar la exploración visual de los resultados.

Apéndice H

Anexo de sostenibilización curricular

H.1. Introducción

Este anexo recoge una reflexión personal sobre cómo los principios de sostenibilidad han influido en el presente trabajo. A través de decisiones técnicas, metodológicas y documentales realizadas, se exploran los aprendizajes adquiridos en materia de sostenibilidad y cómo se han aplicado.

H.2. Análisis de sostenibilidad

- Salud y bienestar: el presente trabajo contribuye potencialmente a la salud y bienestar mediante la investigación sobre el preprocesamiento y procesamiento de señales electroencefalográficas, es decir de la actividad cerebral. El desarrollo de protocolos de preprocesamiento de EEG ayuda a mejorar la compresión de procesos cognitivos y neurofisiológicos. Con potencial en aplicaciones clínicas o diagnosticas más accesibles, a pesar de que el dispositivo EEG empleado en el trabajo está diseñado únicamente para investigación y no para el ámbito clínico-sanitario.

  La selección del dispositivo representa compromiso con sostenibilidad ambiental e investigación descentralizada, su portabilidad y bajo consumo energético lo hacen significativamente más sostenible que infraestructuras clínicas centralizadas. A pesar de requerir una inversión

sustancial, es significativamente más accesible que sistemas clínicos tradicionales.

- Industria, innovación e infraestructuras: la innovación responsable es un pilar clave en este trabajo. El empleo de *software* de código abierto y metodologías reproducibles representan innovación que democratiza acceso a tecnologías de análisis neurocientífico sin depender de pagar licencias. Esta combinación demuestra que innovación de calidad técnica puede ser simultáneamente accesible, verificable y reproducible. Permitiendo que futuros investigadores puedan verificar, cuestionar y mejorar cada paso del análisis.

- Reducción de las desigualdades: el trabajo aborda desigualdades en el acceso a investigación neurocientífica debido a la inversión económica requerida para acceder a un dispositivo EEG. Sin embargo, el empleo de *software* de código abierto junto con documentación detallada garantiza el acceso a la metodología, herramientas y conocimiento de forma libre. Esta transparencia metodológica fomenta el acceso equitativo al conocimiento neurocientífico y reduciendo desigualdades en la investigación neurocientífica. El *software* de código abierto elimina barreras secundarias de licencias de pago, permitiendo que investigadores que cuenten con recursos limitados puedan acceder a esta metodología y puedan trabajar con ella para implementar, mejorar y verificar las metodologías una vez tengan acceso al dispositivo EEG o *datasets* de este. O incluso emplear esta metodología sobre datos de otros dispositivos EEG modificando simplemente algunas de las funciones y parámetros implementados en este trabajo.

- Paz, justicia e instituciones sólidas: el presente trabajo contribuye a este objetivo de desarrollo sostenible mediante el cumplimiento de los derechos humanos fundamentales de todos los participantes. La investigación en neurotecnología que implica la adquisición de datos cerebrales, como en este trabajo, requiere protección especial de privacidad y autonomía individual del participante. En este trabajo se han implementado múltiples métodos de seguridad: todos los participantes deben leerse y firmar el consentimiento informado para poder participar en la adquisición de datos EEG, garantizando que cada persona fue informada sobre la naturaleza del estudio, posibles riesgos, beneficios y derechos, como su voluntad de recibir los resultados obtenidos tras el procesamiento de sus datos; la protección de datos mediante el cumplimiento de regulaciones de privacidad, incluyendo la anonimización de datos, almacenamiento seguro y acceso restringido; y uso

ético de datos asegurando que sus datos y su información obtenida fue utilizada exclusivamente para los propósitos especificados, no siendo comercializada ni compartida sin permiso.

H.3. Conclusión

La realización de este trabajo ha puesto de manifiesto que la sostenibilidad en investigación en el ámbito biomédico no es un aspecto suplementario sino un pilar de la calidad científica. Las diversas elecciones tomadas sobre *hardware*, *software* y metodología representan como el rigor científico aumenta mediante la integración de responsabilidades ambientales, sociales, económicas, institucionales y éticas.

Mi concepción de calidad científica ha evolucionado de una visión unidimensional centrada en el rigor técnico, hacia un enfoque multidimensional que incluye la precisión técnica, la accesibilidad, la verificabilidad, la contribución a la sostenibilidad global y el respeto de los derechos humanos y derechos fundamentales.

Bibliografía

[authors, 2025] authors, M.-P. (2025). Mne-python license. Último acceso: 02 de febrero de 2026.

[Bitbrain, 2024] Bitbrain (2024). *Bitbrain - SennsLite user guide*. Bitbrain. Último acceso: 07 de enero de 2026.

[Bitbrain, sfa] Bitbrain (s.f.a). Diadem | wearable dry-eeg headset. Último acceso: 07 de enero de 2026.

[Bitbrain, sfb] Bitbrain (s.f.b). Export format differences. Último acceso: 07 de enero de 2026.

[Community, sf] Community, T. J. B. (s.f.). Baseline correction. Último acceso: 10 de septiembre de 2025.

[contributors, 2026] contributors, E. (2026). What is eeglab? Último acceso: 08 de enero de 2026.

[contributors, 2025] contributors, M.-P. (2025). Mne-python: Open-source python package for exploring, visualizing, and analyzing human neurophysiological data. Último acceso: 08 de enero de 2026.

[Corporation, 2021] Corporation, M. (2021). Onnx runtime license. Último acceso: 02 de febrero de 2026.

[Corporation, sf] Corporation, M. (s.f.). Download visual studio code. Último acceso: 05 de enero de 2026.

[de Vitoria, 2026] de Vitoria, U. F. (2026). ¿cuánto cobra un ingeniero biomédico en españa? Último acceso: 21 de enero de 2026.

[Delorme, 2026] Delorme, A. (2026). Welcome to the eeglab wiki - eeglab wiki. Último acceso: 08 de enero de 2026.

[Developers, 2025a] Developers, M. (2025a). Mne-python: mne.io.raw. Último acceso: 09 de enero de 2026.

[Developers, 2025b] Developers, M. (2025b). Mne-python: mne.viz.plot_topomap. Último acceso: 09 de enero de 2026.

[Developers, 2025c] Developers, M. (2025c). mne.preprocessing.ica. Último acceso: 08 de enero de 2026.

[Developers, 2024] Developers, N. (2024). Numpy license. Último acceso: 02 de febrero de 2026.

[Developers, 2019] Developers, S. (2019). Streamlit license. Último acceso: 02 de febrero de 2026.

[Developers, 2025d] Developers, S. (2025d). Scipy license. Último acceso: 02 de febrero de 2026.

[Foundation, 2024] Foundation, P. S. (2024). Python license. Último acceso: 02 de febrero de 2026.

[Initiative, sf] Initiative, O. S. (s.f.). The 3-clause bsd license. Último acceso: 02 de febrero de 2026.

[Li et al., 2022] Li, A., Feitelberg, J., Saini, A. P., Höchenberger, R., and Scheltienne, M. (2022). Mne-icalabel: Automatically annotating ica components with iclabel in python. *Journal of Open Source Software*, 7(76):4484.

[Library, 2023] Library, M. P. D. (2023). How to select a license for research software. Último acceso: 02 de febrero de 2026.

[MNE, 2022] MNE (2022). mne-icalabel license. Último acceso: 02 de febrero de 2026.

[Pion-Tonachini et al., 2019a] Pion-Tonachini, L., Kreutz-Delgado, K., and Makeig, S. (2019a). Iclabel: An automated electroencephalographic independent component classifier, dataset, and website. *NeuroImage*, 198:181–197.

[Pion-Tonachini et al., 2019b] Pion-Tonachini, L., Kreutz-Delgado, K., and Makeig, S. (2019b). Iclabel: An automated electroencephalographic independent component classifier, dataset, and website. *NeuroImage*, 198:181–197.

[Sasatake and Matsushita, 2025] Sasatake, Y. and Matsushita, K. (2025). Eeg baseline analysis for effective extraction of p300 event-related potentials for welfare interfaces. *Sensors*, 25(10).

[sccn, sf] sccn (s.f.). Eeglab tutorial: Independent component analysis (runica). Último acceso: 08 de enero de 2026.

[Shure and Mínguez, 2024] Shure, C. and Mínguez, J. (2024). La historia olvidada de las ondas cerebrales alfa. Blog de Bitbrain. Último acceso: 10 de septiembre de 2025.

[Slepian, 1978] Slepian, D. (1978). Prolate spheroidal wave functions, fourier analysis, and uncertainty — v: the discrete case. *The Bell System Technical Journal*, 57(5):1371–1430.

[Team and Team, 2023] Team, I. D. and Team, J. D. (2023). Jupyter notebook license. Último acceso: 02 de febrero de 2026.

[Team, 2016] Team, M. D. (2016). Matplotlib license. Último acceso: 02 de febrero de 2026.

[Team, 2026] Team, P. D. (2026). Pandas license. Último acceso: 02 de febrero de 2026.

[Vallat, 2018] Vallat, R. (2018). Compute the average bandpower of an eeg signal. Último acceso: 10 de septiembre de 2025.

[Webster and Nimunkar, 2020] Webster, J. G. and Nimunkar, A. J. (2020). *Medical Instrumentation: Application and Design*. Wiley, 5th edition.

[Welch, 1967] Welch, P. (1967). 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*, 15(2):70–73.

[Wikipedia, 2025] Wikipedia (2025). Regla de simpson — wikipedia, la enciclopedia libre. [Internet; descargado 17-septiembre-2025].