



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Escola Tècnica
Superior d'Enginyeria
Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Interfaz Mental Para El Control de Aplicaciones

Proyecto Final de Carrera

Ingeniería Informática

Autora: Rocío Alegre Marzo

Directores: Francisco José Abad Cerdá & Beatriz Rey Solaz

Valencia 30 de Septiembre de 2013

Dedico este trabajo a mis compañeros y compañeras de los laboratorios de investigación del *Ai2* y del *LabHuman*, por su enseñanza, junto a los profesores de esta Universidad y sobre todo a mi familia y amigos que me han apoyado en todo momento.

Resumen

Este Proyecto Final de Carrera presenta una aplicación que utiliza un dispositivo de Electroencefalografía para realizar estudios de tareas cognitivas. Se estudia el caso concreto de mejora de una aplicación existente creada por el laboratorio *LabHuman* de la *Universidad Politécnica de Valencia*. La aplicación ha sido desarrollada para trabajar con el dispositivo *EEG Neuroheadset* [1] de la marca *Emotiv* (ver figura 3 dispositivo de la derecha). Dicho dispositivo de medición de ondas de Electroencefalografía (EEG) es más ligero y sencillo que los que se usan normalmente en Medicina. Normalmente este tipo de dispositivos tienen muchos más sensores y su diseño es distinto, requiriendo máquinas complejas dedicadas a la captura y amplificación de las señales, lo cual reduce la movilidad del sujeto de estudio y hace difícil su transporte. El *EEG Neuroheadset* pretende resolver estos problemas y la aplicación anterior formaba parte de un estudio para validar dicho dispositivo. La aplicación creada durante este PFC añade la automatización de algunos procesos, como la introducción automática de marcadores, para que sea más fácil el estudio posterior de los resultados obtenidos. Se busca también mejorar la comodidad del sujeto durante la prueba, la facilidad de uso y la portabilidad del sistema.

Como se parte de un proyecto anterior, parte de los requisitos del sistema vienen impuestos por dicho proyecto. Se parte de una aplicación desarrollada en C# con *Microsoft Visual Studio 2010*, a la que se le realizan una serie de cambios para su mejora y para hacerla compatible con la nueva aplicación. La nueva aplicación la programación se ha realizado con *Microsoft Visual Studio 2008* debido a limitaciones de la SDK del casco, y se continua usando el lenguaje C# y *multithreading*.

La nueva aplicación consta de una Interfaz Gráfica de Usuario (IGU), que funcionará en un PC distinto al de la aplicación existente. A la nueva aplicación se conectará el casco de captación de señal EEG, realizando la captura de los datos. Se muestrearán la señal del sujeto con los sensores del casco y se añadirán una serie de funciones para mejorar su uso, el procesado de los datos y la introducción de marcadores automática cuando la situación lo requiera (decidido por los investigadores).

Palabras clave: Electroencefalograma, Electroencefalógrafo, EEG, C#, sockets, IGU, multithreading, Keystrokes, SDK, LabHuman, Ai2, Emotiv neuroheadset, Emotiv Insight.

Tabla de contenidos

1.	Introducción	9
2.	Contexto del proyecto	13
3.	Antecedentes	16
4.	Análisis	23
5.	Diseño	29
6.	Resultado	39
7.	Conclusiones	41
8.	Trabajo futuro.....	43
9.	Bibliografía	45
12.	ANEXOS	47
13.	Manual de Usuario de la Aplicación	47



1. Introducción

La tecnología avanza constantemente para incorporar mejoras a la vida cotidiana y para optimizar las técnicas existentes. En el área de la Medicina, estos avances ayudan a mejorar la experiencia de los pacientes, a disminuir la mortalidad y a reducir los tiempos de ejecución de tareas como es el caso de las operaciones, entre otros muchos ejemplos.

En éste proyecto, se parte de una aplicación para el estudio con Electroencefalogramas (a partir de ahora EEG) de usuarios durante la realización de tareas de distintos tipos. Dicha aplicación utiliza el casco *Emotiv* [1], y ha sido llevada a cabo por el *I3BH (Instituto Interuniversitario de Investigación en Bioingeniería y Tecnología Orientada al Ser Humano)* concretamente, por el grupo de trabajo *LabHuman*, Línea de Mente, Bienestar y Tecnología de la Universidad Politécnica de Valencia. La aplicación se basa en el trabajo de doctorado “Validación de un sistema de EEG inalámbrico como medida neurofisiológica de ondas de actividad cerebral en sujetos humanos.” de Alejandro Rodríguez [2].

En dicha aplicación un operador introduce algunas marcas asociadas a determinados eventos durante la ejecución de cierta tarea. Dichas marcas permiten determinar el estado de las ondas EEG del sujeto en cada experimento. Debido a la diferencia entre el momento en que empezaba una tarea y el momento en el que el investigador presionaba el botón, se introducía un pequeño retardo artificial. Aunque para el estudio planteado el retardo era asumible, pues se analizaba globalmente la respuesta en un periodo de tiempo de varios minutos, en otros tipos de análisis de EEG se analizan respuestas a determinados eventos en el rango de tiempos de milisegundos, con lo que no sería factible usar esta metodología para introducir las marcas. Es por este motivo que se plantea el presente proyecto.

Los objetivos a alcanzar durante el desarrollo de éste Proyecto Final de Carrera son:

- Automatizar la inserción de las marcas a la señal EEG, lo que permite reducir errores y retrasos.
- Añadir correctamente los marcadores de la aplicación al fichero de datos (en tiempo y significado – ej. qué tarea se está realizando)
- Minimizar la latencia de envío de los datos, ya que existe una comunicación entre dos aplicaciones que podrían estar ejecutándose en máquinas distintas.
- Disminuir el número de errores producidos por la interacción con la introducción de las marcas manuales.
- Disponer de control sobre la ejecución de la aplicación del cliente. Anteriormente la ejecución debía completarse, no se podía parar las tareas a mitad y el investigador no tenía control sobre la ejecución de la misma.
- Abstracter el HW de modo que se pueda usar el casco sin hacer uso de las herramientas que proporciona el fabricante, sino usando las librerías internas y adecuando el uso a lo necesario, obteniendo portabilidad y una mejor adaptación a los objetivos requeridos.

A grandes rasgos la arquitectura del sistema es la siguiente:

- Una aplicación cliente basada en una refactorización de la aplicación existente, a la que se le ha añadido la siguiente funcionalidad:
 - o Parar las tareas, manteniendo el fichero de datos de almacenamiento de la información y retorno al inicio para proseguir con la tarea deseada.
 - o Comunicación con la aplicación servidor.
- Una aplicación servidor, que se encarga de:
 - o La comunicación directa con el casco mediante las librerías, sin necesidad de hacer uso de las herramientas del fabricante y dotando así de mayor libertad a la programación de las funcionalidades requeridas en cada momento.
 - o Almacenamiento de los datos obtenidos del casco en un fichero de datos individualizado para cada sesión de estudio.
 - o Inserción automática de las marcas (por tarea, imagen, error, paradas , etc), para facilitar el estudio posterior de las señales.
 - o Muestreo gráficamente en pantalla la calidad de la señal de los sensores del casco y del resto de elementos (tales como nivel de batería del casco, calidad de la conexión cliente-servidor y conexión servidor-casco, etc.). Así se optimiza el uso del casco ya que sabemos si la carga de la batería es buena, además de otro tipos de fallos que harían repetir la prueba.
 - o Fichero de log, para almacenar la información de la ejecución del programa, tanto para verificar el correcto funcionamiento como guía para ayudar a la depuración en caso de fallo.
- Un casco *EEG Neuroheadset* de la marca Emotiv, que se comunica con la aplicación servidor, mediante una conexión Bluetooth y se encarga de captar las ondas EEG del sujeto, almacenarlas en el buffer propio y transmitir las al PC junto con las marcas añadidas, que posteriormente quedará reflejado en el fichero de datos con extensión .csv

La siguiente figura muestra la configuración del sistema:

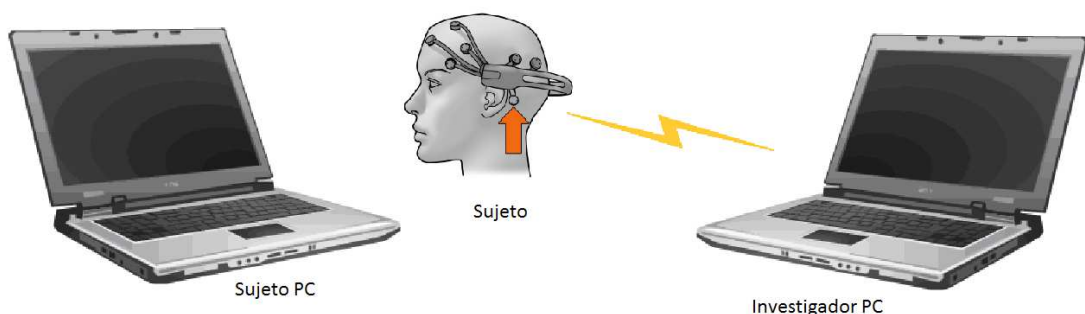


Figura 1. Montaje para la realización del experimento
(Imágenes aisladas obtenidas del manual SDK)

El resto de la memoria del Proyecto Final de Carrera está organizada como se expone a continuación:

El capítulo “Contexto del Proyecto”, presenta las principales características de la aplicación en la que se basa este proyecto.

En el capítulo “Antecedentes” se exponen algunos conceptos y términos de Electroencefalografía, que ayudarán a comprender mejor el Proyecto. Se explica qué es la EEG, las aplicaciones médicas de éste tipo de ondas, los diferentes tipos de dispositivos de medición y por último una justificación de la selección del hardware utilizado.

En los capítulos “Análisis” y “Diseño” como su nombre indica, expone el proceso de definición de requisitos de la aplicación, y el diseño de alto nivel de la solución que finalmente se ha construido.

Por último, “Resultado” y “Conclusiones” invitan a pensar en las bondades o desventajas de haber usado éste dispositivo y para terminar en “Trabajo Futuro”, una posible continuación con un dispositivo de la misma marca, que acaba de salir al mercado.

2. Contexto del proyecto

Partiendo del trabajo “Validación de un sistema de EEG inalámbrico, como medida neurofisiológica de ondas de actividad cerebral en sujetos humanos.” realizado por Alejandro Rodríguez. Se crea una aplicación de monitorización de sujetos mediante las respuestas EEG a ocho tareas diferentes; cuatro de las cuales se correspondía con tareas cognitivas y las otras cuatro eran situaciones de reposo. Entre las distintas tareas, el sujeto miraba imágenes neutras. Dichas imágenes, denominadas *LineaBase*, se intercalaban entre las tareas cognitivas teniendo así, monitoreo de toda la sesión.

La duración de cada tarea fue de un minuto, por los cuales el sujeto tenía que pasar de forma consecutiva y sin pausa. En total la prueba duraba aproximadamente unos 15 minutos, contando con los 8 que duraba el estudio y el resto era el tiempo dedicado a colocar el dispositivo en la cabeza y preparar los programas que se utilizarían.

La aplicación anterior, utilizada para este estudio fue realizada mediante el programa Visual Studio 2010, utilizando como lenguaje de programación el C#. El objetivo de dicha aplicación era poder reproducir, con una sencilla aplicación, los 8 momentos de forma secuencial y en una misma duración sin que el experimentador tuviera que realizar ninguna labor salvo controlar que los registros de las señales fueran correctamente.

Previamente al inicio del estudio, el sujeto fue colocado delante de una pantalla de retroproyección de gran tamaño. Lo siguiente fue colocarle el dispositivo EEG y arrancar la aplicación del estudio.

Tarea 1: “Tarea de Relajación”. Cerrando los ojos, deberían intentar relajarse llevando a cabo una respiración pausada e intentando no pensar en nada. Para incentivar aun más el efecto del relax se añadió a esta etapa sonidos de la naturaleza.

Tarea 2: “Tarea de Atención”. Forzando a los participantes a mantener un nivel de atención elevado, se les realizaba el test de *Stroop*.

Tarea 3: “Tarea de Concentración”. Completar el máximo número de casillas de un sudoku durante 1 minuto.

Tarea 4: “Tarea de Frustración”. Buscando frustrar al sujeto mediante un ejercicio que ya había realizado anteriormente, se realiza el test de *Stroop*. Existía una probabilidad de un 70% de que el sistema se lo marcara como erróneo y por lo tanto reproduciese un ruido estridente.

En la siguiente figura se puede ver una pequeña muestra con el formato de las ventanas de las tareas principales, a falta del conjunto de imágenes de transición entre los ejercicios.

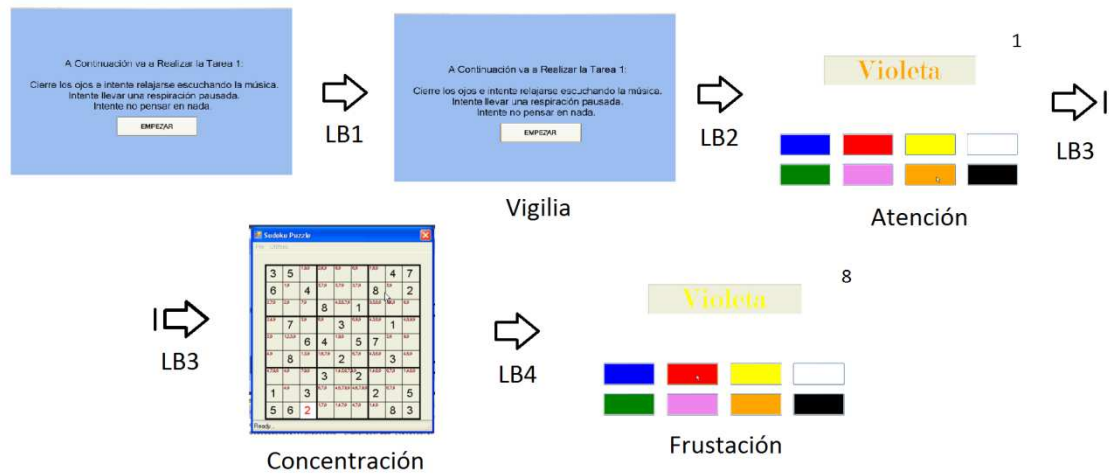


Figura 2. Resumen gráfico del estudio *EEG y Tareas Cognitivas*

La aplicación se realizó para validar el casco Emotiv durante la realización de tareas cognitivas básicas. Se esperaba evaluar si las activaciones observadas con el Emotiv coincidían con las observadas en la literatura en estudios previos. Por tanto, no va dirigida a pacientes, sino a la población en general.

3. Antecedentes

3.1 ¿Qué es un EEG?

Son las siglas para designar la palabra Electroencefalograma. Se trata de un estudio de la función cerebral que recoge la actividad eléctrica del cerebro, en situación basal y con métodos de activación como la hiperventilación y la fotoestimulación.

El estudio de señales EEG se llevan realizando desde 1875 con animales y 1920 con humanos aproximadamente [3]. Desde entonces varios han sido los métodos para captar numéricamente de los valores de éstas señales. Los métodos iniciales podían ser muy invasivos para los pacientes y con el paso de los años, se ha buscado minimizar esas molestias tanto por peso, diseño, molestias dolorosas para el paciente y portabilidad.

La señal eléctrica recogida se amplifica y representa en forma de líneas, interpretándose la actividad de las distintas áreas cerebrales a lo largo del tiempo. Hay varios rangos de frecuencia de interés:

- Ritmos alfa: 8-13 Hz.

Surgen de la actividad eléctrica sincrónica y coherente de las células cerebrales de la zona del tálamo. Se originan sobre todo en el lóbulo occipital durante periodos de relajación, con los ojos cerrados, pero todavía despierto.

Sus efectos característicos son: relajación agradable, pensamientos tranquilos y despreocupados, optimismo y un sentimiento de integración de cuerpo y mente. Se piensa que representan la actividad de la corteza visual en un estado de reposo.

- Ritmos beta: 14-60 Hz.

Se registran cuando la persona se encuentra despierta y en plena actividad mental. Los sentidos se hallan volcados hacia el exterior, de manera que la irritación, inquietud y temores repentinos pueden acompañar este estado.

- Ritmos delta 0-4 Hz.

Surgen principalmente en el sueño profundo y muy raras veces se pueden experimentar estando despierto. Sus estados psíquicos correspondientes son el dormir sin sueños, el trance y la hipnosis profunda.

- Actividad theta: 4-7 Hz.

Se producen durante el sueño (o en meditación profunda, entrenamiento autógeno, yoga...), mientras actúan las formaciones del subconsciente. Las características de éste estado son: memoria plástica, mayor capacidad de aprendizaje, fantasía, imaginación e inspiración creativa.

Existen patrones normales, y patrones anormales que hacen sospechar lesiones o enfermedades características.

A continuación se exponen algunos tipos de trastornos que se pueden detectar haciendo uso de EEG [4]:

- Convulsiones
- Alzheimer
- Confusión
- Traumatismos craneales
- Infecciones
- Tumores
- Trastornos del sueño
- Períodos de pérdida del conocimiento
- Monitorización mientras se realiza cirugía cerebral
- Coma profundo
- Hemorragias internas
- Problemas de déficit de la atención/hiperactividad (TDAH)
- Infartos cerebrales
- Trastornos del espectro autista (TEA)
- Problemas psiquiátricos
- Trastornos alimenticios

NOTA: NO sirve para medir la inteligencia.

Las neuronas del cerebro se comunican entre sí produciendo pequeñas señales eléctricas, llamadas impulsos.

Un EEG ayuda a medir esta actividad. El examen lo realiza un técnico especialista en electroencefalografías en un consultorio médico, en un hospital o en un laboratorio.

Al sujeto se le pide acostarse boca arriba sobre una cama o en una silla reclinable. Le colocan discos metálicos planos, llamados electrodos, en el cuero cabelludo, los cuales se sostienen en su lugar con una pasta conductora. Los electrodos van conectados por medio de cables a un amplificador y a una grabadora que convierte las señales eléctricas en patrones que se pueden observar en una computadora como un montón de líneas ondeadas. Es de una magnitud de microvoltios.

Es necesario permanecer inmóvil y con los ojos cerrados durante el examen, debido a que el movimiento puede cambiar los resultados. Sin embargo, es posible que se le solicite hacer ciertas cosas durante el examen, como respirar profunda y rápidamente durante algunos minutos o mirar hacia una luz muy brillante y centellante.

3.2 Métodos de medición

Normalmente y para estudios donde se requieren unos resultados rigurosos, se utiliza el método nombrado anteriormente. Debido a su poca portabilidad y a que puede resultar molesto para el sujeto, se están estudiando nuevos sistemas que utilizan otro HW para la medición, aunque hay que recalcar que de momento pueden no ajustar óptimamente la señal captada a la real, por lo que pueden introducir fallos por no ser tan precisos y no se usan en casos propiamente de medicina, sino para investigar, ocio, y otro tipo de estudios.

Algunos modelos son:

- Emotiv
 - o Usado en éste proyecto y del cual hablaremos posteriormente.
- NeuroSky [5]
 - o Utiliza un dispositivo HW muy simple con el que capta las señales y cuyo reclamo es su simpleza y su ligereza. Se usa principalmente en el sector del entretenimiento y deportes, pero también en la investigación para validar su uso. Incluso tienen en venta un chip para poderlo incorporar en cualquier dispositivo. La frecuencia de muestreo indica que es de 512 Hz.
- OCZ Neural Impulse Actuator [6]
 - o Realmente NO se trata de un lector de EEG, utiliza los movimientos musculares que se realizan facialmente, haciéndolos corresponder con determinadas actuaciones. Se utiliza principalmente en el área del entretenimiento. [Video explicativo de uso : http://www.youtube.com/watch?v=FWo12rhz_TQ]

3.3 Emotiv & EEG Neuroheadset

El dispositivo en concreto que se usa en éste proyecto es el *EEG neuroheadset* de Emotiv, que han desarrollado nuevo HW para facilitar el uso y el confort de dichos dispositivos.

En la siguiente figura se puede apreciar a simple vista la diferencia en cuanto a tamaño, menor número de cableado, mayor libertad de movimiento y diseño estético.



Figura 3. Dispositivos de medida de señales EEG.

Aunque también se aprecia la diferencia de sensores de captación de un método al otro. Es por ello que deben de realizar estudios de validación del dispositivo para ver que los valores obtenidos no estriban mucho de lo que se espera obtener, o de los resultados obtenidos con los sistemas usados hasta el momento.

Según el uso y la exactitud que se requiera en la precisión de los valores muestreados, será mejor usar el método de siempre o las nuevas alternativas.

A raíz de una serie de charlas realizadas en el grupo de trabajo del Instituto Ai2 de la Universidad Politécnica de Valencia [7] sobre éste dispositivo HW y debido a los estudios en EEG del laboratorio LabHuman con éste dispositivo; se profundizó en su

estudio para comprobar si los resultados obtenidos, son lo suficientemente exactos o aproximados a lo necesario para poder concluir que el dispositivo es correcto, ya que dicho HW podría no ser todo lo preciso que se necesita.

El dispositivo HW, en éste caso un casco, consta de 16 sensores de captación de señal, 2 de los cuales están dedicados a una toma de tierra GND. Tiene una frecuencia de captación de señal de 128Hz aprox. y se comunica con el PC mediante una conexión Bluetooth. También tiene una serie de herramientas de control, que facilitan el uso automático del dispositivo si así se requiere, con un entrenador de movimientos, un display de las señales captadas, un traductor de señales a movimientos (por ejemplo para ser usado en juegos).

Algunas de las herramientas con las que cuenta el paquete de instalación son:

- Emotiv EPOC Control Panel: que es una IGU que contiene una serie de pestañas con aplicaciones para familiarizarse con el casco. Además tiene también herramientas de Expresividad, Afecto, Cognitivas y de simulación con el ratón. Todas ellas de fácil uso una vez se conecta el casco y se reconoce por la herramienta.

A continuación una captura (sin casco conectado):

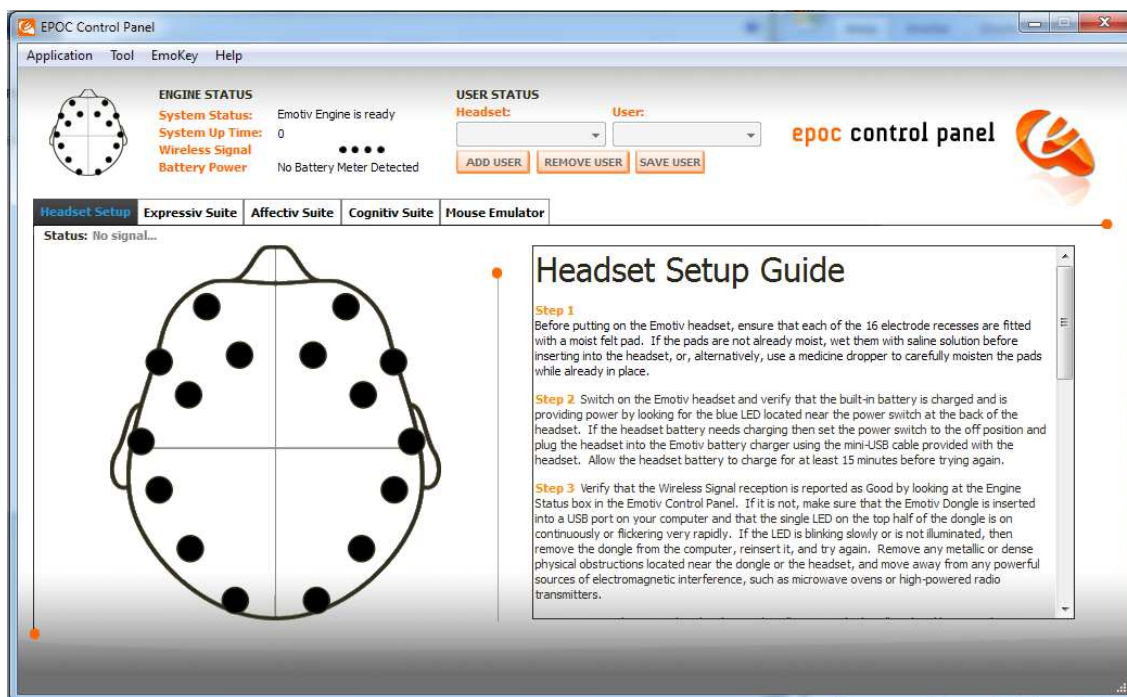


Figura 4. Emotiv EPOC Control Panel ©

Por tratarse de la *versión desarrollador*, en éste caso se cuenta también con un conjunto de librerías y un SDK ampliado, que permite acceder a las funciones haciendo uso de las herramientas, dando mayor flexibilidad y adaptación a las necesidades del proyecto en cuestión. Por ejemplo un panel de Control mejorado para poder ver las señales captadas por el casco. Del mismo modo también se cuenta con una serie de ejemplos de desarrollo iniciales que servirán para familiarizarse con las librerías y los métodos proporcionados a la hora de programar nuevas aplicaciones.

Además se dispone de otras herramientas no usadas en este proyecto pero de gran utilidad, como son:

- EmoComposer: Para generar, de modo Interactivo o mediante Scripts, señales o movimientos en los ejemplos de robot faciales que se proporciona.
- EmoKey: Para hacer uso o generar *Keystrokes*, que posteriormente su podría usar en juegos u otro tipo de dispositivo, como inputs programados sin necesidad de realizar dicha acción y solo usando el casco (requiere un entrenamiento previo de reconocimiento de movimientos/pensamientos).
- TestBench: el más similar a la extensión realizada, ya que en el laboratorio se precisaba de dicha herramienta, pero cambiando algunos aspectos e introduciendo reformas como la de introducción de marcadores automáticos y facilitar el uso de los archivos generados y su formato.

3.4 Estudiando los resultados

Dependiendo del tipo de estudio que se vaya a realizar, nos interesará fijarnos en un bloque de señales o en otras, o en unos rangos determinados, pero en todos los casos deberemos de tratar las señales obtenidas y estudiarlas numéricamente. Para ello el programa que utiliza el grupo del LabHuman es el proporcionado por el *EEG Lab* [8]. Tiene también una wiki donde poder ver toda la información y descargar los archivos para el estudio de los resultados.

Para ver los archivos generados por la herramienta aportada por Emotiv, tienen extensión .edf y se hace uso de la versión utilizada 7.1.3 de EEG Lab.

En el caso de los nuevos archivos generados, la extensión de dichos archivos generados son .csv, por lo que pueden verse directamente como una hoja Excel (abriéndolo como *Datos >> Obtener datos externos >> Desde texto >> Delimitados >> Siguiente con separador punto y coma*) y de ese modo analizar mas cómodamente los datos a simple vista.

Con EEG Lab, se hace uso del programa Matlab y podemos observar los datos numéricamente, con diagramas y dibujos y hacer mediciones o ver las señales tal cual.

3.5 Utilidades prácticas

Actualmente se están haciendo estudios y actualizando los principios del marketing. Para ello herramientas como éstas que ayudan a ver las reacciones emocionales de los sujetos, son de gran utilidad. Saber hacia dónde mira un usuario en una película [9] o cómo reacciona al ver un anuncio/programa [10] pueden dar una idea a la marca, de qué camino seguir en su enfoque de marketing, lo que ahora se llama *neuromarketing* y cómo aumentar sus beneficios y la fidelidad de sus usuarios. Temas que para algunas compañías son de gran importancia y que según el modo de gestionarlo, pueden hacer que marquen la diferencia con respecto a las empresas de la competencia.

Hay más campos en los que pueden tener una gran aplicación. Es el caso de los videojuegos y el entretenimiento, poder manejar a los personajes de nuevas maneras o usar estas tecnologías para nuevos modelos de juego como el yoga, la relajación etc. abren nuevas posibilidades de entretenimiento y diversión. Si además se unen a dispositivos ya existentes como la Kinect [11] seguro que aumentará las expectativas. Es posible ya, la integración con motores de juego como Unity3D, con el que se han creado juegos como "*Spirit Mountain*" (disponible en versión demo en la página de Emotiv, junto una serie de herramientas <http://emotiv.com/store/> >> APP Store).

El campo de la medicina también se puede aprovechar de estos nuevos dispositivos. Seguramente aún no sean todo lo adecuados ya que la precisión en la señal y la exactitud de los datos aún no es todo lo exacta que se pueda esperar para poder usarlo por ejemplo en una operación, pero sí que pueden ser adecuados estos dispositivos por ejemplo, para proyectos del estilo de *Optimi* [12], en el que lo que prima es la portabilidad, facilidad de uso y diseño del dispositivo, para que los sujetos puedan llevar consigo el dispositivo a lo largo de sus quehaceres cotidianos y sean menos invasivos con los pacientes.

Y en la investigación, para seguir desarrollando HW y SW que valide los métodos existentes e innove implementando nuevos sistemas que mejoren los actuales.

4. Análisis

Este proyecto parte de un proyecto base ya creado, que se ha querido automatizar como ya se explica en los apartados de “*Introducción*” y “*Contexto del proyecto*” y que ejecuta por un lado una serie de tareas que un sujeto deberá llevar a cabo y por otro una aplicación de monitoreo de los sensores del casco y de captación de datos obtenidos del mismo.

El sistema con el que se contaba era:

- 2 PCs:
 - o 1 gestiona el ejercicio y los *clicks* de ratón del cliente.
 - o 1 gestiona el casco y las órdenes del investigador.
- 1 sujeto:
 - o Ejecuta los ejercicios, haciendo *clicks* de ratón.
 - o Porta el casco Emotiv.
- 1 investigador:
 - o Gestiona el programa de ejercicios.
 - o Gestiona el programa de captura de datos de Emotiv.
 - o Introduce marcadores en los datos manualmente para estudios posteriores.
- 1 Casco Emotiv:
 - o Recoge la señal EEG del sujeto.
 - o Manda por Bluetooth los datos al ordenador del servidor.
- Uso de la herramienta TestBench de Emotiv:
 - o Aporta el archivo con formato.edf.
 - o Realiza la conversión de formato si es necesario a .csv.
 - o Visualización de las señales obtenidas de manera gráfica.
- Uso del sistema EEG Lab para el estudio de los resultados.

Y lo que se pretende conseguir es:

Partiendo y aprovechando el sistema que ya existe y la aplicación ya creada:

- Poder grabar los datos del casco, **mediante la SDK**, sin hacer uso de las Herramientas que Emotiv proporciona.
- Introducir, **automáticamente** los marcadores de cada uno de los ejercicios, para de éste modo facilitar el estudio posterior (tanto los realizados por el sujeto, como los generados por el investigador).
- Disponer el investigador de una **Interfaz de Usuario**, en la que poder:
 - o Ver el correcto funcionamiento de los nodos del casco Emotiv.
 - o Poder introducir marcadores extra, manualmente.
 - o Poder parar la aplicación, sin perder los datos de la sesión.
- Recoger los **datos** procedentes del casco para su estudio posterior.

Para poder llevarlo a cabo, se toman algunas decisiones previas para intentar sacar el mayor provecho posible de la aplicación.

- La herramienta proporcionada por Emotiv, se cambiará por una Interfaz Gráfica de Usuario (a partir de ahora llamada IGU) que se ajustará mejor a las características requeridas.

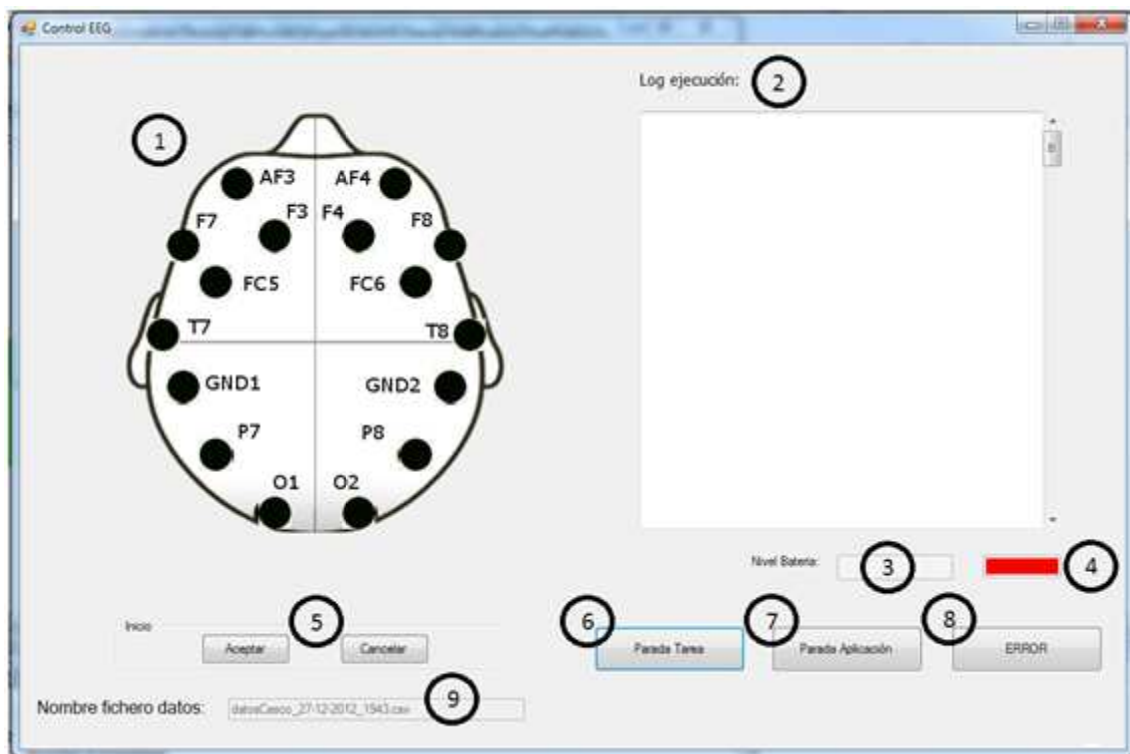


Figura 5. IGU de la aplicación del servidor

En la figura anterior podemos ver la IGU del servidor que se ha desarrollado. En la parte izquierda se dibuja la calidad de cada uno de los sensores, en la superior derecha una salida de texto que muestra el log de lo que va ocurriendo, válido tanto para la depuración de la programación, como para el seguimiento por parte del investigador del correcto transcurso del proceso. Bajo él, un indicativo de batería del casco y un display de color, que será verde si todo funciona bien y muestreará en rojo si ocurre algún error, ya sea de baterías, de sensores, de conexiones, etc. todos ellos a su vez mandarán un marcador de error a las señales, para tenerlo en cuenta a la hora de usar los datos posteriormente. El conjunto de botones (5, 6, 7 y 8) aportan funcionalidad de distintos tipos a la aplicación, como son el comienzo de la aplicación, parada de la tarea actual, parada de la aplicación y marcador de error manual (mejor explicados en el capítulo posterior de “Manual de Usuario”). Por último en la parte inferior izquierda, se ve el nombre del fichero de la sesión actual, que sigue la norma “**datosCasco_fecha_hora.csv**”.

La IGU del cliente se ha mantenido igual exteriormente. Los cambios realizados son de desarrollo y se comentarán posteriormente en el capítulo de “Diseño”.

- La decisión de mantener los 2 PCs o la posibilidad de ejecutar todo en un mismo PC se estuvo estudiando, pero al final se optó por continuar con el esquema de 2PCs, debido a que el *input* de datos por los dos tipos de usuarios podía coincidir y podría

llevar a fallos o pérdida de datos, también hay que tener en cuenta la sobrecarga del sistema por casos de concurrencia en la ejecución al hacer uso del *multithreading* (computación en paralelo de un conjunto de tareas).

- El tipo de estructura de red que se iba a usar también era algo a tener en cuenta. Se requería una latencia de envío de datos de un PC a otro lo mínimo posible, tanto por el envío en sí, como por el almacenamiento disponible en el buffer del casco, que no debía excederse para no perder información. La conexión se realiza por cableado de red Ethernet y poniendo los dos PCs en la misma subred y los envíos de paquetes se hacen con una conexión TCP haciendo uso de sockets.

La tabla de especificaciones del casco Emotiv es la siguiente:

Specifications	
Number of channels	14 (plus CMS/DRL references)
Channel names (Int. 10-20 locations)	AF3, AF4, F3, F4, F7, F8, FC5, FC6, P3 (CMS), P4 (DRL), P7, P8, T7, T8, O1, O2
Sampling method	Sequential sampling, Single ADC
Sampling rate	~128Hz (2048Hz internal)
Resolution	16 bits (14 bits effective) 1 LSB = 0.51µV
Bandwidth	0.2 - 45Hz, digital notch filters at 50Hz and 60Hz
Dynamic range (input referred)	256mVpp
Coupling mode	AC coupled
Connectivity	Proprietary wireless, 2.4GHz band
Battery type	Li-poly
Battery life (typical)	12 hrs
Impedance measurement	Contact quality using patented system

Figura 6. Tabla de especificaciones aportada por Emotiv

Y el desarrollo de las aplicaciones se ha realizado sobre un SO Windows XP (servidor), programando sobre Visual Studio 2008 y Windows 7 (cliente), programando sobre Visual Studio 2010, en ambos casos el lenguaje elegido es C#; junto con las librerías (*DotNetEmotivSDK.dll*, *edk.dll*, *edk_utils.dll*) aportadas en el SDK del set Emotiv.

Otros aspectos a tener en cuenta:

- Existe la posibilidad de que dos PCs puedan mandar información al fichero de datos “simultáneamente”, de todos modos se soluciona canalizando la escritura por la función de la librería (`engine.DataSetMarker((uint)userID, marcador)`) de modo que evita el problema y genera una sola escritura a la vez.

- Se adecua la aplicación del cliente existente previamente, para la comunicación con la nueva aplicación. Para ello se hace uso de sockets TCP que envían y reciben la información necesaria. Se creará también una superclase de la que heredarán las ya generadas para poder aportar nuevos métodos y variables globales de modo que no se duplique el código.
- Debe poderse gestionar la aplicación del cliente desde la aplicación del servidor. Con la nueva implementación la aplicación del cliente puede pararse, cerrarse y reiniciarse siempre que el investigador lo estime oportuno, sin necesidad de cerrar la sesión actual del cliente.
- El buffer de captación de datos del casco es limitado, por lo que una mala gestión de los tiempos de recogida de datos del casco, puede llevar a pérdidas de información. Se decide optar por ciclos de 15 ms (51,56% de su capacidad) de modo que sea lo suficiente para que el buffer no se llene y la escritura sea rápida y constante, comprobando que no hay pérdida de paquetes.
- Se trabaja con la SDK del casco para manipular automáticamente los datos del casco y las herramientas aportadas por Emotiv. Obliga a tener en cuenta la inclusión de las dlls oportunas en los proyectos y a gestionar bien las funciones aportadas. La SDK tiene un fichero de ayuda para familiarizarse con ellas incluido en la documentación (del SDK developer).
- Se guardará, como opción extra, un fichero *log* con todos los sucesos ocurridos para la posterior depuración de errores o despistes con los sujetos, de modo que si ocurre alguna anomalía en los datos se sepa si se ha producido por temas de fallos en los dispositivos, fallos de conexión u otro tipo o si son debidos por anomalías en las señales de los propios sujetos.

Resumiendo, la opción que se ha seguido para la implementación de la solución ha sido:

- Generar una aplicación, externa a la ya existente, para automatizar el proceso de trabajo con el casco y de marcadores para el post-procesamiento de la señal.
 - o Para el trabajo con el casco, se han usado las librerías de la SDK de modo que se accede a los datos continuamente guardándolos en el buffer del casco hasta que se lleva a cabo su escritura en un fichero, teniendo en cuenta los tiempos de captura de los datos para no perder información.

También se obtiene información del estado de los sensores para en caso necesario parar la aplicación y recolocar los mismos con el fin de no perder datos y por último, en lo referente al casco, se obtienen los puntos necesarios donde introducir los marcadores, para la posterior escritura en el buffer y fichero de datos para su estudio posterior.

- Para la comunicación, se opta por una conexión TCP asíncrona de doble dirección, de modo que en cualquier momento tanto cliente como servidor puedan mandar/recibir información (marcadores, datos, log de ejecución, eventos de parada ...).
 - Para facilitar el uso al investigador, se dota de una IGU propia y externa a Emotiv, donde se puede ver los sensores, el transcurso de la ejecución de las aplicaciones, hacer uso de los botones para mandar información extra o para parar la aplicación.
- Además se cuenta con los dos ficheros donde se guarda toda la información obtenida del casco, añadiéndole marcas de guía y un fichero de depuración para facilitar la manipulación de los datos posteriores.

La siguiente figura muestra el flujo de datos de la información para que se vea con más claridad el proceso:

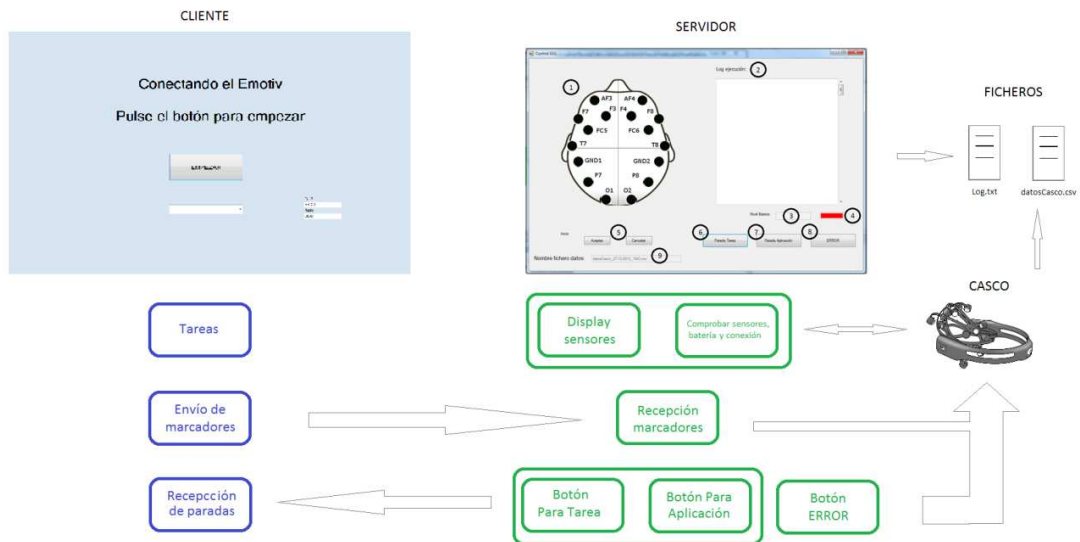


Figura 7. Flujo de datos entre los distintos componentes

5. Diseño

Como se comenta anteriormente, se parte de un proyecto anterior, por lo que los requisitos del sistema ya estaban en parte establecidos. Se parte de una aplicación desarrollada en C# con la aplicación *Microsoft Visual Studio 2010* a la que se le realizan una serie de cambios para su mejora y la creación de una nueva aplicación para que se comuniquen. En el caso de la nueva aplicación la programación se realizará con *Microsoft Visual Studio 2008* por temas de requisitos al usar ejemplos de la SDK aportados. Se continua con el lenguaje C# y se añade funcionalidad de *multithreading*.

5.1 Diagramas e Interfaz Gráfica

En primera instancia se procede a conocer en profundidad el programa del que se parte. En el esquema de la Figura 2 del apartado “*Contexto del Proyecto*” se puede apreciar el conjunto de ventanas principales de ejecución de tareas de la aplicación cliente. Además señalar que se trata de una ejecución SIN posibilidad de detención de la aplicación, a no ser que se “*mate*” la aplicación. Si se quiere ver con más detalle qué es lo que hace la aplicación, mirar en el capítulo “*Análisis*”.

Los diagramas de Caso de Uso representados a continuación nos dan una idea general del conjunto de actividades que llevarán a cabo las aplicaciones del proyecto. En la Figura 8 se plantea el caso de uso de la parte de la aplicación del cliente, mientras que en la Figura 9 se plantea la parte del servidor. Dado que los sistemas interactúan entre ellos, también aparecen como actores.

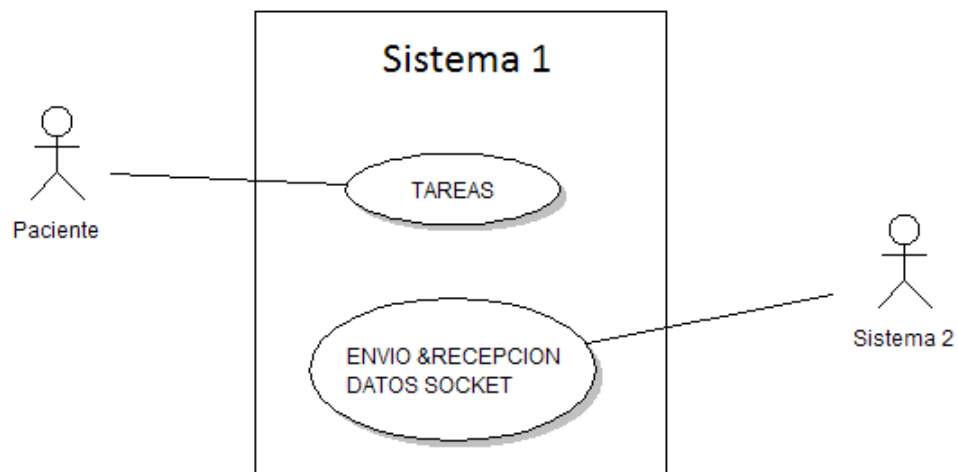


Figura 8. Caso de Uso (parte cliente)

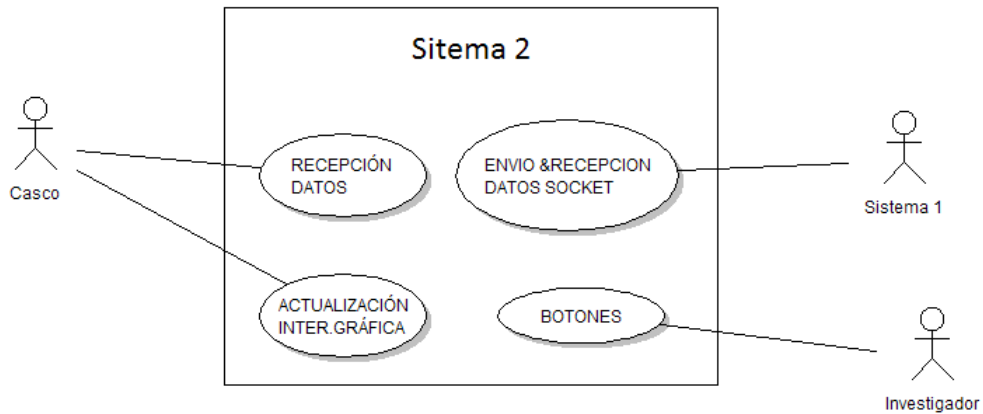


Figura 9. Caso de Uso (parte servidor)

Seguidamente se ve la funcionalidad y herramientas que ofrece el casco, junto con las librerías de la SDK para la manipulación “libre” del casco.

La librería *edk.dll* aportada en la SDK de Emotiv, es la que permite realizar operaciones encapsuladas, como por ejemplo el acceso a los datos obtenidos por el casco.

En la siguiente figura, hay un ejemplo de integración de la librería *edk.dll* con la API *EmoEngine* usada para la creación de un videojuego.

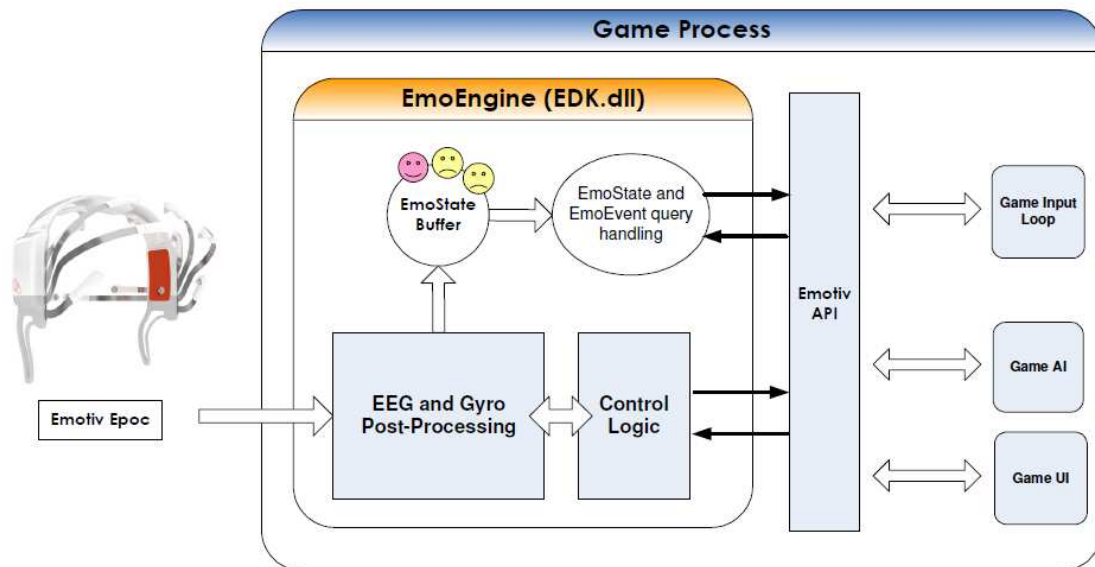


Figura 10. Uso de *EmoEngine* y resto de librería *edk.dll* con un videojuego

El juego del ejemplo consta de un módulo de control de valores de entrada del juego (*Game Input Loop*), el módulo de la parte de Inteligencia Artificial (*Game IA*) y el módulo que se encarga de la Interfaz de Usuario (*Game UI*). Todos ellos si quieren hacer uso del casco, se deberán comunicar utilizando los métodos que se proporcionan en la *Emotiv API*. Para trabajar sólo se puede acceder a estas funciones, el resto (que aquí se encierra en el módulo *EmoEngine*) se trata como una caja negra a la que el desarrollador no tiene acceso, excepto por las funciones proporcionadas.

En éste caso la API se comunica con el módulo *EmoState buffer*, buffer donde se guardan los datos. La lógica de control, los datos EEG y el giroscopio son inaccesibles para el desarrollador (excepto si se usan las funciones predefinidas), pero sí que se puede acceder a la señal ya post-procesada haciendo uso de la API.

MUY IMPORTANTE: Es requisito indispensable que la librería EDK.dll esté instalada en el PC del cliente (compilada con Microsoft Visual Studio 2005 SP1) y compartir las librerías de tiempo de ejecución de C/C++(CRT)

Para tener mejor visión de cómo se lleva a cabo la ejecución y las distintas etapas de la aplicación general, a continuación en la siguiente figura, se detalla el diagrama de estados.

En éste caso hay dos puntos de inicio, debido al arrancado de las aplicaciones cliente y servidor en distintos PCs, pero la terminación de la misma es concurrente a un sólo estado posible; para poder cerrar bien las conexiones creadas entre los dos sistemas y realizar la correcta liberalización de los recursos, usados previamente.

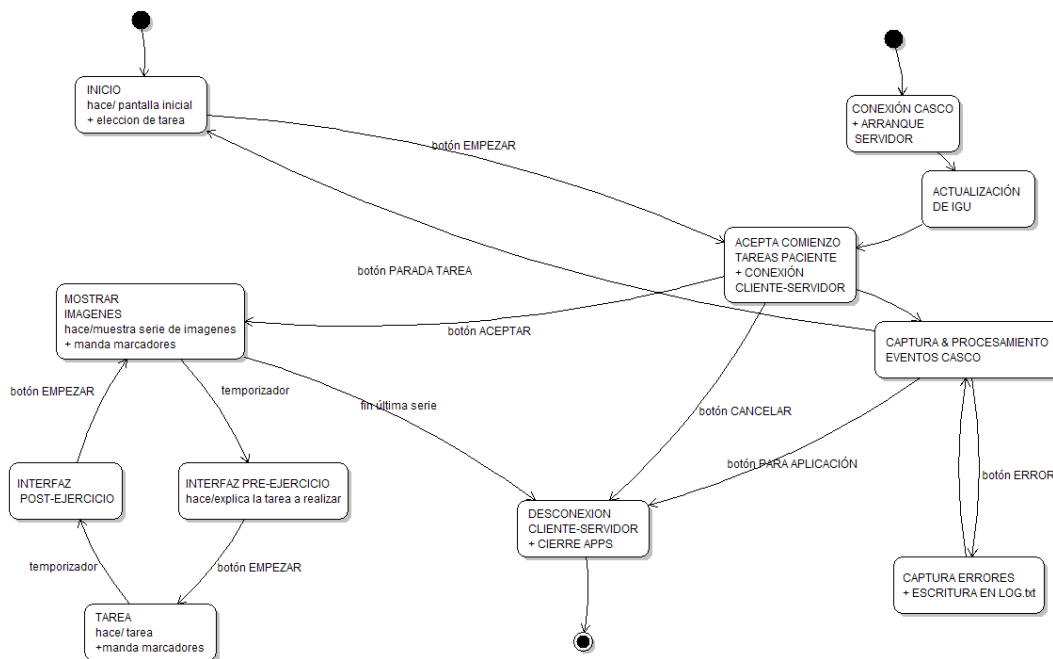


Figura 11. Diagrama de estados.

A la parte de la izquierda están organizados los estados referentes a la aplicación cliente y en la derecha los de la aplicación servidor. En la parte central está el último estado encargado del cierre de las conexiones, liberación de recursos y cierre de aplicaciones.

Viendo la estructura de las herramientas que el propio casco aportaba y ajustándose a las necesidades de la demanda del laboratorio, se llegó al diseño de la interfaz siguiente, pensando que era la mínima posible, pero con toda la funcionalidad necesaria. En la Figura 7 (capítulo “Análisis”) hay una imagen de una captura de pantalla del modelo

final de la aplicación del servidor y del cliente, junto con el flujo de datos . La apariencia de la interfaz del cliente se deja igual.

5.2 Estructura en ventanas/formularios

La aplicación cliente creada anteriormente, estaba diseñada con Visual Studio 2010, programada en C#, mediante un sistema de Formularios. No se quería cambiar mucho la estructura de lo ya creado y lo que se hizo fue añadirle la funcionalidad requerida.

Para ello se continuó con una creación de Formularios, pero en vez de sólo hacerlos visibles o no (uso `Form.Visible = true/false`), se ha optado por cerrar los formularios no necesarios, de modo que sea más fácil trabajar con los ya abiertos y se use menos memoria. Para ello se usan variables globales (uso `public Imagenes m_oImagenesForm`) con las que se referencia los formularios abiertos al crearlos y que se cerrarán cuando ya no sean necesarios.

Debido a la existencia de la opción de *Parada de Tarea*, es necesario mantener en una variable auxiliar y global (uso `public static Form vFormAnt`) que hace referencia a la ventana que actualmente está en ejecución. Si está en ejecución por ejemplo, la tarea 1 (en cualquiera de sus ventanas) y se recibe desde la aplicación servidora la orden de *Parar Tarea*, en ese instante, la referencia global `vFormAnt` hará referencia a dicha ventana y a continuación el control pasará a la ventana *Inicio* (que es donde se encuentra el método que regula la recepción de mensajes desde el servidor y a la que daremos el foco). Posteriormente cerrará el formulario referenciado por `vFormAnt` y de nuevo pondrá en bucle de espera, la recepción de datos del servidor.

El formulario/ventana *Inicio*, **NO** se cerrará **NUNCA**, hasta el cierre final de la aplicación, ya que es uno de los formularios principales y se perdería la conexión de datos.

El resto de formularios quedarán del mismo modo aunque añadiendo las llamadas a los métodos de envío de marcadores (generadas en las imágenes de muestra y en las tareas).

Para poder facilitar el trabajo y no repetir código, se genera una superclase/formulario padre (*AuxiliarClass*), del que heredarán los formularios ya creados. De éste modo contendrán las variables globales y métodos de éste formulario, como son:

- La variable de conexión cliente-servidor
- La variable de referencia a formulario de tarea actual
- Las variables de referencia a los formularios abiertos
- El método de envío de marcador cliente-servidor

5.3 Conexión entre aplicaciones

Como se ha comentado previamente, se realiza una conexión TCP cliente-servidor de doble sentido para poder gestionar el envío/recepción del conjunto de marcadores y mensajes de comunicación.

La dirección IP y el puerto para la conexión la obtendremos del fichero **config.ini** que a continuación explicaremos (en el punto 5.5). Si siempre se realiza en el mismo dispositivo es posible que no se tenga que cambiar el valor, pero en caso de ser

necesario, se puede hacer la modificación fácilmente con éste fichero (manteniendo el formato actual).

Una vez disponemos de la IP y el puerto de trabajo (por defecto el 30000) se activará el servidor (en la dirección "0.0.0.0" y mismo puerto) y se permanecerá a la espera, en un bucle, a que se realice una petición por parte del cliente (`AceptaCliente`). Posteriormente se hace uso de los manejadores y métodos de envío y recepción de información:

- `// Callback de recepción de marcadores que llegan desde el cliente`
`private void recibeCallback(IAsyncResult result)`
- `// Callback de envío de eventos desde el servidor al cliente`
`private void eventCallback(IAsyncResult res)`
- `// Callback de recepción de datos del servidor`
`private void recibeCallback(IAsyncResult res)`
- Y envío de marcadores (explicado a continuación)

Las variables que hacen referencia a la conexión cliente-servidor son:

- `public static TcpListener server` (en servidor)
- `public static TcpClient cliente` (en cliente)

La aplicación cliente, debe solicitar la conexión con el servidor y así se activará la recepción y envío de datos para comunicaciones posteriores.

5.4 Envío de marcadores

Una de las partes más destacables del proyecto y por lo que se solicitaba, es la **automatización de las marcas/marcadores** que se añaden en el transcurso del experimento.

Para luego, facilitar el estudio posterior de las señales EEG captadas por el casco, se añaden a los datos de captación del casco, unos valores extra para diferenciar partes de las señales, se trata de los marcadores (MARKER). La correspondencia marcador-significado, se ve en la siguiente tabla:

Cód.	Información
300	Aplicación cliente en pantalla Inicio (Sujeto)
10	Inicio conexión establecida (casco y cliente-servidor).
1	Inicio tarea 1
11	Fin tarea 1
2	Inicio tarea 2
21	Fin tarea 2
3	Inicio tarea 3
31	Fin tarea 3
3xy1	Aparición en tarea 'x' de foto 'y' (ANTES de la tarea)
35y1	Aparición en tarea '5' de foto 'y' + FINAL DE LA APLICACION
4	Inicio tarea 4
41	Fin tarea 4
5	Final de aplicaciones (servidor y cliente) y cierre conexión con casco
6	Botón ERROR
7	Botón PARADA TAREA
8	Botón PARADA APLICACIÓN
9	Parada de tarea actual en ejecución y vuelta al formulario de Inicio
112	Inicio - Aceptar (Desde Aplicación Servidor, dará comienzo la ejecución de la Aplicación Cliente)
666	Inicio - Cancelar (Desde la Aplicación Servidor se cerrarán las aplicaciones Cliente y Servidor)

Figura 12. Tabla con significado de los marcadores

Dichos marcadores responden a ciertos eventos generados cuando hay un:

- Inicio/Fin de una tarea específica en la Aplicación Cliente/Servidor
- Cuando se muestra cada una de las imágenes del experimento en *LineaBase* (comentados en el capítulo de “*Antecedentes*”).
- Fin de la aplicación general
- Cierre de conexión cliente-servidor o con el desconexión con el casco
- Pulsación de Botones, en la aplicación servidor (Error, Parada Tarea, Parada Aplicación)
- Botón de Aceptación/Cancelación, en la aplicación servidor, para dar paso/no al inicio de la aplicación cliente.

Normalmente dichos eventos los producirá la aplicación cliente (aunque también pueden producirse en la parte servidor) y se enviarán mediante el socket creado en la conexión TCP al servidor. Un ejemplo de ejecución sería:

- La aplicación cliente genera el evento a registrar
- El código se envía del cliente al servidor automáticamente en ese momento npor el socket creado para ello:
`envioMarcador(socket, BitConverter.GetBytes(marcador));`

- La aplicación servidor hace la petición al casco de incluir el marcador en la secuencia de datos
- El código se escribe en el buffer de datos del casco mediante la API, reflejándolo así en la columna MARKER (que posteriormente se escribirá en el documento de escritura de los datos):
`EmoEngine.Instance.DataSetMarker((uint)userID, marcador);`
- Al realizar la escritura de datos del buffer del casco al fichero de datos, se escribe un bloque de varias líneas de datos almacenados en el buffer del casco. La escritura sería del siguiente modo

(**NOTA:** Para no exceder el tamaño del buffer y la consecuente pérdida de datos, la escritura de datos se realiza cada 15 ms):

```

1  ...
2  int _bufferSize = data[EdkDll.EE_DataChannel_t.TIMESTAMP].Length;
3
4  // Escritura de datos en fichero
5  TextWriter file = new StreamWriter(filename, true);
6
7  for (int i = 0; i < _bufferSize; i++)
8  {
9      // Escritura ...
10     foreach (EdkDll.EE_DataChannel_t channel in data.Keys)
11     {
12         file.Write(data[channel][i] + ";");
13         flagCasco = true;
14         button7.BackColor = Color.Lime;
15         ...
16     }
17     file.WriteLine("");
18
19 }
20 file.Close();
21 ...

```

Figura 13. Fragmento de código de escritura en fichero datosCasco.csv

NOTA: Se activa un botón en color Lime para que el investigador pueda ver en todo momento si el proceso se ejecuta correctamente o no (en caso negativo será de color Rojo)

Por tanto el marcador queda registrado en el fichero en el momento lo más exacto posible para cada uno de los eventos.

5.5 Ficheros auxiliares

5.5.1 Fichero de configuración inicial (*config.ini*)

Para hacer más cómoda la ejecución de la aplicación, se parte un fichero de configuración inicial (*config.ini*) en el que se pueden poner los valores por defecto que se van a usar y por tanto no tener que introducirlos manualmente en cada ejecución (Eso sí, respetando el formato).

En el caso actual los únicos valores que necesitamos son la IP del servidor sobre la que solicitará la conexión el cliente y el puerto sobre el cual se ejecutará la aplicación.

En caso que más adelante fuese necesario añadir valores (o modificar los existentes), podría hacerse, pero siempre manteniendo el mismo formato para que la lectura del fichero se realice correctamente.

5.5.2 Fichero *fichLog.txt*

En dicho fichero, se recogen los eventos que van teniendo lugar en la aplicación. Se muestra en la IGU de la Aplicación Servidor y además se queda grabado, por si posteriormente hiciera falta acceder a el por temas de *depurado* de la aplicación en sí, o porque los investigadores lo requieran para ver el trascurso de la aplicación y de que todo funciona correctamente.

5.5.3 Fichero *datosCasco_dia-mes-año_hhmm.csv*

Lo primero, explicar el por qué de ese nombre. Para que las distintas sesiones con los sujetos queden reflejadas y diferenciadas unas de otras, por cada vez que se ejecuta la aplicación general (Inicio >> realización de tareas >> Parada de Aplicación) se genera un archivo distinto. Para que el investigador posteriormente sepa de quien se trata el estudio realizado, solo deberá guardar datos como el día y la hora en la que realiza el estudio, para poder relacionarlo.

El formato del fichero, o las partes que interesan para el estudio son las siguientes:

ELEMENTO	DESCRIPCIÓN
COUNTER	Índice para la comprobación de que no hay pérdidas de datos en la recepción [0 -128]
AF3, F7, F3, ... F4, F8, AF4	Recoge los valores de los sensores del casco según el gráfico de la IGU.
MARKER	Marcador. Aporta información extra (mirar la tabla)

Figura 14. Formato fichero datosCasco.csv

Dichos ficheros se guardarán en la misma ubicación que donde esté el ejecutable de la aplicación.

5.6 Correspondencia sensores con regiones del cerebro

Para el diseño de la parte gráfica de los sensores, donde se muestra su estado (bueno, regular y malo) se ha llevado a cabo una correspondencia de valores. En el caso de Emotiv, lo regula mediante cinco valores (negro, rojo, naranja, amarillo y verde), en este caso, se usan solo cuatro, ya que el naranja se ha desechado por no tener tantos colores de muestra y dar mayor seguridad. Los colores validos serían:

- Negro: el sensor no capta señal (nula, o por el soporte o por el sujeto)
- Rojo: el sensor no capta buena señal (débil o falta de batería)
- Amarillo: valor intermedio entre rojo y verde (precaución)
- Verde: valor óptimo de uso.

En caso que un número elevado de sensores no capten la señal correcta, el luminoso del botón de la IGU, se pondrá rojo como indicativo de que algo falla.

La correspondencia de los sensores se muestra a continuación:

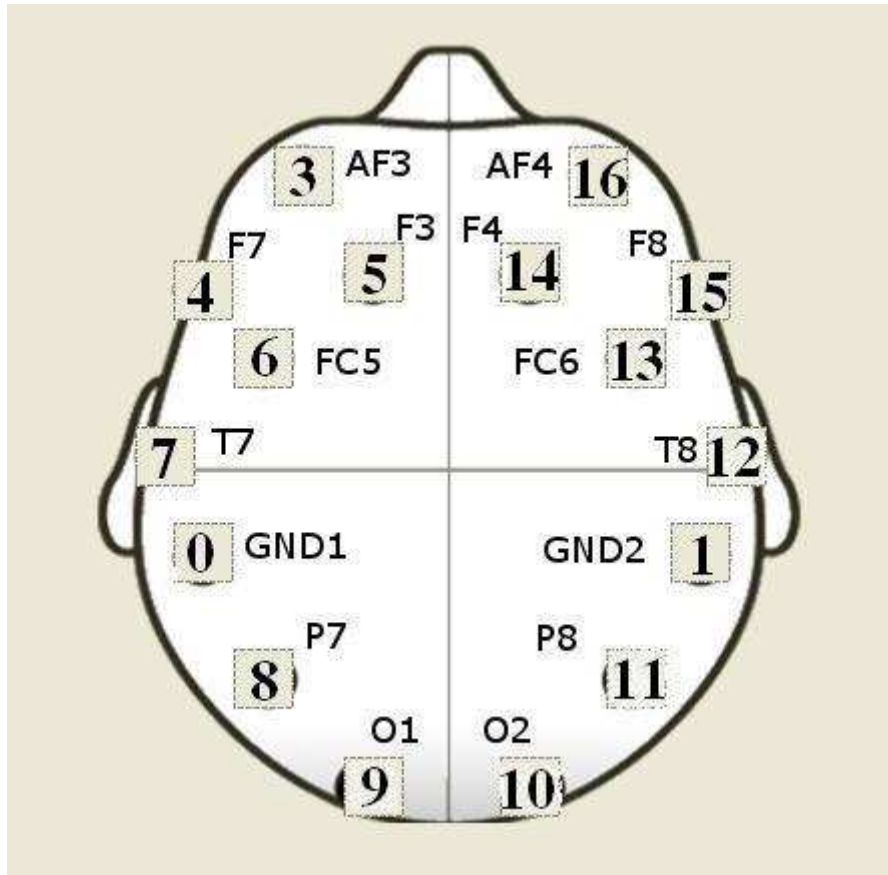


Figura 15. Correspondencia científica-numérica de los sensores

En el dibujo se muestra la cabeza con los distintos puntos donde se encuentran los sensores, por un lado el nombre científico y por otro el número correspondiente en la aplicación servidor creada.

En la carpeta **ANEXOS** del Proyecto se observa una ejecución de la aplicación al completo.

6. Resultado

Teniendo en cuenta los objetivos que se marcaron al inicio y que se reflejan en el capítulo de “*Introducción*” a continuación se comentan los logros principales obtenidos durante la realización del Proyecto.

- La automatización de la introducción de las marcas sin necesidad de una persona se ha logrado, por lo que se ha cumplido el objetivo principal. Además se han añadido a la aplicación nuevas funcionalidades como:
 - o La visión del estado de los sensores del casco.
 - o Un indicador que muestra si se están obteniendo bien o no los datos del casco. También alerta cuando hay errores de batería, conexiones, sensores, etc.
 - o Un display de log para que el investigador pueda seguir mejor el transcurso del estudio.
 - o Botones auxiliares que aumentan la funcionalidad de la aplicación, dándole ligereza y consistencia.
 - o Unos ficheros (log y de datos) con la fecha y hora para mejorar la correspondencia estudio-sujeto a la hora de plantear los experimentos.

- Reducir la latencia de envío de datos entre casco-servidor, y escritura en fichero hace que los marcadores estén colocados en la posición correcta.
- En cuanto a la disminución de errores por factor humanos, no ha sido posible hacer mediciones, pero se plantea como trabajo futuro.
- Anteriormente no se podía parar la aplicación a no ser que se “matase” el proceso. Con la incorporación de los botones de “Parada Tarea” y “Parada Aplicación”, en cualquier momento es posible pararla correctamente y sin esperas.
 - o En concreto, con el botón “*Parada Tarea*” es posible parar cualquier tarea, en cualquier punto (Imagen de muestreo, tarea como tal) y volver a la pantalla de “*Inicio*” sin necesidad de generar otro archivo de grabado de datos, ni comienzo nueva sesión o tener varios ficheros para un sujeto concreto.
- El uso de las marcas, como inicio de aplicación, inicio de tarea, final de tarea, imagen de reposo mostrada, botón de error extra gestionado por el investigador, etc.
- Se ha implementado el acceso directo a los datos del casco, sin hacer uso de las herramientas de alto nivel del fabricante. Ello permite tener más flexibilidad y control de los ajustes del caso sin necesidad de abandonar el entorno.

Con las ampliaciones realizadas se han cumplido los objetivos y se han desarrollado nuevas opciones que dan valor añadido a la aplicación.

7. Conclusiones

En este capítulo se presentan las experiencias obtenidas durante la realización del proyecto. Con respecto al casco y las herramientas de la SDK, se pueden resaltar una serie de ventajas e inconvenientes del uso del mismo.

- Ventajas:
 - o Buena movilidad y reducido peso
 - o Facilidad de colocación en la cabeza del sujeto (con pelos cortos)
 - o Integración con videojuegos
 - o Posibilidad de hacer uso de los *keystrokes* para control en entretenimiento u otros campos
 - o Posibilidad, con la versión de Investigador, de acceder a los datos, aunque sean post-procesados y no la señal capturada
- Inconvenientes:
 - o Los sensores en ocasiones no captan bien la señal en función del espesor del cuero cabelludo de la persona
 - o Los sensores son móviles y con la solución acuosa para aumentar la conductividad se oxidan, por lo que pierden sensibilidad de captación y hay que sustituirlos
 - o Menor número de sensores (y por lo tanto de canales) que uno médico profesional
 - o Es un poco complicado al inicio usar la SDK y la configuración con las dlls

En resumen, en cuanto al dispositivo, aun no alcanzando la precisión de un sistema de EEG profesional, consideramos que es una buena herramienta, bien sea para estudios médicos no tan restrictivos, para el entretenimiento o para ayudar a personas con discapacidades. Este casco puede formar parte de sistemas que mejoren la calidad de vida personas con discapacidad.

A nivel personal, este proyecto me ha dado la oportunidad de trabajar con un dispositivo de captación de ondas EEG, de aprender a entender y continuar con el trabajo realizado por otras personas y tener la libertad de poder diseñar la aplicación, teniendo en cuenta los requisitos marcados.

Por último también quisiera mencionar la experiencia de trabajar con *multithreading*, que aunque en un principio se quiso evitar, al final fue necesaria para la realización de la funcionalidad de “Parada Tarea”.

8. Trabajo futuro

En esta sección presentaré diversas líneas que pueden surgir a partir del trabajo realizado. El dispositivo tiene un gran potencial y se puede usar para distintos tipos de estudios, pero antes se deberían hacer pruebas de validación para asegurar la calidad de los datos.

Por el momento el grupo de trabajo de *LabHuman* quiere continuar haciendo uso del mismo para hacer estudios de “potenciales evocados”. Queda pendiente también la separación de los envíos de marcadores y el resto de funcionalidad, de modo que sea independiente y se pueda construir una librería externa que se pueda usar en otros estudios.

Este dispositivo también se puede utilizar en entornos virtuales haciendo uso, por ejemplo, de un *plugin* existente para *Unity3D*.

Por último queremos mencionar el nuevo dispositivo comercializado por Emotiv, denominado “Emotiv Insight” [13], que es en apariencia similar al actual pero que tiene las siguientes diferencias:

- Menos nodos de captación, 5 EEG +2 (Para aumentar la resolución espacial)
- Sensor polímero propietario mejorado, que aumenta la conductividad
- Mejor agarre (está pensado para uso deportivo)
- Compatibilidad con dispositivos móviles

9. Bibliografía

1. **Emotiv.** *EEG Neuroheadset* [En línea] <http://www.emotiv.com/>
2. **Alejandro Rodríguez, Beatriz Rey Solaz y otros.** *Validación de un sistema de EEG inalámbrico como medida neurofisiológica de ondas de actividad cerebral en sujetos humanos.* Comunicación del SEPNECA.
3. **Wikipedia.** *Electroencefalografía* [En línea] <http://es.wikipedia.org/wiki/Electroencefalograf%C3%ADa>
4. **Medline Plus.** *Información de Salud* [En línea] <http://www.nlm.nih.gov/medlineplus/>
5. **NeuroSky.** *Brain wave sensor for everybody* [En línea] <http://www.neurosky.com/Default.aspx>
6. **OCZ.** *OCZ Neural Impulse Actuator* [En línea] http://www.legitreviews.com/cebit-2007-oczs-neural-impulse-actuator_475
7. **Instituto Ai2 (UPV).** *Subgrupo de trabajo de Gráficos por Computador de la Universidad Politécnica de Valencia. LOTO.* [En línea] Expo 1: <https://jira.ai2.upv.es/confluence/display/LOTO/2011/10/25/WGM+35.+EEG+Control+%28I%29> Expo 2: <https://jira.ai2.upv.es/confluence/display/LOTO/2011/12/13/WGM+40.+EEG+Control+%28II%29>
8. **EEG Lab.** *An open source environment for electrophysiological signal processing.* [En línea] <http://sccn.ucsd.edu/eeglab/>
9. **Neuromarketing.** *Eye Tracking the Burlesque Movie Trailer with EEG Emotion Response Analysis.* [En línea] <http://www.youtube.com/watch?v=RaLCLVSBIfc>
10. **Neuromarketing.** *The re-socialisation of TV.* [En línea] <http://www.youtube.com/watch?v=rm2gPgGNx1k>
11. **Videojuegos.** *Kinect + Emotiv + Brain Hacking.* [En línea] <http://www.youtube.com/watch?v=zHwyKLk2LhE>
12. **Medicina.** *Optimi, Online Predictive Tools for Intervention in Mental Illnes.* [En línea] <http://www.optimiproject.eu/>
13. **Emotiv.** *Emotive Insight* [En línea] <http://emotivinsight.com/>

14. **Ayuda programación C#** [En línea] <http://code.msdn.microsoft.com/>

15. **Francisco Javier Ceballos Sierra**. *Enciclopedia de Microsoft Visual C#. 3*, Madrid: Ra-ma, 2010

NOTA: Algunas imágenes se han tomado del manual del SDK del dispositivo

Otras fuentes tomadas como ejemplos y documentación han sido de la SDK de Emotiv, tales como:

- Ficheros >> examples
- Ficheros >> examples_DotNet
- API References for .NET
- API References
- Emotiv SDK User Manual

12. ANEXOS

13. Manual de Usuario de la Aplicación

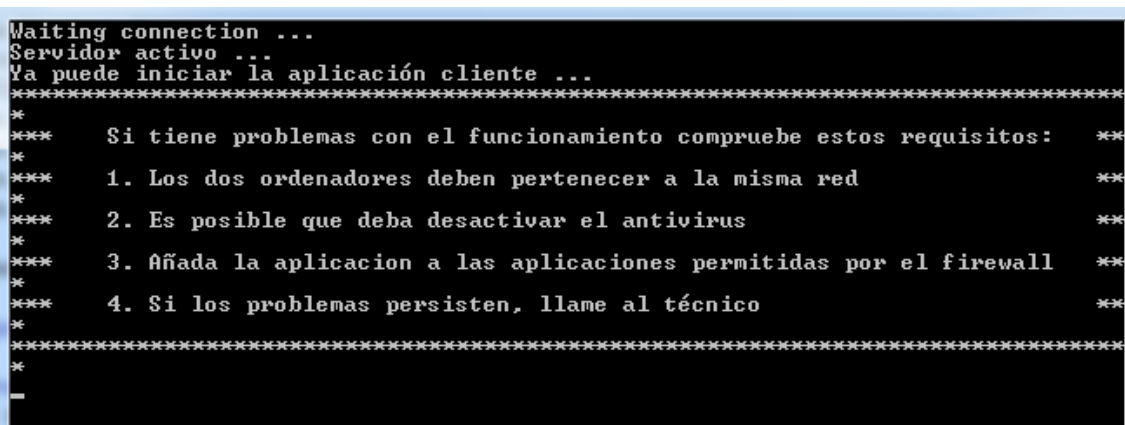
La aplicación, que sirve para la validación de un dispositivo de monitorización de EEG, consta de dos subprogramas; el primero para la ejecución de la aplicación del servidor, que gestiona el correcto funcionamiento del programa y que los dispositivos funcionen correctamente y captar los resultados obtenidos del casco; el segundo para la ejecución de la aplicación del cliente, donde se realizarán las tareas del estudio.

A continuación se procede a la explicación detallada de las aplicaciones, su funcionamiento y la obtención de los datos de respuesta.

13.1 Manual Aplicación Administrador (Servidor)

Ésta aplicación deberá ejecutarse en primer orden, para poder posteriormente establecer la comunicación entre las dos aplicaciones.

Nada más ejecutarse aparecerá la siguiente ventana:



```
Waiting connection ...
Servidor activo ...
Ya puede iniciar la aplicación cliente ...
*****
*** Si tiene problemas con el funcionamiento compruebe estos requisitos: ***
*** 1. Los dos ordenadores deben pertenecer a la misma red ***
*** 2. Es posible que deba desactivar el antivirus ***
*** 3. Añada la aplicacion a las aplicaciones permitidas por el firewall ***
*** 4. Si los problemas persisten, llame al técnico ***
*****
*
```

Fig 1. Pantalla de inicio de aplicación servidor

En ella se muestra el mensaje de espera de conexión del casco (dispositivo HW para realizar las mediciones de EEG), que el servidor está activo y a la espera de conexión y algunos consejos a tener en cuenta en caso de fallo.

Tras ello se iniciará la Aplicación Cliente y se llevará a cabo la conexión servidor-cliente dando al botón **EMPEZAR (de la aplicación Cliente)**, **además de hacer click en el mensaje que aparecerá en pantalla**. Tras ello, en la Aplicación Servidor, se tendrá que (Aceptar/Cancelar (5)) la ejecución por parte del investigador. Obteniendo en pantalla, la siguiente ventana:

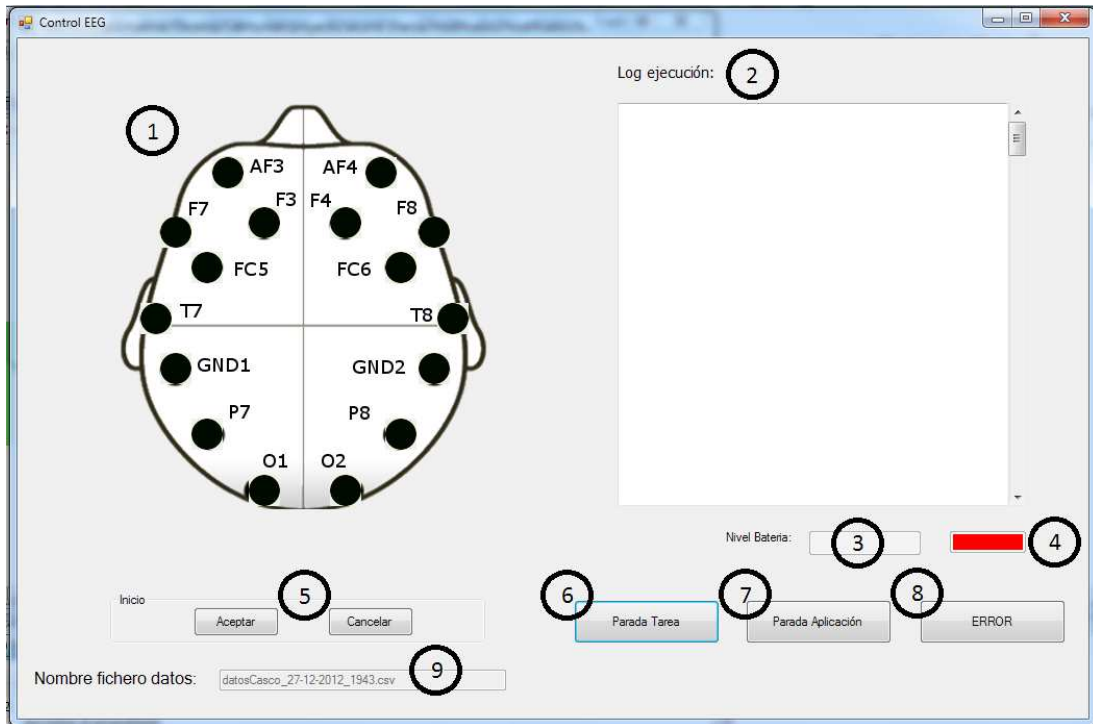


Fig 2. Aplicación servidor

En la parte superior-izquierda (1) se encuentra el dibujo con la calidad de conexión de los sensores del casco. Hay 4 colores:

- Negro: sensor no activo.
- Rojo: mala calidad de la conexión.
- Amarillo: conexión no estable.
- Verde: conexión correcta.

En la parte superior-derecha (2) se encuentra el espacio de muestreo de la información relevante durante la ejecución; conexiones, envíos de marcadores, errores... Dicha información también se guarda en el archivo **“fichLog.txt”**, único para la aplicación (aunque se realicen varios estudios, el fichero será siempre el mismo, siempre que no termine la aplicación o se de al botón “Parada Aplicación”).

Bajo él, se muestran dos espacios (3) con el nivel de batería del casco y (4) que indica la correcta captación de los datos procedentes del casco (verde: todo bien, rojo: algo falla, mirar en fichLog.txt o avisar al técnico).

Una vez realizada la conexión y con el zócalo (4) de color verde, indicará que todo es correcto y que los datos del casco se recogen correctamente, sino el color será rojo y habrá que ver de dónde viene el fallo.

En éste momento comenzará el experimento y los ejercicios pertinentes.

Desde la IGU actual también se puede gestionar la ejecución del cliente; se recogerá la información de los sensores, del envío de información y 3 acciones extra más, que se detallan a continuación:

- Botón Parada Tarea (6):

Los ejercicios de los sujetos constan de varias tareas, la funcionalidad de éste botón es poder en cualquier momento parar la tarea actual, sea por el motivo que sea, de modo que se retornará a la pantalla de Inicio, donde se activarán las casillas de “Nueva Tarea”. Para poder elegir a qué tarea queremos ir y dar la orden con el botón “Confirmar” (se mostrará la página inicial de dicha tarea) y comenzar la ejecución de ella. Todo ello quedará reflejado en el fichero de datos según la tabla de marcadores que posteriormente se indicará. Los datos del sujeto seguirán guardados en el mismo fichero.

- Botón Parada Aplicación (7):

Como su nombre indica, para la Aplicación general, se parará tanto la Aplicación Cliente como la Aplicación Administrador, la conexión de datos entre las mismas y el cierre de los ficheros de almacenamiento de la información, tanto del sujeto como de la aplicación.

- Botón ERROR (8):

Dicho botón manda un marcador al fichero de recolección de datos del sujeto con el código de Error. Será usado cuando se quiera reflejar un error sin importancia en el envío de los datos pero no sea necesaria la finalización de la Aplicación porque así lo estime el Administrador.

Por último la información de los datos recogidos por los sensores del casco se guardará en un archivo con extensión .csv cuyo nombre (9) se asigna automáticamente cada vez que se ejecuta la Aplicación Cliente siguiendo el criterio "**datosCasco_dia-mes-año_hhmm.csv**" y que posteriormente se podrá abrir con el programa Excel (abriéndolo como *Datos >> Obtener datos externos >> Desde texto >> Delimitados >> Siguiente con separador punto y coma*).

13.2 Manual Aplicación Cliente (Sujeto)

Tras iniciarse la Aplicación Administrador, se iniciará la Aplicación Cliente, apareciendo así la pantalla de Inicio (Fig. 3), donde se indicará la dirección

IP del servidor (Aplicación Administrador) y la tarea por la que iniciar el ejercicio (por defecto la Tarea 1).

Inicialmente los valores de IP y puerto estarán escritos porque se obtienen de un fichero de configuración, pero en caso de ser necesario cambiarlo, se podría hacer, ya que los cambios quedarían reflejados en el fichero config de nuevo.

Posteriormente y para realizar el inicio de la aplicación, se hará *click* en el botón Aceptar del mensaje por pantalla y finalmente al botón EMPEZAR, para dar comienzo a los ejercicios.

NOTA: Se debe tener en cuenta que ambas aplicaciones estén conectadas a la misma red y añadir la aplicación a las excepciones del cortafuegos para que no la bloqueen. (El antivirus también puede dar problemas)

NOTA (para administradores): se dispone de un fichero config.ini para indicar la IP y el Puerto.

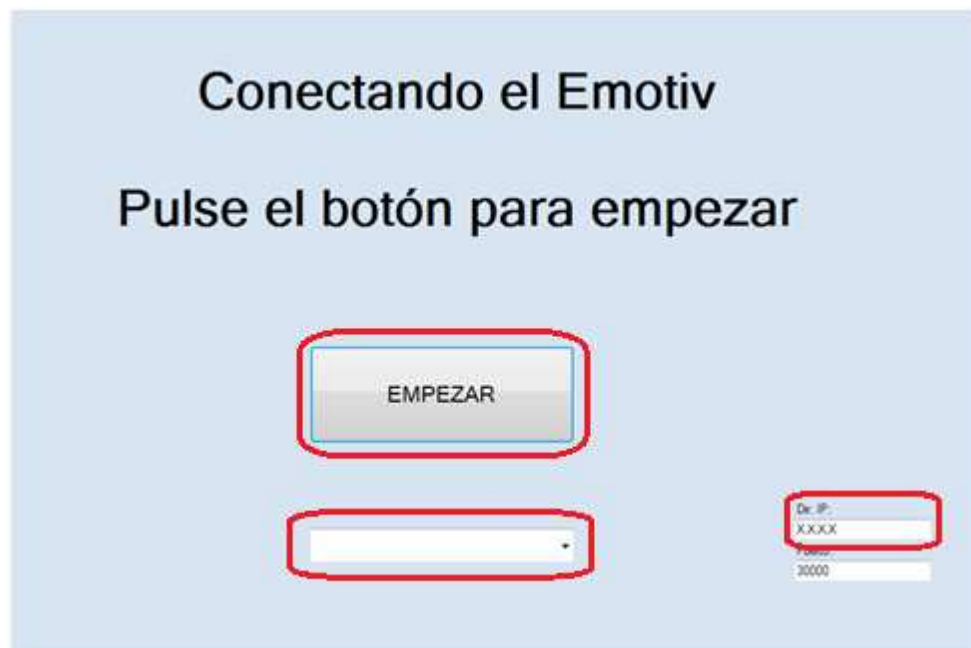


Fig 3. Aplicación Cliente. Pantalla Inicio

Las tareas darán comienzo con el muestreo de unas imágenes, que servirán de calibración del estado del sujeto y cuyos impulsos serán registrados por el casco y enviado automáticamente al archivo de datos. Seguidamente se realizarán los ejercicios.

En todo momento el Administrador puede gestionar la Aplicación Cliente como se detalla en el punto anterior.

13.3 Lectura del **fichero de datos**

En el fichero "**datosCasco_dia-mes-año_hhmm.csv**" se guarda la información de la sesión de cada sujeto.

Tener en cuenta que por cada ejecución de la Aplicación Cliente se generará un fichero nuevo, en caso de usar el botón Parada Aplicación será el Administrador quien deberá manualmente de encargarse de coger los datos del sujeto que quedarán separados en varios ficheros, se aconseja para ello mirar la fecha y hora.

Dicho fichero se encuentra en la carpeta del fichero ejecutable.

Para la interpretación de dicho fichero nos fijaremos en su cabecera:

ELEMENTO	DESCRIPCIÓN
COUNTER	Índice para la comprobación de que no hay pérdidas de datos en la recepción [0 -128]
AF3,F7, F3, ... F4, F8, AF4	Valores de los sensores del casco según el gráfico de la IGU.
MARKER	Marcador. Aporta información extra (mirar la tabla)

Fig 4. Formato fichero datosCasco.csv

13.4 Tabla de Marcadores

Cód.	Información
300	Aplicación cliente en pantalla Inicio (Sujeto)
10	Inicio conexión establecida (casco y cliente-servidor).
1	Inicio tarea 1
11	Fin tarea 1
2	Inicio tarea 2
21	Fin tarea 2
3	Inicio tarea 3
31	Fin tarea 3
3xy1	Aparición en tarea 'x' de foto 'y' (ANTES de la tarea)
35y1	Aparición en tarea '5' de foto 'y' + FINAL DE LA APLICACIÓN
4	Inicio tarea 4
41	Fin tarea 4
5	Final de aplicaciones (servidor y cliente) y cierre conexión con casco
6	Botón ERROR
7	Botón PARADA TAREA
8	Botón PARADA APLICACIÓN
9	Parada de tarea actual en ejecución y vuelta al formulario de Inicio
112	Inicio - Aceptar (Desde Aplicación Servidor, dará comienzo la ejecución de la Aplicación Cliente)
666	Inicio - Cancelar (Desde la Aplicación Servidor se cerrarán las aplicaciones Cliente)

	y Servidor)
300	Aplicación cliente en pantalla Inicio (Sujeto)

Fig 5. Tabla con significado de los marcadores

13.5 Fichero **config.ini**

Creado para hacer más cómoda la ejecución de la aplicación. Se parte de un fichero de configuración inicial (*config.ini*) en el que se pueden poner los valores por defecto que se van a usar y por tanto no tener que introducirlos manualmente en cada ejecución (Eso sí, respetando el formato).

En caso que más adelante fuese necesario añadir valores (o modificar los existentes), podría hacerse, pero siempre manteniendo el mismo formato para que la lectura del fichero se realice correctamente.

En el caso actual los únicos valores que necesitamos son la IP del servidor sobre la que solicitará la conexión el cliente y el puerto sobre el cual se ejecutará la aplicación.

Este fichero es conveniente que lo manipule una técnico, para evitar problemas futuros.

13.6 Fichero **fichLog.txt**

En dicho fichero, se recogen los eventos que van teniendo lugar en la aplicación. Se muestra en la IGU de la Aplicación Servidor y además se queda grabado, por si posteriormente hiciera falta acceder a el por temas de *depurado* de la aplicación en sí, o porque los investigadores lo requieran para ver el trascurso de la aplicación y de que todo funciona correctamente.