# Universidad de Mendoza


## Facultad de Ingeniería

### Tesis de Maestría en Teleinformática

# "Uso de VRPN para la implementación de una interfaz cerebro-computadora"

Javier José Rosenstein

Director de Tesis:
Mg. Ing. Osvaldo Marianetti

2020

# Agradecimientos

Quiero agradecer en igual medida a todos los que de una u otra manera me ayudaron a llevar adelante este proyecto.

Muy especialmente a mi director de tesis al Magíster Ingeniero Osvaldo Marianetti, por su guía y orientación, sus aportes y criterio preciso que me permitieron darle el mayor aprovechamiento al trabajo realizado.

Al Doctor Especialista en Neurología Infantil Raúl E. Otoya Bet, quien aportó sus invaluables conocimientos y experiencia en neurología y quien tuvo la enorme tarea de ayudarme con el aprendizaje de cada uno de los conceptos médicos, durante mi participación bajo su dirección en NNT (Neuromed Neuro Technology) y por corregir y ayudarme en todo lo relacionado a la neurología y medicina de este proyecto.

A los integrantes del equipo de NNT, al Ingeniero y futuro Bioingeniero José Raúl Morán, por sus prototipos en las placas electrónicas de adquisición EEG/EOG, a la Bioingeniera Lorena Nardi por sus estudios sobre ataques de epilepsia y adquisición de datos EEG, a la Licenciada Jimena Arroyo por sus aportes en todo lo relacionado con las técnicas de neurofeedback y epilepsia, la Psicóloga Cecilia Cáceres por su ayuda en el planteo de escenarios para tratar al paciente con problemas cognitivos y al Ingeniero Mecatrónico Andrés Manelli por su aporte en el trabajo con el prototipo de avatar humano y diseño de cabezal, todos en NNT fueron de gran ayuda para mi avance en el presente trabajo.

A todo el equipo de docentes y directivos que conforman la Facultad de Ingeniería de la Universidad de Mendoza, la Maestría en Teleinformática y la Dirección de Posgrado, quienes han sido de gran ayuda en esta instancia de mi formación académica.

Finalmente agradezco a mi familia que me apoyó durante todo este proceso y me dio las fuerzas necesarias para culminarlo.

# Resumen

En el desarrollo de sistemas de realidad virtual uno de los inconvenientes que se encuentran es la comunicación entre las aplicaciones y los dispositivos de adquisición, ya sea por no disponer de un método de acceso en forma directa de los dispositivos o por necesitar independencia entre ambos, es decir que las aplicaciones corran en una plataforma y los dispositivos en otras.

Para lograr esta independencia y a su vez permitir la integración de todo el sistema de realidad virtual, es necesario la implementación de algún protocolo de comunicaciones que permita esta vinculación heterogénea en tiempo real.

Los dispositivos generalmente están asociados a funciones o características de los individuos que los utilizan y se necesita integrar los movimientos que estos representan a la aplicación de realidad virtual correspondiente.

El presente trabajo trata del análisis e implementación del protocolo de comunicaciones VRPN (Virtual Reality Protocol Network) entre las partes de un entorno multimedia donde interactúan la adquisición de movimientos del usuario y la representación visual en un escenario virtual que permita la retroalimentación al usuario en tiempo real logrando una experiencia interactiva e inmersiva.

# Índice general

# Índice de figuras

# Índice de algoritmos

# Parte I

# Introducción

En el desarrollo de los sistemas de realidad virtual totalmente interactivos, es necesario contemplar: a) el desarrollo de la interfaz gráfica que se le presenta al usuario, y b) la adquisición de datos provenientes de los movimientos del mismo como entrada de datos del sistema de realidad virtual. El usuario es quien voluntariamente debe comandar esta interfaz digital, de modo que todo esto se convierta en una interacción ágil, bidireccional y produzca el efecto de virtualidad deseado, conformando un sistema usuario-máquina tan parecido a la realidad como sea posible. Para lograr mediante este sistema cumplir con los objetivos que se deseen obtener, ya sea el caso del sólo efecto lúdico de entretener al participante o con fines mucho más específicos como un complejo sistema de rehabilitación cognitiva en pacientes con capacidades físicas disminuidas conocidos como "Juegos Serios"[1] que se trata posteriormente en el capítulo 1.2.1.

En el área médica neurológica se habla mucho sobre el concepto de Neurofeedback [1], éste consiste en un tratamiento empleado o técnica neurológica, en la cual se le estimula al paciente, de manera visual no invasiva, para que el cerebro del mismo interprete estos estímulos indicados. Para que a modo de retroalimentación, esto permita que cierto grupo de neuronas, regulen la intensidad de las señales que generan, y de este modo, se estimule la aparición de patrones necesarios y así ayuden en la eficacia de los métodos de aprendizaje. Estimulando el interés en la tarea y logrando la concentración del paciente en el entorno virtual presentado, favoreciendo el avance dentro de la interfaz a travez de los objetivos que ésta plantea, logrando la superación de las etapas del juego y permitiendo de este modo, el avance esperado y el aprendizaje correspondiente.

Por experiencias de trabajos en conjunto, entre el autor de este trabajo y diversos especialistas del área médica, se plantea la necesidad de solucionar el recurrente requisito de contar con herramientas, que ayuden a los institutos de neurología en la tarea de trabajar con sus pacientes que poseen patologías relacionadas con lo conductual, en especial, cuando éstos además presentan limitaciones físicas, ya sea por miembros faltantes o reducción motora de algún tipo.

Dada entonces la necesidad planteada, el presente trabajo propone sentar las bases para el desarrollo y las implementaciones de los escenarios de realidad virtual, para ser usados como sistemas de rehabilitación cognitiva. Resolviendo los problemas que puedan ocurrir debido a la ausencia de prototipos, inexistencia de documentación clara y de metodologías concretas, relacionadas con las técnicas y protocolos de comunicación que se requieren para intercomunicar las interfaces y el hardware de adquisición.

Puntualmente se trata de resolver los inconvenientes que se encuentran en la comu-

[1] Juegos serios: juegos útiles o serios, orientados al aprendizaje, diseñados con un propósito diferente al solo efecto lúdico de entretener al participante.

nicación entre las aplicaciones y los dispositivos de adquisición cuando se necesita independencia entre ambos. Para lograr esta independencia y a su vez permitir la integración de todo el sistema de realidad virtual, es necesario la implementación de algún protocolo de comunicaciones que permita esta vinculación heterogénea en tiempo real. Por lo tanto se realiza el análisis e implementación del protocolo de comunicaciones VRPN entre las partes de un entorno multimedia donde interactúan la adquisición de movimientos del usuario y la representación visual en un escenario virtual. que permita la retroalimentación al usuario en tiempo real logrando una experiencia interactiva e inmersiva.

El trabajo comienza brindando los fundamentos a las tecnologías que se deben integrar, para lograr implementar lo que pasaremos a llamar una Interfaz Cerebro-Computadora o (BCI)[2]. Abordando los temas de cerebro, neuronas, señales que se generan por la activación de éstas, cómo se pueden adquirir eléctricamente mediante técnicas de Electroencefalografía (EEG)[3], y la adquisición de movimientos oculares mediante las técnicas de Electrooculografía (EOG)[4]. Luego se describe los pasos necesarios para el desarrollo de las interfaces virtuales como así también lo necesario para comprender el protocolo VRPN[5] y el desarrollo de su implementación específica en el prototipo planteado.

Es importante destacar que el presente trabajo surge como requerimiento de un proyecto de la clínica neurológica Neuromed Argentina S.A., y del cuerpo de neurólogos liderado por el Doctor Raúl Otoya Bet (codirector de la presente tesis), quienes desean obtener un sistema de rehabilitación mediante realidad virtual que pueda ser utilizado como: a) herramienta de capacitación en el caso de juegos serios, b) método de ayuda al tratamiento contra la enfermedad de epilepsia, ya que mediante la implementación de técnicas de neurofeedback permita disminuir la probabilidad de ocurrencia de una crisis. c) técnica que permita al paciente cognitivo reconstituir el programa cerebral faltante o deteriorado mediante la implementación de técnicas de neuronas en espejo (descripto en el apartado 1.2.3 y 3.1.2).

En base a lo anterior, la hipótesis que se plantea en el presente trabajo consiste en demostrar la posibilidad de establecer a VRPN como un nuevo estándard a emplear en las comunicaciones entre sensores e interfaces en los escenarios de rehabilitación

[2]Por sus siglas en inglés Brain Computer Interfaces (BCI), constituyen una tecnología que se basa en la adquisición de ondas cerebrales para luego ser procesadas e interpretadas por una máquina o computadora.

[3]La electroencefalografía es una exploración neurofisiológica que se basa en el registro de la actividad bioeléctrica cerebral mediante un equipo de electroencefalografía.

[4]Es una técnica de adquisición de datos en donde mediante pequeños electrodos colocados cerca de los músculos de los ojos permite medir el movimiento de estos. se utiliza para la generación de un electrooculograma y es utilizado en la polisomnografía, que consiste en un estudio del paciente y su actividad ocular durante el sueño.

[5]Por sus siglas en inglés: Virtual Reality Peripherical Network o Red de Periféricos de realidad virtual.

neurológica, esto permitirá lograr la independencia necesaria entre los equipos y procesos de adquisición de datos y la representación o activación en la interfaz del dispositivo final.

Por lo tanto se propone la realización de un prototipo de sistema de rehabilitación cognitiva, que permita interactuar entre las señales de EEG/EOG que puedan provenir de un paciente neurológico y luego de ser interpretadas, deban ser enviadas hacia una interfaz de realidad virtual mediante el protocolo VRPN, con el propósito de cumplir el principio de neurofeedback esperado por el neurólogo.

Los objetivos que se desean alcanzar en principio son los siguientes:

- Objetivos Generales:
  - Desarrollo de un prototipo de interfaz gráfica de aprendizaje implementando las comunicaciones mediante el protocolo VRPN.
- Objetivos específicos:
  - Desarrollar el hardware que permita generar las señales que simulen la adquisición de EEG y EOG necesarias como entrada de datos del sistema BCI para expresar la voluntad de movimiento del individuo.
  - Codificar las señales generadas a comandos en el protocolo VRPN que permita transmitir la información al componente software/hardware que la requiera.
  - Desarrollo de las librerías y drivers correspondientes para la implementación de VRPN en el prototipo que permita comandar la interfaz gráfica.
  - Plantear los procedimientos y herramientas necesarias que permitan a futuro la adquisición de señales analógicas reales mediante técnicas de EEG y la adquisición de movimientos oculares mediante las técnicas de EOG.

# Publicaciones realizadas

Este proyecto ha sido presentado en los siguientes Congresos, Workshop y Encuentros de divulgación de las ciencias informáticas:

- IX Encuentro de Investigadores y Docentes de Ingeniería - ENIDI 2017 - Sesión de Doctorandos y Tesistas, UNCuyo - UM - UTN, Mendoza - Argentina. [2]
- XX Workshop de investigadores en ciencias de la computación (WICC), 26 y 27 de Abril de 2018, UNNE, Corrientes - Argentina. [3]

- II Congreso Internacional de Ciencias de la Computación y Sistemas de Información - (CICCSI), 5 al 9 de Noviembre de 2018, UCH - UNSJ, Mendoza y San Juan - Argentina. [4]
- XXI Workshop de investigadores en ciencias de la computación (WICC), 25 y 26 de Abril de 2019, UNSJ, San Juan - Argentina. [5]
- XXII Workshop de investigadores en ciencias de la computación (WICC), 7 y 8 de Mayo de 2020, UNPA, El Calafate, Santa Cruz - Argentina. [6]

## Organización del trabajo

A continuación se presenta a modo de resumen, cada uno de los capítulos abordados en el trabajo.

- Capítulo 1, Interfaz Cerebro-Computadora: comenzamos comprendiendo en que consiste una BCI o Interfaz Cerebro-Computadora, cual es el principio de funcionamiento y sus aplicaciones en el campo de las neurociencias, para dar contexto luego a la introducción a los juegos serios (sección 1.2.1), los cuales nos van a permitir implementar y conseguir el aprendizaje cognitivo, que es uno de los fundamentos de este trabajo, se detallan algunas técnicas neurológicas como lo es la de "neuronas en espejo" (sección 1.2.3).
- Capítulo 2, Adquisición de datos del Cerebro: en este capítulo se profundiza en la adquisición de señales cerebrales, por lo tanto se debe conocer como es la constitución fisiológica del cerebro y fundamentalmente las neuronas, ya que son nuestro foco de estudio, comprender su funcionamiento y las bases de comportamiento de la sinapsis y las redes neuronales, para poder aplicar las técnicas de adquisición correspondientes. Se estudian las técnicas de EEG y EOG (sección 2.2), cómo se adquieren mediante electrodos y los sistemas de posicionamiento de los mismos sobre la cabeza, las diferentes señales a analizar y como tomar registro de estas para su evaluación. Se estudian los ritmos Mu (sección 2.3), que son los principales indicios de la voluntad de movimiento, al que tenemos que enfocarnos para determinar el deseo del individuo, analizando la actividad o intención motora, llamada en este caso "Imaginería Motora".
- Capítulo 3, Técnicas de aprendizaje mediante estimulación neuronal inducida: aquí se plantean las técnicas estudiadas del área de las neurociencias sobre el aprendizaje mediante la retroalimentación de estímulos cerebrales o Neurofeedback y las técnicas de aprendizaje de imitación del comportamiento humano llamadas "neuronas en espejo", ampliamente estudiado y demostrada su efectividad en primates y bebés humanos.

- Capítulo 4, VRPN - Interconexión de dispositivos para Realidad Virtual: en este punto se describe la necesidad de contar con este set de herramientas de comunicaciones para realidad virtual, estudio de sus funcionalidades y uso.

- Capítulo 5, Desarrollo e implementación de los prototipos de hardware y software: este apartado, el más importante del trabajo, consiste en la presentación de los prototipos desarrollados y la implementación completa del sistema de adquisición y representación visual, todo esto mediante el desarrollo de las funciones necesarias para comunicarnos según las especificaciones del protocolo VRPN, tanto a nivel de hardware desde los equipos de adquisición y análisis de datos de usuario, conformando el esquema VRPN Server, el cual brindará los puntos de conexión maestros, como a nivel de software mediante el desarrollo de las librerías drivers necesarias para que las interfaces virtuales puedan consumir los flujos o streaming de datos disponibles y finalmente sean representados en la interfaz prototipo correspondiente.

- Capítulo 6, Otras técnicas estudiadas y modelos de escenarios para rehabilitación cognitiva: aquí se detallan las prácticas realizadas sobre otros conjuntos de software o librerías que responden al estándar VRPN como lo es las utilidades FAAST que permiten la comunicación mediante el uso del sensor Kinect para la adquisición del cuerpo humano y la utilización del software de diseño de interfaces virtuales Vizard, así como también se presenta un prototipo desarrollado con el software de animación Unity3D, si bien Vizard posee un driver implementado en forma nativa de VRPN Cliente, no así Unity3D que se vale de las librerías de la implementación del paquete de software OSVR.

- Capítulo 7, Alternativa futura en la adquisición de datos de EEG/EOG y generación de VRPN: Como planteo de futuras lineas de investigación o como posible camino de continuidad de este trabajo, se presenta aquí la utilización del cabezal de adquisición de señales de origen comercial EPOC, para su uso mediante VRPN, ya que el mismo se está usando mucho en el ámbito de las neurociencias, y si bien el paquete de hardware y software ofrecido como producto comercial carece de implementación del protocolo VRPN, se ha formado recientemente un grupo de desarrollo dedicado a la realización de una implementación de su software de comunicación basados en lenguaje C y Python, de modo abierto, se pretende entonces a futuro aprovechar este interesante trabajo para poder implementar VRPN y de este modo poder utilizar sus señales en este proyecto.

- Capítulo 8, Conclusiones: finalmente se presentan las conclusiones del trabajo, con respecto a los problemas encontrados durante el desarrollo del mismo,

como los aciertos y resultados positivos en cuanto al estudio e implementación del protocolo VRPN.

# Parte II

# Marco Teórico

# Capítulo 1

# Interfaz Cerebro-Computadora

## Introducción

Se define la composición de una BCI o Interfaz Cerebro-Computadora, detalle de su principio de funcionamiento y sus aplicaciones en el campo de las neurociencias, introducción a los juegos serios, y su aplicación en la búsqueda por lograr el aprendizaje cognitivo, parte fundamental de este trabajo. Finalmente se detalla la técnica neurológica de "neuronas en espejo".

## 1.1. Principios de funcionamiento

Una BCI es un canal de comunicación que no depende directamente de las vías de salida normales del cerebro, de sus nervios periféricos y/o músculos, como podemos profundizar en el estudio de Wolpaw sobre interfaces cerebro-computadoras para comunicación y control [7]. Esta provee tanto a los pacientes paralizados de un nuevo modo de comunicarse con el medio ambiente que lo rodea, al paciente disminuido cognitivamente de una metodología de aprendizaje y crecimiento personal, como así también le brinda a cualquier tipo de persona sin patología o condición preexistente, de una manera de comandar controles que le permitan interactuar entre dispositivos o aplicaciones de capacitación o simulación en general.

Entre los diversos métodos de monitorización cerebral empleados en las investigaciones actuales basadas en BCI, se encuentra el estudio mediante electroencefalograma (EEG). Este es el principal enfoque que se toma y resulta de mayor interés, debido a sus ventajas de bajo costo, operación conveniente y fundamentalmente por ser mínimamente invasivo.

En BCI basados en los estudios mediante EEG de hoy en día, se le vienen prestando mucha atención a las siguientes señales:

- Potenciales Evocados a eventos Visuales (VEP),
- Sensoriomotoras mediante ritmos mu o beta,
- Potenciales evocados a eventos o P300,
- Potenciales cortical lento (SCP),
- Potenciales corticales relacionados con el movimiento (CPRM).

Estos sistemas ofrecen algunas soluciones prácticas, por ejemplo la del movimiento del cursor y procesamiento de textos, por citar algunas aplicaciones para los pacientes con discapacidad motora.

En el presente trabajo se pretende introducir en algunas implementaciones de aplicaciones de la vida real, utilizando sistemas BCI prácticos de entre estos tipos descriptos, y realizar una implementación a modo de prototipo.

Para el planteo de nuestra interfaz BCI, nos apoyaremos en las técnicas de sistemas basados en imaginería motora, que analizan los ritmos mu y/o beta, tal es el caso del estudio de Pfurtscheller sobre BCI [8] y como se detalla en el libro "Brain-Computer Interfaces con EEG", de Wang [9].

## 1.2. Aplicaciones

Son múltiples las aplicaciones en las que se puede aplicar una BCI y en especial la que se pretende implementar en este trabajo. Una de las aplicaciones mas importantes es la utilización de la BCI en software de Videojuegos, esto es por el alto grado de retroalimentación que permite lograr en esta área, ya que el usuario se beneficia notoriamente con estas interfaces a diferencia de utilizar joystick, teclados o mouse, que en ciertos casos son imposibles de utilizar dado el grado de la patología del paciente, ya que el uso de estos puede resultar incómodo o incluso imposible, llegado el caso común de la ausencia de las extremidades necesarias para controlar el dispositivo en cuestión.

Estamos hablando entonces no solo de videojuegos para la actividad lúdica en si misma, sino en este caso de utilizar las tecnologías empleadas en el desarrollo de videojuegos, en el aprendizaje y capacitación, llamados "Juegos Serios" o "Serious Games".

### 1.2.1. Serious Games

La utilización de BCI y la tecnología de videojuegos nos permite trabajar en otra aplicación mucho mas interesante, que solo jugar como un pasatiempo, es la de la im-

plementación de “Serious Games” o “Juegos Serios”, el término se refiere literalmente al planteo del Juego como elemento de capacitación y aprendizaje.

Los juegos serios se han convertido en un creciente mercado en la industria de los videojuegos y un campo de investigación académica, como podemos ver en el artículo: “Why so Serious?” de Breuer y Bente [10]. Si bien la gran mayoría de estos juegos están dirigidos al aprendizaje y a la educación, muchos estudios y textos sobre juegos serios no tienen una visión general de las posibilidades de utilizarlos para aprender y mucho menos todavía para la aplicación de técnicas de reconstitución de programas cognitivos en el campo de la rehabilitación neurológica. Para abordar este tema, se realizó una revisión de la literatura y se examinaron las definiciones existentes de los juegos en general y los juegos serios en particular.

Luego de discutirlo con quienes se interesan por el momento en estas tecnologías para llevar a cabo avances verdaderos en el campo de la neurología, los neurólogos, plantearon diversas practicas y/o entrenamientos básicos necesarios para ser utilizados por el paciente, quien posee la patología de disminución cognitiva, dada por la ausencia de un programa motor o “software” que le impide desarrollarse socialmente con libertad.

Como ejemplo de investigación, hemos tomado a modo de prueba de concepto las sugerencias de los neurólogos, tal es el caso de un paciente “post stroke” (posterior a un ataque epiléptico) o ACV, este paciente no puede desarrollar ciertas tareas cotidianas comunes, como las de vestirse sólo o prepararse una taza de café.

Uno de los planteos subyacentes de este trabajo, consiste en ampliar el alcance actual de que los juegos serios pueden ayudar en relación a la mejora del aprendizaje y la capacitación, para llevarlo a un nivel mayor en vista hacia los desarrollos futuros, aprovechando los avances alcanzados en la industria del desarrollo de videojuegos, principalmente se pretende demostrar que estas nuevas técnicas y herramientas, son válidas para llevarlas a cabo en forma profesional en el ámbito de la rehabilitación neurológica.

### 1.2.2. El Aprendizaje Cognitivo

En el aprendizaje tradicional, se producen fenómenos internos en la mente del individuo, generalmente relacionados con la estructura mental de las prácticas aplicadas en el diseño del proceso de aprendizaje, los objetos y sus relaciones. Se desarrollan técnicas para asimilar el conocimiento sobre estos aspectos y teorías, para luego medir el nivel de logro en general cuando se evalúan las conductas del individuo, por lo tanto, luego se evalúan el aspecto conductual, esto es lo medible y es lo tenido en cuenta por la corriente conductista. Medimos también resultados de aprendizaje, por ejemplo cuando aplicamos un test de evaluación.

El aprendizaje cognitivo pone por el contrario énfasis en lo que ocurre dentro de la mente, indagando cómo se acomoda el nuevo conocimiento con respecto a los ya adquiridos previamente. Para esta posición, el aprendizaje se construye conformando una estructura, en un proceso dinámico. Los estímulos no son determinantes directamente de la conducta, sino los procesos internos por los cuales el sujeto procesa esos estímulos, a través de la percepción, la memoria, el lenguaje y el razonamiento, que le permitirán resolver problemas que se planteen.

En el aprendizaje cognitivo, la clave consiste en tratar de que el individuo logre adquirir la información, en forma voluntaria o no, mediante el desarrollo de experiencias que permitan alcanzar el objetivo final planteado.

### 1.2.3. Neuronas en Espejo

En la ciencia neurológica y psicológica se están estudiando fuertemente las técnicas de neuronas en espejo, se estudió durante mucho tiempo con animales, especialmente con los monos.

De acuerdo a diferentes estudios sobre el tema, como el de Thomas acerca del efecto especial en las neuronas en espejo [11], una neurona en espejo es una neurona que se dispara tanto cuando un animal actúa como cuando el animal observa la misma acción realizada por otro. Así, la neurona "refleja" el comportamiento del otro, como si el observador estuviera actuando. Dichas activaciones de neuronas se han observado directamente en especies de primates en numerosos estudios.

En conclusión podemos considerar, que un gran conjunto de neuronas de la sección motora del cerebro, comienzan a "pre-actuar" cuando el individuo visualiza una acción llevada a cabo por otra persona de su misma especie, en la cual él se siente reflejado y por lo tanto su cerebro realiza la acción de intentar llevarla a cabo, por lo tanto podemos decir que se asemeja a cuando uno se ve reflejado a sí mismo en un espejo, nuestra propia imagen nos muestra el movimiento realizado, generando de este modo una especie de reacción en cadena dentro de nuestro cerebro, esta coacción que se produce es lo que se desea provocar en el paciente, y la mejor manera de implementarlo es mediante la representación gráfica de un escenario virtual personificado con un avatar en el cual el participante crea verse reflejado.

# Capítulo 2

# Adquisición de Datos del Cerebro

## Introducción

Debemos profundizar en la adquisición de señales cerebrales, comenzamos con la constitución fisiológica del cerebro y las neuronas, comprender su funcionamiento y comunicación mediante la sinapsis y las redes neuronales, conocer las técnicas de adquisición EEG y EOG mediante electrodos y los sistemas de posicionamiento sobre la cabeza, las diferentes señales a analizar y como realizar su evaluación. Se estudiarán los ritmos Mu mediante los cuales se previsualiza la voluntad de movimiento o lo que se considera como el deseo del individuo, finalmente se analiza la intención motora, llamada "Imaginería Motora".

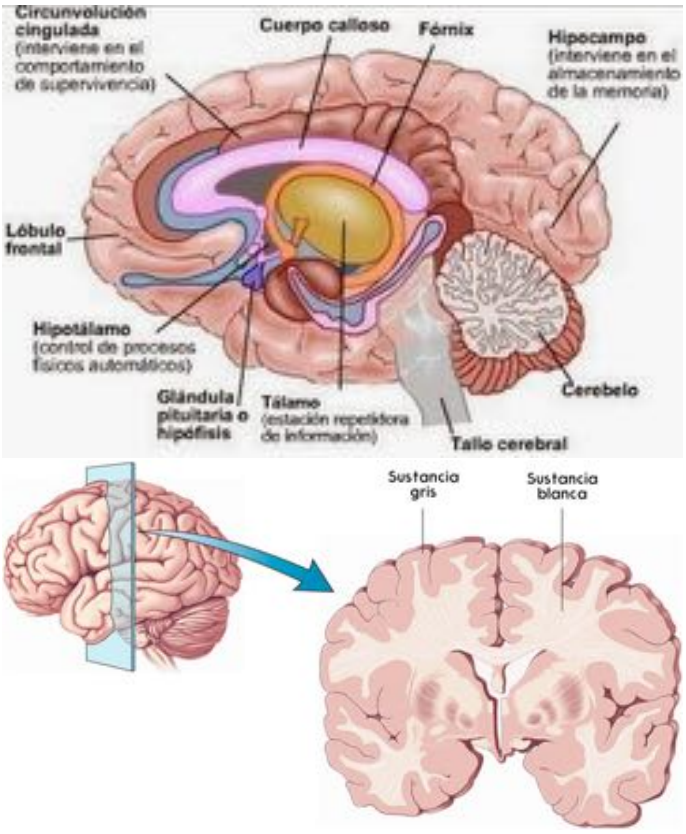
La tecnología de Interfaz cerebro-computadora es un sistema de interacción hombre-máquina capaz de traducir nuestras intenciones en la interacción real con un mundo físico o virtual. El funcionamiento básico de una BCI consiste en medir la actividad cerebral, procesarla para obtener las características de interés y una vez obtenidas, interaccionar con el entorno de la forma deseada por el usuario. Desde un punto de vista de interacción hombre-máquina, esta interfaz tiene dos características que la hacen única frente a todos los sistemas existentes: la primera de ellas es su potencial para construir un canal de comunicación natural con el hombre, la segunda es su posibilidad de acceder a la información cognitiva y emocional del usuario. La presente exposición aborda la tecnología de interfaces cerebro computador desde un punto de vista tecnológico, presentando su contexto actual y las problemáticas tecnológicas asociadas con las que hay que lidiar.

## 2.1. El Cerebro

El cerebro es parte del sistema nervioso central (SNC) y es el responsable de interpretar la información sensorial recibida del medio ambiente, para que los seres

humanos puedan comportarse como lo hacen. Cada región del cerebro cumple su función en este proceso. La corteza cerebral es una capa fina de 2 - 3mm de espesor, que cubre toda la superficie del cerebro. Las distintas regiones funcionales de la corteza (lóbulos temporal, parietal, occipital y frontal) son responsables del control motor, como así también de las funciones cognitivas y de la memoria. La información sensorial pasa a la corteza desde un sistema subcortical conocido como el tálamo, quien además juega un rol importante de regulación de la interacción entre distintas regiones. La corteza está formada por 10 a 100 billones de células nerviosas, densamente interconectadas llamadas neuronas. La materia o sustancia gris en la Figura 2.1 es la corteza, plegada para formar circunvoluciones o surcos y contiene todas las neuronas corticales. Entre la corteza y la subcorteza se encuentra la sustancia o materia blanca, una región compuesta principalmente de conexiones entre distintas áreas del cerebro. La mayoría de esas conexiones son entre las variadas regiones corticales, pero los sistemas subcorticales como el tálamo, también se comunican con la corteza a través de la materia blanca. Muy pocas neuronas pueden encontrarse en esta región, como podemos profundizar en el libro de Varsavsky sobre epilepsia y mediciones en EEG [12].

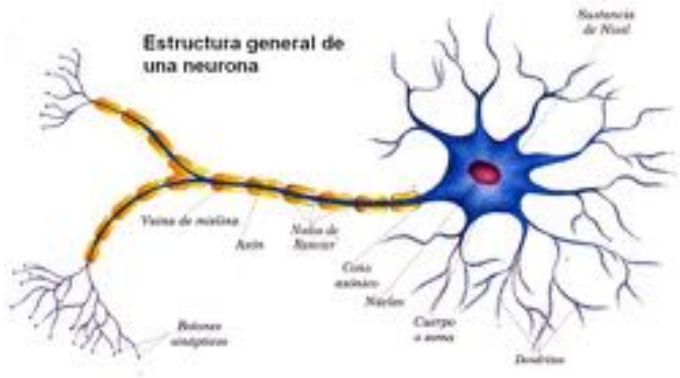
Figura 2.1: Cerebro, Sustancia gris y sustancia blanca



### 2.1.1. Las Neuronas

Las neuronas son las responsables del procesamiento y transmisión de la información en el SNC, pero no son el único tipo de células presentes. Las células gliales en la corteza, superan en número a las neuronas, aunque su función no es todavía bien entendida. Se cree que cumplen una función de soporte, como provisión de estructura, aislación y mantenimiento. Las neuronas piramidales son las células nerviosas más comúnmente halladas en la corteza. Las neuronas tienen distintas formas y tamaños, pero todas están compuestas de 4 estructuras básicas: las dendritas, el soma o cuerpo celular, el axón y los terminales sinápticos (Figura 2.2).

**Figura 2.2: Estructura general de una neurona**




#### Estado de reposo

En estado de reposo, las neuronas mantienen un potencial estable a ambos lados de su membrana celular. El potencial de reposo posee un rango entre 50 y 100 mV en el interior de la célula con respecto al exterior.

#### Estado activo

Las células excitables pueden conducir PA (potencial activo) cuando son estimuladas adecuadamente. Un estímulo adecuado es aquel que despolariza la membrana celular lo suficiente como para superar un umbral de disparo. El PA es de tipo todo o nada y viaja por la membrana a una velocidad constante. El valor pico alcanzado normalmente es 115 mV por encima del potencial en reposo.

Luego de que la membrana posee un PA, su sensibilidad a otro estímulo se ve alterada. Durante un intervalo de tiempo, ningún estímulo, sin importar la intensidad del mismo, produce un PA. Este intervalo es llamado periodo refractario absoluto. A continuación de este periodo, durante el periodo refractario relativo, la célula es

excitada solo por estímulos de mayor amplitud que los normales. Ese periodo limita la frecuencia máxima de descarga de una célula excitable. Para un PA que se propaga a lo largo de un axón, la zona que se encuentra en estado de activación se llama región activa.

#### 2.1.1.1. Transmisión sináptica

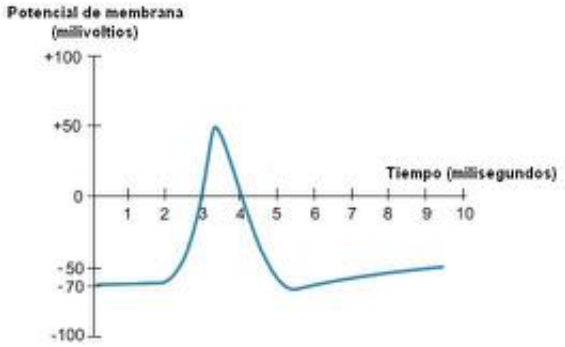
Las entradas que ingresan en una neurona son corrientes químicas (cargadas eléctricamente) que ocurren en los terminales (o pulsadores) sinápticos dispersados a lo largo del árbol dendrítico. Estas corrientes se transmiten hacia el cuerpo celular que va acumulando todas estas entradas. Si esta acumulación alcanza un umbral de voltaje, se dispara el potencial de acción, si no se alcanza este umbral, el mismo no se genera. Estos potenciales de acción son las salidas de la neurona, que hará sinapsis con las dendritas de otras neuronas, lo que se convierte en la entrada de la célula receptora. En la Figura 2.3 se muestra una representación esquemática de este sistema para una sola neurona. Se puede observar que la función del cuerpo celular va más allá de actuar como integrador de entradas dendríticas, ya que también restaura las concentraciones de carga en el fluido extracelular.

**Figura 2.3: Modelo de un sistema equivalente de una neurona real en el que se definen las entradas y salidas.**


Si se alcanza repetidamente el umbral de voltaje en el soma (también llamado cuerpo celular), pueden generarse múltiples potenciales de acción hasta una frecuencia máxima de 1000 Hz. Los potenciales de acción viajan a 5 – 10 m/s y no se deterioran en el camino debido a procesos regenerativos. La forma de las diferencias de potencial creadas por un potencial de acción típico entre el interior y el exterior del axón se muestra en la Ilustración 1.1-6.

La transmisión sináptica es el proceso por el cual los potenciales de acción que llegan al final del axón de una neurona transmisora (prehispánica) son interpretados por las dendritas de la célula receptora (neurona postsináptica). La neurona prehispánica

**Figura 2.4: Forma del potencial de acción de una neurona.**




libera químicos conocidos como neurotransmisores que controlan la respuesta de la célula postsináptica, ver Figura 2.5. El tipo de neurotransmisor liberado varía, pero estos químicos son los responsables de generar el potencial postsináptico (PPS) que puede caer en una de dos categorías:

**PPSE** Potencial postsináptico excitador: La respuesta a la sinapsis es como una versión miniatura del potencial de acción en la célula post-sináptica, sólo que más largo en duración y más pequeño en amplitud. La generación de un PPSE lleva al soma más cerca del umbral, por lo que aumenta la probabilidad de que se dispare un potencial de acción.

**PPSI** Potencial postsináptico inhibidor: El efecto es opuesto al de un PPSE. La probabilidad de que se dispare un potencial de acción disminuye.

Un PPSI típicamente tiene un efecto más largo que un PPSE porque las sinapsis inhibitorias tienden a formarse cerca del soma. Sin embargo, el número total de PPSEs es mayor que los PPSI y por ello su efecto es mayor. El balance de los PPSIs y los PPSEs entrantes en una misma neurona determinará si la célula postsináptica disparará un potencial de acción.

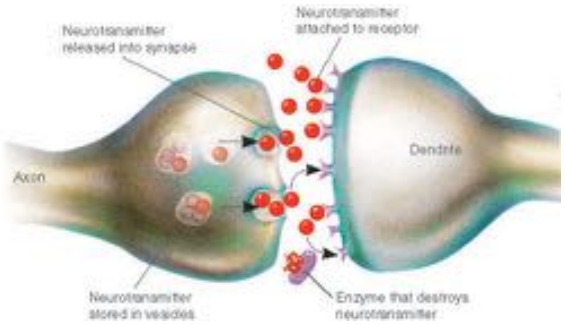
#### 2.1.1.2. Redes neuronales

En la corteza cerebral hay dos tipos de neuronas que representan el 90 % de la población total: las células piramidales y las interneuronas.

Los axones de las células piramidales forman sólo sinapsis excitatorias con otras neuronas y las interneuronas sólo forman sinapsis inhibitorias.

La interconectividad masiva entre éstas significa que una misma neurona puede estar transmitiendo y recibiendo información hacia y desde miles de otras neuronas. Por

**Figura 2.5: Sinapsis, neurotransmisor y neuroreceptor**




ello, no es un solo PPSE o PPSI el factor determinante de un potencial de acción postsináptico, sino el resultado de todas las señales entrantes integradas por el soma.

En la corteza, 1mm de tejido contiene alrededor de 50'000 neuronas y cerca de 3x108 sinapsis, 84 % excitatorias y el resto inhibitorias [12]. Este volumen es conocido como columna cortical – un compartimento dentro del cual todas las neuronas están conectadas entre sí, pero con relativamente muy pocas proyecciones hacia fuera de la columna cortical.

Se cree que la columna cortical es la unidad funcional básica del cerebro, aunque en realidad (Según los estudios al respecto) no se trata de un volumen discreto con límites bien definidos. Las conexiones entre columnas corticales se denominan proyecciones córtico-corticales. En menor proporción hay también conexiones con redes subcorticales, principalmente el tálamo, responsable de comunicar información sensorial con la corteza, entre otras cosas. Éstas son proyecciones tálamo-corticales. El primer tipo de proyecciones (córtico) supera en número a las segundas (tálamo) en una relación de 100 a 1.

Funcionalmente la existencia de columnas corticales que pueden actuar relativamente de forma autónoma permite procesamiento de información paralelo y más rápido.

Se cree que uno de los roles centrales del tálamo es sincronizar y desincronizar la actividad entre múltiples columnas corticales, de manera que puedan actuar juntas o independientemente en una tarea determinada.

Las conexiones córtico-corticales están formadas tanto por proyecciones excitatorias como inhibitorias, pero en general las proyecciones inhibitorias se mantienen relativamente locales. Las proyecciones córtico-corticales son más densas entre columnas corticales que están cercanas entre sí. La mayoría este tipo de conexiones son recíprocas, es decir, si una columna cortical tiene sinapsis que conecta con neuronas de otra columna, entonces es muy probable que esta segunda columna también proyecte axones a la primera.

## 2.2. Adquisición de Datos y Análisis de EEG/EOG

En Nuestro proyecto necesitamos la identificación de la voluntad del individuo de realizar un movimiento por lo tanto se necesita identificar mediante los electrodos c3 y c4, analizar los ritmos mu y beta, por lo tanto vamos a estudiarlos a continuación.

Con el análisis de las señales de electrooculografía se va a determinar la dirección de la voluntad del movimiento para poder entender que está queriendo realizar el individuo.

### 2.2.1. EEG

El electroencefalograma es la técnica clínica más ampliamente utilizada para explorar la actividad neuronal. Permite valorar la actividad eléctrica de la corteza en procesos agudos y/o crónicos, siendo una prueba repetible, accesible y económica.

La actividad eléctrica de fondo del cerebro en animales no anestesiados se describió cualitativamente en el siglo diecinueve, pero se analizó por primera vez de forma sistemática por el psiquiatra alemán Hans Berger, quien introdujo el término electroencefalograma (EEG) para referirse a las fluctuaciones de potencial obtenidas del cerebro. Convencionalmente, la actividad eléctrica del cerebro se registra con tres tipos de electrodos – electrodos de superficie, corticales y profundos. Cuando los electrodos se ubican en la superficie expuesta del cerebro (corteza), el registro se denomina electrocorticograma (ECoG). También pueden utilizarse finos electrodos de aguja de distintas formas para introducirlos en el cerebro, y en este caso se obtendrá un registro en profundidad (sorprendentemente, se produce muy poco daño al tejido cerebral cuando se emplean electrodos de tamaño correcto). Ya sea que se obtengan sobre cuero cabelludo (superficie), de la corteza o de la profundidad del cerebro, los potenciales fluctuantes eléctricos registrados representan una superposición de los potenciales de campo producidos por una variedad de generadores de corriente neuronales activos dentro del medio o volumen conductor. Las fuentes que generan estos potenciales son agregados de elementos neuronales con complejas interconexiones. Esos elementos neuronales son las dendritas, somas y axones de las células nerviosas. Es más, la arquitectura del tejido nervioso no es uniforme entre distintas regiones del cerebro.

Por lo tanto para estudiar el resultado de la adquisición de EEG, deben conocerse:

1. La anatomía y función del cerebro.
2. La estructura de la corteza cerebral.
3. Los potenciales generados en una sola neurona que llevan a la interpretación de los potenciales extracelulares que se obtienen en la corteza.

**Potenciales bioeléctricos del cerebro** El tejido nervioso presenta como una de sus funciones básicas la capacidad de generar potenciales eléctricos que son la base de la excitabilidad del organismo. Para comprender la forma en que se generan estos potenciales es preciso un conocimiento de la estructura y las conexiones de aquellas partes del cerebro que los originan. Todo el sistema nervioso posee capacidad electrogénica. Sin embargo, para los propósitos del EEG bastará con considerar la corteza cerebral y las regiones directamente relacionadas con ella.

**Electrogénesis cortical** Un fragmento de tejido cortical aislado es asiento de actividad eléctrica espontánea. Esta actividad se caracteriza por disparos de ondas lentas sobre las que se superponen ritmos rápidos. Entre un disparo y otro aparecen períodos de silencio eléctrico.

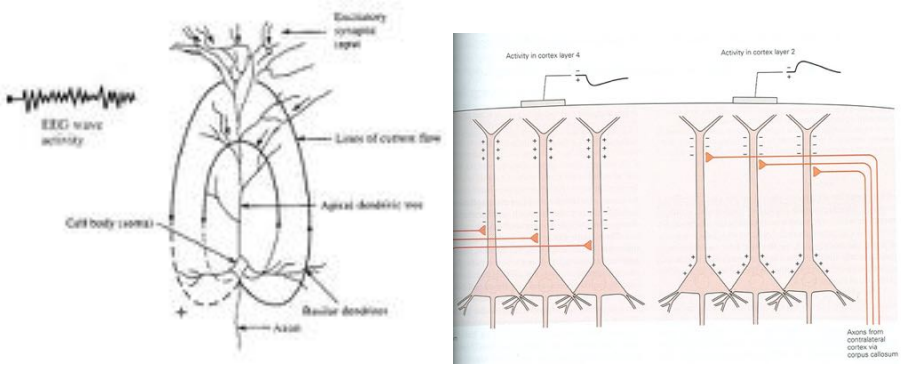
Estas señales son producidas como consecuencia de la actividad sináptica general de regiones discretas de tejido: los PPSE: potenciales postsinápticos excitadores, de menor incidencia frente a los inibidores pero numerosos. y los PPSI: Potenciales postsinápticos inhibidores, de mayor incidencia frente a los excitadores pero en menor número. se suman entre si y dan origen a potenciales lentos que son las ondas registradas. Una de estas porciones de tejido capaz de producir actividad eléctrica se llama un GENERADOR.

Los registros unipolares (explicados más adelante) de los potenciales de la superficie cortical respecto de un potencial de referencia remoto, pueden verse como la medición de un potencial integrado en las fronteras de un gran volumen conductor que contiene un arreglo de fuentes de corriente. Bajo condiciones normales, los potenciales de acción conducidos por los axones en la corteza contribuyen muy poco al potencial de superficie integrado, debido a que hay tantos axones en la corteza que se orientan en tantas direcciones respecto a la superficie y que se activan asincrónicamente. Consecuentemente, su influencia espacial y temporal neta sobre el potencial de la superficie es mínima. Constituye una excepción, por supuesto, el caso de las respuestas evocadas por la estimulación simultánea (sincrónica) de una entrada cortical (por ejemplo, la estimulación eléctrica directa de los núcleos talámicos o de sus caminos aferentes, que se proyectan directamente en la corteza vía axones tálamo-corticales-entrada cortical). Estas respuestas sincrónicas se denominan potenciales evocados, y son relativamente grandes en amplitud. El sincronismo de las fibras subyacentes y la actividad neuronal cortical es el principal factor que influye en la magnitud del potencial de superficie. Potenciales unipolares registrados dentro de las capas corticales han demostrado que el potencial de superficie es en gran parte debido a al efecto neto de potenciales postsinápticos locales de células corticales [13]. Éstos pueden ser de cualquier signo (excitatorios o inhibitorios) y pueden ocurrir directamente debajo del electrodo o a cierta distancia de él. Un cam-

bio en el potencial registrado en la superficie es una medida de la caída de potencial neta (corriente x resistencia iR) entre el punto superficial y el electrodo de referencia distante. Es obvio, sin embargo, que si todos los cuerpos celulares y dendritas de las células corticales se ubicaran aleatoriamente en el medio cortical, la influencia neta de las corrientes sinápticas sería cero. Esto resultaría en una situación de "campo cerrado" que produce potenciales relativamente pequeños. Por ello, cualquier cambio eléctrico registrado en la superficie es debido al arreglo ordenado y simétrico de algunos tipos de células dentro de la corteza.

Las células piramidales de la corteza cerebral (Figura 2.3) se orientan verticalmente, con sus dendritas apicales paralelas unas a otras. Los cambios de potencial en una parte de la célula respecto de otra parte crea campos de potenciales "abiertos" en los que puede fluir la corriente y pueden medirse diferencias de potencial sobre la superficie cortical. Las sinapsis de entrada al árbol dendrítico apical causan la despolarización de la membrana dendrítica. Como resultado, fluye una corriente subumbral en un camino cerrado por el citoplasma de las dendritas y del soma de las células piramidales, retornando finalmente a los puntos sinápticos superficiales por el medio extracelular (Figura 2.6).

**Figura 2.6: Líneas de flujo de corriente en neuronas piramidales de la corteza debido a estímulos excitatorios en las dendritas distales. Si el estímulo fuese inhibitorio, se invertirían las polaridades y la región apical se convertiría en la fuente (+).**




De la dirección indicada de las líneas de flujo de corriente, el medio extracelular alrededor del soma se comporta como un emisor (+), mientras que la parte superior del árbol dendrítico apical se comporta como un colector (-).

La influencia de un potencial postsináptico (PPS) dendrítico particular en el registro de superficie cortical depende de su signo [excitatorio (+) o inhibitorio (-)] y de la ubicación relativa al sitio de medición. El efecto de cada PPS puede considerarse como si se creara un dipolo de corriente radialmente orientado. Por ello, entradas

sinápticas continuas crean una serie de dipolos de potencial y los resultantes flujos de corriente que son fluctuantes pero se superponen temporal y espacialmente. Una población de fibras presinápticas y las células en las que éstas terminan pueden generar potenciales de superficie de cualquier forma, dependiendo de la proporción de inhibiciones o excitaciones, el nivel de las células postsinápticas en la corteza, etc.

Las dendritas apicales de las células piramidales constituyen una malla de unidades similarmente orientadas en las capas más externas de la corteza. Al activarse múltiples terminaciones sinápticas de cada célula, puede fluir una corriente en cualquier dirección dependiendo de si las sinapsis son excitatorias o inhibitorias. La relación emisor-colector entre la dendrita y la célula es la de un dipolo de corriente oscilante, donde las variaciones en la orientación y magnitud del dipolo producen fluctuaciones con forma de ondas en el potencial de superficie (Figura 2.6). Cuando la suma de la actividad dendrítica es negativa respecto de la célula, la célula está despolarizada y es excitable. Cuando esta suma es positiva, la célula está hiperpolarizada y es menos excitable [13].

#### 2.2.1.1. Adquisición del EEG

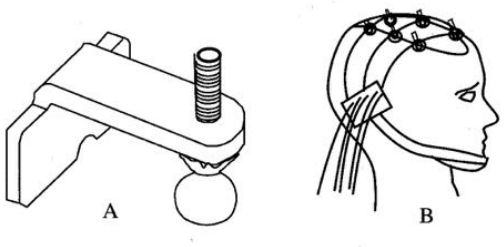
Técnicamente la actividad bioeléctrica cerebral es sensada a nivel del cuero cabelludo por los electrodos. Luego, esta señal es amplificada en las distintas etapas del electroencefalógrafo para finalmente ser registrada ya sea en papel o en un visualizador digital. Los electrodos deben ser diseñados y construidos de manera que permitan registrar la actividad eléctrica de forma eficiente y con el mínimo de distorsión. Según con qué procedimiento se obtenga la actividad bioeléctrica cerebral (sobre el cuero cabelludo, en la base del cráneo, en cerebro expuesto o en localizaciones cerebrales profundas) y con qué fin se está obteniendo el registro (fines diagnósticos, prequirúrgicos, etc), pueden emplearse distintos tipos de electrodos. 1. Electrodos de superficie: Se colocan sobre el cuero cabelludo. 2. Electrodos basales: Se aplican en la base del cráneo sin necesidad de procedimiento quirúrgico. 3. Electrodos quirúrgicos: Para su aplicación se realiza una cirugía. Pueden ser corticales o intracerebrales Existen varios tipos de electrodos superficiales, y debido a que son los que más comúnmente se usan en electroencefalografía, son los únicos que se describirán: a. Electrodos adheridos: Son pequeños discos metálicos de 5mm de diámetro. Se adhieren con pasta conductora y se fijan con colodión, que es una sustancia aislante. Aplicados correctamente, presentan resistencias de contacto muy bajas (1-2KΩ), figura 2.7.

b. Electrodos de contacto: Consisten en pequeños tubos de plata clorurada roscados o soportes de plástico. En su extremo de contacto se coloca una almohadilla que se humedece con solución conductora. Se sujetan al cráneo con bandas elásticas y se

**Figura 2.7: Electrodos superficiales adhesivos tipo copa.**

conectan con pinzas de "cocodrilo". Son de muy fácil colocación, pero incómodos para el paciente. Por este último motivo, no se emplean para registros de larga duración (Figura 2.8).

**Figura 2.8: (A) Esquema de un electrodo de contacto. (B) Colocación de los electrodos de contacto.**




c. Electrodos en casco de malla: De introducción reciente. Como se ve en la figura 2.9, los electrodos están incluidos en una especie de casco elástico. Existen cascos de diferentes tamaños, dependiendo de la talla del paciente. Se sujetan con cintas a una banda torácica. Como características más importantes, pueden enumerarse la comodidad de colocación, comodidad para el paciente en registros de larga duración, inmunidad frente a artefactos y precisión en la colocación, lo que los hace muy útiles en estudios comparativos [14].

#### 2.2.1.2. Sistemas de posicionamiento de los electrodos superficiales

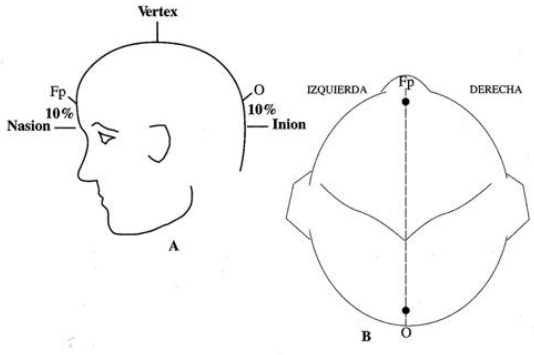
Le llamamos sistema de posicionamiento a la disposición de los electrodos sobre el craneo del individuo.

**Figura 2.9: Electrodos de casco, de malla o gorro.**

Aunque hay varios sistemas diferentes (Illinois, Montreal, Aird, Cohn, Lennox, Merlis, Oastaut, Schwab, Marshall, etc), el sistema internacional "Diez-Veinte" (10-20) es el más usado. Para colocar los electrodos según esta distribución, se procede de la siguiente forma [15]:

- Se mide la distancia entre el nasion y el inion pasando por el vértex. El 10 % de esta distancia sobre el nasion señala el punto donde se ubicará el electrodo Fp (Frontal Polar). El 10 % de esta distancia sobre el inion señala el punto donde se ubicará el electrodo O (Occipital) (Figura 2.10).

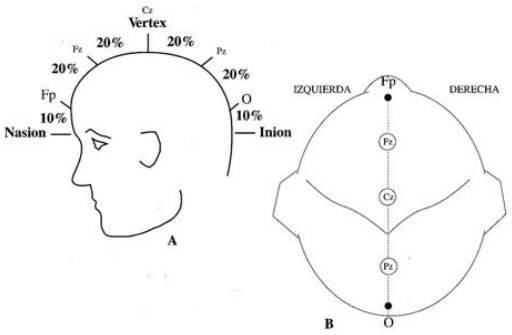
Figura 2.10: (A) Vista de perfil. (B) Vista superior. Fp, punto frontal polar; O, punto occipital.




- Entre los puntos Fp y O se sitúan otros tres puntos espaciados a intervalos iguales (entre cada dos el 20 % de la distancia nasion-inion). Estos tres puntos

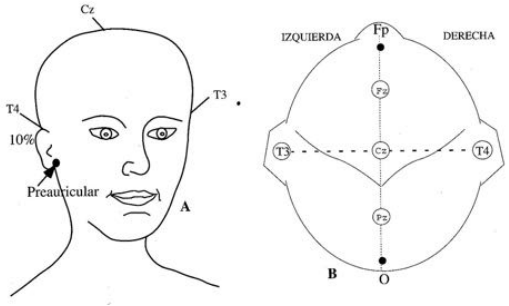
son, de adelante hacia atrás, el Fz (Frontal), el Cz (Central o Vértex) y el Pz (Parietal) (Figura 2.11).

Figura 2.11: (A) Vista de perfil. (B) Vista superior. FZ, punto frontal; Cz, punto central; Pz, punto parietal.




- Se mide la distancia entre los puntos preauriculares (situados por delante del pabellón auditivo) pasando por el vértex (Cz). El 10 % de esta distancia marca la posición de los puntos temporales mediales, T3 (izquierdo) y T4 (derecho) (Figura 2.12).

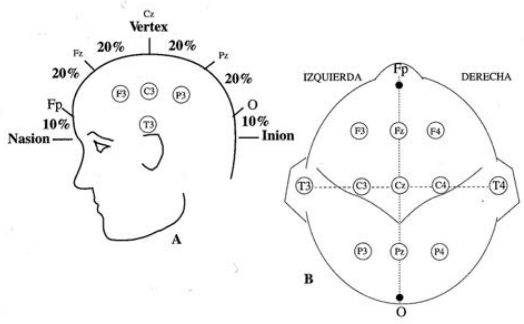
Figura 2.12: Medición coronal lateral. (A) Vista frontal. (B) Vista superior. Situación de los electrodos T3 y T4.




- Un 20 % de la medida por encima de los puntos temporales medios se colocan los electrodos C3 (izquierda) y C4 (derecha). El vértex es ahora el punto de intersección entre la línea anteroposterior y la línea coronal lateral (Figura 2.13).
- Los electrodos F3 y F4 (izquierda y derecha, respectivamente) están situados de forma equidistante entre el punto frontal medio (Fz) y la línea de electrodos temporales (Figura 2.13).

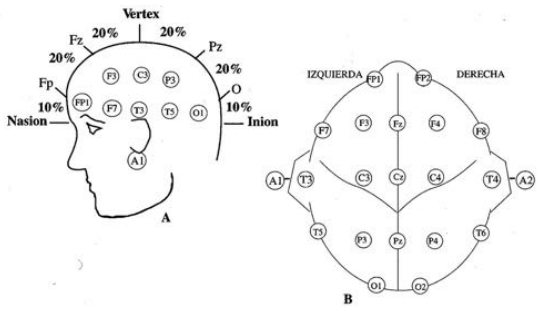
- Los electrodos P3 y P4 (izquierda y derecha, respectivamente) equidistan entre el punto P medio y la línea de los electrodos temporales (Figura 2.13)

Figura 2.13: (A) Vista frontal. (B) Vista superior. Situación de los electrodos F3 y F4.




- Se mide la distancia entre el punto medio Fp y el punto medio O a través de T3. El 10 % de esta distancia a través de Fp corresponde a los electrodos FP1 y FP2. El 10 % de esta distancia a través de O corresponde a los electrodos O1 y O2.
- Los electrodos F7-F8 se sitúan equidistantes entre los puntos FP1-FP2 y T3-T4, respectivamente.
- Los electrodos T5-T6 se sitúan equidistantes entre los puntos T3-T4 y O1-O2, respectivamente (Figura 2.14).
- A un 10 % de los temporales T3 y T4 se sitúan los electrodos auriculares Al y A2 respectivamente (Figura 2.14).

Figura 2.14: (A) Vista de perfil. (B) Vista superior. Situación de los electrodos A1 y A2.

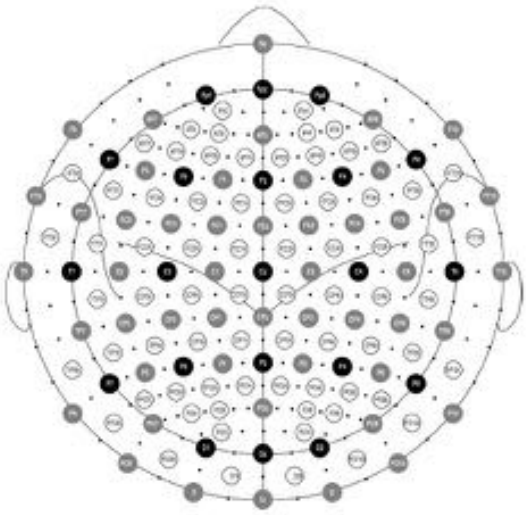


Como regla general, los electrodos del lado izquierdo llevan numeración impar, mientras que los del lado derecho la llevan par. Además, como ya se dijo, los electrodos de la línea media reciben subíndice "z". Hasta aquí se ha considerado sólo la disposición de 20 electrodos, pero siguiendo la misma metodología de:

1. medir la distancia entre electrodos en una misma línea,
2. obtener el 10 % de esa distancia,
3. separar los nuevos electrodos esa distancia de los ya existente ubicándolos sobre la misma línea, se pueden ubicar tantos electrodos como el tamaño del paciente lo permitan.

Por ejemplo, en la Figura 2.15 se observa una disposición con 128 electrodos.

Figura 2.15: Disposición de 128 electrodos ubicados siguiendo los principios de posicionamiento del sistema internacional 10-20.



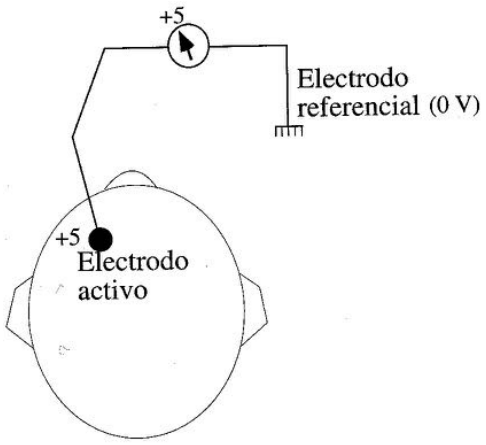
#### 2.2.1.3. Registro de Señales de EEG

**Montajes de un EEG** Para empezar a registrar señales de EEG se parte de una serie de electrodos situados sobre la superficie del cuero cabelludo en ubicaciones precisas, determinadas según el sistema internacional diez-veinte. Cada electrodo es un punto de registro. Sin embargo, para poder realizar este registro es preciso disponer de dos terminales.

Por este motivo, habrá que determinar cuáles son las posiciones de los electrodos donde se obtendrá la señal EEG, lo cual depende del número de canales disponibles y del propósito específico del registro que se obtendrá. En este sentido, se puede optar por trabajar con Registros Monopolares o Registros Bipolares [13].

**Registros monopolares** En los Registros Monopolares o Referenciales (figura 2.16) se toma la señal de cada uno de los electrodos independientemente de la de los demás. En esta situación, el electrodo de registro se llama electrodo activo y el segundo cable de entrada al equipo se toma de un electrodo llamado electrodo de referencia.

Figura 2.16: Esquema del montaje para un registro monopolar




Teóricamente, este electrodo debe estar situado en un punto de potencial cero, pero en la práctica esta condición no se puede conseguir. Por este motivo, se emplean referencias aproximadas como son el uso de electrodos en el lóbulo de la oreja, en el mentón o en el mastoides.

Otra forma de obtener un electrodo referencial es unir todos los electrodos activos en un punto (disposición promediada). El potencial de este punto será la suma de los potenciales de todos los electrodos. Presumiblemente, esta suma será cero, pero sólo permitirá registrar la actividad de un electrodo por vez, ya que todos los demás estarán cortocircuitados entre sí. Para solucionar este inconveniente, la interconexión se realiza por medio de resistencias de bajo valor (1 a 1.5MΩ). Con esta conexión se pueden tomar tantos pares activo-referencia como se desee, con la única limitación dada por el número de canales disponibles en el equipo (figura 2.17).

**Registros bipolares** En las mediciones con este tipo de montaje se toman parejas de electrodos secuencialmente para registrar las diferencias de potencial entre cada par de puntos (figura 2.18). Los dos electrodos de cada pareja se denominan electrodos activos.

Con esta forma de medición, es posible realizar un gran número de registros diferentes dado por la enorme cantidad de combinaciones posibles de pares de electrodos. Sin embargo, no todas las combinaciones brindarán información útil sobre la actividad cerebral. Por ello, es preciso seleccionar, de entre todas las posibilidades, sólo

Figura 2.17: Esquema de la disposición promediada. E1-E8, electrodos; A, amplificador

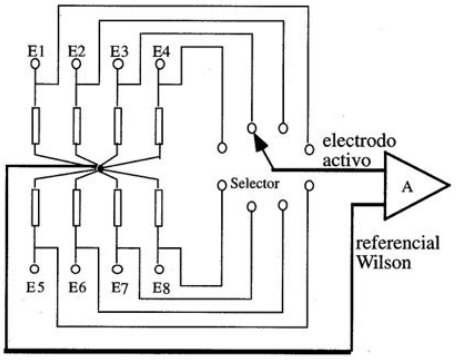
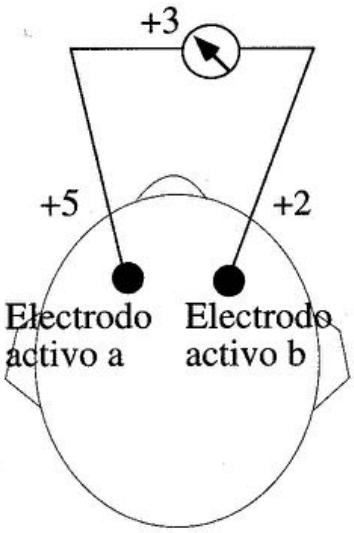



Figura 2.18: Montaje de medición bipolar entre dos electrodos activos a y b

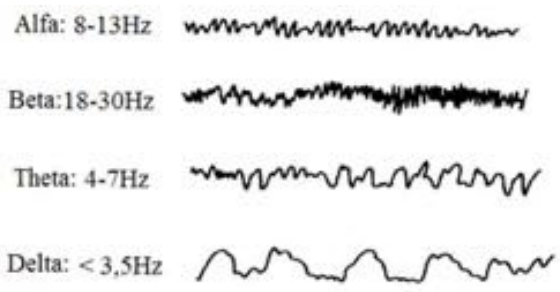


las más representativas. Cada grupo de combinaciones seleccionadas constituye un montaje distinto [13].

#### 2.2.1.4. Análisis de las Señales de EEG

En los registros capturados de EEG se reconocen ciertos patrones de ondas o ritmos cerebrales, que se observan en la figura 2.19

Figura 2.19: Patrones de señales eléctricas de EEG.




Las ondas α (alfa) poseen frecuencias entre 8 y 13Hz. Se registran en sujetos normales despiertos sin ninguna actividad y con los ojos cerrados, localizándose sobre todo en la región occipital del cerebro, pero también pueden registrarse en las regiones parietal y frontal. Su amplitud está comprendida entre 20 y 200μV. Cuando el sujeto está dormido, las ondas α desaparecen completamente. Cuando el sujeto despierto dirige su atención a algún tipo de actividad mental específica, este ritmo es reemplazado por ondas asincrónicas de mayor frecuencia pero menor amplitud. La figura 2.19(b) demuestra el efecto sobre el ritmo α del hecho de abrir los ojos cuando hay una luz brillante y luego cerrarlos nuevamente.

Las ondas β (beta) normalmente presentan un rango de frecuencias entre 14 y 30Hz, y a veces – particularmente durante actividad mental intensa – hasta 50Hz. Se registran con más frecuencia de las regiones parietal y frontal del cuero cabelludo.

Las ondas ϑ (theta) poseen frecuencias entre 4 y 7Hz. Se observan principalmente en las regiones parietal y temporal en niños, pero también durante situaciones de stress emocional en algunos adultos.

Las ondas δ (delta) incluyen todas las ondas del EEG por debajo de 3.5Hz. A veces estos ritmos se observan sólo una vez cada dos o tres segundos. Ocurren durante el sueño profundo, la infancia y en algunas enfermedades del SNC [13].

### 2.2.2. EOG

#### La electrooculografía

El Electrooculograma (EOG) es un método de registro de los movimientos oculares basado en la detección de la diferencia de potencial existente entre la córnea y la retina, el método detecta el movimiento angular del ojo. El origen de esta diferencia se encuentra atrás de la retina y permite considerar la presencia de un dipolo, donde la córnea corresponde al extremo positivo y la retina al extremo negativo, ver figura 2.20. Si el ojo se mueve desde la posición central directo a la periferia la retina se aproxima a uno de los electrodos, mientras que la córnea al electrodo del lado opuesto. Este cambio en la orientación del dipolo se refleja en un cambio en la amplitud y la polaridad de la señal EOG. Por lo tanto el análisis de estos cambios permite determinar el movimiento de los ojos [16].

#### Pruebas realizadas en laboratorio

La señal EOG puede obtenerse fácilmente mediante un par de electrodos bipolares conectados a los lados de los ojos para las mediciones horizontales y de dos pares de electrodos bipolares sobre y debajo de los ojos para las mediciones verticales como se muestra en la figura 2.21, como se necesitan solo un par de electrodos bipolares verticales en un solo ojo, el hecho de tener ubicados en ambos ojos permite utilizar el par de electrodos que tenga mejor señal o dado el caso de un ojo que tuviera movilidad reducida entonces se utiliza el par de electrodos del otro ojo. De esta manera se tienen entonces dos canales correspondiendo al movimiento horizontal y al movimiento vertical, respectivamente. Las señales EOG muestran amplitudes de señales que van de 5 uV a 20 uV por grado de movimiento, con un ancho de banda de 0 Hz a 30 Hz De acuerdo según vemos en los siguientes estudios: [17] y [18]. Una forma de onda ideal EOG puede observarse en la figura 2.22.

Figura 2.20: El ojo y las señales eléctricas

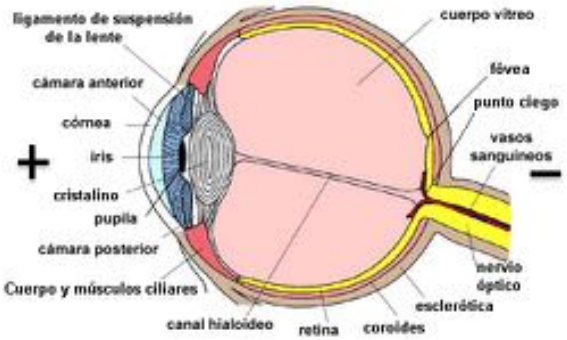


Figura 2.21: Ubicación de los electrodos bipolares horizontales y verticales.

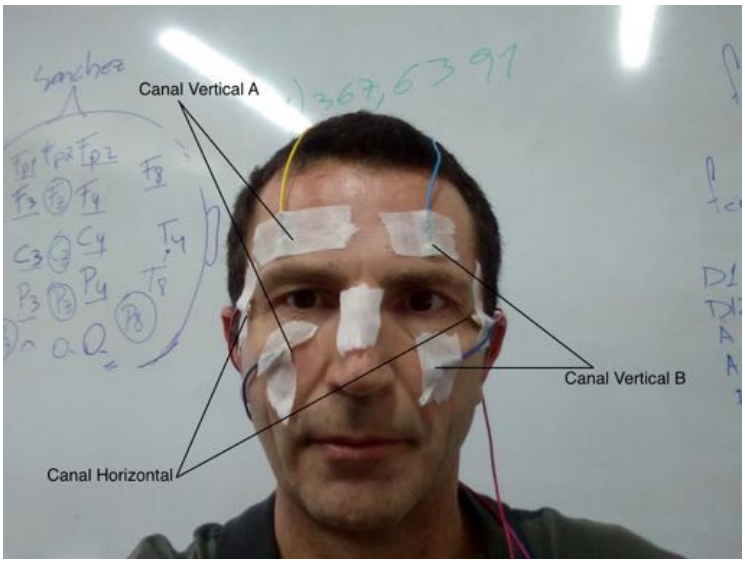
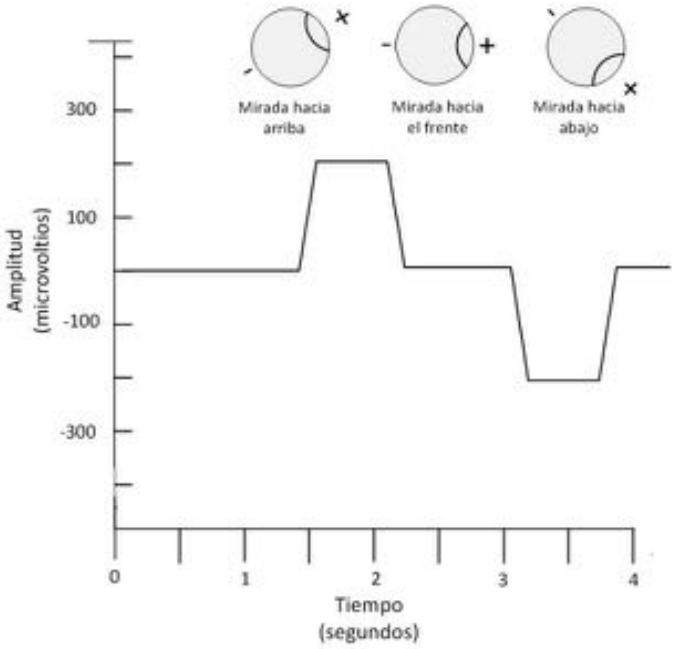



Figura 2.22: Formas de onda típicas, según la posición vertical del ojo.



Tal como vemos entonces la técnica de adquisición mediante ElectroOculoGrama o EOG, nos permite mediante electrodos ubicados en los músculos oculares realizar el análisis correspondiente que nos permitan determinar la posición de los ojos para determinar el movimiento a representar, esto lo logramos analizando diversos tipos de movimientos como los sacádicos [1], fijaciones [2] y parpadeos [3], que en el presente trabajo vamos a enfocarnos en los sacádicos, para evitar confusiones en los movimientos y las fijaciones para identificar los objetivos de movimiento así como las detenciones que indican las confirmaciones correspondientes.

Por lo tanto en el presente caso se utilizaron 6 electrodos del tipo copa, 2 horizontales a los lados de los ojos y 4 verticales de los cuales son 2 superiores y 2 inferiores, según se visualiza en la figura 2.21

Estos electrodos son del tipo bipolares ya que los electrodos no se toman como par de señal y referencia, sino como 2 niveles de señal que se suman, ya que son tensiones opuestas nos determinan un valor final en concreto mas fácil de analizar para luego determinar la dirección del movimiento ya sea este par de electrodos del canal horizontal o vertical correspondiente.

Si se quisieran usar los electrodos en modo unipolares, sería necesario ubicar otro electrodo usado como referencia de cada uno de los otros. por lo tanto las señales tanto verticales como horizontales serían capturadas de a pares y se dificultaría su análisis, por lo tanto se determinó su disposición en modo bipolar para tener una sola señal a analizar por par de electrodos ya sea horizontales como verticales.

Las señales de EOG presentan amplitudes muy pequeñas del orden de 10 uV a 300 uV con un ancho de banda de 0 a 30 Hz [17], [18], por consiguiente un amplificador de EOG además de presentar alta ganancia debe tener una excelente respuesta en bajas frecuencias.

Los electrodos se conectan al circuito amplificador, el cual estabiliza estas señales y nos devuelve en valores digitales los niveles de señales analógicas recibidas que luego de la etapa de amplificación son del orden de los -1,7 a +1,7 volts correspondientes a las amplitudes positivas y negativas de las señales, estas tensiones se trasladan a valores positivos en el orden de tensión de 0 a 5v, para pasar a la etapa de digitalización para que finalmente logremos obtener en valores de 8 bits, 0 a 1024 en decimal, y poder procesarlos digitalmente y obtener el vector dirección correspondiente que nos determine el cuadrante y el valor dentro del mismo.

En las pruebas realizadas se detectaron los siguientes patrones:

[1] Los sacádicos corresponden a los movimientos del ojo al cambiar rápidamente la mirada de un punto a otro (generalmente mayor a los 30 grados).

[2] Las fijaciones resultan de mantener la mirada en un mismo punto durante un tiempo establecido.

[3] Los parpadeos son los movimientos involuntarios y periódicos del ojo para humedecer la córnea y la conjuntiva.

En los electrodos bipolares verticales se observan picos positivos en la vista hacia arriba del ojo y picos negativos en la vista hacia abajo del ojo tal como los demostrados idealmente en la figura 2.22, estos picos no son permanentes, es decir que se producen en instantes con formas de picos superiores o inferiores mientras el ojo permanece hacia arriba o abajo, por lo tanto es necesario estar capturando permanentemente estas señales para detectarlos y poder determinar la posición del mismo, se determina que es necesaria una frecuencia de captura del orden de las 128 muestras por segundo para que no se pierdan los picos correspondientes.

En los electrodos bipolares horizontales se observan patrones con formas de mesetas superiores de 0 en el caso de la vista hacia la izquierda del individuo y una meseta negativa o inferior para el caso de la vista a su derecha.

cuando el ojo apunta hacia arriba y se activan los electrodos verticales, se observa escasa y nula actividad en los horizontales, esta se visualiza en forma plana como desconectada, y viceversa, cuando el ojo se inclina a uno de los lados, se activan fuertemente las señales características del movimiento horizontal pero en los electrodos verticales el movimiento es casi nulo.

La actividad resultante de los electrodos debe ajustarse o calibrarse entre pacientes diferentes, debido a que existe una diferencia de amplitud de las señales debido a las diferencias en la fisionomía de los ojos entre un paciente y otro diferente, por lo tanto es necesario realizar un proceso de ajuste de máximos y mínimos en la respuesta de amplificadores y digitalizadores utilizados.

Para determinar los valores máximos posibles de señales capturadas de los electrodos se evalúan las máximas tensiones posibles en un proceso de calibración que consiste en rotar los ojos alrededor de una imagen representada al individuo que debe seguirla con sus ojos.

El proceso diseñado para determinar estas extensiones consta de 8 puntos:

4 puntos centrales: superior, inferior, derecha e izquierda y 4 puntos extremos oblicuos.

Se planifica que este proceso de ajuste debe ser llevado a cabo por el software de realidad virtual quien le indique al paciente que debe mirar a los extremos propuestos para permitir que durante este proceso el hardware evalúe los patrones recibidos y pueda calcular las extensiones en los valores a entregar posteriormente.

## 2.3. Análisis de los Ritmos Mu para la detección de movimiento real o imaginario.

### 2.3.1. Ritmos Mu

Análisis de los Ritmos Mu en la adquisición de señales

El ritmo mu es una oscilación electroencefalográfica (EEG) con frecuencias dominantes en las bandas alfa, de 8 a 13 Hz. Estas oscilaciones están limitadas a cortos períodos de 0,5 a 2 s de duración y pueden también ser registradas en la corteza sensoriomotora humana en ausencia de movimiento real.

Debido a que parece ocurrir en ausencia de procesamiento de información sensorial o respuesta motora, el ritmo mu es concebido como el reflejo de un estado de reposo.

En general, los dos factores principales que afectan la ocurrencia o mantenimiento de los ritmos mu son:

- (a) la inactividad motora
- (b) el nivel y calidad de la atención.

Los ritmos mu son típicamente identificados como una oscilación alfa, en su medida máxima sobre la corteza sensoriomotora cuando el individuo está en reposo y es atenuada [conocido como "supresión de mu"] con el movimiento voluntario o la estimulación sensoriomotora, pero es mínimamente afectada por la estimulación visual. Ya que las frecuencias del ritmo mu se solapan con aquellas del ritmo alfa occipital o clásico (aprox. 10 Hz), a veces es difícil separarlos y con mucha frecuencia los dos han sido confundidos e identificados inapropiadamente, en el artículo de Pineda: [19] se evalúa el significado funcional de los ritmos "mu rhythms", desde ver y escuchar hacia hacer.

A partir de los hallazgos de algunos estudios, el autor del citado artículo, Pineda, sugiere que la corteza sensoriomotora despliega una variedad de ritmos mu con propiedades funcionales y topográficas específicas, en lugar de un solo ritmo uniforme. Incluso, las áreas cerebrales implicadas en los movimientos de la mano, del pie o del rostro, parecen producir distintos ritmos mu.

Por lo tanto se podría llegar a identificar cada uno y llegar a realizar representaciones específicas por ejemplo entre pies y manos.

**En resumen:** El ritmo mu es una oscilación que se observa en el electroencefalograma humano, incluso desde bebés muy pequeños, en la banda de frecuencia de

8-13 y de 15-25 Hz en ausencia de movimiento del sujeto. este ritmo es de la "familia" de los ritmos alfa (pues se da en una frecuencia cercana al alfa clásico que es de 8 a 12 Hz), sólo que se da específicamente en la corteza sensoriomotora contralateral a la del lado en que se realizará el movimiento, en la preparación del movimiento, o bilateral, durante la ejecución del movimiento.

La desincronización (desorden en el disparo de las señales de las neuronas) del ritmo refleja la activación de redes neurales (en el caso de mu, se refiere a redes neurales en la corteza sensoriomotora), mientras que la sincronización, la "desactivación" de las redes neurales o no activación de las mismas.

Por lo tanto, el ritmo mu realiza un efecto similar a que se "enloquece" (desincroniza o suprime o se disminuye) cuando el sujeto realiza un movimiento. Pero de manera más interesante aún, mu se "enloquece" también cuando el sujeto observa o imagina un movimiento (Imaginería Motora).

Los ritmos mu, que pueden ser distintos para mano, pie o rostro, reflejan la traducción de lo que se oye o se ve a lo que se hace en definitiva, o que se pretende hacer.

### 2.3.2. Imaginería Motora

**¿Qué es la imaginería motora?** El uso de la imaginación para pensar un movimiento en ausencia de movimiento, tal como lo describe en su estudio Richardson sobre imaginería motora [20] entre otros temas relacionados con la física del cuerpo y su reflejo en la mente. Este concepto se lo utiliza mucho en el estudio y la práctica de la rehabilitación psicofísica en general.

Es el estado dinámico durante el cual un individuo simula mentalmente una acción determinada. Este tipo de experiencia implica que el sujeto se siente a sí mismo que realiza la acción.

Es la representación mental de movimiento aunque no exista ningún movimiento del cuerpo.

En un proceso de rehabilitación se utiliza esta técnica para tratar problemas de dolor y movimiento relacionados con el sistema nervioso alterado mediante el ejercicio del cerebro en pasos medidos y controlados que aumentan en dificultad a medida que se avanza.

Estudios de imagen de mapeo del cerebro, como la resonancia magnética funcional o RMf[4] han aportado fundamentos sólidos de la reorganización cortical, observando

[4]La Resonancia Magnética funcional mide los pequeños cambios en el flujo sanguíneo que ocurren con la actividad del cerebro. Puede utilizarse para examinar la anatomía funcional del cerebro (determinando las partes del cerebro que están manejando funciones críticas), evaluar los efec-

la activación de neuronas en espejo del cerebro que se activan cuando miramos un movimiento o nos lo imaginamos.

Cuando hablamos de neuronas en espejo nos referimos a las técnicas de copia y simulación de movimientos mediante los cuales un individuo aprende o mejora nuevos movimientos. tal como se describe posteriormente en el capítulo 3.1.2.

Los primeros tratamientos utilizando esta técnica se hicieron en pacientes con dolor del miembro fantasma [5].

En entrenamiento de alto rendimiento se utiliza la imaginería motora para activar las áreas corticales motoras similares a las que se activan en movimientos reales, por lo tanto permite practicar el movimiento sin realizarlo y de este modo mejorar el movimiento y perfeccionarlo sin cansar el músculo correspondiente.

**¿Cómo podemos hacer que funcione?**

- Observando movimientos
  - Observando imágenes de movimientos.
  - Observando vídeos de movimiento.
  - Imaginando movimientos sectoriales, sólo una mano.
- Imaginando a otro haciendo el movimiento.
- Imaginándose a uno mismo haciendo el movimiento.
- Imaginándose a uno mismo en un movimiento implicado en una actividad.
- Imaginándose la actividad.

**¿En qué puede ayudar al individuo aplicar esta técnica en un serious game?**

- Aumentar el rendimiento (estimulación mental del movimiento especifico) velocidad, precisión, mejora la fuerza y la funcionalidad.
- Mejorar el equilibrio en personas con esclerosis múltiple, y en las mujeres de edad avanzada, rehabilitar los déficits motores en pacientes con accidente cerebrovascular y trata el dolor crónico.

tos del derrame u otras enfermedades, o guiar el tratamiento cerebral. La fRMN puede detectar anormalidades dentro de cerebro que no se pueden encontrar con otras técnicas por imágenes.

[5] Patología psicológica en la cual el paciente amputado siente el dolor característico como si todavía existiera el miembro extraído.

- Activar vías motoras (de manera que se asemeja a la actividad durante una acción normal, pero no causa ningún movimiento explícito).
- Activar áreas cerebrales inicialmente lesionadas.

# Capítulo 3

# Técnicas de aprendizaje mediante estimulación neuronal inducida.

## Introducción

Se plantean las técnicas estudiadas del área de las neurociencias sobre el aprendizaje mediante la retroalimentación de estímulos cerebrales o Neurofeedback y las técnicas de aprendizaje de imitación del comportamiento humano llamadas "neuronas en espejo", ampliamente estudiado y demostrada su efectividad en primates y bebés humanos.

## 3.1. Aprendizaje mediante Neurofeedback y Neuronas en Espejo.

En neurología se utilizan mucho las técnicas de aprendizaje mediante la retroalimentación de estímulos cerebrales que ayudan a lograr los avances requeridos tanto en materia de comportamiento como de activación neuronal, luego su avance es detectado y determinado mediante las técnicas de EEG, EEG Cuantificado, Potenciales relacionados a eventos y neuroterapias [21].

### 3.1.1. EEG-Neurofeedback

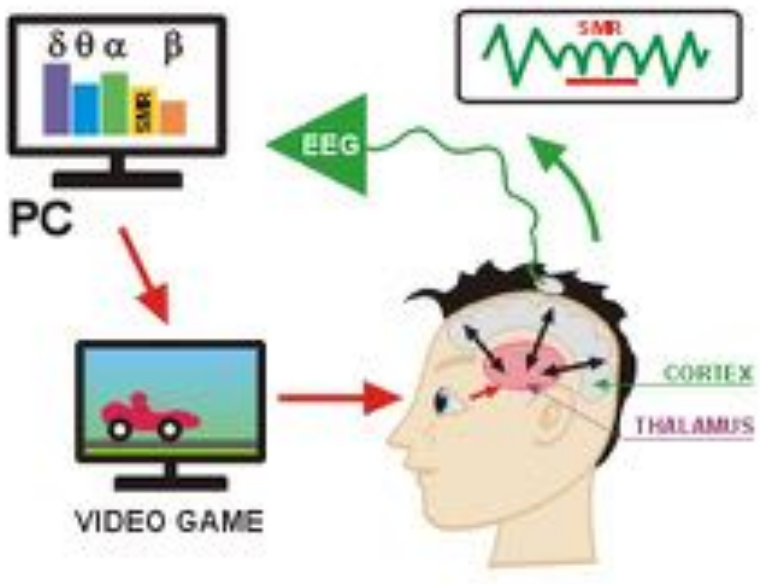
Se trata de una estrategia terapéutica de neuromodulación endógena no invasiva por lo tanto no es necesario ningún tipo de intervención quirúrgica, es muy fácil de implementar y de forma totalmente inocua para el paciente, pero de amplios resultados en el tratamiento.

Posibilita la autorregulación voluntaria de la actividad neuronal a partir de un proceso de aprendizaje cognitivo que emplea técnicas de condicionamiento operante con el fin de normalizar un patrón electroencefalográfico (EEG) considerado patológico o anormal, estudiado del artículo de Rief [22]principios y startup con Neurofeedback.

En el caso de Neurofeedback aplicado a los video games o serious games, la intervención consiste en recompensar al individuo si alcanzó un nivel de activación cerebral esperable en función de un objetivo terapéutico determinado, de este modo se logra la activación esperada, tal como se puede observa en la figura: 3.1.

Otros estudios explican detalladamente como se pueden aplicar las técnicas de neurofeedback utilizando técnicas musicales y logrando buena respuesta en los casos de pacientes propensos al sucicidio por patologías depresivas y baja estima, con muy buenos resultados, el artículo de Ramirez y sus colaboradores [23] nos explica la aplicación de neurofeedback basado en tratamientos aplicando música en casos de depresión en adultos mayores.

Figura 3.1: Neurofeedback mediante el empleo de videojuegos




### Principios Básicos del neurofeedback

1. Implica la posibilidad de detectar y evaluar la disfunción cerebral de un determinado cuadro clínico a través de diversos instrumentos neurofisológicos como la electroencefalografía cuantificada (QEEG) y los potenciales relacionados a eventos (ERPs),[21]estudio sobre electroencefalografía cuantificada (QEEG) y los potenciales relacionados a eventos (ERPs).

2. Requiere de la capacidad que presentan los individuos para producir cambios voluntarios a nivel cerebral que se asocian con el incremento o disminución de determinados parámetros cerebrales.

3. Se necesita de la posibilidad de memorizar y aprender este nuevo estado de activación neuronal y mantenerlo en el tiempo no únicamente en contextos de experimentación, sino que pueda generalizarse a otros entornos de la vida cotidiana del paciente.

De acuerdo a los estudios de Huster[1] donde se trabaja sobre las mejoras logradas en las funciones ejecutivas del cerebro mediante técnicas de entrenamiento aplicado al comportamiento y el uso de la neuroestimulación y el neurofeedback.

El neurofeedback es una interfaz cerebro computadora y su técnica se compone de cinco pasos :

1. Adquisición de la señal EEG,
2. Procesamiento de la señal,
3. Extracción de las características y análisis de ritmos sensoromotores. SMR/RSM,
4. Generación de una señal feedback,
5. Aprendizaje por condicionamiento operante.

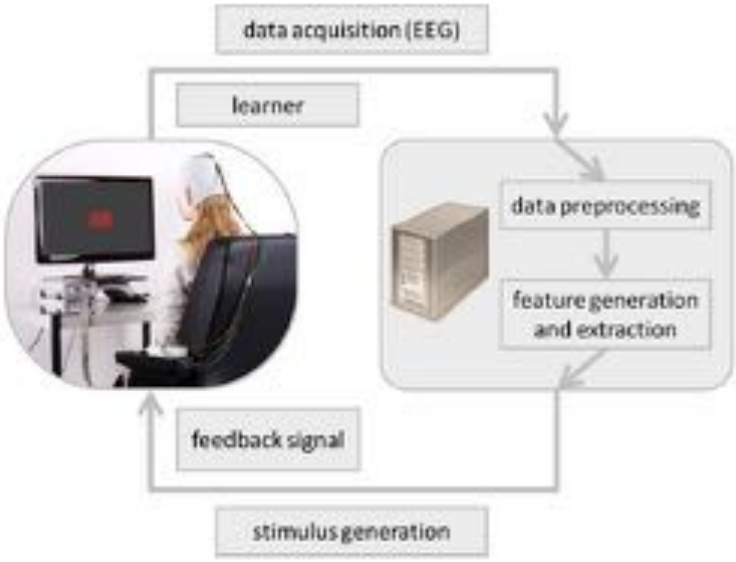
Como se observa en la figura 3.2.

Tal como se ha descripto en el punto 2.2.1.4 donde se detalla el análisis EED, mediante la Electroencefalografía (EEG) se analizan los estímulos correspondientes a las señales correspondientes a 2 grandes grupos:

- Rápidos
  - Alfa, de 8 a 13Hz.
  - Beta, de 13 a 30 Hz.
  - Gamma, de 30 a 100 Hz.
- Lentos
  - Delta, de 0,5 a 4 Hz.
  - Theta, de 3 a 8 Hz.

El Neurofeedback en definitiva es una técnica en la cual entrenamos al cerebro para ayudarlo a mejorar su propio funcionamiento y el del resto de organismo. El incorrecto funcionamiento del cerebro puede observarse a través de un electroencefalograma computado o mapeo cerebral computado (C-EEG).

Figura 3.2: Pasos del neurofeedback.




### 3.1.2. Neuronas en Espejo

La técnica de aprendizaje llamada Neuronas en espejo, se refiere a la didáctica aplicada en la cual se desarrolla mediante el empleo de una BCI un escenario en donde la imagen ilustra al individuo realizando la tarea que se desea que este aprenda y repita, si el reflejo del individuo es lo suficientemente real y logra que el participante se imagine a sí mismo realizando el movimiento, entonces se logra el resultado esperado, que consiste en que el cerebro active las neuronas correspondientes para "espejar" o imitar el movimiento correspondiente.

Por numerosos estudios sabemos que una porción de nuestro cerebro encargada del movimiento se activa cuando observamos el movimiento de otros seres humanos, a este efecto es el que se denomina Neuronas en espejo o Mirror Neurons, tal como se puede apreciar en los trabajos de Rizzolatti et al. [24] y [25], en los cuales queda claro que se reporta una mayor activación de la corteza motora durante la observación de las acciones realizadas por otros.

Los estudios sugieren un fuerte vínculo entre la experiencia de la acción y la resonancia motora [neuronal] durante la observación de la acción, siendo el mecanismo neural que subyace este fuerte apareamiento entre la acción y la percepción, el proporcionado por el sistema de neuronas en espejo, el cual se activa tanto durante la observación como durante la ejecución de la misma acción.

- Cuando uno observa a alguien ejecutando una acción, se activan más o menos las mismas áreas cerebrales que están implicadas en la ejecución de la acción. Este proceso de imitación es altamente automático y se cree que soporta el

reconocimiento y la interpretación de la acción, ya que permite una representación motora interna de la acción observada, esto se evidencia también cuando se estudian los casos de síndrome de miembro fantasma[1], en pacientes amputados.

**Comportamiento estudiado en Bebés:** Es sabido que un bebé aprende considerablemente más rápido cuando se encuentra en un grupo de pares que en la soledad, en el caso del aprendizaje en el gateo, hay estudios como el de van Elk [26]:

Van Elk et al. se propusieron investigar si la propia experiencia motora de los bebés (gatear y caminar) estaba relacionada con la activación de su sistema motor durante la percepción de las mismas acciones, llevadas a cabo por otros.

Doce bebés participaron en el estudio. Los estímulos fueron vídeos de otros bebés caminando y gateando, presentados en bloques de aprox. 10 segundos. Adicionalmente, los papás proporcionaron información acerca de las habilidades motoras de su bebé e indicaron la edad a la cual el bebé había empezado a gatear o a caminar. Para la medición de EEG, los investigadores usaron un gorro de EEG de 29 electrodos. El EEG, los movimientos de los ojos y los movimientos de las piernas y los brazos fueron medidos continuamente durante el experimento para obtener ensayos en los que el bebé mirara continuamente a la pantalla y no moviera sus propias extremidades.

El análisis de EEG del promedio del grupo reveló una fuerte desincronización en la banda de frecuencia mu (7-9 Hz) para la observación de gatear, en comparación con la observación de caminar, máxima para los electrodos centrales [C3, Cz, C4]. Además, también encontraron una desincronización bilateral más fuerte en la banda beta (17-19 Hz) de frecuencia en la observación de gatear, comparado con la observación de caminar.

Finalmente, la diferencia en la potencia o amplitud mu/beta entre los vídeos de caminar y gatear estuvo correlacionada con la propia experiencia de los bebés con caminar y gatear. Específicamente, el tamaño de los efectos tanto de mu como de beta (diferencia en el poder entre la observación del gateo y el caminar) estuvo altamente correlacionado con la cantidad de experiencia en el gateo de los infantes.

Es decir, a menor experiencia gateando, menor desincronización. En cambio, a mayor experiencia gateando, la supresión o desincronización en mu/beta dada fue mayor, por lo tanto la cantidad de experiencia gateando estuvo directamente relacionada con el grado de desincronizaciones mu y beta.

[1]Se denomina Síndrome de miembro fantasma al cuadro de sensaciones, dolor, picor, sensación térmica, que sienten algunas personas en un miembro amputado, que persiste pese a no tenerlo.

Después de controlar por la edad, no se encontraron correlaciones significativas entre el tamaño de los efectos de mu y beta y la cantidad de experiencia caminando.

La resonancia motora [neuronal] en la infancia es dependiente de la experiencia.

la adquisición de nuevas habilidades motoras en bebés, tales como aprender a gatear, está acompañada por una resonancia motora [neuronal] mayor durante la observación de acciones similares [a las que el bebé conoce]. El cerebro bebé transforma automáticamente la información visual en una representación motora que ya está establecida en su propio repertorio motor [25].

Estos estudios nos demuestran la justificación en la actividad neuronal que fortalece esta actividad y nos enseñan que es verdaderamente una técnica de aprendizaje eficiente para ser aplicada en Serious Games.

# Capítulo 4

# VRPN - Interconexión de Dispositivos para Realidad Virtual

## Introducción

Se describe la necesidad de contar con un set de herramientas de software / protocolo, que permita realizar las comunicaciones entre las partes de un sistema de realidad virtual, se plantea entonces el estudio de sus funcionalidades y uso en el proyecto.

Los sistemas de realidad virtual deben ser integrados tanto entre sus componentes de tipo software que conforman las interfaces visuales de representación digital, como así también los componentes de hardware tales como los sistemas de adquisición y análisis de datos, los movimientos o la detección de la voluntad de estos. Como se ha evaluado previamente en este trabajo, requieren comunicarse entre sí en tiempo real, en general estas comunicaciones responden a diseños particulares propios de cada fabricante y se hace difícil la integración de los mismos dado la heterogeneidad de las implementaciones realizadas por cada uno.

Los dispositivos generalmente están asociados a funciones o características de los individuos que los utilizan y se necesita integrar los movimientos que estos representan a la aplicación de realidad virtual correspondiente.

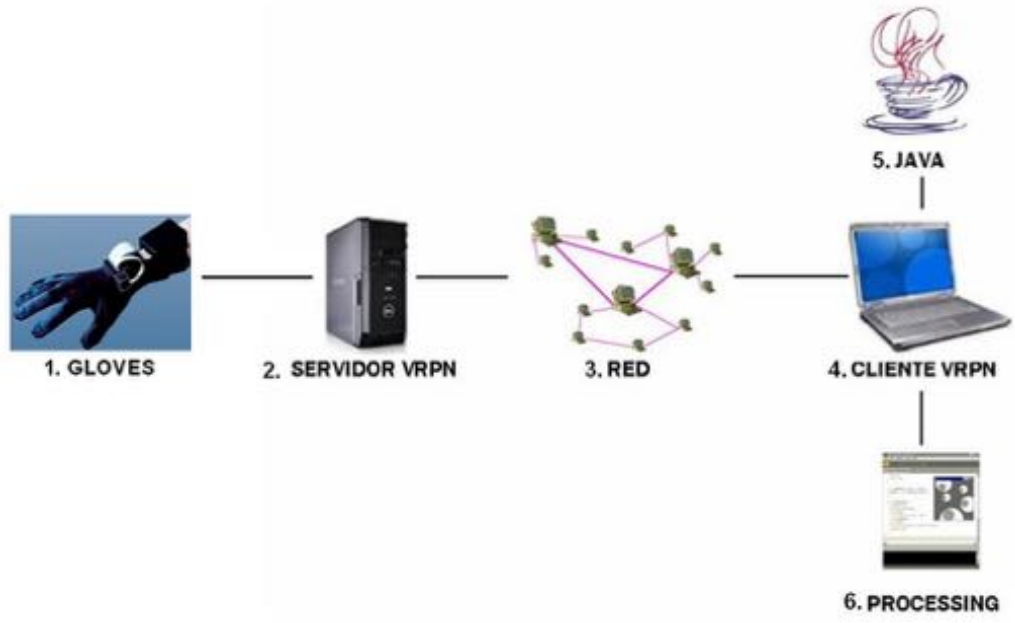
Para definir un método de acceso en forma directa de los dispositivos o por necesitar independencia entre ambos, es decir que las aplicaciones corran en un equipo y los dispositivos en otro.

Se necesita entonces de la integración de todo el sistema de realidad virtual, mediante la implementación de algún set de herramientas o protocolo de comunicaciones estándar que a su vez nos permita esta vinculación en tiempo real.

Para lograr todo esto existen una serie de herramientas y especificaciones llamadas VRPN, por ser tan necesaria su implementación es que el estudio del mismo forma

parte de los objetivos principales de este trabajo por lo tanto se procede a detallar y explicar a continuación, permitiendo de este modo su implementación de un modo mas fácil y guiado.

Figura 4.1: Esquema de trabajo de VRPN




**VRPN** (Virtual Reality Peripherical Network o Red de Perifericos para Realidad Virtual)

Es un set de herramientas Multiplataforma, de código abierto, libre y gratuito, que permite formar una red transparente de periféricos o dispositivos para su uso en realidad virtual, VRPN establece también un set de protocolos y especificaciones para poder vincular un grupo de dispositivos mediante un lenguaje común y poder comunicarse con estos enviando ordenes de comando y recibir estado y feedback de los mismos. De este modo interactuar formando un sistema de realidad virtual.

Fue desarrollado inicialmente en el centro de investigaciones CISMM (Computer Integrated Systems for Microscopy and Manipulation) de la Universidad de Carolina del Norte en Chapel Hill, luego pasó a ser desarrollado y mantenido por la empresa Sensics, quienes lo implementaron en el desarrollo de su plataforma de software de realidad virtual OSVR. Actualmente continúa su desarrollo y mantenimiento con los recursos de la empresa ReliaSolve, fundada por el autor original del proyecto VRPN, Russel M. Taylor, junto a una comunidad de colaboradores que participan en el proyecto actual alojado en Github [27].

Se compone de un conjunto de clases y librerías, que proveen las funciones e interfaces necesarias para funcionar a modo de programas servidores que nos permitan

interactuar entre las aplicaciones y los dispositivos físicos, usados en un sistema de realidad virtual. De este modo se tiene una pc que controla los dispositivos y estas puedan estar comunicadas entre sí formando un sistema intercomunicado.

No pretende constituir una API de realidad virtual, sino ser una interfaz unificada para acceder a un amplio rango de dispositivos, permitiendo el acceso a estos de un modo robusto, de baja latencia y de conectividad transparente.

Provee de las conexiones necesarias entre las aplicaciones y todos los dispositivos de un mismo tipo que compartan el link de conexión, de este modo las aplicaciones desconocen la topología de red utilizada, Teniendo en cuenta que es posible usar VRPN con dispositivos que son conectados directamente a la máquina donde corre la aplicación, ya sea usando programas de control separados o todos en un mismo programa.

Permite abstraerse de la capa del driver que maneja el dispositivo, y por lo tanto una aplicación que "hable" VRPN puede interactuar con un dispositivo compatible, sin necesidad de conocer sus especificaciones técnicas ni su lógica de control a nivel de driver, es decir que en este caso VRPN actúa como una capa de middleware, entre el driver del dispositivo y las aplicaciones que necesiten acceder al mismo. Los dispositivos finales se agrupan en los siguientes tipos: Analogs, Buttons y Trackers.

**Analogs:** Entradas analógicas, también algunos dispositivos de seguimiento y control pueden ser representados en forma de valores analógicos. Fig.4.2

**Buttons:** Son dispositivos de botones, se obtienen de estos valores lógicos o binarios, son entradas digitales. Fig.4.3 y Fig.4.4

**Trackers:** Son Dispositivos de seguimiento y localización que se utilizan para identificar la ubicación y orientación del individuo, también el punto de vista de este con todas las variables asociadas que permiten representar la realidad. Fig.4.5 y Fig.4.6

En forma nativa VRPN soporta numerosos dispositivos como: Joysticks, FOB [1] de la empresa Ascension, Tracker fastrak de la empresa Polhemus, Trackers de Intersense, Wii Remote de Nintendo, y otros.

A todos estos dispositivos se les provee una capa de abstracción que permite su control por software.

El lado servidor de VRPN puede usarse en diversas plataformas, incluyendo SGI/Irix, PC/Win32, PC/Linux, Mac/OSX y otros.

[1] FOB por sus siglas en inglés: FlocOfBirds o sensores magnéticos de rastreo de movimiento.

Figura 4.2: Dispositivo de Tracking de dedos

Figura 4.3: Mando de control de juego

Figura 4.4: Dispositivo Vara Mágica / Magic Wand

Figura 4.5: Dispositivo de Tracking de cabeza

Figura 4.6: Lentes de realidad virtual

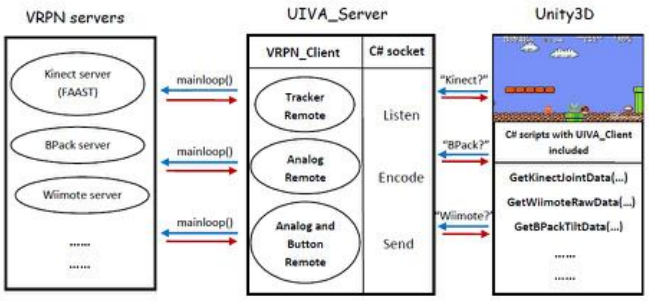
## Como trabaja VRPN:

VRPN provee tanto el lado Servidor como Cliente, los cuales pueden comunicarse entre sí mediante el protocolo TCP/IP, Cliente y servidor pueden ejecutarse en diferentes computadoras incluso de diferentes sistemas operativos.

VRPN server debe ejecutarse en la computadora que tiene conectados los dispositivos.

VRPN cliente debe incluirse junto al software de realidad virtual que necesita adquirir los datos de los dispositivos de entrada.

Figura 4.7: Esquema UIVA - VRPN - Unity




Los programas servidores están compilados uno diferente para cada dispositivo, y por cada arquitectura, en general sólo deberían parametrizarse y reconfigurarse mediante los archivos de configuración correspondiente, de acuerdo a los datos particulares de cada uno, direcciones de red, puertos, y parámetros de adquisición. No obstante la diversidad de dispositivos hace que haya que trabajar desarrollando nuevos servidores para adaptarlos a los dispositivos no soportados continuamente.

## Nota importante:

Desde la versión de VRPN 7.32, se introdujo una primera versión de soporte al desarrollo en dispositivos móviles de las clases clientes en java para poder trabajar en app nativas para android.

Esto es muy importante ya que nos permite desarrollar una aplicación nativa en android que se conecte con un servidor en una computadora y por intermedio de este a los dispositivos conectados, de este modo poder representarlos e interactuar con ellos en el dispositivo móvil.

## Como se establece la conectividad:

Todo gira en torno a la clase vrpn_Connection, esta clase se encarga de enviar y recibir mensajes entre cliente y servidor, los mensajes son enviados en modo asincrónico usando callbacks, en el código de la aplicación normalmente no se instancia esta clase en forma directa sino que se lo hace a través de las clases de los correspondientes dispositivos, en el código del servidor generalmente crea un socket con esta clase y luego llama a su función mainloop() permanentemente pero no trata con la clase directamente tampoco. Es la clase específica del dispositivo la que interactúa internamente con la clase vrpn_Connection, de este modo las clases de dispositivos son las que deben comprender minuciosamente su funcionamiento para poder interactuar con ella.

Una clase que deriva de otra existente específica de dispositivo, no debería lidiar con la clase vrpn_Connection, ya que la clase base es la que lo hará intercambiando mensajes y haciendo de sender. Si la clase derivada tuviera que enviar mensajes que no puede hacer la clase padre entonces debe describir estos mensajes a la conexión. Igualmente si puede recibir mensajes directamente, deberá registrar callbacks handlers para estos mensajes.

Todos los mensajes en VRPN tienen 3 datos, el timestamp, el sender correspondiente y type asociado al mensaje. El timestamp es definido por el sender, y usualmente corresponde al momento en el cual la acción que genera el mensaje ocurre, hora de clock local.

Por ejemplo la clase vrpn_Button reporta el timestamp del momento en que el botón es presionado o liberado.

El sender de un mensaje hace referencia al canal del dispositivo que se está enviando o recibiendo el mensaje, Por ejemplo en este caso sería “Button0”.

El type del mensaje sería “Button Toggle” que indica que un botón fue apretado o soltado .

Todos los senders y types se especifican usando un "string name" alfanumérico, pero en realidad son accedidos por un token.

Antes de que puedan ser usados los nombres deben ser registrados con la conexión para obtener un token. Inicialmente solo hay 2 tipos de conexiones vrpn_Connection, una que escucha en un socket y otra que establece una conexión remota a las que escuchan, en el futuro habrán diferentes clases para comunicaciones sharedmemory y para conexiones que tratan con sólo 1 proceso.

## Ejemplo de uso de vrpn_Connection

Este ejemplo crea un servidor VRPN que emite un mensaje test a cualquiera que se conecte a él.

También registra un callback para manejar cada mensaje que es enviado. Este callback es llamado tanto para los mensajes enviados en forma local tanto para cualquier otro enviado por una conexión remota. Los mensajes son enviados en una frecuencia de 1 mensaje por segundo.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/time.h>
#include <vrpn_Connection.h>
// Este es el callback handler que imprime el string de cada
// mensaje entrante (y también de los locales).
int my_handler(void *userdata, vrpn_HANDLERPARAM p) {
        char    *prompt = (char*)userdata;      // casteado al tipo correcto
        printf("%s:%s\n", prompt, p.buffer);    // Asumo que se trata de un mensa-
je string
        return 0;
}
main() {
        // Abro una conexion y registro el sender name y el type del mensaje.
        //
        vrpn_Connection *connection = new vrpn_Connection();
        long my_type = connection->register_message_type("Test_type");
        long my_id = connection->register_sender("Test0");
        // Registro el callback handler
        connection->register_handler(my_type, my_handler,
                                     (void*)"Test message", my_id);
        // Una vez por segundo, empaqueto un mensaje y llamo a mainloop() para ha-
cer que
        // se envie, también para llamar al callback for cualquier mensaje entrante.
        while (1) {
                struct timeval now;
                gettimeofday(&now,NULL);
                connection->pack_message(sizeof("Hola!"), now, my_type, my_id,
                         "Hola!", vrpn_CONNECTION_RELIABLE);
                connection->mainloop();
sleep(1); // espera de 1 segundo
        }
}
```

**Algoritmo 1:** vrpn_Connection

La versión remota del ejemplo es igual, excepto que se debería pasar el nombre del server al constructor de vrpn_Connection.

Nótese que los dispositivos forman una capa de abstracción encima de la clase vrpn_Connection, ocultando los detalles del empaquetado y desempaquetado de los mensajes entre el código de la aplicación y servidor, es posible para un código a nivel de usuario poder acceder a una conexión directamente, como se detalla arriba, VRPN puede ser usado como un sistema de intercambio de mensajes entre aplicaciones de cualquier otro tipo.

# Parte III

# Desarrollo de la implementación

# Capítulo 5

# Desarrollo e implementación de los prototipos de hardware y software

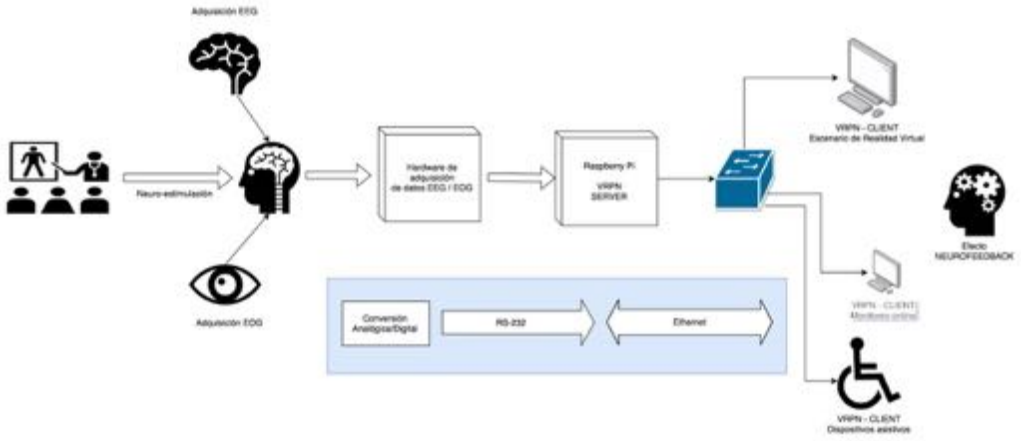
## Introducción

Se presenta el diseño de los prototipos a desarrollar para la implementación completa del sistema de adquisición y representación visual, se plantea el desarrollo de las funciones necesarias para comunicarnos mediante las especificaciones de las herramientas de VRPN, a bajo nivel, desde los equipos de adquisición y análisis de datos de usuario, conformando el esquema VRPN Server, el cual ofrecerá los flujos de datos que luego a nivel de software de interfaces gráficas, deberán consumirlos para que finalmente sean representados en la interfaz prototipo planteada.

Luego de abordar todo lo necesario debemos encargarnos del desarrollo de nuestro sistema, el cual se distribuye modularmente de la siguiente manera, según se ilustra en la imagen 5.1

- Adquisición y análisis de datos: Inicialmente el sistema debe recibir las señales que provienen de los sensores de EEG y EOG, Analizar el comportamiento de estas y generar la trama de información que se transmitirá hacia el módulo de generación VRPN-SERVER mediante niveles de tensión TTL y vía protocolo RS-232.
- Servidor VRPN: Esta es la etapa más importante del sistema, ya que en este módulo luego de la recepción de la trama de información generada por el módulo de adquisición de datos vía RS-232, se realiza el procesamiento de la misma y mediante el desarrollo de un driver se incorpora la información recibida al sistema VRPN-SERVER conformando los canales correspondientes para poder ser transmitido a quien solicite el flujo de datos creado.

Figura 5.1: Esquema modular del sistema propuesto




- Clientes VRPN: En esta etapa nos encontramos con que nuestro servidor VRPN ya está ofreciendo los flujos de información para los clientes que deseen conectarse y consumirlos, por lo tanto vamos a desarrollar diferentes alternativas que nos permitan conectar a nuestro servidor VRPN e interactuar con la información suministrada por el mismo en tiempo real proveniente de la adquisición de datos original, por lo tanto se plantean las siguientes sub etapas en este módulo:

    - Implementar la depuración del sistema mediante el monitoreo y muestra del detalle de información que se recibe en el stream de datos del canal al cual nos conectamos con los clientes VRPN.
    - Un escenario de realidad virtual prototipo que permita ilustrar el concepto planteado de un sistema de juego serio en el cual se realice una capacitación a un paciente con un trastorno cognitivo y que sirva para aplicar el concepto de neurofeedback al ver reflejado el individuo su voluntad en el escenario virtual.

Como complemento a la implementación realizada y presentada de nuestro sistema se presentan las prácticas iniciales de en este trabajo, las cuales se realizaron partiendo de la adquisición de datos con el sensor Kinect mediante las librerías FAAST que nos permiten interactuar con el sensor y crearnos el streaming de flujo de datos VRPN SERVER, el cual consumimos con nuestro escenario virtual desarrollado en el software Vizard.

## Diseño de una BCI partiendo de la captura de señales de EOG y EEG de imaginería motora.

Siguiendo la experiencia de numerosos estudios sobre el caso y con la intención de mejorar las técnicas utilizadas en ellos, tal como el estudio de Bautista [16] y muchos otros [28, 29, 30, 17, 31], se pretende realizar la captura de movimientos oculares o EOG, para poder identificar la dirección del movimiento deseado por el individuo, y tomar como disparo de la voluntad de realizar el mismo mediante un par de electrodos bipolares ubicados en C3 o C4, éstos nos deben indicar de acuerdo a su amplitud, el deseo de iniciar el movimiento correspondiente, esto corresponde a las técnicas de análisis de imaginería motora.

El prototipo a desarrollar consiste en un avatar humano y el movimiento de una mano virtual que desea agarrar un objeto y trasladarlo desde un punto A hasta otro B.

El proceso es el siguiente:

Se inicia el movimiento cuando se detecta activación en C3, mientras se mantenga en actividad el movimiento continúa en movimiento.

La dirección del movimiento se determina por lo que indique el análisis de EOG en este instante, no se requiere de alta definición en la posición exacta del destino hacia donde debe moverse la mano, solo basta con saber la dirección básica, para luego ir cambiando la misma sobre la marcha, ya que el movimiento que se reflejará es lento, por lo tanto permite realizar múltiples correcciones de la dirección del objetivo.

Para indicar que se ha llegado al objeto que se desea agarrar, se debería haber desactivado C3, y al estar en coincidencia con el objeto a agarrar, comienza un conteo de 3 segundos, en forma visual, indicando al individuo que se está en función de agarre, si no fuera lo deseado entonces el individuo debería activando C3 realizar un nuevo movimiento.

Se va produciendo un efecto de neurofeedback en el aprendizaje de uso del comando, por lo que el paciente comienza a regular el movimiento del ojo cuando ve representado como reacciona el sistema a su movimiento, de este modo entonces va aprendiendo a comandarlo con mayor precisión, el paciente entones logra de este modo calibrarse a si mismo para el uso del sistema mediante este entrenamiento.

Luego de los 3 segundos que la mano se encuentra sobre el objeto, éste queda linkeado a la mano y comienza el proceso que consiste en llevar el objeto al lugar destino.

El paciente entonces debe activar nuevamente C3 para que sea detectado por el sistema y éste se repite el proceso de análisis de EOG para determinar la dirección del movimiento, este movimiento va a ser corregido durante la marcha, en forma lenta hacia el punto que se está indicando en forma permanente mediante EOG.

Cuando se detecta la desactivación de C3, y si el objeto se encuentra dentro del radio de destino final, entonces se identifica como el fin del recorrido y comienza el proceso de confirmación del movimiento esperando y recorriendo visualmente los 3 segundos correspondientes que determinarán que el objeto agarrado inicialmente es ahora liberado para quedar depositado en el destino final.

# Sistema de Adquisición y Análisis de EEG/EOG

Como base para la adquisición de datos de este proyecto, se tomó el prototipo realizado por el ingeniero José Moran para su próxima tesis de Bioingeniería, el prototipo presentado por el tesista plantea un circuito electrónico basado en microcontroladores y conversores A/D, Amplificadores y con la ayuda de multiplexores que permitan adquirir las señales de 32 electrodos monopolares o bipolares, estos pueden ser los correspondientes a adquisición tanto de EEG como de EOG según lo que necesitemos.

## 5.1. Presentación del prototipo de adquisición de EEG/EOG

### 5.1.1. Cabezal de electrodos en disposición 10-20

Si bien para nuestro trabajo no se necesitan muchos electrodos y podrían usarse perfectamente estos electrodos de tipo copa de la figura 5.5, Para una correcta ubicación de los electrodos según la disposición internacional 10-20, es preferible valernos de un cabezal que tenga preubicados los electrodos o base de los mismos, en los lugares correspondientes, para de esta forma no equivocar su utilización, aunque no necesitemos ubicar todos ellos siempre conviene utilizar uno.

Como cabezal a emplear se probaron diversos prototipos tal como el primero de ellos de la figura 5.3, este tipo gorro, utiliza electrodos tipo punta o banana, no es muy preciso en su registro por lo que se descartó.

El segundo cabezal que se utilizó corresponde a otro prototipo que forma parte de un desarrollo del equipo de Neurotecnologías de Neuromed S.A., este cabezal que se observa en la figura 5.4, soporta la ubicación de unos electrodos de múltiples contactos.

Figura 5.2: Electrodos tipo copa

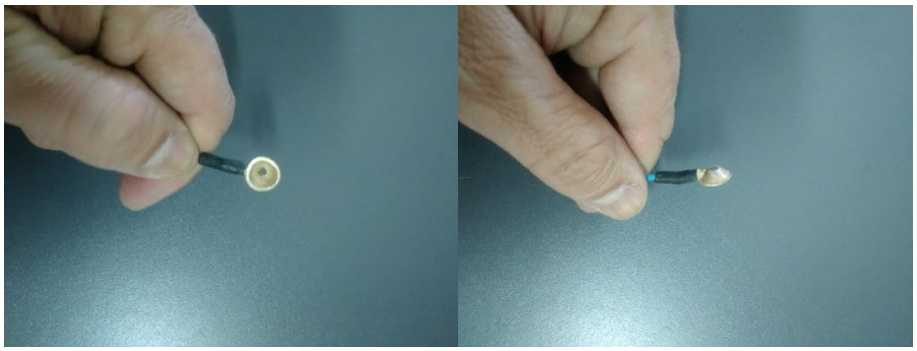


Figura 5.3: Gorro cabezal NNT prototipo1

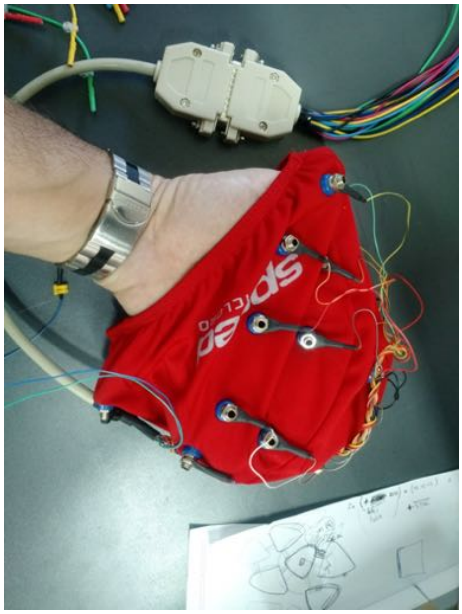
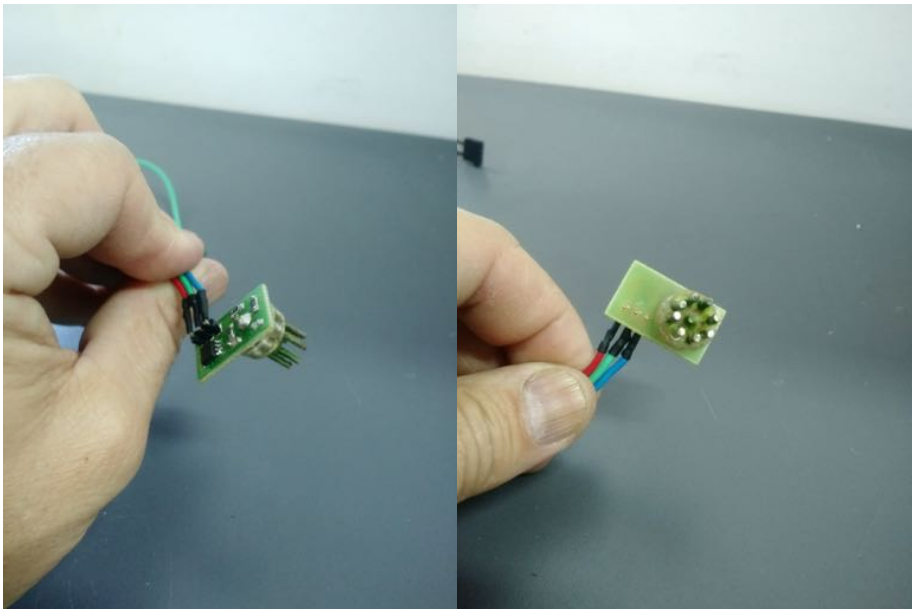


Figura 5.4: Gorro cabezal NNT prototipo 2

Figura 5.5: Electrodos activos prototipos



### 5.1.2. Hardware de adquisición y conversión A/D

El hardware para la adquisición y preprocesado de las señales útiles para el presente proyecto se encuentra en desarrollo por lo que se puede visualizar en la figura 5.6. Estas placas ensambladas superpuestas y serán las responsables de la adquisición de las señales.

Figura 5.6: Placas de adquisición prototipo

El diseño está basado en el microcontrolador de Microchip PIC18F4550 o PIC16F877, se utiliza su conversor A/D mediante un multiplexor de 16 canales analógicos 4067N para manejar los 16 electrodos bipolares , los que primero pasan por un amplificador de instrumentación de bajo consumo AD620R, el cual permite el ajuste de ganancia correspondiente para lograr el nivel de ingreso necesarios, para luego en el microcontrolador filtrar, y preprocesar estas señales y enviarlas al hardware encargado de recibirlas, analizar las señales recibidas y generar los comandos correspondientes en el protocolo VRPN sobre ethernet, esta ultima etapa se detalle en el capitulo siguiente.

Esta placa receptora y preprocesadora de señales tiene que analizar 2 electrodos monopolares que son c3 y c4, con respecto a una referencia cada uno, esto nos va a permitir según lo demostrado anteriormente, la voluntad de la acción motora, de acuerdo a las técnicas de imaginería motora correspondientes.

Por lo tanto el microcontrolador debe analizar la amplitud de c3 y c4 y evaluar el momento de la desincronización que es lo que le llamamos activación, entonces mientras tengamos una activación c3/c4 decimos que el movimiento se encuentra en marcha, y cuando se desactiven c3/c4 diremos que el movimiento o la voluntad del mismo, se detiene.

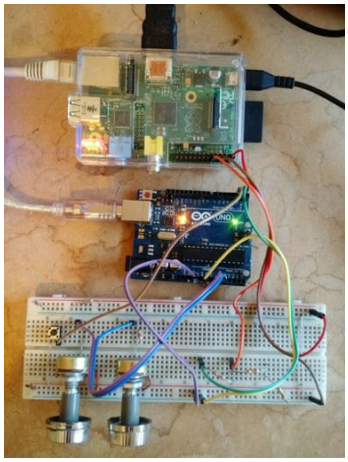
**Simulación de la funcionalidad de la placa de adquisición y digitalización**

Debido a que el hardware de adquisición y conversión A/D no se encuentra terminado ya que es producto de otro proyecto correspondiente al futuro bioingeniero José Moran, para poder generar las señales de entrada a nuestro proyecto VRPN y poder representar el procesamiento y las salidas hacia las interfaces gráficas para este proyecto, estas señales base EOG/EEG se las simula mediante una placa microcontroladora del tipo Arduino, esta nos permite generar las salidas necesarias y formar la trama correspondiente, para que luego de recibidas poder generar los comandos en el protocolo VRPN definitivo.

El equipo Arduino entonces recibe sus señales de 2 potenciómetros uno genera el movimiento horizontal y el otro el vertical de la adquisición y análisis EOG, además recibe mediante un pulsador la activación o desactivación del sistema de imaginería motora compuesto de los electrodos C3/C4 que determinan el movimiento, existe una tercer señal analógica que indica el nivel de confirmación de la señal digital de imaginería motora, para poder analizar en la interfaz final si el movimiento es realmente voluntario y en que grado de certeza.

Figura 5.7.

Figura 5.7: Prototipo Arduino - Raspberry Pi



El desarrollo del código del prototipo generador de tramas que alimentará al servidor VRPN fue desarrollado en El entorno de desarrollo por defecto de Arduino, utilizando su versión de C/C++ para la programación de las sentencias que componen el firmware de este, ver figura 5.8.

En el desarrollo del prototipo generador de tramas hacia Raspberry se generan las coordenadas cartesianas correspondientes que permiten identificar claramente el vector de dirección determinado para simular el movimiento, Esto lo podemos visualizar al seleccionar el switch correspondiente a la generación de tramas humanizadas desde

Figura 5.8: Arduino IDE

Arduino, de este modo podemos ver las coordenadas correspondientes que se generan, previo al envío de la trama pura hacia VRPN, para el envío de estas tramas binarias que son interpretadas por VRPN debemos seleccionar el switch correspondiente, ver Figura 5.9.

**Etapa de Generación Analógica/digital** Mediante el prototipo Arduino se reciben las señales correspondientes a valores de posición Horizontal, Vertical, Nivel de certeza y confirmación digital de actividad motora.

Primero se declaran las variables necesarias y se configuran las entradas Analógicas y digitales que necesitamos, esto lo hacemos en la función setup()...

```
byte pushButton=2, selectDebug=3, LedStat=12, LedDebug=13;
short int m, quadr, qVer, qHor;
short int EogVer, EogHor, C34Level;
float val1, val2, val3;
String StrDebug;
byte debugState=0, buttonState=0;
void setup() {
  Serial.begin(9600);
  pinMode(pushButton,INPUT);
  pinMode(selectDebug,INPUT);
  pinMode(LedStat,OUTPUT);
  pinMode(LedDebug,OUTPUT);
  buttonState=1;
  debugState=1;
}
```

**Algoritmo 2:** Etapa de Generación Analógica/digital - Función setup()

Luego en nuestro loop() principal repetitivo, hacemos lectura de las variables analógicas y digitales que necesitamos,

En el caso de las 3 analógicas, luego de recibirlas tenemos que convertirlas a nuestra escala definida de 0 a 1000, ya que son los valores que vamos a manejar en nuestros 4 cuadrantes, cada uno de ellos va a manejar 500 unidades (ver figura 5.9):

**Cuadrante1:** (sup. derecho) x:[0-500],y:[0-500]

**Cuadrante2:** (sup. izquierdo) x:[(-500)-0],y:[0-500]

**Cuadrante3:** (inf. izquierdo) x:[(-500)-0],y:[(-500)-0]

**Cuadrante4:** (inf. derecho) x:[0-500],y:[(-500)-0]

```
void loop() {
  val1 = analogRead(0);                    //Horizontal EOG
  val2 = analogRead(1);                    //Vertical EOG
  val3 = analogRead(2);                    //Level of C3/C4 certain
  buttonState = digitalRead(pushButton);   //button for C3/C4 activation
  debugState = digitalRead(selectDebug);   //debug switch selector
  EogHor = (int) val1 * (1000.0 / 1023.0);
  EogVer = (int) val2 * (1000.0 / 1023.0);
  C34Level = (int) val3 * (1000.0 / 1023.0);
```

**Algoritmo 3:** Etapa de Generación Analógica/digital - Función loop()

En caso de que necesitemos ver la generación de estos valores en los cuadrantes correspondientes, seleccionamos el switch correspondiente a la entrada digital "debugstate" con lo cual activamos la composición de la trama humanizada de salida.

En este caso entonces activamos el led de debug en la placa, para que de este modo sepamos visualmente que la trama de salida es humanizada y no nos sirve para ser enviada a VRPN, esto es para el análisis de la adquisición de las analógicas y digitales correspondientes.

Para el caso de nuestra trama pura binaria a ser enviada al servidor VRPN, necesitamos conformarla mediante un inicio de "*" y un final de trama: "#" con un fin de linea "\n".

y como primer dato útil enviamos la activación de c3/c4:

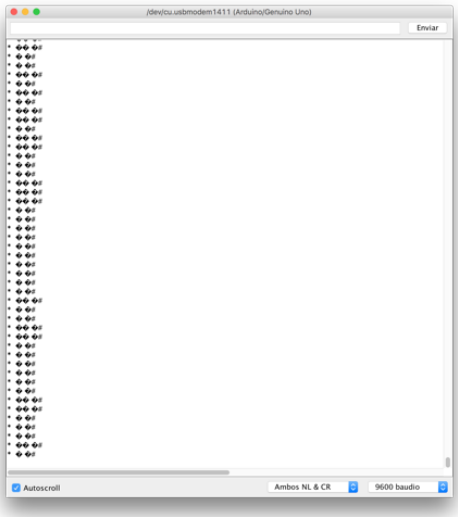
Los valores analógicos al ser de 10 bits y estar almacenados en variables enteras de 16bits, necesitamos separarlos en grupos de 8 bits para ser transmitidos y luego en el servidor VRPN vamos a realizar la conversión inversa para recuperar el valor original enviado.

Realizamos entonces las 3 conversiones y los enviamos en la trama,

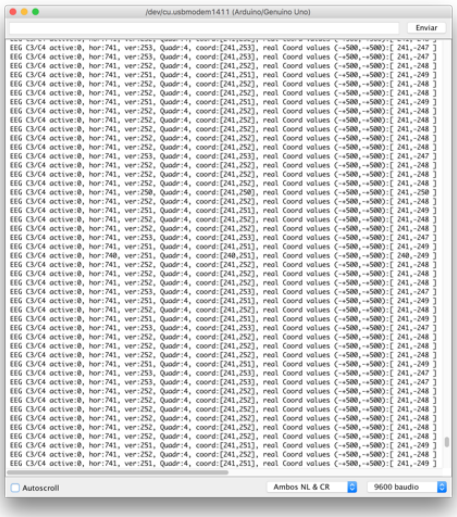
y como últimos caracteres se envían un final de trama: "#" con un fin de linea "\n".

Finalmente realizamos un debug visual de la actividad de comunicaciones de nuestra placa Arduino, que además nos sirve para agregar un delay al envío sucesivo de las mismas y evitar que nuestra pequeña cpu se sature y que las comunicaciones se sobrecarguen en destino...

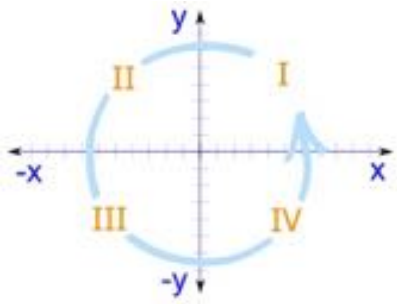
Figura 5.9: a) Trama binaria para VRPN b) Trama Humanizada desde Arduino c) Esquema de cuadrantes utilizado



a)



b)




c)

```
     //Modo depuración...
if(debugState){
    /*
     *      EogEeg frame - Human readable for debug
     */
    StrDebug= " EEG C3/C4 active:";
    StrDebug += buttonState;
    StrDebug += ", hor:";
    StrDebug += EogHor;
    StrDebug += ", ver:";
    StrDebug += EogVer;
    if(EogVer<500){
      qVer=EogVer;
      if(EogHor<500){
        quadr=3;
        qHor=EogHor;
      }else{
        quadr=4;
        qHor=EogHor-500;
      }
    }else{
      qVer=EogVer-500;
      if(EogHor<500){
        quadr=2;
        qHor=EogHor;
      }else{
        quadr=1;
        qHor=EogHor-500;
      }
    }
    StrDebug += ", Quadr:";
    StrDebug += quadr;
    StrDebug += ", coord:[";
    StrDebug += qHor;
    StrDebug += ",";
    StrDebug += qVer;
    StrDebug += "]";
    StrDebug += ", real Coord values (-+500,-+500):[ ";
    StrDebug += EogHor-500;
    StrDebug += ",";
    StrDebug += EogVer-500;
    StrDebug += " ]";
    digitalWrite(LedDebug, HIGH);      // Led debug on

    Serial.println(StrDebug);
```

**Algoritmo 4:** Etapa de Generación Analógica/digital - modo depuración

```
  Serial.write('*');
  Serial.write(buttonState+21);
  Serial.write((byte)((EogHor & 0xFF00)>>8));
  Serial.write((byte)(EogHor & 0x00FF));
  Serial.write((byte)((EogVer & 0xFF00)>>8));
  Serial.write((byte)(EogVer & 0x00FF));

  Serial.write((byte)((C34Level & 0xFF00)>>8));
  Serial.write((byte)(C34Level & 0x00FF));
  Serial.write('#');
  Serial.write("\n");
  delay(50);
  digitalWrite(LedStat, HIGH);
  delay(50);
  digitalWrite(LedStat, LOW); delay(50);
  digitalWrite(LedStat, HIGH);
  delay(50);
  digitalWrite(LedStat, LOW);
```

**Algoritmo 5:** Etapa de Generación Analógica/digital - modo binario puro

### 5.1.3. Hardware de recepción de tramas de señales preprocesadas y generación de vrpn.

El hardware utilizado para la recepción de las tramas de datos para la posterior generación de comandos VRPN se compone de un prototipo de hardware Raspberry PI en su versión 2, que funciona perfectamente para nuestro proyecto, ya que se trata de una microcomputadora de propósitos generales que nos brinda conectividad serie para comunicarnos con la placa de electrodos y también nos brinda conectividad ethernet para poder comunicarnos con el exterior, sea cual fuere el equipo destino, ya que nos amplía las posibilidades de comunicación ilimitadamente permitiendo luego de acá en más utilizar routers, switch, cualquier dispositivo estándar ethernet, wifi, etc.

Este hardware RPi corre con sistema operativo base Linux, en este caso usamos Raspbian, que se trata de una versión de Debian para RPi. Esto nos abre el panorama a todo tipo de lenguajes y compiladores para procesar nuestras pruebas de generación VRPN sin problemas, C/C++, Python, etc, 5.10

Figura 5.10: Prototipo Raspberry Pi v2

## 5.2. Generación VRPN

### 5.2.1. Recepción de trama Serial

La trama se emite desde la placa de adquisición de acuerdo al protocolo rs232 en niveles de tensiones ttl, 0 a 5volt, y se recibe en el prototipo RPi mediante la uart del mismo, usando los pines de su multipuerto GPIO, tx (14), rx (15) y gnd, pero las tensiones que maneja éste son en el orden de los 3.3volt, es decir que se necesita adaptar estas tensiones entre 5 y 3.3.

Para llevar acabo esta adaptación de tensiones tenemos varias formas de hacerlo, la técnica correcta serías utilizando el integrado 74LC245, que trabaja convirtiendo los niveles entre 3.3 y 5v, necesitamos agregar un regulador de voltaje que a partir de 5v nos genere 3.3 para el chip adaptador, el LM7833 o el LM317 más unas resistencias de setup para lograr 3.3v, y armando un circuito adaptador de tensiones correspondientes.

Otra técnica simple consiste en armar un práctico circuito del tipo divisor de tensiones, esto se logra colocando una resistencia de un valor en serie y una en paralelo a masa del doble del valor anterior, digamos de 1k la resistencia en serie y 2k en paralelo a masa. Si bien esto depende de la impedancia de entrada del dispositivo receptor, ya que tendremos una caída en la corriente en función de las resistencias usadas, no obstante, para nuestro caso no hay problema y la caída es tolerada correctamente por el RPi.

Para procesar los datos que vamos a recibir en el equipo nos basaremos en el Framework VRPN, por lo tanto vamos a desarrollar un driver que nos permita recibir las tramas correspondientes y poder identificar los valores analógicos y digitales que se envían desde Arduino.

VRPN provee acceso a una gran variedad de dispositivos a través de interfaces extensibles. En el caso particular de este proyecto se usaron estas interfaces para crear un servidor para que se conecte y procese los eventos de nuestro dispositivo Arduino.

El driver que desarrollamos nos va a permitir ingresar el paquete de datos al sistema VRPN para que sean transmitidos hacia los dispositivos VRPN Client que lo soliciten, tal como vamos a hacer nosotros usando el programa Print_Devices (incluido en el paquete de VRPN) para verificar que efectivamente se están adquiriendo las lecturas adecuadamente.

### Desarrollo del driver VRPN Server

Las funciones VRPN Server del framework necesitan de Drivers que permitan interpretar a los dispositivos y se entiendan con ellos, por esto necesitamos volcar lo desarrollado en la generación de las tramas para poder hacer que se interpreten correctamente, lo hacemos en C++ y compilamos con gcc.

De toda la estructura de VRPN vamos a trabajar con algunos archivos que necesitamos modificar e incorporar a la misma.

dentro de la estructura principal VRPN, debemos generar nuestro driver, para la implementación del servidor se creó una librería especializada en obtener las lecturas del dispositivo Arduino. Esta librería consta de dos archivos:

vrpn_EogDevice.h

vrpn_EogDevice.C

Los anteriores archivos se ubican en el subproyecto vrpn. Adicionalmente, se modificó la implementación del Loop principal en el archivo vrpn_generic_server_objet.C :

**vrpn_EogDevice.h** Contiene las definiciones de las variables y las estructuras de datos a usar por la implementación:

```
#ifndef VRPN_EOGDEVICE_H
#define VRPN_EOGDEVICE_H
#include "vrpn_Analog.h"                // for vrpn_Serial_Analog
#include "vrpn_Button.h"                // for vrpn_Button_Filter
#include "vrpn_Configure.h"             // for VRPN_API
#include "vrpn_Connection.h"            // for vrpn_CONNECTION_LOW_LATENCY, etc
#include "vrpn_Dial.h"                  // for vrpn_Dial
#include "vrpn_Shared.h"                // for timeval
#include "vrpn_Types.h"                 // for vrpn_uint32
class VRPN_API vrpn_EogDevice: public vrpn_Serial_Analog ,public vrpn_Button_Filter
{
public:
vrpn_EogDevice (const char * name, vrpn_Connection * c,
const char * port, int baud);
~vrpn_EogDevice () {};
vrpn_EogDevice (const char * name, const char * port, vrpn_Connection * c);
virtual void mainloop ();
  protected:
int _status;
int _numbuttons; // How many buttons to open
int _numchannels; // How many analog channels to open
unsigned _expected_chars; // How many characters to expect in the report
unsigned char _buffer[512]; // Buffer of characters in report
unsigned _bufcount; // How many characters we have so far
struct timeval _timestamp; // Time of the last report from the device
virtual void clear_values(void); // Set all buttons, analogs and encoders back to 0
virtual int get_report(void); // Try to read a report from the device
     // send report iff changed
        virtual void report_changes
                   (vrpn_uint32 class_of_service
                    = vrpn_CONNECTION_LOW_LATENCY);
        // send report whether or not changed
        virtual void report
                   (vrpn_uint32 class_of_service
                    = vrpn_CONNECTION_LOW_LATENCY);
};
#endif
```

**Algoritmo 6:** Generación VRPN - vrpn_EogDevice.h

**vrpn_EogDevice.c** Este código implementa la inicialización del puerto, conexión, validación de la conexión y ejecución del loop principal donde se realiza la lectura del buffer para el parseo de las tramas recibidas con la información del estado de los sensores del dispositivo Arduino, para generar los callback correspondientes en vrpn:

Primero inicializamos y nos preparamos a recibir los datos:

Luego de la Inicialización del puerto y preparación de lo necesario para que la información alimente a nuestro buffer, comenzamos el parseo de los mismos.

```
unsigned char inbuf[45];
vrpn_Button::num_buttons = 1;
vrpn_Analog::num_channel = 3;
...
int i;
vrpn_Button::buttons[0] = vrpn_Button::lastbuttons[0] = 0;
for (i = 0; i < 3; i++) {
vrpn_Analog::channel[i] = vrpn_Analog::last[i] = 0;
}
...
El parseo de los datos al recibir una trama completa...
...
vrpn_flush_input_buffer(serial_fd);
printf("vrpn_EogDevice: start reading frame\n");
_expected_chars = 10;
...
     if ( _buffer[0] == '*') {
      printf("... Got the 1st char\n");
...
   if (_buffer[_expected_chars-1] == '\n') {
   printf("got a complete report (%d of%d)!\n", _bufcount, _expected_chars);
...
char bits = (char)(_buffer[nextchar++] - offset);
buttons[0] = ( (bits & 0x01) != 0);
    printf("buttons[0]=%d\n",buttons[0]);
_numchannels=3;
for (i = 0; i < _numchannels; i++) {
intval =  (int)((_buffer[nextchar++])<<8) | (int)((_buffer[nextchar++]));
realval = intval;
channel[i] = realval;
    printf("channel[%d]=%f\n",i,channel[i]);
}
...
y finalmente las llamadas a la generación de reportes, para que se invoquen los call-
backs correspondientes...
...
report_changes();
...
   printf("get report - capture end ok!\n");
...
```

**Algoritmo 7:** Generación VRPN - vrpn_EogDevice.c

Para la compilación de este driver necesitamos editar el archivo Makefile correspondiente en este mismo directorio, incluyendo los archivos que se agregaron al árbol de estructura VRPN main.

invocamos a make y este se encarga de compilar toda esta rama del framework con nuestro nuevo driver.

```
$ root@pi:.../vrpn# make
```

Para poder trabajar correctamente el sistema VRPN se configura mediante 1 archivo de configuración llamado vrpn.cfg que debe encontrarse junto a la aplicación vrpn_server, en este archivo de configuración se definen todos los drivers que se utilizarán y los parámetros de configuración de los mismos, como puerto, velocidad de la conexión en baudios, etc

en nuestro caso nuestra configuración es la siguiente:

```
...
vrpn_EogDevice   EogDev0          /dev/ttyAMA0     9600
...
```

Aquí estamos indicando que nuestro driver se llama: vrpn_EogDevice, el dispositivo vrpn server que se generará será EogDev0 y recibirá sus datos desde el puerto /dev/ttyAMA0 en el raspberry y la velocidad del puerto es de 9600 baudios/s.

Para poder procesar este archivo de configuración necesitamos ajustar dentro de la rama server_src el archivo vrpn_generic_server_objet.C y vrpn_generic_server_objet.h, para que se incluya a nuestro driver junto al resto de la estructura vrpn y vrpn server pueda trabajar correctamente con el mismo.

Estos definen cada uno de los drivers y prepara el servidor para utilizarlos.

**vrpn_generic_server_objet.C** Genera los callback a partir de la lecturas de los analógicos (EOG hor/ver, c3/c4 certeza) y el botón pulsador.

Agregamos el bloque de código que define nuestro nuevo driver.

debemos compilar la rama server_src, modificando previamente el archivo Makefile correspondiente y llamando a make para la recompilación de la rama completa.

```
$ root@pi:.../vrpn/server_src# make
```

Ya estamos en condiciones de probar nuestro driver, para lo cual vamos a ejecutar vrpn_server desde el directorio correspondiente a nuestra plataforma, en este caso ...pc_linux

```
...
// ##########################################
//              EOG DEVICE
// ##########################################
int vrpn_Generic_Server_Object::setup_EogDevice(char *&pch, char *line,
                                                FILE * /*config_file*/)
{
    char name[LINESIZE], port[LINESIZE];
    int baud;
    verbose=1;
    VRPN_CONFIG_NEXT();
    // Get the arguments (class, serialbox_name, port, baud)
    if (sscanf(pch, "%511s%511s%d", name, port, &baud) != 3) {
        fprintf(stderr, "Bad vrpn_EogDevice line:%s\n", line);
        return -1;
    }
    // Open the Eog
    if (verbose)
        printf("Opening vrpn_Eog:%s on port%s\n",name,port);

    _devices->add(new vrpn_EogDevice(name, connection, port, baud));
    return 0;
}
...
```

**Algoritmo 8:** Generación VRPN - vrpn_generic_server_objet.C

```
$ root@pi:.../vrpn/server_src/pc_linux#./vrpn_server -v
Reading from config file vrpn.cfg
Opening vrpn_Eog: EogDev0 on port /dev/ttyAMA0
vrpn_EogDevice: start reading frame
...
```

vrpn_server ya está procesando nuestras tramas desde Arduino:

para poder monitorear lo que está sucediendo vamos a utilizar una utilidad cliente vrpn_print_devices la cual nos permite mostrar la información disponible de consultar

```
$ ./vrpn_print_devices EogDev0@10.56.0.250
Opened EogDev0@10.56.0.250 as: Tracker Button Analog Dial Text.
Press ^C to exit.
Button EogDev0@10.56.0.250 has 1 buttons with states: 0
Analog EogDev0@10.56.0.250:          99.00, 253.00, 1000.00 (3 chans)
Analog EogDev0@10.56.0.250:          94.00, 250.00, 1000.00 (3 chans)
Analog EogDev0@10.56.0.250:          153.00, 250.00, 1000.00 (3 chans)
...
########################################
Button EogDev0@10.56.0.250, number 0 was just pressed
########################################
...
Analog EogDev0@10.56.0.250:          99.00, 252.00, 1000.00 (3 chans)
Analog EogDev0@10.56.0.250:          98.00, 252.00, 1000.00 (3 chans)
Analog EogDev0@10.56.0.250:          99.00, 255.00, 1000.00 (3 chans)
...
########################################
Button EogDev0@10.56.0.250, number 0 was just released
########################################
...
```

Esto nos confirma que vrpn_server está recibiendo correctamente nuestras tramas de datos enviadas desde Arduino.

pasamos entonces a la recepción de estos datos desde nuestra interfaz gráfica en Unity.

## 5.3. Implementación de la interfaz Unity cliente de los datos VRPN.

Para la representación modelo de los datos generados en nuestro servidor VRPN, se desarrolló una interfaz prototipo que pretende ilustrar la representación de los datos recibidos, pero en este caso no corresponde al entrenador virtual, solo es una prueba de concepto que permite establecer las bases para trabajar posteriormente en la interfaz de entrenamiento final.

Para poder recibir los datos se necesita implementar una interfaz que reciba los mismos vía VRPN Client y comunique estos como stream de datos en variables que pueda procesar Unity correctamente.

Para nuestro prototipo inicial llamamos a nuestras 3 analógicas como Yaw, Pitch y Roll para corresponder con los movimientos que le daremos a un Cubo el cual debe reproducir estos movimientos de acuerdo a las analógicas correspondientes, asimismo la entrada digital correspondiente a la activación C3/C4, en nuestra interfaz hará que el cubo se eleve dando un salto.

### 5.3.1. Desarrollo de la Interfaz cliente DLL

Se desarrolló un cliente de VRPN implementado en una librería Dinámica de Enlace - Dynamic Link Library (DLL)-.

Este cliente tiene como propósito servir como interfaz entre el servidor vrpn y nuestro prototipo en el motor de videojuegos (Unity).

Este cliente no realiza ningún tipo de procesamiento de información con los datos vrpn que recibe, sólo hace de interfaz entre vrpn y Unity.

Este se desarrolló en C++ por medio de Visual Studio y consta principalmente de los siguientes archivos:

**EogClientDevice.h:** Definición de variables y estructuras de datos para capturar los callback del servidor VRPN.

**EogClientStructs.h:** Definición de las estructuras de datos para pasar las lecturas analógicas y de la activación digital a Unity.

**EogClientDevice.cpp:** Acá es donde se implementa la captura de los callbacks generados por el servidor.

**EogClientDeviceInterface.cpp:** Se definen los encabezados de los métodos que pasan las estructuras de datos y el control a Unity.

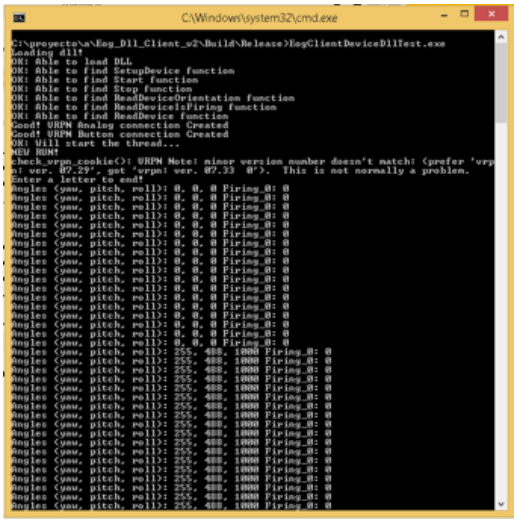
**EogClientDevice.cpp** Este es el código principal de la DLL donde se realiza la captura de los callback provenientes del servidor VRPN.

Paralelamente al desarrollo de la dll se programó una pequeña utilidad de test, la cual utiliza la librería cliente Eog dll generada previamente, para poder chequearla y comprobar la comunicación con el servidor y si todo está ok, poder ver en consola los datos adquiridos del servidor VRPN.

Esta misma función es la que cumpliría luego Unity, por eso es importante contar con esta utilidad que nos permita monitorear en esta etapa por si surgiera un error posterior en Unity.

el resultado de ejecutar el test se ilustra en la figura 5.11

Figura 5.11: Salida del test del cliente dll Eog




## 5.3.2. Interfaz Unity de representación y prueba

Finalmente para poder representar los datos generados desde VRPN, se desarrolló una interfaz prototipo en el motor Unity.

Esta interfaz no pretende ser utilizada como herramienta final de capacitación cognitiva, sino la misma debe permitir demostrar la representación de los datos enviados desde el prototipo Arduino inicial y comunicados vía VRPN.

Estos datos que inicialmente son enviados como 3 lecturas analógicas que representan las coordenadas EOG horizontales, verticales y una señal analógica que representa el nivel de certeza de imaginería junto con una señal digital de pulsador correspondiente. En nuestra interfaz de prueba no representan estos valores, sino que las tomamos como 3 parámetros de movimientos Yaw, Pitch y Roll, mediante los cuales movemos

```
#include "stdafx.h"
#include "stdio.h"
#include "EogClientDevice.h"
#include <signal.h>
#include <iostream>
EogClientDevice *EogClientDevice::smIntance = NULL;
EogClientDevice * EogClientDevice::GetInstance()
{
if( smIntance == NULL )
smIntance = new EogClientDevice;
return smIntance;
}
EogClientDevice::EogClientDevice(void)
{
mAnalog = NULL;
mButton = NULL;
button0 = false;
button1 = false;
button2 = false;
button3 = false;
button4 = false;
button5 = false;
button6 = false;
button7 = false;
}
EogClientDevice::~EogClientDevice(void)
{
Stop();
}
void EogClientDevice::Start()
{
stopDll = false;
// Create vrpn connections
mAnalog = new vrpn_Analog_Remote(deviceConnection.c_str());
if( mAnalog != NULL )
{
printf("Good! VRPN Analog connection Created\n");
mAnalog->register_change_handler(this, handle_analog);
}
else
{
printf("Error: cannot connect to analog device\n");
}
mButton = new vrpn_Button_Remote(deviceConnection.c_str());
if( mButton  != NULL )
{
printf("Good! VRPN Button connection Created\n");
mButton->register_change_handler(this, handle_button);
}
else
{
printf("Error: cannot connect to button device\n");
}
deviceThreadHandle = CreateThread(NULL, 0, &EogClientDevice::ThreadDevice, (LP-
VOID) this, 0, NULL);
}
void EogClientDevice::Stop()
{
stopDll = true;
if( deviceThreadHandle != NULL )
WaitForSingleObject(deviceThreadHandle, INFINITE);
deviceThreadHandle = NULL;
// Stop vrpn connections
if( mAnalog != NULL ){
mAnalog->unregister_change_handler( this, handle_analog );
delete mAnalog;
mAnalog = NULL;
}
if( mButton != NULL ){
mButton->unregister_change_handler( this, handle_button );
delete mButton;
mButton = NULL;
}
}
```

**Algoritmo 9:** Driver cliente VRPN - EogClientDevice.cpp (parte 1)

```
// ------------------------------------------------------------------------------
// Device methods
DWORD WINAPI EogClientDevice::ThreadDevice( LPVOID pThis )
{
//printf("We are into the ThreadDevice...\n");
EogClientDevice * device = (EogClientDevice*)pThis;
device->RunDevice();
return 0;
}
void EogClientDevice::SetupDevice( const char* deviceName, int yawChannel, int pitch-
Channel, int rollChannel, int deviceButton_0, int deviceButton_1, int deviceBut-
ton_2, int deviceButton_3, int deviceButton_4, int deviceButton_5, int deviceBut-
ton_6, int deviceButton_7)
{
this->deviceConnection = "";
this->deviceConnection.append(deviceName);
this->yawChannel = yawChannel;
this->pitchChannel = pitchChannel;
this->rollChannel = rollChannel;
this->deviceButton_0 = button0;
this->deviceButton_1 = button1;
this->deviceButton_2 = button2;
this->deviceButton_3 = button3;
this->deviceButton_4 = button4;
this->deviceButton_5 = button5;
this->deviceButton_6 = button6;
this->deviceButton_7 = button7;
}
void EogClientDevice::RunDevice()
{
while( !stopDll )
{
mButton->mainloop();
mAnalog->mainloop();
Sleep((DWORD)1);
}
ExitThread(0);
}
void VRPN_CALLBACK EogClientDevice::handle_button( void* userDa-
ta, const vrpn_BUTTONCB b )
{
EogClientDevice * device = (EogClientDevice *)userData;
device->ProcessButton( b );
}
void EogClientDevice::ProcessButton( const vrpn_BUTTONCB b )
{
if ( b.button == deviceButton_0 ) button0 = b.state == 1 ? true : false;
if ( b.button == deviceButton_1 ) button1 = b.state == 1 ? true : false;
if ( b.button == deviceButton_2 ) button2 = b.state == 1 ? true : false;
if ( b.button == deviceButton_3 ) button3 = b.state == 1 ? true : false;
if ( b.button == deviceButton_4 ) button4 = b.state == 1 ? true : false;
if ( b.button == deviceButton_5 ) button5 = b.state == 1 ? true : false;
if ( b.button == deviceButton_6 ) button6 = b.state == 1 ? true : false;
if ( b.button == deviceButton_7 ) button7 = b.state == 1 ? true : false;
}
void VRPN_CALLBACK EogClientDevice::handle_analog( void* userDa-
ta, const vrpn_ANALOGCB a )
{
EogClientDevice * device = (EogClientDevice *)userData;
device->ProcessAnalog( a );
}
void EogClientDevice::ProcessAnalog( const vrpn_ANALOGCB a )
{
if( a.num_channel < pitchChannel || a.num_channel < yawChan-
nel || a.num_channel < rollChannel )
return ;
yaw = (float)a.channel[yawChannel];
pitch = (float)a.channel[pitchChannel];
roll = (float)a.channel[rollChannel];
}
```

**Algoritmo 10:** Driver cliente VRPN - EogClientDevice.cpp (parte 2)

un cubo en pantalla y el pulsador recibido se representa a modo de trigger como un salto del cubo hacia arriba.

Esta lógica de representación utilizada permite la visualización de los datos recibidos en tiempo real mediante todo el sistema planteado, en la figura 5.12 podemos observar la interfaz de desarrollo Unity3D donde se aprecian las propiedades principales de nuestro cubo, y en la figura 5.13 podemos ver las imágenes correspondientes al cubo cuando es accionado por los comandos vrpn que le proporcionan el movimiento lateral o Yaw.

El desarrollo de la interfaz se basa en la utilización de un objeto Cubo, al cual se le asignó un script en C# que regula su comportamiento: CubeBehavior.cs (Ver código fuente en el apéndice 2) desarrollado en C# mediante el IDE Mono incluido en Unity.

En nuestro proyecto Unity debemos incorporar como plugins las librerías dll, tanto la desarrollada por nosotros: EogClientDeviceDll.dll como la librería dll del árbol del proyecto vrpn: vrpndll.dll la cual contiene las funciones vrpn clientes que son las que se comunican realmente con el servidor vrpn y accede a los datos correspondientes.

Figura 5.12: Desarrollo prototipo del Cubo móvil en Unity

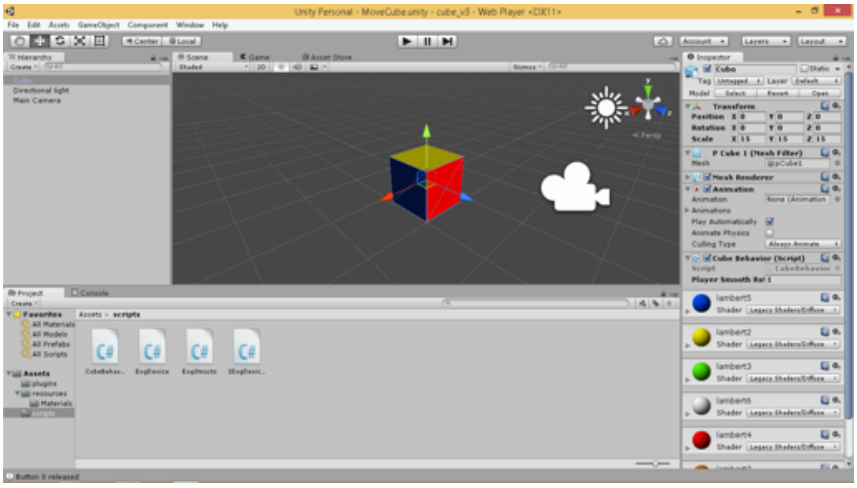
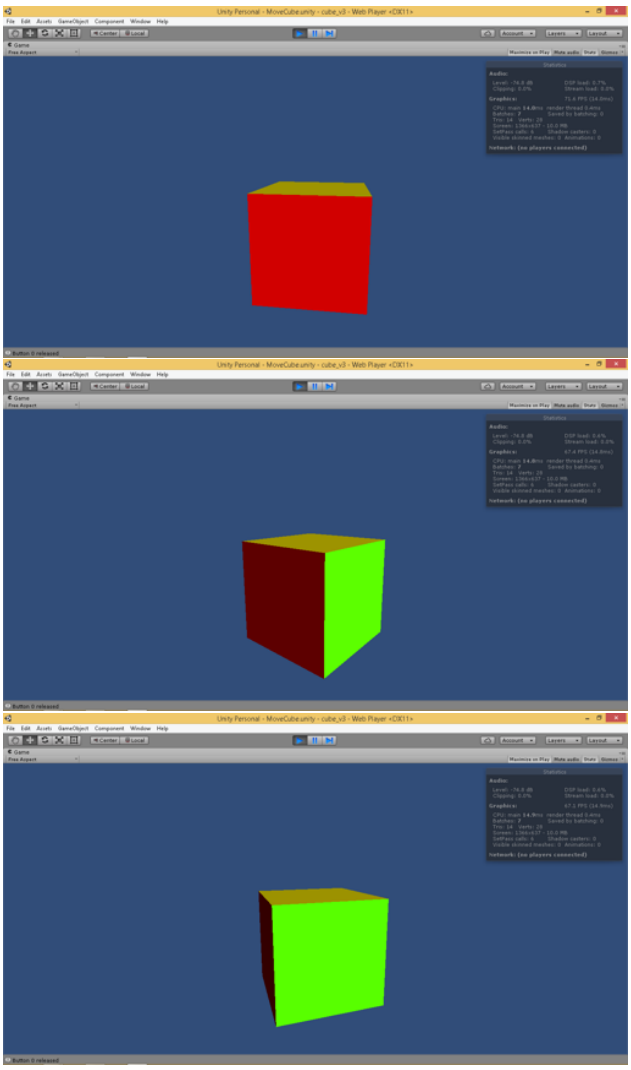


Figura 5.13: Cubo móvil en Unity rotando



# Capítulo 6

# Otras técnicas estudiadas y modelos de escenarios para rehabilitación cognitiva

## Introducción

Se detallan las prácticas realizadas sobre otros conjuntos de software o librerías que responden al estándar VRPN, tal como el driver FAAST, que permite la comunicación mediante el uso del sensor Microsoft Kinect, para la adquisición de las coyunturas del cuerpo humano, combinado con la utilización del software de diseño de interfaces virtuales Vizard, por otro lado, también se presenta un prototipo desarrollado con el software de animación Unity3D y VRPN, si bien Vizard posee un driver implementado en forma nativa de VRPN Cliente, no así Unity3D que se vale de las librerías de la implementación del paquete de software OSVR, como alternativa al desarrollo de driver cliente presentado en el capítulo anterior.

## Conjunto de Librerías FAAST

Por sus siglas en Inglés: Flexible Action and Articulated Skeleton Toolkit, FAAST, consiste en un set de herramientas y librerías desarrollados por un grupo de investigadores de la USC (University of Southern California) [32], su función es la de facilitar la integración del reconocimiento de cuerpo completo que realiza el sensor Microsoft Kinect, con juegos y aplicaciones de realidad virtual, FAAST incorpora un servidor de VRPN para ofrecer los flujos de datos a las aplicaciones que necesiten consumirlo, implementando éstas su parte de VRPN cliente para lograr la comunicación correspondiente.

Al momento sólo funciona bajo Windows y mediante una licencia de libre uso y distribución, pero no se proveen los fuentes correspondientes, por lo tanto no se pudo tomar el trabajo realizado para realizar algún tipo de contribución o continuación del mismo.

## 6.1. Pruebas de VRPN mediante Kinect y las librerías FAAST

Para poder trabajar en los escenarios virtuales, inicialmente necesitamos valernos de una fuente de datos VRPN confiable, que sea generada por alguna librería o software existente y nos permita liberarnos del problema de la generación correcta de comandos VRPN, para luego de desarrollada la interfaz 3d poder dedicarnos a realizar pruebas VRPN pero ya con el prototipo de escenario probado y funcionando tal como se realizó en la sección anterior con un prototipo de interfaz genérico tipo Cubo dinámico.

Utilizamos el sensor Kinect de Microsoft xbox con una interfaz de adaptación que nos permite usarlo en una pc.

Kinect nos permite capturar las extremidades humanas, y mediante la librería de adquisición FAAST generar trackers VRPN-server en forma directa sin necesidad de tener que programar un driver vrpn para acceder a los sensores Kinect en forma directa.

FAAST es un set de librerías que permiten trabajar con Kinect y brindar un módulo vrpn-server al cual nos conectaremos con cualquier aplicación que respete el estándar vrpn-cliente, ya sea en forma nativa o a travez de clientes dll o middleware que cumplan esta función.

Captura de datos mediante la utilidad cliente vrpn: vrpn_print_devices

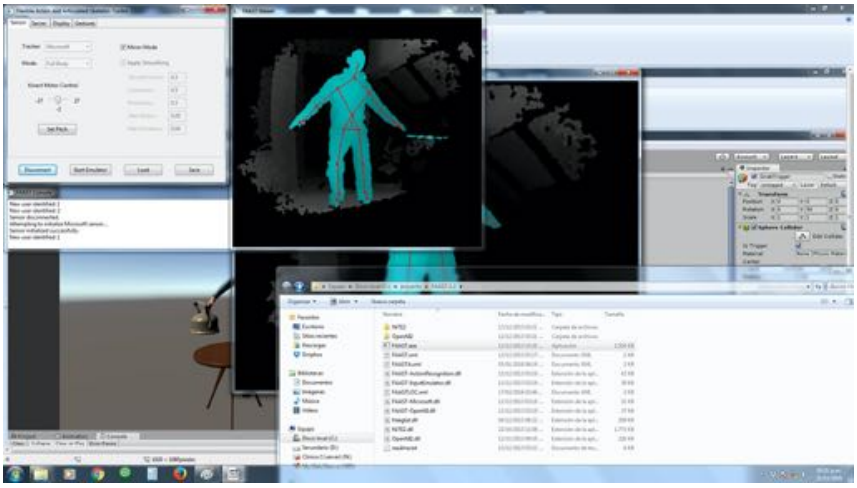
Se reciben los juegos de coordenadas correspondientes al Tracker0 y todos sus canales, correspondientes a todas las coyunturas que representan los movimientos del individuo 1 frente al sensor Kinect.

```
$ ./vrpn_print_devices Tracker0@10.56.0.63
Tracker Tracker0@10.56.0.63, sensor 0:
        pos ( 0.00,  0.14,  0.01); quat ( 0.00,  0.00, -0.00,  1.00)
Tracker Tracker0@10.56.0.63, sensor 1:
        pos ( 0.00,  0.21,  0.00); quat ( 0.03,  0.00, -0.09,  0.99)
Tracker Tracker0@10.56.0.63, sensor 2:
        pos (-0.00,  0.04, -0.00); quat ( 0.01,  0.00, -0.01,  1.00)
Tracker Tracker0@10.56.0.63, sensor 3:
        pos ( 0.07, -0.12,  1.08); quat (-0.09, -0.21, -0.06,  0.97)
Tracker Tracker0@10.56.0.63, sensor 4:
        pos (-0.09, -0.10, -0.00); quat ( 0.01, -0.07,  0.57,  0.83)
Tracker Tracker0@10.56.0.63, sensor 5:
        pos (-0.18,  0.00, -0.00); quat (-0.79, -0.53, -0.05,  0.31)
Tracker Tracker0@10.56.0.63, sensor 6:
        pos (-0.13,  0.00, -0.00); quat (-0.04, -0.02,  0.01,  1.00)
Tracker Tracker0@10.56.0.63, sensor 7:
```

```
        pos (-0.04, -0.00, -0.00); quat ( 0.00,  0.00,  0.00,  1.00)
Tracker Tracker0@10.56.0.63, sensor 8:
        pos ( 0.15, -0.07, -0.00); quat (-0.31,  0.67, -0.04,  0.68)
Tracker Tracker0@10.56.0.63, sensor 9:
        pos ( 0.14, -0.00, -0.00); quat (-0.08, -0.99, -0.05,  0.07)
Tracker Tracker0@10.56.0.63, sensor 10:
        pos ( 0.14,  0.00,  0.00); quat (-0.01, -0.00, -0.02,  1.00)
Tracker Tracker0@10.56.0.63, sensor 11:
        pos ( 0.05,  0.00, -0.00); quat (-0.00,  0.00,  0.00,  1.00)
Tracker Tracker0@10.56.0.63, sensor 12:
        pos (-0.06, -0.07,  0.01); quat (-0.23,  0.12,  0.02,  0.97)
Tracker Tracker0@10.56.0.63, sensor 13:
        pos ( 0.00, -0.25,  0.00); quat (-0.32,  0.17, -0.56,  0.74)
Tracker Tracker0@10.56.0.63, sensor 14:
        pos ( 0.00, -0.17, -0.00); quat (-0.00,  0.00, -0.00,  1.00)
Tracker Tracker0@10.56.0.63, sensor 15:
        pos (-0.03,  0.07, -0.01); quat (-0.00,  0.00, -0.00,  1.00)
Tracker Tracker0@10.56.0.63, sensor 16:
        pos ( 0.06, -0.04, -0.02); quat (-0.11,  0.16, -0.73,  0.66)
Tracker Tracker0@10.56.0.63, sensor 17:
        pos (-0.00, -0.29,  0.00); quat ( 0.26,  0.24,  0.89,  0.29)
Tracker Tracker0@10.56.0.63, sensor 18:
        pos ( 0.00, -0.23, -0.00); quat ( 0.00,  0.00, -0.00,  1.00)
Tracker Tracker0@10.56.0.63, sensor 19:
        pos (-0.00, -0.06, -0.03); quat ( 0.00,  0.00, -0.00,  1.00)
...
```

En la figura 6.1 se puede ver el proceso de adquisición de datos del paciente, el mismo es capturado mediante el sensor Kinect.

Figura 6.1: Ventana de monitoreo de la captura de Kinect provista por la herramienta FAAST.



## 6.2. Desarrollo del prototipo de escenario mediante Vizard

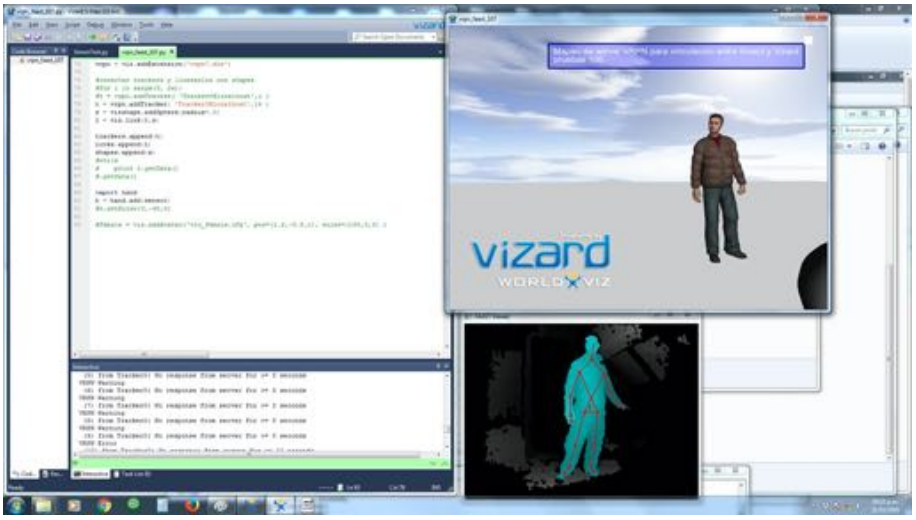
Para evaluar las técnicas de realidad virtual se trabajó inicialmente desarrollando un escenario virtual básico en la herramienta de desarrollo Vizard, la cual permite mediante un lenguaje basado en python realizar el diseño y brindarle interactividad al escenarios 3d y un avatar que representa los movimientos del paciente.

Se desarrolla este prototipo utilizando esta herramienta porque nos permite trabajar nativamente con VRPN y no necesitamos desarrollar una interfaz cliente dll, ni utilizar algún software intermediario entre VRPN y nuestro prototipo, como fue el caso del prototipo completo del cubo móvil presentado anteriormente.

Las pruebas se realizan generando comandos VRPN mediante la utilidad FAAST que actúa como vrpn-server interactuando con el driver Microsoft-Kinect.

Se reciben las coyunturas humanas provenientes de Kinect en formato Tracker vrpn en diferentes canales, cada uno representa las coordenadas de cada parte del cuerpo correspondiente.

Figura 6.2: Escenario de prueba VRPN desarrollado en Vizard




## 6.3. Desarrollo de prototipo entrenador cognitivo mediante Unity

Luego de la prueba de concepto realizada entre Faast generando VRPN desde kinect y Vizard, se decide desarrollar un entrenador específico que nos permita representar una prueba de uso de nuestra BCI en un modo mas real y concreto de acuerdo a las indicaciones del terapeuta de los problemas cognitivos a los cuales esta herramienta pretende servir como entrenador.

Se plantea el desarrollo de un entrenador virtual que refleje los movimientos del paciente, y que permita que éste tome en su mano una Tetera de agua caliente de una mesa A y la deposite en la mesa B.

Las características que se tienen en cuenta para la función de agarre del objeto consiste en acercarse a la zona del mismo y mediante un indicador visual representar

el tiempo que la mano se mantiene en posición de agarre del objeto, cumplido el tiempo correspondiente el objeto queda en forma sostenida a continuación de la muñeca.

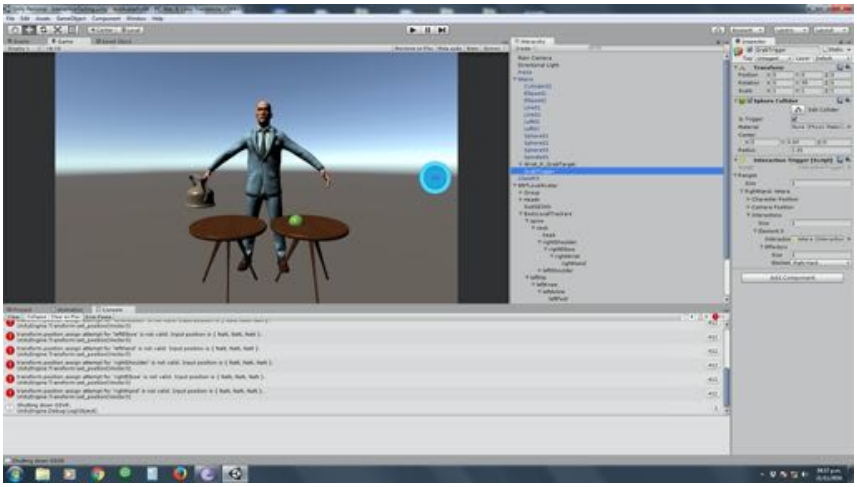
Luego el avatar permite que el usuario realice el movimiento necesario que consiste en llevar el objeto a la mesa B y mediante la misma técnica anterior, mantener el objeto en el destino durante el tiempo necesario, cumplido el mismo, el objeto queda finalmente en el lugar destino.

Esta tarea simple que vemos representada por el avatar en el escenario virtual es una tarea muy compleja que en general no puede llevar a cabo el paciente motora y cognitivamente disminuido.

La adquisición de los datos provenientes de FAAST como vrpn-server, es realizada por un software de VRPN-Cliente OSVR, este software actúa como middleware entre vrpn-server y Unity evitando que tengamos que desarrollar nuestra propia interfaz dll, tal como se realizó en el prototipo del Cubo giratorio.

La figura 6.3nos muestra el escenario virtual terminado.

Figura 6.3: Escenario de prueba VRPN desarrollado en Unity



# Futuro del Proyecto

# Capítulo 7

# Alternativa futura en la adquisición de datos de EEG/EOG y generación de VRPN

## Introducción

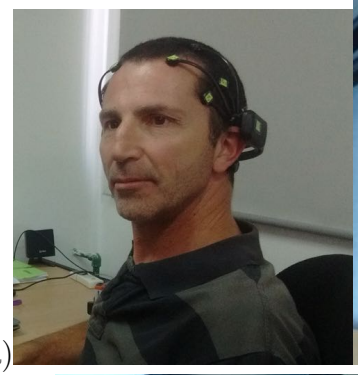
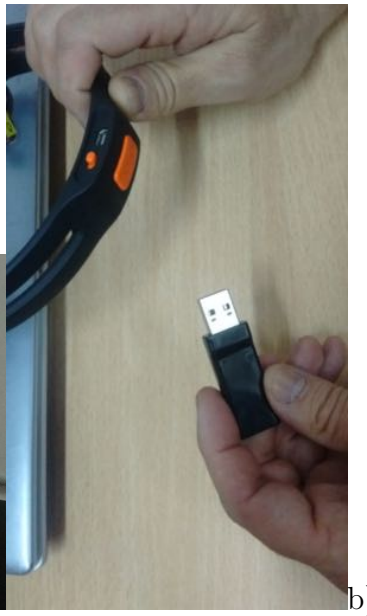
Como posible camino de continuidad de este trabajo, se presenta el cabezal de adquisición de señales de origen comercial EPOC, ya que su uso se está evaluando últimamente en el ámbito de las neurociencias, y si bien el paquete de hardware y software ofrecido, carece de implementación del protocolo VRPN, existe un grupo de desarrolladores dedicado a implementar el acceso a la información que provee este cabezal, basados en lenguaje C y Python de código abierto, se plantea entonces a futuro aprovechar este interesante trabajo en crecimiento, para poder implementar VRPN y de este modo poder utilizar sus señales para incorporarlas en este proyecto.

## 7.1. Alternativa de captura de señales con el cabezal comercial EPOC

Las señales que captura el casquete comercial EPOC de la empresa Emotiv, viajan encriptadas y son desencriptadas en el equipo final mediante el software propietario de la empresa fabricante, luego son procesadas y utilizadas por los sistemas graficadores y utilidades provistas previo pago extra dependiendo de que herramienta se trate. Por lo tanto no sirven para otra cosa que ser utilizado con sus utilidades o comprando alguna otra herramienta de la empresa Emotiv.

Recientemente en el ámbito del software libre a surgido un proyecto de un grupo de programadores/hacker sobre estos dispositivos, que lograron desencriptar la trama

Figura 7.1: Casquete Epoc - Emotiv, a) Casquete colocado b) Modulo de comunicación inalámbrica c) Detalle electrodos secos.

a)   b)

c) 

correspondiente a la info adquirida y transmitida vía bluetooth y ello nos permitiría trabajarla como cualquier cabezal y amplificador convencional a nuestro gusto, bajo ciertas limitaciones.

El proyecto se encuentra basado en python: https://github.com/openyou/emokit y en C: http://www.github.com/openyou/emokit-c

Emokit es un set de programas desarrollados en python, para permitir el acceso al stream de datos crudo/raw desde el cabezal Epoc de la firma Emotiv.

# Capítulo 8

# Conclusiones

Es muy importante destacar en primera medida los problemas encontrados durante el estudio y desarrollo planteado a lo largo de este trabajo.

En primer lugar el mayor inconveniente fue el no contar con un estándar de adquisición de datos de EEG/EOG lo cual hubiera facilitado enormemente conseguir estas señales, no tener que simularlas y hubiera permitido trabajar con datos "reales" provenientes de pacientes neurológicos, por lo tanto los objetivos planteados inicialmente debieron ajustarse y relegar a una continuación del presente trabajo la verdadera captura de estas señales para la implementación final del trabajo.

Otro de los inconvenientes fue que al momento de trabajar con el sensor Kinect y las librerías FAAST de manejo de VRPN no se pudo contar con el código fuente de estas librerías por lo tanto no se pudieron plantear aportes al proyecto como se hubiera querido, lamentablemente forman parte de trabajos que no están disponible como código abierto, esperemos que en un futuro existan otras librerías similares que si lo sean y permitan el trabajo colaborativo, sería conveniente que en caso de necesitar a futuro avanzar con kinect establecer lazos de colaboración mutua con los realizadores del proyecto para poder avanzar en conjunto.

Finalmente podemos decir que otro de los inconvenientes encontrados se debió a que en el área médica neurológica en Mendoza no está muy desarrollada la investigación tecnológica, que no han desarrollado mucho sus áreas de I+D, faltan convenios concretos con entidades educativas y de investigación del medio, y que las empresas médicas por si mismas no destinan inversiones suficientes para lograr avances concretos en estas áreas, por lo tanto se hace evidente la inexistencia de convenios concretos entre entidades universitarias y los centros neurológicos del medio que se dedican a estos estudios y son los primeros beneficiados con los avances de este tipo de trabajos, como el presente.

Luego del estudio de VRPN y de la investigación y las implementaciones prácticas realizadas, podemos concluir en que se trata de una herramienta muy útil y es la

apropiada para las comunicaciones entre aplicaciones a desarrollar y los dispositivos de adquisición de datos de los cuales se pretende recibir información, con el objeto de sentar las bases para trabajos e investigaciones posteriores que se deseen iniciar mediante escenarios de realidad virtual o cualquier tipo de equipamiento asistivo, ya que trabajará a partir de ahora con mayor profundidad tomando como punto de partida la experiencia aquí vertida con este trabajo.

## Contribuciones y aportes

Se considera que el verdadero aporte de este trabajo consiste en la investigación y estudio del protocolo VRPN, además de la implementación realizada del driver de software, que nos permitió la comunicación desde el hardware de microcontroladores hacia el hardware Raspberry donde se ejecuta el corazón VRPN Server, gracias al desarrollo de este driver y su descripción en este trabajo, le permitirá a quien necesite comunicar cualquier hardware desarrollado para adquisición de datos, poder realizar lo mismo y lograr fácilmente también ser acoplado al sistema VRPN y en un corto plazo permitir su implementación en un sistema de representación visual, no obstante vale aclarar, que esta implementación no solo puede ser usada como base de comunicación en entornos de realidad virtual, sino también en la implementación de cualquier otro dispositivo que necesite consumir los flujos de información vía VRPN para su funcionamiento, como pueden ser sillas de ruedas, brazos ortopédicos y cualquier dispositivo que necesite ser comandado.

# Nomenclatura

| | |
|---|---|
| ACV | Accidente Cerebro Vascular |
| BCI | Interfaz Cerebro - Computadora |
| C-EEG | Electroencefalograma Computado o Mapeo Cerebral Computado |
| CPRM | Potenciales Corticales Relacionados con el Movimiento |
| ECoG | electrocorticograma |
| EEG | Electroencefalografía |
| EOG | Electrooculografía |
| ERPs | potenciales relacionados a eventos |
| PPS | potencial postsináptico |
| PPSE | Potencial postsináptico excitador |
| PPSI | Potencial postsináptico inhibidor |
| QEEG | Electroencefalografía Cuantificada |
| RMf | Resonancia Magnética funcional |
| SCP | Potenciales Cortical Lento |
| SMR/RSM | Ritmos sensoromotores. |
| SNC | sistema nervioso central |
| VEP | Visual Evoked Potential o Potenciales Evocados a eventos Visuales |
| VRPN | Virtual Reality Peripherical Network o Red de Periféricos de Realidad Virtual |

# Bibliografía


[1] S. Enriquez-Geppert, R. J. Huster, and C. S. Herrmann, "Boosting brain functions: Improving executive functions with behavioral training, neurostimulation, and neurofeedback," *International Journal of Psychophysiology*, vol. 88, no. 1, pp. 1–16, 2013.

[2] J. Rosenstein, O. E. Marianetti, and R. Otoya, "Uso de vrpn para la implementación de bci partiendo de señales eeg/eog." *Encuentro de Investigadores y Docentes de Ingeniería - ENIDI 2017 - Sesión de Doctorandos y Tesistas*, UNCuyo - UTN - UM, Mendoza, Noviembre 2017.

[3] J. Rosenstein, O. Marianetti, and R. Otoya, "Uso de vrpn en la implementación de una bci para rehabilitación neurológica," *XX Workshop de Investigadores en Ciencias de la Computación - WICC 2018*, vol. 20, no. 1, pp. 776–779, Universidad Nacional del Nordeste - Facultad de Ciencias Exactas, ISBN 978-987-3619-27-4, Abril 2018.

[4] J. Rosenstein, O. E. Marianetti, and R. Otoya, "Diseño y desarrollo de un prototipo de serious game destinado a la rehabilitación de problemas neurológicos implementando vrpn para la comunicación de la bci," *II Congreso Internacional de Ciencias de la Computación y Sistemas de Información - CICCSI 2018*, vol. 1, no. 1, Universidad Champagnat - Universidad Nacional de San Juan, ISBN en prensa, Noviembre 2018.

[5] J. Rosenstein, R. Gonzalez, N. I. Zárate, and M. Campos, "Prototipo de serious game para rehabilitación de problemas neurológicos usando vrpn para comunicación entre cerebro y computadora," *XXI Workshop de Investigadores en Ciencias de la Computación - WICC 2019*, vol. 21, no. 1, Universidad Nacional de San Juan - Facultad de Ciencias Exactas, Físicas y Naturales, ISBN 978-987-3984-85-3, Abril 2019.

[6] J. J. Rosenstein, R. Gonzalez, N. I. Zárate, and M. Campos, "Integración de una bci para rehabilitación de problemas cognitivos en pacientes neurológicos infantiles implementando vrpn como protocolo de comunicaciones," *XXII Workshop*

*de Investigadores en Ciencias de la Computación - WICC 2020*, vol. 21, no. 1, Universidad Nacional de la Patagonia Austral, ISBN en prensa, Mayo 2020.

[7] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, "Brain-computer interfaces for communication and control," *Clinical neurophysiology*, vol. 113, no. 6, pp. 767–791, 2002.

[8] G. Pfurtscheller and C. Neuper, "Motor imagery and direct brain-computer communication," *Proceedings of the IEEE*, vol. 89, no. 7, pp. 1123–1134, 2001.

[9] Y. Wang, X. Gao, B. Hong, and S. Gao, "Practical designs of brain-computer interfaces based on the modulation of eeg rhythms," in *Brain-Computer Interfaces*. Springer, 2009, pp. 137–154.

[10] J. S. Breuer and G. Bente, "Why so serious? on the relation of serious games and learning," *Eludamos. Journal for Computer Game Culture*, vol. 4, no. 1, pp. 7–24, 2010.

[11] B. Thomas, "What's so special about mirror neurons?" 2012. [Online]. Available: https://blogs.scientificamerican.com/guest-blog/whats-so-special-about-mirror-neurons/

[12] A. Varsavsky, I. Mareels, and M. Cook, *Epileptic seizures and the EEG: measurement, models, detection and prediction*. CRC Press, 2016.

[13] J. Webster, *Medical instrumentation: application and design*. John Wiley & Sons, 2009.

[14] C. Del Aguila, *Electromedicina*, 2nd ed. HASA, 1994.

[15] R. Barea Navarro, "Tema 5: Electroencefalografía," *Universidad de Alcalá – Departamento de Electrónica – Instrumentación Biomédica*.

[16] D. P. Bautista, I. A. Badillo, D. De la Rosa Mejía, and A. H. H. Jiménez, "Interfaz humano-computadora basada en señales de electrooculografía para personas con discapacidad motriz," *ReCIBE*, vol. 3, no. 2, 2016.

[17] H. Singh and J. Singh, "A review on electrooculography," *International Journal of Advanced Engineering Technology*, vol. 3, no. 4, pp. 115–122, 2012.

[18] M. Brown, M. Marmor, E. Zrenner, M. Brigell, M. Bach *et al.*, "Iscev standard for clinical electro-oculography (eog) 2006," *Documenta ophthalmologica*, vol. 113, no. 3, pp. 205–212, 2006.

[19] J. A. Pineda, "The functional significance of mu rhythms: translating "seeing" and "hearing" into "doing"," *Brain Research Reviews*, vol. 50, no. 1, pp. 57–68, 2005.

[20] A. Richardson, "Mental practice: a review and discussion part i," *Research Quarterly. American Association for Health, Physical Education and Recreation*, vol. 38, no. 1, pp. 95–107, 1967.

[21] J. D. Kropotov, *Quantitative EEG, event-related potentials and neurotherapy*. Academic Press, 2010.

[22] W. Rief, *Getting started with neurofeedback*. THE BOULEVARD, LANGFORD LANE, KIDLINGTON, OXFORD OX5 1GB, ENGLAND: PERGAMON-ELSEVIER SCIENCE LTD, 2006.

[23] R. Ramirez, M. Palencia-Lefler, S. Giraldo, and Z. Vamvakousis, "Musical neurofeedback for treating depression in elderly people." *Frontiers in neuroscience*, vol. 9, pp. 354–354, 2014.

[24] G. Rizzolatti, L. Fogassi, and V. Gallese, "Neurophysiological mechanisms underlying the understanding and imitation of action," *Nature Reviews Neuroscience*, vol. 2, no. 9, pp. 661–670, 2001.

[25] G. Rizzolatti and L. Craighero, "The mirror-neuron system," *Annu. Rev. Neurosci.*, vol. 27, pp. 169–192, 2004.

[26] M. van Elk, H. T. van Schie, S. Hunnius, C. Vesper, and H. Bekkering, "You'll never crawl alone: neurophysiological evidence for experience-dependent motor resonance in infancy," *Neuroimage*, vol. 43, no. 4, pp. 808–814, 2008.

[27] R. Taylor. (2018, 05) Virtual reality peripheral network (vrpn). [Online]. Available: https://github.com/vrpn/vrpn/wiki

[28] A. Bulling, J. A. Ward, H. Gellersen, and G. Troster, "Eye movement analysis for activity recognition using electrooculography," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 4, pp. 741–753, 2011.

[29] A. C. Gaviria, I. C. Miller, S. O. Medina, and D. R. Gonzales, "Implementación de una interfaz hombre-computador basada en registros eog mediante circuitos de señal mixta psoc," in *V Latin American Congress on Biomedical Engineering CLAIB 2011 May 16-21, 2011, Habana, Cuba*. Springer, 2013, pp. 1194–1197.

[30] V. C. C. Roza, "Interface para tecnologia assistiva baseada em eletrooculografia," 2014.

[31] S. Yathunanthan, L. Chandrasena, A. Umakanthan, V. Vasuki, and S. Munasinghe, "Controlling a wheelchair by use of eog signal," in *2008 4th International Conference on Information and Automation for Sustainability*. IEEE, 2008, pp. 283–288.

[32] E. A. Suma, B. Lange, S. Rizzo, D. Krum, and M. Bolas. (2019, 01) Flexible action and articulated skeleton toolkit (faast). [Online]. Available: https://web.cs.wpi.edu/~gogo/hive/UIVA/

[33] I. N. de Estad. y Censos I.N.D.E.C., *Estudio nacional sobre el perfil de las personas con discapacidad : resultados provisorios 2018. - 1a ed.*, I. Instituto Nacional de Estad. y Censos, Ed. Instituto Nacional de Estad. y Censos, INDEC, 2018, vol. 1.

[34] J. Wang and R. W. Lindeman. (2015, 10) Unity indie vrpn adapter (uiva). [Online]. Available: https://web.cs.wpi.edu/~gogo/hive/UIVA/

[35] R. Otoya, "Bci aplicado a escenarios virtuales de rehabilitación cognitiva y neuronas en espejo." 12 2016, neuromed Argentina S.A. - Neuromed Neurotechnology.

[36] A. Manelli, "Estudio e implementación de osvr y unity3d en sistemas de neurofeedback," 12 2016, neuromed Argentina S.A.

[37] A. Majumder and M. S. Brown, *Practical multi-projector display design*, L. A K Peters, Ed. A K Peters, Ltd., 2007.

[38] M. Billinghurst, H. Kato, and I. Poupyrev, "The magicbook-moving seamlessly between reality and virtuality," *IEEE Computer Graphics and applications*, vol. 21, no. 3, pp. 6–8, 2001.

[39] U. de Los Andes. (2016, 03) Universidad de los andes - facultad de ingeniería de sistemas y computación - ambientes interactivos 3d. [Online]. Available: https://sistemas.uniandes.edu.co/es/

[40] (2017, 03) Psicología y neurociencias. [Online]. Available: http://psicologiayneurocienciaenespanol.blogspot.com.ar

[41] P. Rego, P. M. Moreira, and L. P. Reis, "Serious games for rehabilitation: A survey and a classification towards a taxonomy," in *5th Iberian Conference on Information Systems and Technologies*. IEEE, 2010, pp. 1–6.

[42] H. R. Hartson, *Advances in human-computer interaction*. Intellect Books, 1988, vol. 2.

[43] E. Kandel, J. Schwarz, and T. Jessel, *Principios de Neurociencia*, 4th ed. Mc. Graw- Hill, 2001.

[44] M. E. Avila Perona and P. F. Diez, "Bioinstrumentación ii," *UNSJ*, 2011.

[45] L. C. Nardi and F. Gutiérrez-Barquin, "Implementación de técnicas de modelado inverso para la ubicación del generador eléctrico de descarga en pacientes epilépticos," Thesis, Trabajo Final de Graduación (Bioingeniería). UNSJ, facultad de Ingeniería., Agosto 2011.

[46] A. C. V. Maggio, A. Garcés, L. Orosco, and E. Laciar, "Detección de crisis ictales en registros eeg de pacientes epilépticos usando análisis temporal y frecuencial," *UNSJ*, vol. 1, no. 1, 2010.

[47] V. Gallese and A. Goldman, "Mirror neurons and the simulation theory of mind-reading," *Trends in cognitive sciences*, vol. 2, no. 12, pp. 493–501, 1998.

[48] A. Ehmann. (forthcoming) Messung und invarianz - ein beitrag zum metrologischen strukturenrealismus. [Online]. Available: http://philpapers.org/rec/EHMMUI

# Apéndices

# Código Test de la aplicación dll cliente vrpn

Este es el código en C++ que nos permite chequear tanto el funcionamiento de la dll cliente como también la correcta recepción de los datos vía vrpn.

- Primero están las declaraciones generales, principalmente los datos de variables y estructuras.
- Luego los datos de conexión a nuestro servidor vrpn remoto: "EogDev0@10.56.0.250"
- Dentro del main() chequeamos la posibilidad o no de cargar la dll correspondiente que es quien se comunica con nuestro vrpn_server EogDev0 , así como la existencia de cada una de las funciones que necesitamos utilizar para la obtención de los datos que recibimos vía vrpn cliente.
- Luego se dispara un thread que se encarga de recibir la información y mostrarla en pantalla...

El código completo es el siguiente:

```
// dllmain.cpp : Defines the entry point for the DLL application.
#include <iostream>
#include <windows.h>
#include "EogClientStructs.h"
#define TEST_DATA_COUNT 100
#define DEVICE_NAME "EogDev0@10.56.0.250"
#define PRINT_Eog_DATA
using namespace std;
typedef void (__cdecl * SetupDevice)(const char * deviceName, int yawChan-
nel, int pitchChannel, int rollChannel, int deviceButton_0, int deviceBut-
ton_1, int deviceButton_2, int deviceButton_3, int deviceButton_4, int deviceBut-
ton_5, int deviceButton_6, int deviceButton_7);
typedef void (__cdecl * StartFunc) ( void );
typedef void (__cdecl * StopFunc) ( void );
typedef void (__cdecl * ReadDeviceOrientationFunc)(FAngles * angles);
typedef void (__cdecl * ReadDeviceIsFiringFunc)(bool * fire);
typedef void (__cdecl * ReadDeviceFunc)(DeviceValues * dValues);
bool sStopVrpnThread;
DWORD WINAPI ThreadDevice (LPVOID pArg);
void PressToContinue() {
    cout << "Enter a letter to end!\n";
    cin.clear();
    cin.ignore(INT_MAX,'\n');
    int a;
    cin >> a;
}
int main() {
    //Load the DLL
    cout << "Loading dll!" << endl;
    HINSTANCE DllInstance = LoadLibrary(TEXT("EogClientDeviceDll.dll"));
    if( DllInstance == NULL ) {
        cout << "Error: Unable to load DLL" << endl;
        PressToContinue();
        return -1;
    }
    else {
        cout << "OK: Able to load DLL" << endl;
    }
    //Find the setup function
    SetupDevice pSetupDevice = (SetupDevi-
ce)GetProcAddress(DllInstance, "Eog_SetupDevice");
    if( pSetupDevice == NULL ) {
        cout << "Error: Unable to find SetupDevice function" << endl;
        PressToContinue();
        return -1;
    }
    else {
        cout << "OK: Able to find SetupDevice function" << endl;
    }
    //Find the start function
    StartFunc pStartFunc = (StartFunc)GetProcAddress(DllInstance, "Eog_StartDevice");
    if( pStartFunc == NULL ){
        cout << "Error: Unable to find Start function" << endl;
        PressToContinue();
        return -1;
    }
    else {
        cout << "OK: Able to find Start function" << endl;
    }
    //Find the stop function
    StopFunc pStopFunc = (StopFunc)GetProcAddress(DllInstance, "Eog_StopDevice");
    if( pStopFunc == NULL ){
        cout << "Error: Unable to find Stop function" << endl;
        PressToContinue();
        return -1;
    }
    else {
        cout << "OK: Able to find Stop function" << endl;
    }
```

**Algoritmo 11:** Test de la aplicación dll cliente vrpn (parte 1)

```
    //Find the ReadDeviceOrientationFunc
    ReadDeviceOrientationFunc pReadDeviceOrientation = (ReadDeviceOrientation-
Func)GetProcAddress(DllInstance, "Eog_ReadDeviceOrientation");
    if( pReadDeviceOrientation == NULL ) {
        cout << "Error: Unable to find ReadADeviceOrientation function" << endl;
        PressToContinue();
        return -1;
    }
    else {
        cout << "OK: Able to find ReadDeviceOrientation function" << endl;
    }
    //Find the ReadDeviceIsFiring Function
    ReadDeviceIsFiringFunc pReadDeviceIsFiring = (ReadDeviceIsFiring-
Func)GetProcAddress(DllInstance, "Eog_ReadDeviceIsFiring");
    if( pReadDeviceIsFiring == NULL ) {
        cout << "Error: Unable to find ReadDeviceIsFiring function" << endl;
        PressToContinue();
        return -1;
    }
    else {
        cout << "OK: Able to find ReadDeviceIsFiring function" << endl;
    }
    //Find the ReadDeviceFunc
    ReadDeviceFunc pReadDevice = (ReadDevice-
Func)GetProcAddress(DllInstance, "Eog_ReadDevice");
    if( pReadDevice == NULL ) {
        cout << "Error: Unable to find ReadADevice function" << endl;
        PressToContinue();
        return -1;
    }
    else {
        cout << "OK: Able to find ReadDevice function" << endl;
    }
    //Setup 3 encoders and 8 buttons
    pSetupDevice(DEVICE_NAME, 0, 1, 2, 0, 1, 2, 3, 4, 5, 6, 7);
    sStopVrpnThread = false;
    pStartFunc();
    printf("OK: Will start the thread... \n");
    HANDLE deviceThreadHandle = CreateThread(NULL, 0, &ThreadDevice, (LP-
VOID)pReadDevice, 0, NULL);
    printf("NEW RUN! \n");
    PressToContinue();
    pStopFunc();
    sStopVrpnThread = true;
    if( deviceThreadHandle != NULL ){
        WaitForSingleObject(deviceThreadHandle, INFINITE);
    }
    deviceThreadHandle = NULL;
    pStartFunc();
    printf("\n\n\nEnd! \n");
    PressToContinue();
    pStopFunc();
    FreeLibrary(DllInstance);
}
DWORD WINAPI ThreadDevice (LPVOID pArg) {
    DeviceValues dValues;
    ReadDeviceFunc pReadDevice = (ReadDeviceFunc)pArg;
    while( !sStopVrpnThread ) {
        pReadDevice( &dValues );
        cout << "Angles (yaw, pitch, roll): " << dValues.yaw << ", " << dVa-
lues.pitch << ", " << dValues.roll << " Firing_0: " << dValues.isButtonPress0 << endl;
        Sleep((DWORD)1);
    }
    return 0;
}
```

**Algoritmo 12:** Test de la aplicación dll cliente vrpn (parte 2)

# Apéndice 2 - prototipo de Unity cubo móvil.

## Código C# del script que comanda el comportamiento del cubo mediante la interfaz cliente DLL.

### CubeBehavior.cs

```
using UnityEngine;
using System.Collections;
public class CubeBehavior : MonoBehaviour, IEogDeviceListener
{
    // ------------------------------------------------------------------------------
----------------------
    // Public attributes
    // ------------------------------------------------------------------------------
----------------------
    /// <summary>
    /// Factor used to smooth rotation data
    /// </summary>
    public float PlayerSmoothRotationFactor = 0.1f;
    // ------------------------------------------------------------------------------
----------------------
    // Private attributes
    // ------------------------------------------------------------------------------
----------------------
    /// <summary>
    /// Used to read or not form the dll
    /// </summary>
    private bool bCanReadDll;
    /// <summary>
    /// Device values
    /// </summary>
    private DeviceValues mDeviceValues;
    /// <summary>
    /// Get or set actor's parent transform
    /// </summary>
    public virtual Transform Parent
    {
        get { return mTransform.parent; }
        set { mTransform.parent = value; }
    }
    /// <summary>
    /// Speed up access for transform data
    /// </summary>
    protected Transform mTransform;
    /// <summary>
    /// Get or set object's transform
    /// </summary>
    public virtual Transform Transfrom
    {
        get { return mTransform; }
        set { mTransform = value; }
    }
    /// <summary>
    /// Get or set actor's rotation
    /// </summary>
    public virtual Quaternion Rotation
    {
        get { return mTransform.rotation; }
        set { mTransform.rotation = value; }
    }
// Use this for initialization
void Start () {
        mTransform = gameObject.transform;
        mDeviceValues = new DeviceValues();
        bCanReadDll = false;
        Invoke("StartEog", 1);
}
    private void StartEog()
    {
        Debug.Log("Starting Eog Device....");
        EogDevi-
ce.Eog_SetupDevice("EogDev0@10.56.0.250", 0, 1, 2, 0, 1, 2, 3, 4, 5, 6, 7);
        Debug.Log("Starting Eog device....");
        EogDevice.Eog_StartDevice();
        Debug.Log("Reading Eog device....");
        StartReadFromDevice();
    }
```

**Algoritmo 13:** CubeBehavior.cs (parte 1)

```
// Update is called once per frame
void Update () {
        if (bCanReadDll)
        {
            EogDevice.Eog_ReadDevice(out mDeviceValues);
            mTransform.localRotation = Quaternion.Lerp(mTransform.localRotation,
                                                       Quater-
nion.Euler(mDeviceValues.pitch * 0.36f,
                                                                                  mDeviceVa-
lues.yaw * 0.36f,
                                                                                  mDeviceVa-
lues.roll * 0.36f),
                                                       PlayerSmoothRotationFactor);
if (mDeviceValues.isButtonPress0)
            {
                print("Button 0 pressed");
                mTransform.localPosition = new Vector3(0,0.5f,0);
            }
            else
            {
                print("Button 0 released");
                mTransform.localPosition = new Vector3(0, 0, 0);
            }
        }
}
    /// <summary>
    /// Allow the object to read from the dll
    /// </summary>
    public void StartReadFromDevice()
    {
        bCanReadDll = true;
    }
    /// <summary>
    /// Stop the object to read from the dll
    /// </summary>
    public void StopReadFromDevice()
    {
        bCanReadDll = false;
    }
}
```

**Algoritmo 14:** CubeBehavior.cs (parte 2)