Universidad Zaragoza

Trabajo Fin de Grado

Ingeniería Informática

# Análisis de correlatos motores en entornos de realidad virtual

# Analysis of motor-related brain correlates in virtual reality environments


Autor

Irene Sánchez Montejo

Director

Pablo Urcola Irache

Ponente

Ana Cristina Murillo Arnal



ESCUELA DE INGENIERÍA Y ARQUITECTURA

2019

# AGRADECIMIENTOS

En primer lugar, me gustaría dar las gracias a mi director Pablo Urcola y a Luis Montesano, por ayudarme durante el proyecto, guiarme, por todo su gran apoyo y su dedicación en mí.

A Carlos Escalano, Ana Cris y a todo el equipo de BitBrain, por todos sus consejos, por su dedicación e interés.

A Dani, por estar a mi lado durante los últimos 5 años, por cuidarme y confiar en mí en todo momento.

A mis padres, por todo su cariño y sobretodo por confiar en mí desde el principio.

A mis amigas, amigos y compañeros por hacerme desconectar y hacerme disfrutar de esta etapa.

Gracias a todos

# RESUMEN

Este proyecto tiene como objetivo la creación de un entorno virtual donde se puedan realizar experimentos neurocientíficos y poder estudiar correlatos motores.

El diseño de experimentos en neurociencia es crítico puesto que se necesita un entorno muy controlado, para asegurar la reproducibilidad, e instrumentalizado para poder situar las señales registradas del cerebro en un contexto.

Estas restricciones hacen que sea inmanejable crear escenarios que permitan estudiar el comportamiento humano en situaciones complejas, algo que cada vez está más demandado.

El uso de la realidad virtual simplifica el diseño de escenarios aptos para este tipo de experimentos de neurociencia que permitan estudiar los correlatos motores de los sujetos. Además, permite introducir de forma sencilla situaciones inesperadas por los sujetos. La percepción de estos eventos provocan potenciales de error en el cerebro de los sujetos, que es otro de los elementos objeto de estudio en la neurociencia.

Para conseguir el objetivo del proyecto, se han completado distintas tareas: Primero, se ha integrado la adquisición de señal de EEG en el software de realidad virtual, con especial cuidado en la sincronización de los datos. Se ha diseñado un entorno virtual de primera persona con un avatar que imita los movimientos reales de los brazos y la cabeza del sujeto. En ese entorno, se han diseñado dos protocolos de neurociencia para medir los correlatos motores y los potenciales de error. Finalmente, se han procesado los datos obtenidos, analizando los correlatos motores, la desincronización asociada a eventos y los potenciales de error provocados por situaciones discordantes entre la realidad y el mundo virtual.

# Índice

# Capítulo 1

# Introducción y objetivos

## 1.1. Motivación

La neurociencia es un campo de la ciencia dedicada al estudio del sistema nervioso, donde se pretende aprender sobre la cognición y la conducta del ser humano. Algunos estudios neurocientíficos tienen como objetivo entender como funciona el cerebro y poder mejorar la calidad de vida de gente con patologías.

Actualmente existen empresas dedicadas al campo de la neurociencia, una de ellas es *Neuralink*[1] cuyo objetivo es devolver a pacientes la capacidad de movilidad, revertir la visión o la audición. Otra de las empresas que también se dedica a este campo es *Neurable*[2] donde hacen uso de la actividad cerebral generada por el usuario para controlar una realidad virtual o una realidad aumentada. Otra de las empresas que también se dedica a este campo es Bit&Brain Technologies[3], spin-off de la Universidad de Zaragoza, con la cual se ha realizado una colaboración para el desarrollo de este proyecto.

Además de diseñar y fabricar sus dispositivos, Bit&Brain es pionera en el diseño de protocolos y en el análisis de correlatos neuronales, para ello es necesario registrar la actividad cerebral en la región correspondiente del córtex donde se generan los correlatos (Figura 1.1). Existen dos métodos para poder registrar la actividad cerebral: *métodos invasivos* donde es recogida a partir de varios electrodos colocados dentro del cráneo, y *métodos no invasivos* donde se registra la actividad eléctrica cerebral recogida a partir de varios electrodos colocados sobre la superficie craneal. En este último grupo se encuentra el electroencefalograma (EEG), tecnología principal utilizada por los dispositivos de Bit&Brain.

La primera grabación de actividad neuronal fue recogida gracias al descubrimiento

[1] https://www.neuralink.com
[2] http://www.neurable.com
[3] https://www.bitbrain.com

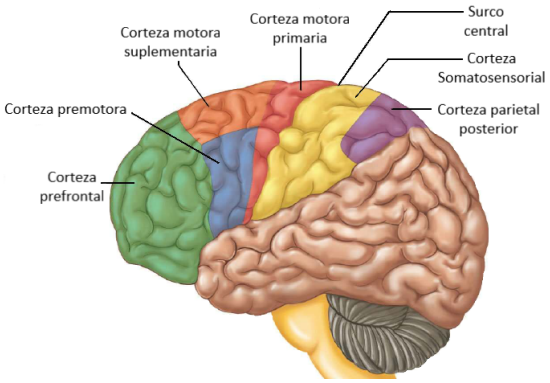



Figura 1.1: Regiones del córtex cerebral [1].

en 1929 realizado por el psiquiatra alemán Hans Berger [2], que inventó un sistema con el cual se registraba la actividad eléctrica cerebral sobre el cuero cabelludo. Después de este descubrimiento, los avances en la medición de la actividad neuronal han crecido exponencialmente, con aplicaciones en diferentes de campos tales como el diagnóstico de enfermedades o el control de dispositivos a partir de su actividad neuronal.

En los estudios de EEG, como se puede observar en la Figura 1.2, el usuario se coloca el dispositivo que registra la actividad cerebral producida mientras ejecuta un protocolo en el cual deberá de realizar tareas para producir distintos eventos (*onset*). Tras finalizar el experimento se procesa la señal y se estudia la actividad neuronal relacionada con los eventos. Donde se pueden estudiar correlatos neuronales cognitivo, por ejemplo, la onda p300, la carga de trabajo soportada por el usuario, los potenciales relacionados con errores (ERPs); correlatos neuronales emocionales, por ejemplo, la asimetría en alfa; correlatos motores, donde se encuentran los (MRCPs)[4] y los ERDs (desincronización relacionada con eventos), estos dos correlatos se anticipan al *onset*, indicando que el cerebro se "prepara" para las acciones.

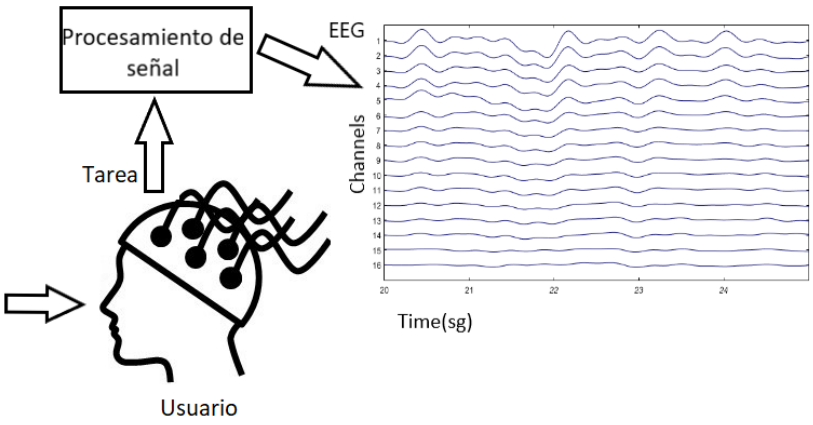



Figura 1.2: Adquisición y procesamiento EEG.

[4]En inglés, potencial cortical relacionado con el movimiento

Existen grandes limitaciones al diseñar protocolos EEG. Por un lado está el entorno de experimentación, que debe prepararse de forma muy rigurosa, teniendo un gran número de restricciones sobre el espacio y la colocación de los objetos para asegurar la repetibilidad y un alto nivel de instrumentalización para medir el contexto en el que se registra el EEG. Por otro lado está la limitación en poder generar en el usuario potenciales relacionados con errores.

Debido a estas restricciones, Bit&Brain tiene interés en el uso de la realidad virtual para la creación y ejecución de protocolos. Los entornos virtuales ofrecen ventajas a la hora de diseñar experimentos EEG como la movilidad del experimento, la posibilidad de recrear escenarios complejos (objetos móviles, "flotantes", animales) o la capacidad de alterar la percepción y el movimiento del usuario entre otras.

La combinación de la realidad virtual y EEG es muy novedosa, con lo cual no hay muchos artículos que traten sobre este tema.

En el estudio [3] hacen uso de 5 pantallas diferentes, cada una de ellas con versión en 2D y 3D. Los resultados muestran que en las pantalla de 3D recibieron una mejor retención en los pacientes, es decir, se vieron en un grado mayor de implicación durante la escena. Otro estudio [4] detectó que el uso de una vista en primera persona, en lugar de usar una vista en tercera persona, proporciona implicación por parte del usuario en realidad virtual. Otro estudio [5] crea un entorno virtual donde se representa de forma individual 4 tipos de animaciones diferentes: una mano estática, una mano en movimiento, un cubo estático y un cubo en movimiento. El experimento consiste en presentar las animaciones a los sujetos mientras se registra la actividad cerebral. Se obtuvo una mayor oscilación en la banda alfa y beta en el caso de tener un objeto relacionado con un componente del cuerpo humano en movimiento.

Estos resultados son prometedores a la hora de utilizar la realidad virtual en protocolos de neurociencia.

## 1.2. Objetivos

Este proyecto tiene como objetivo el desarrollo de un protocolo de experimentación para el análisis de correlatos motores en un sistema de realidad virtual, usando la tecnología de Bit&Brain para el registro de la actividad cerebral. Para poder cumplir con este objetivo se abordarán los siguientes subobjetivos:

- Integración de las tecnologías de realidad virtual con las librerías de comunicación y registro de los dispositivos Bit&Brain.

- Diseño e implementación de un protocolo en un entorno de realidad virtual para generar correlatos motores y cognitivos.
- Ejecutar el protocolo creado con varios sujetos.
- Analizar los resultados obtenidos a nivel de grandes medias de señal y, en caso de ser posible, clasificación.

La distribución temporal de las tareas para poder alcanzar los objetivos del proyecto se puede ver en el diagrama de Gantt de la Figura 1.3.

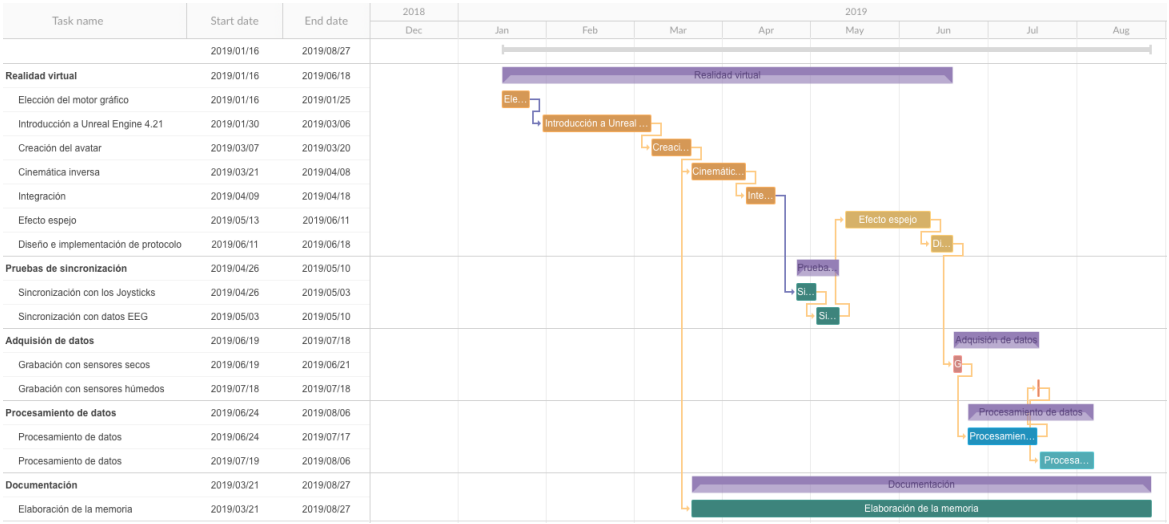



Figura 1.3: Diagrama de Gantt de la ejecución del proyecto

# Capítulo 2

# Tecnologías

En este capítulo se presentan las herramientas hardware y software que permiten la creación de entornos de realidad virtual en primera persona para la ejecución de experimentos de neurociencia.

## 2.1. Equipos de Bit&Brain

Bit&Brain ha diseñado el dispositivo Hero [6], el cual fue adaptado para poder incorporarlo a las Oculus Rift[1], esta adaptación llamada minimal EEG inmersive (Figura 2.1) cuenta con 12 sensores secos colocados en la zona de la corteza motora.

Figura 2.1: Gorro de captura de EEG con 12 sensores secos [6] adaptado al dispositivo de realidad virtual

Bit&Brain también ha diseñado Mobby [7], un gorro que cuenta con 16 sensores húmedos (Figura 2.2). A diferencia del anterior, éste no requiere la adaptación para

[1]HMD usado para la visualización de la experiencia virtual

incorporarlo a las Oculus Rift. Cabe mencionar que el uso de sensores húmedos proporciona medidas menos ruidosas de la actividad cerebral en comparación con sensores secos.


Figura 2.2: Gorro de captura de EEG con 16 sensores húmedos [7]

Por último, también se utilizaron otros instrumentos de medición como fotodiodos y un botón, diseñados por Bit&Brain, para poder marcar eventos durante la preparación del escenario.

Todos estos dispositivos disponen de librerías en C++ que permiten la conexión y adquisición de datos.

## 2.2. Equipos de Realidad virtual

### 2.2.1. Introducción

Las tecnologías de la realidad virtual (RV) fueron introducidas en el mercado en 1994, donde se simulaban entornos artificiales creados a partir de computadoras, contaban con salida de audio, un *Game Controller* y sistema háptico. Consiguiendo la inmersión total del usuario en la escena [4].

El surgimiento de esta disciplina fue anterior a 1994 [8] [9]. En 1838 fue creado el primer estereoscopio, por Charles Wheatstone. La base de este producto consiste en dadas dos imágenes idénticas, de las cuales solo difiere la perspectiva, cada una de las imágenes es presentada en una lente distinta del estereoscopio, produciendo

en el cerebro del usuario una mezcla de ambas imágenes consiguiendo un efecto tridimensional en la imagen. Al conseguir tal efecto, se le considera el principio de la RV, aunque por aquella época no existía tal concepto.

Al cabo de más de un siglo, en 1968, gracias a Ivan Sutherland, surgió el concepto de realidad virtual, gracias a la creación del primer casco de realidad virtual (Head-Mount Display, HMD).

En 1977, Richard Sayre creó el primer guante sensitivo, el cual calculaba la flexión de los dedos y la capacidad de monitorizar los movimientos de la mano. Posteriormente fue mejorado por Dan Sandin y Thomas Defanti. Siguiendo por esta linea, Dr. Gary Grimes, en 1983, creó el primer guante, capaz de reconocer la posición de la mano.

Gracias a todos estos avances se empezó a comercializar hardware capaz de mostrar RV como son las Oculus Rift o las HTC vive, entre otros.

### 2.2.2. Elección de motor gráfico

Bit&Brain posee las Oculus Rift como HMD, cuyos requisitos mínimos se muestran en la Figura 2.3. Sin embargo, no había ninguna restricción a priori en cuanto a qué motor gráfico de RV utilizar. Además se dispone de los controladores (*joysticks*) para interactuar con el entorno de RV.

| SO | CPU | RAM | Tarjeta gráfica | Salida de vídeo | Puertos |
|---|---|---|---|---|---|
| Windows 7 SPI o superior | Intel i5 4590 o superior | 8GB | Nvidia GTX 970/AMD 290 o superior | HMDI 1.3 | 2 USB 3.0 |

Figura 2.3: Requisitos mínimos para utilizar el dispositivo Oculus Rift.

Para poder conseguir mostrar escenas en RV existe una amplia gama de motores gráficos, por lo que se realiza un análisis de los motores gráficos más populares en RV, listados a continuación:

- Unity 5: Motor de videojuegos desarrollado por Unity technologies en 2005. Posee una versión gratuita y dos versiones de pago. En el caso de comercializar el producto es obligatorio hacer uso de las versiones de pago. Unity permitía el uso de tres lenguajes de programación: C#, JavaScript y Boo. Aunque actualmente, solo es admitido C#.

- Unreal Engine 4: Motor de videojuegos desarrollado por la compañía Epic Games en 1998. Es un entorno gratuito desde 2015. En el caso de querer comercializar

el producto y obtener unas ganancias superiores a 3.000$, Epic Games obtendrá un 5 % de las ganancias cada trimestre. Unreal Engine permite los lenguajes de programación de C++11 y Blueprints[2]. Blueprints son unos scripts visuales, desarrollados por Epic Games que permiten interactuar también con código en C++.

- CryENGINE: Motor de videojuegos desarrollado por la compañía CryTek, en 2002. Es un entorno que no ofrece versión gratuita, teniendo una suscripción mensual de 10$. Los lenguajes de programación admitidos son: C++, Flash, ActionScript y Lua.

Para este proyecto, se desechó CryENGINE, ya que no ofrecía una versión gratuita. Por lo tanto se compararon las características de Unity y Unreal Engine, para poder elegir el más apropiado para el proyecto. Basándonos en los análisis realizados por la literatura [10], los principales puntos a considerar son los siguientes:

Primero de todo, para poder hacer uso de estos motores gráficos, se necesita comprobar si el equipo con el que se parte, cumple con los requisitos mínimos exigidos.

| | Unity 5.5.2 Requirement | Unreal 4.15 Requirement |
|---|---|---|
| Operating System | Windows 7, 8, 10; Mac OS X 10.8+ | Windows 7, 8 64-bit; Mac OS X 10.9.2; (Ubuntu 15.04) |
| CPU | 1 GHz or faster | Quad-core Intel or AMD, 2.5 GHz or faster |
| Memory | 2GB RAM | 8GB RAM (16GB RAM) |
| Video Card | Graphics Card with DirectX9+ | DirectX 11, OpenGL 4.1 (NVIDIA GeForce 470 GTX) |

Figura 2.4: Requisitos mínimos, Unreal Engine y Unity.

Se puede observar que Unreal Engine posee un motor gráfico más potente que Unity. Esta diferencia se debe a la calidad de los gráficos, siendo los de Unreal Engine superiores a los de Unity. Unreal Engine hace uso de funcionalidades avanzadas de iluminación dinámica y posee una velocidad de rendering superior a Unity, ya que la velocidad de render de C++11 es superior a la de C#. Además Unreal Engine posee una optimización superior a la de Unity.

Actualmente, Unity suele ser usado para el desarrollo de videojuegos para plataformas móviles, en cambio, en el resto de plataformas se suele desarrollar en Unreal Engine. Esto puede ser comprensible teniendo en cuenta lo citado anteriormente.

La curva de aprendizaje de Unity es inferior a la de Unreal Engine, debido a que Unity solo hace uso de C# y Unreal Engine hace uso de Blueprints y C++11, ofreciendo bibliotecas propias en C++. Según la complejidad que se desea añadir, puede no ser

[2]https://docs.unrealengine.com/en-US/Engine/Blueprints/index.html

cubierto totalmente por Blueprints, por ejemplo en el caso de querer crear hilos, siendo necesario hacer uso de C++ para cubrir tales necesidades.

Para poder obtener un aprendizaje en ambos motores, existe una gran cantidad de documentación, diversos foros activos, entre ellos los oficiales en su web, videotutoriales, etc.

Unreal Engine, posee una desventaja frente a Unity, ya que a la hora de depurar el proyecto, en el caso de Unity es muy sencillo, pero en el caso de Unreal Engine puede volverse muy compleja si se desea depurar código en C++. Además, en el caso de poseer código C++ el tiempo de compilación de Unreal Engine es superior al de Unity, en cambio si solo se poseen Blueprints el tiempo de compilación en Unity es superior al de Unreal Engine.

Ambos motores requieren el uso de IDEs, en el caso de Unreal Engine es obligatorio hacer uso de Visual Studio. Dependiendo de la versión de Unreal Engine le correspondera una versión especifica de Visual Studio. En cambio Unity es más abierto en este aspecto, ya que permite el uso de Visual Studio o editores de texto, como Notepad++ o Sublime Text.

Por otra parte, uno de los requisitos de este proyecto es la virtualización de brazos humanos, por lo que podría ser un aspecto a tener en cuenta la existencia de librería que lo ofrezcan, aunque ambos motores ofrecen la posibilidad de importar mallas creadas a partir de herramientas de modelado 3D.

Ambos motores, ofrecen mallas de esqueleto[3] [4] en las que solo son visibles los dos brazos, con una apariencia muy realista, además son totalmente compatibles con VR.

Una vez estudiadas las ventajas y desventajas que ofrecen Unreal Engine 4 y Unity, se tomó la decisión de hacer uso de Unreal Engine 4.21.2 para la creación de la RV del proyecto. El motivo principal de esta decisión fue debido a los excelentes gráficos en RV ofrecidos por Unreal Engine. Esta cualidad es importante ya que se busca tener el mayor realismo posible y poseer escenas que sean totalmente agradables a la vista del usuario. De cara a la implementación se cuenta con conocimientos avanzados en C++ y conocimientos básicos en Blueprints, además se toma la bibliografía oficial de Unreal Engine ([11],[12], [13]) para poder tener mayor conocimiento sobre el desarrollo en Unreal Engine. Otra ventaja de este motor gráfico es el uso de C++, ya que la librería de Bit&Brain necesaria para poder comunicarse con los amplificadores está implementada en C++.

Para poder llevar a cabo el desarrollo del proyecto se cuenta con la siguiente

[3]https://assetstore.unity.com/packages/tools/animation/full-arms-vr-ultimate-interaction-toolset-102839
[4]https://www.unrealengine.com/marketplace/en-US/slug/aaa-first-person-arms-pack

infraestructura, la cual es capaz de soportar el uso de Unreal Engine 4.21.2 y del desarrollo de una RV.

- CPU i5 8400 2.80GHz
- 8 GB de RAM
- 2 USB 3.0 y 3 USB 2.0
- Nvidia GTX 1050Ti
- HMDI 2.0

# Capítulo 3

# Entorno virtual para experimentación con EEG en primera persona

El objetivo es crear un entorno en primera persona que permita el estudio de variables cognitivas, donde se recreen todos los movimientos locomotores del usuario usando los joysticks como fuente de información de posición y rotación de cada mano.

Esto permite el estudio de los correlatos motores. Además, gracias a la capacidad de alteración del entorno virtual, se realizará un efecto espejo en el movimiento de los brazos: el usuario controlará su brazo virtual derecho con el joystick de su mano real izquierda y viceversa. Este efecto creará inconsistencias entre el mundo real y el virtual, permitiendo el estudio de los potenciales de error generados.

## 3.1. Creación del avatar

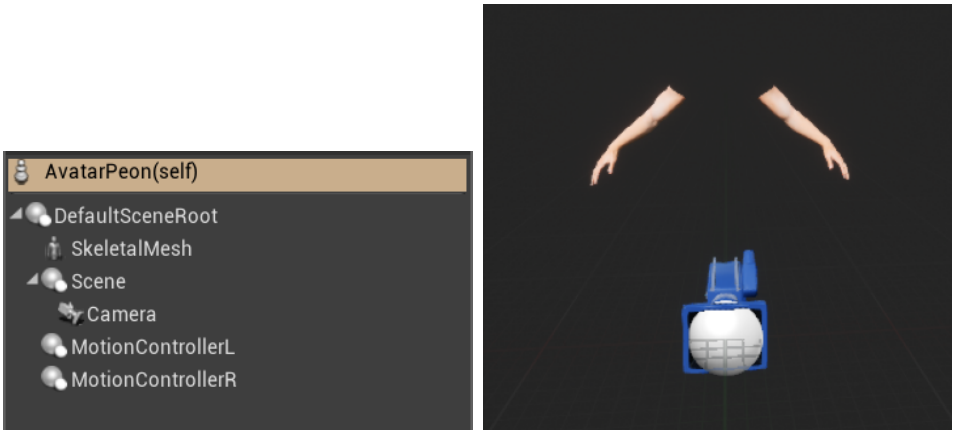
Para la creación del avatar [1] junto a la escena en RV, se hará uso de los Blueprints, ya que estos poseen una compilación más rápida que C++, y todas las características deseadas que posea el avatar son posibles de desarrollar con los Blueprints.

Para poder crear un avatar Unreal Engine ofrece tres clases de actores, *Actor*, *Character* y *Pawn*. Cada uno de ellos se diferencian por el objetivo a resolver. Por un lado la clase *Actor* es un tipo de objeto que puede ser colocado o generado en el mundo, es la clase padre de la cual extienden *Character* y *Pawn*. La clase *Character* posee la capacidad de caminar por el mundo. Esta clase es usada cuando se desea crear personajes que posean movimientos locomotores. Por último la clase *Pawn* tiene la capacidad de recibir información de los controladores pertenecientes a la RV, es decir, permiten la adquisición de los datos proporcionados por los joysticks.

[1] Un avatar en RV es la representación gráfica de la figura del jugador en la RV.

Para este proyecto se toma la clase de *Pawn* para poder representar el avatar, ya que gracias a ella se recibe la información procedente de los joysticks. Se hace uso de una malla de esqueleto disponible en el *marketplace* de Unreal [2] que representa brazos humanos. Una vez que se posee la clase *Pawn* como clase principal en la RV, esta clase posee un atributo de *malla de esqueleto*, en este atributo se deberá de indicar el uso de la malla de esqueleto descargada de la librería mencionada anteriormente. Por otro lado para poder visualizar la RV es necesario hacer uso del atributo de *Camera Components*. Por último, para poder hacer uso de los joysticks, se añaden dos componentes de tipo *Motion Controller Component*, cada uno de ellos representara a un joystick. Cada uno de ellos se le le indica su fuente, es decir, si se trata del joystick derecho o el izquierdo.

Tras añadir todos los componentes mencionados se obtiene el avatar representado en la Figura 3.1.




(a) Componentes añadidos. (b) Resultado gráfico.

Figura 3.1: Resultado en el editor de Unreal Engine tras la creación avatar

## 3.2. Cadenas cinemáticas

Una cadena cinemática esta constituida por un conjunto de cuerpos rígidos unidos mediante articulaciones. Esta cadena poseerá un número de grados de libertad. Un grado de libertad *DOF* representa la posibilidad de movimiento en un eje, siendo este el movimiento de traslación o de rotación. En la Figura 3.2 se puede observar una cadena cinemática en un espacio 2D, formada por 3 rígidos ($L_1$,$L_2$,$L_3$), estos cuerpos pueden ser rotados en tres ángulos ($\theta_1$, $\theta_2$, $\theta_3$) teniendo 3 grados de libertad en la cadena cinemática.

[2]https://www.unrealengine.com/marketplace/en-US/slug/aaa-first-person-arms-pack

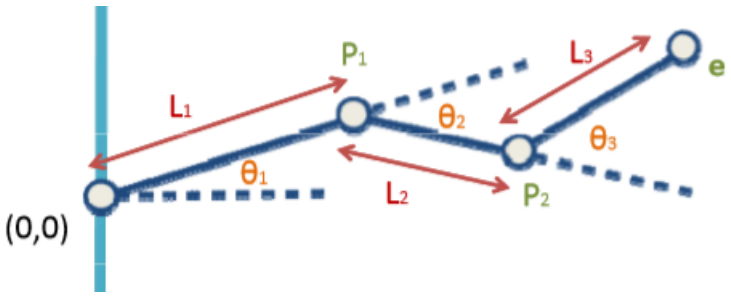



Figura 3.2: Cadena de articulación

Para poder obtener la posición y la orientación de una cadena cinemática se recurre a resolver a un problema de cinemática directa o a un problema de cinemática inversa.

En un problema de cinemática directa se conocen a priori los ángulos de cada articulación de la cadena con respecto al segmento anterior. Por lo tanto para recrear la posición del extremo final de la cadena solo es necesario aplicar los valores correspondientes a cada articulación.

En un problema de cinemática inversa, se parte únicamente de los valores de la posición y de la orientación del extremo final de la cadena. En este problema se tiene que calcular los ángulos que tendría cada articulación de la cadena dada. A diferencia del problema de cinemática directa, la indirecta presenta diversas dificultades. Si no consideramos restricciones en las articulaciones, el problema tiene múltiples soluciones, y la complejidad del problema crece según el número de articulaciones contenidas en la cadena. Se suele resolver por métodos de optimización numérica, por lo que puede haber casos que no se obtenga la solución óptima al problema.

Tras conocer las precondiciones de cada problema, se hace uso del problema de cinemática inversa, aplicándola a cada brazo del esqueleto. Se toma este problema debido a que únicamente se posee la posición y orientación del extremo de la cadena cinemática, es decir, solo se conoce el punto $e$ visible en la Figura 3.2. Estos datos son obtenidos a partir de los joysticks.

### 3.2.1. Cinemática inversa en Unreal Engine

Teniendo en cuenta la cadena de cinemática explicada en la sección anterior, si se traspasa a la malla de esqueleto obtenida en Unreal Engine se obtiene la cadena visible en la Figura 3.3.

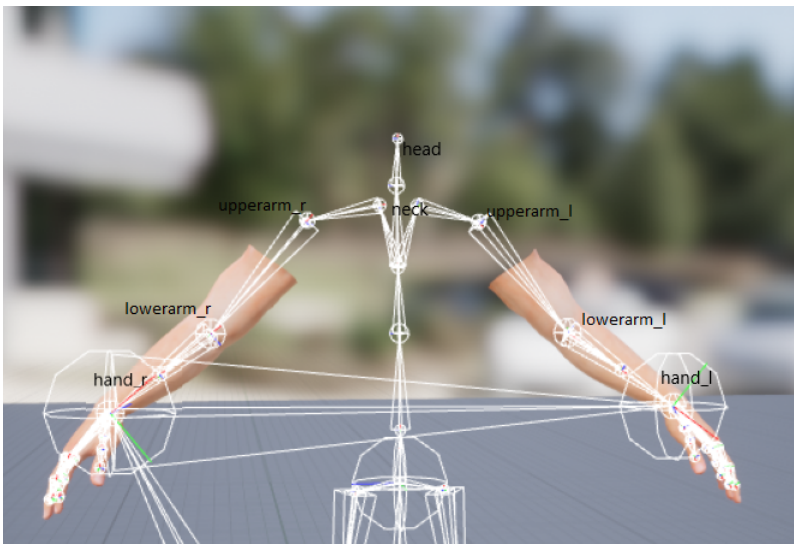



Figura 3.3: Cadenas de articulaciones en los brazos, Unreal Engine

Se toman dos cadenas cinemáticas, por un lado el brazo derecho y por otro el brazo izquierdo, las cuales deberán ser dotadas de movimiento, Unreal Engine ofrece clases específicas para la creación de movimiento, se trata de *Animation Blueprint*, cuya clase padre es *AnimInstance*, donde se indica a qué malla de esqueleto se le debe dotar de movimiento. Esta clase esta subdividida en dos componentes *EventGraph* y *AnimGraph*.

El componente *EventGraph* se encarga de recolectar todas las actualizaciones de los datos usados para la recreación del movimiento, estos datos son:

- Posición y orientación del HMD, que determina desde que punto el usuario está visualizando la RV y su orientación.
- Posición y rotación de cada joystick, las cuales representaran la posición y la orientación de cada mano.
- Coordenadas de las manos de la malla de esqueleto: este punto es usado para conocer la posición de las manos en la RV. Hay que tener en cuenta la distancia comprendida entre el joystick y el punto de la mano, se añade a cada mano de la malla de esqueleto un *socket*, ocupando la misma posición que ocuparía el joystick correspondiente.

Debido a que hay ocasiones donde los joysticks no están activados hasta que estos son usados, se simula una posición inicial indicando la localización y la rotación de las manos. De esta manera el usuario podrá observar sus brazos de una forma más cercana a la realidad.

El componente *AnimGraph* es usado para generar la posición final, resultado de la cinemática inversa. Para ello Unreal Engine ofrece varios métodos capaces de ofrecer una solución a este problema.

Un método es *Two-bone IK*, en el cual se debe indicar el hueso que ocuparía el extremo final de la cadena de articulación, se realiza la cinemática inversa evaluando primero los dos huesos superiores al hueso indicado como extremo final de la cadena de articulación. En este proyecto se indica como hueso del extremo inferior *Hand*, resolviendo la cinemática inversa en el siguiente orden: *Lowerarm*, *Upperarm* y *Hand*.
Teniendo en cuenta que la cinemática inversa produce un amplio abanico de soluciones, es posible dirigir la solución para que sea más cercana a la deseada. Se trata del parámetro llamado *Join Target*, que viene a significar la compensación que se genera con el hueso anterior al indicado como extremo de la cadena de articulaciones, modificando el desplazamiento generado en la articulación que une este hueso con el extremo superior de la cadena de articulaciones. En este caso, al tratarse del brazo, el parámetro modificaría la posición del codo con respecto a la mano. Así se consigue evitar la posibilidad de que el hombro termine en una posición equivocada o que el antebrazo esté orientado en una posición no natural. El valor final de este parámetro fue decidido considerando que en nuestro caso la articulación de la muñeca se iba a girar muy poco con respecto al brazo.
Por último, este método no tiene en cuenta la rotación de la cadena de articulación. Para tenerla en cuenta se hace uso del método de *Transformar(Modificar) hueso* donde se especifica el hueso al que se aplica *Two-bone IK* indicándole la rotación recibida desde los joystick para ser aplicadas a la orientación de las manos.

Otro método que se ofrece es *Fabrik* donde se aplica la cinemática inversa desde el hueso indicado como *hueso raíz* hasta el hueso indicado como *punta de hueso*. Sin restringir el número de articulaciones que exista en la cadena. Este método, al igual que el anterior, se evalúa de forma iterativa 10 veces.
Al poseer un método que no restrinja el número de articulaciones en la cadena, se hace uso del método para la cadena de articulación formada entre la cabeza (punto de hueso) y el cuello (hueso raíz), ya que únicamente están unidos por una articulación. Como fuente de información de posición y orientación de la cabeza se tomó el proporcionado por el HMD. Otra opción también probada fue el marcar la punta del hueso en la cabeza y el hueso raíz en la pelvis, pero esto provocaría la posibilidad de una rotación completa hasta la pelvis, provocando giros del tronco superior, lo que no correspondería a la acción esperada. Se optó por la primera opción, ya que el usuario no movería su tronco superior y solo movería los brazos y la cabeza.

Por último, también se debe tener en cuenta la posición y la rotación del esqueleto dentro del mundo virtual que es ajustado en el momento en el que se inicia la RV y cuando el usuario lo desee. Los puntos usados para determinar la posición y la rotación serán proporcionados por el HMD, para determinar la orientación del esqueleto se toma la orientación en el eje Z del HMD, es decir la orientación horizontal, para determinar la posición del esqueleto se tomo la localización en el eje X,Y,Z del HMD.
Estos valores serán indicados en *root*. Este punto indica en que posición y orientación se encuentra el esqueleto completo, ya que se trata del hueso principal del esqueleto. Al representar la posición en la que se encuentra el esqueleto es necesario modificar el valor del eje Z de la posición del HMD, ya que este eje determina la altura (el sistema de coordenadas se puede observar en la Figura 3.5), por lo que este valor deberá de ser a la altura a la que se comprende el suelo.
Se hace uso del método de *Trasformar(modificar) hueso* para determinar los nuevos valores que deberá tomar *root*.

El resultado en el componente AnimGraph para poder resolver la cinemática inversa es visible en el Apéndice B. El resultado de la representación del movimiento en el mundo real al mundo virtual se puede visualizar en la Figura C.1 del Apéndice C.

## 3.3. Efecto espejo

Además de la representación en la RV de los movimientos locomotores realizados por el usuario en el mundo real, se reproducirá en la RV de forma aleatoria un efecto espejo. La intención de este efecto, es el mostrar el brazo contrario al que en verdad se ha movido realizando un movimiento acorde a una reflexión. Un ejemplo de una reflexión en un espacio 2D, realizada sobre el eje Y, es el siguiente:

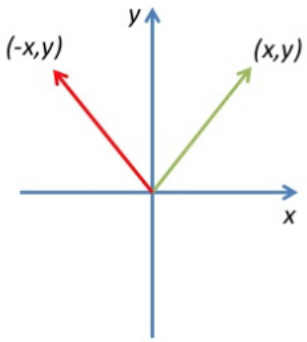



Figura 3.4: Reflexión 2D

En el caso de este proyecto se trabaja con un espacio 3D, hay que tener en cuenta que la reflexión se realizará en el espacio del esqueleto. El sistema de coordenadas con

el que se trabaja en la reflexión es visible en la Figura 3.5, realizando la reflexión en el plano YZ.

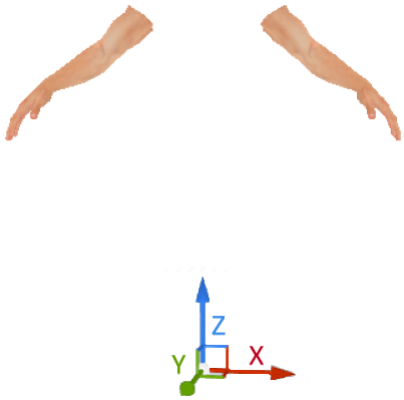



Figura 3.5: Sistema de coordenadas del espacio del esqueleto, Unreal Engine.

### 3.3.1. Efecto espejo en Unreal Engine

Primero de todo para poder recrear el efecto espejo es necesario conocer los espacios de traslación y rotación con lo que se trabaja en cada mano (Figura 3.6).


(a) Espacio de rotación


(b) Espacio de traslación

Figura 3.6: Espacio traslación y rotación en las manos de cada brazo, Unreal Engine.

Para recrear el efecto espejo se resuelve primero la cinemática inversa, y se realiza el problema de la reflexión en cada articulación de la cadena (mano, codo y hombro), y se asignan los valores de cada articulación a la nueva cadena mediante un problema de cinemática directa. Para poder hacer este proceso se toman dos esqueletos: uno oculto dedicado a la realización de la cinemática inversa, otro visible que genera el movimiento final.

El modo en el que se realiza la reflexión para cada junta, será calculando la reflexión del resultado de la posición, dada por la cinemática inversa de cada junta. Primero se realiza para el extremo superior de la cadena, es decir el hombro, ya que este guía la orientación del resto de la cadena.

Para poder calcular los puntos que corresponden en el espacio del esqueleto para cada hombro, se toma como punto de referencia un punto del esqueleto que sea común ente ambos brazos, se toma el punto del esqueleto *spine03*, que corresponde al pecho, con este punto se realiza la siguiente transformación:

$$\begin{aligned} T_D &=^L T^{-1}_{spine03} \cdot^L T_{hombroD} \\ T_I &=^L T^{-1}_{spine03} \cdot^L T_{hombroI} \end{aligned} \tag{3.1}$$

Siendo $T_{hombroD}$ la transformada del hombro derecho, $T_{hombroI}$ la transformada del hombro izquierdo. $T_D$ y $T_D$ serán las transformadas donde se indica los puntos que corresponden en el espacio del esqueleto a cada hombro.

Tras conocer la transformación de cada hombro en el espacio local, se calcula la reflexión.

$$\begin{aligned} R_D &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \cdot T_D \\ R_I &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \cdot T_I \end{aligned} \tag{3.2}$$

Otro aspecto a considerar es el hueso de la clavícula, hueso padre del hombro. La clavícula guía la orientación del brazo al que corresponde. Por lo tanto los puntos calculados en la reflexión deberán pertenecer al mismo espacio de coordenadas de las clavículas del brazo al que se ha realizado su reflexión. Se realizan las siguientes operaciones:

$$\begin{aligned} H_{CL} &=^L T^{-1}_{spine03} \cdot^L T_{claviculaL} \\ H_{CR} &=^L T^{-1}_{spine03} \cdot^L T_{claviculaR} \end{aligned} \tag{3.3}$$

Siendo $H_{CL}$ y $H_{CR}$ la transformada en el espacio global del pecho de la clavícula izquierda y derecha, respectivamente.

$$\begin{aligned} U_L &= H_{CL}^{-1} \cdot R_D \\ U_R &= H_{CR}^{-1} \cdot R_I \end{aligned} \tag{3.4}$$

Siendo $U_L$ y $U_R$ la transformada final del hombro izquierdo y derecho, respectivamente, en el espacio global. Pero aun no se posee la rotación adecuada, para ello se rota en $U_L$y $U_R$ 180º grados en el eje X,Y.

Finalmente $N_{HI}$ corresponde con la transformada del hombro izquierdo y $N_{HD}$ a la del hombro derecho. El resto de la cadena mantiene la misma orientación, gracias a tener en cuenta la clavícula del brazo opuesto. Tras obtener el resultado esperado se usa esta versión para la resolución del efecto espejo. En el algoritmo 1 se resume esta solución. El resultado final se puede visualizar en las figuras C.2 y C.3 del Apéndice C.

**Algoritmo 1:** Calculo del efecto espejo

**Data:** $X_D$ coordenadas del joystick derecho, $X_I$ coordenadas del joystick izquierdo.

**Result:** $Y_D$ coordenadas de cada articulación del brazo derecho de la nueva postura, $Y_I$ coordenadas de cada articulación del brazo izquierdo de la nueva postura.

**begin**

$C_D \longleftarrow cinematicaInversa(X_D)$
$C_I \longleftarrow cinematicaInversa(X_I)$
$R_D \longleftarrow reflexion(C_D)$
$R_I \longleftarrow reflexion(C_I)$
$Y_D \longleftarrow cinematicaDirecta(R_I)$
$Y_I \longleftarrow cinematicaDirecta(R_D)$

**end**

Sin embargo, durante este proceso se realizaron diversas técnicas para la resolución del efecto, pero tuvieron que ser desechadas por no obtener el efecto deseado. Todas ellas están basadas en aplicar primero una reflexión a las posiciones de las manos obtenidas de los joysticks y posteriormente aplicar la cinemática inversa sobre el brazo contrario.

El primer intento de reflexión se intentó directamente sobre el eje X (izquierda-derecha) del avatar, puesto que la profundidad y altura (Y y Z respectivamente) que no deberían verse afectadas por el intercambio de los brazos. Esta técnica supone que el avatar está estático durante la prueba. La reflexión es calculada de la siguiente manera:

$$E = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot X \tag{3.5}$$

Siendo $X$ las coordenadas de una mano y $E$ las coordenadas reflejadas.

Tras realizar la reflexión se calcula la cinemática inversa. Se explica de forma más detallada en el Apéndice A.1. Se muestra el pseudocódigo de esta solución en el algoritmo 2.

**Algoritmo 2:** Cálculo del efecto espejo (Intento 1)

**Data:** $X_D$ coordenadas del joystick derecho, $X_I$ coordenadas del joystick izquierdo.
**Result:** $Y_D$ coordenadas de cada articulación del brazo derecho de la nueva postura, $Y_I$ coordenadas de cada articulación del brazo izquierdo de la nueva postura.

```
begin
    R_D ⟵ reflexionEjeX(X_D)
    R_I ⟵ reflexionEjeX(X_I)
    Y_D ⟵ cinematicaInversa(R_I)
    Y_I ⟵ cinematicaInversa(R_D)
end
```

Esta solución se descartó porque proporciona posiciones erróneas en los brazos, haciendo que estos estuvieran a una gran distancia del esqueleto puesto que no tiene en cuenta la rotación, por pequeña que sea durante el experimento, del avatar.

Para solucionar estos problemas, se planteó otra solución donde se tiene en cuenta la posición y la rotación del avatar, usando el punto de *root* como punto de referencia para realizar la reflexión. Una vez conocidas las posiciones de las manos respecto de *root*, se realiza la reflexión, usando la misma fórmula usada en 3.5. Esta reflexión estará calculada en el espacio local del esqueleto. Tras conocer los puntos de cada mano tras la reflexión, deberán ser transformadas al espacio global y poder calcular la cinemática inversa. Esta idea sigue el algoritmo presentando en Algoritmo 3. En el Apéndice A.2 se detalla esta solución.

Sin embargo esta solución tampoco conseguía los resultados deseados fundamentalmente porque un sistema de RV no es capaz de calcular de forma precisa la rotación de todo el avatar usando únicamente la información de la orientación de la cabeza y las manos.

**Algoritmo 3:** Calculo del efecto espejo (Intento 2)

**Data:** $X_D$ coordenadas del joystick derecho, $X_I$ coordenadas del joystick izquierdo.

**Result:** $Y_D$ coordenadas de cada articulación del brazo derecho de la nueva postura, $Y_I$ coordenadas de cada articulación del brazo izquierdo de la nueva postura.

**begin**

$M_D \longleftarrow posicionLocalMano(X_D)$
$M_I \longleftarrow posicionLocalMano(X_I)$
$R_D \longleftarrow reflexionCoordenadas(M_D)$
$R_I \longleftarrow reflexionCoordenadas(M_I)$
$R_D \longleftarrow posicionGlobalMano(R_D)$
$R_I \longleftarrow posicionGlobalMano(R_I)$
$Y_D \longleftarrow cinematicaInversa(R_I)$
$Y_I \longleftarrow cinematicaInversa(R_D)$

**end**

Otra idea fue calcular un plano de reflexión imaginario perpendicular al torso, provocando una división en el esqueleto en dos mitades de forma vertical. Los brazos quedan divididos y quedan a la misma distancia del plano imaginario. La solución propuesta fue calcular la proyección ortogonal del punto, el cual corresponde al punto donde está la nueva posición de la mano. Una vez obtenida esta proyección se le deberá de aumentar la distancia existente entre el plano imaginario y la mano opuesta, este resultado se asigna a la posición de la mano opuesta. Una vez calculadas las posiciones finales de las manos se realiza la cinemática inversa. Esta solución posee ciertas restricciones, ya que al poner un plano de forma vertical al torso, este no podría ser rotado, además no se podrá atravesar dicho plano ya que no habría solución. Por estas restricciones la solución fue descartada.

Por todos estos problemas se descarto la reflexión sobre la posición de los mandos y se optó por calcular primero la cinemática inversa y posteriormente invertir en el espacio de configuración de las articulaciones, como se ha explicado anteriormente.

## 3.4. Integración RV + EEG

El objetivo de esta integración es dotar al entorno de RV la capacidad de obtener datos de los equipos de registro de EEG. Desde un punto de vista de desarrollo, esta integración se realiza integrando en el motor gráfico de Unreal Engine la librería propia de Bit&Brain, la cual ofrece todas las funciones necesarias para poder realizar las comunicaciones con todos los dispositivos propios de Bit&Brain.

Al ser necesario importar una librería externa de Unreal Engine, las importaciones de librerías externas en el editor de Blueprints no son posibles, solo es posible realizarlo

en el editor de C++. Esta restricción no influye negativamente a lo ya implementado, ya que las funciones declaradas e implementadas en C++ pueden ser invocadas desde el editor de Blueprints.

### 3.4.1. Comunicación entre C++ y Blueprints

La clase de la que debe extender para poder hacer uso de funciones visibles desde el editor de Blueprints, es *Blueprint Function Libraries*. Para poder compartir funciones entre C++ y Blueprints, se debe de utilizar la macro de *UFUNCTION()* antes de definir la función en cuestión. Esta macro requiere la siguiente sintaxis[14]:
UFUNCTION([specifier, ...], Category=name, [meta(key=value,...)])
ReturnType FunctionName([Parameter, Parameter, ...])

Donde los campos de *specifier* y *meta*, tomaran los siguientes valores:

- En el campo de *specifier* se hace uso del especificador de *BlueprintCallable*, ya que se desea tener un nodo que ejecute la función definida.
- En el campo de *meta* se hace uso de la opción de *KeyWords="value"*, donde se indican las palabras claves que faciliten la búsqueda de la función en el editor de Blueprint.

Por último, teniendo en cuenta que todas las funciones de Unreal Engine poseen una categoría, se hace uso del campo de *Category*, donde se indica el nombre de la categoría a la que pertenece la función declaradas, esta categoría no es estrictamente necesaria su existencia.

### 3.4.2. Comunicación con los dispositivos de Bit&Brain

Para poder comunicarse con los dispositivos de Bit&Brain es necesario crear un algoritmo que sea capaz de realizar la conexión y desconexión de forma satisfactoria. Durante la comunicación con el amplificador las señales registradas serán almacenadas en un fichero con formato *csv* por parte de la RV, siendo necesario realizar la comunicación con el amplificador de forma concurrente a la RV. Se decidió realizar la comunicación mediante un hilo separado para que la carga de procesamiento gráfico no afectara al registro de datos de EEG.

Para poder crear un hilo en Unreal Engine desde el editor de C++ es necesario hacer uso de la siguiente función [15] :
*static FRunnableThread* Create(class FRunnable * InRunnable, const TCHAR * ThreadName, uint32 InStackSize, EThreadPriority InThreadPri)*

La función devuelve el nuevo hilo creado, en caso de creación exitosa, o Null, en caso de error. El parámetro de tipo FRunnable, corresponde con el objeto que se desea ejecutar de forma arbitraria. Este objeto es de tipo *FRunnable*. Este tipo de objeto posee tres funciones establecidas: Init(), Run(), Exit().

- El método *Init()* se encarga de inicializar el hilo y en el caso de fallar la inicialización, el hilo detiene la ejecución y devuelve un código de error. Si tiene éxito, se llama al método de *Run()*.
- El método de *Run()*, es llamado de forma automática, será donde se realiza el algoritmo que se desea ejecutar de forma concurrente. En este proyecto, esta será la función encargada de realizar toda la adquisición de datos proporcionados por los amplificadores, también será donde se crearán y se modificarán los ficheros que contengan las datos de las señales registradas por el amplificador.
- El método *Stop()*, se ocupa de terminar el hilo de forma correcta.

Para poder conectarse y desconectarse con el amplificador, se realiza desde Blueprints, donde se llama a las funciones compartidas con C++ para que empiece o termine el hilo de comunicación. En la Figura 3.7 se puede visualizar como se declaran funciones en C++ y como son estas buscadas desde el editor de Blueprints.

```
UCLASS()
class  UCallToThread : public UBlueprintFunctionLibrary
{
    GENERATED_BODY()

        UFUNCTION(BlueprintCallable, Category = "Custom", meta = (Keywords = "thread"))
        static UCallToThread* ConstructLoad();

        UFUNCTION(BlueprintCallable, Category = "Custom", meta = (Keywords = "thread"))
         void EmpezarDriver(const FString name);

        UFUNCTION(BlueprintCallable, Category = "Custom", meta = (Keywords = "thread"))
         bool TerminarDriver(const bool endD);
```

(a) Definición de las funciones fichero.h .


(b) Resultado en Blueprints.

Figura 3.7: Funciones compartidas entre C++ y Blueprints.

Se ha tomado la decisión de comenzar la conexión con el dispositivo de forma automática al ejecutar la RV, mientras que la desconexión se realizará cuando se pulse la tecla $X$ del teclado.

## 3.5. Sincronización

Uno de los aspectos fundamentales de la integración es garantizar la sincronización entre los estímulos producidos por la RV sobre el usuario y los datos registrados por los amplificadores. Se realizaron dos pruebas para verificar esta sincronización.

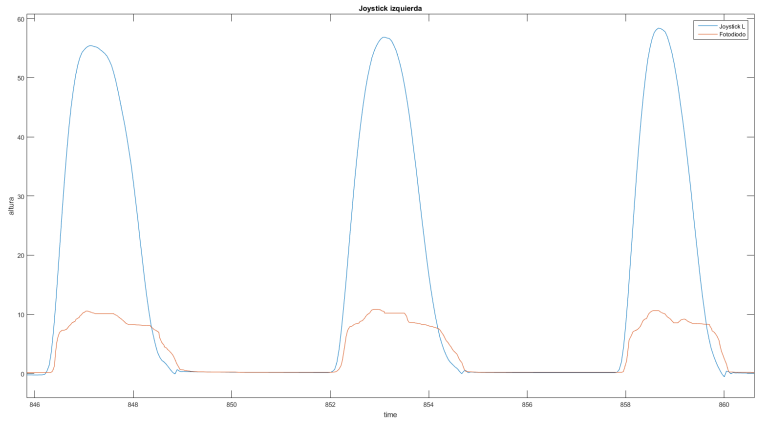
### 3.5.1. Prueba de sincronización con los joysticks

La primera prueba realizada tiene el objetivo de comprobar si el timestamp de los datos recibidos desde los joysticks coinciden con el timestamp asignado por el ordenador al recibir los datos provenientes del amplificador de Bit&Brain. Cabe destacar que el timestamp de los datos provenientes del amplificador poseen un cierto retardo, consecuencia del tiempo de vuelo del bluetooth.

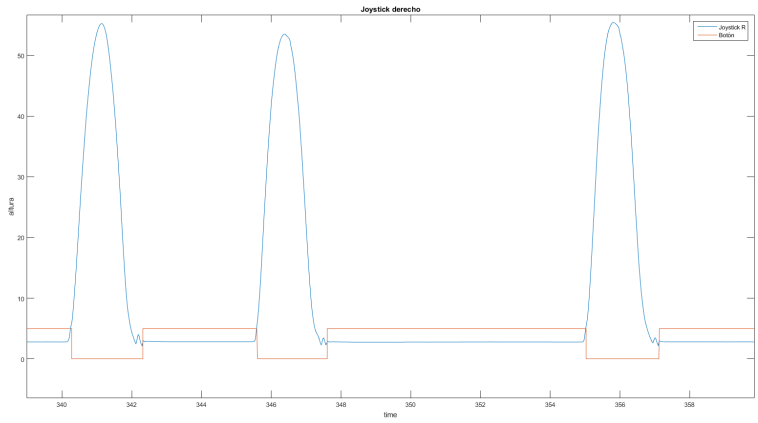
La prueba realizada consiste en colocar un fotodiodo debajo del joystick izquierdo y un botón debajo del joystick derecho. De esta manera al levantar un joystick su fotodiodo o su botón cambiará de amplitud, este cambio de amplitud indicará el valor en el eje Z del joystick en cada ínstate.

Sin embargo el fotodiodo y el botón no son precisos, debido a poseer un cierto retraso. En el caso del fotodiodo, el joystick correspondiente deberá de ser levantado entre 1cm y 2cm, permitiendo al fotodiodo recibir la suficiente luminosidad como para poder aumentar la amplitud de la señal. En el caso contrario de detectar la bajada del joystick, se tendrá que ocultar de forma total el fotodiodo impidiendo la recepción de luminosidad y obtener el valor mínimo de la amplitud en la señal.
En el caso del botón debido al diseño del joystick el botón queda presionado por el propio joystick y tendrá que ser levantado para dejar de presionarlo. En el caso de la bajada del joystick tendrá que presionarse el botón con el joystick.
Estas dos limitaciones provocan un cierto retraso de sincronización entre la señal del amplificador y los valores de Z del Joystcik. En la Figura 3.8 se muestra la posición en Z de los dos joysticks sincronizada con la señal del fotodiodo y del botón

Tras comprobar en la Figura 3.8, que la amplitud de las señales de los sensores varían de forma sincronizada con el valor del eje Z del joystick correspondiente, teniendo en cuenta el retardo existente producido por el bluetooth y dependiendo de la posición del joystick sobre el dispositivo, para que este se vea alterado, se concluye que se posee




(a) Joystick izquierdo junto a su fotodiodo. La amplitud del fotodiodo se ve alterada tras levantar el joystick 1cm de su posición de reposo, y cuando el joystick vuelve a la posición de reposo la amplitud del fotodiodo toma valor 0 cuando este está oculto por el joystick.




(b) JoyStick derecho junto a su botón. La amplitud de la señal toma valor 0 en el momento en el que el joystick es levantado unos 2.5cm, que corresponde con la distancia comprendida entre el inicio y el final del joystick. La amplitud crece cuando el joystick presiona el botón, se puede observar en los valores de Z del joystick en el momento en el que se toma la posición de reposo los valores de Z se ven alterados ya que se busca presionar el botón.

Figura 3.8: Sincronización en los joysticks con el fotodiodo y el botón

una sincronización entre la información recibida de la RV y los datos provenientes del amplificador.

Para completar este estudio, se detecta el momento en el que se va a levantar el joystick, este momento deberá ser anterior al momento de la variación de la amplitud de la señal del dispositivo correspondiente.

Para calcular el momento en el que se produce la subida del joystick, se realiza el algoritmo, visible en Algoritmo 4, en el cual se busca el momento en el que

el joystick supera el valor indicado como el límite en el que el joystick está ya elevado en el eje Z. Una vez que se supera este límite se presupone que el joystick ha sido levantado. Se buscan instantes anteriores en los que está por debajo de este límite y se encuentra en una posición de reposo, sin alterar los valores en Z. La comprobación del estado de reposo se realiza trazando una recta entre un instante y tres instantes anteriores. Si la pendiente es cercana a 0.2 indica que el joystick no ha sufrido ninguna modificación en Z, mientras que si es superior a 0.2 el joystick está en movimiento en el eje Z. Se tomará el instante en el que se obtenga una pendiente cercana a 0.2, como el momento en el que el joystick empieza a moverse. En la Figura 3.9, se puede observar el instante que determina el inicio del movimiento del joystick, instante anterior al incremento de la amplitud de la señal.

**Algoritmo 4:** Predicción del momento de subida del joystick

**Data:** $Z$ array con coordenadas del eje Z del joystick, $T$ array con timestamp de cada coordenada del joystick, *Limite* entero con el límite .
**Result:** *subida* instantes en los que se produce el levantamiento del joystick.

```
for i ← 3 to length(Z) do
    if Z[i] ≤Limite & Z[i+1]>Limite then
        q ⟵ i
        while (abs((Z[q-3]-Z[q])./(T[q-3]-T[q])) ≥ 0,2 & (Z[q] ≤ Limite)) do
            q ⟵ q − 1
        end
        subida.add(q)
    else
    end
```

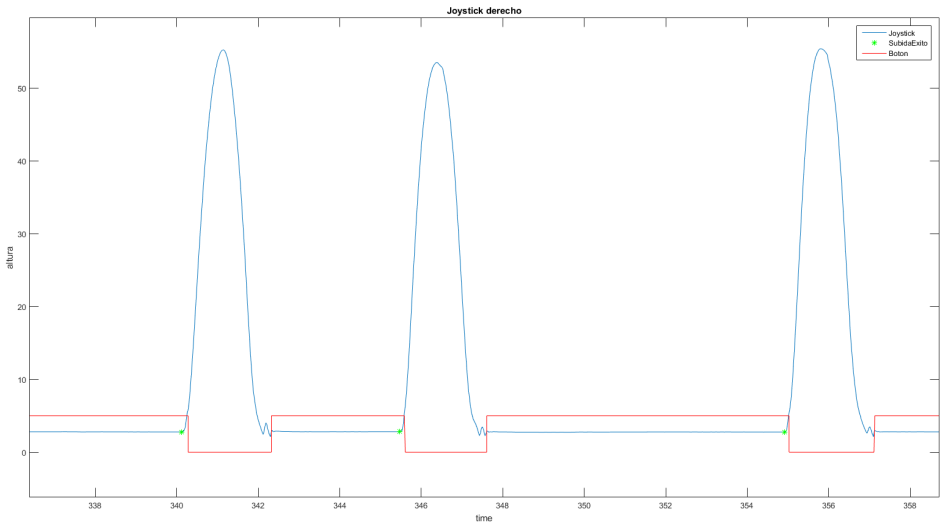



Figura 3.9: Detección de levantamiento del joystick derecho, marcado con un punto verde, junto a la señal procedente de un botón indicando el estado del joystick(altura=0, joystick levantando, altura=2 joystick en posición de reposo)

En la Figura 3.9 se observa la detección del levantamiento del joystick, este punto corresponde cuando el valor de Z empieza a incrementar.

Por último, se comprueba cuantitativamente la diferencia de tiempo entre el momento detectado con el joystick y el momento en el que el botón cambia de amplitud. La media de la diferencia es de 0.1464 sg con una desviación estándar de 0.0284 segundos. El retraso medio entre uno y otro es bastante grande y está afectado fundamentalmente por el bluetooth y el recorrido del botón utilizado. El tiempo de bluetooth tiene un retraso medio de unos 0.05 segundos y el botón tiene un recorrido de unos 2 cm desde que está totalmente pulsado hasta que se detecta que el botón se ha dejado de pulsar. Sin embargo, el mayor inconveniente para la sincronización no es un retraso constante sino la variabilidad en el retraso. En este caso, la desviación estándar es de unos 28 milisegundos, que se asemeja a la del bluetooth (unos 30 milisegundos). Esto nos hace concluir, que el método de detección de movimiento del brazo desarrollado no introduce un nuevo factor de desincronización entre las señales.

### 3.5.2. Prueba de sincronización con pestañeos

De forma complementaria a la prueba anterior, se comprueba la correcta grabación EEG. En cada levantamiento realizado por el brazo derecho se realizó un pestañeo, los cuales son visibles en los canales frontales del EEG.

Para poder visualizar los pestañeos es necesario crear una ventana temporal, cuyo *onset* corresponda al momento en el que se levanta el brazo derecho. Al comprobar anteriormente la correcta detección del levantamiento de los joysticks, se toma como punto predicho como el *onset*.

Por otro lado el timestamp de la señal deberá estar sincronizado con el timestamp del joystick, ya que al hacer uso de EEG se deberá de ser muy riguroso con los instantes. El timestamp de cada paquete es asignado en el momento en el que el amplificador envía dicho paquete al ordenador, al recibirse será el propio ordenador el que asigne el timestamp, correspondiendo este al momento de recepción con un cierto retardo debido al tiempo de vuelo del bluetooth. En cambio se posee en los paquetes recibidos del amplificador un número de secuencia asignado por el propio amplificador, será este número el que se tome para calcular el momento de recepción de cada paquete.

Se debe de tener en cuenta que los amplificadores que registran EEG toman 8 muestras, es decir una secuencia tendrá 8 muestras por cada canal, obteniendo en total 96 muestras por paquete. Cada canal tiene una frecuencia de 256Hz y cada secuencia una frecuencia de 32Hz. Sin embargo, el momento en el que se recibe cada secuencia

al ordenador conectado con el amplificador, viene alterado por el tiempo de vuelo del bluetooth que puede ser variable. Usando la siguiente fórmula, se calcula el tiempo de recepción teórico del paquete con secuencia 0, asumiendo una frecuencia de recepción constante.

$$t_0(t_i) = t_i - \frac{i}{32}, \forall i \tag{3.6}$$

Siendo $t_i$ el timestamp de la secuencia i y $t_0(t_i)$ el momento en el que se debería haber recibido el paquete con secuencia 0 dada una frecuencia de recepción constante.

Al realizar la media de todos los $t_0$ se obtiene el momento en el que se recibe el primer paquete minimizando la variabilidad de tiempo de vuelo del bluetooth.

$$T_0 = mean(t_0(t_i)), \forall i \tag{3.7}$$

Para poder sincronizar el evento producido en la RV con la señal EEG, se toma el timestamp del evento $t_{z_i}$ y se calcula a qué muestra $n_i$ de señal de EEG corresponde, utilizando lo siguiente:

$$n_i = (t_{z_i} - T_0) \cdot 256 - s_0 \cdot 8, \forall i \tag{3.8}$$

Siendo $t_{z_i}$ el momento del evento en RV (por ejemplo, levantamiento i-ésimo de la mano) y $s_0$ el número de la primera secuencia.

Tras realizar lo anterior para cada levantamiento, se toman las ventanas temporales de un segundo antes del levantamiento hasta los dos segundos después del levantamiento, siendo el momento 0 como el instante de detección del movimiento. Se muestra en la Figura 3.10 la señal del primer canal frontal del EEG alineada en torno al 0 de cada uno de los levantamientos.

Se observa en la Figura 3.10, antes del instante 0, la onda posee una amplitud casi nula. A partir de ese instante se produce un pestañeo y la amplitud se ve incrementada, en el momento en el que acaba el pestañeo la amplitud vuelve a valores cercanos a cero. La amplitud se ve aumentada tras unos milisegundos de la producción del onset, esto está relacionado tanto por la forma en la que el sujeto realizó el experimento, debido a que tras levantar el brazo pestañeaba, y por el retardo de 0.1464 segundos.
Además se observa un cierto ruido, ya que hay algunas ondas que poseen una amplitud diferente al resto de ondas, esto puede deberse a movimientos oculares o a ruido introducido en los sensores. Por lo general, en la gran mayoría de intentos correctos, en el momento en el que se empieza a subir el brazo, es cuando el usuario empieza la acción del pestañeo.

Finalmente se concluye la correcta grabación de datos EEG desde la experiencia virtual.

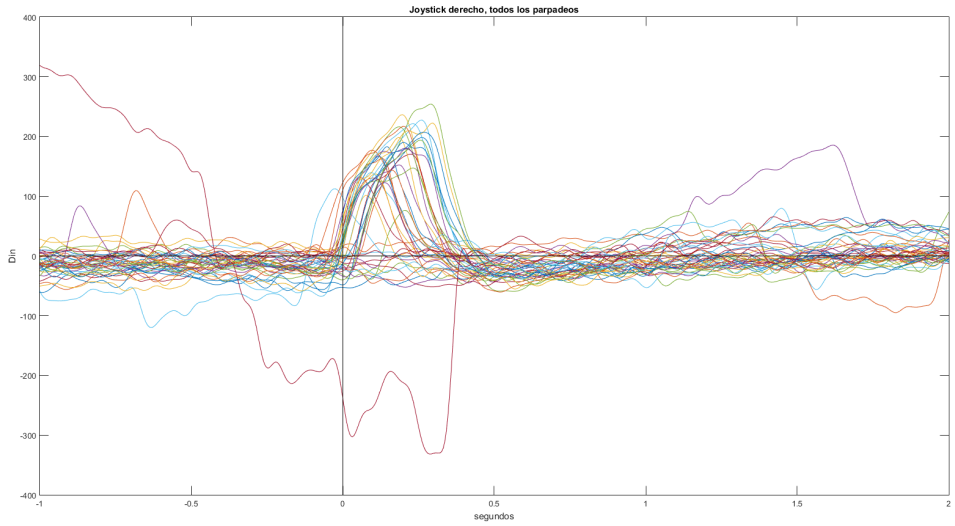



Figura 3.10: Primer canal EEG donde se observan los pestañeos producidos. El instante 0 corresponde al levantamiento del joystick derecho, en ese instante será cuando se produzcan los pestañeos, visibles por la alteración de la amplitud de la señal.

# Capítulo 4

# Protocolos de EEG en Realidad Virtual

Como ya se ha mencionado, la RV representará un escenario en el cual se realizará un protocolo EEG, con el objetivo de poder captar los correlatos neuronales (MRCPs y ERDs) y los potenciales relacionados con errores (ERPs) captados cuando el usuario perciba o realice un error. Producir tal potencial en un experimento físico puede convertirse en una tarea compleja. En cambio al hacer uso de la RV, es posible producir cualquier fenómeno que produzca algún tipo de efecto sobre el usuario.
En este proyecto, para poder producir un ERP, se ha hecho uso de un efecto espejo en los movimientos de los brazos de forma aleatoria. Se han creado dos protocolos para poder obtener el potencial.

## 4.1. Máquina de estados

Para poder estudiar los correlatos motores es necesario tener un margen de tiempo entre movimientos, ya que es necesario observar la intención de movimiento. Con 3 segundos de margen es suficiente. En el caso de los potenciales de error, interesa los instantes en el que se produce el movimiento.

La RV deberá de ser capaz de controlar dicho margen, para ello se ha definido la siguiente máquina de estados, visible en la Figura 4.1.

Donde cada estado corresponde con lo siguiente:

- $Q_0$ (Reposo): Estado inicial de la máquina de estados. Momento en el que los joysticks están en la posición inicio. En el momento en el que un joystick abandone la posición de inicio, será por la realización de un movimiento, provocando el abandono del presente estado.

- $Q_1$ (Movimiento): Momento en el que algún joystick está en movimiento. En el

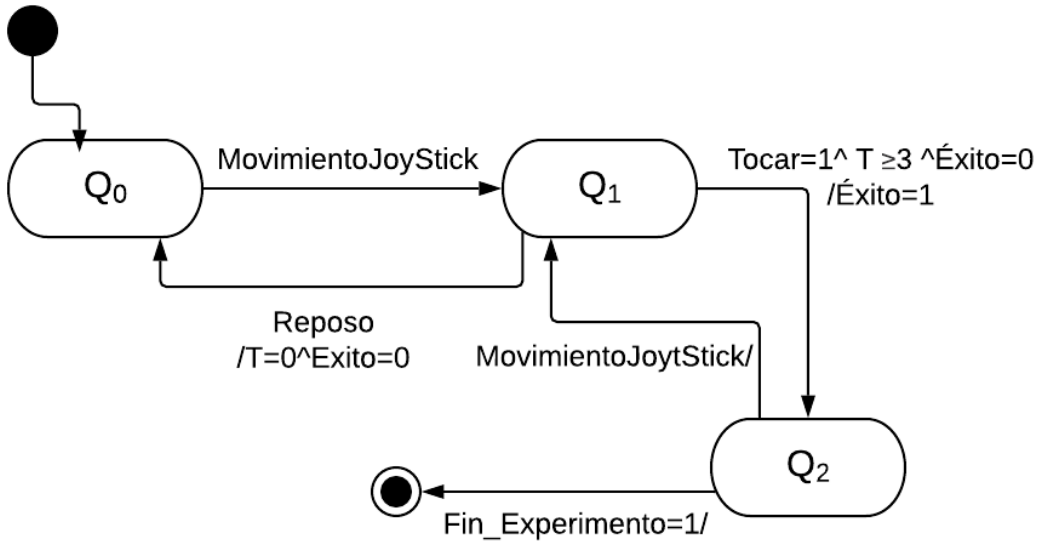



Figura 4.1: Máquina de estados creada para los protocolos EEG.

momento en el que se realice la tarea requerida en el protocolo, se comprueba si ha cumplido con el margen de tiempo estipulado de 3 segundos, en el caso de su cumplimiento se abandonará el estado $Q_1$ para adentrarse en el estado $Q_2$ incrementando la tarea realizada en el correspondiente joystick. En el caso de no cumplir con el margen la tarea sera ignorada. Cuando el joystick vuelva a la posición inicio el reloj será reseteado y dará lugar al estado $Q_0$.

- $Q_2$ (Toca): Estado en el que tras haber realizado la tarea, se comprobará si se ha completado todo el experimento, terminando la máquina de estados. En el caso en el que no se haya completado todo el experimento, se volverá al estado $Q_1$ ya que el joystick todavía se encuentra en movimiento.

## 4.2. Escenarios creados en los protocolos

Se diseñan dos protocolos cuyo objetivo es obtener tres resultados: correlatos motores (MRCPs y ERDs) asociados al movimiento locomotor realizado por el usuario durante el experimento y los potenciales de error (ERPs) (asociados a inconsistencias entre el mundo real y el virtual).

Como escenario común entre los dos protocolos, se decidió representar una habitación, donde el usuario se encontraría sentando enfrente de una mesa, en la cual es capaz de apoyar sus antebrazos. Para poder garantizar que el usuario se encuentra cómodo, el experimentos se realizará sentado enfrente de una mesa real, en la cual podrá

apoyar sus antebrazos, al igual que en la RV. Además podrá observar en la pared de enfrente los contadores del número de tareas realizadas de forma satisfactoria. Hay dos contadores, cada uno correspondiente a un joystick/mano.

### 4.2.1. Protocolo 1: alcance de altura.

Partiendo del escenario citado, se coloca una barra por encima de la vista del usuario, lo suficientemente larga como para que el usuario sea capaz de tocarla con cualquiera de los brazos mediante un levantamiento vertical. En la Figura 4.2 se presenta el escenario resultante para realizar este protocolo.

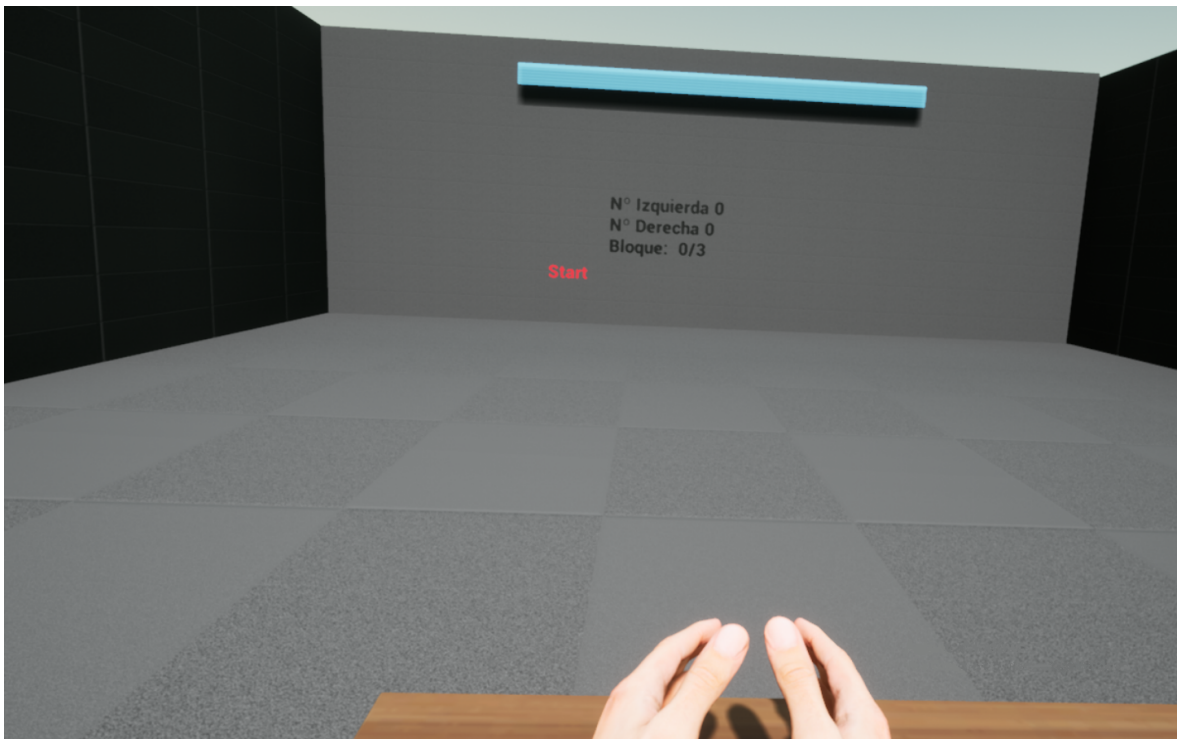



Figura 4.2: Escenario resultante del protocolo 1, alcance de altura.

La tarea que tendrá que realizar el usuario será tocar con una mano la barra colocada a la altura de los ojos del sujeto. Si la tarea se ha realizado respetando el margen de tiempo de reposo, podrá observar en el contador el incremento de la tarea realizada por la mano correspondiente.

Este protocolo consta de una fase de familiarización de 10 intentos por cada brazo, con esta fase se pretende buscar la familiarización del usuario con el entorno y la tarea requerida. Tras esta fase de familiarización se dará lugar a la fase de experimentación, compuesta por 3 bloques con 15 movimiento por cada brazo, habiendo un total de 30 movimientos en cada bloque. En la fase de experimentación estará activo el efecto espejo. Para evitar que el sujeto se acostumbre a tal efecto solamente aparecerá 10

veces en cada bloque, el efecto será activado de forma aleatoria. Cada vez que un bloque termine dará lugar a un periodo de descanso, cuya duración sera determinada por el usuario, siendo este el que de lugar al siguiente bloque. En la Figura 4.3 se muestra un esquema del protocolo.

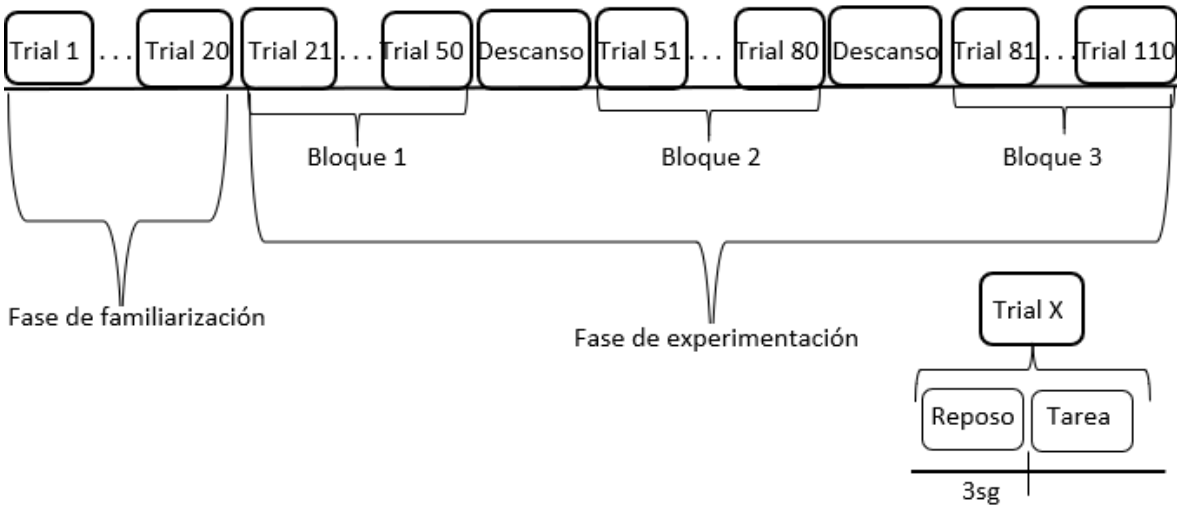



Figura 4.3: Proceso del protocolo 1, alcance de altura.

### 4.2.2. Protocolo 2: alcance de una posición x-z

Este protocolo pretende buscar una mayor complejidad en la tarea que desempeña el usuario. Debido a que el protocolo anterior puede llegar a convertirse en una tarea monótona, ya que en el momento en el que se produce el efecto espejo el usuario no tiene que realizar ningún tipo de corrección para cumplir con la tarea. Se añade a este nuevo protocolo la necesidad de corregir la trayectoria en tiempo real para poder cumplir con la tarea, haciendo que el usuario obtenga un efecto de confusión al realizar el movimiento en efecto espejo.
En este caso, se coloca una esfera, la cual cambiará de localización cada 5 tareas exitosas. Las posiciones establecidas de cada localización de la esfera será en la zona superior derecha de la escena y en la zona superior izquierda de la escena.
Se presenta en la Figura 4.4 el escenario resultante para realizar este protocolo.

La tarea que deberá desempeñar el usuario será levantar el brazo que se desee y dirigirse hacia la esfera, hasta tocarla. En el caso que se produzca el efecto espejo, tendrá que tocarla con el brazo controlado por la RV, haciendo que el usuario tenga que corregir la trayectoria planeada. Esta corrección producirá una percepción de error, lo cual conlleva a realizar un potencial de error.

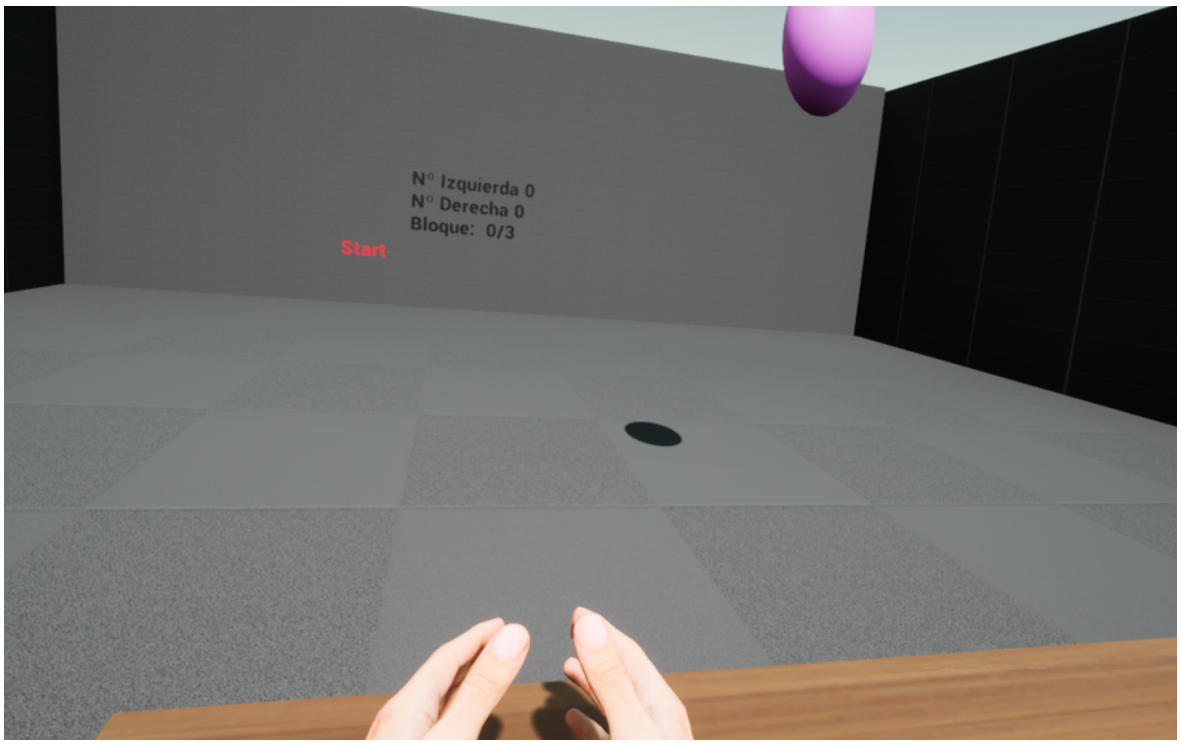



Figura 4.4: Escenario resultante del protocolo 2, alcance de una posición. Opción con la esfera colocada en la parte derecha de la escena.

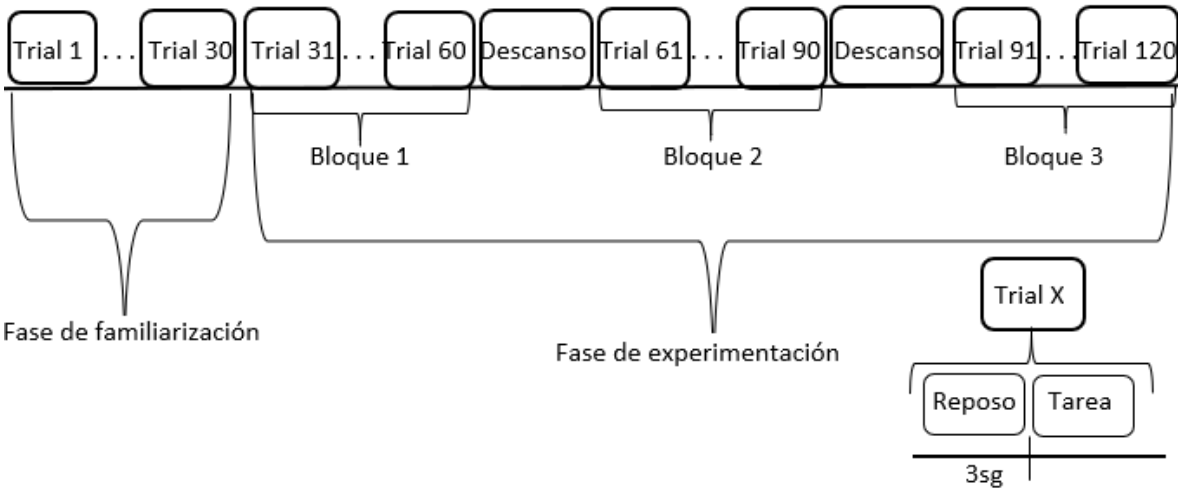



Figura 4.5: Proceso del protocolo 2, alcance de una posición

Este protocolo consta de una fase de familiarización de 15 intentos por cada brazo. Una fase de experimentación de 3 bloques, cada uno de ellos constituido por 15 movimientos de cada brazo, habiendo en total de 30 movimientos en cada bloque. En esta fase estará activo el efecto espejo. Para evitar que el sujeto se acostumbre a tal efecto solamente aparecerá 10 veces en cada bloque, la aparición de este efecto será aleatoria. Cada vez que un bloque termine, dará lugar a un periodo de descanso el cual terminará y dará lugar al siguiente bloque cuando el usuario lo desee. Se representa en

la Figura 4.5 un esquema del protocolo.

# Capítulo 5

# Adquisición y procesamiento de la señal

## 5.1. Adquisición de datos

Para la adquisición de datos se tomó primero el dispositivo usado en las pruebas anteriores, pero al realizar el análisis de datos se obtuvieron errores, ya que los sensores del dispositivo se movieron durante la grabación, debido a encontrarse el dispositivo en una fase de diseño. Se decidió hacer uso de un gorro de agua, de 16 sensores, cuya distribución es visible en la Figura 5.1. Encima de él se colocó las Oculus Rift, de esta manera se obtendría las señales EEG durante la experiencia de RV.
El cambio al uso del gorro de agua proporcionó varias mejoras como son: Un mayor número de sensores; una mejor distribución de los sensores sobre la corteza craneal; los sensores de húmedos captan menos ruido en comparación con los sensores de agua; los sensores secos se mueven más que los sensores húmedos.

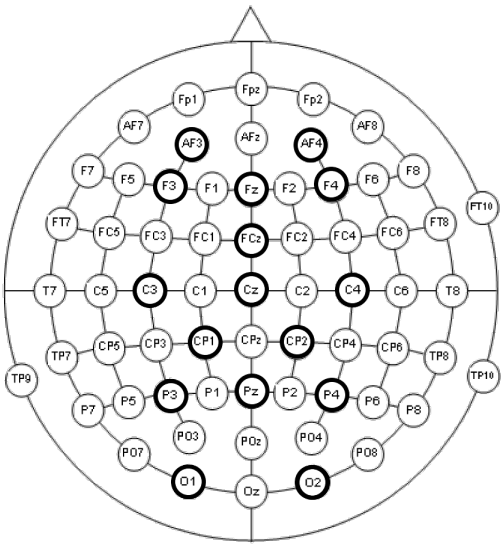



Figura 5.1: Distribución de los sensores sobre la cabeza para la adquisición de datos.

Las regiones del córtex dedicadas a las acciones motoras se encuentran los sensores F3, Fz, F4, FCz, C3, Cz y C4.

## 5.2. Experimentación

El experimento cuenta con cinco protocolos, que son los siguientes:

- Protocolo de calibración ocular: En este protocolo el sujeto deberá permanecer 30 segundos con los ojos cerrados, contabilizados por la persona que este supervisando el experimento. Tras ellos permanecerá 30 segundos con los ojos abiertos. Realizará 10 movimientos oculares de izquierda a derecha, 10 movimientos oculares de arriba a abajo y 10 pestañeos, los cuales deberán ser repetidos en el mismo orden 3 veces. Por último deberá permanecer 30 segundos con los ojos cerrados y 30 segundos con los ojos abiertos. El objetivo es poder obtener información sobre los artefactos oculares del sujeto, los cuales podrán estar presentes en los datos a estudiar, influyendo de forma negativa en el estudio en el caso de no ser eliminados.

- Protocolo de movimiento de abrir/cerrar mano: En este protocolo el sujeto se colocará en su mano derecha un fotodiodo, de tal manera que cuando cierre la mano el fotodiodo quede totalmente oculto, mientras que cuando abra la mano el fotodiodo quede al descubierto. De esta forma se provocará un cambio en la amplitud de la señal del fotodiodo, relacionado con el momento de abrir y cerrar mano. El sujeto deberá de realizar las acciones de abrir y cerrar 60 veces cada una, dejando 5 segundos de margen en cada movimiento. El objetivo de este protocolo, es la comprobación de la correcta captación EEG en el experimento, ya que es un movimiento con resultados visible en los correlatos motores.

- Protocolo 1, movimiento de alance de altura : En este protocolo se deberá de hacer uso de la experiencia virtual, este protocolo corresponde con el ya comentado en la sección **4.2.1 Protocolo 1: alcance de altura**.

- Protocolo tocar esfera: Este protocolo al igual que el anterior, se deberá de hacer uso de la experiencia virtual, este corresponde con el ya comentado en la sección **4.2.2 Protocolo 2: alcance de una posición x-z**.

- Protocolo de calibración ocular: Protocolo con las las mismas directrices del primer protocolo del experimento, con el objetivo de poseer una mayor información sobre los artefactos oculares.

Durante los protocolos de alcance de altura y alcance de una posición, el sujeto debe cumplir ciertas normas, las cuales son:

- Tener apoyados los antebrazos en la mesa colocada enfrente suyo. Esto dará una sensación mayor de realismo en la RV ya que en ella observará también que sus brazos virtuales están apoyados en una mesa.
- Para poder comenzar el experimento en la RV debe pulsar el botón B del joystick derecho, con el cual se calculará donde se encuentra el punto de referencia del avatar respecto a la posición del sujeto. Este punto será utilizado en el efecto espejo.
- Durante el proceso en el que el sujeto realiza la tarea, entendiendo esta como el momento en el que levanta un brazo y se dirige a completar la tarea encargada en el protocolo correspondiente, no podrá pestañear.
- Cuando se encuentre en la posición de reposo, podrá pestañear, pero no podrá abandonar la posición de referencia.
- Cuando se encuentre el sujeto en el periodo de descanso entre bloques podrá realizar el movimiento que desea, tanto locomotores como oculares, ya que cuando tenga que volver a reanudar el experimento tendrá que pulsar el botón B del joystick derecho, donde se calculará el punto de referencia del avatar respecto a la posición del sujeto.

El cumplimiento de estas normas mejorará los correlatos motores y los potenciales de error.

## 5.3. Procesado de la señal

Para poder realizar el procesamiento de la señal, se ha tomado la herramienta de Matlab para poder obtener dos correlatos neuronales; la desincronización asociada a eventos (ERDs) y los potenciales corticales relacionados con el movimiento (MRCPs), y al generar un error al usuario se estudiarán los potenciales relacionados con el error (ERP) generados por el usuario, para poder realizar posteriormente los grandes averages de las señales de todos los usuarios.

Para poder realizar de forma correcta los tres estudios citados es necesario completar la señal EGG con los eventos generados, correspondiendo al brazo que realizó el movimiento, dependiendo del brazo movido la actividad neuronal se deberá de observar en determinadas regiones del córtex. Además, al estudiar también los potenciales de

error, es necesario saber si el movimiento realizado por el sujeto, fue representado en la RV con efecto espejo o no.

Los MRCPs muestran cambios en la amplitud de EEG, los cuales se encuentran en frecuencias comprendidas entre 0.3HZ a 3HZ. Para poder estudiar un MRCP es necesario observar instantes anteriores al evento (*onset*), donde la amplitud de la señal empieza a decrecer antes de que suceda el *onset*, aproximadamente 2 sg antes, tras producirse el *onset* la amplitud de la señal crece.

Los ERDs [16] se encuentran en frecuencias comprendidas entre 8 Hz a 30 Hz, ya que tienen en cuenta las bandas alfa [8-12 Hz] y betas[12-30Hz]. La intención de los ERDs es estudiar la preparación motora, es decir, la intención de movimiento. Para ello es necesario que la ventana temporal comience 2 sg antes del *onset*, ya que en los movimiento voluntarios hay una desincronización entre alfa y beta cuyo comienzo es de 2 sg antes del *onset*. Esa desincronización es relativa a la actividad previa tomada como referencia (baseline). Normalmente el baseline se considera cuando empieza la ventana temporal del ERDs y termina 1 segundo antes al *onset*. El ERD se obtiene de la siguiente forma para cada canal EEG y banda de frecuencia.

$$\%ERD = \frac{A - R}{R} \cdot 100 \tag{5.1}$$

Siendo R el baseline y A la energía de la señal.

Por último mediante los ERP se estudia la diferencia en la amplitud generada por las señales pertenecientes a los eventos con error y sin error. Usualmente a los 300ms se percibe el error, lo que se traduce a una alteración en la amplitud. Para poder medir los ERPs se realiza un filtrado paso banda de 1 Hz a 10 Hz, usando una ventana temporal de 0.2 sg a 1 sg, ya que en este caso solo interesa a partir del *onset*.

Para poder realizar MRCPs, ERDs, ERPs, existe un toolbox disponible en Matlab llamado EEGLAB, con el cual se puede procesar EEG. Para poder hacer uso de EEGLAB se puede optar por su GUI, lo cual es útil como una primera toma de contacto, aunque también se ofrece en código abierto, lo cual facilitará la automatización de todo el análisis EEG además de poder realizar manipulaciones de los datos EEG con funciones propias de Matlab, y usar su resultado en EEGLAB.

### 5.3.1. Preprocesamiento de la señal

Durante el procesamiento de datos se observo la existencia de ruido en la señal, este ruido podía ser resultado de dos problemas.

Por un lado el ruido puede ser resultado de artefactos estereotipados realizados por el sujeto durante el experimento como son: Pestañeos; movimientos oculares por la búsqueda de la barra o de la esfera colocada en la escena virtual, o la búsqueda del brazo controlado por la reflexión. Estos artefactos estereotipados provocan ruido que se propaga por el resto de canales EEG, influyendo negativamente al analizar los correlatos motores siendo necesario la eliminación del ruido. Usualmente se hace uso del Análisis de componentes independientes (ICA), en el cual se localizan artefactos estereotipados, permitiendo su posterior eliminación en la señal.

ICA es un técnica en la cual dado un conjunto de datos se pretende reducir la dimensión del espacio de los datos. El nuevo espacio estará constituido por los componentes que mayor información de la estructura de datos contengan. El objetivo de ICA es encontrar los componentes para poder realizar la reducción de dimensión. Al hacer uso de ICA para la deteccion de artefactos se consigue encontrar los componentes que contengan una mayor cantidad de ellos para poder eliminarlos.

EEGLAB ofrece de forma predeterminada el uso de ICA para poder eliminar artefactos, mediante la eliminación de los componentes que mayor información sobre artefactos contengan.

Existe un plugin desarrollado por Jason Palmer, llamado *Adaptive Mixture Independent Component Analysis* AMICA, el cual proporciona mejor resultados que ICA [17]. Debido a que dados unos datos de entrenamiento se realiza una descomposición ICA para múltiples modelos ICA. Y realiza la probabilidad en los datos de validación en cada modelo a que componente pertenecen, para poder obtener un mayor refinamiento hacen uso de la función de densidad de probabilidad (PDF) a cada componente [18].

Se ha realizado un prueba de cómo puede afectar el uso o no uso de AMICA. Se tomó el movimiento de abrir y cerrar la mano derecha, tomando como evento el de cerrar la mano, el resultado se muestra en la Figura 5.2.

Donde se observa en el caso de no aplicar AMICA la señal es mayormente plana, al eliminar los artefactos provoca que la señal no sea plana y posea una pendiente negativa en el momento que ocurre el *onset*, relacionado por el movimiento de cerrar. Los motivos por lo que la señal sea plana en el caso de no aplicar AMICA se debe a artefactos, ya que estos son propagados en los sensores recibiendo en algunos una cantidad superior de ruido que en otros sensores, al realizar la media de todos los trials del experimento, poseyendo algunos de ellos ruido, hace que al realizar la media de todos los trials se compensen las amplitudes, obteniendo una media cercana a cero.

Por otro lado el ruido puede ser procedente de sensores cercanos, para ello se aplican

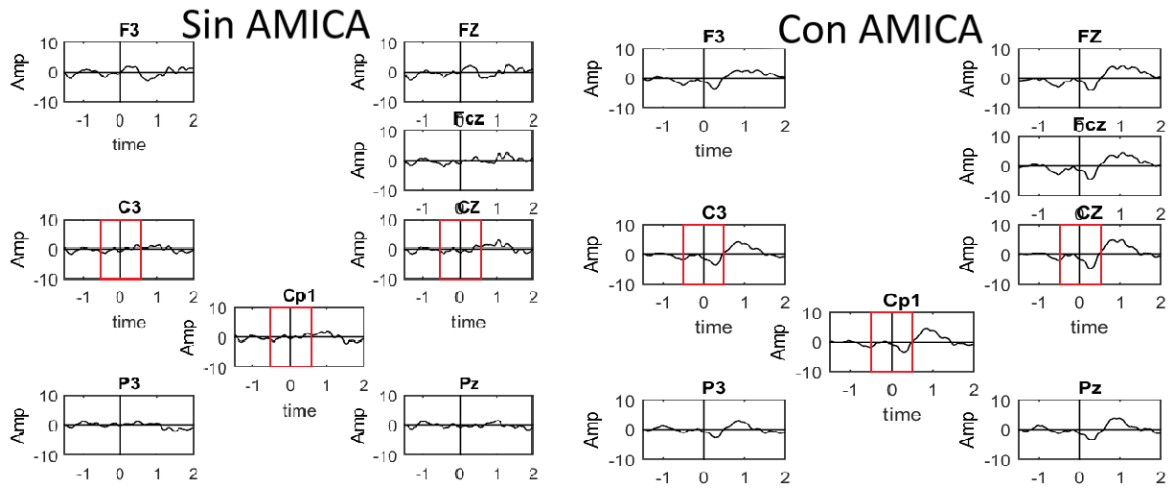



Figura 5.2: Diferencia de aplicar y no aplicar AMICA en la señal EEG. Se obtiene al aplicar AMICA una mejora en la señal, debido a la eliminación de artefactos que provocaban una atenuación en la amplitud de la señal.

filtros espaciales en los cuales se tienen en cuenta los sensores cercanos, algunos de los filtros espaciales utilizados son el filtro de la media, o el filtro CAR (*Common Average Reference*).

Se ha realizado tres procesamientos distintos, en uno se ha aplicado el filtro de media, en otro el filtro CAR y en otro no se ha aplicado ningún filtro, ya que puede no existir ruido durante el experimento. El resultado será indicado en el capítulo 6.

Filtro de media, este filtro cosiste en coger una ventana 3x3, en la cual se encuentran los vecinos $V$ del sensor del punto central de la ventana, (2,2). Realizando lo siguiente para cada sensor $c$ en cada instante de tiempo $t$.

$$Y_{c,t} = X_{c,t} - \overline{M_{c,t}}, \forall c \in C \, \forall t \in T \tag{5.2}$$

Donde $X_{c,t}$ es el valor del sensor $c$ en un instante del tiempo $t$, $\overline{M_{c,t}}$ es la media de los vecinos del sensor $c$ en un instante del tiempo $t$ e $Y_{c,t}$ es el valor resultante del filtrado de la media en el sensor $c$ en el instante $t$.

Filtro CAR (*Common Average Reference*: este filtro consiste en que al valor de la señal de cada sensor en un instante del tiempo se le resta la media de todos los sensores en ese instante del tiempo. Quedando de la siguiente forma:

$$Y_{c,t} = X_{c,t} - \frac{1}{|C|} \cdot \sum_{i=1}^{C} X_{i,t}, \forall c \in C \, \forall t \in T \tag{5.3}$$

Donde $T$ representa el tiempo, $C$ representa el conjunto de sensores, $X_{c,t}$ la señal original de un sensor $c$ en un instante $t$ e $Y_{c,t}$ la señal filtrada en el instante $t$ para el sensor $c$.

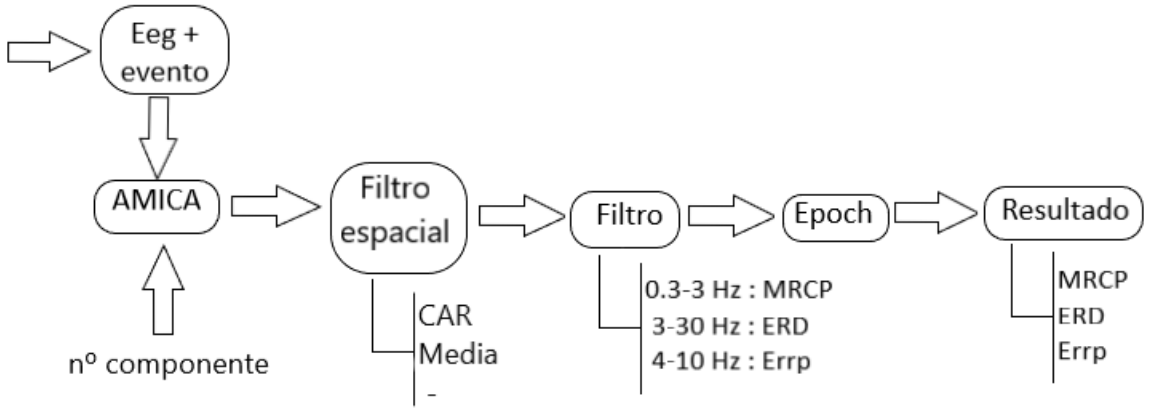



Figura 5.3: Pipeline para el procesamiento y preprocesamiento de la señal EEG para el cálculo de los correlatos neuronales y los potenciales de error.

Finalmente, se calculan los MRCPs,ERDs y los ERPS de cada sujeto siguiendo el pipeline detallado en la Figura 5.3.

Debido a que los filtrados realizados por el filtro espacial son calculados para cada instante de tiempo, al calcular los ERDs,MRCPs y ERPs es necesario realizarlos dentro de una ventana temporal (*epoch*), para ello tras realizar el filtrado espacial y el correspondiente filtrado en la señal, la señal EEG tendrá que ser segmentada en varias ventanas temporales. En el caso de los ERDs y MRCPs es necesario hacer uso de instantes anteriores al onset, por lo se necesita una ventana temporal que trascurra desde un cierto tiempo anterior al onset hasta un cierto tiempo posterior al onset, el número total de ventanas temporales obtenidas será el mismo número de trials que se hayan producido en el experimento.

Una vez que se poseen los correlatos motores y los potenciales de error de cada sujeto, se realizan los grandes averages de los tres correlatos obtenidos de los sujetos.

Se tomaron cuatro sujetos para la realización del experimento, usando el gorro de agua para la captación de las señales EEG. Realizaron el experimento detallado en la sección 5.2, realizándose en una media de 46 minutos.

Seis sujetos realizaron los protocolos de alcance de altura y de alcance de posición utilizando el dispositivo con sensores secos. Sin embargo, los resultados tuvieron que ser desechados ya que se el dispositivo no se ajustaba del todo bien a la cabeza y provocaba movimientos en los sensores, debido a encontrarse en proceso de desarrollo por el equipo de Bit&Brain. Estos movimientos de los sensores impedían una correcta medición de la señal EEG.

# Capítulo 6

# Resultados

Para poder comprobar la correcta captación EEG se analizan primero los correlatos motores obtenidos del protocolo de abrir y cerrar la mano derecha. Los movimientos realizados por la mano derecha provoca una actividad cerebral visible en el córtex izquierdo, siendo los canales F3, C3 y Cp1 donde se observan los correlatos motores.

En el caso de los MRCPs, Figura 6.1, se observa que en el momento en el que sucede en el *onset*, la amplitud posee una pendiente negativa, poseyendo el mínimo global después del *onset*, lo cual es correcto ya que la señal empieza a decrecer cuando se produce la intención de movimiento, y una vez que ocurre la amplitud aumenta.

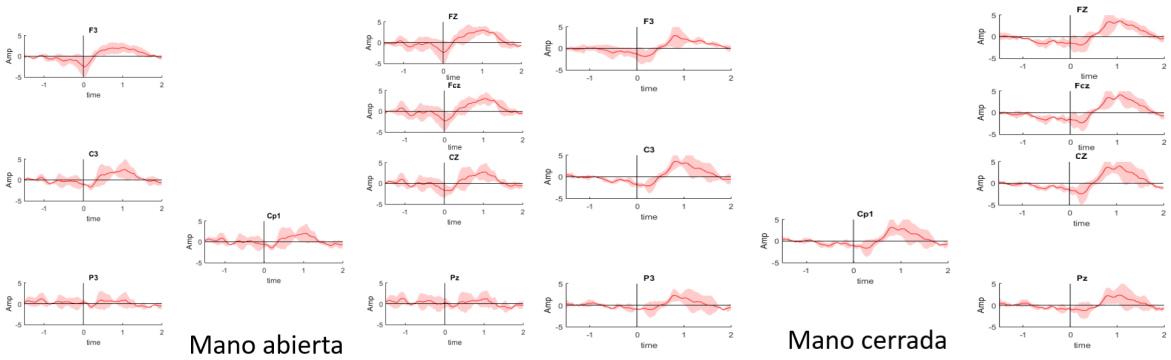



Figura 6.1: Grandes averages de los MRCPs de todos los sujetos del protocolo de abrir y cerrar la mano, la desviación estándar está sombreada sobre la media. Se observan claros ejemplos de MRCPs en los canales F3, C3, CZ, FCZ, donde instantes anteriores al onset(t=0) empieza a decrecer la amplitud de la señal tomando su punto mínimo en el momento en el que se produce el onset o milisegundos después de producirse, tras este punto la amplitud de la señal crece.

En el caso de los ERDs, Figura 6.2, se observa una mayor desincronización en el canal de C3, en las bandas superiores de alfa [10-12 Hz] y en beta [12-30Hz], la cual comienza instantes anteriores al onset y continúa a lo largo del movimiento, lo cual es lo esperado en un ERD.

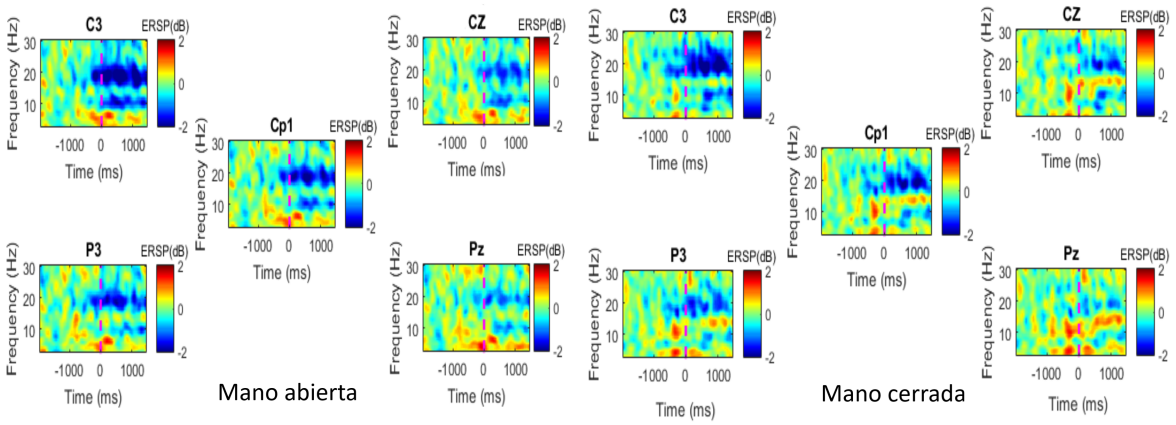



Figura 6.2: Grandes averages de los ERDs de todos los sujetos del protocolo de abrir y cerrar la mano. Se visualiza la desincronización, color azul, en los canales C3,CP1 y P3 en las bandas superiores de alfa e inferiores de beta. La desincronización empieza instantes anteriores a producirse el onset(t=0) y continúa a largo del movimiento.

Tras observar la detección de los correlatos neuronales relacionados con movimientos de las manos, se puede analizar los correlatos relacionados con los movimientos realizados en los protocolos de alcance de altura y alcance de posición. La actividad neuronal producida por el movimiento del brazo derecho provoca la activación del córtex izquierdo, donde se encuentran los canales F3, C3 y Cp1. Los movimientos realizados por el brazo izquierdo provocan la activación del córtex derecho, donde se encuentran los canales F4, C4 y Cp2. Estos canales serán los estudiados en los MRCPs y ERDs.

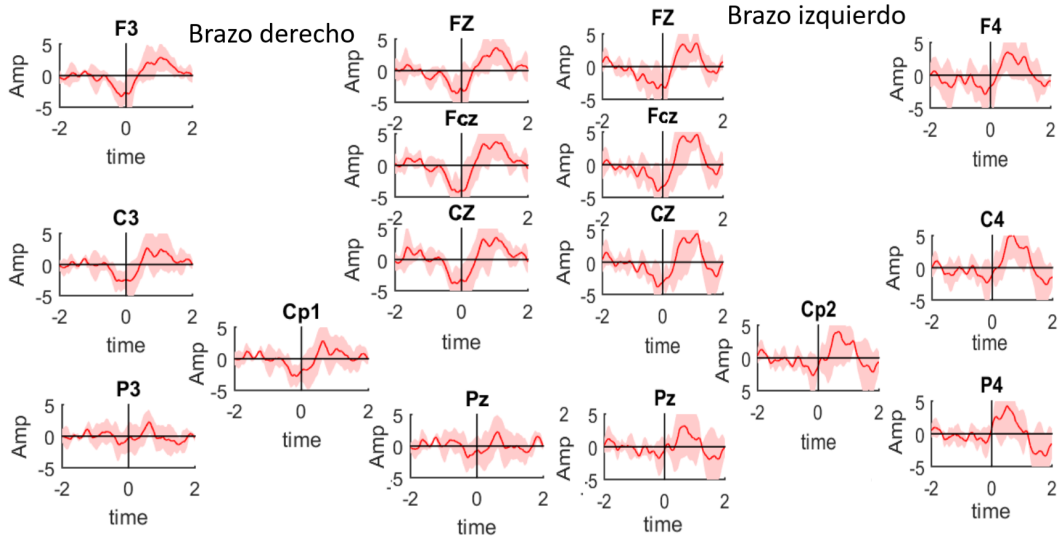



Figura 6.3: Grandes averages de los MRCPs de todos los sujetos del protocolo de alcance de altura sin filtro espacial, la desviación estándar está sombreada sobre la media. Se observa en los canales C3,C4,CZ,Cp1,Cp2 y FCz como la señal decrece medio segundo antes del onset(t=0), al producirse la señal toma su mínimo valor y crece la amplitud.

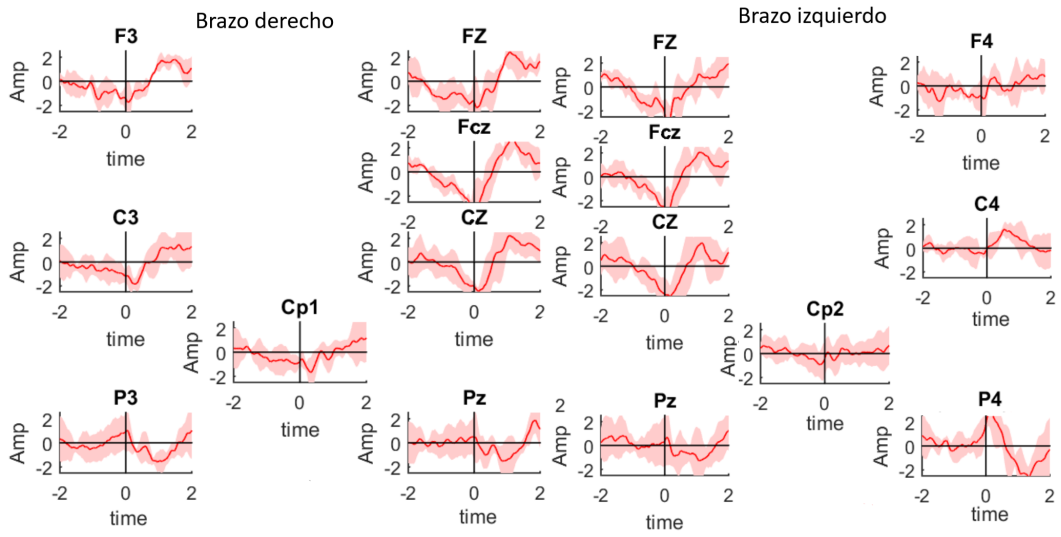



Figura 6.4: Grandes averages de los MRCPs de todos los sujetos del protocolo de alcance de altura con filtro de la media, la desviación estándar está sombreada sobre la media. Se observa en los canales C3,CZ y FCz como la señal decrece un segundo antes del onset(t=0), a su producción la señal toma su mínimo valor, tras el cual la amplitud crece. En el caso del canal C4 presenta el mismo comportamiento, pero en este canal posee una amplitud menor al resto.

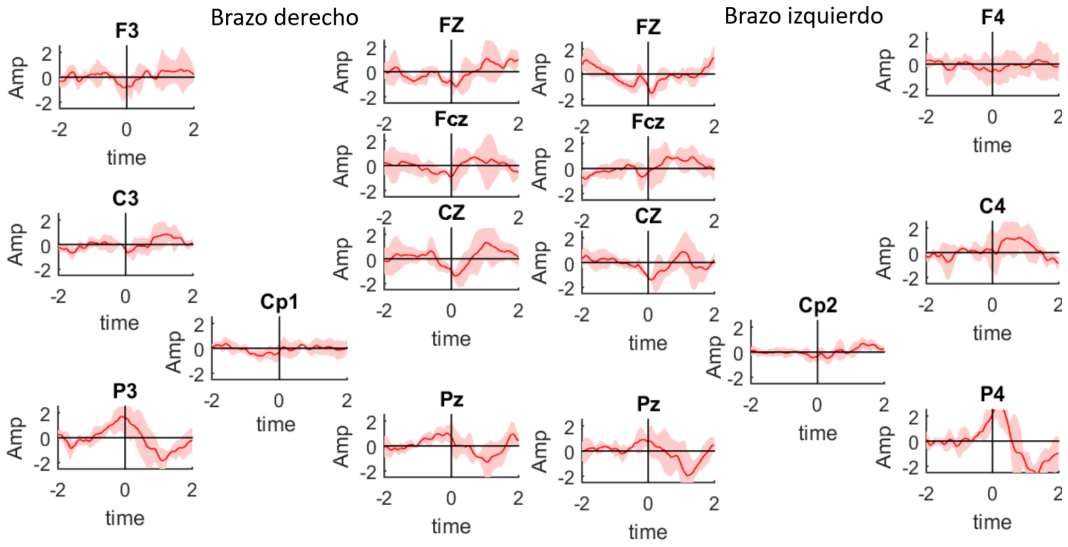



Figura 6.5: Grandes averages de los MRCPs de todos los sujetos del protocolo de alcance de altura con filtro CAR, la desviación estándar está sombreada sobre la media. Se observa como la señal se ha atenuado por la aplicación del filtro CAR, solamente es visible los MRCPs en el canal CZ, donde menos de medio segundo anterior al producirse el onset(t=0) la amplitud de la señal decrece, la señal toma valor mínimo en el momento en el que se produce el onset tras el cual la amplitud aumenta.

En el caso del protocolo del alcance de altura, el movimiento realizado sera un levantamiento del brazo de forma vertical. Se muestran los resultados obtenidos en los MRCPs sin aplicar ningún filtro espacial, Figura 6.3; aplicando el filtro de la media,

Figura 6.4 y el filtro CAR, Figura 6.5.

La primera diferencia observada en los MRCPs del protocolo de alcance de altura, en el filtro CAR la amplitud de la señal de los canales C3,C4,Cz, Cp1 y Cp2 se ha atenuado, esto es consecuencia por realizarle el decremento de la media del resto de sensores, ya que al tener una amplia distribución, algunos sensores tendrán actividad ajena a la actividad generada por el movimiento del brazo.
En el caso del filtro de la media se obtiene una señal similar a la señal original, con cambios en determinados canales (como son C4,C3,Cp1) donde en el momento del onset, la señal obtiene su mínimo global instantes siguientes al onset, en cambio al no aplicar ningún filtro este mínimo esta alineado al onset o instantes anteriores al onset, lo cual corresponde a lo esperado.

Para completar el análisis de los correlatos neuronales se estudian los ERDs de cada filtro, visibles en las Figuras 6.6, 6.7 y 6.8.

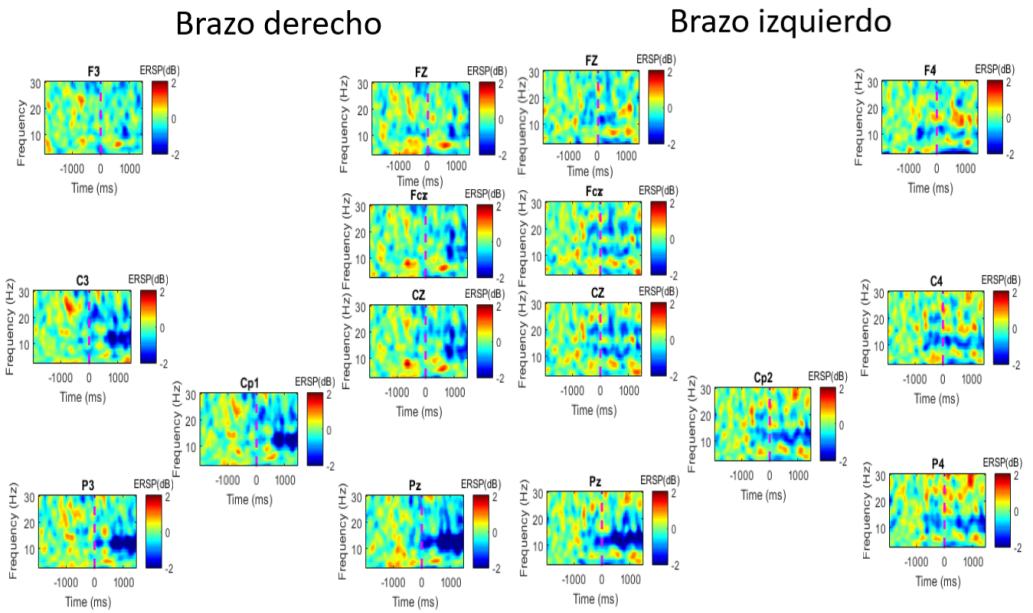



Figura 6.6: Grandes averages de los ERDs de todos los sujetos del protocolo de alcance de altura, sin filtro espacial. Se observa en los canales C3 y C4, la existencia de algo de desincronización(color azul), pero no es lo esperado por un ERD, ya que la desincronización no empieza antes del onset(t=0) y continúa a lo largo del movimiento.

Como resultado en los ERDs, en el caso de no aplicar ningún filtro se obtiene en el canal C4, del momento del brazo izquierdo una desincronización que empieza antes del onset y continúa después de este, lo cual se relaciona con la intención y la realización del movimiento, esto se puede relacionar con el resultado del correspondiente MRCP, Figura 6.3, donde la señal del canal C4 se veía decrementada instantes anteriores al onset debido a la intención de movimiento y aumentada después del onset.

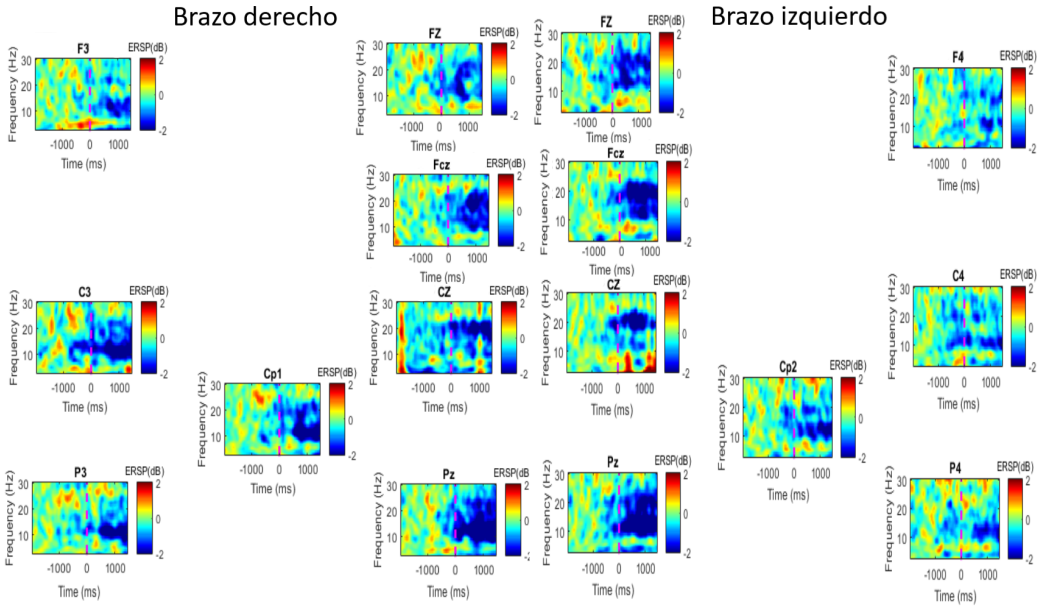



Figura 6.7: Grandes averages de los ERDs de todos los sujetos del protocolo de alcance de altura con filtro de la media. Se observa en los canales C3, C4, Fcz, Cz, Cp1 y Cp2. Desincronizaciones (color azul) por los valores superiores de alfa y en los valores de beta, empiezan instantes anteriores al onset(t=0) y continúa a lo largo del movimiento

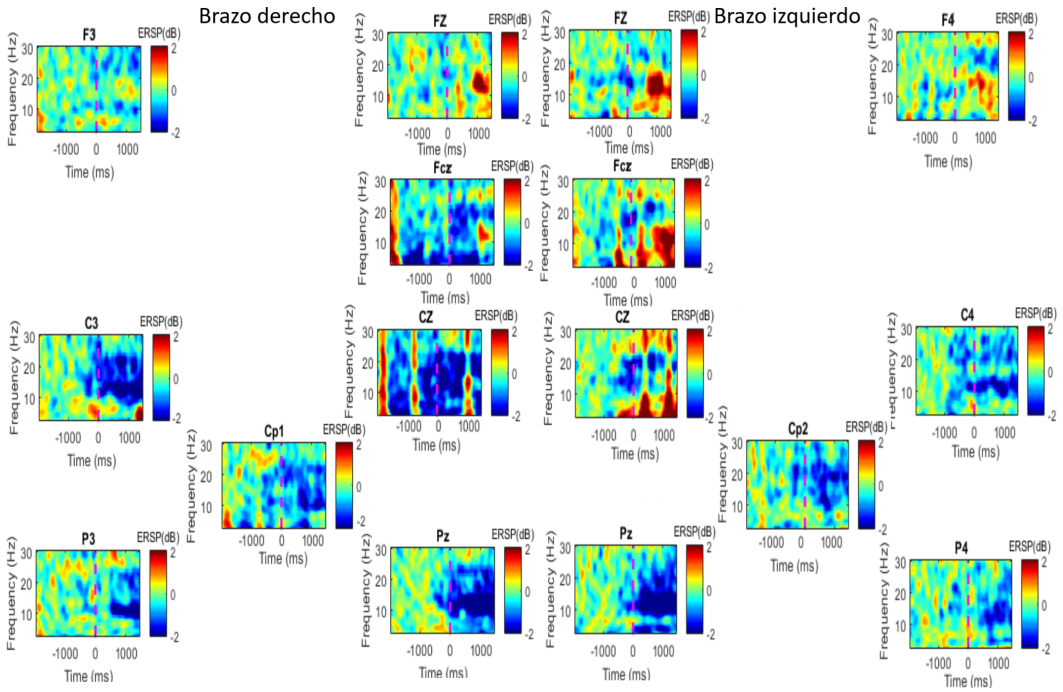



Figura 6.8: Grandes averages de los ERDs de todos los sujetos del protocolo de alcance de altura con filtro CAR. Se observa en los canales de C3 y C4 la desincronización (color azul) en los valores superiores de alfa y en los valores de beta, empiezan en instantes anteriores al onset(t=0) y continúa a lo largo del movimiento.

En el caso de aplicar el filtro de la media, se obtienen en los canales C3, C4 se observa la desincronización antes de que suceda el onset y continúa en el tiempo, proporciona

un mejor resultado que al no aplicar ningún filtro espacial. Por último en el caso del filtro CAR también es visible el fenómeno, pero en el caso del movimiento del brazo derecho en el canal Cz no se obtiene lo esperado.

Para poder hacer el estudio de los potenciales de error se toma el filtro de la media, ya que ha obtenido mejores resultado al estudiar los correlatos neuronales del movimiento. Los potenciales de error son visibles en la Figura 6.9.

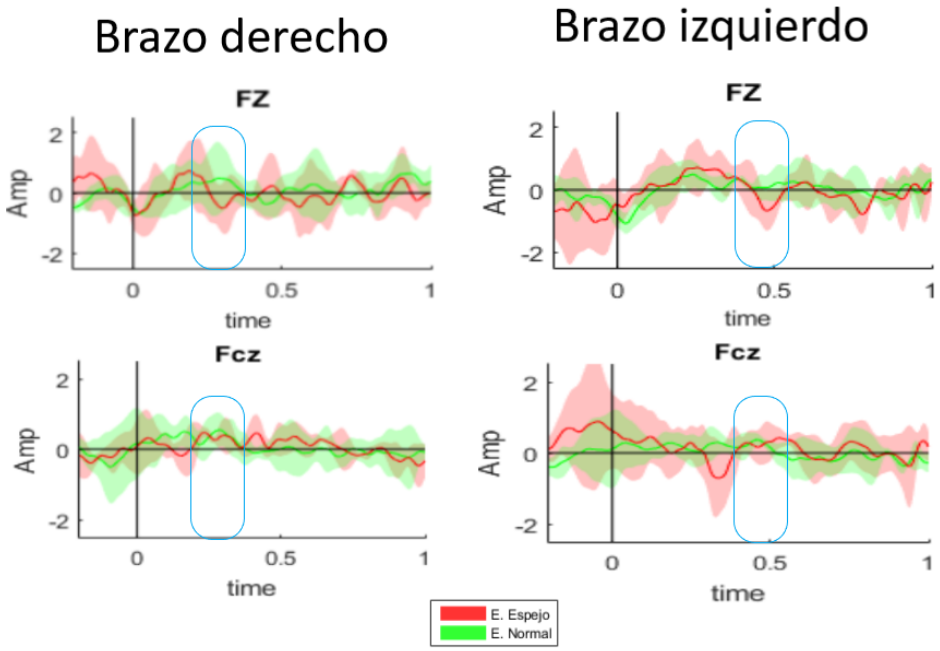



Figura 6.9: Grandes averages de los ERPs de todos los sujetos del protocolo de alcance de altura usando el filtro de la media. Se marca la desviación estándar de cada efecto (efecto normal, color verde, efecto espejo, color rojo) de forma sombreada sobre su correspondiente media. Marcado en azul los instantes donde se produce la diferencia en la amplitud al producirse el error y no producirse. Se observa como la amplitud del canal Fz, canal en el cual debe observarse mejor el error producido, el error decrece en el momento en el que se detecta el error.

El error es percibido por el usuario tras unos instantes del onset, el instante en el que suele suceder este error será tras los 200 ms o 500 ms, en la Figura 6.9 se puede visualizar este instante con un error más marcado en el canal Fz mientras en Fcz se atenúa.

En el caso del protocolo de alcance de una posición, primero se estudian sus MRCPs, donde se observará la intención de movimiento, estos MRCPs deberían ser relativos a los obtenidos en el protocolo anterior, ya que el movimiento es similar a excepción de la dirección del brazo, ir hacia la derecha o ir hacia la izquierda o una trayectoria vertical.

Al igual que lo realizado en el caso anterior se muestran los MRCPs sin aplicar filtro espacial, Figura 6.10, filtro de la media, Figura 6.11 y el filtro CAR, Figura 6.12.

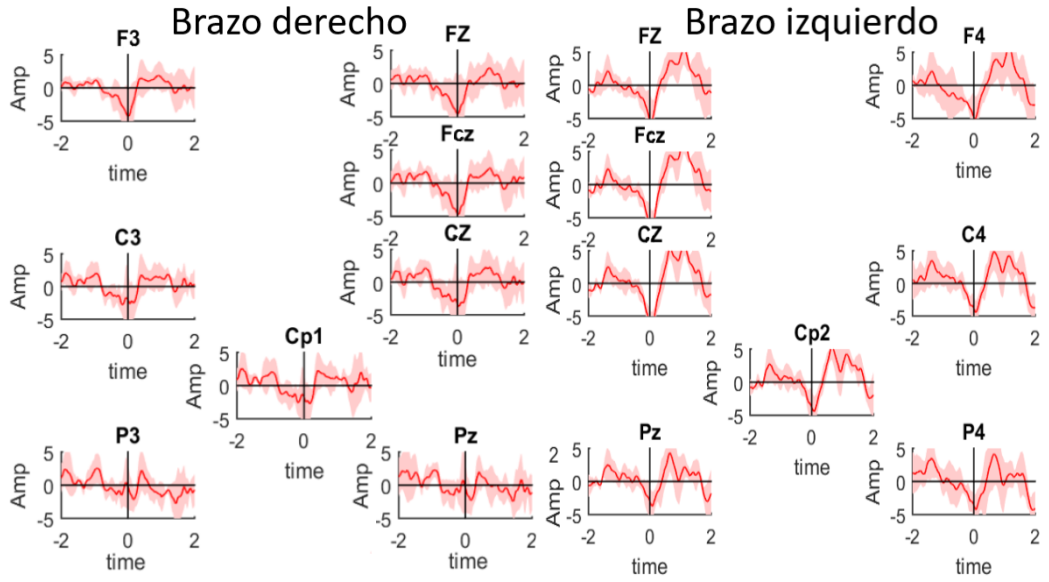



Figura 6.10: Grandes averages de los MRCPs de todos los sujetos del protocolo de alcance de una posición, sin filtro espacial. La desviación estándar está sombreada sobre la media. Se observa en los canales C3, Cz, C4, Fcz, Cp1 y Cp2 como instantes anteriores al onset(t=0) la amplitud de la señal decrece, alcanzando el valor mínimo en el onset, tras este punto la señal crece.

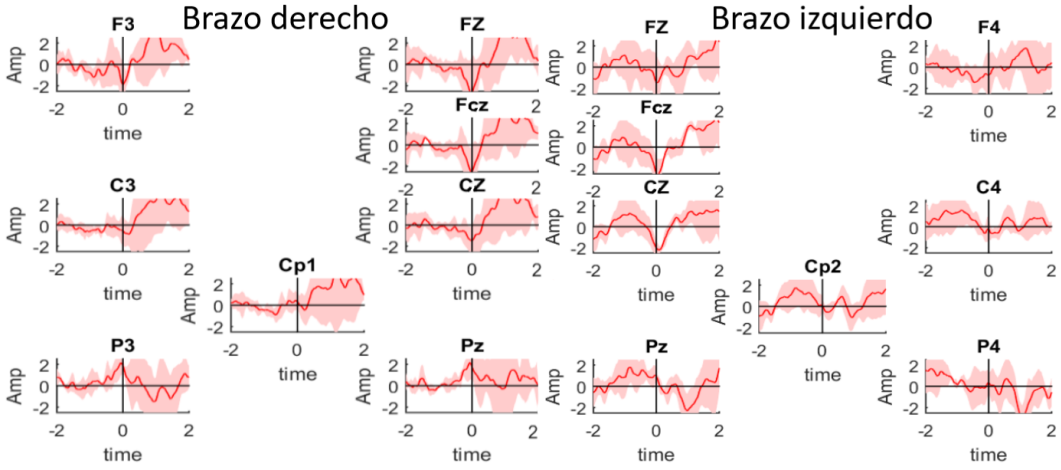



Figura 6.11: Grandes averages de los MRCPs de todos los sujetos del protocolo de alcance de una posición utilizando el filtro de la media. La desviación estándar está sombreada sobre la media. Se observa en los canales C3, Cz, C4 y Fcz como la señal decrece instantes anteriores al producirse el onset(t=0), tras su producción la amplitud de la señal toma su valor mínimo, tras este momento la amplitud aumenta.

Los resultados de los MRCPs del protocolo de alcance de posición son similares al protocolo de alcance de altura, ya que se analiza el mismo tipo de movimiento.

Se estudia el resultado obtenido en los ERDs, aplicando los tres filtros, visibles en las Figuras 6.13, 6.14 y 6.15.

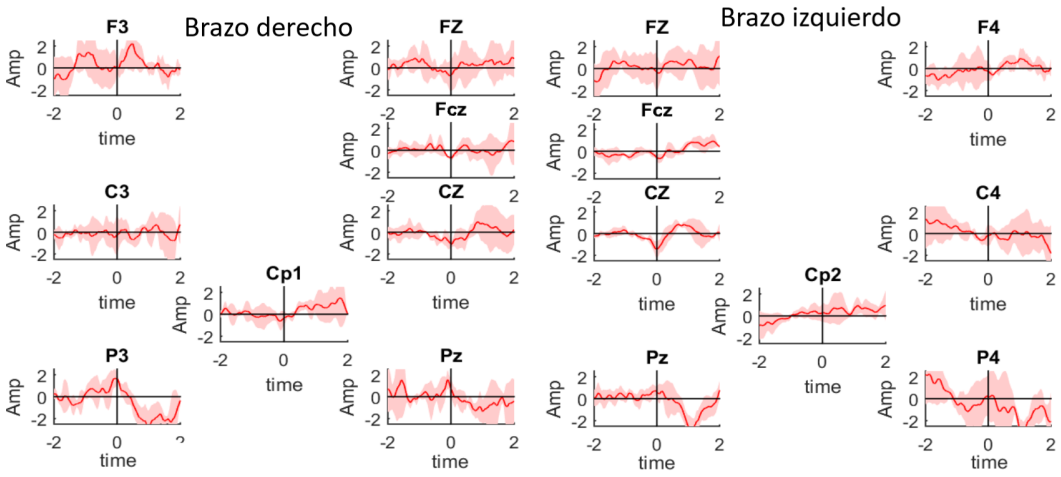



Figura 6.12: Grandes averages de los MRCPs de todos los sujetos del protocolo de alcance de una posición utilizando el filtro CAR. La desviación estándar está sombreada sobre la media. Se observa en el canal Cz como la señal decrece instantes anteriores al producirse el onset(t=0), tras la producción de este la amplitud de la señal toma su valor mínimo, tras este momento la amplitud aumenta. Los demás canales poseen una señal con una amplitud muy baja, consecuencia del filtro CAR.

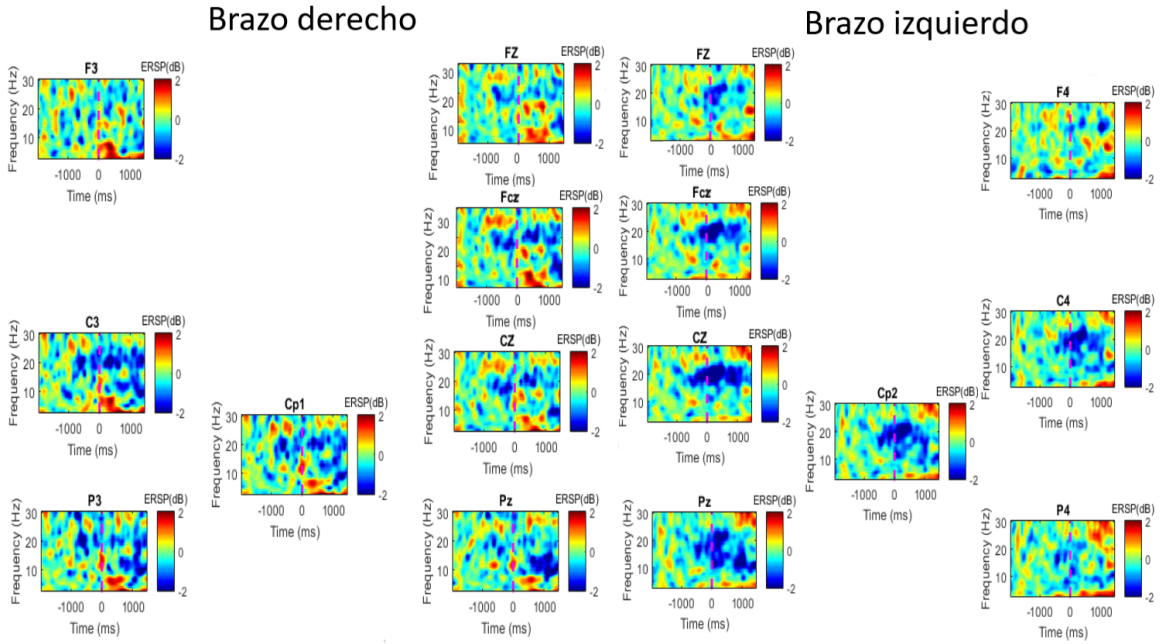



Figura 6.13: Grandes averages de los ERDs de todos los sujetos del protocolo de alcance de una posición, sin filtro espacial. Se observa algo de desincronización (color azul) en los canales Fcz, C4 y C3, en el caso del brazo derecho, esta desincronización no se realiza de forma continua a lo largo del tiempo. En el caso del brazo izquierdo la desincronización es más clara, se observa en los canales C4, Cz y Fcz donde la desincronización empieza antes del onset(t=0) y continúa a lo largo del movimiento.

Como resultado de los ERDs del protocolo de alcance de posición, en el caso de no aplicar ningún filtro espacial, no se observa ninguna desincronizacion en el canal C3 del brazo derecho, entre las bandas de 8Hz a 15 Hz correspondientes a alpha y

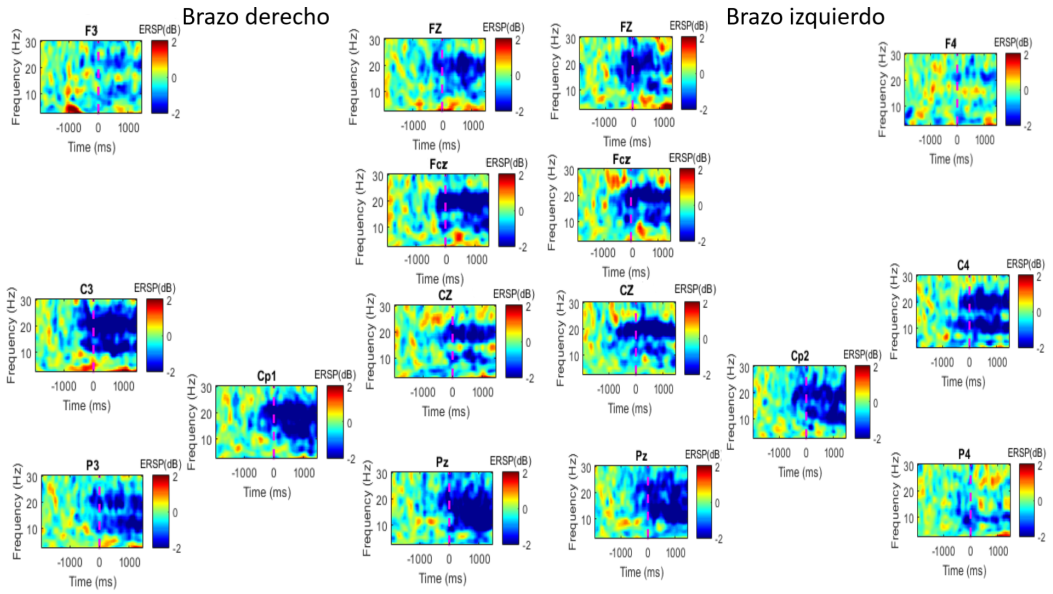



Figura 6.14: Grandes averages de los ERDs de todos los sujetos del protocolo de alcance de una posición, filtro de la media. Se observa desincronización (color azul) en los canales C3, Fcz, C4 y Cz, en los valores superiores de alfa y en los valores de beta. La desincronización empieza instantes anteriores al onset(t=0) y continúa a lo largo del movimiento.

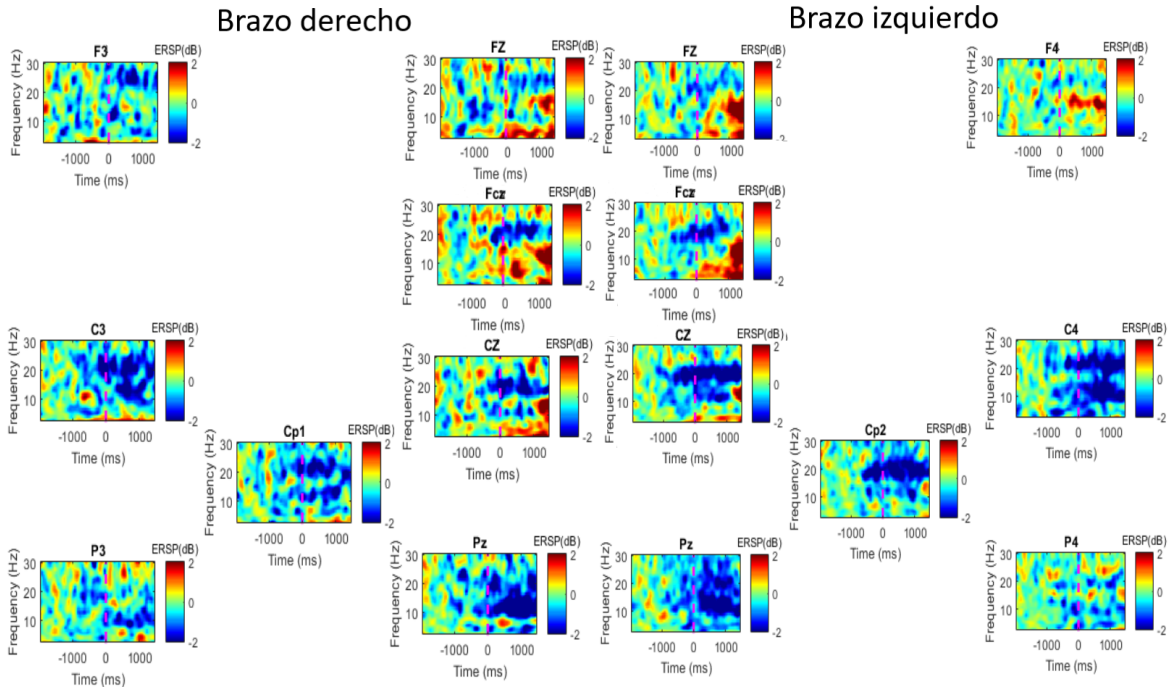



Figura 6.15: Grandes averages de los ERDs de todos los sujetos del protocolo de alcance de una posición, filtro CAR. Se observa desincronización (color azul) en los canales C3, C4, Cz en los valores superiores de alfa y valores de beta. Esta desincronización empieza antes de que suceda el onset(t=0) y continúa a lo largo del movimiento.

beta, en el caso del brazo izquierdo si se puede observar algo de desincronización en el canal C4,Cz, CP2 y Fcz, en las bandas superiores de alfa y en beta. En cambio al aplicar el filtro de la media, esta desincronización es visible en las bandas alpha y beta, en los canales encontrados en córtex motor, ya que empieza antes del onset y continúa a largo de la tarea. Por último con filtro CAR se obtiene mejor resultado que con el protocolo anterior, aunque la desincronización mas visible pertenecen al brazo izquierdo. En general usando el filtro de la media se observa mejor la desincronización.

Se toma el filtro de la media, como filtro espacial que mejor resultado ofrece a la señal, se calculan los ERPs para los canales Fz y Fcz, Figura 6.16, donde se observan de forma más clara en Fz, en cambio en Fcz esta más atenuado. En comparación con el protocolo anterior, el error es percibido con más retraso, lo cual puede deberse a la búsqueda de la esfera.

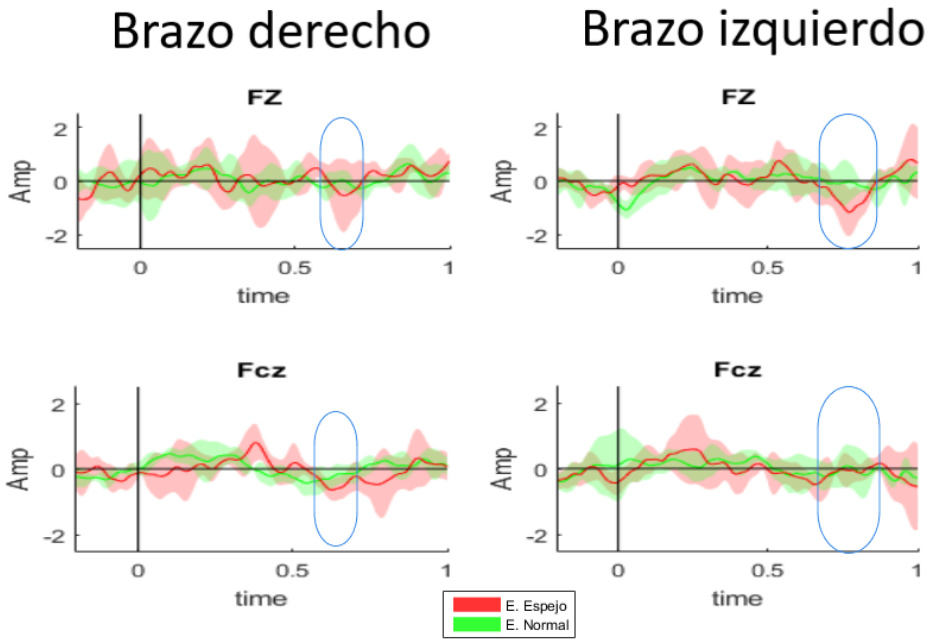



Figura 6.16: Grandes averages de los ERPs de todos los sujetos del protocolo de alcance de una posición aplicando filtro de la media. Se marca la desviación estándar de cada efecto(efecto normal, color verde, efecto espejo, color rojo) de forma sombreada sobre su correspondiente media. Se marca en azul los instantes donde se produce la diferencia en la amplitud al producirse el error y no producirse. Se observa como la amplitud del canal Fz la amplitud de la señal decrece en el momento en el que se detecta el error.

# Capítulo 7

# Conclusiones

Como conclusiones obtenidas sobre el proyecto para poder garantiza el cumplimiento de los subjetivos establecidos al principio del mismo.

Se han integrado las librerías de Bit&Brain para lectura de EEG en Unreal Engine, permitiendo la posibilidad de hacer uso de cualquier dispositivo que requiera dichas librerías en cualquier proyecto creado en Unreal Engine. Se ha verificado la correcta sincronización entre los datos de EEG y de la RV.

Se ha realizado el diseño e implementación de dos protocolos en RV, los cuales generan ficheros log para el posterior estudio de EEG. Fue necesario crear un segundo protocolo en RV ya que este sería una mejora del anterior. Durante la implementación de la reflexión no se esperaba que esta fuese tan tediosa debido a la cadena cinemática que ofrece Unreal Engine. Por otro lado al realizar la integración de Unreal Engine junto a las Oculus Rift también resulto una tarea algo tediosa, ya que los sensores de las Oculus Rift son configurados con unas determinadas posiciones al configurar el área de trabajo de las Oculus Rift, si los sensores son movidos provoca un cambio en el sistema de coordenadas de la RV haciendo que Unreal Engine obtenga unas coordenadas de los joystick erróneas, siendo necesario volver a configurar el área de trabajo de las Oculus Rift.

Se ha realizado la ejecución de los protocolos creados en RV en varios sujetos para poder realizar la adquisición de datos que posteriormente serían analizados. Durante la adquisición se comprobó que los sensores húmedos proporcionan mejor resultado que los sensores secos. Si los sensores secos son incluidos en las Oculus Rift, estas deberán de ajustarse bien a la cabeza del sujeto, ya que por el peso de las Oculus Rift pueden desplazarse, este desplazamiento provoca el desplazamiento de los sensores, lo cual hace que la señal registrada deba de ser desechada ya que se puede registrar zonas del córtex que no correspondan con las deseadas. En cambio al hacer uso de un gorro de agua,

puesto que este se ajusta a la cabeza del sujeto y encima de el se colocan las Oculus Rift el desplazamiento que experimenten no afectara a la señal recogida por los sensores.

Por último se ha realizado el análisis de los datos obtenidos de la adquisición, tras realizar el procesamiento de las señales EEG, se llegó a la conclusión de que se hubiesen obtenido mejores ERPs si se hubiese usado un mayor número de sujetos durante el experimento y un mayor número de trials. Ya que al incrementar estos dos factores se tendría una mayor variedad en las señales EEG, ya al realizar la eliminación de artefactos y ruido se pueden obtener menos de la mitad de trials para el procesamiento, obteniendo poca variedad de datos.

## 7.1. Trabajo futuro

Como trabajo futuro pensado en este proyecto, se prodría plantear la predicción en tiempo real de si el usuario percibe o no percibe error, para provocarle el efecto contrario al usuario y así obtener mejores ERPs. Esta predicción haría uso del estudio en tiempo real de la señal EEG filtrada en 4 y 10 Hz, observando el canal Fz. Y prediciendo dicha señal como el reconocimiento de un error o no. Para ello sería necesario tener un gran número de sujetos los cuales deberían de realizar un gran número de trials para poder obtener un gran número de datos para poder realizar mejor la predicción, ya que las señales EEG tienen una variabilidad muy alta.

Otro punto de futuro es la integración del visor de Bit&Brain en RV, ya que esto facilitaría a la hora de ajustarse los sensores, haciendo esto una tarea más autónoma ya que el propio usuario podría ajustarse los sensores y dar comienzo al experimento desde la RV.

# Capítulo 8

# Bibliografía


[1] Michael Fu, Janis Daly, and M.C. Cavusoglu. Assessment of EEG event-related desynchronization in stroke survivors performing shoulder-elbow movements. *IEEE International Conference on Robotics and Automation.*, 2006.

[2] Mario Tudor, Lorainne Tudor Car, and Katarina Ivana Tudor. Hans berger (1873-1941) - the history of electroencephalography. *Acta medica Croatica : casopis Hravatske akademije medicinskih znanosti*, 59:307–13, 02 2005. accesed Agosto 2019.

[3] Siegfried Othmer and David Kaiser. Implementation of virtual reality in EEG biofeedback. *CyberPsychology Behavior*, 3, 06 2000.

[4] Xinyu Tan, Yi Li, and Yuan Gao. Combining brain-computer interface with virtual reality: Review and prospect. *3rd IEEE International Conference on Computer and Communications*, 2017.

[5] Gert Pfurtscheller, Reinhold Scherer, Robert Leeb, Claudia Keinrath, Christa Neuper, Felix Lee, and Horst Bischof. Viewing moving objects in virtual reality can change the dynamics of sensorimotor EEG rhythms. *Presence*, 16:111–118, 02 2007.

[6] Dispositivo Hero de Bit&Brain Technologies . https://www.bitbrain.com/equipment/products/hero. accesed Agosto 2019.

[7] Dispositivo mobby de bit&brain technologies. https://www.bitbrain.com/equipment/products/versatile-eeg-16. accesed Agosto 2019.

[8] Tomasz Mazuryk and Michael Gervautz. Virtual reality history, applications, technology and future. Technical report, Institute of Computer Graphics Vienna University of Technology, Austria, Febrero 2019.

[9] William R. Sherman and Alan B. Craig. *Understanding Virtual Reality: Interface, Application, and Design, Edition 2*. 2018.

[10] Paul E. Dickson, Jeremy E. Block, Gina N. Echevarria, and Kristina C. Keenan. An experience-based comparison of unity and unreal for a stand-alone 3d game development course. pages 70–75, 06 2017.

[11] Unreal Engine 4 documentation. https://docs.unrealengine.com/en-US/index.html. accesed Mayo 2019.

[12] Unreal Engine API. https://api.unrealengine.com/INT/API/. accesed Mayo 2019.

[13] Mitch McCaffrey. *Unreal Engine VR Cookbook*. Addison-Wesley Professional, 2017.

[14] Unreal Engine 4 documentación UFUNCTION. https://docs.unrealengine.com/en-US/Programming/UnrealArchitecture/Reference/Functions/index.html. accesed Mayo 2019.

[15] Unreal Engine 4 documentación FRunnableThread Create. https://api.unrealengine.com/INT/API/Runtime/Core/HAL/FRunnableThread/Create/index.html. accesed Mayo 2019.

[16] Gert Pfurtscheller and Fernando Henrique Lopes da Silva. Event-related EEG/MEG synchronization and desynchronization: basic principles. *Clinical Neurophysiology*, 110:1842–1857, 1999.

[17] Arnaud Delorme, Jason Palmer, Julie Onton, Robert Oostenveld, and Scott Makeig. Independent EEG sources are dipolar. *PLOS ONE*, 7(2):1–14, 2012.

[18] Wiki Amica. https://sccn.ucsd.edu/wiki/AMICA.

[19] Silvia Erika Kober, Jürgen Kurzmann, and Christa Neuper. Cortical correlate of spatial presence in 2d and 3d interactive virtual reality: An EEG study. *International Journal of Psychophysiology*, Diciembre 2011.

[20] EEGLab. https://sccn.ucsd.edu/eeglab/index.php.

# Lista de Figuras

# Apéndices

# Apéndice A

# Pruebas realizadas en el efecto espejo

## A.1. Primera prueba

La primera idea se basa en las coordenadas que influyen en la reflexión, las coordenadas en el eje Y,Z no varían, ya que determinan la profundidad y la altura respectivamente, las coordenadas en el eje X si varían. Datos las transformaciones de $X_D$ y $X_I$ que determinan la localización y orientación de la nueva posición de la mano derecha e izquierda respectivamente, se realiza la reflexión en las localizaciones, mediante:

$$X'_I = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot X_D \tag{A.1}$$

$$X'_D = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot X_I \tag{A.2}$$

Siendo X'$_I$ la nueva posición en la mano izquierda y X'$_D$ en la mano derecha.

Tras obtener los resultados, se realiza la cinemática inversa. Finalmente, esta solución no proporciona el resultado esperado.

## A.2. Segunda prueba

Esta nueva idea tiene en cuenta la posición y la rotación dadas por los Joystick, ${}^{L}T_{manoD}$ (mano derecha) y ${}^{L}T_{manoI}$ (mano izquierda), que se convierten al espacio local del esqueleto, usando para ello la transformación del punto de origen del esqueleto (${}^{L}T_{root}$). Mediante lo siguiente:

$$\begin{aligned} T_D &= {}^{L}T_{root}^{-1} \cdot {}^{L}T_{manoD} \\ T_I &= {}^{L}T_{root}^{-1} \cdot {}^{L}T_{manoI} \end{aligned} \tag{A.3}$$

Una vez que se conocen los puntos en el espacio del esqueleto, se realiza la reflexión, de la misma manera en la que se propuso anteriormente:

$$\begin{aligned} H_I &= \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot T_I \\ H_D &= \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot T_D \end{aligned} \tag{A.4}$$

Tras realizar las correspondientes operaciones para cada mano, se obtiene la transformación que corresponde a la reflexión, esta transformación deberá ser convertida al espacio global, para ello se realiza la siguiente:

$$\begin{aligned} N_I &=^L T_{root} \cdot H_D \\ N_D &=^L T_{root} \cdot H_I \end{aligned} \tag{A.5}$$

Siendo $N_I$ la posición de la mano izquierda, tras realizar la reflexión, y $N_D$ las coordenadas de la mano derecha tras la reflexión.

# Apéndice B

# Resultados en Blueprints

## B.1. Resultado en AnimGraph

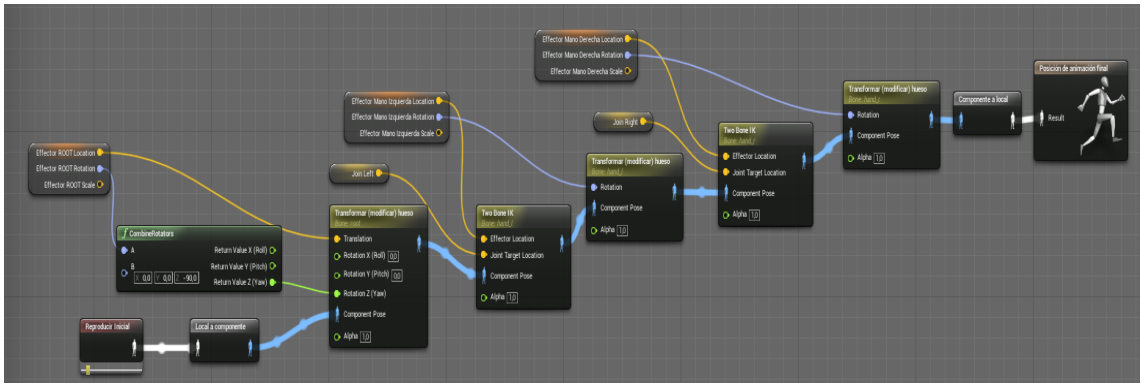



Figura B.1: Resultado de AnimGraph: Todas las funciones usadas en AnimGraph se realizan en el espacio global del esqueleto, siendo necesario que todos los parámetros recibidos por las funciones estén en dicho espacio. Para poder recrear el movimiento resultante de aplicar la cinemática inversa, primero se ajusta la posición y orientación del avatar, mediante la función de *Transformar(modificar) hueso*, realizando esta función sobre el hueso root del esqueleto. Tras esto se realiza la cinemática inversa en cada brazo, mediante las funciones de *Two bone IK* y *Transformar(modificar) hueso*. La función de *Two bone IK* calcula las posiciones en las que se debe de colocar cada junta de la cadena cinemática. La función de *Transformar(modificar) hueso* se encarga de ajustar la orientación de la cadena cinemática.

# Apéndice C

# Resultado de la realidad virtual

Figura C.1: Representación del brazo derecho en la realidad virtual y en el entorno real. El brazo derecho del mundo virtual tiene la misma posición y la misma orientación que en el mundo real usando cinemática inversa con la posición del mando.

Figura C.2: Efecto espejo realizado en la mano derecha. Donde la orientación de la palma derecha está hacia la pared, esta orientación se representa de la misma forma en el mundo virtual.


Figura C.3: Efecto espejo realizado en la mano derecha. Donde la orientación de la palma derecha está hacia el propio usuario, representando la misma orientación en el mundo virtual.