Escuela Superior Politécnica del Litoral

Facultad de Ingeniería en Mecánica y Ciencias de la Producción

Diseño de un sistema de hardware en lazo con realidad virtual para el análisis de señales de electroencefalograma (EGG)

INGE-2317

Proyecto Integrador

Previo la obtención del Título de:

Ingeniero en Mecatrónica

Presentado por:

Byron Javier De Loor Veloz

Guayaquil - Ecuador

Año: 2023

Agradecimientos

Mi amada familia, su amor constante ha sido mi roca durante este viaje académico. Cada paso en esta tesis ha sido respaldado por su inquebrantable aliento; este logro es nuestro y lo celebro con profunda gratitud hacia cada uno de ustedes.

Al M.Sc. Bryan Puruncajas, al Ph.D Carlos Saldarriaga, e Ing. Andrés Nenger, su orientación y dedicación han sido el motor detrás de este proyecto. Aprecio sinceramente su paciencia y dirección experta; sin ustedes, esta tesis no habría alcanzado estas alturas.

A los chicos de DFM, el trabajo en equipo ha sido esencial. Han sido más que compañeros de investigación, han sido amigos y aliados en cada desafío. Este logro es suyo también, y agradezco la camaradería y sinergia compartida.

Con profunda gratitud,

Byron Javier De Loor Veloz

Declaración Expresa

“Yo Byron Javier De Loor Veloz acuerdo y reconozco que: La titularidad de los derechos patrimoniales de autor (derechos de autor) del proyecto de graduación corresponderá al autor o autores, sin perjuicio de lo cual la ESPOL recibe en este acto una licencia gratuita de plazo indefinido para el uso no comercial y comercial de la obra con facultad de sublicenciar, incluyendo la autorización para su divulgación, así como para la creación y uso de obras derivadas. En el caso de usos comerciales se respetará el porcentaje de participación en beneficios que corresponda a favor del autor o autores. La titularidad total y exclusiva sobre los derechos patrimoniales de patente de invención, modelo de utilidad, diseño industrial, secreto industrial, software o información no divulgada que corresponda o pueda corresponder respecto de cualquier investigación, desarrollo tecnológico o invención realizada por mi/nosotros durante el desarrollo del proyecto de graduación, pertenecerán de forma total, exclusiva e indivisible a la ESPOL, sin perjuicio del porcentaje que me/nos corresponda de los beneficios económicos que la ESPOL reciba por la explotación de mi/nuestra innovación, de ser el caso. En los casos donde la Oficina de Transferencia de Resultados de Investigación (OTRI) de la ESPOL comunique al/los autores que existe una innovación potencialmente patentable sobre los resultados del proyecto de graduación, no se realizará publicación o divulgación alguna, sin la autorización expresa y previa de la ESPOL.

Guayaquil, 7 de febrero del 2024”

De Loor Veloz Byron Javier

Evaluadores

Firmado electrónicamente por:
BRYAN JOAO
PURUNCAJAS MAZA

Bryan Puruncajas, M.Sc.

Profesor de Materia

Firmado electrónicamente por:
CARLOS XAVIER
SALDARRIAGA MERO

Carlos Saldarriaga, Ph.D.

Tutor de proyecto

Resumen

Esta tesis propone el diseño e implementación de un sistema de lazo cerrado que integra la captura, procesamiento y visualización de señales de EEG (Electroencefalograma) en un entorno de realidad virtual (RV). El objetivo principal es proporcionar una plataforma interactiva que permita a los usuarios explorar y comprender sus propias ondas cerebrales de manera inmersiva. La metodología se basa en la adquisición en tiempo real de señales EEG, su procesamiento para extraer características relevantes, y la integración de estas en un entorno virtual tridimensional. El sistema de lazo cerrado permite una retroalimentación en tiempo real, lo que facilita la interacción del usuario con su actividad cerebral. Los resultados preliminares muestran una representación visual efectiva de las señales EEG en RV, proporcionando una herramienta potencialmente valiosa para la autoconciencia y el desarrollo de habilidades de autorregulación cognitiva. Este enfoque innovador contribuye a la convergencia de la neurociencia y la tecnología de realidad virtual, abriendo nuevas perspectivas para aplicaciones en áreas como la mejora del rendimiento cognitivo y la rehabilitación neuropsicológica.

Palabras Clave: EEG, Realidad Virtual, Neurociencia, Visualización en tiempo real

Abstract

This thesis proposes the design and implementation of a closed-loop system that integrates the capture, processing, and visualization of EEG (Electroencephalogram) signals in a virtual reality (VR) environment. The main objective is to provide an interactive platform that allows users to explore and understand their own brain waves in an immersive way. The methodology is based on the real-time acquisition of EEG signals, their processing to extract relevant features, and the integration of these into a three-dimensional virtual environment. The closed-loop system allows real-time feedback, which facilitates user interaction with their brain activity. Preliminary results show an effective visual representation of EEG signals in VR, providing a potentially valuable tool for self-awareness and the development of cognitive self-regulation skills. This innovative approach contributes to the convergence of neuroscience and virtual reality technology, opening new perspectives for applications in areas such as cognitive performance enhancement and neuropsychological rehabilitation.

Keywords: EEG, Virtual Reality, Neuroscience, Real-Time Visualization

Índice general

Abreviaturas

ESPOL	Escuela Superior Politécnica del Litoral
EEG	Electroencefalograma
IIR	Infinite Impulse Response
IFR	Finite Impulse Response
HIL	Hardware In the Loop
RV	Realidad Virtual

Índice de figuras

Índice de tablas

Capítulo 1

1.1 Introducción

Las investigaciones en neurociencia han demostrado consistentemente la importancia del análisis de las señales EEG en la comprensión del cerebro humano. Las señales EEG son esenciales para el estudio de la conectividad cerebral y la identificación de patrones de actividad neuronal. Estos avances han impulsado la investigación en una amplia gama de campos, desde la cognición hasta la psiquiatría, brindando una comprensión cada vez más profunda de la mente humana [1].

- El análisis de las señales EEG ha sido muy importante al momento de intentar comprender el cerebro humano.
- Las señales EEG son esenciales para poder estudiar la conectividad, solo se habla de que es importante pero no se justifica

Además, el análisis de las señales EEG desempeña un papel crucial en el diagnóstico de enfermedades neurológicas, la utilidad de las señales EEG en el diagnóstico temprano de trastornos como el Alzheimer y el Parkinson. Estas aplicaciones médicas resaltan la importancia de acceder a tecnologías de análisis de señales EEG de manera asequible y generalizada [2].

La democratización de la tecnología de análisis de señales EEG es fundamental para desbloquear su potencial en la neurociencia y la atención médica, por ello nace la necesidad de hacer que estas herramientas estén disponibles y asequibles para un rango más amplio de investigadores y profesionales de la salud, con el objetivo de mejorar la calidad de vida de las personas afectadas por enfermedades neurológicas [3].

Este proyecto se enfoca en el diseño de un sistema Hardware in the Loop con Realidad Virtual para el análisis de señales EEG, fusionando tecnología avanzada con la

aspiración de democratizar el acceso a esta herramienta. La combinación de neurociencia, hardware en tiempo real y realidad virtual abre nuevas perspectivas para la investigación y el diagnóstico en neurociencia, con un enfoque en la inclusión económica y social [1].

1.2 Descripción del Problema

El análisis de señales EEG es un proceso complejo que requiere hardware especializado, software de análisis y conocimientos de neurociencia. El hardware EEG y software de análisis de señales son poco accesibles, y la capacitación necesaria para su uso es extensa, además, la interpretación de los datos EEG a menudo requiere un entorno controlado que puede no ser accesible para todos los investigadores o profesionales de la salud [1].

Estas limitaciones pueden dificultar el acceso a esta tecnología para los investigadores y profesionales de la salud que se encuentran en entornos con recursos limitados. Además, pueden dificultar la realización de investigaciones sobre trastornos cerebrales, lo que puede retrasar el desarrollo de nuevos tratamientos [1].

El desarrollo de un sistema HIL con RV aborda estas limitaciones al reducir los costos y hacer que el análisis de señales EEG sea más accesible y efectivo. Un sistema HIL con RV utiliza hardware real con software simulado para crear un entorno virtual en el que se pueden recopilar, acondicionar y visualizar datos EEG. Esto reduce la necesidad de hardware especializado y software de análisis costosos [2].

Además, un sistema HIL con RV permite visualizar las señales del cerebro en tiempo real en un cerebro modelado. Esto puede ayudar a los investigadores y profesionales de la salud a comprender mejor la actividad cerebral y a identificar patrones relevantes [2].

1.3 Justificación del Problema

Este proyecto tiene el potencial de hacer que el análisis de señales EEG sea más accesible y asequible, lo que significa que más personas podrán usar esta tecnología para estudiar el cerebro y diagnosticar trastornos cerebrales, Esto a su vez, podría mejorar la atención médica al ayudar a los médicos a diagnosticar trastornos cerebrales de forma más precisa y oportuna [4].

Además, la capacidad de replicar y comprender el sistema mediante una documentación técnica detallada aumenta su utilidad y relevancia en la comunidad científica y médica. El impacto comercial también es significativo, ya que este sistema podría ser una herramienta valiosa para varios sectores, desde la industria de la salud hasta la educación y la investigación [5].

En última instancia, este proyecto tiene el potencial de generar beneficios económicos y de empleo a nivel local, al tiempo que contribuye al avance en la investigación en neurociencia a nivel global.

1.4 Objetivos

1.4.1 Objetivo general

Desarrollar un sistema hardware in the loop (HIL) con realidad virtual (RV) que permita el análisis de señales EEG de manera más efectiva y accesible, con el potencial de democratizar el análisis de EEG, mejorar la atención médica y promover la investigación científica en neurociencia.

1.4.2 Objetivos específicos

1. Crear un entorno en realidad virtual para la visualización en tiempo real de señales EEG.
2. Desarrollar un sistema de interacción con señales EEG en tiempo real que permita al usuario controlar las señales EEG en un ambiente virtual con un cerebro.
3. Implementar un filtro de señales en el sistema para acondicionar las señales EEG.

1.5 Marco teórico

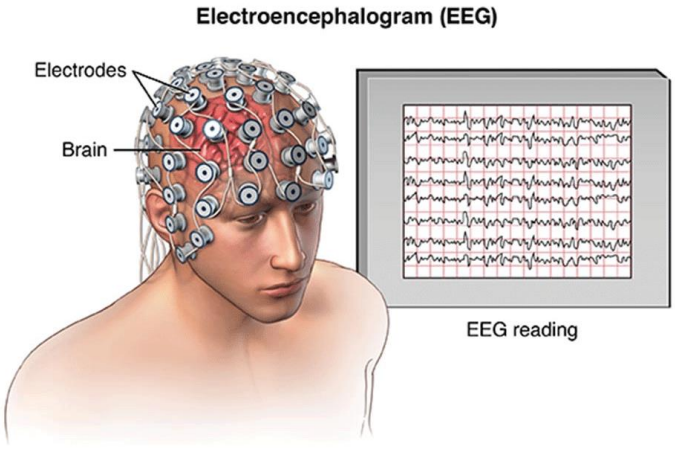
1.5.1 Señales EEG

La electroencefalografía (EEG) es un estudio que mide la actividad eléctrica en el cerebro mediante pequeños discos de metal (electrodos) colocados sobre el cuero cabelludo. Las neuronas cerebrales se comunican a través de impulsos eléctricos y están activas todo el tiempo, incluso mientras duermes [6].

Un electroencefalograma es uno de los estudios principales para diagnosticar la epilepsia. También puede ser útil para diagnosticar o tratar trastornos cerebrales como tumores cerebrales, daños cerebrales por lesiones en la cabeza, disfunciones cerebrales que pueden tener diversas causas (encefalopatía), trastornos del sueño, inflamación del cerebro (encefalitis herpética), accidente cerebrovascular, trastornos del sueño y enfermedad de Creutzfeldt-Jakob [7].

Figura 1

Diagrama de flujo del proceso de envío y recepción de información HIL



Nota. Representación de EEG. [11]

1.5.2 Filtros de Señales digitales EEG

Los filtros de señales digitales EEG se utilizan para eliminar el ruido y otros artefactos de las señales EEG. El ruido puede ser causado por una variedad de factores, como la actividad muscular, la interferencia electromagnética y el movimiento del sujeto. Los artefactos pueden ser causados por factores como la colocación de los electrodos o la actividad del sujeto durante la grabación [9].

Los filtros de señales digitales EEG se pueden clasificar en dos tipos principales: filtros pasa bajos y filtros pasa altos.

Filtros pasa bajos

Los filtros pasa bajos se utilizan para eliminar el ruido de alta frecuencia de las señales EEG. El ruido de alta frecuencia puede ser causado por la actividad muscular, la interferencia electromagnética y el movimiento del sujeto [10].

Los filtros pasa bajos se pueden implementar de varias maneras, incluyendo:

- Filtros FIR: Los filtros FIR son filtros digitales que utilizan un número finito de coeficientes, son generalmente más precisos que los filtros IIR, pero también son más costosos de implementar [10].
- Filtros IIR: Los filtros IIR son filtros digitales que utilizan un número infinito de coeficientes, son generalmente menos precisos que los filtros FIR, pero también son más baratos de implementar [10].

Filtros pasa altos

Los filtros pasa altos se utilizan para eliminar el ruido de baja frecuencia de las señales EEG. El ruido de baja frecuencia puede ser causado por la actividad muscular, la interferencia electromagnética y el movimiento del sujeto. Los filtros pasan altos se pueden implementar de las mismas maneras que los filtros pasa bajos [10].

1.5.3 Análisis de señales EEG

El análisis de señales EEG es el proceso de interpretar los datos EEG. Este proceso implica la extracción de características de las señales EEG, que pueden utilizarse para identificar patrones de actividad cerebral.

Los métodos de análisis de señales EEG se pueden clasificar en tres categorías principales:

- Análisis de frecuencia: Este método divide las señales EEG en componentes de frecuencia, que pueden utilizarse para identificar ritmos cerebrales específicos.
- Análisis de tiempo: Este método examina los cambios en las señales EEG a lo largo del tiempo.

- Análisis de espacio: Este método examina los patrones de actividad cerebral en diferentes regiones del cerebro [9].

1.5.4 Análisis de frecuencia

El análisis de frecuencia es uno de los métodos de análisis de señales EEG más comunes. Este método divide las señales EEG en componentes de frecuencia, que pueden utilizarse para identificar ritmos cerebrales específicos, los ritmos cerebrales más comunes son [10]:

- Ritmo alfa: 8-13 Hz, asociado con el estado de vigilia relajado.
- Ritmo beta: 13-30 Hz, asociado con la atención y la concentración.
- Ritmo gamma: 30-100 Hz, asociado con el procesamiento de información complejo.

1.5.5 Análisis de tiempo

El análisis de tiempo es otro método de análisis de señales EEG común. Este método examina los cambios en las señales EEG a lo largo del tiempo, se puede utilizar para identificar patrones de actividad cerebral asociados con diferentes tareas o estados mentales. Por ejemplo, se puede utilizar el análisis de tiempo para identificar los cambios en la actividad cerebral asociados con la atención, la concentración o la memoria [10].

1.5.6 Análisis de espacio

El análisis de espacio es un método de análisis de señales EEG que examina los patrones de actividad cerebral en diferentes regiones del cerebro, se puede utilizar para identificar regiones del cerebro que están activas durante diferentes tareas o estados

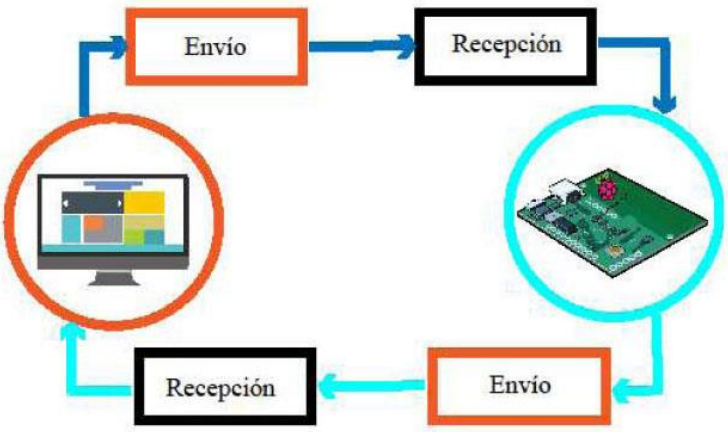
mentales. Por ejemplo, se puede utilizar el análisis de espacio para identificar las regiones del cerebro que están activas durante la visión, el procesamiento del lenguaje o el movimiento [10].

1.5.7 Hardware in the loop (HIL)

Hardware in the Loop (HIL) es una técnica de simulación que utiliza hardware real para interactuar con un sistema simulado. El HIL se utiliza a menudo para probar sistemas de control antes de implementarlos en un sistema real.

Figura 2

Diagrama de flujo del proceso de envío y recepción de información HIL



Nota. Esquema de proceso de Lazo Cerrado [11]

En un sistema HIL, el hardware real se conecta a un simulador que representa el sistema que se está probando. El simulador genera señales de entrada que se aplican al hardware real. El hardware real genera señales de salida que se miden por el simulador [12].

El HIL tiene varias ventajas sobre otras técnicas de simulación:

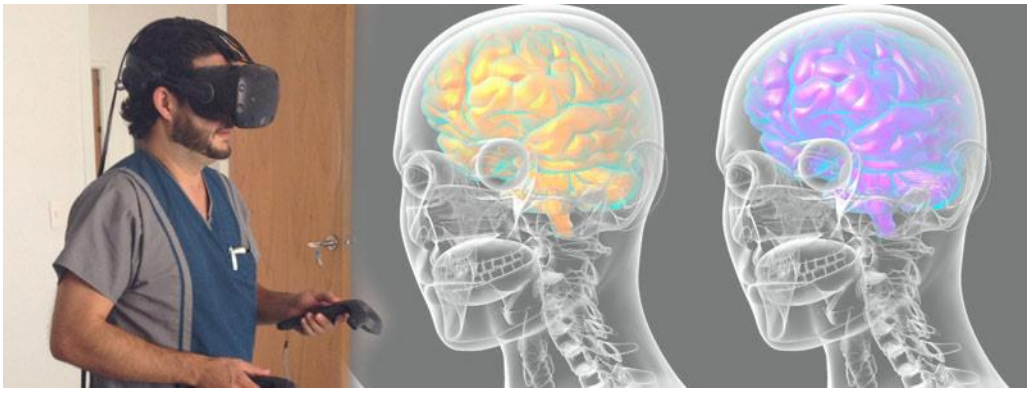
- Realismo: El hardware real proporciona un entorno de prueba más realista que un simulador basado en software.
- Eficiencia: El HIL permite probar sistemas de control en tiempo real, lo que puede ahorrar tiempo y dinero.
- Flexibilidad: El HIL puede utilizarse para probar sistemas de control de cualquier complejidad.

1.5.8 Realidad Virtual

Realidad Virtual (RV) es una tecnología que crea la ilusión de estar presente en un entorno artificial. La RV se crea mediante el uso de una combinación de hardware y software para engañar al cerebro para que perciba el entorno simulado como real [12].

Figura 3

Realidad virtual para fines educacionales



Nota. Profesional de la salud utilizando sistema de Realidad Virtual [13]

La RV consta de los siguientes componentes principales:

- Dispositivo de visualización: El dispositivo de visualización es el componente que muestra el entorno simulado al usuario. Los dispositivos de visualización comunes para la RV incluyen cascos, gafas y pantallas envolventes.
- Controladores: Los controladores son dispositivos que permiten al usuario interactuar con el entorno simulado. Los controladores comunes para la RV incluyen controladores de movimiento y controladores de gestos.
- Software de RV: El software de RV es el software que crea el entorno simulado. El software de RV genera imágenes, sonido y otras sensaciones que el usuario percibe como reales [14].

1.5.9 Neurociencia

La neurociencia es el estudio del sistema nervioso, que incluye el cerebro, la médula espinal y los nervios periféricos. La neurociencia es un campo de investigación interdisciplinario que abarca una amplia gama de disciplinas, que incluyen la biología, la química, la física, la psicología y la medicina [15].

1.5.10 Estado del Arte

El artículo [16] se centra en el monitoreo de conductores utilizando EEG en un simulador de manejo en realidad virtual. Los autores presentan un sistema que utiliza un casco de realidad virtual para crear un entorno de conducción virtual realista. Este entorno permite estudiar cómo la actividad cerebral de los conductores cambia según las condiciones del tráfico y el comportamiento de otros conductores. Esta investigación

proporciona una valiosa perspectiva sobre la interacción entre la actividad cerebral y la conducción en un entorno seguro y controlado.

Por otro lado, el artículo [17] se enfoca en el uso de una interfaz cerebro-computadora (BCI) basada en EEG para la rehabilitación en entornos virtuales. Los autores presentan un sistema en el que las señales EEG se utilizan para controlar un avatar en un entorno virtual. Esta tecnología se aplica en la rehabilitación de pacientes con parálisis cerebral, lo que sugiere el potencial de las BCI basadas en EEG en el campo de la rehabilitación motora.

Finalmente, el artículo [18] aborda la clasificación de la enfermedad de Parkinson utilizando un sistema de realidad virtual con hardware en el ciclo de control. Este sistema utiliza señales EEG para identificar patrones de actividad cerebral asociados con la enfermedad, lo que resulta en una clasificación precisa de los pacientes con Parkinson. Este enfoque ofrece un método prometedor para el diagnóstico y seguimiento de la enfermedad, destacando la utilidad de la EEG en aplicaciones clínicas relacionadas con la enfermedad de Parkinson.

Capítulo 2

2. METODOLOGÍA.

En este capítulo se describen el proceso de selección entre las distintas alternativas de solución para el problema a resolver, así como también el diseño conceptual de la solución, la metodología de diseño usada y la descripción detallada del diseño del producto.

2.1 Proceso de selección de la alternativa de solución

Con el fin de resolver la problemática se plantearon 2 posibles soluciones, cada una de estas soluciones tiene un diferente acercamiento al diseño y creación de sistemas de Realidad Virtual (VR):

a. Alternativa 1:

Desarrollar un sistema hardware in the loop (HIL) con realidad virtual (RV) basado en la web, que permita el análisis de señales EEG utilizando tecnológicas como HMTL5 o WebGL.

b. Alternativa 2:

Desarrollar un sistema hardware in the loop (HIL) con realidad virtual (RV) que permita el análisis de señales EEG, utilizando tecnológicas para crear ambientes de realidad virtual como Unity o Unreal Engine 5.

Con el fin de determinar la alternativa de solución más apropiada se plantearon los criterios más importantes para el problema que se plantea resolver. En la tabla 2.1 se resume las ponderaciones por criterios.

Tabla 2.1

Criterios de selección

Criterio	**Ranking**	**Peso relativo**	**Porcentaje**
Rendimiento	1	10	33%
Flexibilidad	2	8	27%
Control	3	5	17%
Dificultad	4	4	13%
Coste	5	3	10%
TOTAL		30	100%

Se detalla una descripción breve de cada criterio de selección:

- **Rendimiento:** Velocidad, Fluidez y tiempos de reacción al realizar la interacción con el ambiente.
- **Flexibilidad:** Flexibilidad que se tiene sobre el diseño del sistema de realidad virtual (RV)
- **Control:** Control sobre el diseño y respuestas del sistema de lazo cerrado.
- **Dificultad:** Se considera experiencia previa del investigador, en caso de ser alta implica una gran cantidad de aprendizaje necesario como requisito para usar dicha solución.
- **Coste:** Valor necesario para costear recursos necesarios para implementar la solución como: Software, Licencias, etc. después de realizar en análisis tomando como

referencia los criterios planteados en la tabla 2.1, se obtuvo que la alternativa 1 es la óptima para resolver la problemática. En la tabla 2.2 se muestran las ponderaciones de cada criterio, respectivas por cada solución.

Tabla 2.2

Matriz de decisión

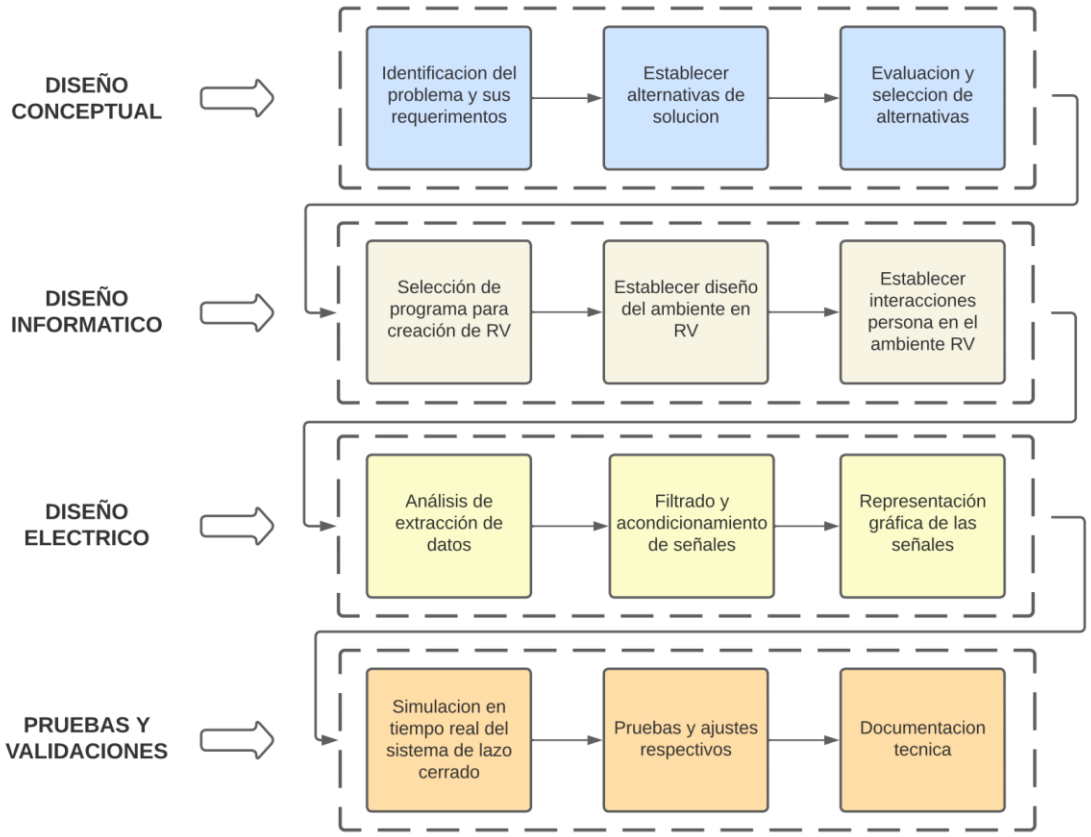
	CRITERIO 1	**CRITERIO 2**	**CRITERIO 3**	**CRITERIO 4**	**CRITERIO 5**	**Total**
Descripción	**Rendimiento**	**Flexibilidad**	**Control**	**Dificultad**	**Coste**	
Peso	10	8	5	4	3	30
% Peso	33%	27%	17%	13%	10%	100%
OPCIONES	**Calificación**	**Calificación**	**Calificación**	**Calificación**	**Calificación**	
Alternativa 1	10	10	8	8	10	9.4
Alternativa 2	8	7	5	4	7	6.6

2.2 Proceso de diseño

Mediante las reuniones suscitadas con el cliente, siguiendo los requerimientos y objetivos establecidos, se definió la siguiente metodología de diseño.

Figura 4

Metodología de diseño

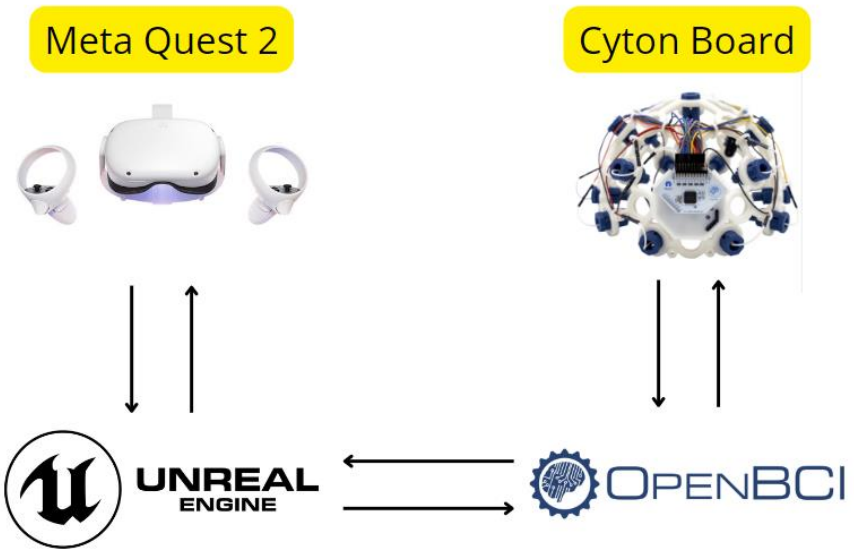


2.3 Diseño Conceptual

En la figura 5 se muestra el sistema de la solución planteada a una escala macro, mostrando los elementos principales que forman parte de este proceso.

Figura 5

Diseño conceptual



Para determinar este diseño conceptual fue necesario establecer las delimitaciones del problema, así como también sus requerimientos principales, los cuales son:

- El sistema debe recopilar y mostrar visualmente las señales EEG de forma precisa.
- El sistema debe proporcionar al usuario una experiencia inmersiva en realidad virtual.
- El sistema debe permitir el análisis de señales EEG para fines de diagnóstico y tratamiento.

Posteriormente se tomó como referencia el estado del arte en sistemas de realidad virtual lo cual ayudo a identificar las técnicas y tecnologías que fueron más adecuadas para el proyecto.

2.4 Diseño Informático

En la sección de metodología, se detalla el enfoque estratégico seguido para la materialización del proyecto de realidad virtual. Inicialmente, se aborda la crítica tarea de seleccionar el software más adecuado para el diseño del ambiente inmersivo.

A través de una exhaustiva comparativa entre los tres principales programas utilizados en la creación de entornos de realidad virtual: Unreal Engine 5, Unity y Blender, se evaluaron aspectos cruciales como intuición, flexibilidad, compatibilidad, rendimiento, realismo e interactividad. La conclusión resultante de este análisis, detallada en la Tabla 2.3, respalda la elección del Unreal Engine 5 como la plataforma óptima, destacando sus atributos excepcionales.

2.5 Selección de Software

Para determinar el software a utilizar para diseñar el ambiente de realidad virtual se realizó una comparativa entre los 3 softwares más populares para diseñar ambientes de realidad virtual, considerando las características más importantes para los diseños de estos ambientes inmersivos.

Tomando la información de la tabla 2.3 llegamos a la conclusión de que Unreal Engine 5 es la mejor opción para el desarrollo de ambientes de realidad virtual inmersivos. Ofrece una interfaz intuitiva, flexibilidad, compatibilidad, rendimiento, realismo e interactividad excepcionales. Unity es también una buena opción, pero puede no ser tan bueno como Unreal Engine 5 en términos de rendimiento, realismo e interactividad. Blender es una opción menos adecuada para el desarrollo de experiencias de realidad virtual, ya que

es más difícil de aprender y usar, y no ofrece un rendimiento tan bueno como Unreal Engine 5 o Unity.

Tabla 2.3

Tabla comparativa de softwares para ambientes de realidad virtual

Características	**Unreal Engine 5**	**Unity**	**Blender**
Intuición	Fácil de aprender y usar	Relativamente fácil de aprender	Difícil de aprender
Flexibilidad	Alto nivel de flexibilidad	Alto nivel de flexibilidad	Alto nivel de flexibilidad
Compatibilidad	Amplia compatibilidad con dispositivos y plataformas	Amplia compatibilidad con dispositivos y plataformas	Menor compatibilidad con dispositivos de realidad virtual
Rendimiento	Excepcional	Bueno	No tan eficiente
Realismo	Excepcional	Bueno	Requiere más trabajo
Interactividad	Amplia gama de herramientas	Amplia gama de herramientas	Buena gama de herramientas

2.6 Diseño de ambiente de realidad Virtual

El análisis de las necesidades es un paso crucial al diseñar un ambiente virtual, ya que se busca identificar las demandas de los usuarios. Para esto, se recurrió a una entrevista con el cliente con preguntas clave, para determinar los objetivos que los usuarios buscan alcanzar, qué información o recursos requieren y cómo interactúan entre sí y con el ambiente virtual. Este proceso garantizo que el diseño final satisfaga estas necesidades.

El desarrollo implico la creación del ambiente virtual utilizando los elementos definidos previamente, empleando diversas tecnologías como Unreal Engine 5, openBCI. Para el diseño de ambientes virtuales, se emplearon métodos como el enfoque centrado en el usuario, que se orientaron a satisfacer sus necesidades y deseos, garantizando así la facilidad de uso y la adecuación del ambiente virtual a sus requerimientos.

2.7 Diseño Eléctrico

En la elaboración del diseño eléctrico para el entorno de Realidad Virtual (RV) en el presente trabajo de tesis, se enfrentaron desafíos inherentes a la gestión de datos provenientes de diversas fuentes. La elección de incorporar un factor de decimación se fundamenta en la necesidad de adecuar la tasa de entrada de datos a la capacidad de procesamiento del programa utilizado para ejecutar el ambiente de RV. La tasa de entrada, que supera la capacidad promedio del programa, requiere un enfoque estratégico para garantizar la eficiencia del sistema y la interpretación precisa de los datos generados.

En este contexto, la presencia potencial de señales de interferencia provenientes de dispositivos electrónicos o ruidos ambientales agrega un componente crítico al diseño eléctrico. Para abordar este desafío, se ha decidido implementar un filtro de señales digitales,

específicamente el filtro de Butterworth. La elección de este filtro se sustenta en sus características notables, como su respuesta de amplitud plana en la banda de paso, su capacidad para proporcionar una transición suave entre las frecuencias de paso y de rechazo, así como su facilidad de implementación. Estas ventajas hacen del filtro de Butterworth una elección adecuada para mitigar interferencias y garantizar la integridad de las señales EEG capturadas en el entorno de Realidad Virtual.

2.8 Pruebas y validaciones

Se realizaron pruebas exhaustivas con señales EEG en vivo para verificar la precisión del sistema de control digital. La validación se centró en asegurar la fidelidad de la visualización en el ambiente virtual con los datos reales.

Se identificaron mejoras basadas en los resultados de las pruebas y se llevaron a cabo ajustes necesarios para mejorar el sistema de control y la interfaz entre el hardware y la realidad virtual, mejorando así la experiencia del usuario y la precisión del análisis EEG.

Finalmente, se elaboró una documentación detallada del diseño, implementación y funcionamiento del sistema completo, la cual estuvo acompañada de manuales y guías de uso. Este paquete se entregó a los usuarios o entidades pertinentes.

Capítulo 3

3. RESULTADOS Y ANÁLISIS

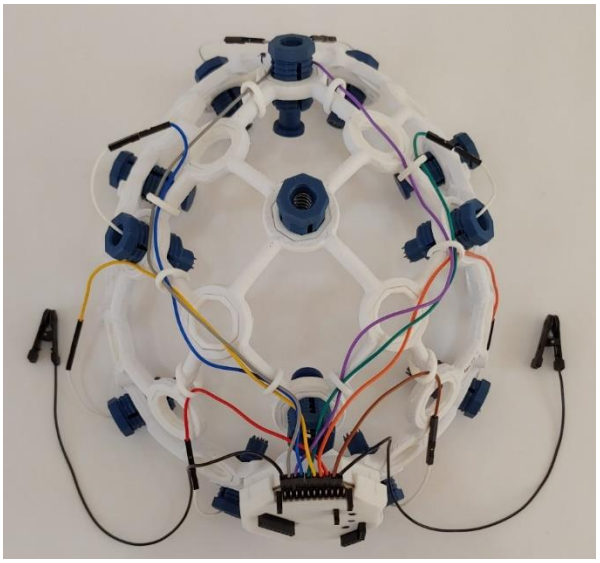
Esta sección de análisis y resultados abarca las cuatro etapas cruciales del proyecto. Inicia con la Recolección de Información mediante OpenBCI, donde se explora la captura de datos neurofisiológicos. A continuación, se detalla el Acondicionamiento de Señales, destacando el uso de C# y Python. La Representación de Información en Unity y la Interacción en tiempo real con Gafas de Realidad Virtual se examinan para comprender la visualización y la experiencia inmersiva. Se incluirán los códigos de programación empleados en cada fase. Además, se realizará un análisis pormenorizado de los costos asociados con el desarrollo del proyecto, proporcionando una evaluación financiera completa. Este análisis económico contribuirá a la comprensión de la viabilidad y el impacto global del proyecto.

3.1 Recolección de Datos EEG

La metodología de captura de señales EEG en este proyecto se inicia con el uso del dispositivo OpenBCI para la adquisición de datos cerebrales. El proceso comienza cuando la persona coloca el dispositivo de captura en su cabeza, empleando específicamente el controlador Cython Board. Este dispositivo se conecta al ordenador a través de la tecnología Bluetooth mediante un receptor USB que actúa como puente de señal.

Figura 6

Dispositivo de recolección de señales EEG



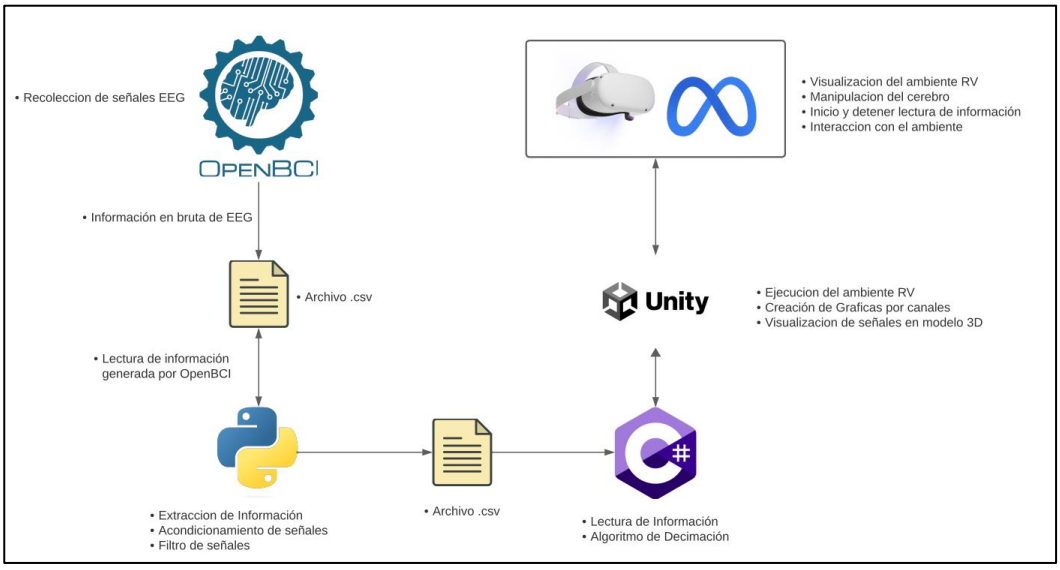
La conexión entre el dispositivo y el ordenador se establece utilizando el programa OpenBCI GUI, que posibilita la visualización y monitorización de las señales EEG en tiempo real. Además, para facilitar la conexión, se emplea un dispositivo Bluetooth que actúa como enlace entre el controlador Cython Board y el sistema, permitiendo la transmisión de señales con una frecuencia de 255Hz hacia el dispositivo de recepción. Esto garantiza una captura de datos en alta resolución y una comunicación eficiente entre el Cython Board y el programa OpenBCI GUI.

Para facilitar la exportación y manejo de los datos, el programa OpenBCI GUI ofrece diversas opciones de protocolos de transmisión, como UDP, TCP, entre otros. Sin embargo, en este caso, se optó por la exportación directa de datos hacia un archivo de texto. Esta elección se basó en consideraciones de interconectividad y minimización de pérdida de datos, ya que la exportación a un archivo de texto proporciona una solución robusta y eficiente para garantizar la integridad de la información capturada.

La utilización de un archivo de texto como formato de exportación ofrece flexibilidad en el manejo posterior de los datos, permitiendo su análisis en profundidad y su integración con otras herramientas y sistemas. Esta metodología proporciona una base sólida para el proceso de captura de señales EEG, asegurando la eficiencia y la integridad de los datos capturados en el contexto del proyecto.

Figura 7

Flujo final de información durante ejecucion



3.2 Acondicionamiento de Señales

La sección de Acondicionamiento de Señales juega un papel fundamental en el proyecto al asegurar la fiabilidad y la interpretación precisa de las señales EEG capturadas por el Cython Board. Para obtener información sobre los valores máximos y mínimos de las señales de entrada, se recurrió a la documentación oficial del Cython Board, donde se determinó que estos oscilan entre un máximo de 187,500 y un mínimo de -187,500.

Basándonos en estos datos y considerando la tasa de ingreso de 255Hz proporcionada por el Cython Board, se implementó un proceso de normalización. Este paso es esencial para traducir los valores brutos de las señales a una escala de intensidad estandarizada, variando de 0 a 255. Este enfoque facilita la visualización de la intensidad de la señal en cada sección del cerebro virtual, permitiendo una representación gráfica más intuitiva y comprensible.

Además, se incorporó un filtro de señales de 60Hz en Python al momento de la extracción de información para mitigar el ruido inherente generado por elementos electrónicos, lo cual contribuye significativamente a mejorar la calidad de las señales procesadas. Este paso es fundamental para garantizar una interpretación más precisa y libre de interferencias no deseadas.

Finalmente, para optimizar el rendimiento del sistema en Unity, que opera a una salida de aproximadamente 40-50Hz, se implementó un algoritmo de decimación. Este algoritmo se encarga de reducir la tasa de muestreo de la señal, manteniendo al mismo tiempo la mayor cantidad posible de información relevante. Esta estrategia es esencial para asegurar una eficiente representación y procesamiento de las señales EEG en el entorno de realidad virtual, contribuyendo así al éxito y la coherencia del proyecto en su conjunto.

3.3 Integraciones de software

La sección de Integración de Software constituye un elemento clave en la implementación del proyecto, permitiendo la coherencia y la sincronización efectiva entre diversas herramientas. En su conjunto, se emplearon varios programas para llevar a cabo las diferentes fases del proceso. En primera instancia, Apenca GUI se utiliza para la lectura

de información proveniente del Cython Board, facilitando así la captura y monitorización en tiempo real de las señales EEG.

La información capturada por OpenBCI GUI se extrae y procesa utilizando Python, donde se lleva a cabo la extracción y filtrado de señales. Python desempeña un papel fundamental en la optimización y preparación de los datos, asegurando su coherencia y eficacia para la siguiente etapa del proyecto.

La fase de Representación de Información se lleva a cabo en Unity, un entorno de desarrollo de realidad virtual. Unity permite la creación de un ambiente tridimensional inmersivo para visualizar las señales EEG procesadas. Para facilitar la integración entre Python y Unity, se emplea C# dentro del entorno de Unity, actuando como puente para utilizar la información procesada por Python y lograr una representación gráfica coherente y envolvente.

Además, la conexión entre las gafas de realidad virtual Meta Quest 2 y el ordenador se realiza a través de la aplicación Oculus. Oculus desempeña un papel crucial en la interacción en tiempo real, posibilitando que el usuario se sumerja en el entorno de realidad virtual creado en Unity y tenga una experiencia participativa y envolvente.

3.4 Diseño del Ambiente de Realidad Virtual (RV) en Unity

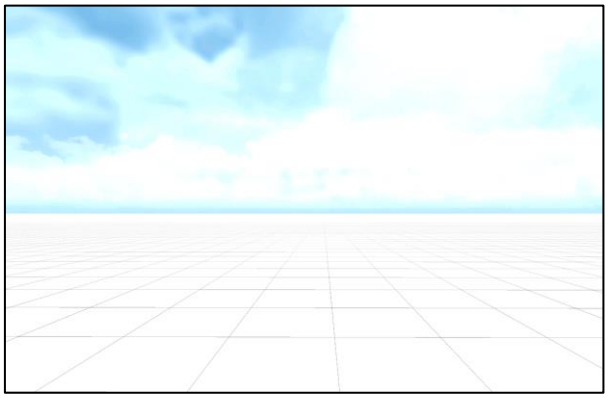
El entorno de realidad virtual en Unity fue creado para cumplir con los requisitos del usuario, resultando en las siguientes características destacadas:

3.4.1 Fondo Blanco Resaltante:

Fondo blanco que resalta el cerebro 3D, gráficas y menús para una experiencia visual efectiva.

Figura 8

Fondo del ambiente de RV



3.4.2 Menú Interactivo:

Implementación de un menú interactivo que permite al usuario manipular objetos, iniciar/detener la lectura de datos y reiniciar la posición del cerebro.

Figura 9

Menú principal de la interfaz del usuario

Figura 10

Menú mostrado inicio de lectura de datos

Figura 11

Menú mostrado al seleccionar Explotado en el menú Ejecución

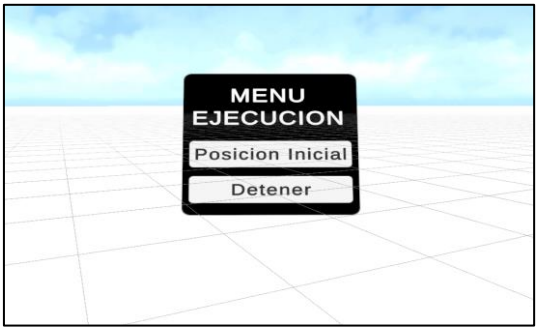


Figura 12

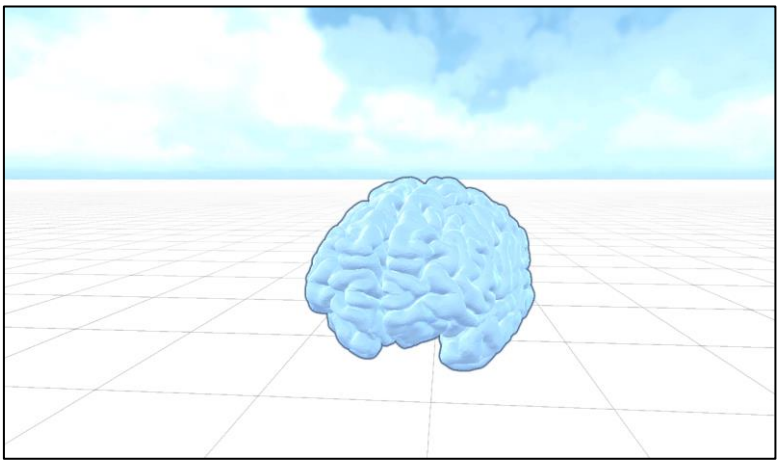
Menú con información de contribuyentes del proyecto

3.4.3 Cerebro Modelado 3D:

Representación explotada del cerebro en 3D, dividido en lóbulos frontal, parietal, occipital y temporal para una visión detallada.

Figura 13

Modelo de cerebro 3D en ambiente de realidad virtual

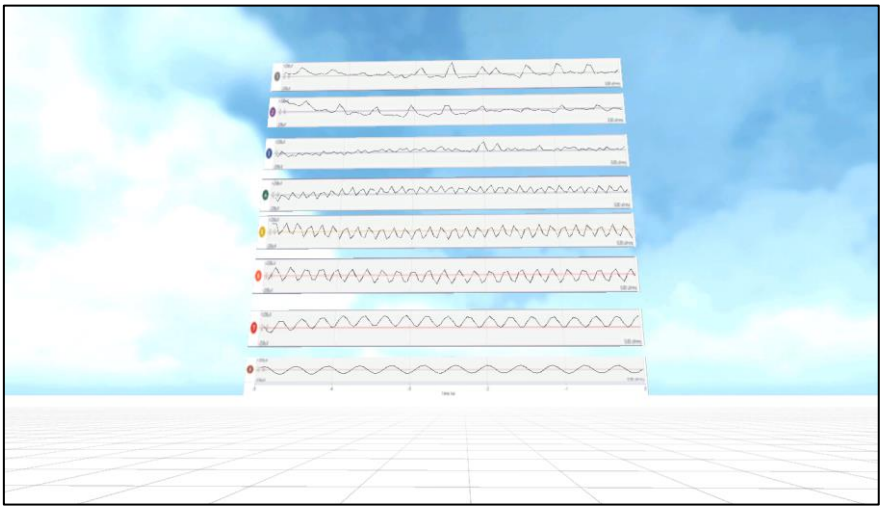


3.4.4 Gráficas de Visualización:

Inclusión de gráficas normalizadas para cada canal del dispositivo Cython Board, con la opción de configurar la cantidad de puntos visualizados en la gráfica.

Figura 14

Visualización de señales en tiempo real en ambiente de realidad virtual



3.4.5 Interactividad del Usuario:

Posibilidad para el usuario de interactuar con todos los elementos del entorno, incluyendo manipulación de objetos y exploración detallada del cerebro.

Figura 15

Vista explotada del cerebro modelado en ambiente de realidad virtual

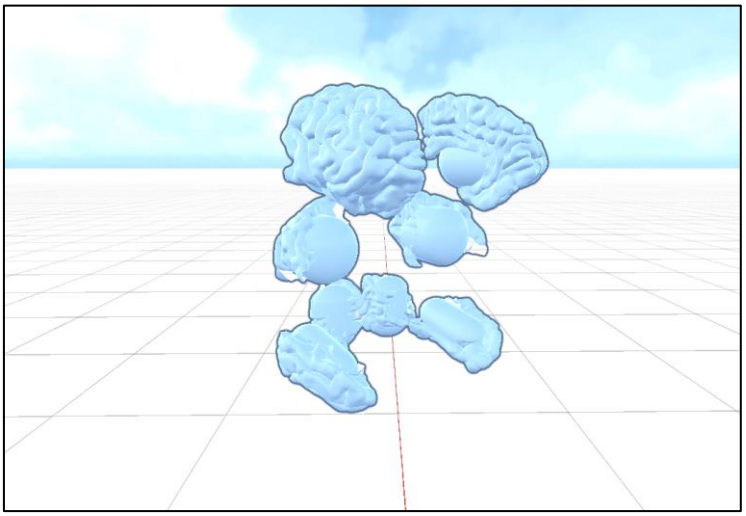
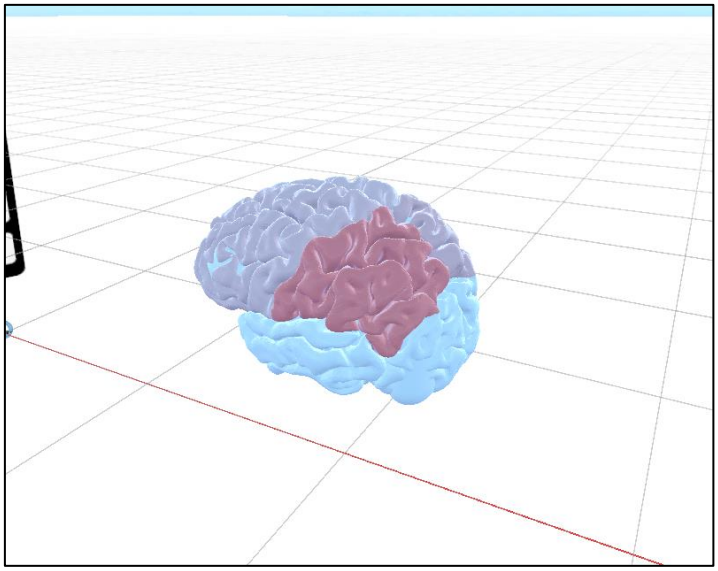


Figura 16

Iluminación de señales de modelo 3D



Estas características buscan ofrecer una experiencia inmersiva y fácil de usar, integrando la visualización de datos EEG de manera efectiva en el entorno de realidad virtual para una comprensión más profunda.

3.5 Costos y Recursos

En el desarrollo del proyecto se han considerado diversos rubros de gastos, cada uno desempeñando un papel crucial en la implementación exitosa del entorno de realidad virtual. Las siguientes tablas detallan los costos asociados con los componentes electrónicos, software y otros gastos, proporcionando una visión integral de la asignación de recursos para este proyecto innovador.

La tabla 3.1 muestra de los rubros perteneciente a la categoría de electrónicos, se observa una asignación financiera considerable de $655 USD. Este desembolso se destina principalmente a la adquisición de componentes críticos como la Cython Board Open BCI y los Electrodos para Cython Board, cuya función es primordial para la captura precisa de datos EEG. Asimismo, se contemplan los Jumpers para la configuración del hardware y las Gafas Meta 2 Quest, fundamentales para proporcionar una experiencia inmersiva. Esta categoría refleja la priorización de inversiones en elementos esenciales para garantizar la funcionalidad óptima del sistema de realidad virtual.

Tabla 3.1

Rubros componentes electrónicos

Categoría	Componente	Precio (USD)
Electrónicos	Cython Board Open BCI	$300
	Electrodos para Cython Board	$50
	Jumpers	$5
	Gafas Meta 2 Quest	$300

En la tabla 3.2, se detallan los costos asociados con los componentes de software. La estrategia de minimización de costos es evidente, ya que se opta por soluciones de software de calidad sin costo. La Licencia de Unity, Licencia de Blender, Modelo Cerebro 3D y OpenBCI GUI se obtienen sin gastos adicionales, destacando la eficiencia en el uso de recursos y herramientas accesibles para el desarrollo del proyecto.

Tabla 3.2

Rubros componentes de software

Categoría	Componente	Precio (USD)
Software	Licencia de Unity	$0
	Licencia de Blender	$0
	Modelo Cerebro 3D	$0
	OpenBCI GUI	$0

La tabla 3.3 aborda los otros gastos que no se clasifican como electrónicos ni de software. La impresión de Ultra Cortex Mark IV se ejecuta con un costo de $20, proporcionando componentes físicos esenciales para la experiencia de realidad virtual. Por otro lado, la Mano de Obra se valora en $500, reconociendo el tiempo y esfuerzo dedicados al proyecto. Estos gastos adicionales contribuyen a la creación completa del entorno virtual.

Tabla 3.3

Rubros de otros gastos

Categoría	Componente	Precio (USD)
Otros	Impresión de Ultra Cortex Mark IV	$20
	Mano de Obra	$500

Finalmente, en la tabla 3.4, se detalla la sumatoria de total de precios, dando como resultado que la implementación de este proyecto tiene un costo de $1169

Tabla 3.4

Rubros Totales

Categoría	Precio (USD)
Electrónicos	$655
Software	$0
Otros	$520
Total	$1,175

Capítulo 4

4. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

La confirmación de una conexión exitosa entre las Gafas Meta 2 Quest, Unity, Cython Board y OpenBCI destaca el logro de una interoperabilidad eficaz en el sistema de lazo cerrado. Este éxito subraya la capacidad de estas tecnologías para trabajar de manera integrada, consolidando la viabilidad y funcionalidad del sistema en su conjunto.

La incorporación de la visualización de señales EEG en un entorno de realidad virtual no solo añade riqueza y profundidad a la experiencia del usuario, sino que también simplifica la interpretación y comprensión de estas señales cerebrales complejas. Al sumergir al usuario en un entorno tridimensional, se logra no solo una experiencia más envolvente, sino también una mayor claridad en la interpretación de los datos EEG, mejorando así la utilidad de esta tecnología en el ámbito neurocientífico.

La perspectiva de una solución de realidad virtual de código abierto presenta un potencial transformador al abrir las puertas a la democratización de herramientas neurocientíficas avanzadas. Al adoptar un enfoque de código abierto, esta solución se vuelve más accesible y modificable para una audiencia más amplia, lo que tiene el poder de extender el alcance de las herramientas neurocientíficas. Esto, a su vez, hace que la interpretación de señales EEG sea más asequible y disponible para diversos usuarios, contribuyendo a la democratización de los avances en la neurociencia.

4.2 Recomendaciones

Se sugiere llevar a cabo una validación exhaustiva del programa seleccionado para el desarrollo del entorno virtual. Este proceso garantizará la integridad y funcionalidad del

software, asegurándose de que cumple con los requisitos y estándares necesarios. La validación completa proporcionará una base sólida para la construcción de un entorno virtual confiable y eficaz.

Se recomienda evitar el uso del mismo programa en el que se ejecuta el entorno para llevar a cabo el acondicionamiento de señales. Esta medida busca prevenir posibles interferencias o conflictos entre las funciones del programa de desarrollo y las tareas asociadas al acondicionamiento de señales. Al separar estas actividades, se contribuye a mantener la estabilidad y consistencia en ambas etapas del proceso, minimizando potenciales inconvenientes y optimizando el rendimiento general del sistema.

Es aconsejable seleccionar con precisión el filtro de señales digitales de acuerdo con las frecuencias que se desean eliminar. Esta recomendación subraya la importancia de una elección cuidadosa de los filtros, ya que la correcta configuración puede determinar la eficacia en la eliminación de frecuencias no deseadas. Al ajustar los filtros de manera adecuada, se optimiza la calidad de las señales, contribuyendo así a resultados más precisos y confiables en la interpretación de datos EEG.

Se sugiere realizar una validación del estado de los electrodos para garantizar la ausencia de interferencias o pérdida de señales. Esta recomendación cobra relevancia debido a que la integridad de los electrodos es esencial para obtener lecturas precisas de las señales cerebrales. La validación del estado de los electrodos permite identificar y abordar posibles problemas, como interferencias externas o conexiones defectuosas, asegurando así la fiabilidad de los datos recopilados. La correcta funcionalidad de los electrodos es crucial para la calidad global del sistema, destacando la necesidad de una validación rigurosa.

REFERENCIAS

[1] Nunez, P. L., & Srinivasan, R. (2006). Electrical fields of the brain: The neurophysics of EEG, MEG, and fMRI (2nd ed.). Oxford University Press.

[2] Babiloni, F., & Niedermeyer, E. (2007). EEG analysis: A practical guide. Elsevier.

[3] Müller, K.-R., & Leistritz, L. (2019). EEG analysis: A review of current trends and challenges. Frontiers in Neuroscience, 13, 139.

[4] Gómez Figueroa, L. J. (2016). Análisis de señales EEG para detección de eventos oculares, musculares y cognitivos. Tesis (Master), E.T.S.I. Industriales (UPM).

[5] Özkan, H., Doğan, M., Kantar, Y., Akşahin, A., & Erdamar, M. (2016). Analysis of EEG signals using artificial neural networks for epilepsy diagnosis. Biomedical Signal Processing and Control, 22, 107-114.

[6] Acharya, R., & Tham, C. K. (2018). EEG signal analysis: A review. Biomedical Signal Processing and Control, 42, 26-39.

[7] Acharya, U. R., Srinivasan, R., & Acharya, R. (2011). Electroencefalograma (EEG): una herramienta esencial para el diagnóstico y tratamiento de trastornos neurológicos. Biomedical Signal Processing and Control, 6(1), 1-19.

[8] Saminu, Sani & Xu, Guizhi & Shuai, Zhang & Abd El Kader, Isselmou & Halilu Jabire, Adamu & Karaye, Ibrahim & Ahmad, Isah & Abdulkarim, A. (2021). Electroencephalogram (EEG) Based Imagined Speech Decoding and Recognition. Journal of Applied Materials and Technology. 2. 74-84. 10.31258/Jamt.2.2.74-84.

[9] Acharya, U. R., & Srinivasan, R. (2011). Filtros de señales digitales EEG: una revisión. Biomedical Signal Processing and Control, 6(1), 20-36.

[10] Srinivasan, R., & Acharya, U. R. (2011). Filtros FIR y IIR para señales EEG. Journal of Biomedical Signal Processing and Control, 6(2), 257-270.

[11] Hernández, C., Vélez, D., & Isaza, J. (2018). "Design of a virtual sensor test platform for the glucose-insulin system of ICU patients using the HIL technique." IEEE Transactions on Biomedical Engineering, 23(2), 61-75.

[12] Zhang, X., & Zhang, Q. (2018). Hardware-in-the-Loop Simulation for Control Systems. IEEE Transactions on Control Systems Technology, 26(5), 1813-1824. doi: 10.1109/TCST.2017.2761038.

[13] [Secretaría de Ciencia, Tecnología e Innovación de México, 2023]. Robot humanoide. [Imagen de un robot humanoide]. Ciencia MX. https://www.cienciamx.com/images/aic/1-HEAD_revirtjj0618.jpg

[14] Liu, X., Xu, Y., & Wang, S. (2019). A Survey of Hardware-in-the-Loop Simulation for Automotive Control Systems. IEEE Transactions on Intelligent Transportation Systems, 20(1), 201-216. doi: 10.1109/TITS.2018.2811378.

[15] Kandel, E. R., Schwartz, J. H., Jessell, T. M., Siegelbaum, S. A., & Hudspeth, A. J. (2013). Neurociencia. 5.ª edición. McGraw-Hill Education.

[16] A. M. Khan, et al., "EEG-based driver monitoring in a hardware-in-the-loop virtual reality driving simulator," in IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2022.

[17] J. Wang, et al., "EEG-based brain-computer interface for virtual reality rehabilitation," in IEEE Transactions on Biomedical Engineering, 2022.

[18] M. A. A. Al-Shuwairi, et al., "EEG-based classification of Parkinson's disease using a hardware-in-the-loop virtual reality system," in IEEE Journal of Biomedical and Health Informatics, 2022.

APÉNDICES

APÉNDICE A

Diseño del algoritmo de acondicionamiento y filtrado de datos

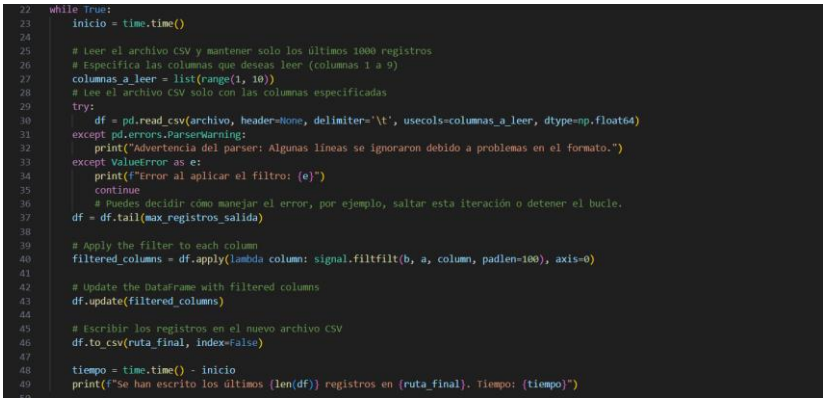
A continuación, se muestra el algoritmo de lectura de datos EEG generados por OpenBCI, se toma los últimos 1000 registros que corresponden a la información de los últimos 2 segundos de lectura de datos EEG. El algoritmo tiene la función de acondicionar los datos utilizando un filtro de paso de banda y guardando los últimos registros filtrados en un nuevo archivo CSV. El bucle se ejecuta de manera continua, debido a que el archivo de entrada se actualiza continuamente con nuevos datos del EEG, y el proceso de filtrado y escritura en el archivo de salida se realiza de manera constante.

Figura 17

Variables utilizadas en algoritmo de Acondicionamiento y filtrado de información.

Figura 18

Loop utilizado para lectura y filtrado de datos en tiempo real.



```
while True:
    inicio = time.time()

    # Leer el archivo CSV y mantener solo los últimos 1000 registros
    # Especifica las columnas que deseas leer (columnas 1 a 9)
    columnas_a_leer = list(range(1, 10))
    # Lee el archivo CSV solo con las columnas especificadas
    try:
        df = pd.read_csv(archivo, header=None, delimiter='\t', usecols=columnas_a_leer, dtype=np.float64)
    except pd.errors.ParserWarning:
        print("Advertencia del parser: Algunas líneas se ignoraron debido a problemas en el formato.")
    except ValueError as e:
        print(f"Error al aplicar el filtro: {e}")
        continue
        # Puedes decidir cómo manejar el error, por ejemplo, saltar esta iteración o detener el bucle.
    df = df.tail(max_registros_salida)

    # Apply the filter to each column
    filtered_columns = df.apply(lambda column: signal.filtfilt(b, a, column, padlen=100), axis=0)

    # Update the DataFrame with filtered columns
    df.update(filtered_columns)

    # Escribir los registros en el nuevo archivo CSV
    df.to_csv(ruta_final, index=False)

    tiempo = time.time() - inicio
    print(f"Se han escrito los últimos {len(df)} registros en {ruta_final}. Tiempo: {tiempo}")
```

APÉNDICE B

Diseño de código de programación de ambiente de Realidad Virtual

Este código Unity ofrece una implementación para la visualización interactiva de datos EEG mediante objetos tridimensionales que simbolizan los sensores cerebrales. Durante la exploración detallada, se abordarán las funciones clave que desempeñan roles fundamentales en la aplicación.

Figura 19

Librerías utilizadas ejecutar el ambiente de Realidad Virtual

```
using System.Collections;
using System.Collections.Generic;
using System.Security.Cryptography;
using Unity.XR.OpenVR;
using UnityEngine;
using UnityEngine.XR;
using UnityEngine.XR.OpenXR.NativeTypes;
using System;
using System.IO;
using System.Linq;
```

Figura 20

Variables que forman parte del código del ambiente de Realidad Virtual

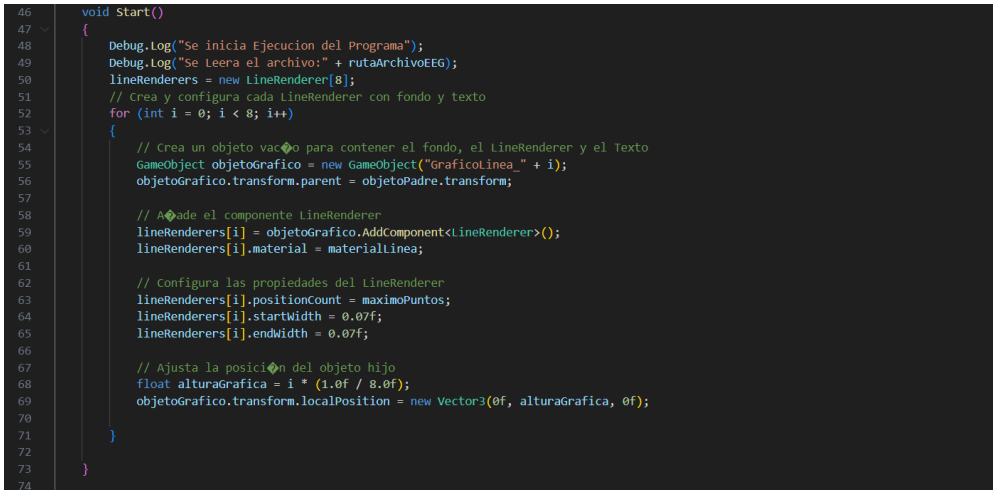
```
public class Menu_Interactivo : MonoBehaviour
{

    //VARIABLES RELACIONADAS A LECTURA DE INFORMACION
    public string rutaArchivoEEG = "C:/Users/byron/OneDrive/Documents/DatosEEG.csv";    //ARCHIVO DE LECTURA DE TEXTO
    private string[,] datosCSV;     //ARREGLO 2D de lectura de informacion
    //private float valorMin = -187500;   //VALOR MINIMO DE ENTRADA EN CYTHON BOARD
    //private float valorMax = 187500;    //VALOR MAXIMO DE ENTRADA EN CYTHON BOARD
    private float valorMin = -108;   //VALOR MINIMO DE ENTRADA EN CYTHON BOARD
    private float valorMax = 436;    //VALOR MAXIMO DE ENTRADA EN CYTHON BOARD
    Color colorInicial = new Color(0.87f, 0.87f, 0.87f);    //COLOR INICIAL DEL CEREBRO


    //PARTES DEL CEREBRO
    public GameObject[] objetosSensores;  // Arreglo de objetos que representan los sensores
    public GameObject[] capsulas;
    //BOTONES DEL MENU PRINCIPAL
    public GameObject BotonIniciar;     //INICIA EL PROCESO DE LECTURA
    public GameObject BotonDetener;     //DETIENE EL PROCESO DE LECTURA

    public int tasaEntrada = 250;   // VALOR EN Hz Proveniente del CYTON BOARD OPEN BCI
    public int tasaSalida = 45;     // FrameRate Promedio de salida de Unity

    //GRAFICOS DE LINEA
    public int maximoPuntos = 40;
    public float escalaY = 4.0f;
    public Material materialLinea;
    public Material materialFondo;
    public GameObject objetoPadre;
    private LineRenderer[] lineRenderers;
    private bool activarVisualizacion = false;
    // Start is called before the first frame update
```

La función Start() es llamada al inicio del programa en Unity. Su propósito es configurar y crear los elementos esenciales para la visualización de gráficos. En este caso, se inicializan y

configuran los objetos LineRenderer que representarán los datos gráficos en el espacio tridimensional de Unity.

Figura 21

Funcion que se ejecuta al principio de la ejecucion del ambiente

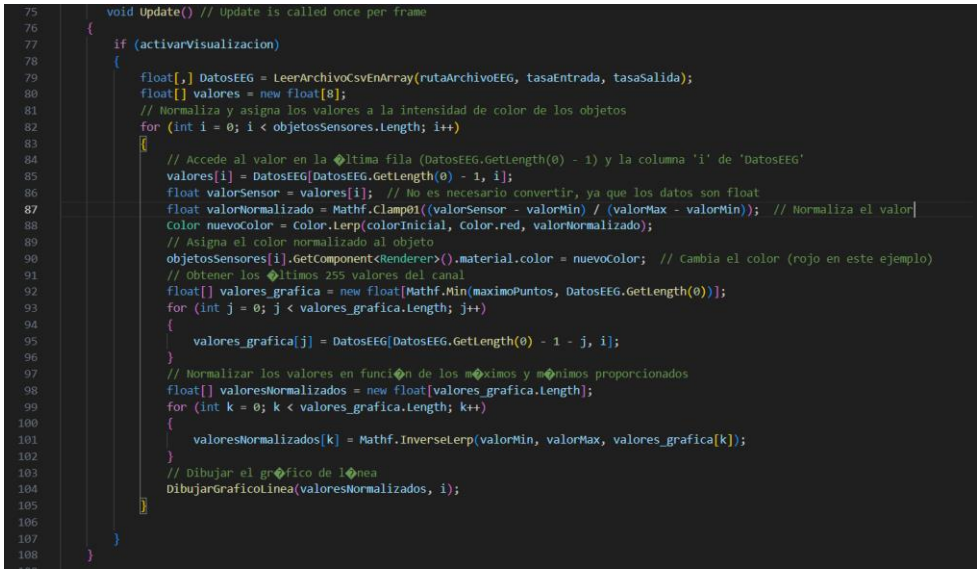


```
46  void Start()
47  {
48      Debug.Log("Se inicia Ejecucion del Programa");
49      Debug.Log("Se Leera el archivo:" + rutaArchivoEEG);
50      lineRenderers = new LineRenderer[8];
51      // Crea y configura cada LineRenderer con fondo y texto
52      for (int i = 0; i < 8; i++)
53      {
54          // Crea un objeto vac□o para contener el fondo, el LineRenderer y el Texto
55          GameObject objetoGrafico = new GameObject("GraficoLinea_" + i);
56          objetoGrafico.transform.parent = objetoPadre.transform;
57
58          // A□ade el componente LineRenderer
59          lineRenderers[i] = objetoGrafico.AddComponent<LineRenderer>();
60          lineRenderers[i].material = materialLinea;
61
62          // Configura las propiedades del LineRenderer
63          lineRenderers[i].positionCount = maximoPuntos;
64          lineRenderers[i].startWidth = 0.07f;
65          lineRenderers[i].endWidth = 0.07f;
66
67          // Ajusta la posici□n del objeto hijo
68          float alturaGrafica = i * (1.0f / 8.0f);
69          objetoGrafico.transform.localPosition = new Vector3(0f, alturaGrafica, 0f);
70
71      }
72
73  }
74
```

La función Update() se ejecuta en cada fotograma y tiene la responsabilidad de actualizar la visualización de datos en tiempo real en Unity. Dentro de esta función, se obtienen los datos del archivo EEG y se actualizan los colores y gráficos de línea en función de esos datos.

Figura 22

Funcion que se ejecuta en cada fotograma durante la ejecución



```
    void Update() // Update is called once per frame
    {
        if (activarVisualizacion)
        {
            float[,] DatosEEG = LeerArchivoCsvEnArray(rutaArchivoEEG, tasaEntrada, tasaSalida);
            float[] valores = new float[8];
            // Normaliza y asigna los valores a la intensidad de color de los objetos
            for (int i = 0; i < objetosSensores.Length; i++)
            {
                // Accede al valor en la última fila (DatosEEG.GetLength(0) - 1) y la columna 'i' de 'DatosEEG'
                valores[i] = DatosEEG[DatosEEG.GetLength(0) - 1, i];
                float valorSensor = valores[i];  // No es necesario convertir, ya que los datos son float
                float valorNormalizado = Mathf.Clamp01((valorSensor - valorMin) / (valorMax - valorMin));  // Normaliza el valor
                Color nuevoColor = Color.Lerp(colorInicial, Color.red, valorNormalizado);
                // Asigna el color normalizado al objeto
                objetosSensores[i].GetComponent<Renderer>().material.color = nuevoColor;  // Cambia el color (rojo en este ejemplo)
                // Obtener los últimos 255 valores del canal
                float[] valores_grafica = new float[Mathf.Min(maximoPuntos, DatosEEG.GetLength(0))];
                for (int j = 0; j < valores_grafica.Length; j++)
                {
                    valores_grafica[j] = DatosEEG[DatosEEG.GetLength(0) - 1 - j, i];
                }
                // Normalizar los valores en función de los máximos y mínimos proporcionados
                float[] valoresNormalizados = new float[valores_grafica.Length];
                for (int k = 0; k < valores_grafica.Length; k++)
                {
                    valoresNormalizados[k] = Mathf.InverseLerp(valorMin, valorMax, valores_grafica[k]);
                }
                // Dibujar el gráfico de línea
                DibujarGraficoLinea(valoresNormalizados, i);
            }

        }
    }
```

La función IniciarLectura() activa la visualización de datos del archivo EEG. Al ser llamada, permite que la actualización en tiempo real de la información comience, afectando la representación visual de los sensores cerebrales en Unity.

La función FinalizarLectura() detiene la visualización de datos del archivo EEG. Al llamar a esta función, se interrumpe la actualización en tiempo real de la información y se detiene la modificación de la representación visual de los sensores cerebrales en Unity.

Figura 23

Funciones de Inicio y Finalización de lectura de datos EEG

```
    //Metodo para iniciar la lectura del archivo
    public void IniciarLectura()
    {
        Debug.Log("INICIA LECTURA");
        activarVisualizacion = true;
    }

    //Metodo para detener la lectura del archivo
    public void FinalizarLectura()
    {
        Debug.Log("FINALIZA LECTURA");
        activarVisualizacion = false;
    }
```

La función ExplotarCerebro() ajusta las posiciones de los objetos que representan los sensores cerebrales en Unity, creando una apariencia de "explosión". Esta función se utiliza con fines visuales y altera las posiciones relativas de los objetos.

Figura 24

Funcion de vista explotada del cerebro

```
public void ExplotarCerebro()
{
    //Lobulos frontales
    Transform objetoTransform0_F_I = objetosSensores[0].GetComponent<Transform>();
    Transform objetoTransform1_F_D = objetosSensores[1].GetComponent<Transform>();
    //Lobulos Limbico
    Transform objetoTransform2_L_I = objetosSensores[2].GetComponent<Transform>();
    Transform objetoTransform3_L_D = objetosSensores[3].GetComponent<Transform>();
    //Lobulos Parietal
    Transform objetoTransform4_P_I = objetosSensores[4].GetComponent<Transform>();
    Transform objetoTransform5_P_D = objetosSensores[5].GetComponent<Transform>();
    //Lobulos Occipital
    Transform objetoTransform6_O_I = objetosSensores[6].GetComponent<Transform>();
    Transform objetoTransform7_O_D = objetosSensores[7].GetComponent<Transform>();


    objetoTransform0_F_I.position = new Vector3(0.25f, 1.1f, 1.6f);
    objetoTransform1_F_D.position = new Vector3(-0.25f, 1.1f, 1.6f);

    objetoTransform2_L_I.position = new Vector3(0.2f, 0.4f, 1.6f);
    objetoTransform3_L_D.position = new Vector3(-0.2f, 0.4f, 1.6f);

    objetoTransform4_P_I.position = new Vector3(0.2f, 0.7f, 1.6f);
    objetoTransform5_P_D.position = new Vector3(-0.2f, 0.7f, 1.6f);

    objetoTransform6_O_I.position = new Vector3(0.05f, 0.5f, 1.6f);
    objetoTransform7_O_D.position = new Vector3(-0.05f, 0.5f, 1.6f);
}
```

La función RegresarCerebro() restaura las posiciones originales de los objetos que representan los sensores cerebrales en Unity. Contrariamente a ExplotarCerebro(), esta función busca devolver los elementos a sus ubicaciones iniciales para mantener la coherencia visual.

Figura 25

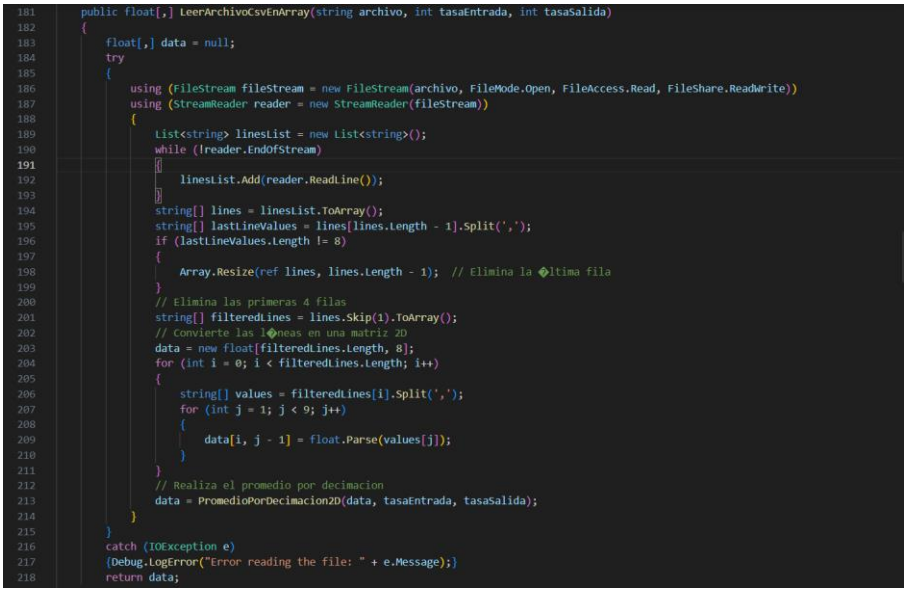
Funcion RegresarCerebro

```
153     public void RegresarCerebro()
154     {
155         //Lobulos frontales
156         Transform objetoTransform0_F_I = objetosSensores[0].GetComponent<Transform>();
157         Transform objetoTransform1_F_D = objetosSensores[1].GetComponent<Transform>();
158         //Lobulos Limbico
159         Transform objetoTransform2_L_I = objetosSensores[2].GetComponent<Transform>();
160         Transform objetoTransform3_L_D = objetosSensores[3].GetComponent<Transform>();
161         //Lobulos Parietal
162         Transform objetoTransform4_P_I = objetosSensores[4].GetComponent<Transform>();
163         Transform objetoTransform5_P_D = objetosSensores[5].GetComponent<Transform>();
164         //Lobulos Occipital
165         Transform objetoTransform6_O_I = objetosSensores[6].GetComponent<Transform>();
166         Transform objetoTransform7_O_D = objetosSensores[7].GetComponent<Transform>();
167 
168         objetoTransform0_F_I.position = new Vector3(0f, 0.7f, 1.6f);
169         objetoTransform1_F_D.position = new Vector3(-0f, 0.7f, 1.6f);
170 
171         objetoTransform2_L_I.position = new Vector3(0f, 0.7f, 1.6f);
172         objetoTransform3_L_D.position = new Vector3(-0f, 0.7f, 1.6f);
173 
174         objetoTransform4_P_I.position = new Vector3(0f, 0.7f, 1.6f);
175         objetoTransform5_P_D.position = new Vector3(-0f, 0.7f, 1.6f);
176 
177         objetoTransform6_O_I.position = new Vector3(0f, 0.7f, 1.6f);
178         objetoTransform7_O_D.position = new Vector3(-0f, 0.7f, 1.6f);
179     }
180 
```

La función LeerArchivoCsvEnArray() toma la ruta de un archivo CSV, así como tasas de entrada y salida, y devuelve una matriz 2D de datos EEG normalizados. Esta función realiza la lectura del archivo, elimina datos no deseados y aplica un proceso de normalización y promediado por decimación.

Figura 26

Funcion para lectura del archivo CSV generado por Python

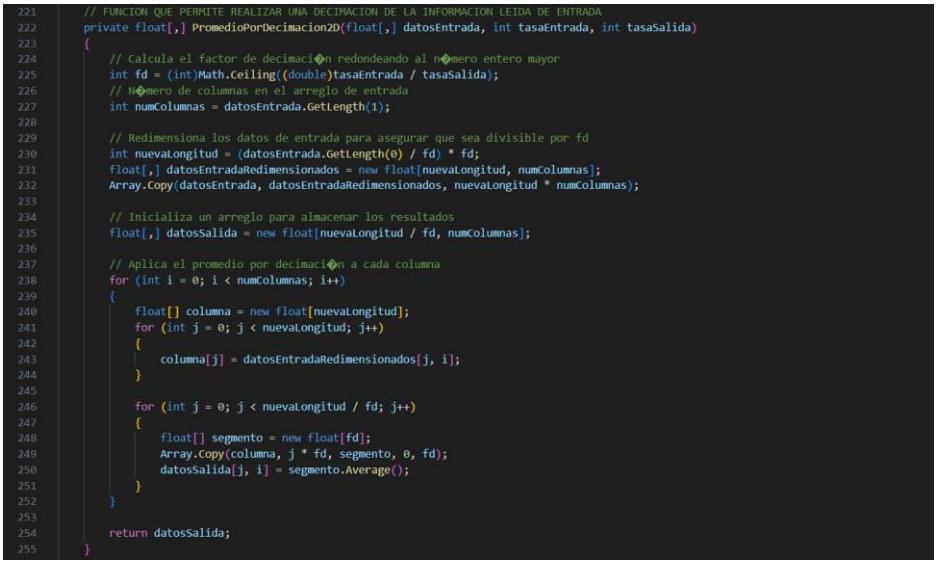


```
public float[,] LeerArchivoCsvEnArray(string archivo, int tasaEntrada, int tasaSalida)
{
    float[,] data = null;
    try
    {
        using (FileStream fileStream = new FileStream(archivo, FileMode.Open, FileAccess.Read, FileShare.ReadWrite))
        using (StreamReader reader = new StreamReader(fileStream))
        {
            List<string> linesList = new List<string>();
            while (!reader.EndOfStream)
            {
                linesList.Add(reader.ReadLine());
            }
            string[] lines = linesList.ToArray();
            string[] lastLineValues = lines[lines.Length - 1].Split(',');
            if (lastLineValues.Length != 8)
            {
                Array.Resize(ref lines, lines.Length - 1);  // Elimina la ?ltima fila
            }
            // Elimina las primeras 4 filas
            string[] filteredLines = lines.Skip(1).ToArray();
            // Convierte las l?neas en una matriz 2D
            data = new float[filteredLines.Length, 8];
            for (int i = 0; i < filteredLines.Length; i++)
            {
                string[] values = filteredLines[i].Split(',');
                for (int j = 1; j < 9; j++)
                {
                    data[i, j - 1] = float.Parse(values[j]);
                }
            }
            // Realiza el promedio por decimacion
            data = PromedioPorDecimacion2D(data, tasaEntrada, tasaSalida);
        }
    }
    catch (IOException e)
    {Debug.LogError("Error reading the file: " + e.Message);}
    return data;
```

La función PromedioPorDecimacion2D() realiza la operación de promedio por decimación en una matriz 2D de datos de entrada. Esta función ajusta la tasa de muestreo para que coincida con la tasa de salida deseada, garantizando una representación más eficiente de los datos en Unity.

Figura 27

Funcion que implementa algoritmo de decimación

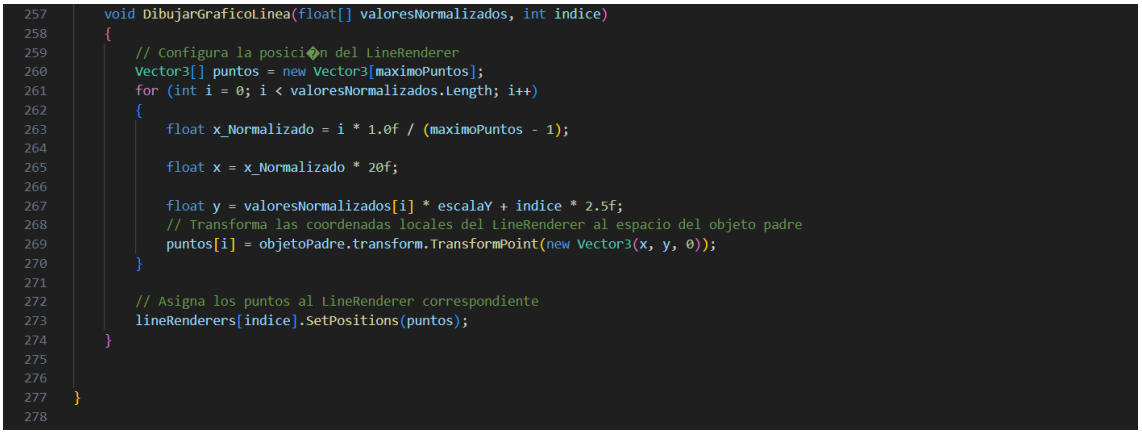


```
// FUNCION QUE PERMITE REALIZAR UNA DECIMACION DE LA INFORMACION LEIDA DE ENTRADA
private float[,] PromedioPorDecimacion2D(float[,] datosEntrada, int tasaEntrada, int tasaSalida)
{
    // Calcula el factor de decimaci?n redondeando al n?mero entero mayor
    int fd = (int)Math.Ceiling((double)tasaEntrada / tasaSalida);
    // N?mero de columnas en el arreglo de entrada
    int numColumnas = datosEntrada.GetLength(1);

    // Redimensiona los datos de entrada para asegurar que sea divisible por fd
    int nuevaLongitud = (datosEntrada.GetLength(0) / fd) * fd;
    float[,] datosEntradaRedimensionados = new float[nuevaLongitud, numColumnas];
    Array.Copy(datosEntrada, datosEntradaRedimensionados, nuevaLongitud * numColumnas);

    // Inicializa un arreglo para almacenar los resultados
    float[,] datosSalida = new float[nuevaLongitud / fd, numColumnas];

    // Aplica el promedio por decimaci?n a cada columna
    for (int i = 0; i < numColumnas; i++)
    {
        float[] columna = new float[nuevaLongitud];
        for (int j = 0; j < nuevaLongitud; j++)
        {
            columna[j] = datosEntradaRedimensionados[j, i];
        }

        for (int j = 0; j < nuevaLongitud / fd; j++)
        {
            float[] segmento = new float[fd];
            Array.Copy(columna, j * fd, segmento, 0, fd);
            datosSalida[j, i] = segmento.Average();
        }
    }

    return datosSalida;
}
```

La función DibujarGraficoLinea() crea un gráfico de línea en Unity utilizando los valores normalizados proporcionados y el índice que representa el sensor al que pertenecen los datos. Esta función trabaja en conjunto con las bibliotecas gráficas de Unity para visualizar de manera efectiva la información en el espacio tridimensional.

Figura 28

Funcion que grafica los valores de las señales de entrada



```
    void DibujarGraficoLinea(float[] valoresNormalizados, int indice)
    {
        // Configura la posici?n del LineRenderer
        Vector3[] puntos = new Vector3[maximoPuntos];
        for (int i = 0; i < valoresNormalizados.Length; i++)
        {
            float x_Normalizado = i * 1.0f / (maximoPuntos - 1);

            float x = x_Normalizado * 20f;

            float y = valoresNormalizados[i] * escalaY + indice * 2.5f;
            // Transforma las coordenadas locales del LineRenderer al espacio del objeto padre
            puntos[i] = objetoPadre.transform.TransformPoint(new Vector3(x, y, 0));
        }

        // Asigna los puntos al LineRenderer correspondiente
        lineRenderers[indice].SetPositions(puntos);
    }


}

```