UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE
TELECOMUNICACIÓN

DISEÑO Y DESARROLLO DE UNA APLICACIÓN PARA CONTROLAR UN TELÉFONO MÓVIL MEDIANTE SISTEMAS BRAIN COMPUTER INTERFACE (BCI) ORIENTADA A PERSONAS CON GRAVE DISCAPACIDAD

Autor:
D. Eduardo Santamaría Vázquez
Tutor:
Dr. D. Roberto Hornero Sánchez
Valladolid, 26 de Julio de 2016

TÍTULO:	**Diseño y desarrollo de una aplicación para controlar un teléfono móvil mediante sistemas Brain Computer Interface (BCI) orientada a personas con grave discapacidad**
AUTOR:	**D. Eduardo Santamaría Vázquez**
TUTOR:	**Dr. D. Roberto Hornero Sánchez**
DEPARTAMENTO:	**Teoría de la Señal y Comunicaciones e ingeniería Telemática.**

TRIBUNAL

PRESIDENTE:	**Dr. D. Roberto Hornero Sánchez**
VOCAL:	**Dr. D. Jesús Poza Crespo**
SECRETARIO:	**Dr. Dña. María García Garañón**
SUPLENTE:	**Dr. D. Carlos Gómez Peña**
SUPLENTE:	**Dr. D. Miguel López Coronado**

Resumen del TFG

Desde el descubrimiento del electroencefalograma (EEG), que fue desarrollado por Hans Berger en 1929, la electrofisiología ha avanzado enormemente. Gracias a ese descubrimiento, las ondas cerebrales, hasta entonces prácticamente desconocidas, comenzaron a estudiarse para el diagnóstico de enfermedades como la epilepsia o diversos trastornos del sueño.

Fue en la segunda mitad del siglo XX cuando se empezó a sugerir la aplicación de las señales cerebrales para la comunicación con el medio sin la mediación de músculos y nervios periféricos, algo que podría mejorar significativamente la vida de personas con grave discapacidad motora. El primer sistema de este tipo fue desarrollado por el Dr. Jacques Vidal en 1977, bautizándolo posteriormente como BCI (*Brain Computer Interface*). Estos sistemas permiten el control de máquinas, traduciendo directamente en comandos las intenciones de un usuario mediante la extracción y clasificación de características.

Este trabajo tiene el objetivo de desarrollar y evaluar una aplicación BCI que permita el control de un Smartphone y sus principales funcionalidades. Tras realizar una revisión del estado del arte, se concluye que la mejor manera de alcanzar este objetivo es mediante una aplicación basada en el paradigma *oddball*. Este paradigma BCI permite seleccionar comandos representados en las celdas de una matriz mediante la detección en el EEG de potenciales P300 provocados por estímulos visuales que aparecen en tiempos aleatorios y generados de forma endógena por el usuario de la aplicación.

El sistema diseñado se encuentra distribuido entre un ordenador y el dispositivo móvil a controlar. Debido a su mayor capacidad, la implementación del paradigma *oddball* y el procesado de señal se realizan en el ordenador, que traduce en comandos las intenciones del usuario. Posteriormente, estos se envían al dispositivo móvil mediante tecnología Bluetooth. Una vez recibido el comando, el Smartphone lo ejecutará. Las tareas disponibles en la aplicación permiten al usuario realizar llamadas, mandar SMS, añadir, ver o eliminar contactos, hacer fotos y visualizarlas, o añadir alarmas y citas al calendario.

Una vez desarrollada, la aplicación fue evaluada por cinco sujetos sanos que tuvieron que realizar distintas tareas de control del Smartphone con dificultad creciente, repartidas en dos sesiones realizadas en días diferentes. Los resultados obtenidos fueron satisfactorios, con una precisión media cercana al 95% bajo condiciones controladas en un laboratorio. Tras la realización de las tareas, los sujetos de estudio completaron un cuestionario de satisfacción que permitió conocer su opinión del sistema implementado y realimentar el proyecto con sus sugerencias.

Palabras clave

Brain Computer Interface, electroencefalograma, potenciales evocados P300, extracción y traducción de características, artefactos, plataformas móviles, Android, paradigma *odd-ball*.

Abstract

Since the development of electroencephalogram (EEG) by Hans Berger in 1929, electrophysiology has grown up greatly. Thanks to this discovery, the brain waves began to be studied for the diagnosis of diseases such as epilepsy or sleep disorders.

It was in the second half of the twentieth century when the scientific community began to suggest the application of Brain signals for communication without the mediation of peripheral muscles or nerves. This could improve people's lives with severe motor disability significantly. This first system was developed by Dr. Jacques Vidal in 1977, naming it BCI (Brain Computer Interface). These systems permit to control machines, translating the user's intentions into commands by means of extracting and translating the EEG signal features.

This paper aims to develop and evaluate a BCI application that allows the users to control a mobile phone and its main features. After doing a review of the state of the art, it is concluded that the best way to accomplish this goal is by developing an application based on the *oddball* paradigm. This BCI paradigm selects commands represented in the cells of a matrix by detecting P300 potentials in EEG, provoked by random visual stimuli and generated by the user, endogenously.

The designed system is distributed between a computer and the mobile device which the user has to control. Due to its grater capacity, *oddball* paradigm and the signal processing are performed on the computer that translates the user's intentions into commands. Afterwards, they are sent to the mobile device using Bluetooth technology. As son as the command is received, the mobile device will run it. The tasks available in the application allow the user to make calls, send SMS, add, view or delete contacts, take pictures and display them, or add alarms and events to the calendar.

The application was developed and evaluated by five healthy subjects, who had to perform various tasks to control the Smartphone with increasing difficulty, spread over two sessions held on different days. The results obtained were satisfactory, with an average accuracy of 95% performed under controlled laboratory conditions. After these tasks, study subjects completed a satisfaction questionnaire that allowed to know their personal opinion of the system, and fed back the project with their suggestions.

Keywords

Brain Computer Interface, electroencephalogram, P300 evoked potentials, feature extraction and translation, artifacts, mobile platforms, Android, *odd-ball* paradigm.

AGRADECIMIENTOS

En primer lugar, quisiera agradecer a Roberto Hornero Sánchez la oportunidad que me ha brindado para realizar este trabajo y, en general, al Grupo de Ingeniería Biomédica de la Universidad de Valladolid.

A Victor Martínez, por ayudarme constantemente en el desarrollo de este trabajo compartiendo su experiencia en el mundo de la Ingeniería Biomédica y los sistemas BCI conmigo.

A todos mis compañeros y amigos, quienes han hecho posible que este ciclo conluya felizmente y afronte con ganas lo que está por llegar.

A Cristina por apoyarme incansablemente durante todos estos años y estar presente tanto en los buenos como en los malos momentos.

Y finalmente, a mi familia, especialmente a mis padres, por el esfuerzo que han dedicado en mi formación como persona y su apoyo durante todo este periodo, sin el cual no podría haber concluido esta etapa de mi vida.

Muchas gracias a todos.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

CAPÍTULO 1: INTRODUCCIÓN

CAPÍTULO 2: SISTEMAS BRAIN COMPUTER INTERFACE

CAPÍTULO 3: ANÁLISIS DE LOS POTENCIALES EVOCADOS P300

CAPÍTULO 4: DISEÑO Y DESARROLLO DE LA APLICACIÓN

CAPÍTULO 5: EVALUACIÓN DE LA APLICACIÓN

CAPÍTULO 7: DISCUSIÓN

ÍNDICE DE TABLAS

CAPÍTULO 1: INTRODUCCIÓN

CAPÍTULO 2: SISTEMAS BRAIN COMPUTER INTERFACE

CAPÍTULO 3: ANÁLISIS DE LOS POTENCIALES EVOCADOS P300

CAPÍTULO 5: EVALUACIÓN DE LA APLICACIÓN

CAPÍTULO 6: RESULTADOS

CAPÍTULO 6: CONCLUSIONES Y LÍNEAS FUTURAS

CAPÍTULO 1

INTRODUCCIÓN

1.1. INTRODUCCIÓN A LA INGENIERÍA BIOMÉDICA

La Ingeniería Biomédica es una disciplina que aplica las técnicas de la ingeniería para modificar, controlar o simplemente comprender, sistemas biológicos que ayuden en el desarrollo de la medicina.

Algunos de los mayores descubrimientos se dieron ya a principios del siglo XX, siendo el más destacado antes de la primera guerra mundial, la invención del electrocardiograma (ECG) en 1903 por William Eindhoven. Después de la guerra otro de los grandes hitos de este campo fue el electroencefalograma (EEG), descubierto en 1929 por Hans Berger. A partir de este momento, la Ingeniería Biomédica se desarrolló rápidamente. Hasta los años 50 y 60 la actividad principal consistió en el desarrollo de instrumentos médicos para diagnóstico (radiografías por rayos X). Sin embargo, a medida que el conocimiento generado ha ido creciendo, las aplicaciones se diversificaron, incluyendo hoy en día numerosas áreas de investigación [1]. Algunos ejemplos son:

- Aplicación de ingeniería de análisis de sistemas a problemas biológicos.
- Detección, medición y monitorización de señales fisiológicas (biosensores e instrumentación biomédica).
- Diagnóstico de enfermedades gracias al procesamiento de señales bioeléctricas.
- Desarrollo de dispositivos terapéuticos y de rehabilitación.
- Desarrollo de dispositivos que reemplacen alguna función corporal (órganos artificiales).
- Análisis por ordenador de datos de pacientes para tomar decisiones clínicas (informática médica e inteligencia artificial).
- Análisis y procesamiento de imágenes médicas.
- Interpretación e ilustración gráfica de funciones fisiológicas
- Creación de nuevos productos biológicos.
- Desarrollo de biomateriales que puedan ser usados en el para sustituir estructuras biológicas.

Entre los objetivos de la Ingeniería Biomédica, en la actualidad, se encuentra: la reducción de la subjetividad en diagnósticos manuales, la mejora de la precisión en la toma de datos, el diagnóstico de enfermedades, mejora de los tratamientos actuales y el diseño de instrumentos médicos. Para ello, una parte clave es el procesado de las señales que emite el sistema biológico en estudio, las cuales, en el caso de que vayan a ser usadas con fines médicos, se conocen con el nombre de señales biomédicas [1].

1.2. SEÑALES BIOMÉDICAS

Tal y como hemos introducido en el apartado anterior, las señales biomédicas son aquellas utilizadas en el campo de la medicina. Por tanto, solo difieren de otro tipo de señales en el propósito de su adquisición. Estas señales tienen orígenes muy distintos, pudiendo hacerse varias clasificaciones atendiendo a distintos criterios como su descripción matemática, su origen o sus características.

Según su descripción matemática se clasifican como señales deterministas o señales aleatorias. Las señales deterministas son aquellas que pueden representarse matemáticamente de forma explícita mediante una expresión cerrada, de manera que sus posibles valores futuros son predecibles conociendo poca información sobre su pasado.

Las señales aleatorias no permiten conocer su valor futuro con exactitud ni siquiera conociendo todos sus valores pasados. Estas señales son muy abundantes en los procesos físicos, y en consecuencia el análisis de su estacionariedad y su ergodicidad cobra importancia.

La estacionariedad y la ergodicidad son dos propiedades que pueden cumplir las señales aleatorias. La primera de ellas exige que el comportamiento estadístico de la señal no cambie a lo largo del tiempo. Con respecto a la segunda, se dice que una señal aleatoria es ergódica si toda su aleatoriedad está presente en cualquiera de sus realizaciones, y por tanto, consideramos que una realización es representativa de todas las demás [2].

Según el origen de las señales biomédicas, se pueden clasificar como señales bioeléctricas, señales de bioimpedancias, señales bioacústicas, señales biomecánicas, señales biomagnéticas, señales bioquímicas o imágenes.

Las señales más conocidas se encuentran dentro de la clasificación de señales eléctricas, las cuales recogen la actividad eléctrica de distintas partes del cuerpo humano, como son el electroencefalograma (EEG) y el electrocorticograma (ECoG), que recogen la actividad cerebral; el electrooculograma (EOG), que recoge la actividad ocular; el electromiograma (EMG), que recoge la actividad muscular; y el electrocardiograma (ECG), que recoge la actividad del corazón. En la Figura 1-1 se ilustran tres de estas señales: EEG, ECG, EMG [3], mientras que la tabla 1-1 resume la clasificación de las señales bioeléctricas con diferentes parámetros típicos [1].

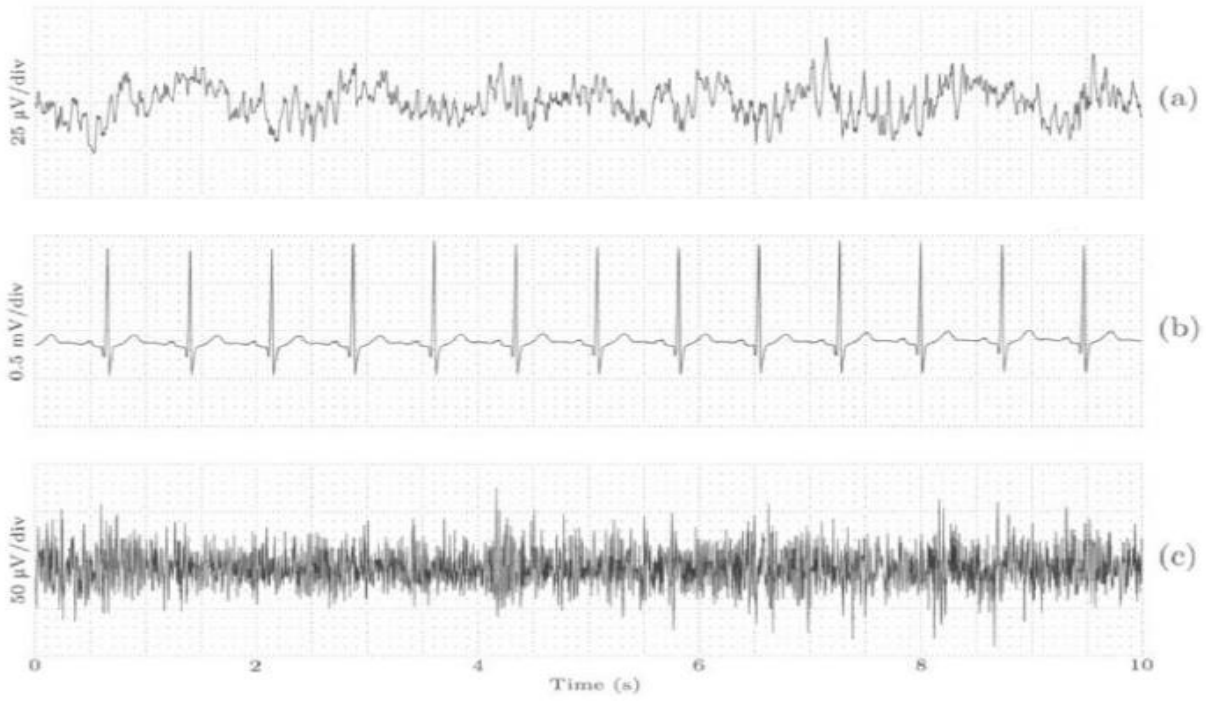

Figura 1-1. Ejemplos de las tres señales bioeléctricas principales recogidas sobre la superficie corporal: (a) un electroencefalograma (EEG), (b) un electrocardiograma (ECG), y (c) un electromiograma (EMG) [3].

Classification	Acquisition	Frequency range	Dynamic range	Comments
Bioelectric				
Action potential	Microelectrodes	100 Hz–2 kHz	10 μV–100 mV	Invasive measurement of cell membrane potential
Electroneurogram (ENG)	Needle electrode	100 Hz–1 kHz	5 μV–10 mV	Potential of a nerve bundle
Electroretinogram (ERG)	Microelectrode	0.2–200 Hz	0.5 μV–1 mV	Evoked flash potential
Electro-oculogram (EOG)	Surface electrodes	dc–100 Hz	10 μV–5 mV	Steady-corneal-retinal potential
Electroencephalogram (EEG)				
Surface	Surface electrodes	0.5–100 Hz	2–100 μV	Multichannel (6–32) scalp potential
Delta range		0.5–4 Hz		Young children, deep sleep and pathologies
Theta range		4–8 Hz		Temporal and central areas during alert states
Alpha range		8–13 Hz		Awake, relaxed, closed eyes
Beta range		13–22 Hz		
Sleep spindles		6–15 Hz	50–100 μV	Bursts of about 0.2–0.6 sec
K-complexes		12–14 Hz	100–200 μV	Bursts during moderate and deep sleep
Evoked potentials (EP)	Surface electrodes		0.1–20 μV	Response of brain potential to stimulus
Visual (VEP)		1–300 Hz	1–20 μV	Occipital lobe recordings, 200-msec duration
Somatosensory (SEP)		2 Hz–3 kHz		Sensory cortex
Auditory (AEP)		100 Hz–3 kHz	0.5–10 μV	Vertex recordings
Electrocorticogram	Needle electrodes	100 Hz–5 kHz		Recordings from exposed surface of brain
Electromyography (EMG)				
Single-fiber (SFEMG)	Needle electrode	500 Hz–10 kHz	1–10 μV	Action potentials from single muscle fiber
Motor unit action potential (MUAP)	Needle electrode	5 Hz–10 kHz	100 μV–2 mV	
Surface EMG (SEMG)	Surface electrodes			
Skeletal muscle		2–500 Hz	50 μV–5 mV	
Smooth muscle		0.01–1 Hz		
Electrocardiogram (ECG)	Surface electrodes	0.05–100 Hz	1–10 mV	
High-frequency ECG	Surface electrodes	100 Hz–1 kHz	100 μV–2 mV	Notchs and slus waveforms superimposed on the ECG

Tabla 1-1. Ejemplos de las tres señales bioeléctricas principales recogidas sobre la superficie corporal: (a) un electroencefalograma (EEG), (b) un electrocardiograma (ECG), y (c) un electromiograma (EMG) [1].

El magnetoencefalograma (MEG), destaca dentro de las señales magnéticas destaca, que registra la actividad funcional del cerebro mediante la captación de campos magnéticos, permitiendo estudiar las relaciones entre las estructuras cerebrales y sus funciones.

Otro tipo de señales son las señales bioquímicas, las cuales son el resultado de mediciones químicas de los tejidos vivos o de muestras analizadas en el laboratorio clínico.

También destacan las imágenes biomédicas, obtenidas mediante diferentes técnicas. La imagen por resonancia magnética funcional (fMRI), que permite localizar regiones cerebrales específicas; la tomografía por emisión de positrones (PET), que mide la actividad metabólica del cuerpo humano; o la retinografía, que permite visualizar el estado de la retina para diagnosticar enfermedades como la retinopatía diabética, son algunos ejemplos. En la Figura 1-2 se muestra una imagen PET en el estudio de la enfermedad de Alzheimer [4].

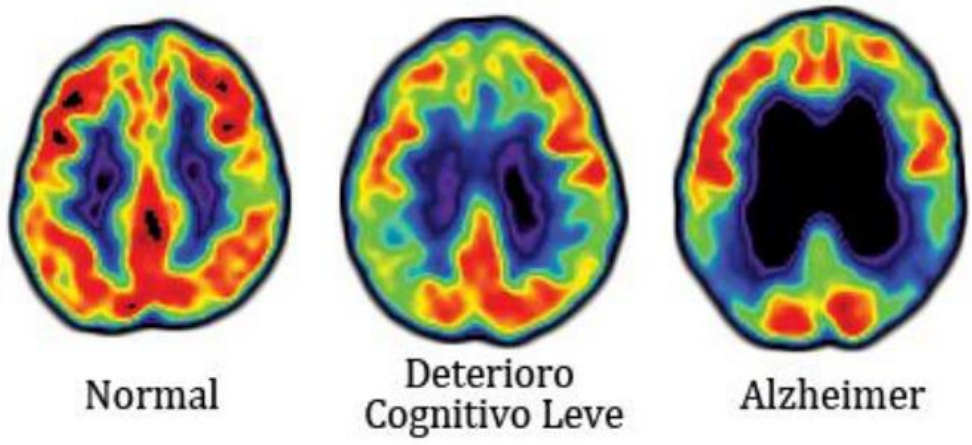

Figura 1-2. Imágenes PET que detectan la disminución de glucosa asociada con la enfermedad del Alzheimer. A la izquierda un sujeto sano, en el centro un sujeto con DCL (Deterioro Cognitivo Leve) y a la derecha un enfermo de Alzheimer [4].

Con el fin de poder extraer conclusiones de estas señales, es necesario realizar sobre ellas un proceso de análisis. En la actualidad, este procesado está implementado y automatizado mediante ordenadores, lo que permite analizar y realizar operaciones con gran cantidad de datos en tiempos pequeños.

Para este análisis, la señal tiene que sufrir una serie de trasformaciones que se aplicarán, generalmente, en tres etapas: obtención y registro de las señales (muestreo, cuantificación y digitalización de la señal), eliminación de artefactos y, finalmente, procesado de la señal (segmentación y filtrado de la misma). El diagrama de bloques típico utilizado para la adquisición de señales biomédicas se muestra en la figura 1-3 [1]:

El siguiente apartado está enfocado a profundizar en el EEG, el cual es una de las piezas fundamentales del desarrollo de la aplicación en la que se basa este texto.

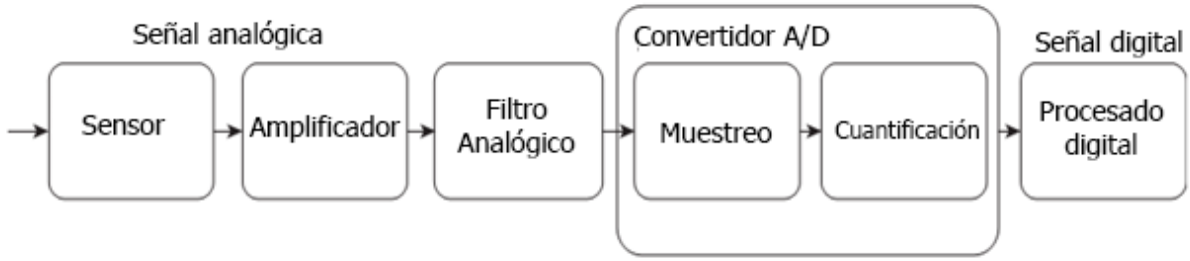

Figura 1-3. Diagrama de bloques típicamente utilizado para la adquisición de señales biomédicas [1]

1.3. ELECTROENCEFALOGRAMA (EEG)

El electroencefalograma (*electroencephalogram*, EEG) es una señal biomédica que registra la actividad eléctrica del cerebro de forma no invasiva mediante la implantación de una serie de electrodos sobre el cuero cabelludo.

Esta actividad es la agrupación de las señales eléctricas producidas por millones de neuronas al mismo tiempo, y se comporta de manera oscilatoria y repetitiva. Por ello, nos solemos referir a esta actividad conjunta como *ritmo*. Los ritmos suelen clasificarse según su rango frecuencial y su amplitud relativa.

Al ser una señal compuesta por la suma de las señales de neuronas individuales, la amplitud de la señal EEG está relacionada con el grado de sincronía con el cual interactúan. Una excitación sincronizada de un grupo de neuronas producirá una amplitud alta en la superficie del cráneo debido a la interferencia constructiva que se originará gracias a esa sincronización. Sin embargo, una excitación asíncrona de neuronas producirá una señal EEG irregular y de baja amplitud, debido a la interferencia destructiva de las actividades individuales. Este rango de amplitudes abarca desde unos pocos μV hasta 100μV [3].

El rango frecuencial de la señal EEG depende de la actividad que se esté realizando y suele estar relacionado con la amplitud de la misma. Un estado de somnolencia producirá bajas frecuencias y una sincronía entre las neuronas, causando una amplitud grande. El caso contrario se producirá cuando el cerebro está concentrado en alguna tarea determinada, desplazando el espectro a frecuencias más altas y generando una actividad asíncrona de las neuronas, forzando una amplitud más baja. En la Figura 1-4 se observan varios ritmos según el estado del sujeto [3]. El rango frecuencial comprende desde 0.5Hz (*banda delta*) hasta más de 30Hz (*banda gamma*).

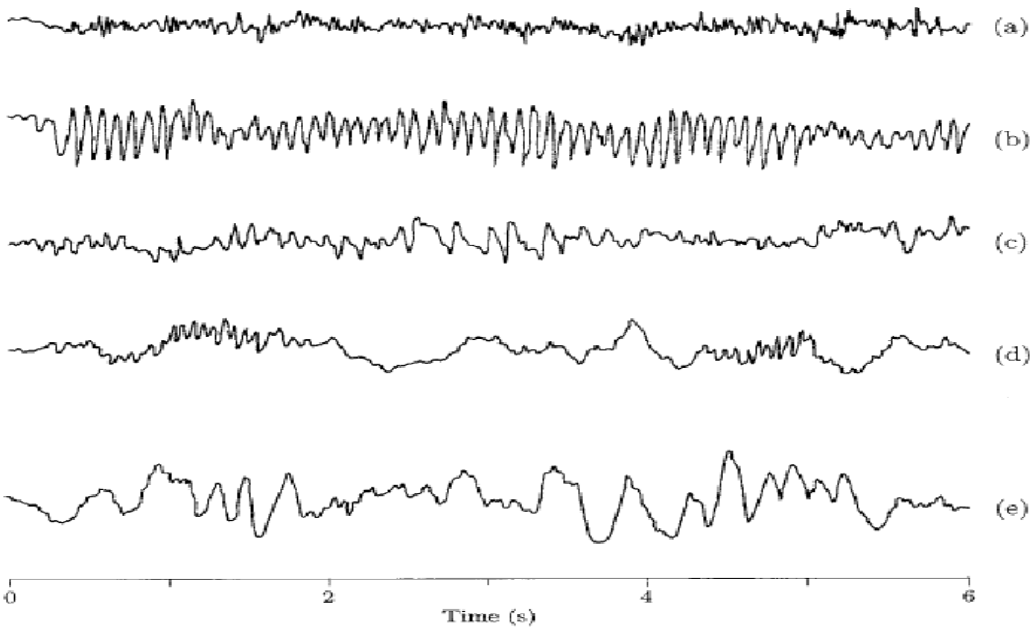

Figura 1-4. Ritmos electroencefalográficos observados durante el paso de vigilia a sueño profundo: (a) agitado, (b) relajado, (c) adormecido, (d) dormido, (e) sueño profundo. Se aprecia una disminución frecuencial a medida que el sujeto se duerme [3].

La adquisición de la señal se basa en la colocación de electrodos sobre la superficie del cuero cabelludo que recojan la señal eléctrica. La colocación de los electrodos debe atenerse a un estándar, conocido como el *sistema internacional 10/20*, aprobado por la Federación Internacional de Sociedades de Electroencefalografía y Neurofisiología Clínica [5].

Este estándar distribuye los electrodos diferenciándolos según las distintas zonas del córtex y el hemisferio al que pertenecen. El cráneo se divide en seis zonas: frontopolar (Fp), frontal (F), central (C), parietal (P), occipital (O) y temporal (T). Tomando como medida de referencia la distancia recorrida entre el entrecejo (nasión) y la nuca (inión) del sujeto, la sección central se encontrará a mitad de distancia y se separará de las secciones contiguas (frontal y parietal) un 20% de la distancia total. A su vez, estas dos secciones estarán separadas de las secciones frontopolar y occipital, respectivamente, por un 10% de la longitud total.

La nomenclatura de los electrodos está compuesta de la letra que identifica la sección junto a un subíndice que depende del hemisferio en el que se encuentren: para el hemisferio izquierdo se asignan números impares y para el hemisferio derecho se asignan números pares. En el caso de encontrarse en el centro de ambos hemisferios, se identifican mediante una letra "z" (*zero*) minúscula en lugar de un número. Con respecto a los electrodos de referencia, suelen ubicarse en los lóbulos de las orejas y se denominan A1 (izquierdo) y A2 (derecho). En la Figura 1-5 aparecen las distribuciones de los electrodos recomendadas según el sistema internacional 10/20.

Dado que muchas componentes de ruido, incluso las que son externas al cuerpo humano, aparecen de manera común en todas las señales EEG, hay que aplicar un filtro espacial antes de procesar la señal. La referencia en la oreja elimina muchas de estas componentes, no obstante, adicionalmente se utilizan canales bipolares, métodos de media común o filtros Laplacianos.

Los canales bipolares toman la diferencia de potencial entre dos electrodos no muy alejados. El método de referencia de media común (*Common Average Reference,* CAR) resta a la señal recogida por un electrodo la media de todo el conjunto de electrodos. Sin embargo, el filtro Laplaciano resta a la señal recogida por un electrodo las señales adyacentes.

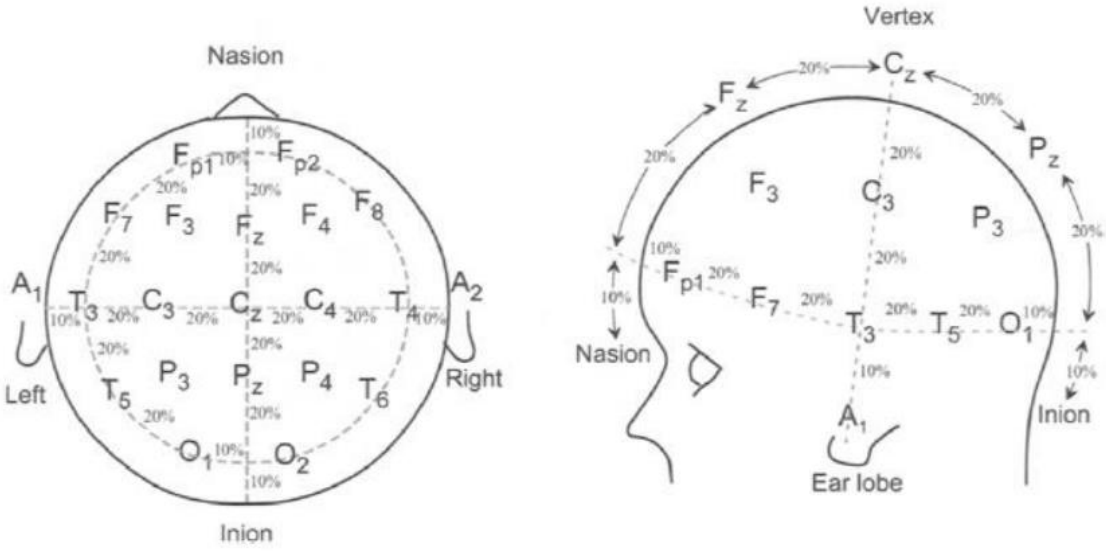

Figura 1-5. Distribución de los electrodos según el sistema internacional 10/20 [5].

Además de las componentes de ruido externas al cuerpo humano, la señal puede verse afectada por una serie de artefactos o señales diferentes a la de interés. En el EEG los artefactos más problemáticos son el movimiento ocular y el movimiento muscular [6].

1.4. BRAIN COMPUTER INTERFACE (BCI)

En general, todos los sistemas de transmisión de información necesitan de la interacción de señales nerviosas transmitidas mediante nervios periféricos con músculos u otros órganos para una comunicación satisfactoria.

Por ejemplo, cuando hablamos, espiramos aire desde nuestros pulmones por la laringe, donde se encuentran las cuerdas vocales. Las cuerdas vocales son un sistema móvil capaz de contraerse y cambiar de posición, lo que produce los diferentes sonidos que somos capaces de emitir. La señal cerebral que controla los músculos del sistema que forman las cuerdas vocales es, por tanto, la responsable de nuestra capacidad de hablar. Otro ejemplo puede ser la interacción con un ordenador mediante el ratón y el teclado, necesitando los músculos propios del manejo de estos aparatos para realizar la tarea.

Los sistemas BCI son un canal de transmisión de información que no se necesita la interacción del cerebro con nervios periféricos o músculos, y que permite a la persona que lo está utilizando transmitir sus intenciones al exterior directamente desde el cerebro. Las señales de control,

Figura 1-6. Sistema BCI basado en EEG. La señal es captada mediante electrodos colocados en el gorro, y mediante los cables se transmiten hasta el amplificador. Este dispositivo la amplifica y acondiciona para convertirla al dominio digital y la transmite al ordenador, donde se realiza el procesado [7].

producidas en el cerebro, son captadas generalmente mediante EEG, ya que es una manera no invasiva, barata y fácil de captación, aunque también existen otros métodos de los que hablaremos posteriormente en el capítulo 2.

Para identificar las intenciones del usuario y traducirlas en comandos, el dispositivo BCI analiza y procesa señales cerebrales. Hay varios tipos de señales que puede utilizar un sistema BCI, tales como los potenciales evocados visuales (VEP), los potenciales evocados P300, los potenciales corticales lentos (SCP) o los ritmos sensoriomotores μ y β. Estas señales de control pueden pertenecer a dos grupos: sistemas endógenos y sistemas exógenos.

Los sistemas BCI endógenos no necesitan de ningún tipo de estimulación externa para generar la actividad cerebral necesaria para clasificar las intenciones del usuario. Estos sistemas requieren entrenamiento para poder controlar satisfactoriamente la aplicación BCI. Habitualmente utilizan potenciales SCP y ritmos sensoriomotores μ y β.

En contrapartida, los sistemas BCI exógenos requieren una estimulación externa para producir la actividad cerebral necesaria para clasificar las intenciones del usuario. Estos sistemas no requieren entrenamiento alguno puesto que utilizan la respuesta natural del cerebro a diversos estímulos externos. Habitualmente utilizan potenciales VEP y potenciales evocados P300.

La señal debe ser tratada antes de poder traducir las intenciones del usuario. Este tratamiento está compuesto por dos etapas principales: adquisición de la señal y procesado de la señal. En la primera etapa la recoge la mediante electrodos situados sobre el cuero cabelludo. Posteriormente, en la segunda etapa, la señal se transmite a un sistema que trata de amplificarla y digitalizarla, además de eliminar aquellos artefactos que se han acoplado a la misma.

El procesado de la señal se compone de dos sub-etapas: la extracción y la clasificación de características. La extracción de características utiliza una serie de combinaciones y transformaciones sobre la señal EEG con el objetivo de obtener una información más discriminatoria de las intenciones del usuario. La clasificación transforma las características extraídas de la señal EEG en señales de control de un dispositivo o de una aplicación. El procesado de la señal se tratará con más detalle en el segundo y tercer capítulo.

El desarrollo de los sistemas BCI tiene como objetivo el aumentar la independencia y la capacidad de comunicación en usuarios con enfermedades neurodegenerativas o graves discapacidades. Los sistemas BCI pueden servir de gran ayuda para personas afectadas con afasia, apraxia, autismo, parálisis, lesión medular, amputaciones, distrofias neuromusculares o esclerosis múltiple o lateral amiotrófica, entre otras enfermedades. En consecuencia, las aplicaciones BCI abarcan campos tan dispares como el control de prótesis hasta el control mediante software de dispositivos electrónicos, como un ordenador o un móvil.

1.5. OBJETIVOS DEL TRABAJO DE FIN DE GRADO

El objetivo de este trabajo es desarrollar una aplicación que permita controlar un *Smartphone* con sistema operativo Android mediante BCI. Este objetivo general engloba otros más específicos y concretos que permitirán un desarrollo satisfactorio del proyecto, entre los cuales se encuentran:

- Estudiar los sistemas de adquisición de señales EEG que registran la actividad cerebral.

- Evaluar las diferentes posibilidades de diseño de un sistema BCI en función de las señales gracias a las cuales se traducen las intenciones del usuario, además de los métodos de procesado de señal necesarios para ese objetivo.
- Realizar una revisión del estado del arte de los sistemas BCI en dispositivos móviles desarrollados con anterioridad.
- Establecer los requisitos de la aplicación en cuestión, la señal de control a utilizar y las herramientas que se necesitan en el desarrollo y la arquitectura del sistema.
- Evaluar la aplicación una vez finalizada en cinco sujetos sanos en dos sesiones distintas.
- Discutir los resultados obtenidos y compararlos con otros estudios.
- Extraer conclusiones de los resultados obtenidos.

1.6. ESTRUCTURA DEL TRABAJO DE FIN DE GRADO

El trabajo de fin de grado está compuesto por ocho capítulos: introducción, sistemas *Brain Computer Interface*, análisis de la señal y potenciales evocados P300, diseño de la aplicación, evaluación de la aplicación, resultados, discusión y conclusiones y líneas futuras.

En el primer capítulo, introducción, se hace una breve explicación de la Ingeniería Biomédica y los diferentes campos que abarca y se describen los distintos tipos de señales biomédicas y su clasificación atendiendo a distintas características de la señal. Posteriormente, se detallan las características de la señal EEG y la adquisición de la señal mediante el estándar internacional 10/20. A continuación, se introduce el concepto de sistema BCI, y se establece, brevemente, la estructura del procesado de la señal. Finalmente, se detallan los objetivos y la estructura del presente trabajo.

En el segundo capítulo, sistemas Brain Computer Interface, se tratan, de manera más detallada, los aspectos sobre BCI previamente introducidos en el primer capítulo. En primer lugar, se estudian los distintos métodos para registrar la actividad cerebral, seguido por la enumeración y explicación de las distintas señales de control. Posteriormente se detalla el procesado de la señal EEG y se enumeran las distintas aplicaciones que han surgido gracias a los sistemas BCI. En último lugar, se comentan las limitaciones actuales de dichos sistemas.

En el tercer capítulo, análisis de la señal y potenciales evocados P300, se especifican las propiedades de los potenciales evocados P300 (señal de control utilizada en la aplicación web). Posteriormente, se introducen los distintos métodos utilizados para procesar la señal en la aplicación desarrollada, tanto para la extracción de características como para la clasificación de las mismas.

En el cuarto capítulo, diseño la aplicación, se describen los objetivos, la estructura y los procedimientos de desarrollo de la aplicación.

El quinto capítulo, evaluación de la aplicación, describe la metodología utilizada para evaluar la aplicación con 5 sujetos sanos y las tareas que realizaron.

En los siguientes capítulos se muestran, analizan y discuten los resultados obtenidos, además de compararlos con los estudios presentados anteriormente, y se evalúa el funcionamiento de la aplicación en general. Para terminar, se incluye un capítulo sobre conclusiones y líneas futuras de investigación a partir del presente trabajo.

CAPÍTULO 2

SISTEMAS BRAIN COMPUTER INTERFACE

2.1. INTRODUCCIÓN

El cerebro humano es una de las grandes incógnitas a las que se enfrenta la humanidad. Su funcionamiento ha sido siempre una pregunta sin resolver y es una de las fronteras que debe cruzar la ciencia en las próximas décadas. Una de las técnicas que mas ha ayudado a aumentar nuestra comprensión de este complejo órgano es sin duda el electroencefalograma (EEG), inventado en 1929 por Hans Berger, como se dijo en la introducción. Esta técnica permite investigar las ondas cerebrales de manera fácil, inocua, y barata, abriendo un mundo de posibilidades en el estudio del sistema nervioso central.

Durante el siglo XX se amplió el conocimiento de los tipos de ondas cerebrales, aplicándose sobre todo en medicina. Fue en 1977 cuando, por primera vez, se utiliza el término Brain-Computer-Interface, acuñado por el Dr. Jacques Vidal. En sus estudios, diseñó el primer sistema que funcionaba directamente con ondas cerebrales controlando, el movimiento de un cursor en dos dimensiones mediante potenciales evocados visuales (VEP) que se detectaban mediante EEG.

El sistema mostraba cuatro botones que se iluminaban con luces intermitentes de xenón a distintas frecuencias, los cuales representaban las cuatro direcciones básicas: arriba, abajo, derecha e izquierda. Cuando la mirada del usuario se centraba en uno de estos botones quedaban reflejados en el EEG los potenciales VEP que aparecen como respuesta a un estímulo visual de manera natural. De esta manera, el usuario era capaz de manejar, rígidamente, un cursor con el objetivo de escapar de un laberinto, por ejemplo. El esquema de éste primitivo e innovador sistema BCI aparece en la Figura 2-1 [8].

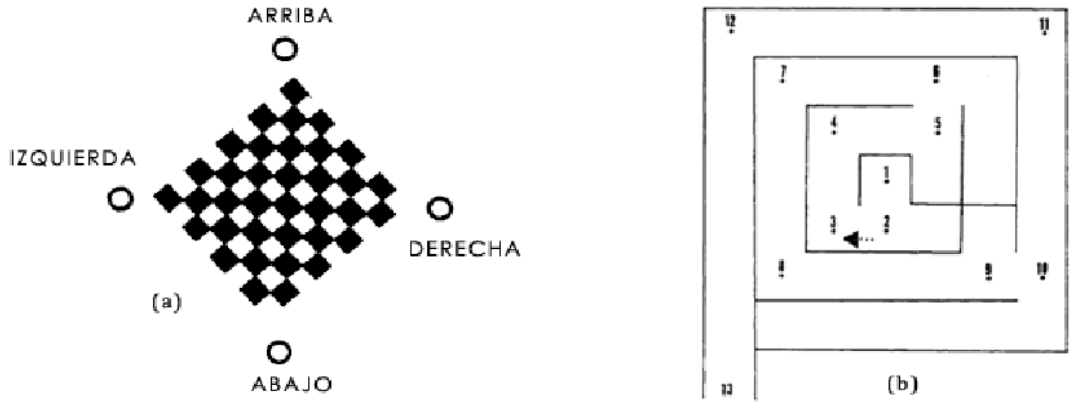

Figura 2-1. Esquema del primer sistema BCI controlado por VEP y desarrollado por Dr. Jacques Vidal en 1977. En (a) aparecen los botones que se iluminaban intermitentemente para controlar el cursor. En (b) aparece el laberinto del cual debían salir los usuarios moviendo el cursor [8].

A pesar de este primer ejemplo, no es hasta la década de los 90 cuando los sistemas BCI se desarrollan completamente, en gran parte debido elevado coste computacional que supone procesar la señal EEG en tiempo real.

En la actualidad, el interés por los sistemas BCI es creciente. Cada vez más universidades y empresas se interesan por esta tecnología y el avance en los últimos años es notable.

La principal aplicación de los sistemas BCI es la mejora en la calidad de vida de personas con grave discapacidad, ayudándolas a controlar ciertos aparatos y dispositivos que hasta entonces estaban reservados a personas sanas, aumentando su independencia. Pacientes con enfermedades neurodegenerativas como la esclerosis múltiple o la esclerosis lateral amiotrófica (ELA), o personas tetrapléjicas, podrían ver cambiar sus vidas por completo con la implementación de estos sistemas a su vida diaria. Algunos ejemplos, como veremos posteriormente, son las prótesis BCI, el control de un ordenador para navegar por internet, o un sistema domótico que permita el control, por ejemplo, de una televisión.

En la actualidad, la principal dificultad para la implementación de los sistemas BCI en la vida diaria de estas personas, es el rendimiento de las aplicaciones. Generalmente, para medir este rendimiento se recurre a la tasa de transferencia de información (*Information-Transfer-Rate,* ITR*)* medida en bits por minuto.

La ITR depende del número de opciones que el usuario puede elegir, el tiempo que se necesita para clasificarlas y el porcentaje de acierto de la clasificación. Hasta ahora, la mayoría de estudios consiguen una ITR de entre 30 y 90 bits/min, alcanzando los valores más altos siempre en condiciones controladas en laboratorio. Sin embargo, como veremos posteriormente, el rendimiento en aplicaciones BCI se ve muy afectado por las diferencias individuales de los sujetos y el entorno del experimento, habiendo muy pocos estudios al respecto con un gran número de usuarios.

2.2. MÉTODOS PARA REGISTRAR LA ACTIVIDAD CEREBRAL

Existen una gran cantidad de métodos para registrar la actividad cerebral en la actualidad, pudiéndose clasificar en invasivos y no invasivos.

Los sistemas invasivos, menos utilizados en la práctica con seres humanos, necesitan una intervención previa para implantar los electrodos que recogen la actividad bioeléctrica del cerebro. Estos sistemas implican un riesgo para la salud, a cambio de obtener mayor calidad en las señales registradas. En cambio, los sistemas no invasivos no requieren de ningún tipo de implantación previa para recoger la actividad cerebral, convirtiéndose así en los sistemas más utilizados en la práctica.

Para aplicaciones BCI, los requerimientos son la portabilidad del sistema, el bajo coste, métodos de adquisición de señal no invasivos y facilidad en el uso. Esto hace que la mayoría de los métodos queden descartados, habiendo solo unos pocos capaces de cumplir estas exigencias. A continuación, se describirán brevemente los métodos para registrar la actividad cerebral más utilizados y veremos más en detalle la posibilidad de usarlos en aplicaciones BCI [9]:

- **Imágenes de resonancia magnética funcional (*functional Magnetic Resonance Imaging*, fMRI).** La fMRI hace uso de los fenómenos de resonancia magnética para medir la respuesta hemodinámica del cerebro con fines clínicos y de investigación. La principal utilidad de esta técnica es la de facilitar información sobre la localización de las

funciones cerebrales mientras se realiza alguna acción. Normalmente se utiliza para pacientes que requieren cirugía cerebral. Es una técnica no invasiva y que tiene una gran resolución espacial, pero una mala resolución temporal. Esto hace que sea difícil de utilizar en aplicaciones en tiempo real.

- **Magnetoencefalografía (*Magnetoencephalography,* MEG).** La técnica MEG es un método no invasivo que registra la actividad funcional del cerebro mediante la captación de campos magnéticos, permitiendo investigar las relaciones entre las estructuras cerebrales y sus funciones. Esta técnica tiene una gran resolución, tanto espacial como temporal, pero la hace inaccesible en la mayoría de aplicaciones por el alto coste que tiene. Además, requiere de aislamiento magnético, lo que hace necesario el uso de una habitación especial.
- **Espectroscopía de infrarrojo cercano (*Near Infared Spectroscopy*, NIRS).** La espectroscopía estudia la interacción de la radiación electromagnética con la materia. En este caso se emiten fotones de luz cercanos al infrarrojo (NIR, longitudes de onda entre 780 ηm y 3.000 ηm) sobre la piel de la frente del paciente. Después de dispersarse por el interior del cuero cabelludo, cráneo y cerebro, parte de esos fotones vuelven a la piel gracias a los fenómenos de reflectancia espectral. Al medir la cantidad de fotones recibidos se extraen conclusiones sobre la respuesta hemodinámica del cerebro y los niveles de saturación de oxígeno y azúcar en sangre. Esta técnica, al igual que la fMRI tiene una pobre resolución temporal, lo que hace que no sea adecuada para aplicaciones en tiempo real.
- **Tomografía por emisión de positrones (*Positron Emission Tomography,* PET).** La técnica PET se basa en detectar y analizar la distribución tridimensional que adopta en el interior del cuerpo un radiofármaco administrado a través de una inyección intravenosa. Esta técnica se clasifica como una técnica no invasiva de diagnóstico *in vivo* por imagen capaz de medir la actividad metabólica del cuerpo humano.
- **Electroencefalografía (*Electroencephalography*, EEG).** La técnica EEG es un método no invasivo que registra la actividad bioeléctrica del cerebro mediante la colocación de una serie de electrodos sobre el cuero cabelludo. El número de electrodos utilizados para adquirir el EEG varía según su aplicación, siendo normalmente suficiente con 21 o menos, según el estandar 10/20 [5]. Sin embargo, existen otros estándares menos utilizados, pudiendo llegar a usarse mas de 300 electrodos [10]. Es una técnica no invasiva, de bajo coste y que no requiere un equipamiento difícil de desplazar. Por eso es la técnica más utilizada actualmente, a pesar de tener una resolución espacial limitada, y contaminación de la señal por artefactos (los cuales veremos posteriormente).
- **Electrocorticografía (*Electrocorticography*, ECoG).** La técnica ECoG persigue el mismo objetivo que la técnica EEG, sin embargo, ésta sí es una técnica invasiva. Se basa en implantar electrodos directamente sobre la superficie del córtex, proveyendo mayor resolución temporal y espectral a cambio de implicar un riesgo para el paciente.
- **Electrodos epidurales o intracorticales.** Los electrodos epidurales se basan en la colocación de electrodos de cristal con forma de cono hueco en neuronas aisladas. Estos electrodos ofrecen mayor resolución espacial y temporal que las técnicas anteriores, sin embargo, el tiempo que pueden permanecer implantados es limitado, además de que la neurona en cuestión podría morir u otra neurona podría pasar a realizar sus funciones, haciendo inservible la obtención de la señal a través de éste método.

Las técnicas expuestas anteriormente se pueden dividir en 2 grupos según el tipo de señales que utilicen, eléctricas o no.

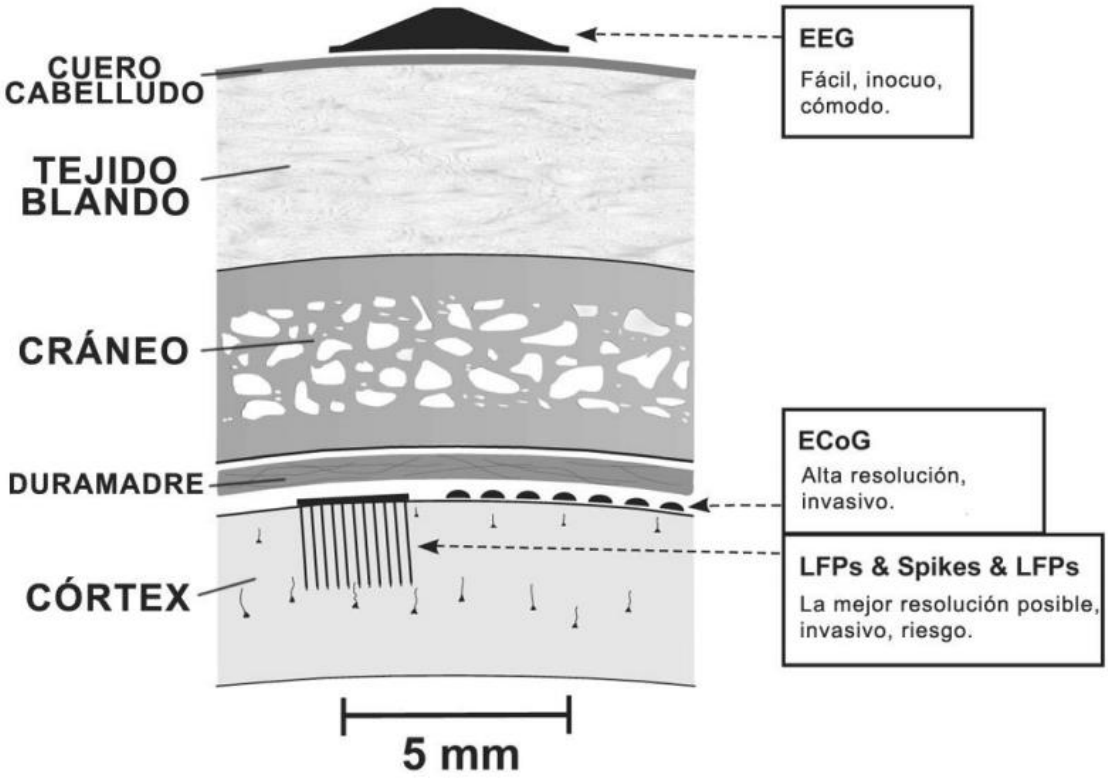

Figura 2-2. Localización de los electrodos de distintas técnicas que emplean señales eléctricas. [9]

Los 4 primeros métodos (fMRI, MEG, NIRS y PET), que no utilizan señales eléctricas, presentan varios inconvenientes. El fMRI y el NIRS tienen una pobre resolución temporal lo que los hace inadecuados para aplicaciones en tiempo real. Las técnicas MEG nos ofrecen la resolución adecuada, tanto temporal como espacial, sin embargo, su alto coste y su dificultad para realizarlas, la descartan como una de las soluciones ideales.

En cuanto a los métodos que utilizan señales eléctricas (EEG, ECoG y Electrodos epidurales), son más sencillos de implementar y su costo es menor. Sin embargo, el electrocorticograma y los electrodos epidurales necesitan de una intervención para la implantación de los sensores que reciben la señal eléctrica cerebral (figura 2-2 [9]). Esto hace que sean descartados para el desarrollo de la aplicación.

Concluimos que el único candidato posible es el EEG, un método de bajo coste, sencillo y que no necesita de un equipamiento difícil de transportar. Es una tecnología madura que ha sido muy estudiada desde su invención en 1929 y es sin duda la más utilizada en el campo de las aplicaciones BCI.

2.3. TIPOS DE SEÑALES DE CONTROL EN BCI

Una vez seleccionado el EEG como método para registrar la actividad cerebral en la aplicación BCI a desarrollar, es necesario elegir una señal de control específica que permita gobernar la aplicación. Estas señales se pueden generar de manera natural como respuesta a un estímulo

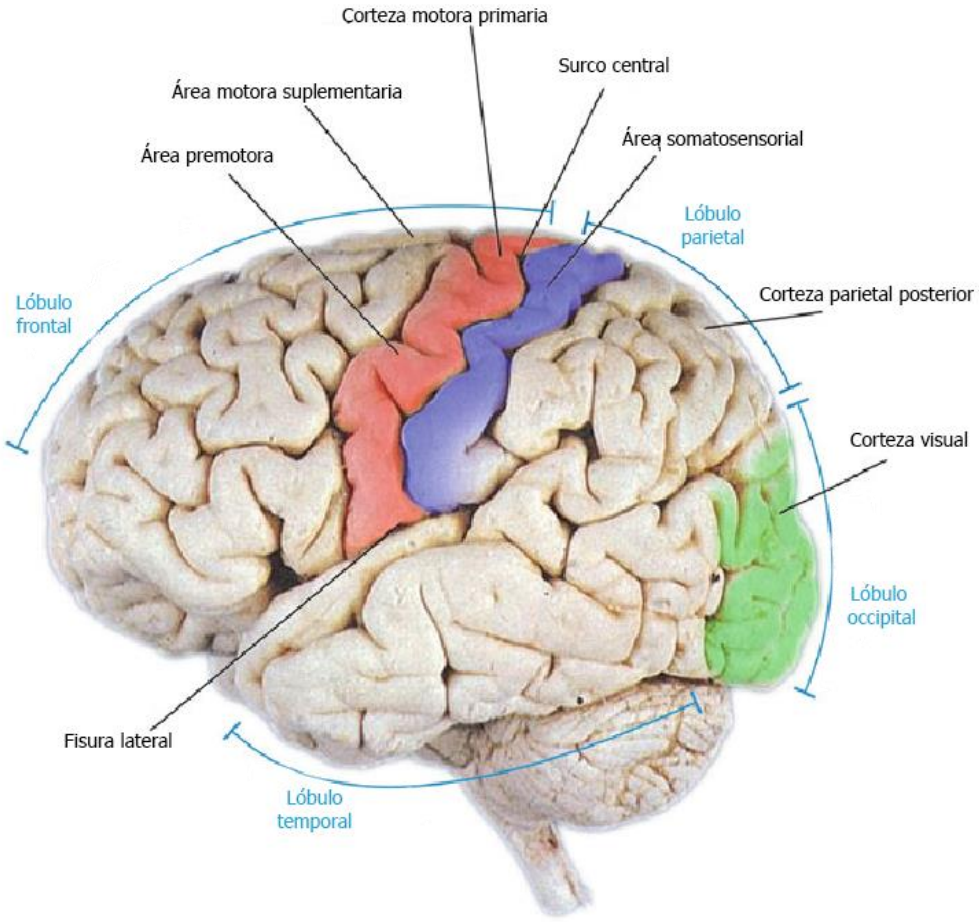

Figura 2-3. Regiones en las que se divide la corteza cerebral [7].

externo (señales de control exógenas), o se pueden generar de manera voluntaria con entrenamiento (señales de control endógenas).

Para registrar un EEG en los sistemas BCI habitualmente se utilizan 64 electrodos, sin embargo, para aplicaciones que usan señales de control concretas se puede reducir este número conociendo la localización específica donde se generan. Para comprender la localización de los electrodos a la hora de utilizar una aplicación BCI se incluye la figura 2-3 donde se indican las distintas regiones en las que se divide la corteza cerebral.

Las señales de control que trataremos a continuación son las siguientes: potenciales VEP, potenciales corticales lentos SCP, ritmos sensoriomotores μ y β, potenciales de neuronas corticales y potenciales evocados P300.

2.3.1. Potenciales evocados visuales

Los potenciales evocados visuales (*Visual Evoked Potentials,* VEP) son una respuesta del sistema nervioso central a una estimulación luminosa, en forma de diferencias de potencial a lo largo de la corteza visual.

La respuesta eléctrica que percibimos mediante EEG consiste en unas variaciones de potencial, resonantes con una fuente luminosa que parpadea a una frecuencia comprendida entre 4 y 20 Hz [11].

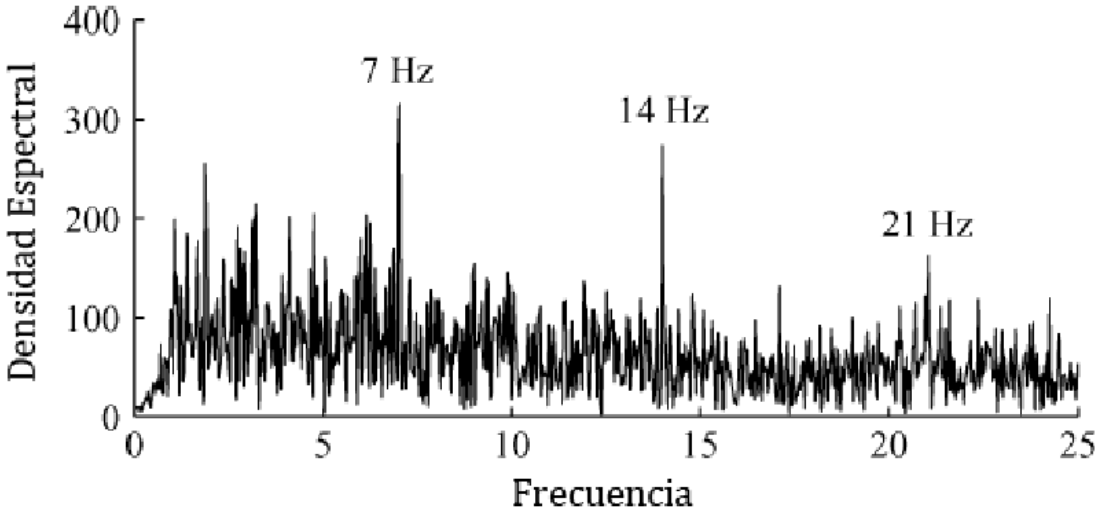

Figura 2-4. Espectro frecuencial de una señal EEG recogida durante una estimulación visual a 7Hz. El fenómeno de resonancia de los potenciales VEP produce picos en 7Hz y sus armónicos [11].

Este fenómeno de resonancia se utiliza en la práctica para detectar donde está mirando un sujeto. El método más utilizado es la presentación al usuario de una serie de fuentes luminosas parpadeando a distintas frecuencias. Cuando el usuario fija su atención en una en concreto, se genera el fenómeno de resonancia y el sistema mide la frecuencia con la que aparecen los VEP (en torno a 100 ms después de la iluminación) y determina la fuente que está mirando. La medida se realiza analizando en el dominio de la frecuencia la señal EEG, que tendrá varios picos en la frecuencia de la fuente y sus correspondientes armónicos. En la figura 2-4 aparece el espectro frecuencial de un usuario durante un experimento con una fuente luminosa de 7 Hz.

Esta señal de control es exógena, está provocada por una respuesta natural del cerebro a un estímulo luminoso y por tanto no requiere entrenamiento. Es posible así alcanzar grandes precisiones (de hasta el 90%) en pocas sesiones [12]. Además, Friman *et al* [11] demostró que el número de electrodos afecta en la precisión del sistema, concluyendo que los mejores resultados se obtienen cuando se utilizan pocos electrodos colocados correctamente en sobre la zona donde se generan los potenciales VEP.

El rango de aplicaciones es grande, pasando desde la selección de diversos comandos al control de un cursor en dos dimensiones.

2.3.2. Potenciales corticales lentos

Los potenciales corticales lentos (*Slow Cortical Potentials*, SCP) son variaciones de voltaje, de entre 0,5 y 10 segundos de duración, que se detectan en el EEG a bajas frecuencias. Los valores SCP negativos están asociados al movimiento y otras funciones que involucran un aumento de la activación cortical, mientras que valores SCP positivos están asociados a la ausencia de movimiento o reducción de la activación cortical [6].

En la Figura 2-5 se pueden observar las diferencias entre el aumento y reducción de la activación cortical. Estos potenciales se generan en el vértex, la zona superior de la cabeza, en torno a la unión del lóbulo frontal y parietal, asociada con el movimiento.

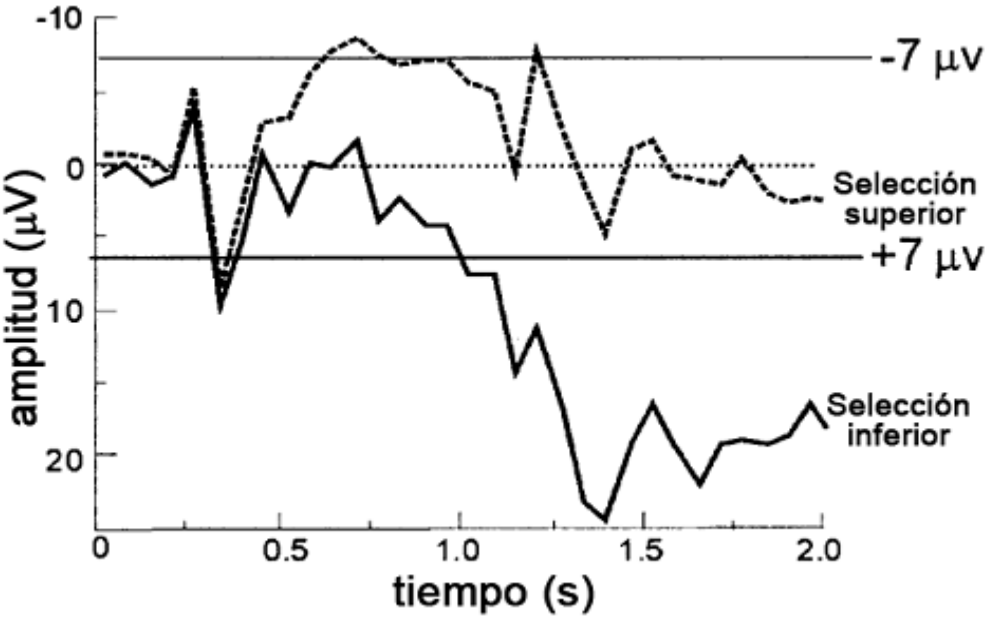

Figura 2-5. SCP recogidos durante una selección binaria [6]

Esta señal de control es endógena. Esto quiere decir que no es el resultado de una estimulación exterior, sino que es el propio sujeto el que la tiene que generar y para controlarla es necesario un entrenamiento.

La aplicación principal es el "dispositivo de interpretación de pensamiento" (*Thought Translation Device,* TTD). En primer lugar, esta aplicación realiza un registro basal de la actividad del EEG del usuario durante los dos primeros segundos y durante los dos siguientes, se determina si se ha producido o no un aumento de la actividad cortical. El dispositivo TTD utiliza un cursor que se mueve horizontalmente a velocidad constante, mientras que el movimiento vertical se controla a través de los SCP [13]. Por encima del movimiento del cursor aparecen una serie de selecciones, y cuando el usuario desea seleccionar alguna de ellas basta con generar un SCP que produzca que el cursor se desplace verticalmente sobre ella. Es, por tanto, un sistema de selección binaria.

El entrenamiento para controlar adecuadamente estas señales de control varía entre 1 y 5 meses (obteniendo precisiones en torno al 75%), constituyendo la principal desventaja de esta señal. Las tasas binarias conseguidas se sitúan en torno a los 15 bits por minuto, inferiores a las obtenidas con los potenciales VEP [14].

Los artefactos que más afectan a los potenciales SCP son la respiración del sujeto y, sobre todo, el movimiento ocular. En consecuencia, deberán ser eliminados en la etapa de adquisición de la señal para poder analizarla adecuadamente.

2.3.3. Ritmos sensoriomotores (ritmos μ y β)

Recordemos que la señal EEG tiene un comportamiento oscilatorio y repetitivo. Esta actividad se denomina *ritmo*.

En estado de vigila, cuando no se realiza acción motora o procesado de estímulo externo alguno, la mayoría de personas muestran una actividad en la banda de 8-12Hz. Esta actividad se

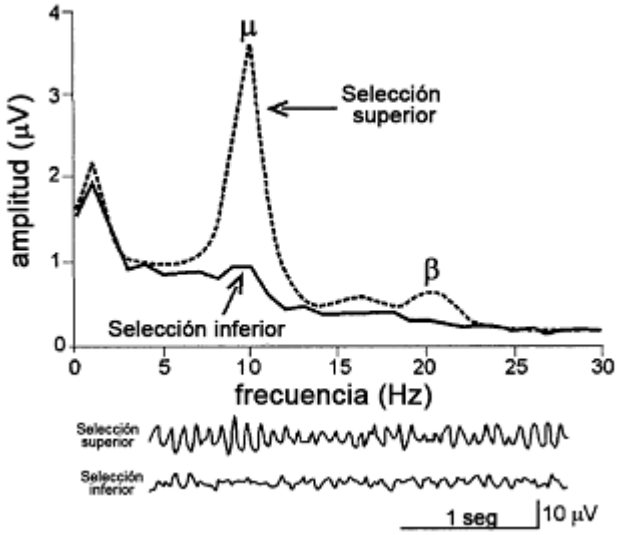

Figura 2-6. Ritmos μ y β. En la selección inferior se percibe una disminución de los mismos [6].

denomina *ritmo* μ cuando se produce sobre la zona somatosensorial o motora del córtex y *ritmo* α cuando se produce sobre la zona visual del córtex. Los electrodos que son capaces de obtener estos *ritmos* se deben colocar en estas zonas, ocupando las posiciones Cz, C3 y C4; y en ocasiones también F3, F4, O1 y O2 [6].

Los *ritmos* μ (de 8 a 12 Hz) están asociados con otros ritmos de mayor frecuencia llamados *ritmos* β (de 18 a 26 Hz) y, aunque alguno de los *ritmos* β se correspondan con los armónicos de los *ritmos* μ, otros son independientes, pudiendo usar ambos como dos características distintas o dos señales de control autónomas [6].

En la Figura 2-6 podemos apreciar una señal EEG con ritmos μ y β, que corresponde a una selección de un comando en una aplicación BCI.

La preparación del movimiento o la intención de realizarlo conllevan una disminución de los *ritmos* μ y β, particularmente contralateral al movimiento. Esta disminución se denomina ERD (*Event-Related Desynchronization,* Desincronización del Evento Relacionado) y ocurre pocos segundos antes de realizar el movimiento. Se conoce como "desincronización" porque la señal pierde su carácter oscilatorio, provocando una disminución de los picos frecuenciales que la caracterizaban [6].

En contraposición, una vez se realiza o se finaliza la intención de realizar un movimiento, se produce un aumento de los *ritmos* μ y β. Este aumento se denomina ERS (*Event-Related Synchronization,* Sincronización del Evento Relacionado) y determina el punto en el cual la señal volverá a obtener su carácter oscilatorio, provocando la recuperación de sus picos frecuenciales característicos. Adicionalmente, tanto los ERD como los ERS se localizan en el lado del cerebro contrario al movimiento que se desee realizar.

En la Figura 2-7 se recogen varios registros que muestran estos fenómenos[15]. En la gráfica superior, se muestran tres tipos de oscilaciones a distintas frecuencias, recogidas por el mismo electrodo (C3) sobre la zona sensoriomotora, cuando el usuario mueve el dedo índice de la mano.

Se puede apreciar un claro ERD en el *ritmo* μ que aparece, aproximadamente, 2,5 segundos antes de la realización del movimiento (marcado en línea roja, correspondiente a t=0s) y recupera el nivel inicial pasados unos segundos.

La actividad del *ritmo* β presenta un ERD de corta duración cerca del inicio del movimiento, seguido por un ERS con un máximo al segundo de ejecutar el movimiento. Con respecto a las oscilaciones de la *banda* γ (36-40Hz), presentan un ERS antes de realizar el movimiento, pudiendo proveer información y ser útiles en ocasiones, sin embargo, estas oscilaciones de alta frecuencia no están presentes en todos los seres humanos.

En la figura 2-7 se muestra también el EEG en cada electrodo para el mismo movimiento. Se pueden observar claramente los ERD en las posiciones centrales de los electrodos (C3, C4 y Cz) antes de realizar el movimiento y los ERS inmediatamente después del mismo sobre los electrodos de la zona visual del córtex (*ritmo* α, P3, P4, Oz).

Se ha demostrado que no solamente la ejecución del movimiento genera los ERD y los ERS, también la imaginación del movimiento o la visualización del mismo provocan los mismos cambios en los *ritmos* μ y β. Esta es la razón por la cual un sistema BCI basado en ritmos sensoriomotores se trata como un sistema independiente, en el cual la capacidad del usuario para fijar la vista en un punto concreto o mover una articulación no se necesita, ya que simplemente es necesario imaginar el movimiento para generar la señal de control [16].

Al igual que los SCP, la señal de control generada gracias a los ritmos sensoriomotores es endógena y, por tanto, requiere entrenamiento para poder ser controlada de manera adecuada. Con un entrenamiento de unas 6 horas se pueden llegar a alcanzar precisiones por encima del 80% y tasas binarias de 20-25 bits por minuto [6].

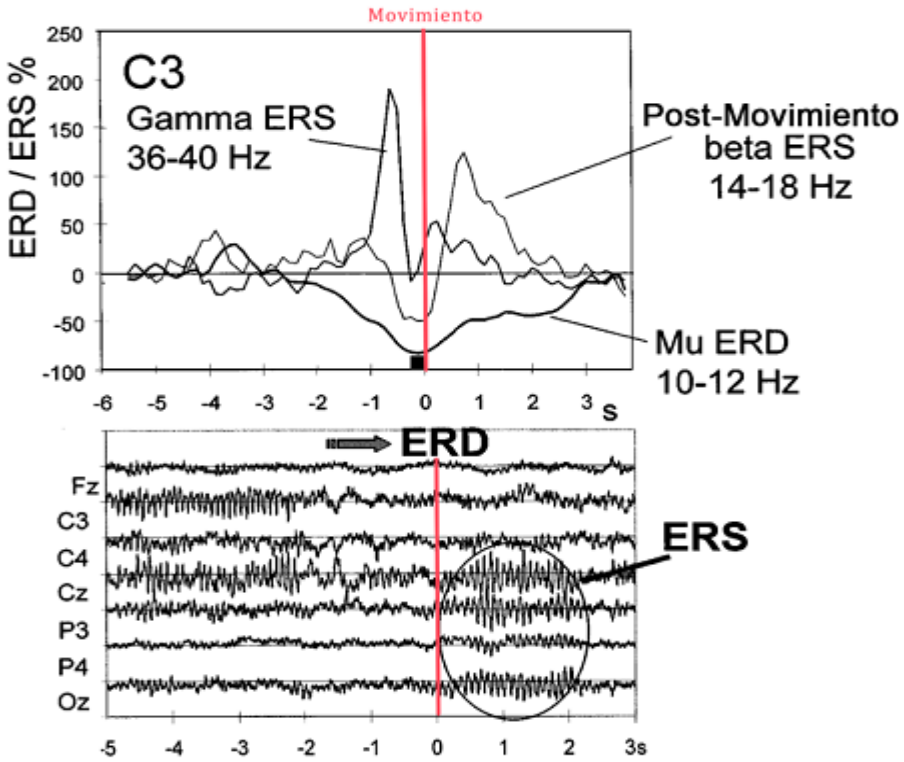

Figura 2-7. Señal EEG registrada antes y después de realizar un movimiento con el dedo índice de la mano donde se aprecian los ERD y ERS. El panel superior se corresponde al electrodo C3 [6].

2.3.4. Potenciales de neuronas corticales

Los potenciales de neuronas corticales se corresponden con la actividad de neuronas aisladas, recogida recogen mediante electrodos epidurales o intracorticales alojados dentro del cráneo. Este método es invasivo, ya que requiere cirugía para la implantación de los electrodos, lo que impide la investigación exhaustiva en humanos. En la Figura 2-8 aparece un esquema sobre la implantación de un electrodo intracortical.

Diversos estudios con monos han demostrado que éstos son capaces de aprender a controlar la activación de neuronas aisladas, así que se espera que los humanos sean capaces también. Sin embargo, la evaluación de esta especulación se retrasó debido a la ausencia de los electrodos adecuados y a que no era posible registrar la actividad neuronal de forma estable a largo plazo (normalmente no superan el año). De hecho, la neurona que está siendo monitorizada por el electrodo en cuestión puede morir o relevar su función a otra neurona distinta, causando que el electrodo se vuelva inútil [6].

Las utilidades de esta señal de control podrían ir desde controlar un cursor en 1D hasta seleccionar comandos o letras en una pantalla. También podría llegar a controlarse un cursor en 2D si se logra el control de la actividad EMG residual (se obtuvieron tasas de 15bits por minuto y precisiones del 67% en animales) [6]. Sin embargo, no se puede generalizar con total seguridad para los humanos y debido a la escasez de experimentos con los mismos no se puede determinar que su uso sea satisfactorio.

Dado que los experimentos con animales se efectuaron cuando éstos realizaron los movimientos oportunos, no se conoce si se establecerán los mismos patrones en la actividad neuronal cuando no se ejecute el movimiento debido a la incapacidad del usuario para producirlo.

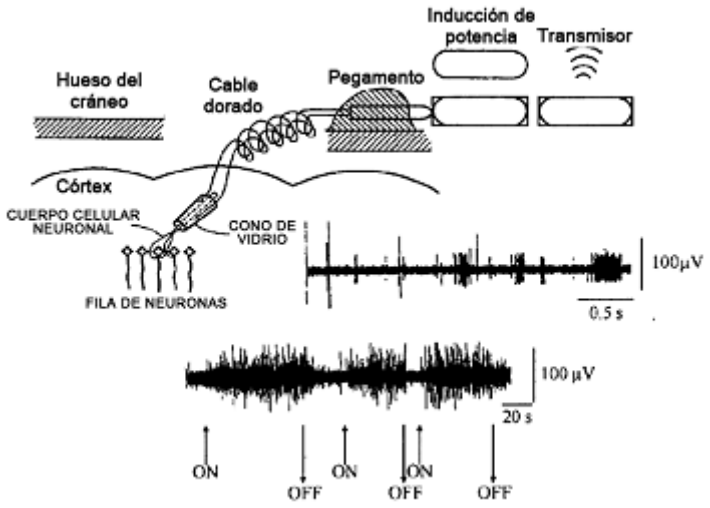

Figura 2-8. Esquema que muestra la implantación de electrodos epidurales o intracorticales [6].

2.3.5. Potenciales evocados P300

Los potenciales evocados P300 son deflexiones positivas de voltaje localizadas en la zona parietal del córtex como respuesta a un estímulo que aparecen en torno a 300ms después de que éste se produzca. Por tanto, el potencial P300 es una señal exógena (no requiere entrenamiento) provocada por una respuesta natural del cerebro ante un estímulo, ya sea visual, auditivo o somato-sensorial unos 300 ms después. Los electrodos que se emplean para recoger este potencial normalmente se sitúan en la región centroparietal del córtex, típicamente en Cz, Fz y Pz [6].

Para que la señal P300 se genere con suficiente amplitud y pueda ser detectada, se deben dar 4 condiciones [17]:

- La secuencia de estímulos debe ser aleatoria. Si el usuario es capaz de predecir el momento en el que se va a dar el estímulo, la amplitud de la señal es menor.
- Una regla de clasificación que separe las series de eventos en dos categorías.
- La tarea que realice el usuario debe usar la regla anterior.
- Los estímulos deben de ser infrecuentes. Cuanto más continuos sean, menor será la amplitud del P300.

La aparición de los potenciales evocados P300 se provoca de acuerdo al paradigma *odd-ball*, el cual trata de presentar un estímulo deseado de forma infrecuente camuflado entre estímulos frecuentes. Es decir, el potencial evocado P300 se produce cuando la opción que pretende elegir el usuario se encuentra entre muchas otras opciones que no interesan. En consecuencia, se concluye que cuanto más improbable sea el estímulo deseado, más probable será que aparezca el potencial evocado P300 y que su amplitud sea mayor.

La aplicación más frecuente de este paradigma, desarrollada por primera vez por Emanuel Donchin *et al* [18], se basa en presentar una matriz cuyas celdas sean las opciones a seleccionar: habitualmente letras, números o comandos de una aplicación.

En cada intento se iluminan aleatoriamente las filas y las columnas de forma que, tras un número determinado de iluminaciones, cada fila y columna se haya iluminado una vez. El usuario, fijándose en la celda que desea seleccionar, provocará dos potenciales evocados P300 cuando se iluminen la fila y la columna que contienen dicha celda. Promediando la respuesta para cada elemento de la matriz y detectando el potencial P300 con mayor amplitud se determina cuál es la celda seleccionada por el usuario. El resultado de este tipo de aplicaciones se puede observar en la Figura 2-9 [6].

Dado que se trata de una señal exógena y no requiere entrenamiento, se pueden alcanzar precisiones muy altas en intervalos muy cortos de tiempo (90% de precisión con 4 selecciones por minuto en sujetos de control). Sin embargo, su rendimiento puede verse afectado a lo largo del tiempo, cuando el usuario se acostumbra a los estímulos que generan los P300, pudiendo dejar de ser notables.

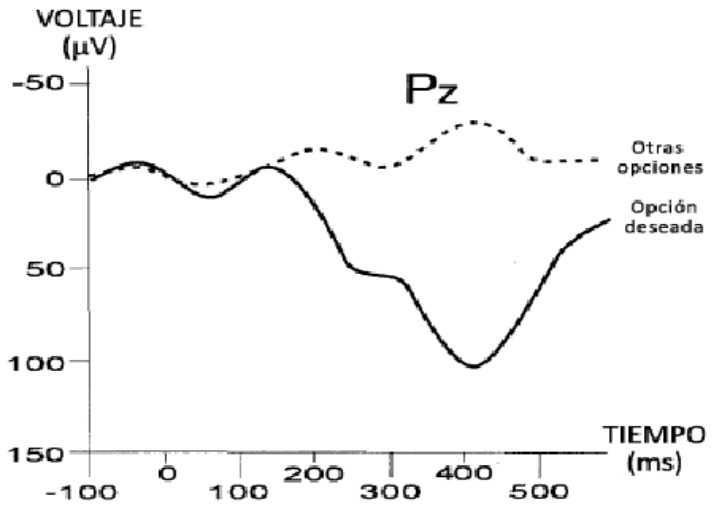

Figura 2-9. Potencial evocado P300 recogido sobre el área centro-parietal del córtex mediante la técnica *odd ball* con una matriz luminosa [6].

2.4. ETAPAS DE LOS SISTEMAS BCI

El procesado de la señal EEG puede dividirse en tres etapas principales, comunes con el tratamiento digital del resto de señales biomédicas: adquisición de la señal, procesado de la señal y aplicación.

2.4.1. Adquisición de la señal

En esta etapa se recoge la señal y se prepara para su posterior análisis. Este proceso de acondicionamiento incluye amplificación, conversión al dominio digital mediante un CAD, y eliminación de artefactos e interferencias que la degraden.

Como se comentó en el capítulo de introducción, la señal EEG se obtiene mediante la colocación de electrodos de alta sensibilidad en la superficie del cuero cabelludo según el *sistema internacional 10/20.* Normalmente, para reducir la impedancia entre la piel de la superficie de la cabeza y el electrodo se utiliza un gel conductor.

Posteriormente, la señal se amplifica con un amplificador de bajo de ruido y se convierte al dominio digital.

A continuación, se filtra para eliminar los artefactos que puedan afectar en su procesado. Los principales artefactos de una señal EEG son las señales producidas por pestañeos, contracciones musculares y movimientos oculares (EOG) [17]. En menor medida y dependiendo de las condiciones, la señal cardiaca (ECG) también puede afectar. A continuación, se profundiza el efecto de estos artefactos:

- **Movimientos oculares y pestañeos.** La actividad eléctrica producida por el movimiento ocular es claramente visible en la señal EEG debido a su gran intensidad. El EOG refleja la diferencia de potencial que aparece entre la córnea y la retina durante el movimiento ocular. Esta diferencia de potencial es proporcional al ángulo de visión. Cuanto más

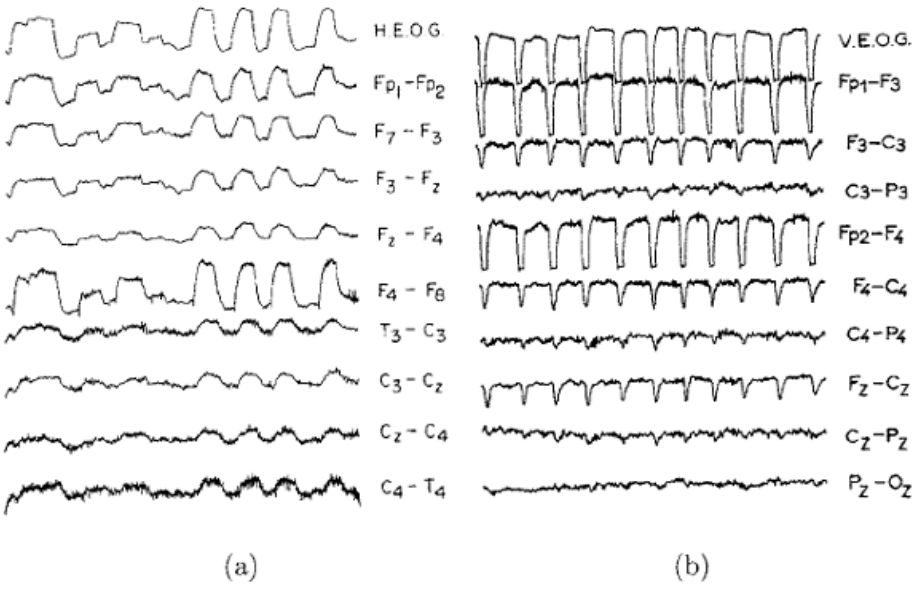

Figura 2-10. Artefactos en la señal EEG causados por (a) el movimiento ocular y (b) pestañeos repetitivos y voluntarios [17].

cerca se encuentre el electrodo de los ojos mayor será la distorsión del EEG. Por tanto, la región que se ve más afectada es la frontopolar del córtex.
Los pestañeos también afectan a los electrodos de esta región, induciendo una señal de mayor frecuencia que el EOG y de cambios mas abruptos.

En términos de procesamiento, el artefacto causado por el movimiento de los ojos y los pestañeos se puede reducir obteniendo una señal de referencia en la que solamente aparezca la molestia, colocando electrodos cerca de los ojos. De esta manera se podría eliminar el artefacto, por ejemplo, con un filtrado adaptativo.

En la figura 2-10 se observan dos estos efectos.

- **Actividad muscular cercana.** Otro artefacto común lo causa la actividad eléctrica muscular, sobre todo, de músculos cercanos. Esta actividad se encuentra normalmente cuando el sujeto está en estado de vigilia y se encuentra tragando, haciendo muecas, ciñendo el cejo, masticando, hablando, o, en resumen, realizando cualquier actividad que implique mover los músculos faciales o la mandíbula mientras que se reduce drásticamente cuando el sujeto se encuentra relajado o dormido.

  El efecto que produce depende de la fuerza de la contracción muscular, pudiendo generar irregularidades de baja amplitud si la contracción es débil o altas irregularidades que se asemejan a ruido blanco cuando la contracción es fuerte.

  Este artefacto puede observarse en la Figura 2-11. En términos de procesamiento, la actividad muscular es un artefacto mucho más molesto que el movimiento ocular, puesto que solapa la *banda* β dentro del rango entre 15 y 30Hz. Esta desventaja está agravada por el hecho de que es imposible adquirir una señal de referencia que solamente contenga la actividad muscular interferente, a diferencia del EOG.

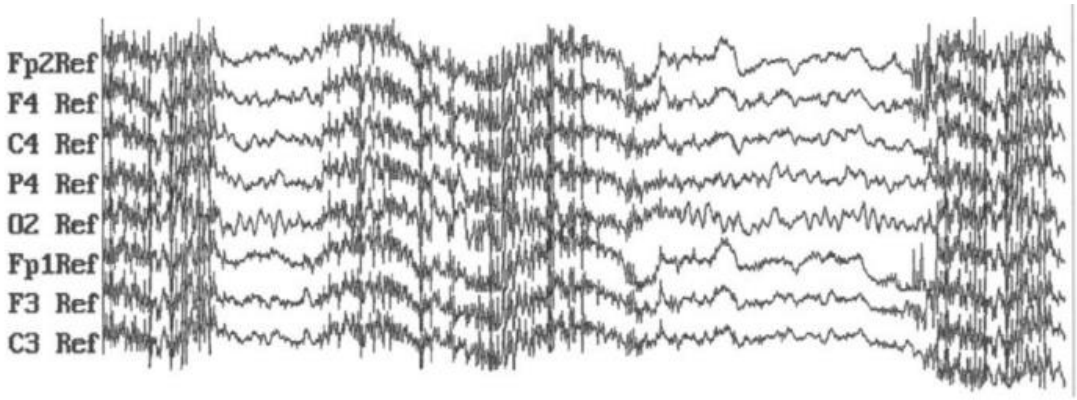

Figura 2-11. Registro de 5 segundos de duración de la señal EEG contaminada por episodios intermitentes de artefactos electromiográficos (EMG) [17].

- **Actividad cardiaca.** El ECG puede llegar a interferir con la señal EEG en algunas personas. Aunque la amplitud del ECG normalmente es baja en el cráneo en comparación con el EEG, puede aparecer en determinados electrodos y en determinados cuerpos, por ejemplo, en personas bajas con cuellos cortos.
  Debido a la naturaleza repetitiva y regular que caracteriza al ECG, la eliminación del mismo resulta sencilla.

Otro efecto a tener en cuenta es la interferencia que produce la red eléctrica, que en España tiene una frecuencia de 50 Hz. Es importante tenerla en cuenta y filtrarla correctamente, ya que puede producir efectos negativos al digitalizar la señal.

En la figura 2-12 vemos un ejemplo del efecto que produce en una señal EEG con un ancho de banda de 40 Hz muestreada a 80 Hz sin filtrar la frecuencia de operación de la red eléctrica. El pico en 30 Hz es producido por el *aliasing* que se produce, contaminando el ancho de banda de la señal [1]. Esto tiene una fácil solución si se filtra correctamente la señal antes de muestrearla. Sin embargo, si el ancho de banda de la señal de interés contiene la frecuencia de operación de la red eléctrica deberá considerarse un procesado mas complejo.

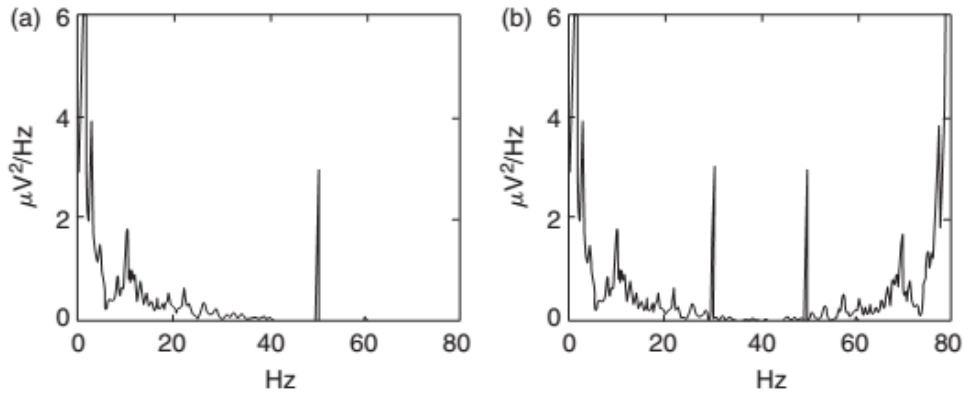

Figura 2-12. Densidad espectral de potencia de una señal EEG (originalmente limitada en banda hasta 40 Hz). La presencia de un pico en 50 Hz en la señal original (a) causa aliasing en la señal muestreada (b) si la frecuencia de muestreo es de 80 Hz [1].

2.4.2. Procesado de la señal

La segunda etapa es el procesado de la señal, a su vez subdividido en la extracción de características y la traducción de las mismas.

a) Extracción de características

La etapa de extracción de características se basa en obtener información útil de la señal EEG para el control del sistema BCI. Esta información se denomina característica y será lo que permita diferenciar entre distintas patologías o clasificar las intenciones del usuario una vez sea analizada.

El objetivo, por tanto, es maximizar la SNR entre la señal de control y el ruido basal. Para ello habrá que realizar un procesado de señal para eliminar los artefactos ya mencionados. Además de estos, existen otros factores que pueden afectar a las características de la señal del usuario. Entre ellos se encuentran la motivación, la intención, la frustración, la fatiga, etc. El buen funcionamiento del sistema BCI depende de la adaptabilidad del sistema al usuario, así como de la selección de las técnicas de procesado de señal adecuadas [6].

Aunque en otros tipos de sistemas que tratan también las señales biomédicas, la cancelación de artefactos y la extracción de características están totalmente diferenciadas, en los sistemas BCI muchos autores clasifican técnicas que se pueden emplear para cancelación de artefactos como técnicas de extracción de características, debido a que su salida usualmente se convierte en la entrada del clasificador.

En la Tabla 2-1 se muestran una serie de técnicas de extracción de características empleadas con anterioridad en los sistemas BCI [9]. Los principales métodos se dividen en dos grandes grupos: los que trabajan con señales en tiempo o frecuencia y los que trabajan con señales en el espacio. Lo más común es utilizar un método de cada tipo, tratando las dimensiones temporal y espacial de manera independiente, o emplear un solo método espacio-temporal.

<table>
<tr><th>Tiempo-Frecuencia</th><th>Espacio</th><th>Tiempo-Espacio</th><th>Modelos Inversos</th></tr>
<tr><td>Transformada de Fourier</td><td rowspan="2">Filtro Laplaciano</td><td rowspan="3">Análisis en Componentes de Tiempo y Espacio</td><td rowspan="4">EEG a ECoG</td></tr>
<tr><td rowspan="2">Transformada Wavelet</td></tr>
<tr><td rowspan="2">Análisis de Componentes Principales (PCA)</td></tr>
<tr><td rowspan="2">Modelos Autorregresivos (AR)</td><td rowspan="3">Modelos Autorregresivos Multivariable</td></tr>
<tr><td rowspan="2">Análisis de Componentes Independientes (ICA)</td><td rowspan="5">EEG a Dipolos Fuente</td></tr>
<tr><td rowspan="2">Filtrado Paso-Banda</td></tr>
<tr><td rowspan="2">Patrones Espaciales Comunes (CSP)</td><td rowspan="3">Coherencia</td></tr>
<tr><td rowspan="2">Filtro Adaptado</td><td rowspan="2">Amplitudes</td></tr>
<tr><td rowspan="2">Filtro de Kalman</td></tr>
<tr><td rowspan="2">Proporciones y Diferencias</td></tr>
<tr><td>Detección de Pico</td></tr>
</table>

Tabla 2-1. Clasificación de los métodos empleados para la Extracción de Características [9].

Los métodos aplicados en el dominio temporal destacan por su rapidez y el reducido coste computacional que requieren. Estos métodos suelen utilizarse en aplicaciones BCI a tiempo real y la elección de los mismos depende en gran parte de la señal de control utilizada y de los artefactos que la degraden. Por ejemplo, un simple promediado sincronizado aumenta de manera significativa la capacidad de detección de los potenciales evocados P300 o los VEP, como puede verse en la figura 2-13.

Sin embargo, los métodos que operan en el dominio frecuencial requieren una transformación de la señal, convirtiéndose así en métodos que requieren más coste computacional que los temporales. De esta manera, para aplicaciones a tiempo real interesan aquellos métodos espectrales que operan con segmentos de la señal relativamente cortos.

Los métodos espaciales también dependen de la señal de control que utiliza el sistema y, concretamente, de lo localizada que esté su fuente en la corteza cerebral. Por consiguiente, el método más adecuado para detectar los *ritmos* μ y β , que se encuentran bastante localizados en el córtex, no será el mismo que para detectar los potenciales SCP o los P300, que se encuentran más distribuidos a lo largo del mismo.

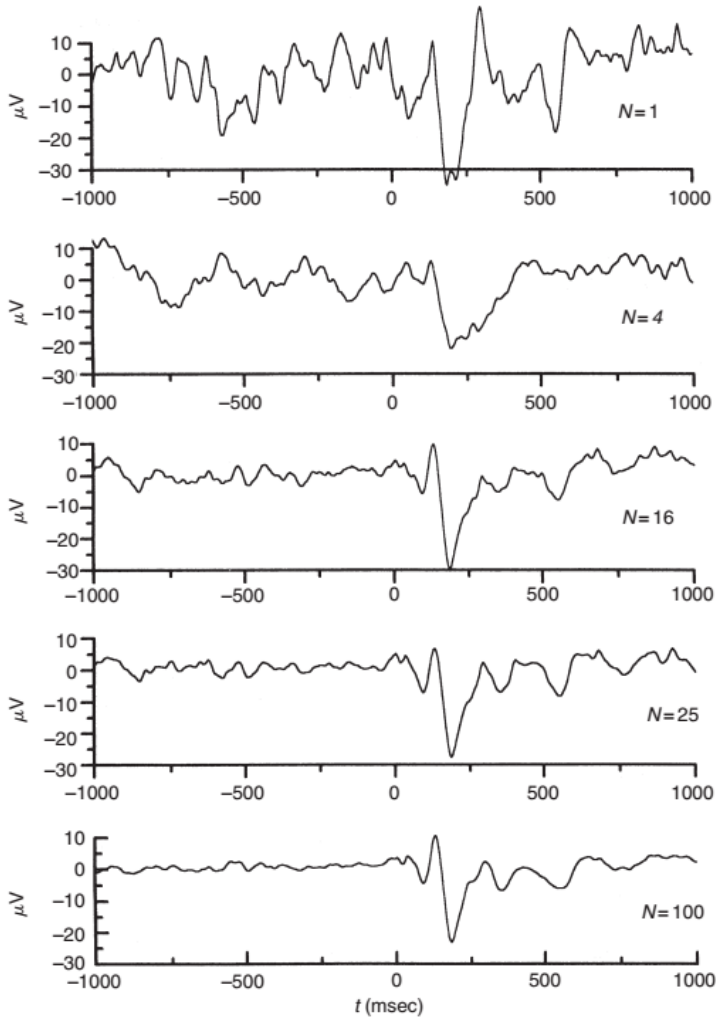

Figura 2-13. Efecto de un promediado sincronizado de varias muestras de una señal EEG que contiene un potencial evocado (EP). Se puede observar como el ruido se reduce de manera muy notoria, permitiendo la detección del potencial con mucha precisión [1].

Una medida útil para realizar un análisis *offline* de los métodos espaciales es el coeficiente de determinación r^2, que básicamente consiste en la correlación de los datos elevada al cuadrado.

En la Figura 2-14 se muestra el resultado de este coeficiente para intentar detectar los *ritmos* μ y β sobre el electrodo C3 empleando cuatro métodos espaciales diferentes, siendo éstos el filtro Laplaciano con dos distancias distintas de separación de los electrodos, el método de referencia de media común (CAR) y el uso del electrodo C3 con una única referencia en la oreja. Tal y como se puede observar, el filtro Laplaciano con una distancia entre electrodos de 6 cm o el filtrado CAR proporcionan mejores resultados que los otros dos métodos analizados [6].

Además de los cuatro métodos espaciales comparados, es frecuente en los sistemas BCI utilizar PCA (*Principal Component Analysis,* Análisis de Componentes Principales) o ICA (*Independent Component Analysis,* Análisis de Componentes Independientes), útiles para realizar una combinación lineal de varios canales y seleccionar la información más relevante. Por esta razón también son conocidos como métodos de reducción de dimensionalidad.

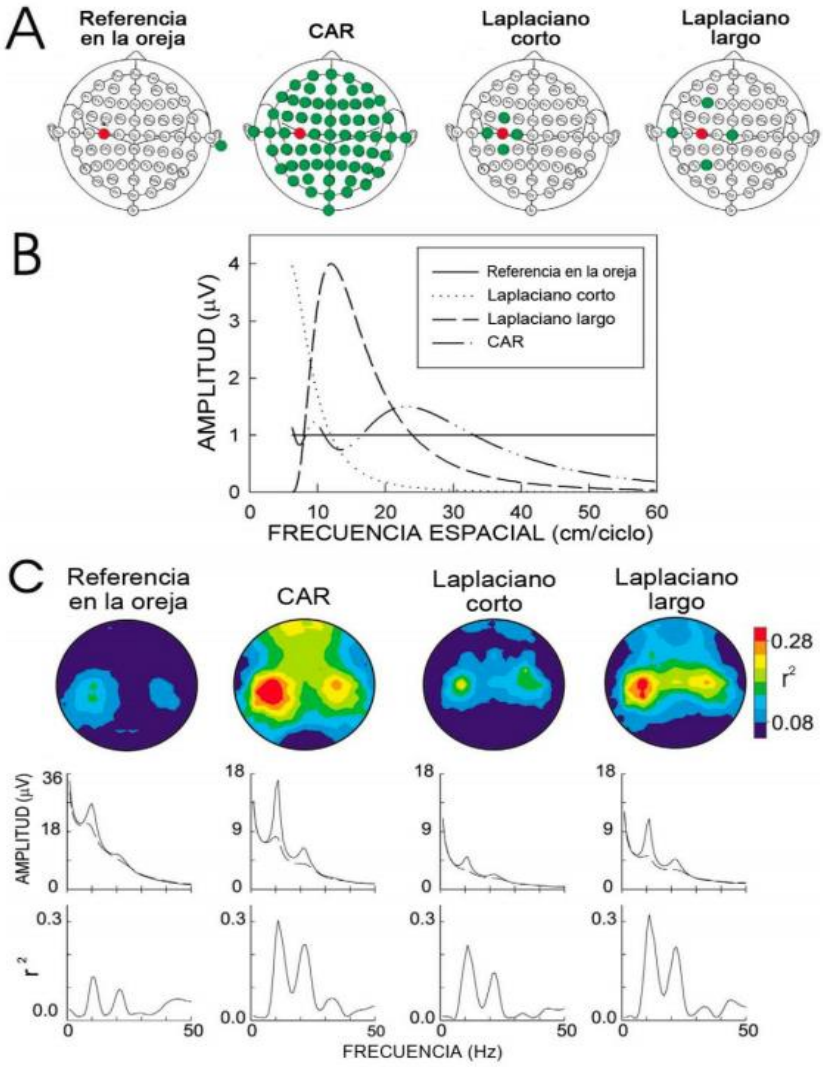

Figura 2-14. Comparación entre cuatro métodos espaciales de extracción de características. (A) Localización de los electrodos utilizados con la señal objetivo en rojo, concretamente, se quiere medir la señal del electrodo C3. (B) Banda de paso para cada método, raíz cuadrada de los valores cuadráticos medios de la señal recogida en C3. (C) Topografía de r_2 medida y amplitud espectral para cada método estudiado [6].

b) Traducción de características

En aplicaciones BCI, la etapa de traducción de características, o algoritmo de clasificación, se encarga de convertir las características extraídas anteriormente en comandos de un dispositivo. Para ello, analizará las características obtenidas en función de diversos parámetros y las clasificará para asociarlas con un comando determinado. El objetivo es asegurar que la selección atribuida se corresponda con la selección que el usuario pretendía realizar [6].

Los clasificadores se pueden dividir en dos grupos: clasificadores lineales y no lineales. En la Tabla 2-2 aparecen los clasificadores más usados en los sistemas BCI para ambos grupos.

Los métodos lineales asumen que los datos pueden separarse linealmente a través de un hiperplano de separación, el cual divide el espacio de datos en varias regiones, cada una correspondiente a una selección distinta. Este hiperplano de separación se coloca de tal manera que minimice el margen mínimo (distancia mínima entre cada punto y el hiperplano) o el margen medio [19]. En la Figura 2-15 se puede observar un clasificador lineal que maximiza el margen mínimo.

Pese a que los métodos lineales suelen ser más robustos que los no lineales y suelen requerir menor carga computacional, también pueden fallar en presencia de ruido u *outliers*, valores atípicos que provocan datos numéricamente distantes del resto, frecuentemente engañosos. En la Figura 2-15 (b) se observa cómo la presencia de *outliers* puede cambiar la localización del hiperplano de separación si la influencia de valores atípicos no está limitada. Se recomienda aplicar una regulación de los datos para limitar su influencia antes de aplicar el clasificador para obtener un hiperplano más fiable.

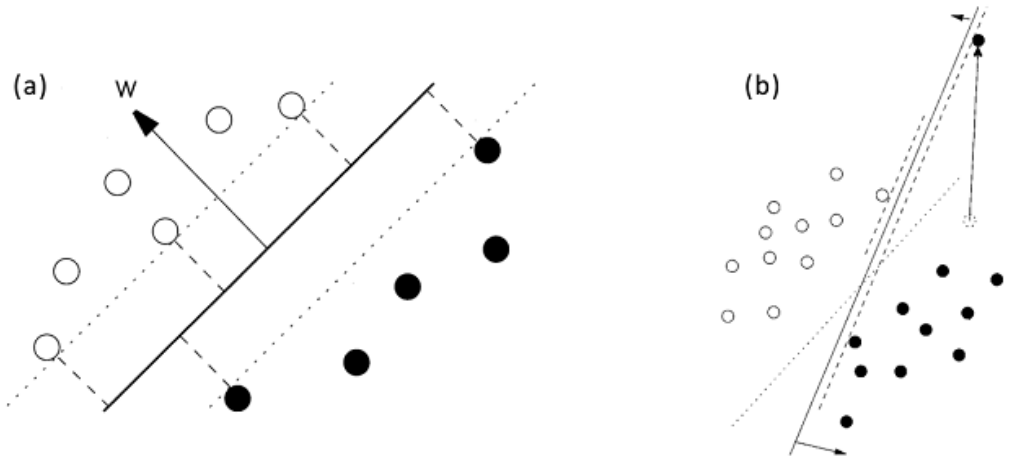

Figura 2-15. Clasificador lineal que maximiza el margen mínimo: en (a) se muestra el comportamiento óptimo y en (b) su comportamiento en presencia de un *outlier* [19].

<table>
<tr><th rowspan="3">Lineales</th><th colspan="4">No Lineales</th></tr>
<tr><th rowspan="2">Estructura Fija</th><th colspan="3">Estructura Modificable</th></tr>
<tr><th>Basados en Memoria</th><th>Combinaciones de No Linealidades Simples</th><th>Modelos Generativos</th></tr>
<tr><td>Análisis Discriminante Lineal (LDA)</td><td rowspan="3">Análisis Cuadrático Discriminante (QDA)</td><td>Algoritmo del vecinos más próximo</td><td>Redes Neuronales Artificiales (ANN)</td><td rowspan="2">Modelo de Mezclas Gaussianas</td></tr>
<tr><td>Perceptrón</td><td>Máquinas de Soporte Vectorial (SVM)</td><td>Árbol de Decisión</td></tr>
<tr><td>Regresión, regularización y adaptación</td><td>Regresión de Mínimos Cuadrados Parciales (PLS Regression)</td><td>Aprendizaje por Cuantificación Vectorial (LVQ)</td><td>Modelo Oculto de Markov (HMM)</td></tr>
</table>

Tabla 2-2. Clasificación de los métodos más empleados para la traducción de Características [9].

Los métodos no lineales son capaces de obtener precisiones mayores al tener en cuenta un mayor número de parámetros que los métodos lineales a cambio de requerir un mayor coste computacional [19].

En el estudio de Fabiani *et al* [12] se comparan tres métodos de traducción de características para diferenciar entre tres clases de datos: opción superior, inferior y central. El resultado puede observarse gráficamente en la Figura 2-16.

Los dos primeros clasificadores son lineales: K1 y K2. Sin embargo, K1 solamente emplea una dimensión, mientras que K2 emplea dos dimensiones. El clasificador K3 se corresponde con una función no lineal de K1. Tal y como se puede observar se aprecia una mejora significativa en los resultados de los clasificadores K2 y K3. Esto se debe principalmente a que K1 consiste en dos hiperplanos de decisión paralelos, por lo que impone una fuerte restricción en la clasificación de los datos.

En aplicaciones BCI, donde ser requiere procesado en tiempo real, generalmente se prefiere la simplicidad de los métodos lineales. Los métodos de clasificación no lineales son preferibles en el caso de que el número de muestras sea pequeño y la información sobre los datos sea limitada [19].

La etapa de traducción de características también debe considerar la adaptabilidad al usuario. Los clasificadores óptimos varían entre usuarios, es decir, el clasificador óptimo para un usuario normalmente no será el mismo que el clasificador óptimo para otro usuario distinto y, debido a esto, el sistema BCI debe cumplir una serie de requisitos repartidos en tres niveles [6].

El primer nivel requiere adaptar las características de las señales a cada usuario, puesto que habrá diferencias significativas entre un usuario y otro, por ejemplo, la amplitud de los potenciales SCP o P300. El segundo nivel requiere adaptar el clasificador a las fluctuaciones que

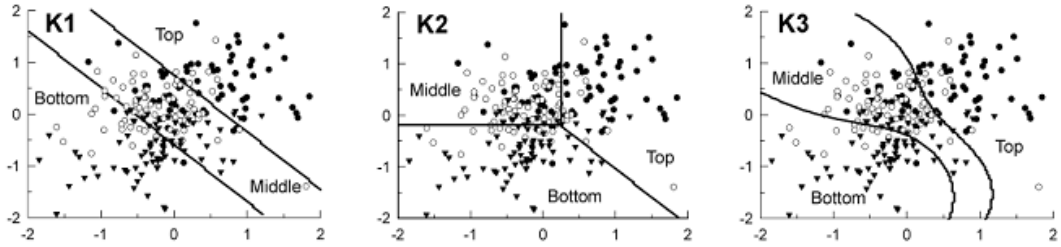

Figura 2-16. Comparación de los hiperplanos de separación para los tres métodos de traducción de características: K1 es un método lineal de una dimensión, K2 es un método lineal de dos dimensiones y K3 es un método no lineal [12].

pueden sufrir las características obtenidas debido a variaciones entre la motivación, la fatiga o la frustración, entre otros, del usuario dependiendo de la sesión. Finalmente, el tercer nivel requiere mantener la precisión del clasificador a largo plazo, siendo capaz de detectar disminuciones en los niveles de atención del usuario, responder ante ellos y ofrecer una realimentación que lo motive para los próximos intentos.

El tercer nivel está fuertemente ligado al fenómeno de aprendizaje de los clasificadores. Existen una serie de clasificadores, habitualmente no lineales, que son capaces de "aprender" las características de la señal de cada usuario y adaptarse a ellas gracias a una serie de datos de entrenamiento. Entre ellos se encuentran las máquinas de soporte vectorial (SVM, *Support Vector Machines*) o las redes neuronales artificiales (ANN, *Artificial Neural Networks*), caracterizados por ser algoritmos muy rápidos una vez están entrenados, y adecuados para aplicaciones en tiempo-real.

2.4.3. Aplicación

La tercera y última etapa del tratamiento de la señal es la etapa de aplicación, la cual se encargará de recibir la señal de control producida por la traducción de características y de implementar un *software* que traduzca ésta señal en comandos que realicen las acciones correspondientes. En la siguiente sección se detallarán las aplicaciones más comunes de los sistemas BCI.

2.5. APLICACIONES DE LOS SISTEMAS BCI

A pesar de que existen multitud de aplicaciones prácticas desarrolladas con sistemas BCI, la mayoría se basa en dos paradigmas sencillos: la selección de comandos o letras en una matriz, y el movimiento de un cursor. A continuación, introduciremos estos dos métodos, para justificar la elección hecha para la aplicación desarrollada en este trabajo.

2.5.1. Selección de letras

La aplicación de selección de letras es, probablemente, la más utilizada en los sistemas BCI. Un sinfín de aplicaciones más complejas utilizan técnicas de selección de letras, números o comandos para implementar alguna funcionalidad, ya sea con potenciales evocados P300, potenciales VEP o potenciales SCP.

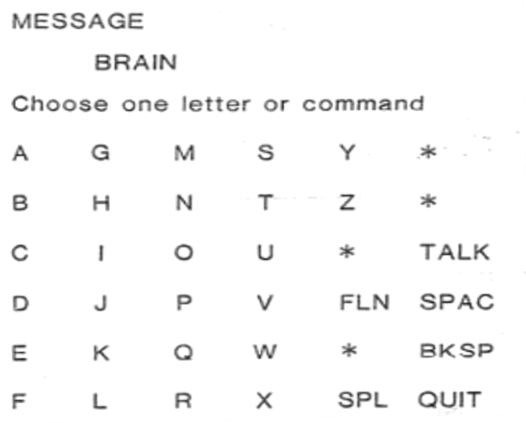

Figura 2-17. Primera matriz que empleo el paradigma odd-ball, utilizada en el estudio de E. Donchin para el desarrollo de la aplicación "*mental prosthesis*" [18]

Potenciales evocados P300

Las aplicaciones de selección de letras, de aquí en adelante conocidas como *P3Speller*, números o comandos con potenciales evocados P300 se basan en la técnica *odd-ball* descrita anteriormente. Se presenta al usuario una matriz de, por ejemplo, 6x6 letras y números. Cada cierto tiempo, del orden de milisegundos, se ilumina una fila o columna de forma que, tras una ronda de un determinado número de iluminaciones, cada fila y columna se han iluminado una vez y, por lo tanto, cada letra se ha iluminado dos veces.

Cada vez que se ilumina la letra que el usuario quiere seleccionar, éste provoca la aparición de un potencial evocado P300. El número de intentos que serán necesarios para obtener buenas precisiones lo determinará el clasificador que se haya generado para cada usuario y la experiencia del mismo con estos sistemas. Normalmente diez intentos son suficientes para obtener una buena precisión en sujetos de control [20].

Fue el equipo de Emanuel Donchin quien diseño este tipo de aplicaciones por primera vez en 1988. En la figura 2-17 se puede ver el esquema presentado por este equipo de la universidad de Illinois [18].

Potenciales Evocados Visuales

A continuación, se introduce una manera de seleccionar letras para mejorar la comunicación de personas con graves discapacidades utilizando como señal de control los potenciales VEP.

La interfaz de la aplicación presenta una matriz de letras, cuyas celdas se iluminan continuamente a distintas frecuencias. El usuario debe mirar fijamente a la celda que contiene la letra que desea escribir, generando un fenómeno de resonancia a la misma frecuencia a la que se ilumina la letra a seleccionar.

El sistema identifica los picos frecuenciales de ésta y sus armónicos, y determina hacia dónde está mirando el usuario. Si la matriz es demasiado grande como para utilizar todas las frecuencias disponibles, se pueden iluminar aleatoriamente subgrupos de letras, hasta que cada una de ellas se haya iluminado a una frecuencia diferente [6].

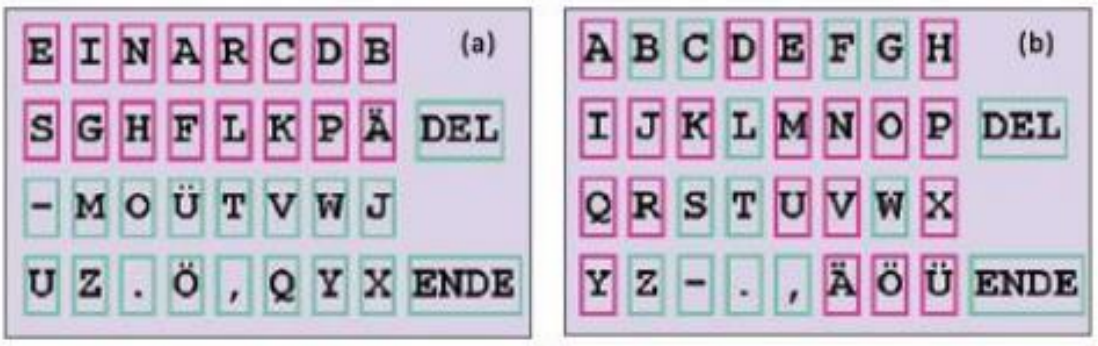

Figura 2-18. Selección de letras utilizando potenciales corticales lentos (SCP) con selecciones binarias: mitad de las mismas en magenta y la otra mitad en azul [21].

El utilizar una señal de control exógena tiene la ventaja de no requerir entrenamiento alguno. Sin embargo, sí es necesaria la capacidad de mantener la mirada fija en un punto.

Potenciales Corticales Lentos

Los potenciales corticales lentos (SCP) únicamente pueden realizar selecciones binarias, de modo que la selección de letras se lleva a cabo dividiendo sucesivamente el alfabeto en dos partes: primero se elige una mitad, luego un cuarto, etcétera. Teniendo en cuenta que los potenciales corticales lentos se clasifican dentro de las señales de control endógenas, el usuario necesitará entrenamiento para poder seleccionar las letras adecuadamente. En la Figura 2-18 se puede observar la solución de Bensch *et al* [21].

2.5.2. Movimiento de un cursor

El objetivo de esta aplicación será el de aprovechar el control de movimiento de un cursor para seleccionar botones, navegar por menús o moverse dentro de algún videojuego.

Potenciales Corticales Lentos

Como se ha comentado anteriormente, los potenciales SCP sólo permiten selecciones binarias, por lo que solo podremos controlar el cursor en 1D. Una aplicación utilizando potenciales corticales lentos se basa en mostrar por pantalla un cursor que se desplace horizontalmente a velocidad constante. Un sistema de este tipo es el implementado por Yun-gong LI *et al* en [22].

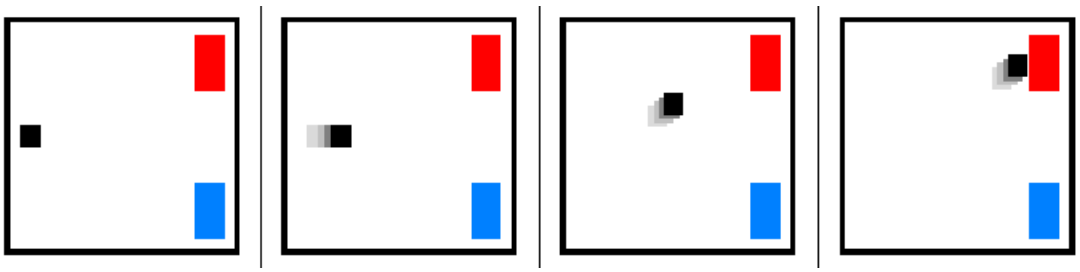

Figura 2-19. Control del movimiento de un cursor verticalmente mediante potenciales corticales lentos para seleccionar entre dos opciones [22].

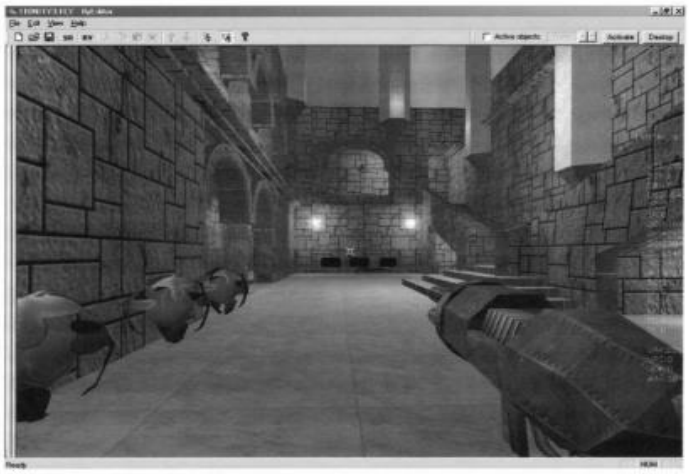

Figura 2-20. Control del movimiento en videojuegos mediante ritmos sensoriomotores μ y β [23].

Ritmos Sensoriomotores μ y β

A diferencia de los cursores controlados mediante potenciales corticales lentos, los ritmos sensoriomotores μ y β pueden llegar a controlar un cursor en dos dimensiones con el suficiente entrenamiento. En este caso, las señales de control de las dimensiones serán las componentes ERP y ERS de estos ritmos. De esta manera, se pueden controlar menús complejos, aplicarse al control de prótesis, al movimiento de sillas de ruedas o el movimiento en videojuegos, tal y como se muestra en la Figura 2-20 [23]. Otro ejemplo se

2.6. APLICACIONES MÓVILES BASADAS EN BCI

En esta sección nos aproximaremos al objetivo final que es el desarrollo de una aplicación Android para controlar un Smartphone mediante BCI. Aquí se detallan algunos de los sistemas BCI implementados en plataformas móviles, los cuales servirán de base para el desarrollo de la aplicación.

En este punto se hará una revisión de sistemas BCI implementados en plataformas móviles publicados en artículos científicos hasta la fecha. Es un campo en expansión con muchas aplicaciones prácticas, de modo que la variedad de artículos es significativa.

2.6.1. Yu-Te Wang et al: "A cell-phone-based brain–Computer interface for communication in daily life" [24]

El objetivo de los autores del artículo era crear un sistema de comunicación mediante BCI que fuera portátil y útil en la vida real, integrando la tecnología móvil con un sistema de EEG inalámbrico para hacer llamadas telefónicas con el terminal.

El diagrama de bloques propuesto por Yu-Te Wang *et al* se muestra en la figura 2-21.

Diseño del Hardware

La pantalla estimuladora consiste en una LCD de 21 pulgadas que mostrara una matriz con los números 0-9, tecla ENTER y retorno de carro. Para la estimulación se utiliza un esquema básico basado en SSVEP entre 9 y 11.75 Hz con espaciado de 0.25 Hz. La señal se recoge en un casco

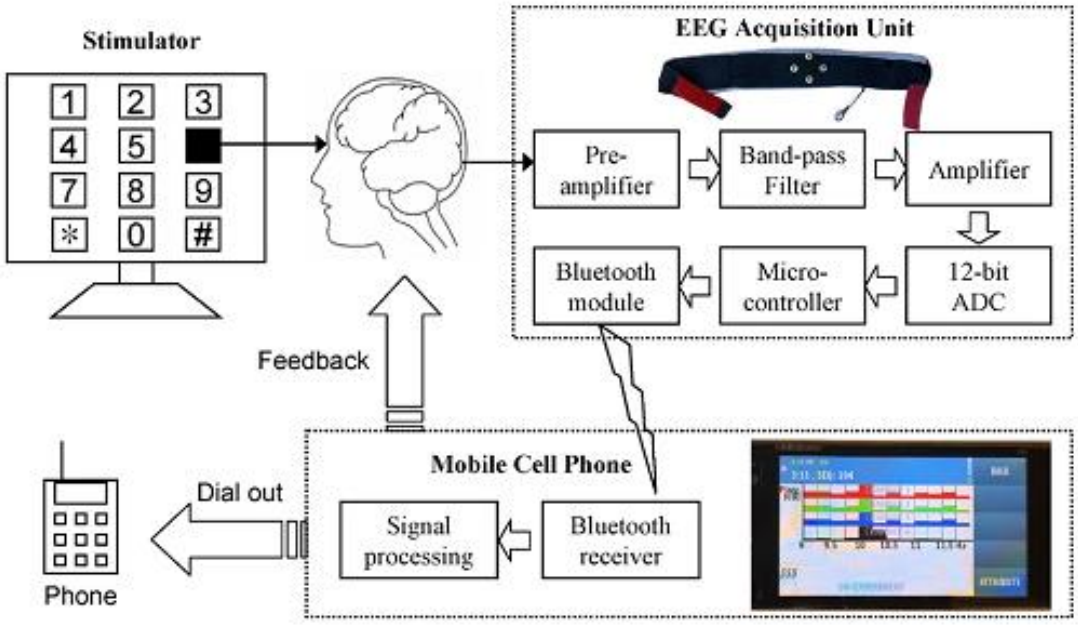

Figura 2-21. Arquitectura empleada por Yu-Te Wang *et al* en su artículo [24].

EEG de cuatro canales y es amplificada y convertida al dominio digital. Posteriormente se envía la información mediante un módulo de Bluetooth al terminal móvil para su procesado, tras el cual se seleccionará un comando.

Procesado de señal

Se realiza en un Nokia N97 con sistema operativo Symbian S60. Se muestra la señal EEG en la pantalla del terminal, se aplica un filtro paso-banda digital y se estima la frecuencia principal mediante FFT (Fast Fourier Transform) o CCA (Canonical Correlation Analysis). Una vez estimada, se selecciona el comando y se realimenta al usuario con la nueva información (aparece el comando seleccionado en la parte de debajo de la pantalla y se reproduce la palabra que lo describe) para la selección de un nuevo comando o la finalización de la aplicación

Experimento, resultados y conclusiones

Se probó el sistema en 10 sujetos, diferenciando entre detección por FFT o CCA. El experimento consistía en marcar el número 1234567890 seguido de la tecla ENTER para llamar, tanto en detección por FFT como por CCAA. Las tablas 2-3 y 2-4 resumen los resultados obtenidos:

Se observa que la ITR (bits/min) alcanzada por la FFT es menor que con el método de detección CCAA. La implementación del sistema ha sido por tanto un éxito, alcanzando valores de efectividad similares a los alcanzados en otros estudios similares con procesado de señal en PC.

Subject	Input length	Time (s)	Accuracy (%)	ITR (bits min^{-1})
s1	11	72	100	32.86
s2	11	72	100	32.86
s3	19	164	78.9	14.67
s4	11	73	100	32.4
s5	17	131	82.4	17.6
s6	11	67	100	35.31
s7	11	72	100	32.86
s8	13	93	92.3	20.41
s9	11	79	100	29.95
s10	11	66	100	35.85
Mean	12.6	88.9	95.9	28.47

Tabla 2-3. Tabla resumen de los resultados basados en FFT [24]

Subject	Online CCA	Online FFT	Offline FFT	Putative ITR from offline FFT			
				Ch1	Ch2	Ch3	Ch4
s1	44.79	32.86	36.68	**36.68**	33.58	32.48	29.77
s2	46.25	32.86	26.49	**26.49**	10.51	5.91	9.29
s6	49.05	35.31	19.43	**19.43**	3.03	3.15	1.92
s10	43.18	35.85	15.24	2.2	8.46	**15.24**	4.21
Mean	45.82	34.22	24.46	21.2	13.9	14.2	11.3

Tabla 2-4. Tabla resumen de los resultados basados en CCCA [24].

2.6.2. Yu Zhou et al: "A Novel Platform of Brain Computer Interface Based on Android" [25]

Los autores de este artículo proponen una arquitectura que sirva de plataforma para controlar una aplicación basada en imagen motora (*motor imaginery*), que utiliza potenciales corticales lentos (SCP) como señal de control. La aplicación es instalada en un dispositivo Android y podría consistir, por ejemplo, en un juego de carreras de coches. El esquema utilizado se describe en la figura 2-22.

Extracción de la señal

Las señales cerebrales se recogen mediante EEG y son preprocesadas. Se aplica una amplificación con factor 10.000. A continuación, se realiza un filtrado paso-bajo en la banda 0.5-100 Hz con rechazo en la banda de 50 Hz para evitar interferencias con la banda de la red eléctrica. Por último, se realiza una conversión analógico-digital, almacenando la señal en un *array* de dos dimensiones.

Procesado en PC

La señal se envía a un PC mediante servicio web, donde se procesa con Matlab para extraer el comando seleccionado por el usuario. La información en aplicaciones de imagen motora se concentra en la banda de 8-30 Hz.

El procesado se realiza en dos etapas, extracción de características y traducción de esas características en comandos que la aplicación pueda entender. La extracción de características se hace mediante el algoritmo de máxima entropía, un estimador eficiente para señales EEG. Una vez detectada la selección del usuario se manda el comando correspondiente a la aplicación.

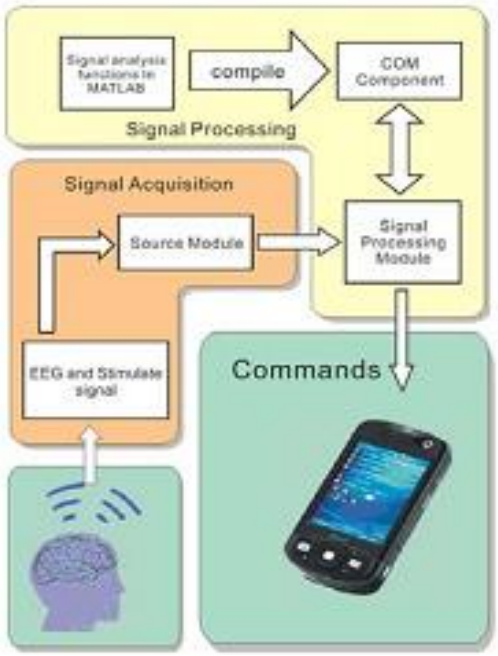

Figura 2-22. Esquema utilizado en la propuesta de Yo Zhou [25]

Aplicación

La aplicación ejecuta la orden que recibe y realimenta al usuario con la nueva información disponible resultado de la ejecución del comando. En concreto, los autores probaron el sistema con una aplicación con 3 comandos: derecha, izquierda y no-movimiento. Los resultados mostraron una tasa de 10 bits/s con un acierto entre el 70 y el 75%.

2.6.3. Scott Vernon and Sanjay S. Joshi: "Brain–Muscle–Computer Interface: Mobile-Phone Prototype Development and Testing" [26]

El artículo trata sobre el desarrollo de un sistema para personas con grave discapacidad (por ejemplo, con afectación de la medula espinal a cualquier altura) para el control de diversos aparatos de la vida diaria. En este caso, no se utilizan señales cerebrales, sino que aprovecha los potenciales que aparecen durante el movimiento de un músculo para predecir las intenciones del usuario. La señal se recoge mediante EMG (electromiografía), situando los electrodos sobre la superficie más cercana al músculo elegido para el control del sistema.

Basándose en anteriores investigaciones del equipo que redacta el artículo, se aprovechará la capacidad de las personas, de producir potencia de señal en dos bandas diferentes de frecuencia al mover un músculo, por lo que tendremos dos canales de información. El sistema constara de dos modos, según sea usado para posicionar un cursor 1D o 2D de una o dos coordenadas respectivamente, variando estas según la potencia de cada una de las dos bandas de frecuencia. La posición de este cursor en una pantalla, determinará la acción a realizar.

La elección del músculo de control debe ser inteligente, siendo uno no utilizado normalmente el candidato ideal. Una posibilidad es el músculo auricular superior (en la parte de arriba de la oreja), el cual no se suele utilizar. Muchas personas requieren de entrenamiento para controlarlo correctamente.

Posteriormente la señal se enviará a un Smartphone para su procesado. Una vez extraídas las características de la señal de entrada y después de haber sido traducidas al comando correspondiente, este se comunicará por Bluetooth con el dispositivo que se quiere controlar.

Arquitectura del hardware

El esquema usado para la adquisición y el procesado de la señal, se muestra en la figura 2-23. El equipo utilizado en el desarrollo del sistema consiste en un módulo de adquisición de datos Motion Lab Systems Z03-000, que es un sensor sEMG que tiene un CMRR (rechazo al modo común) mayor a 100 dB en la banda de 65 HZ y una ganancia de 300dB, y un teléfono HTC Dream con sistema operativo Android.

La salida del módulo de adquisición es analógica, por lo que se convierte al dominio digital en el mismo teléfono utilizando la entrada de audio, la cual dispone de un convertidor ADC que nos permitirá analizar la señal posteriormente. La frecuencia de muestreo es de 8000 Hz, la única disponible, aunque mediante diezmado se consigue una frecuencia equivalente de 4000 Hz.

Mediante el procesado de la señal de entrada, se determinará un comando. Este será enviado posteriormente mediante Bluetooth al aparato que queramos controlar. En el caso de este experimento se utilizó una televisión moderna.

Procesado de señal

Como se ha comentado, el sistema se basará en el control de un cursor 1D o 2D que se desplaza por la pantalla del teléfono para seleccionar la acción que queremos realizar.

En el modo 1D no se diferencia entre bandas de frecuencia, siendo el control del cursor directamente proporcional a la intensidad de la contracción del músculo.

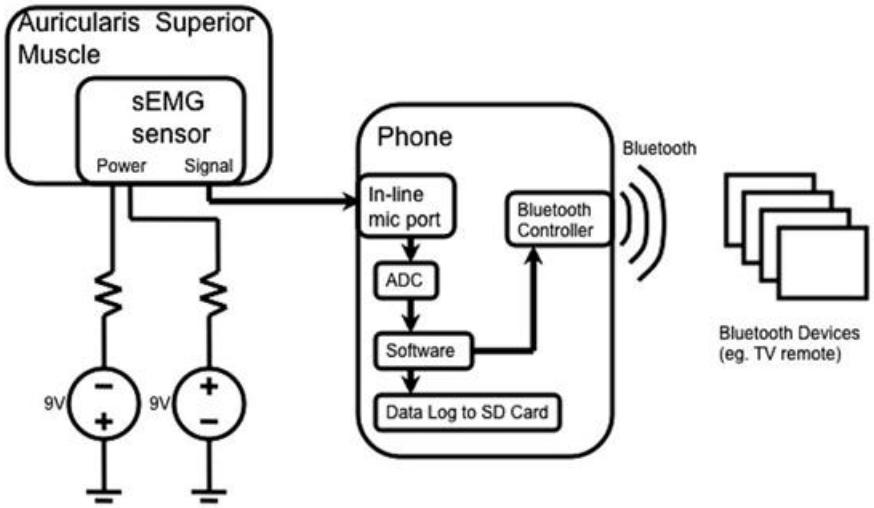

Figura 2-23. Arquitectura del sistema [26]

En el modo 2D se definen dos bandas de frecuencia, la banda 1 [20-40] Hz y la banda 2 [60-80] Hz, que constituirán los dos canales de información necesarios para el control del cursor. Dependiendo de la potencia que el sujeto genere en cada una de las bandas (calculada mediante el teorema de Parseval en intervalos de 0.25 segundos), la posición se modificará de acuerdo a las ecuaciones (2.1) y (2.2).

$$X_{POS} = \frac{1}{Effort_X}\left[1.75\left(\frac{PowerBand_1}{MaxPowerBand_1}\right) - 0.75\left(\frac{PowerBand_2}{MaxPowerBand_2}\right)\right] \quad (2.1)$$

$$Y_{POS} = \frac{1}{Effort_Y}\left[-0.75\left(\frac{PowerBand_1}{MaxPowerBand_1}\right) + 1.75\left(\frac{PowerBand_2}{MaxPowerBand_2}\right)\right] \quad (2.2)$$

Para mejorar la experiencia de usuario, dado que el tiempo de refresco de la pantalla utilizada son 30 Hz y la frecuencia de cálculo de una nueva posición del cursor son 4 Hz, se diseñó un sistema que desplazaba el cursor de manera continua entre dos posiciones consecutivas.

Interfaces de usuario

El sistema dispone de un modo de calibración que lo adapta a la habilidad única de cada sujeto para controlar un músculo en concreto. Este se iniciará al encender el sistema. Se incluye también, además de los modos para el control del sistema 1D y 2D, un modo de aprendizaje, ya que el correcto manejo del sistema requiere de un cierto entrenamiento en el correcto control del músculo elegido. La interfaz gráfica para el modo 1D (izquierda) y 2D (derecha) se muestra en la figura 2-24.

En el caso de 1D el comando el cursor se mueve por los comandos de manera vertical mientras el sujeto mantenga la contracción. Para ejecutar el comando seleccionado, se deberá relajar el músculo. El caso del cursor 2D es más complicado y para seleccionar el comando se deberá realizar un círculo alrededor de él. Posteriormente se deberá relajar el músculo para ejecutarlo y confirmar la selección. Se recomienda manejar el cursor 1D perfectamente antes de aprender el manejo de la opción 2D.

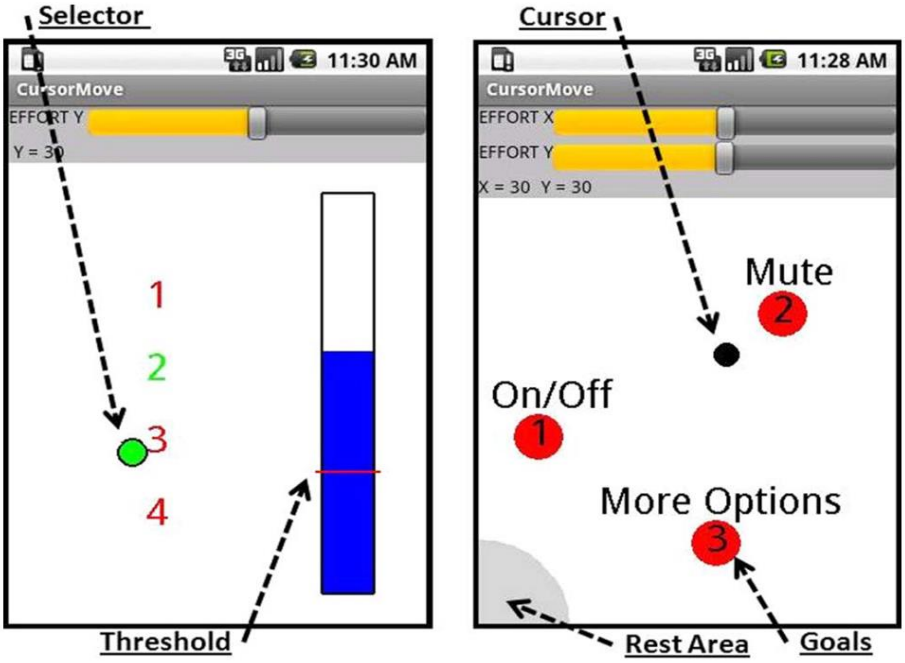

Figura 2-24. Interfaces para 1D y 2D de izquierda a derecha [26]

	Target	Hit Rates %	Avg. Time to Target (s)	STDEV (s)	Successful Contractions Averaged
EV1	1	100	4.40	0.3	20
	2	100	6.47	2.21	17
	3	100	7.83	2.04	21
	4	100	8.60	1.34	19
EV2	1	100	4.15	0.3	12
	2	100	5.77	0.33	19
	3	100	7.37	0.36	18
	4	100	8.85	0.37	19

Hit rates and average time-to-target for the successful contractions for each of the randomly presented targets achieved by subject.

Tabla 2-5. Resumen de resultados para el modo 1D [26]

	Target	Hit Rates %	Avg. Time to Target (s)	STDEV (s)	Successful Contractions Averaged	Average Number of Targets Hit
EV1	1	70.00	6.12	10.77	21	1.76
	2	79.49	6.49	10.56	31	1.65
	3	59.52	9.47	7.72	25	2.16
EV2	1	76.74	2.42	2.27	33	1.39
	2	95.45	4.47	3.17	42	2.24
	3	85.37	3.47	3.94	35	2.29

Hit rates and average time-to-target for the successful contractions for each of the randomly presented targets achieved by subject.

Tabla 2-6. Resumen de resultados para el modo 2D [26]

Resultados del experimento

El sistema fue probado en un sujeto de 26 años con avanzado estado de parálisis, con experiencia en anteriores experimentos del equipo, basados en el manejo de un cursor mediante EMG.

Los resultados de los experimentos diseñados se muestran en las tablas 2-5 y 2-6.

2.6.4. Amr S. Elsawy and Seif Eldawlatly "P300-based Applications for Interacting with Smart Mobile Devices" [27]

Este artículo trata sobre el desarrollo de dos aplicaciones para móviles con sistema operativo Android, controladas por BCI, que permiten al usuario interactuar con el terminal. La primera de las aplicaciones, llamada RunApp permite abrir cualquier aplicación instalada en el terminal. La segunda se llama ImgView y permite la visualización de fotos guardadas en el teléfono.

El clasificador utilizado se basa en el método de análisis de componente principal (PCA, *Principal Component Analysis*), que se caracteriza por un bajo coste computacional, con un nivel

aceptable de acierto en la extracción de características. Para la parte de traducción y clasificación se utiliza la técnica del discriminante lineal de Fisher.

En la recogida de señales EEG se ha utilizado el casco desarrollado por la empresa Emotiv llamado EPOC que es inalámbrico y viene con herramientas para la conexión directa con el terminal móvil.

Diseño de la aplicación RunApp (Figura 2-25 a)

Se diseñó una matriz con los iconos de las aplicaciones instaladas en el terminal. Esta tiene un tamaño de 4 x 4 y se actualiza automáticamente. Si hubiera más de 15 aplicaciones, se dispone de un botón "siguiente" que nos mostrará en la matriz las siguientes 15 aplicaciones.

Para la selección de la aplicación se utiliza el paradigma de iluminación de columnas y filas.

Diseño de la aplicación ImgView (Figura 2-25 b)

Dispone de un área de visualización donde se muestra la foto y una matriz de selección de comandos donde podemos ejecutar una acción, como pasar a la siguiente, hacer zoom, rotarla, etc. La matriz es de 3 x 3, pudiendo albergar así hasta 9 opciones de control.

Métodos y resultados del experimento

Las aplicaciones fueron probadas en 6 sujetos sanos. La adquisición de datos si hizo con el casco inalámbrico Emotiv EPOC, que cuenta con 14 electrodos y se conecta directamente al móvil como se muestra en la figura 2-26. Se fijó además un número máximo de 10 iluminaciones para reducir el tiempo de manejo de las aplicaciones.

Se incluyó también un modo de aprendizaje para cada una de las aplicaciones, en el que los usuarios pudieran practicar y aprender a usarlas.

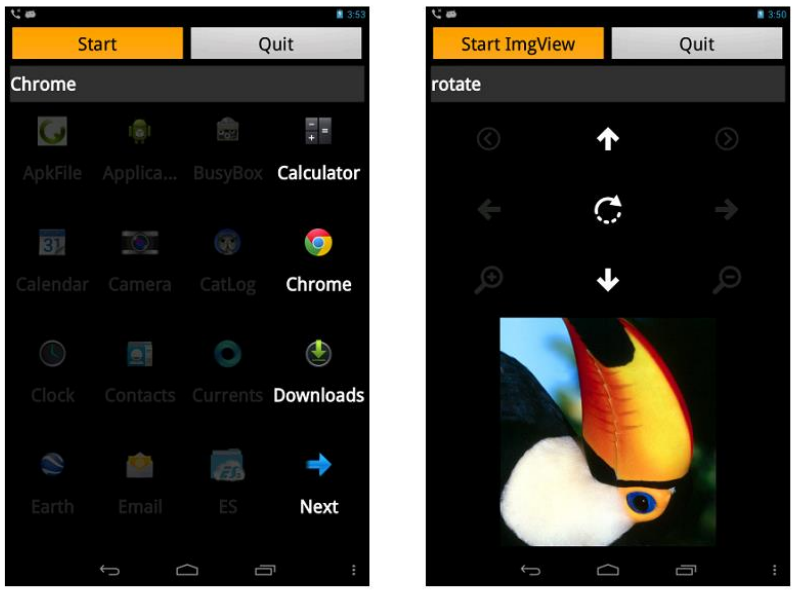

Figura 2-25. Interfaces de RunApp (a) y de ImgVew (b) [27]

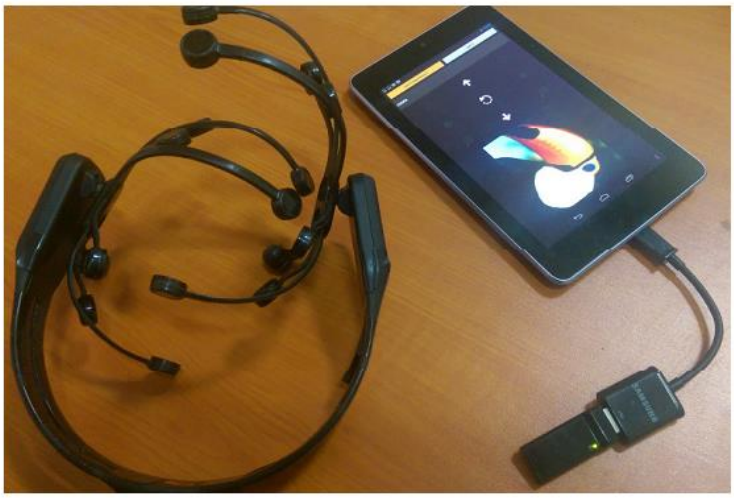

Figura 2-26. Esquema propuesto por el equipo [27]

El experimento consistió en una sesión de entrenamiento seleccionando 20 símbolos en el laboratorio, y un test online seleccionando 12 símbolos para cada aplicación.

En la figura 2-27 se muestran los resultados de la sesión de laboratorio. El acierto máximo es de un 93.33% para el sujeto 5, mientras que el acierto medio es de un 66.3%. La falta de acierto de algunos sujetos se explica en su falta de experiencia en sistemas BCI basados en P300 y en la falta de concentración durante el experimento.

En los test online, en un ambiente más propicio y con menos distracciones, el porcentaje de acierto aumentó notablemente como podemos ver en la figura 2-28 (alcanzando una media de 79.17% para la aplicación RunnApp y 87.5% para ImgView).

La diferencia entre aplicaciones se explica por la mayor cantidad de opciones de RunApp, reduciendo el tamaño de cada símbolo seleccionable.

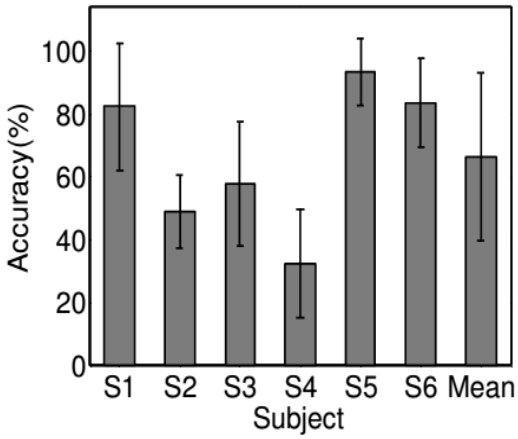

Figura 2-27. Resultados sesión laboratorio [27]

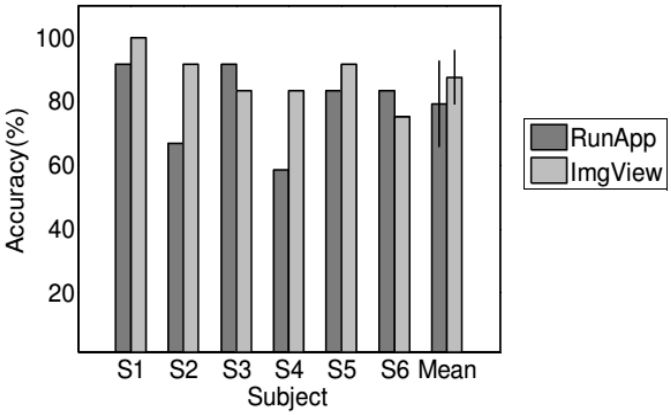

Figura 2-28. Resultados sesión online [27].

CAPÍTULO 3

ANÁLISIS DE LOS POTENCIALES EVOCADOS P300

3.1. ADQUISICIÓN DE LOS POTENCIALES EVOCADOS P300

Como se introdujo en el anterior capítulo, el potencial P300 se corresponde con una deflexión positiva de voltaje detectada mediante EEG en los instantes posteriores a un estímulo inesperado.

Los P300 forman parte de un amplio grupo de potenciales llamados potenciales relacionados con eventos (ERPs, E*vent-Related Potentials)* y fueron descubiertos por *Sutton et al* en 1965 como resultado de la confluencia de investigaciones de la teoría de la información y la neurofisiología [28]. El nombre es debido a la latencia del P300, que aparece 300 ms después del estímulo [29].

Para reproducirlo artificialmente, se recurre al paradigma *oddball,* introducido en el anterior capítulo. Este paradigma consiste en presentar una serie de estímulos objetivo de forma infrecuente camuflados sobre un fondo de estímulos no deseados mucho más frecuentes. Está demostrado que, a menor probabilidad de evento, mayor será el potencial P300 generado. La figura 3-1 ilustra el funcionamiento ideal del paradigma *oddball* para la producción de potenciales P300 [28].

El P300 se localiza en torno a la zona centro-parietal del córtex (Fz, Cz y Pz), y su amplitud se encuentra en torno a los 10μV. Esta amplitud es la diferencia del voltaje entre la media de la señal EEG antes del estímulo y la amplitud máxima del P300 [28].

A pesar de la aparente sencillez de la obtención de este ERP, aún no sabemos la verdadera causa que provoca el P300. La teoría más aceptada, es la actualización del contexto (*Context-Updating*, Donchin *et al*, 1986) del usuario. Esta teoría afirma que el P300 es el resultado de la actividad neuronal que corresponde a una modificación en el entorno del usuario y la correspondiente

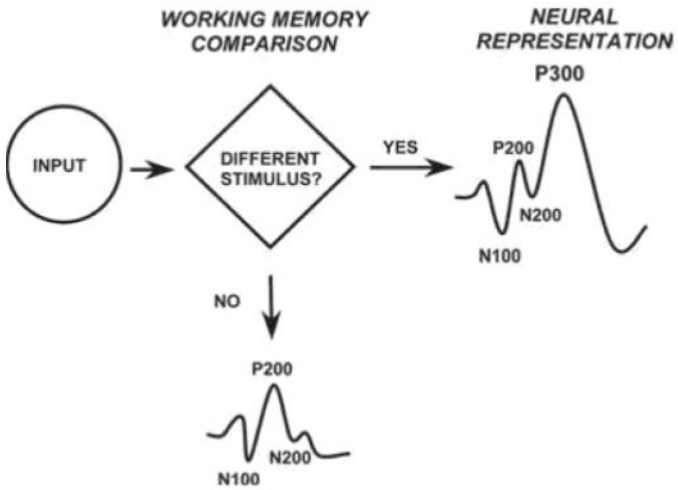

Figura 3-1. Ilustración esquemática del contexto de aparición de un potencial P300. Si el estímulo recibido por el sujeto no es diferente del anterior, sólo se producen los potenciales evocados sensoriales (N100, P200, N200) [28]

actualización de la información para preparar una posible respuesta relacionada con los eventos almacenados en nuestra memoria de corto plazo, que contiene la información sobre la tarea que desempeñamos en un momento determinado.

La amplitud del P300 ha demostrado ser sensible a la habituación y deshabituación, lo que hace suponer que este ERP implica operaciones relacionadas con la memoria, así como también de la atención que el usuario está prestando al estímulo [28]. La teoría de la actualización de contexto fue postulada tras la observación de los cambios que producía en la amplitud la manipulación de la probabilidad del estímulo-objetivo del P300.

3.1.1. Detección y componentes del P300

Numerosos estudios han tratado de caracterizar este potencial y de establecer las pautas óptimas para su obtención. Se ha declarado que la utilización de filtros paso-alto para eliminar las variaciones lentas de la señal influye en la forma del potencial, siendo conveniente aplicar un filtro a partir de 0.1Hz (frecuencias de corte mayores distorsionarían la forma de la señal).

La respuesta de los potenciales P300 es pequeña y aparece superpuesta con la actividad EEG de fondo, así como con ruido, y debido a ello, una respuesta individual no suele ser reconocible. No obstante, resulta fundamental poder distinguir la actividad evocada por el estímulo de la actividad EEG de fondo.

En este sentido conviene mencionar que la señal P300 está solapada en frecuencia con el contenido espectral del EEG de fondo, por lo que no es posible aplicar un filtrado en frecuencia para aislar la respuesta evocada. Como alternativa, comúnmente se realiza un filtrado en el dominio del tiempo aprovechando la relación temporal, repetitiva a lo largo de las diferentes épocas de la señal obtenida, entre la aparición del estímulo y la actividad evocada, típicamente a 300ms desde la presentación del estímulo. Este filtrado temporal se conoce como promediado sincronizado, y consiste en promediar todas las épocas de la señal estableciendo como referencia el punto en el que se ha generado el estímulo que produce el potencial P300 [30]. En la Figura 3-2 se observa el resultado de promediar la señal a la hora de detectar los potenciales P300.

Durante la adquisición de la señal, la eliminación de artefactos, y en concreto la eliminación movimiento ocular, es una etapa muy importante a la hora de analizar la señal. Para eliminar el movimiento ocular y los pestañeos producidos por el usuario que degradan los potenciales P300 se pueden utilizar dos enfoques distintos, explicados a continuación [29].

El primero se basa en rechazar las épocas donde se hayan producido movimientos oculares o pestañeos. Para ello se monitoriza el EOG posicionando electrodos cerca de los ojos (típicamente encima y debajo de ellos) y se rechaza cada época si la señal EOG supera cierto umbral (por ejemplo $\pm 100\ \mu V$).

Este enfoque posee dos inconvenientes principales: el primero reside en que rechazar épocas de la señal disminuye la eficiencia del registro (en el peor caso posible todas las épocas serían rechazadas) y, por tanto, se recomienda a los sujetos no mover los ojos o pestañear mientras se realizan las pruebas; esto conduce al segundo inconveniente, puesto que recomendar a los sujetos no producir movimientos oculares cambia la naturaleza de la prueba causando que los sujetos dividan su atención entre sus ojos y el estímulo, produciendo un cambio en la forma del potencial P300.

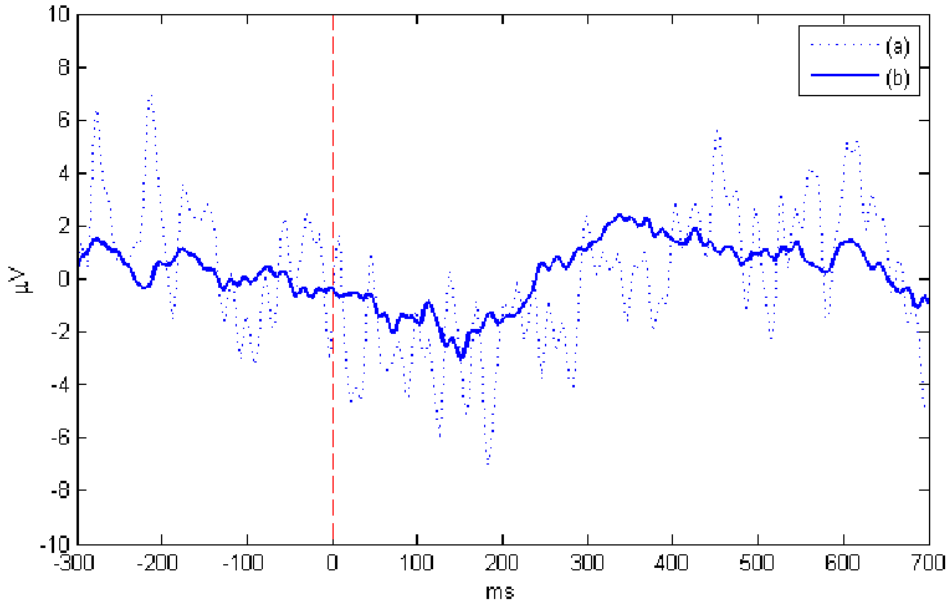

Figura 3-2. Potencial evocado P300 recogido en el electrodo Pz y generado con un *odd-ball* auditivo donde el usuario debía detectar tonos objetivos con probabilidad de ocurrencia de 0,2. La curva punteada (a) se corresponde con la primera época registrada y la curva (b) se corresponde con un promediado sincronizado de 90 épocas, permitiendo distinguir el potencial. La línea roja rayada indica el momento en el que se presentó el estímulo [29].

El segundo enfoque se basa en sustraer la señal EOG del EEG. Para ello se suelen emplear filtros adaptativos, los cuales requieren una señal de referencia fuertemente correlada con el ruido o artefacto que se desea eliminar (en este caso sería la señal EOG extraída con los electrodos situados en torno al ojo). Desafortunadamente esta extracción no es perfecta y puede eliminar ciertas porciones de la señal EEG recogida en los electrodos frontales, más cercanos a los ojos.

La detección de los potenciales evocados P300 se lleva a cabo encontrando los picos de amplitud que se elevan de la actividad basal, los cuales se encuentran retrasados 200 o más milisegundos respecto a la estimulación. Esta detección se realiza en un solo electrodo, típicamente Cz o Pz, y la latencia del P300 varía de un electrodo a otro, alcanzando el valor mínimo en los electrodos situados en la zona frontal del córtex [29].

Sin embargo, esta manera de tratar a los potenciales P300 no tiene en cuenta la posibilidad de que éstos estén conformados por múltiples procesos generados en distintas localizaciones. Un estudio más detallado del potencial P300 revela que éste está compuesto de tres ondas positivas superpuestas e independientes: P3a con un máximo en torno a 250 ms, P3b con un máximo en torno a 350 ms y una onda lenta (SW, *Slow Wave*). Estas componentes se observan en la figura 3-3 [31].

La figura 3-4 presenta un modelo neurofisiológico para las señales P3a y P3b basado diversos estudios [28]. Estos resultados pueden explicar los orígenes de las ondas componentes del P300. La onda P3a es producida en la zona frontal del córtex al procesarse el estímulo. Si el sujeto está atento al estímulo y este tiene que ver con la tarea actual, activa la región de la memoria temporal y se producen la onda P3b, en la zona parietal, y la onda lenta.

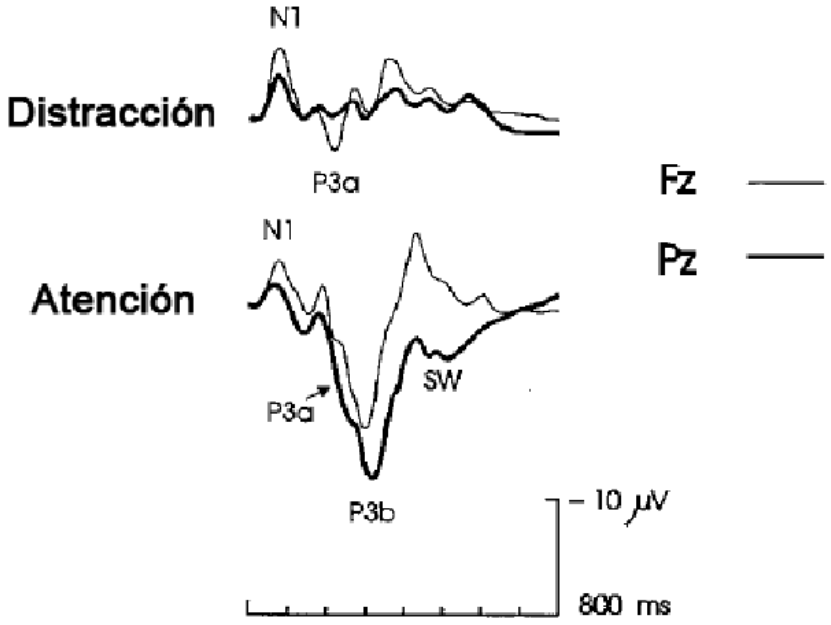

Figura 3-3. Componentes del potencial evocado P300 en dos localizaciones distintas: Fz y Pz. En la gráfica superior aparece la respuesta cuando el sujeto no atiende al estímulo, donde se observa que la onda P3a sigue apareciendo. En la gráfica inferior, por el contrario, aparece la respuesta cuando el sujeto atiende al estímulo, generando las tres componentes principales: P3a, P3b y *slow wave*. [29]

Aunque las tres componentes varían de la misma manera con la probabilidad de estímulo (aumentando su amplitud a medida que el objetivo se hace más improbable), son sensibles a la información obtenida sobre el estímulo en cuestión: la onda P3a no se ve muy afectada si el sujeto no está atendiendo al estímulo, mientras que la onda P3b y la onda lenta aumentan su amplitud si el sujeto pone atención, tal y como se puede observar, de nuevo, en la Figura 3-3. Si fuese necesario identificar las tres componentes por separado se podría realizar un PCA (*Principal Component Analysis*, o Análisis de Componentes Principales).

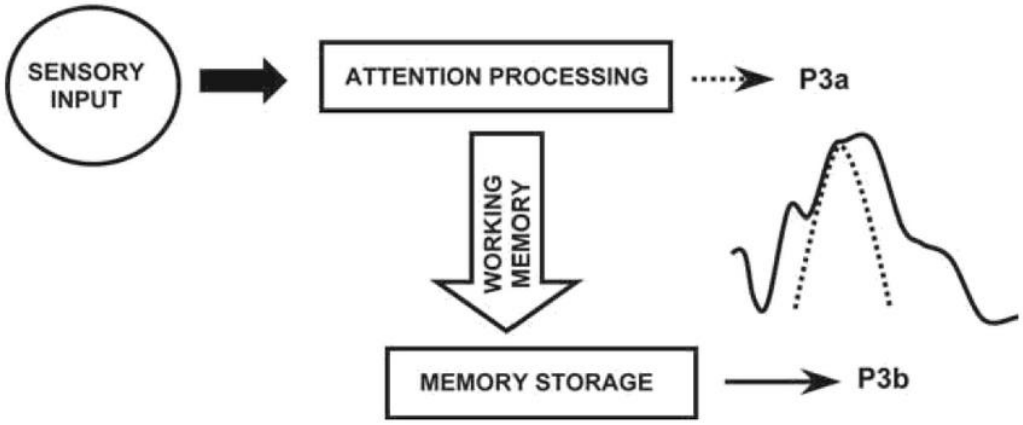

Figura 3-4. Modelo esquemático de la actividad cognitiva del P300 [28].

3.1.2. Factores que afectan al P300

En esta sección se analizarán los principales factores que afectan a la amplitud y la latencia del potencial P300, características de especial importancia en la detección del mismo para aplicarlo en sistemas BCI.

Factores psicológicos

Dentro de los aspectos psicológicos del sujeto cobran importancia, como ya se ha dicho en varias ocasiones, la atención del sujeto y la probabilidad de que ocurra el estímulo que generará el P300. Numerosos estudios han demostrado que la amplitud del potencial P300 aumenta de manera significativa cuando el usuario aumenta su nivel de atención y cuanto menor sea la probabilidad de aparición del estímulo objetivo. En la Figura 3-5 (a) se muestra el efecto de la probabilidad del estímulo en el P300.

Además, los efectos de la probabilidad del estímulo ocurren independientemente del conocimiento sobre ellos del sujeto. Si se modifica la probabilidad del estímulo durante un experimento y el usuario no es consciente de ello, la amplitud del potencial P300 cambia igualmente [29].

Aunque esté fuertemente correlado con la probabilidad de estímulo, otro aspecto importante es la probabilidad temporal, es decir, el tiempo que pasa entre la aparición de un estímulo objetivo y el siguiente. Se ha demostrado que cuanto mayor sea este tiempo, mayor será la amplitud obtenida en el potencial evocado P300, tal y como muestra la Figura 3-5 (b). Sin embargo, la amplitud es independiente de la secuencia de estímulos seguida en el experimento.

También influye la dificultad de la tarea en la forma del P300 obtenido. Cuando discriminar el estímulo objetivo del resto de estímulos se convierte en una tarea ardua, la amplitud del potencial P300 disminuye y su latencia aumenta, tal y como se puede observar en la Figura 3-6.

La disminución de amplitud es producida por una reducción de la confianza del usuario. La amplitud es mayor cuando éste se encuentra confiado porque la tarea es sencilla, sin embargo, al incrementarse la dificultad de la tarea la confianza en sí mismo disminuye, causando que la amplitud del potencial P300 caiga. No obstante, si la tarea se vuelve demasiado sencilla, la amplitud puede disminuir también, puesto que el usuario no dedicará toda su atención [29].

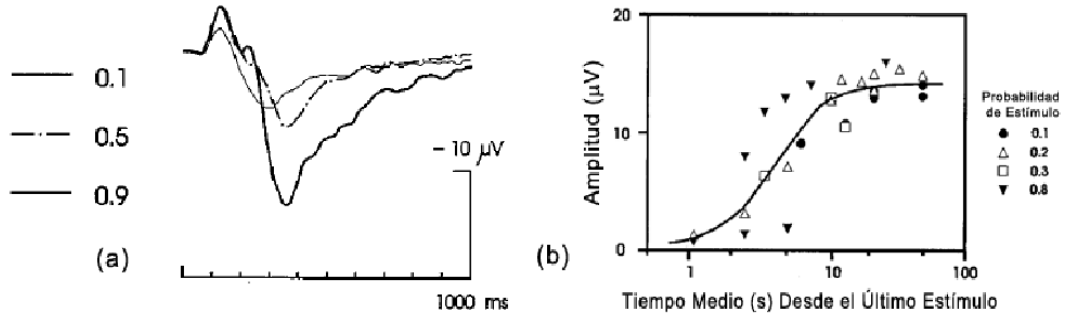

Figura 3-5. Variación de la forma del P300 para el electrodo Pz en función de la probabilidad de estímulo y la probabilidad temporal [29].

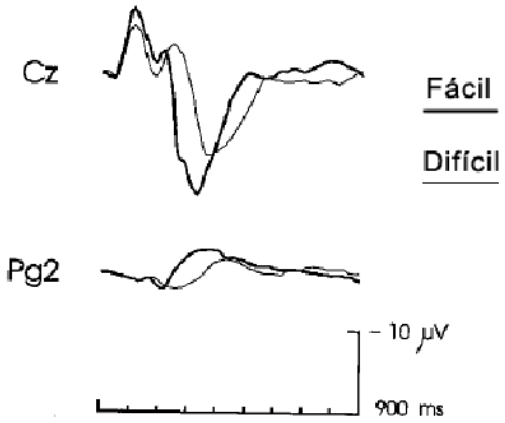

Figura 3-6. Potenciales evocados P300 recogidos en el vértex (Cz) y en un electrodo nasofaríngeo (Pg2) según la dificultad de discriminación de los estímulos objetivos [29].

Edad de los sujetos

Se ha demostrado que tanto la amplitud como la latencia del P300 se ven afectadas de manera lineal con la edad. En la etapa adulta, partir de los 20 años aproximadamente, la latencia aumenta una media de 1.3 ms por año con una desviación estándar de 31 ms mientras que la amplitud decrece. El comportamiento de estas dos características del P300 podemos observarlo en la figura 3-7 [28].

Sin embargo, el comportamiento del potencial P300 varía de forma distinta para los niños (entre 5 y 20 años) que para los adultos. Determinar el comportamiento del potencial P300 en niños es complicado debido al gran número de artefactos que presentan y a que normalmente no son capaces de mantener constante su atención, por lo que existen menos estudios.

El potencial evocado P300 para los niños se solapa con otra onda negativa producida en la zona frontocentral del córtex, más intensa cuanto más pequeño sea el niño. Sin embargo, en la zona parietal del córtex sí se muestra una onda parecida al P300 de los adultos, caracterizada por tener mayor latencia. Esta latencia disminuye a medida que aumenta la edad del niño, pudiendo distinguir dos tramos principales: entre los 5 y 12 años de edad la latencia disminuye con una tasa de 25ms por año, mientras que entre los 12 y 20 años de edad disminuye más lentamente, con una tasa de 1-5ms por año [29].

Adicionalmente, la distinción entre la onda P3b y la SW en niños se convierte en una tarea ardua y confusa. Con respecto a la amplitud, ésta aumenta a medida que aumenta la edad del niño hasta los 13 años, aproximadamente. Después de ese punto la amplitud decrece levemente como en los adultos. Los efectos de la probabilidad del estímulo en los adultos son similares a los encontrados en niños.

Para terminar, cabe destacar que la localización del potencial P300 en el córtex para los niños difiere de la localización de los adultos. A medida que los niños crecen, la localización del mismo

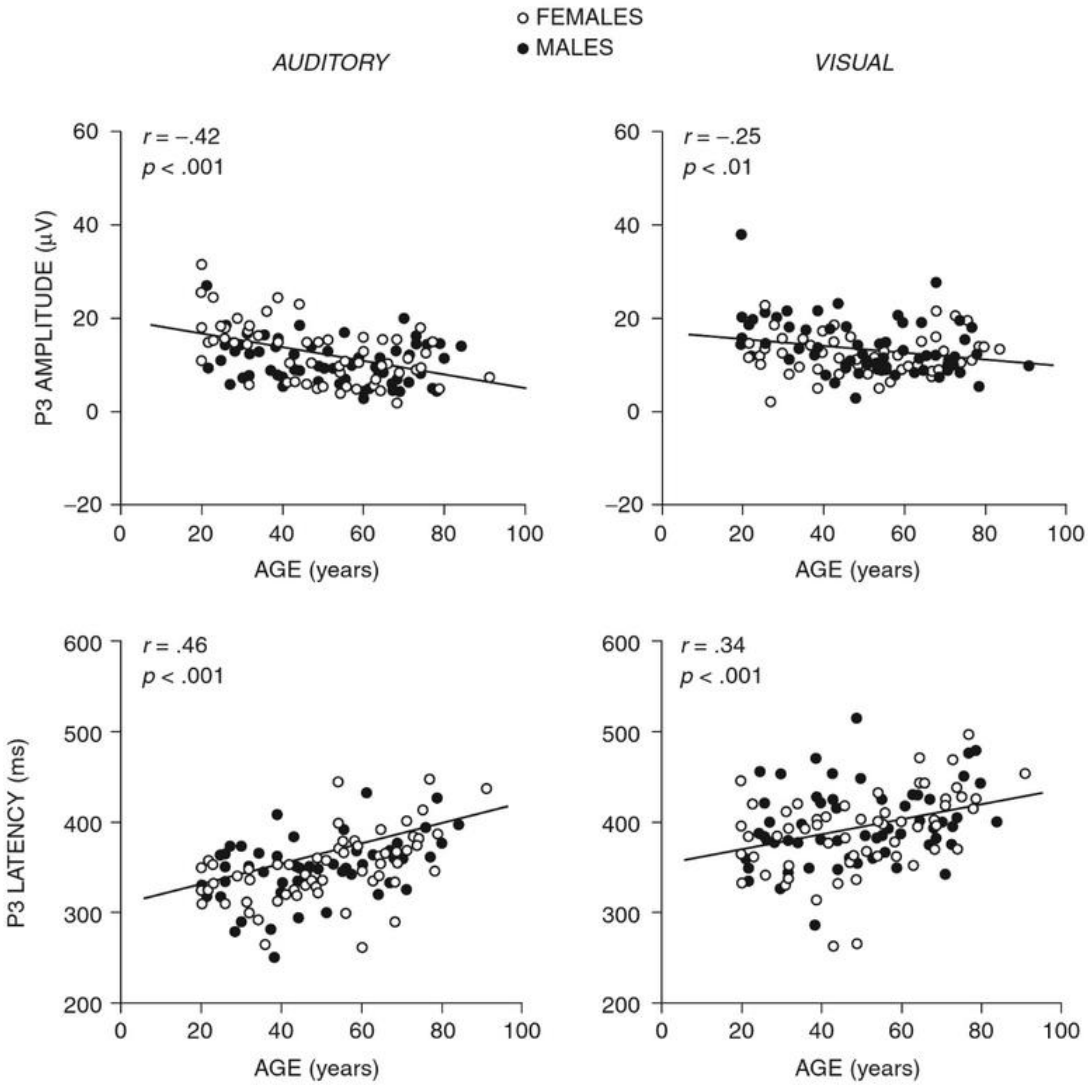

Figura 3-7. Diagramas de dispersión analizando la variación de la amplitud y la latencia del P300 en función de la edad a partir de los 20 años [28].

se desplaza desde la zona parietal del córtex a la zona frontocentral. Además de ello, la ubicación de los potenciales P300 visuales y auditivos en los adultos es similar, sin embargo, en los niños se obtienen mejor los potenciales P300 visuales en las zonas frontocentrales que los potenciales P300 auditivos [29].

Personalidad e inteligencia

Otros factores que pueden afectar al potencial P300 son la inteligencia, la personalidad, el oído y la vista (dependiendo de la si el estímulo objetivo es auditivo o visual), entre otros.

Considerando la inteligencia como el coeficiente intelectual clásico (CI), se ha visto que existe una correlación con la amplitud y la latencia del P300, disminuyendo ambos cuando el CI aumenta.

También se ha estudiado el efecto que pueda tener la personalidad y, concretamente, la extroversión, en las características la forma del P300. Sin embargo, los resultados no son concluyentes hay contradicciones entre unos autores y otros.

En el estudio de Hans Eysenck *et al* [32] los resultados dicen que cuanto más extrovertido sea el usuario, menor será el RT (*Response Time*) medido, posiblemente a causa de la impulsividad característica de éstos sujetos, además de mostrar una disminución de amplitud, posiblemente a causa de la imposibilidad de mantener la atención en largos intervalos de tiempo. También demuestran que el neuroticismo, o inestabilidad emocional, está negativamente correlado con la latencia, es decir, cuanto más inestable sea emocionalmente un sujeto menor será la latencia del P300 recogido. Finalmente, se concluye también que a medida que aumenta el psicoticismo (caracterizado por su agresividad y antipatía) menor será la amplitud del P300.

Sin embargo, el estudio de Cynthia Doucet y Robert M. Stelmack [33] llega a diferentes resultados. El experimento llevado a cabo con 67 sujetos, divididos en 3 grupos según su grado de extraversión, para medir la influencia de una personalidad extrovertida en los ERPs (concretamente en los P300) consistía en la realización de una serie de tareas midiendo el tiempo de respuesta (RT, *Response Time*) y el tiempo de movimiento (MT, *Movement Time*). Las conclusiones a las que se llegaron fueron que en todas las tareas de las propuestas menos en una, el RT de las personas extrovertidas aumentaba respecto a las personas introvertidas, mientras que en todas las tareas el MT disminuía. Esto parece indicar que las principales diferencias entre estos grupos de personas están en los procesos motores. Ver figura 3-8.

Por tanto, no se puede concluir con los estudios analizados los efectos que provoca la extroversión en las características del P300.

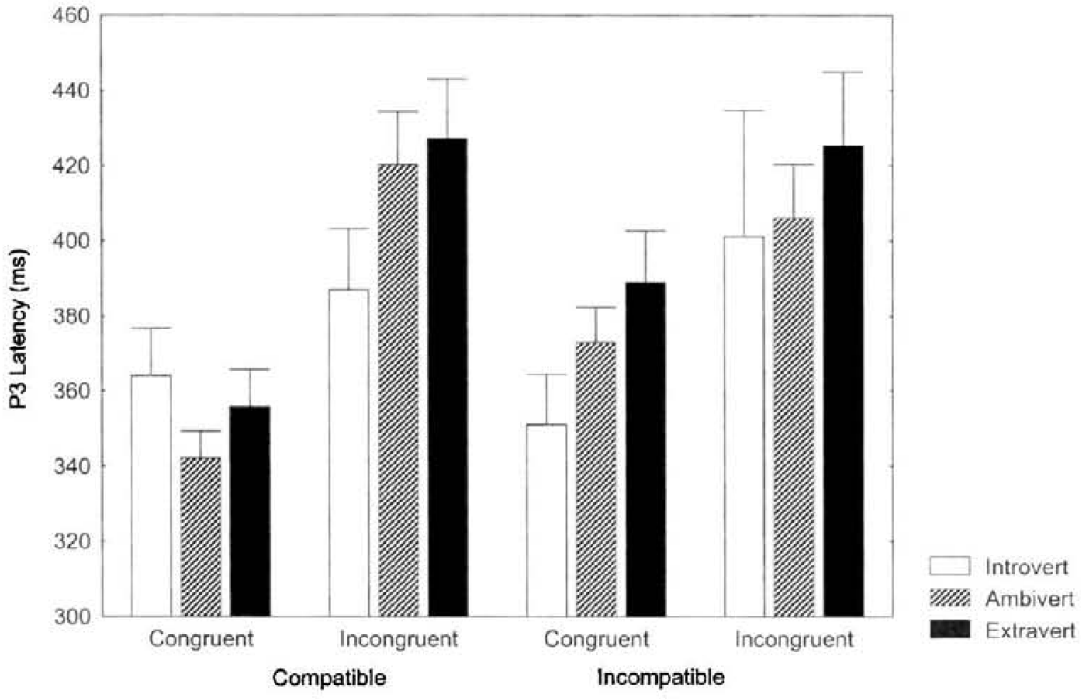

Figura 3-8. Latencia del P300 diferenciando entre los 4 tipos de tareas propuestas y 3 grupos de personas según el grado de extroversión de la persona [33].

Factores farmacológicos

Cuando el sujeto se encuentra bajo los efectos de ciertos medicamentos, del alcohol o de las drogas, la forma de sus potenciales evocados P300 varía.

Concretamente, los efectos del alcohol producen una reducción de amplitud del potencial P300 y un aumento de su latencia [29]. Sin embargo, este fenómeno no aparece cuando la dificultad de la tarea de discriminación entre los estímulos es complicada, probablemente debido a que el sujeto intenta superar los efectos del alcohol cuando la tarea lo requiere.

Los efectos de la nicotina también son relevantes, al ser una de las drogas con mayor consumo en el mundo. Además, la correlación entre el consumo de tabaco y el de alcohol es grande. El estudio realizado por Polich y Ochoa en 2004 [34] arroja varias conclusiones sobre el efecto de estas drogas sobre las características del P300, que se pueden resumir en la figura 3-9. En ella se puede ver como la amplitud del P300 para sujetos no fumadores con bajo riesgo de alcoholismo es mayor comparada con los que tiene un alto riesgo, decreciendo aún más si son fumadores habituales.

Los efectos producidos por las drogas y medicamentos son muy diversos, entre ellos destacan el efecto de la escopolamina (también conocida como *burundanga*), que disminuye drásticamente la amplitud del P300 hasta hacerlo desaparecer en *odd-ball* auditivos debido a la pérdida de memoria a corto plazo; el metilfenidato (psicoestimulante aprobado para el tratamiento de la hiperactividad), que aumenta el RT del sujeto; la clonidina (utilizado como antihipertensivo), las benzodiacepinas (medicamentos psicotrópicos) y los antihistamínicos, que atenúa la amplitud; y la dopamina, que disminuye la latencia; entre otros [29].

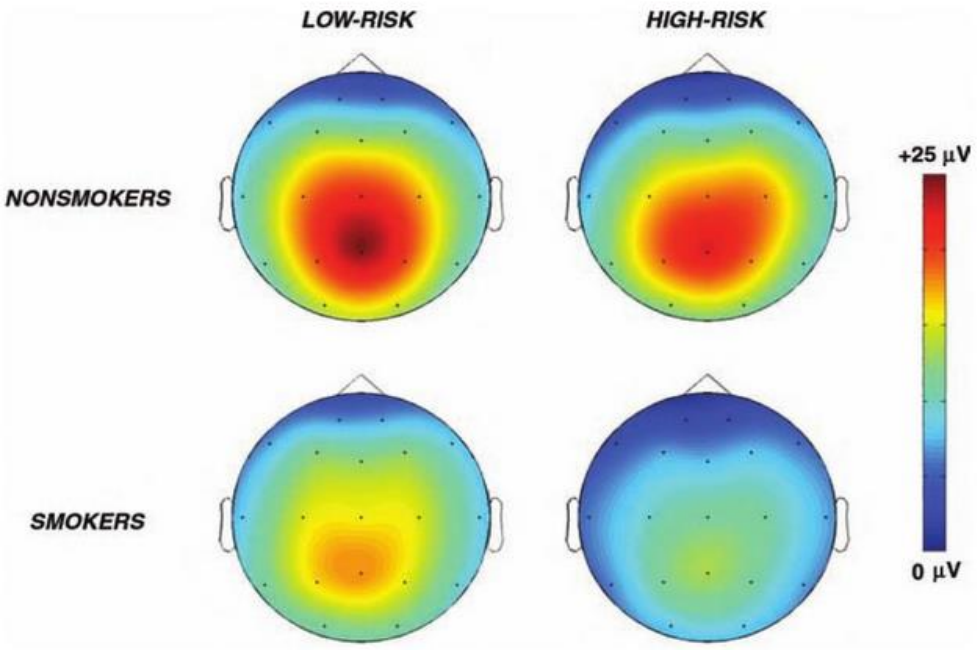

Figura 3-9. Amplitud mediia del P300 para sujetos fumadores o no con un alto o bajo riesgo de alcoholismo [28]

Enfermedades neuronales

Las enfermedades neuronales también afectan a la forma del potencial evocado P300. A continuación, se tratan los efectos de enfermedades como la esquizofrenia, el autismo, la depresión, el trastorno obsesivo-compulsivo, la psicopatía, la demencia o el Alzheimer.

Los sujetos que sufren de esquizofrenia generan potenciales P300 con muy baja amplitud, siendo prácticamente irreconocibles, tal y como se muestra en la Figura 3-6. Este fenómeno no parece estar relacionado con la falta de atención o la medicación sino con la severidad de los síntomas del paciente y, adicionalmente, los sujetos esquizofrénicos muestran un potencial P300 menos concentrado en la zona parietal del córtex que los sujetos de control [29].

El autismo y la depresión severa también afectan a la forma del potencial P300, causando una reducción de su amplitud. Sin embargo, en algunos tipos de pacientes psiquiátricos la latencia del P300 puede disminuir, como es el caso de los pacientes que sufren un trastorno obsesivo-compulsivo. Otro fenómeno característico consiste en obtener un potencial P300 mucho más prolongado, típico de los pacientes que sufren psicopatía o ALS. Para terminar, también se ha demostrado que la dislexia y la demencia producen una disminución de amplitud acompañada de un aumento en la latencia del potencial [29].

Otra enfermedad que afecta a las características del P300 es el Alzheimer. Numerosos estudios han encontrado diferencias en la forma del P300 entre sujetos con la enfermedad y de control. Las mayores diferencias se han observado en la realización de tareas relativamente fáciles, especialmente para la amplitud siendo esta menor. Sin embargo, las diferencias en la latencia varían de unos estudios a otros, no siendo concluyentes y dependiendo de la tarea realizada. En la figura 3-10 podemos ver estas diferencias [28].

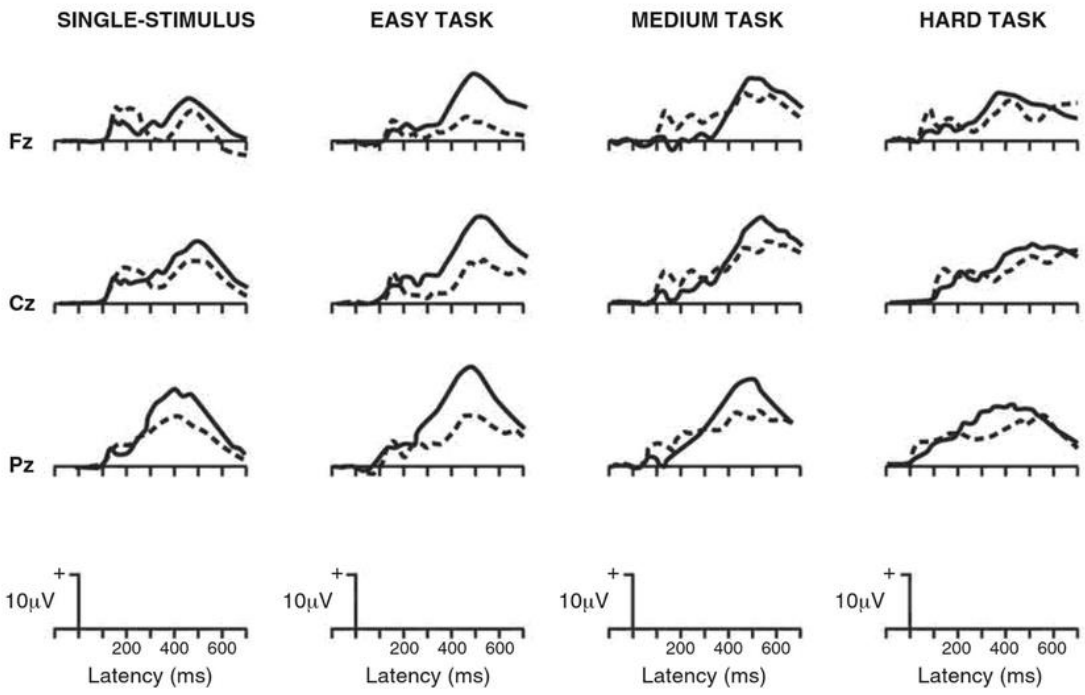

Figura 3-10. Representación de la forma del P300 para estímulos visuales para sujetos con Alzheimer (línea discontinua) y de control (línea continua) [28].

Resumen de los factores que afectan a las características del P300

La tabla 3-1 contiene un resumen de la información desglosada en este punto del capítulo ampliada con algunos otros factores que no se analizaran en profundidad [28].

Factores naturales			
FACTOR	**AMPLITUD**	**LATENCIA**	**COMENTARIOS**
Aumento Temperatura corporal	No afecta	Desciende	
Aumento de pulso cardiaco	No afecta	Desciende	
Ingesta de alimentos	Aumenta	Desciende ligeramente	
Genética			Latencia y amplitud del P300 están muy determinadas por la información genética
Factores psicológicos			
Aumento de atención	Aumenta	Disminuye	
Aumento de probabilidad de estimulo	Disminuye	No afecta	
Aumento de tiempo entre estímulos	Aumenta	No afecta	
Aumento en la sencillez de la tarea	Aumenta	Disminuye	Si las tareas son demasiado sencillas acaba por disminuir
Factores farmacológicos			
Alcohol	Disminuye	Aumenta	Su efecto se ve reducido en tareas complicadas
Escopolamina	Disminuye		La amplitud casi desaparece
Clonidina, Benzodiacepina, Antihistamínimos	Disminuye		
Dopamina		Disminuye	
Cafeína	Aumenta	Disminuye	Los efectos se potencian si el sujeto está fatigado
Nicotina	Disminuye	Disminuye	

Diferencias individuales			
Aumento de edad en adultos (> 20 años)	Disminuye	Aumenta	El potencial se desplaza a la zona frontal
Aumento de edad en niños (5-20 años)	Aumenta	Disminuye	La onda parietal disminuye
Aumento del Coeficiente intelectual (CI)	Disminuye	Disminuye	
Aumento de la extroversión			Contradicciones en los autores
Aumento del Neuroticismo		Disminuye	
Aumento del Psicoticismo	Disminuye		
Enfermedades neuronales			
Esquizofrenia	Disminuye		La amplitud prácticamente desaparece
Autismo y depresión severa	Disminuye		La onda parietal disminuye
Trastorno obsesivo-compulsivo		Disminuye	
Psicopatía y ALS	Disminuye		Potencial mas prolongado
Dislexia y demencia	Disminuye	Aumenta	
Alzheimer	Disminuye	Aumenta	La diferencia se acentúa en tareas fáciles

Tabla 3-1. Resumen de los factores que afectan a las características de los P300 [28].

3.2. EXTRACCIÓN DE LAS CARACTERÍSTICAS DE LOS POTENCIALES P300

Una vez adquirida la señal y eliminados los principales artefactos, es el momento de extraer las características de los P300 que nos permitirán saber si hay P300 o no en un determinado instante. Esta tarea no es del todo independiente de la eliminación de artefactos, pues seguiremos aplicando técnicas que nos permitan eliminar componentes no deseadas de la señal EEG para quedarnos solo con la información útil.

En sistemas BCI, donde la carga computacional debe ser baja ya que funcionan en tiempo real, se busca la simplicidad de los métodos. La extracción de características se lleva a cabo principalmente mediante dos técnicas, el filtrado espacial y el filtrado temporal o frecuencial. En los siguientes subapartados se desarrollarán en detalle los métodos utilizados en sistemas BCI para ambas técnicas, las cuales se aplican conjuntamente como normal general.

3.2.1. Filtrado espacial

El objetivo de realizar un filtrado espacial es reducir el difuminado espacial, que es un efecto provocado por la distancia entre los electrodos y las fuentes que generan la señal dentro el cerebro debido a la no homogeneidad de los tejidos que se encuentran entre ellos.

En este punto analizaremos los métodos más populares para reducir este difuminado espacial en sistemas BCI, como son los filtros Laplacianos, los métodos de referencia de media común (CAR), el análisis de componentes independientes (ICA), y el análisis de componentes principales (PCA) [17].

Filtros Laplacianos

EL filtro Laplaciano estudia la segunda derivada de la distribución de voltaje espacial instantánea de un determinado electrodo, permitiendo analizar las variaciones de la señal en un radio concreto y reduciendo la actividad más difusa captada en el electrodo, que será consecuencia del difuminado espacial.

Para calcular el filtro se combina la señal del electrodo con la de los adyacentes. La distancia a estos va a determinar la frecuencia de corte espacial del filtro, que se comporta como un filtro espacial paso-alto. Si la distancia a los electrodos es pequeña, la frecuencia de corte se eleva dejando pasar solo la señal muy localizada, mientras que si la distancia es más grande permitirá el paso de señales menos localizadas bajo el electrodo [35].

Para obtener el filtro se utiliza un método de diferencias finitas que aproxima la segunda derivada de la distribución de voltaje en dos dimensiones en un electrodo, restando la actividad media de los adyacentes, eliminando así las componentes comunes. Las ecuaciones utilizadas para el cálculo del filtro son [17]:

$$v_h^{LAP}(t) = v_h(t) - \sum_{i \epsilon S_i} w_{h,i} * v_i(t) \quad (3.1)$$

$$w_{h,i} = \frac{\frac{1}{d_{h,i}}}{\sum_{i \epsilon S_i} \frac{1}{d_{h,i}}} \quad (3.2)$$

Donde $v_h(t)$ es la señal de voltaje recogida en el electrodo donde aplicamos el filtrado, $v_i(t)$ la señal de voltaje de los electrodos adyacentes y $w_{h,i}$ es el peso de cada electrodo,

que es función de la distancia, $d_{h,i}$, entre el electrodo h e i. S_i es el conjunto de índices de electrodos contiguos que consideramos en el filtrado.

En función de la distancia entre el electrodo de interés y los adyacentes considerados en el filtrado tendremos dos tipos de filtrado Laplaciano, el Laplaciano corto y el Laplaciano largo (figura 3-11a y 3-11b). Si la distancia es la misma para el conjunto de electrodos contiguos, el peso de cada uno será el mismo. Si se consideran 4 electrodos, el peso de cada uno será $w_{h,i} = 0.25$.

Método de referencia común (CAR)

El método de referencia de media común (CAR, *Common Average Reference*) se basa en el mismo procedimiento que el filtro Laplaciano, sin embargo, difiere de éste en que se resta de la señal obtenida la media común de las señales de todos los electrodos. Si la cabeza está completamente cubierta de electrodos equiespaciados entre ellos y el potencial generado en el córtex está muy localizado en fuentes puntuales, el método CAR provee un voltaje espacial de media cero, además de actuar como un filtro espacial paso-alto, acentuando las componentes que se encuentran más localizadas [35]. En la Figura 3-11 (c) se pueden observar los electrodos involucrados.

El voltaje del canal deseado se obtiene restando a éste la media común del voltaje obtenido en el resto de electrodos:

$$v_h^{CAR}(t) = v_h(t) - \frac{1}{H}\sum_{i=1}^{H} v_i(t)$$

Donde $v_h(t)$ es el voltaje recogido en el electrodo cuya señal queremos filtrar y $v_i(t)$ son el resto de electrodos considerados en el filtrado [17].

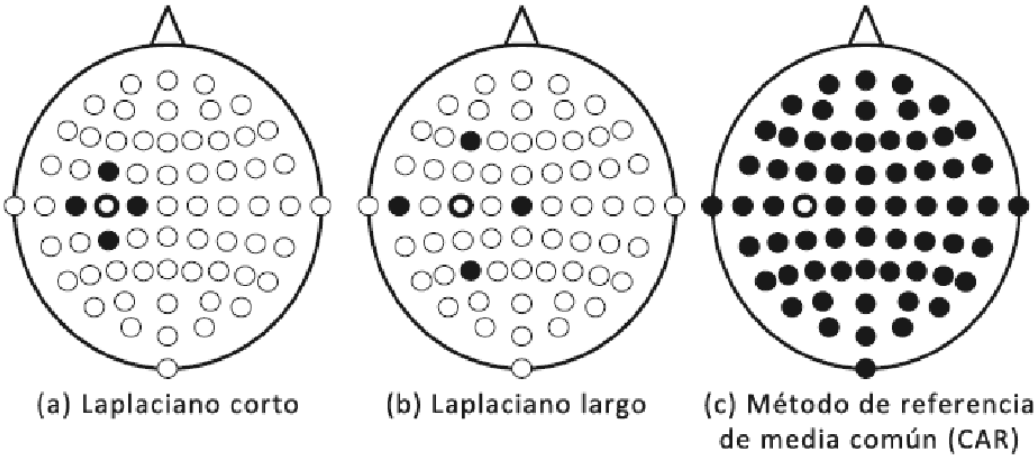

Figura 3-11. Posición de los electrodos considerados en tres tipos de filtrado espacial para la obtención de la señal en C3: (a) Laplaciano corto, (b) Laplaciano largo y (c) método de referencia común (CAR) [17].

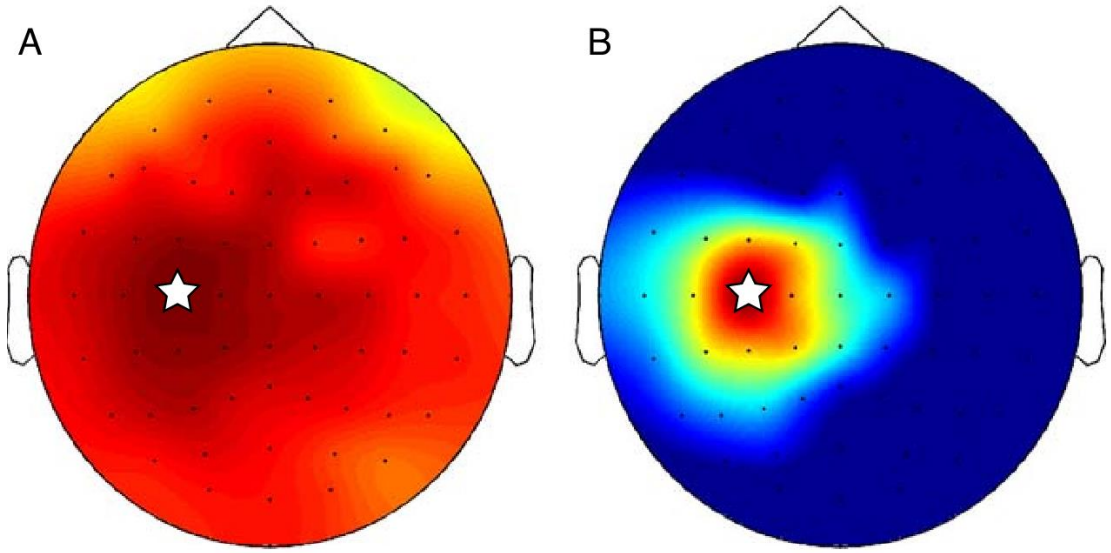

Figura 3-12. Ejemplo del efecto de un filtrado espacial CAR sobre el electrodo C3. La señal gana en resolución espacial facilitando la detección del P300 [17].

En la figura 3-12 podemos ver el efecto que produce un filtro espacial CAR sobre una señal EEG que se recoge en el electrodo C3. Vemos como mejora la resolución espacial, eliminando las señales comunes que no tienen interés (artefactos oculares, artefactos por movimientos musculares, etc) [17].

Análisis de componentes independientes

Como se ha comentado en capítulos anteriores, la señal EEG es una superposición de muchas señales diferentes producidas por nuestro organismo simultáneamente.

La extracción de características en un sistema BCI tiene como objetivo la detección de una señal en concreto, en nuestro caso el potencial P300, para poder controlarlo. Con este objetivo aplicamos el análisis de componentes independientes (ICA, *Independent Component Analysis*), que es una solución estadística particular al problema de la separación de fuentes, consistente en obtener las señales originales de un proceso a partir de una mezcla de dichas señales. Por tanto, ICA trata de subdividir la señal EEG en diversas componentes independientes, suponiendo que las señales de origen tienen una distribución no gaussiana y son estadísticamente independientes [36].

El objetivo de esta técnica es el de revelar características del EEG ocultas por el ruido basal para detectar los potenciales P300 con un solo intento (Li [36] *et al* han conseguido precisiones del 76.67%).

Un ejemplo para entender este método es el presentado por Li *et al* [36] en su artículo. Supongamos que dos personas (S_1 y S_2) están hablando en un extremo de una habitación simultáneamente y en el otro extremo se encuentran dos micrófonos (X_1 y X_2) en distintas localizaciones que registran la conversación. Las señales obtenidas podrían expresarse de la siguiente manera:

$$\begin{cases} X_1(t) = a_{11}S_1(t) + a_{12}S_2(t) \\ X_2(t) = a_{21}S_1(t) + a_{22}S_2(t) \end{cases} \quad (3.4)$$

Si conociésemos los valores de a_{11}, a_{12}, a_{21} y a_{21} podríamos resolver esas ecuaciones para obtener los valores de $S_1(t)$ y $S_2(t)$, y así distinguir ambas conversaciones sin solapamiento alguno. Desgraciadamente, esos pesos son desconocidos y únicamente podemos obtenerlos si consideramos que las fuentes ($X_1(t)$ y $X_2(t)$) son señales independientes y de naturaleza no gaussiana.

Este ejemplo se conoce como el problema *cocktail party*, y es el mismo que intenta resolver el método ICA. La señal EEG también se considera un problema *cocktail party*, puesto que los electrodos registran la señal EEG en distintas localizaciones sobre el cuero cabelludo. Con ICA se intentan distinguir aquellas fuentes que producen esa mezcla y, concretamente, las que causan los potenciales evocados P300 para facilitar su posterior detección [36].

Aplicando la notación convencional del algebra lineal al anterior sistema de N ecuaciones, tenemos que:

$$\boldsymbol{x} = \boldsymbol{A} * \boldsymbol{s} \tag{3.5}$$

Siendo el vector vertical $\boldsymbol{x}$ de las señales mezcladas y conocidas recogidas en cada uno de los electrodos, $\boldsymbol{A}$ la matriz que contiene los pesos a_{ij} y $\boldsymbol{s}$ el vector de fuentes que se desean discriminar. Se asume que los coeficientes de pesos son desconocidos y que permiten que la matriz sea invertible, además de asumir (como ya se ha indicado,) que las componentes son no gaussianas e independientes, que la mezcla es lineal y espontánea y que los datos son estacionarios.

Bajo estas suposiciones, se concluye que existe una matriz $\boldsymbol{W}$ con coeficientes w_{ij} que permite obtener las fuentes estimadas $\hat{\boldsymbol{s}}$ al multiplicarse con las señales mezcladas, de la siguiente manera:

$$\boldsymbol{y} = \boldsymbol{W} * \boldsymbol{x} = \boldsymbol{W} * [\boldsymbol{A} * \boldsymbol{S}] = \hat{\boldsymbol{s}} \xrightarrow{\boldsymbol{yields}} \hat{\boldsymbol{s}} = \boldsymbol{W} * \boldsymbol{x} \tag{3.6}$$

Por tanto, la matriz $\boldsymbol{W}$ es la matriz pseudoinversa de $\boldsymbol{A}$. Existen una gran variedad de algoritmos para hallar la matriz $\boldsymbol{W}$, entre ellos *Infomax*, *JADE* o *FastICA*, siendo *Infomax* el que mejores resultados presenta [36].

Análisis de componentes principales

El método de análisis de componentes principales (PCA, *Principal Component Analysis*) para el filtrado espacial, basa su funcionamiento en la selección de las componentes principales de una señal EEG para reducir su dimensionalidad, eliminando la información redundante que se capta en los electrodos.

Para ello se analiza el espacio geométrico de n dimensiones, tantas como fuentes haya de señal, que componen los datos. Para este espacio se determinan las direcciones que, al proyectar los datos sobre ellas, hacen maximizar la varianza de los datos proyectados.

En la figura 3-13 podemos observar un ejemplo para un espacio geométrico de dimensión dos. Se puede ver a simple vista como la dirección que maximiza la varianza de los datos proyectados es D1, mientras que D2 es una dirección cualquiera [37]. Por tanto, proyectando los datos sobre D1, conseguiremos una compresión de la información en la dirección que asegura menor pérdida de información al maximizar la varianza.

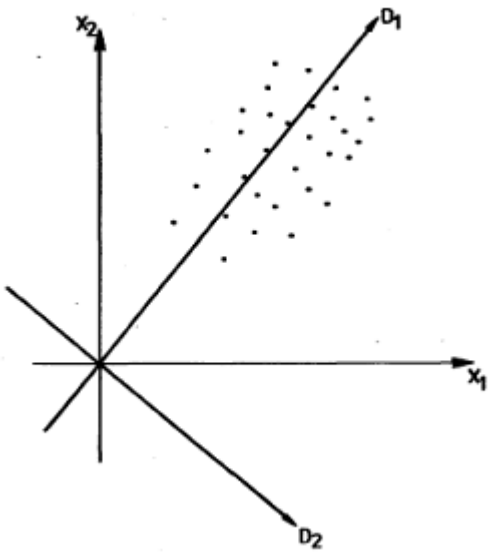

Figura 3-13. Búsqueda de la dirección que consigue que la varianza de los datos proyectados sea máxima (D_1) en un problema de análisis de componentes principales [37].

Para un caso de mayor dimensionalidad (mas electrodos adquiriendo señal) el proceder es el mismo. Sabiendo que el espacio geométrico inicial tiene *n* dimensiones, se pueden seleccionar las *m* direcciones principales que minimizan la pérdida de información, con *n* mayor que *m,* y proyectar los datos sobre ellas para reducir la dimensionalidad del espacio geométrico.

3.2.2. Filtrado y análisis temporal

El análisis en tiempo real de una señal EEG requerido en aplicaciones BCI tiene un alto coste computacional. La necesidad de aplicar métodos de extracción de características sencillos, incluso a costa de perder algo de profundidad en el análisis de la señal, se hace primordial en este tipo de sistemas.

En este contexto el filtrado temporal tiene una gran respuesta, siendo las técnicas más utilizadas el promediado-sincronizado, la detección de picos (*peak picking)*, el cálculo del área de la señal y el filtrado adaptativo.

Algunos de estos métodos ya han sido introducidos, como el promediado-sincronizado, ya que también se aplican en la parte de adquisición de la señal para la eliminación de artefactos.

Método de promediado sincronizado

El método del promediado sincronizado de distintas épocas de señal EEG permite distinguir los potenciales evocados P300 del ruido basal de fondo, según se ha demostrado en numerosos estudios.

Por ejemplo, el paradigma *oddball* presentado por E. Donchin y Farwell utiliza esta técnica para detectar o no la presencia de P300 en una señal EEG y determinar la selección del usuario [18].

Este método es sencillo y de fácil aplicación, pero requiere la adquisición de varias épocas de realización de la señal tras aplicar de manera repetitiva un estímulo y poseer una referencia temporal para alinearlas adecuadamente.

Matemáticamente, la aplicación es sencilla. Se promedian todas las épocas de señal registradas, estableciendo la referencia temporal en el momento en el que se genera el estímulo que produce el P300 a detectar, según la ecuación (3.7) [30].

$$Prom(n) = \sum_{k=1}^{M} y_k(n) = \sum_{k=1}^{M} x_k(n) + \sum_{k=1}^{M} w_k(n), \quad n = 1,2,\dots,N \tag{3.7}$$

Donde las señales $y_k(n)$ son las realizaciones de la señal, cuyas componentes son $x_k(n)$, señal original sin ruido, y $w_k(n)$ el ruido incorporado en la k-ésima época.

N representa el número de muestras que tiene cada época, M el número de épocas y k el índice promediado.

En el caso de la aplicación de la matriz *odd-ball* que genera P300 para la selección de letras, conocida como *P3Speller*, la manera de detectar qué celda se ha seleccionado es realizando un promediado sincronizado. El funcionamiento se verá en el siguiente ejemplo.

Recordemos el funcionamiento del paradigma. Cada celda de la matriz *oddball* se ilumina un determinado número de veces, provocando varios potenciales P300 si el usuario está prestando atención a una celda concreta. Estos potenciales aparecerán, de media, unos 300 ms después de cada iluminación.

Imaginemos un caso muy sencillo con una matriz de dos celdas cuyas iluminaciones no están distribuidas temporalmente de manera aleatoria y uniforme, siendo estas en un instante temporal fijo. Este ejemplo en la realidad no funcionaría, ya que recordemos que estas simplificaciones afectarán muy negativamente a la amplitud del P300, que es mayor cuanta menor sea la probabilidad del estímulo. Sin embargo, nos servirá para para entender la aplicación del promediado sincronizado en este paradigma.

El experimento comienza instruyendo al sujeto para que se fije en una de las celdas de la matriz. Estas celdas se iluminarán una vez cada una (una época). La primera en 0 ms y la segunda en 200 ms. Se comienza a adquirir datos en el momento del primer del estímulo. Si el usuario está prestando atención a la primera celda, el P300 aparecerá en $t = 300\,ms$, mientras que si su atención esta fija en la segunda celda, el P300 aparecerá en $t = 200 + 300 = 500\,ms$.

Si representamos la señal adquirida durante una época, será imposible distinguir el P300 entre el ruido basal y los artefactos que interfieren de manera aleatoria, ya que estos tienen mayor amplitud. Sin embargo, si repetimos el proceso N veces y promediamos las épocas, las componentes aleatorias ruidosas que “manchan” la señal se cancelan entre sí, apareciendo el potencial P300, que no es una componente aleatoria al estar generada por el estímulo en un determinado instante. Ver la figura 3-14, ya expuesta en el capítulo 2, para una mejor comprensión.

Para una detección automática del instante en el que han ocurrido los P300 y poder averiguar cuál es la celda seleccionada se aplicarán los métodos de detección de picos o cálculo del área tras el promediado sincronizado [38].

- La detección de picos es una técnica muy simple que trata de detectar los potenciales P300 monitorizando la amplitud de un promediado sincronizado de la señal. Para ello, se determina la diferencia entre el punto más negativo previo a la ventana establecida y el punto más alto dentro de dicha ventana [18] y se compara con la obtenida para un

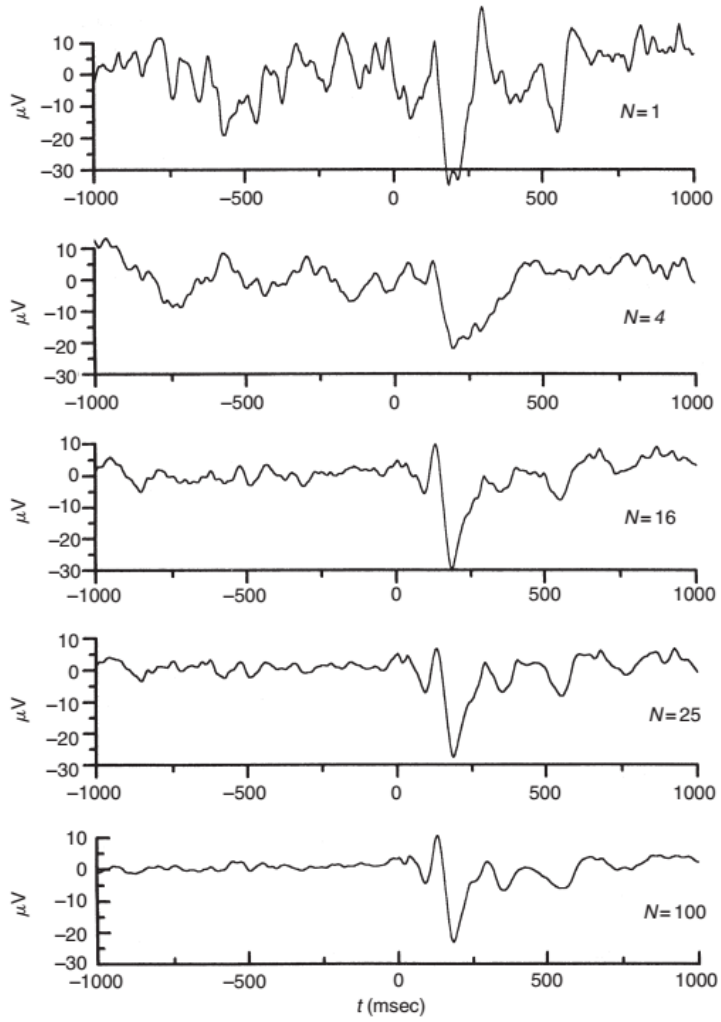

Figura 3-14. Efecto de un promediado sincronizado de varias muestras de una señal EEG que contiene un potencial evocado (EP). El estímulo se realizó en t=0. Se puede observar como el ruido se reduce de manera muy notoria, permitiendo la detección del potencial con mucha precisión [28].

segmento basal. Los tamaños de la ventana son muy diversos dependiendo de la latencia del sujeto en cuestión, normalmente comprendiendo desde cientos de milisegundos antes de cada estímulo hasta más de 500ms después de éste, para que contenga el potencial evocado, típicamente 300ms después del estímulo.

- El cálculo del área comprendida bajo la señal promediada es un método muy sencillo y útil para detectar la presencia de este tipo de potenciales. En la práctica se suman todos los puntos de la ventana establecida y se determina la presencia de un P300 si el área comprendida es drásticamente mayor que en un segmento basal [18].

Filtrado adaptativo

El filtrado adaptativo no suele utilizarse frecuentemente en las aplicaciones que utilizan los potenciales evocados P300 ya que, debido a su estacionariedad, no ofrece buenos resultados.

Sin embargo, es útil para eliminar artefactos conocidos (e.g., el ECG residual). El filtrado adaptativo elimina un artefacto de la señal EEG sin asumir estacionariedad en la señal, no obstante, requiere una señal fuertemente correlada con el ruido a eliminar [30].

Tal y como indica su nombre, este filtro se adapta continuamente a las características de la señal, cambiando el vector de pesos a lo largo del tiempo, como se observa en la Figura 3-15. Para ello partimos de una señal de entrada $x(n)$, mezcla de la señal de interés $v(n)$ y del ruido primario $m(n)$, asumiendo que ambas componentes están incorreladas, como expresa la ecuación (3.8).

El filtro adaptativo LMS (*Least Mean Squares*¸ o Mínimos Cuadrados Promediados) filtra la referencia $r(n)$, fuertemente correlada con $m(n)$, para obtener una señal $y(n)$ lo más parecida al ruido primario que sea posible. Posteriormente, $y(n)$ es sustraída de la señal, obteniendo la estimación de la señal de interés $e(n)$, expresado en la ecuación (3.9).

$$x(n) = v(n) + m(n) \tag{3.8}$$

$$e(n) = \hat{v}(n) = x(n) - y(n) \tag{3.9}$$

El algoritmo LMS trata de ajustar el vector de pesos $\boldsymbol{w}$ para minimizar el error cuadrático medio (MSE, *Mean Square Error*) calculando un nuevo vector de pesos $\boldsymbol{w}\ (n+1)$ en base al vector de pesos actual $\boldsymbol{w}(n)$ más una correlación proporcional al gradiente negativo MSE. La ecuación (3.10) es una particularización de la regla de Widrow-Hoff y se encarga de actualizar los pesos en cada momento.

$$e^2(n) = x^2(n) - 2x(n)\boldsymbol{r^T}(n)\boldsymbol{w}(n) + \boldsymbol{w^T}(n)\boldsymbol{r}(n)\boldsymbol{r^T}(n)\boldsymbol{w}(n) \tag{3.10}$$

Este algoritmo es fácil de implementar y no requiere derivar, elevar al cuadrado o promediar, aunque se base en minimizar el MSE. También puede utilizarse el algoritmo RLS (*Recursive Least Squares*, o Mínimos Cuadrados Recursivos), óptimo cuando las señales varían mucho en oscilaciones rápidas de tiempo [30].

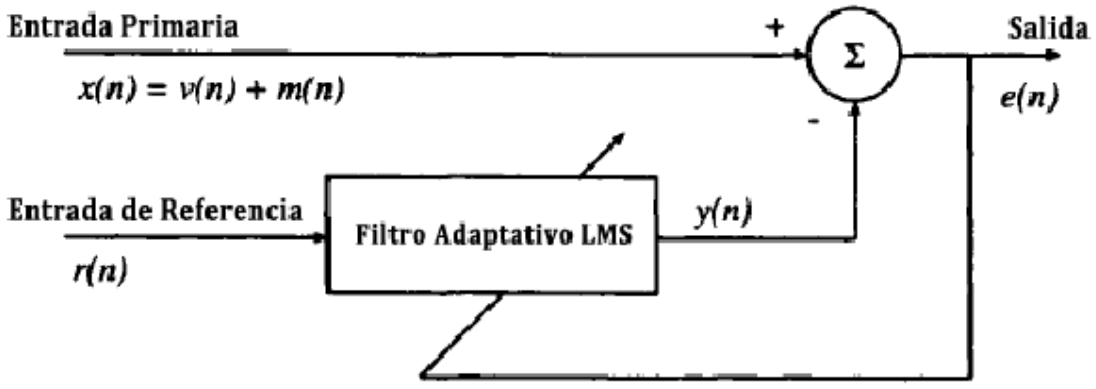

Figura 3-15. Esquema del filtrado adaptativo para eliminar una fuente de ruido [30].

3.2.3. Análisis tiempo - frecuencia

Mientras que las técnicas de filtrado temporal están más enfocadas a el procesado de la señal en tiempo real y la implementación de sistemas BCI, las técnicas de análisis espectral son ampliamente utilizadas en análisis *offline* de la señal EEG.

EL uso de la STFT (*Short-Time Fourier Transform*), las transformadas *Wavelet* y los métodos autorregresivos son los más utilizados y los que se verán en este apartado.

Short-Time Fourier Transform

Este método consiste en calcular la transformada discreta de Fourier clásica de la señal EEG enventanada en tiempo, analizando la señal en periodos finitos que se suponen estacionarios.

Para ello, debemos de multiplicar la señal EEG por una función ventana que seleccione las muestras de los instantes temporales que queremos estudiar, de la siguiente manera:

$$x_w[n] = x[n] \cdot w[n] \qquad (3.11)$$

Siendo $x_w[n]$ la señal enventanada, $x[n]$ la señal original y $w[n]$ la ventana utilizada. Las ventanas temporales más típicas son la rectangular, la de Hamming, la de Hanning, y la de Blackman. En el dominio de la frecuencia, según las propiedades de la transformada de Fourier, el espectro de la señal enventanada es:

$$X_w[k] = X[k] * W[k] \qquad (3.12)$$

Esta convolución en el dominio frecuencial de la señal original con el espectro de la ventana produce dos efectos negativos, la disminución de la resolución espectral y el manchado o *leakage.*

La resolución espectral se ve afectada por el ensanchamiento de las frecuencias que produce un enventanado en tiempo. Esto provoca que sea más difícil diferenciar componentes frecuenciales cercanas.

El *leakage* es la distorsión producida por los lóbulos secundarios de la función ventana, que provoca la pérdida de componentes frecuenciales de baja energía, que quedan solapadas por los lóbulos secundarios de las frecuencias con más energía.

Entre estos dos efectos negativos, hay un compromiso de forma que al aumentar la resolución espectral aumenta también el manchado, y viceversa.

Si mantenemos fija la resolución espectral y calculamos la STFT de una señal dividida en N_T segmentos no solapados, la expresión para el cálculo de la STFT está dada por (3.13).

$$X_w[n,k] = \sum_{m=0}^{N-1} x[m] \cdot w^*[m-n] \cdot \exp\left(-\frac{j2\pi km}{N}\right) \text{ con } \begin{matrix} k \in [0\,,\ L-1] \\ n \in [0,\ N_T-1] \end{matrix} \qquad (3.13)$$

Sin embargo, la resolución fija de esta técnica en el plano tiempo-frecuencia (Figura 3-16 (a)) que presenta su espectrograma una vez fijada la ventana, hacen que otras técnicas como las transformadas tipo *wavelet* sean mucho más adecuadas en el análisis de señales biomédicas, que dependiendo del instante temporal necesitaran una resolución u otra.

Transformadas Wavelet

Las características de las señales biomédicas, y en especial las del EEG hacen que la STFT no sea una técnica adecuada para el análisis espectral de estas señales. Los resultados obtenidos con una resolución espectral fija para todas las frecuencias utilizando las ventanas clásicas no son buenos, lo que provocó el desarrollo de las transformadas *wavelet*.

Todas las técnicas *wavelet* que se exponen en este punto se basan en la traslación y el escalado de una función *wavelet* madre (ecuación 3.14), generando una resolución espectral variable en el plano tiempo-frecuencia.

$$\varphi_{\tau,s} = \frac{1}{\sqrt{s}} \cdot \varphi\left(\frac{t-\tau}{s}\right) \qquad (3.14)$$

Donde $\varphi_{\tau,s}$ es la función *wavelet* madre, τ es el parámetro de traslación y s el parámetro de escalado. Existe un gran conjunto de *wavelets* madre y, habitualmente, el rendimiento de la aplicación se basa principalmente en elegir correctamente la más indicada; entre ellas se encuentran la *wavelet* de Haar, de Morlet, de Meyer y la de de Daubechies.

Este tipo de análisis mejora la resolución espectral de las frecuencias bajas, en decremento de la resolución temporal, mientras que para frecuencias altas prima la resolución temporal, atendiendo al comportamiento característico de las señales biomédicas en general [3].

Existen varios tipos de transformadas *wavelet* que permiten el análisis en el plano tiempo-frecuencia. Entre ellos están la transformada *wavelet* continua (CWT, *Continous Wavelet Transform)*, la transformada de *wavelet* discreta (DWT, *Discret, Wavelet, Transform)* y la transformada de paquetes *wavelet* (WPT, *Wavelet Packet Transform*).

La transformada CWT introduce la ya comentada variación de la resolución frecuencial y temporal, aumentando la frecuencial para las bajas frecuencias y la temporal para las oscilaciones rápidas. La comparación entre un análisis con CWT y STFT puede verse en la figura 3-16.

La transformada DWT muestrea los valores de los parámetros continuos de translación, τ, y escalado, s, eliminando información redundante y mejorando el coste computacional de las operaciones necesarias para calcularla.

No obstante, los resultados más precisos los provee la transformada de paquetes *wavelet* (WPT, *Wavelet Packet Transform*). La diferencia respecto a la DWT radica en que ésta nueva transformada, aparte de la señal de aproximación, subdivide también la señal de detalle, creando un árbol de descomposición mucho más extenso. Este análisis multiresolución extendido permite cambiar la resolución del plano tiempo-frecuencia tal y como desee el usuario a cambio de requerir un mayor coste computacional.

Las diferencias entre los métodos STFT, DWT y WPT son apreciables en la figura 3-17. En el primer caso vemos como la STFT mantiene una resolución fija para todas las frecuencias. Sin embargo, la DWT y la WPT varían esas resoluciones según lo comentado anteriormente [3].

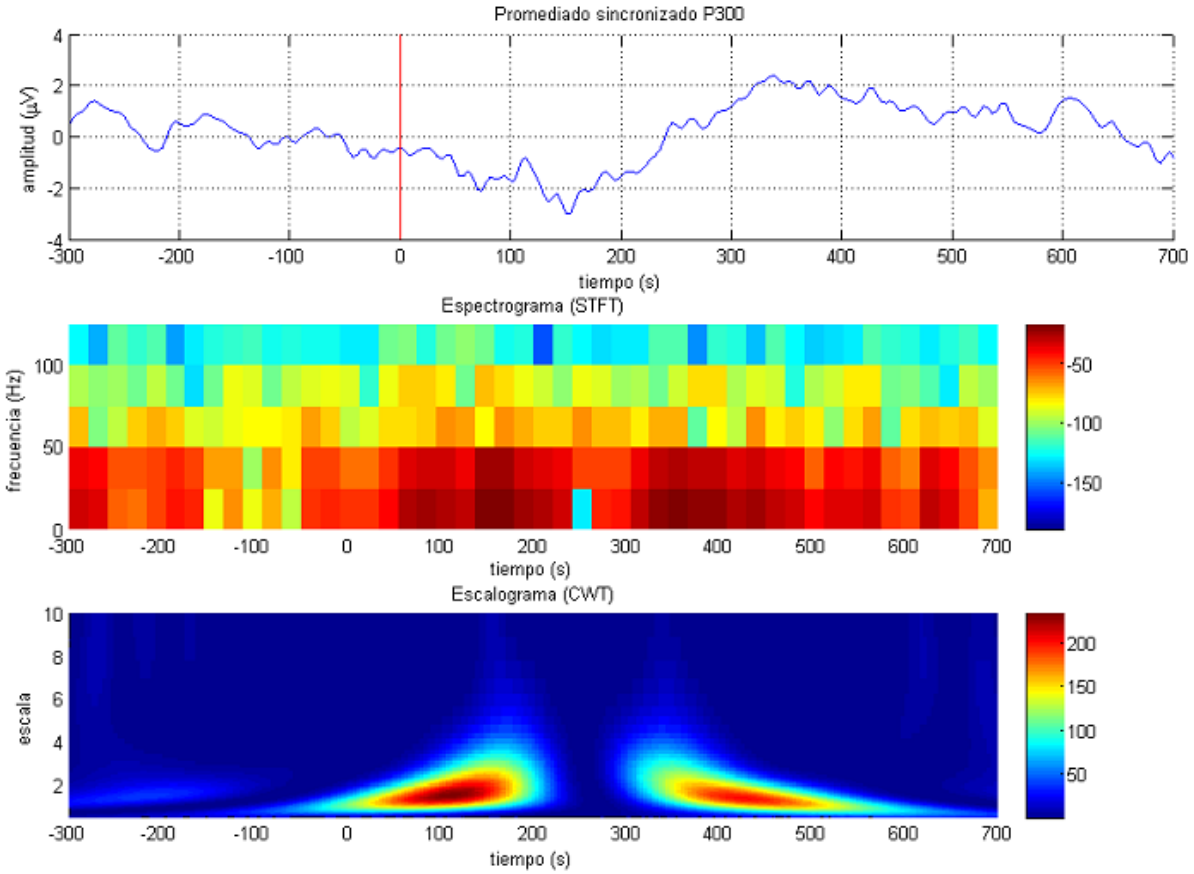

Figura 3-16. En la figura superior, representación del promediado sincronizado de 90 épocas de P300 para un *odd-ball* auditivo. En el centro, representación del espectrograma del mismo obtenido a través de la STFT. En la figura inferior, representación del escalograma del mismo obtenido a través de la CWT [3].

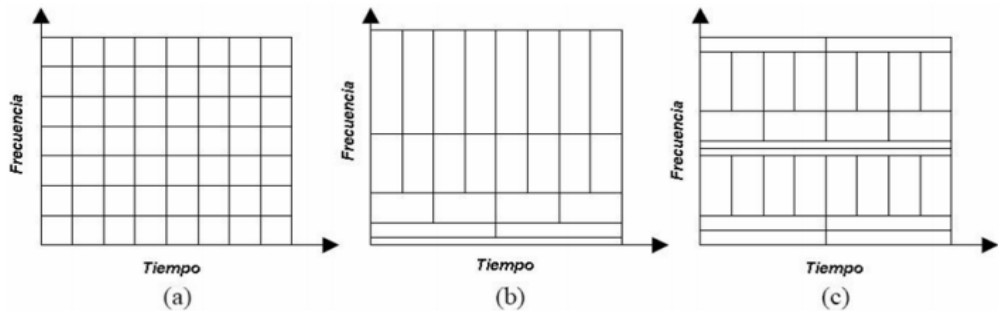

Figura 3-17. Plano tiempo-frecuencia para STFT en (a), con resoluciones fijas; DWT en (b), con resoluciones variables adaptadas a las señales biomédicas; y WPT en (c), con resoluciones variables según la intención del usuario [3].

3.3. TRADUCCIÓN DE LAS CARACTERÍSTICAS DE LOS POTENCIALES P300

Una vez extraídas las características de la señal EEG para los potenciales P300 hay que diseñar un algoritmo que nos permita clasificar si se ha producido un potencial o no. Estos algoritmos se denominan clasificadores.

Para un correcto funcionamiento de los clasificadores con cada usuario, este debe adaptarse a sus características particulares, debido a la gran variabilidad que existe entre diferentes personas, como ya hemos visto con anterioridad.

El problema de clasificación de los P300 es un problema de clasificación binario (se ha dado un potencial o no), en el que la decisión viene determinada por una función discriminante cuyo hiperplano de decisión se define como:

$$\boldsymbol{w} \cdot f(\boldsymbol{x}) + b = 0 \tag{3.15}$$

Donde $\boldsymbol{x}$ es el vector de características, $f(\cdot)$ es la función de transformación, $\boldsymbol{w}$ es un vector de pesos y b el sesgo. Para los clasificadores no lineales, la función $f(\cdot)$ puede representar una transformación que asigna las características a un espacio de dimensión mayor con el propósito de crear un conjunto de puntos que puedan separarse linealmente. Para los métodos lineales, la función $f(\cdot)$ simplemente es $f(\boldsymbol{x}) = \boldsymbol{x}$ [39].

La aplicación de este hiperplano de decisión en las características extraídas dará como resultado una clasificación binaria para cada época. Esta clasificación binaria asignará los valores 1 para estímulo atendido y -1 para estímulo no atendido, para luego sumar estas asignaciones al final del proceso de clasificación. Este diseño permite distinguir la respuesta con mayor distancia positiva de todas las filas y columnas de la matriz *oddball* empleada para el control de la aplicación BCI que se ha desarrollado, permitiendo distinguir la celda seleccionada.

Los objetivos de un clasificador son, por tanto, calcular la función $f(\cdot)$ y el vector $\boldsymbol{w}$ en el caso de los clasificadores no lineales, o solamente el de calcular $\boldsymbol{w}$ en los clasificadores lineales.

En este punto se detallan los clasificadores lineales más utilizados para la traducción de las características de los P300, como son el discriminante lineal de Fisher (FLD, *Fisher's Linear Discriminant*), y el análisis discriminante lineal paso a paso (SWLDA, *StepWise Linear Discriminant Analysis*).

3.3.1. Discriminante lineal de Fisher

El clasificador FLD es un método lineal de traducción de características, que sirve de referencia para determinar el hiperplano óptimo de separación entre dos clases de clasificación posibles, por su simplicidad y robustez, y así poder comparar otros algoritmos más sofisticados.

Además, resulta óptimo en el caso de que la distribución de probabilidad de las dos clases de clasificación (hay o no P300) sea Gaussiana y con matrices de covarianza iguales [39].

El vector de pesos viene dado por la siguiente expresión:

$$\widehat{\boldsymbol{w}} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1} \cdot \boldsymbol{X}^{\boldsymbol{T}} \cdot \boldsymbol{Y} \tag{3.16}$$

Donde $\hat{\boldsymbol{w}}$ es la estimación de los pesos del clasificador, $\boldsymbol{X}$ la matriz formada por los vectores de características e $\boldsymbol{Y}$ la matriz con los valores asignados de la clasificación binaria.

3.3.2. Análisis discriminante lineal paso a paso

Los algoritmos de análisis discriminante lineal (LDA, *Linear Discriminant Analysis*) son una extensión de FLD con reducción del espacio de características utilizadas para la clasificación

El conjunto de métodos LDA trata de encontrar una combinación lineal de características capaz de separar dos o más clases de eventos, mediante el uso de estadísticas de segundo orden para obtener el vector de pesos.

En consecuencia, la calidad del clasificador resultante depende de la covarianza estimada y, en la práctica, por cuestiones de tiempo y coste computacional, es difícil o imposible obtener la cantidad de datos necesarios para establecer un buen clasificador LDA.

En este contexto surge el algoritmo SWLDA, intentando estimar de una manera aceptable con menos datos, la covarianza de las características para subsanar esas deficiencias del LDA simple. Para conseguirlo, combina dos técnicas básicas de selección de características: el análisis paso-a-paso hacia delante y el análisis paso-a-paso hacia atrás [39].

El análisis paso-a-paso hacia delante (*forward stepwise regression*) parte de un conjunto vacío de características que se tendrán en cuenta en el clasificador y evalúa la significación de cada una de ellas, añadiéndola al conjunto si supera un cierto umbral o criterio de entrada y descartándola en caso contrario.

El análisis paso-a-paso hacia atrás (*backward stepwise regression*) realiza la función contraria: parte de un conjunto con todas las características posibles y evalúa cada una de ellas, preguntándose si deberían estar incluidas en el conjunto o no. Para ello, establece un criterio de salida y recorre el conjunto de características comprobando si éstas cumplen el criterio o no. Si una característica cumple el criterio, se descarta, reduciendo así el tamaño del conjunto.

El algoritmo SWLDA combina estos métodos de la siguiente manera [39]:

1) Se parte de un conjunto vacío de características válidas y realiza un análisis paso-a-paso hacia delante para añadir las que cumplan con el criterio de entrada. Este criterio normalmente es, que el *pvalor* para esta característica sea mayor que 0.1.
2) Cada vez que se añade una característica al conjunto de datos se hallan de nuevo los *pvalores.*
3) Se aplica el análisis paso-a-paso hacia atrás y se establece si alguna de las características incluidas en el conjunto cumple un determinado criterio de salida en función de los *pvalores* calculados. Un criterio de salida que se elige con frecuencia es que el *pvalor* sea mayor que 0.15.
4) Si la respuesta es afirmativa, se descartan las características que lo cumplan y se vuelve a realizar un análisis paso-a-paso hacia delante.
5) Este procedimiento se repite constantemente hasta que el conjunto de datos alcance el número máximo de características o hasta que no existan más características que cumplan los criterios descritos.

La efectividad del algoritmo SWLDA ha sido demostrada en numerosos estudios para la clasificación de los potenciales P300 [39]. En la figura 3-18 podemos ver una comparativa entre distintos clasificadores, donde, en general, el SWLDA es el más eficiente, incluso más que otros

más complejos y difíciles de implementar como las máquinas de vectores de soporte LVM y GSVM, dos clasificadores no lineales. Estos son los motivos de que sea el clasificador elegido para la traducción de características de la aplicación BCI desarrollada.

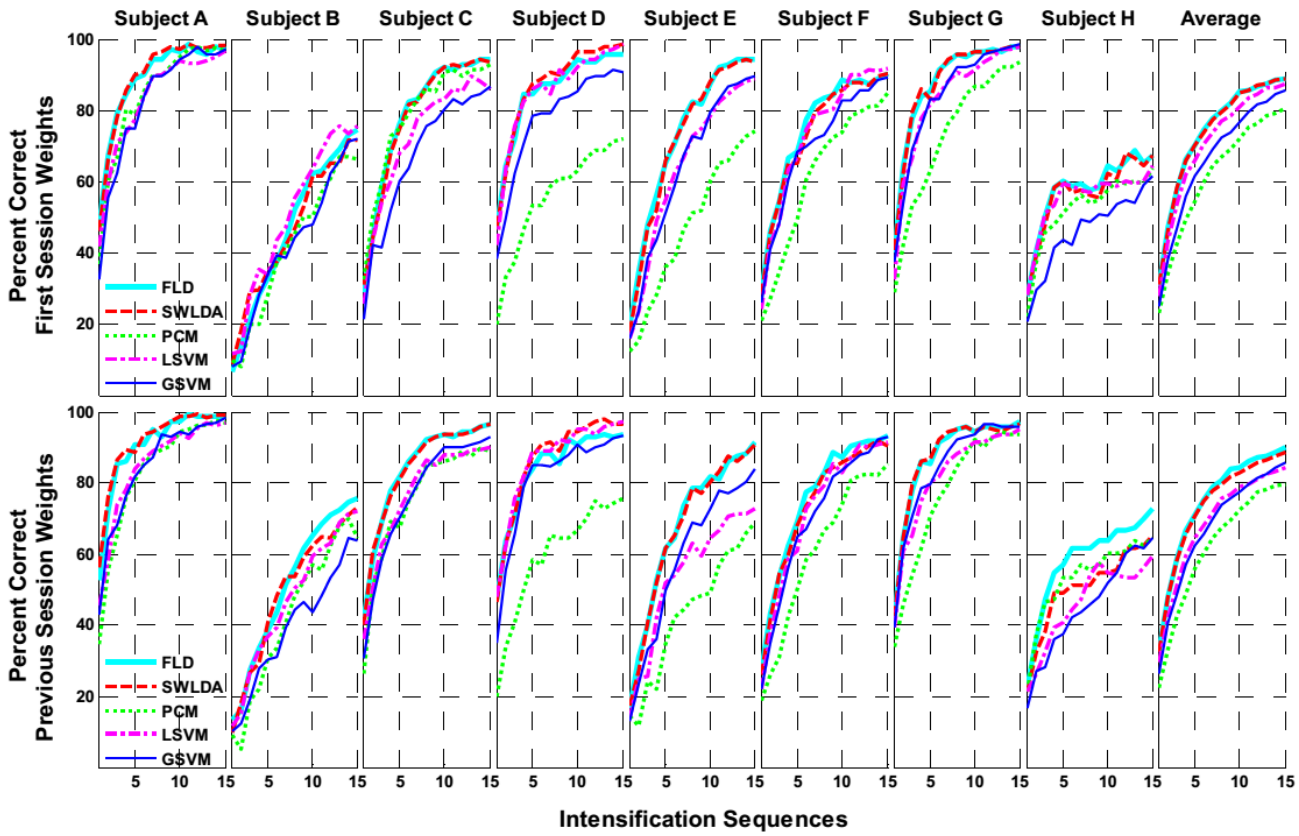

Figura 3-18. Gráficas comparativas del comportamiento entre distintos algoritmos de clasificación con 9 sujetos en dos sesiones diferentes [39]

CAPÍTULO 4

DISEÑO Y DESARROLLO DE LA APLICACIÓN

4.1. OBJETIVO DE LA APLICACIÓN

Las tecnologías móviles están en auge y hoy en día la utilización de los smartphones es algo habitual, y muchas veces es necesaria para la realización de una vida normal. El objetivo de la aplicación desarrollada es acercar a las personas con grave discapacidad a estas tecnologías, que ahora mismo necesitan de la interacción con músculos periféricos para su control. Este objetivo se conseguirá a través de una aplicación BCI, que deberá cubrir de manera suficiente las necesidades de una persona media en cuanto al uso de smartphones y teléfonos móviles y para ello, primero debemos averiguar cuáles son tales necesidades y los dispositivos en los que se va poder utilizar.

Así pues, la primera elección que se debe abordar es el sistema operativo. Basándonos en los últimos datos recogidos por IDC, en el segundo cuatrimestre del año 2015 la cuota de teléfonos con Android es del 82.8% [40], y por ello será el sistema operativo al que se enfocará el sistema BCI.

Basándose en diversos estudios, se realizará además una selección de las aplicaciones móviles más utilizadas, basándose en el top de las más descargadas de Google Play [41] y las aplicaciones nativas del teléfono. El objetivo del sistema, por tanto, es permitir el manejo de tales funcionalidades en un dispositivo Android mediante BCI basado en potenciales P300.

Las funcionalidades implementadas son las siguientes [41][42]:

- **Mensajes de texto:** La aplicación seleccionada para este contexto es la aplicación nativa de mensajes SMS que incorporan los teléfonos móviles. Además, se incluirá alguna funcionalidad con *Whatsapp Messenger*, ya que es número uno del top de aplicaciones más descargadas de Google Play, estando limitada esta funcionalidad por el carácter privado de la aplicación. El 65% de las interacciones con un Smartphone son para mandar, entre otras cosas, mensajes de texto (porcentaje de Me time + Socializing) [41]. Además, prácticamente el 100% de las personas que utilizan un Smartphone utilizan este tipo de aplicaciones [42].
- **Llamadas telefónicas:** Se hará uso de la aplicación nativa de Android para esta operación. El 19% de las interacciones con el teléfono tienen que ver con las llamadas telefónicas, entre otras cosas, por lo que son una parte crucial del sistema a implementar.
- **Fotografías:** El uso de la cámara estará integrado en el sistema. También se incluye una galería que permitirá visualizar las fotografías realizadas con la aplicación.
- **Calendario y Alarmas:** Dos de las funcionalidades más usadas en la vida cotidiana son las alarmas y el manejo de la aplicación calendario nativa de Android para el recordatorio de citas o fechas. Ambas funcionalidades estarán integradas en el sistema.
- **Manejo de contactos:** El manejo de la libreta de contactos es crucial para el desempeño de la aplicación. Poder añadir un contacto, eliminarlo, llamarlo y mandarle un mensaje se antoja primordial a la hora de manejar un teléfono móvil.

Con estas elecciones en las distintas categorías se cubre con suficiente profundidad el 65% [42] (*me time y socializing)* de las interacciones de un usuario medio con un Smartphone.

4.2. ARQUITECTURA DE LA APLICACIÓN

La arquitectura de la aplicación tendrá 3 partes fundamentales, como se puede ver en la figura 4-1: La adquisición de la señal, el procesado de señal y P3Speller corriendo en un ordenador y AppRunner ejecutándose en un dispositivo Android.

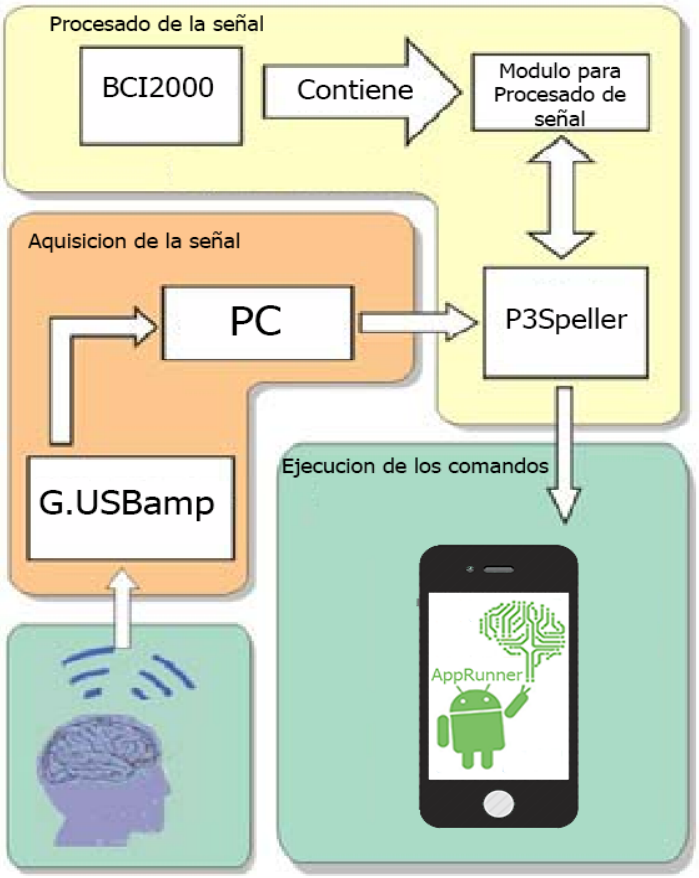

Figura 4-1. Arquitectura del sistema dividido en tres etapas: Adquisición de señal, procesado en el ordenador con BCI2000 y envío de los comandos a la aplicación AppRunner

Adquisición de la señal

La señal es adquirida mediante el dispositivo del laboratorio g.USBamp disponible en el laboratorio del Grupo de Investigación Biomédica de la universidad de Valladolid (GIB). Este dispositivo realiza la conversión analógico-digital de los datos obtenidos mediante EEG, previa amplificación y filtrado de la señal. Luego los almacena en memoria (buffer) a la espera de que la aplicación los requiera mediante el puerto USB, medio por el que se el ordenador con el sistema g.USBamp. Los drivers disponibles son para Windows 7 aunque también son compatibles con Windows 10, y este último este será el sistema operativo elegido para la maquina en que se realizará el procesado de señal [43].

Procesado de la señal

Se realizará en el ordenador debido a la inexistencia de drivers del dispositivo g.USBamp para plataformas móviles (Android, Windows 10 Mobile). El programa elegido para procesar la señal adquirida es BCI2000. Es un programa para aplicaciones BCI de propósito general que permite el procesado de la señal EEG y el diseño de matrices de comandos seleccionables siguiendo el paradigma *oddball* mediante potenciales evocados P300.

El usuario seleccionará el comando que quiere ejecutar en las matrices diseñadas en BCI2000. Posteriormente, se procesará la señal para averiguar el comando seleccionado y se mandará mediante un cliente Bluetooth incorporado a BCI2000 hacia el terminal móvil.

Ejecución de los comandos en el terminal móvil

Para la comunicación entre móvil y ordenador se necesitarán dos programas corriendo en paralelo: BCI2000 que envía los comandos seleccionados, y la aplicación móvil AppRunner que los recibe y los ejecuta. Para la conexión la tecnología elegida es Bluetooth por la existencia de numerosos ejemplos de aplicaciones que usan este tipo de conexiones. Las ventajas del uso de esta tecnología son:

- Conexión inalámbrica entre ordenador y móvil, lo que implica mayor movilidad del usuario y del terminal.
- Numerosos ejemplos de aplicaciones en los que se conecta un ordenador con un móvil a través de esta tecnología.
- Abundante documentación de la API en la página web de desarrolladores de google.

Una vez recibido un comando por Bluetooth, la aplicación deberá ejecutarlo. La aplicación deberá interactuar con el Kernel de Android (basado en Linux y programado en lenguaje C) para manejar el driver del puerto Bluetooth [44]. Hay disponibles clases de alto nivel para esta tarea en la API de google, que nos permitirán simplificar la programación para la recepción y envío de datos por Bluetooth. Uno de los retos del sistema será ejecutar la aplicación que escucha en segundo plano mientras se realizan las tareas seleccionadas por el usuario.

4.3. FUNCIONAMIENTO GENERAL DE LA APLICACIÓN

Como se dijo en el apartado anterior, la aplicación consta de tres partes bien diferenciadas que se comunican entre sí (figura 4-1). En este apartado se explicará de manera breve cada una de las funcionalidades que ofrecen los bloques de cara al usuario y al manejo de la aplicación, en vez desde el punto de vista de la arquitectura.

Primero está el bloque de adquisición de señal, compuesto por el casco de electrodos, el amplificador g.USBamp y el cable USB que lo conecta al ordenador. En este bloque se recoge la señal EEG y se condiciona para su posterior procesado en el ordenador.

La segunda parte de la aplicación se localiza en el ordenador, donde tendremos la aplicación P3Speller funcionando. Esta será la encargada de mostrar al usuario las matrices de comandos disponibles en cada momento al usuario. Se han diseñado 2 matrices, una con las acciones disponibles para ejecutar y otra con letras y números, que hará las veces de teclado, para completar los campos que lo requieran para obtener una funcionalidad completa.

Cuando la aplicación detecte que el usuario ha selecciono un comando de la matriz, el cliente Bluetooth integrado en P3Speller enviará la orden al móvil, donde habrá un servidor Bluetooth esperándola.

La tercera parte de la aplicación se encuentra en el dispositivo móvil. Esta parte es la encargada de recibir y ejecutar los comandos que llegan desde el módulo de procesado de señal en el ordenador.

La interfaz gráfica de la actividad (cada una de las diferentes ventanas de una aplicación Android se denomina actividad) principal consta de un botón que permite activar o desactivar el servidor Bluetooth para recibir o no comandos y una caja de texto donde se va informando al usuario de las acciones que se van tomando en la aplicación. Así, el usuario podrá obtener siempre un *feedback* de las operaciones que se están ejecutando en todo momento. Ver figura 4-2.

Además de la actividad principal, tenemos otros dos tipos de actividades en la aplicación. Uno de esos tipos se refiere a las actividades que se abren cuando el usuario selecciona un comando y que realizan una determinada acción, como puede ser abrir la cámara o visualizar una foto en la galería. El otro topo consiste en un formulario que el usuario deberá completar necesariamente para realizar algún tipo de acción, como puede ser añadir un contacto (número de teléfono, nombre, y opcionalmente el email).

En los puntos posteriores se describirá de manera más exhaustiva el funcionamiento de cada bloque de la aplicación.

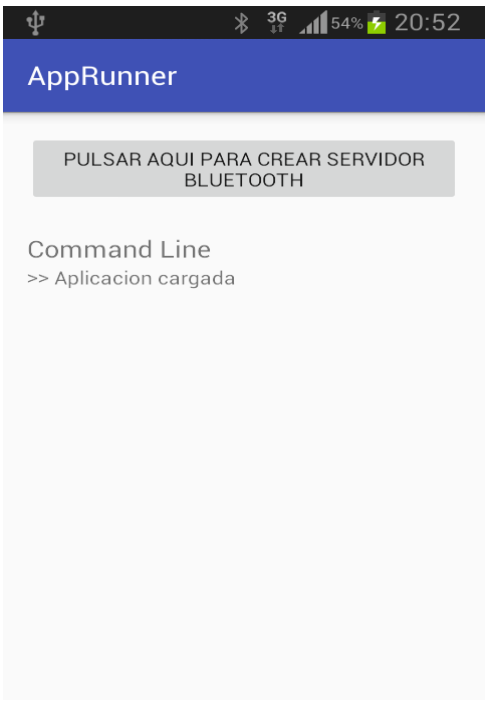

Figura 4-2. Actividad principal de ApppRunner. Se compone de un botón para activar el servidor Bluetooth y de la caja de texto “Command Line”, donde se visualizarán los mensajes de la aplicación.

4.4. ADQUISICIÓN DE LA SEÑAL EEG

El gorro que recoge la señal EEG del cuero cabelludo del usuario utiliza los siguientes electrodos: Fz, Cz, Pz, P3, P4, PO7, PO8 y Oz, cuya localización puede observarse en la Figura 4-3.

Tal y como se ha explicado anteriormente, es necesario colocar un gel conductor entre los electrodos y el cuero cabelludo para reducir la impedancia del cráneo y poder amplificar la señal correctamente. La amplificación la realiza el dispositivo *g.USBamp*, un amplificador desarrollado por la empresa *G·Tec*. En cuanto a la frecuencia de muestreo, esta aplicación la fija en 256Hz, adecuada para el análisis del EEG puesto que cumple con creces el teorema de Nyquist.

4.5. BCI2000

BCI2000 es un programa de código abierto, escrito en C++, y de propósito general que ha sido diseñado por Schalk Lab para la investigación de sistemas BCI. Implementa muchas funcionalidades, entre las que se encuentra P3Speller, que es la que se utilizará para el desarrollo de AppRunner.

P3Speller implementa el paradigma *oddball* ya comentado diseñado por Donchin y Farwell [18], permitiendo personalizar las matrices de comandos, cambiar multitud de parámetros e introducir porciones propias de código.

4.5.1. Funcionamiento de BCI2000 y P3Speller

BCI2000 será parte fundamental en la extracción y traducción de características, ya permite el procesamiento en tiempo real de señales EEG.

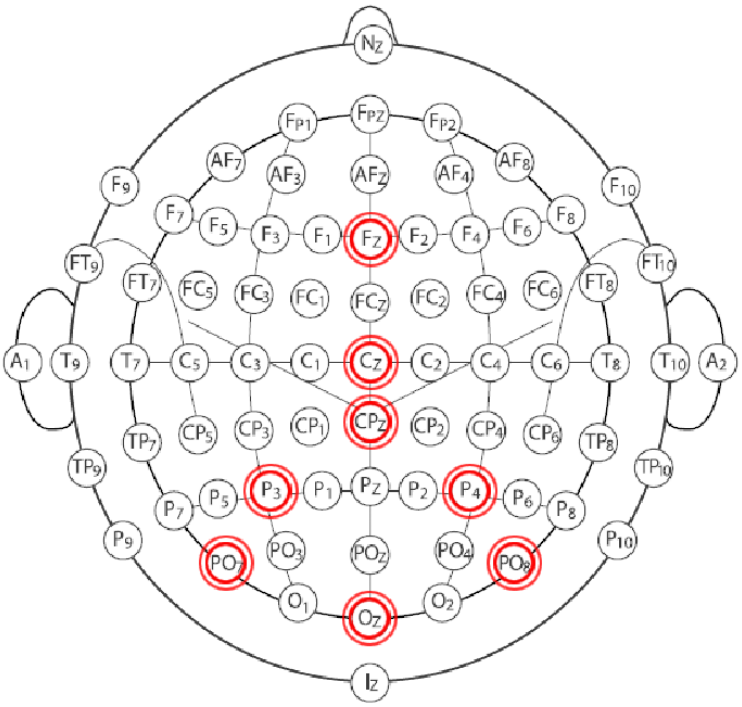

Figura 4-3. Disposición de los electrodos utilizados en la aplicación según el sistema internacional 10/20: Fz, Cz, Pz, P3, P4, PO7, PO8 y Oz [17].

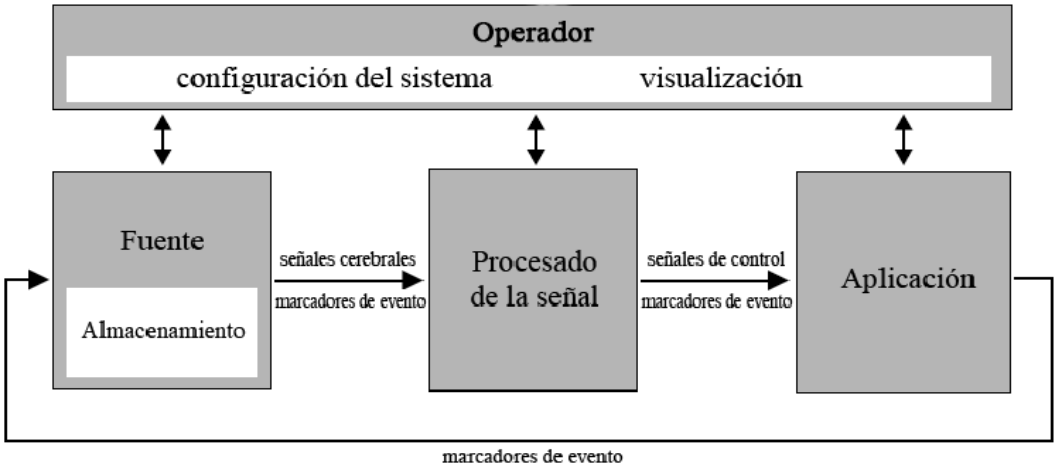

Figura 4-4. Estructura del sistema de propósito general BCI2000, compuesta por cuatro módulos independientes [17].

La arquitectura de BCI2000 esta descrita en la figura 4-4 [17]. Se compone de 4 módulos que se comunican entre ellos: la "Fuente", el "Procesado de la Señal", la "Aplicación" y el "Operador". Estos módulos son programas independientes y se comunican entre sí a través de un protocolo basado en TCP/IP, sin embargo, dado que esta aplicación no requiere un sistema distribuido, los módulos se comunican en la máquina local.

La señal EEG se adquiere en el módulo "Fuente" y se procesa de manera síncrona, en bloques que contienen un determinado número de muestras.

Cada vez que se adquiere un nuevo bloque de datos, el módulo "Fuente" lo envía al módulo de "Procesado de la Señal", a su vez subdividido en dos partes: extracción y traducción de características, que se encarga de producir señales de control que envía al módulo de "Aplicación".

Finalmente, el módulo de "Aplicación" envía la señal EEG analizada y marcada con eventos que facilitan la distinción de las diversas señales de control a utilizar al módulo "Fuente", con el fin de poder realizar un análisis *off-line* gracias al almacenamiento en disco de la señal por parte de este último módulo.

El módulo "Operador" se encuentra apartado de este análisis EEG, sin embargo, provee al usuario una interfaz gráfica que permite configurar los distintos parámetros del sistema y visualizar en tiempo-real el análisis de la señal. Por esta razón, el módulo "Operador" se comunica con el resto de módulos y permite al usuario iniciar, parar, suspender o continuar las operaciones que realiza el sistema [17].

El sistema BCI2000 tiene implementada ya una solución del paradigma *odd-ball* visual con el fin de generar los potenciales evocados P300 a través de una matriz cuyas filas y columnas se iluminan aleatoriamente, llamada P3Speller. Esta solución sugiere utilizar un filtro CAR y un promediado sincronizado como métodos de extracción de características y un algoritmo SWLDA como método de traducción de las mismas. En el presente trabajo se ha mantenido la utilización de estos métodos, pero, sin embargo, se ha adaptado la aplicación *P3Speller.*

Para producir los comandos específicos de la aplicación y enviarlos por Bluetooth se han modificado varios ficheros adaptando el código a las necesidades de este sistema. Las principales

modificaciones consisten en un algoritmo de traducción para asignar a cada celda de la matriz *oddball* seleccionada un comando a ejecutar en el terminal móvil y un cliente Bluetooth que permite enviar cadenas de texto con la información necesaria para la ejecución del comando en la aplicación corriendo en el móvil. Para ello se ha modificado principalmente el módulo de "Aplicación".

4.5.2. Configuración de los parámetros

Nada más iniciar la aplicación se inicia el módulo "Operador", donde el usuario puede configurar diversos parámetros como la ruta de los archivos *.dat* que generará, el número de canales a analizar, el nombre de los electrodos utilizados, el filtro espacial a utilizar, los pesos del clasificador del usuario, el tamaño y las celdas de las matrices que generan los P300, la duración y el número de las secuencias, la duración de los estímulos o el modo de funcionamiento o el umbral de cada usuario, entre otros. En la Figura 4-5 aparece la interfaz gráfica del "Operador".

Para cambiar la configuración se deberá pulsar en *config*, abriéndose una segunda ventana, que aparece en la figura 4-6.

La configuración que se ha usado del módulo "Operador" con sus parámetros más importantes se describe a continuación por pestañas. Estos parámetros determinarán el funcionamiento de P3Speller y el paradigma *oddball*.

Pestaña "Application"

En esta pestaña se configura el contenido de las matrices y el desempeño temporal del paradigma, pudiendo configurar los siguientes parámetros:

- **PreRunDuration: 4s**. Establece la pausa que inicia el intento.
- **PreSequenceDuration: 4s**. Establece la pausa que precede a la primera secuencia. Se considera como una secuencia la iluminación de todas las filas y de todas las columnas que componen la matriz y, por tanto, el número de secuencias de cada intento es equivalente al número de veces que se va a iluminar cada fila y cada columna antes de determinar la selección del usuario. Por ejemplo, para la matriz de navegación, de tamaño 5x3, una secuencia se corresponde con "5+3 = 8" iluminaciones.
- **StimulusDuration: 62.5ms**. Establece la duración del estímulo visual, es decir, cada iluminación.
- **ISIMinDuration, ISIMaxDuration: 125ms, 250ms**. Establecen la duración mínima y máxima del intervalo entre estímulos, respectivamente. Especifican el tiempo que pasa

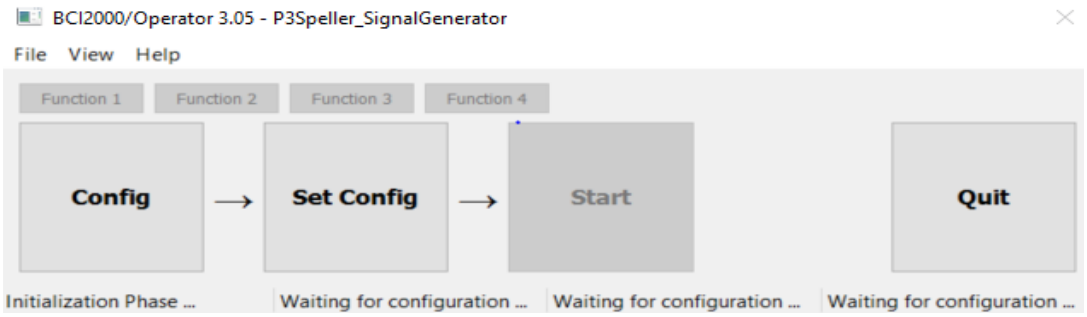

Figura 4-5. Interfaz gráfica del "Operador"

entre dos iluminaciones, siendo éste aleatorio, con una distribución uniforme para los valores intermedios.

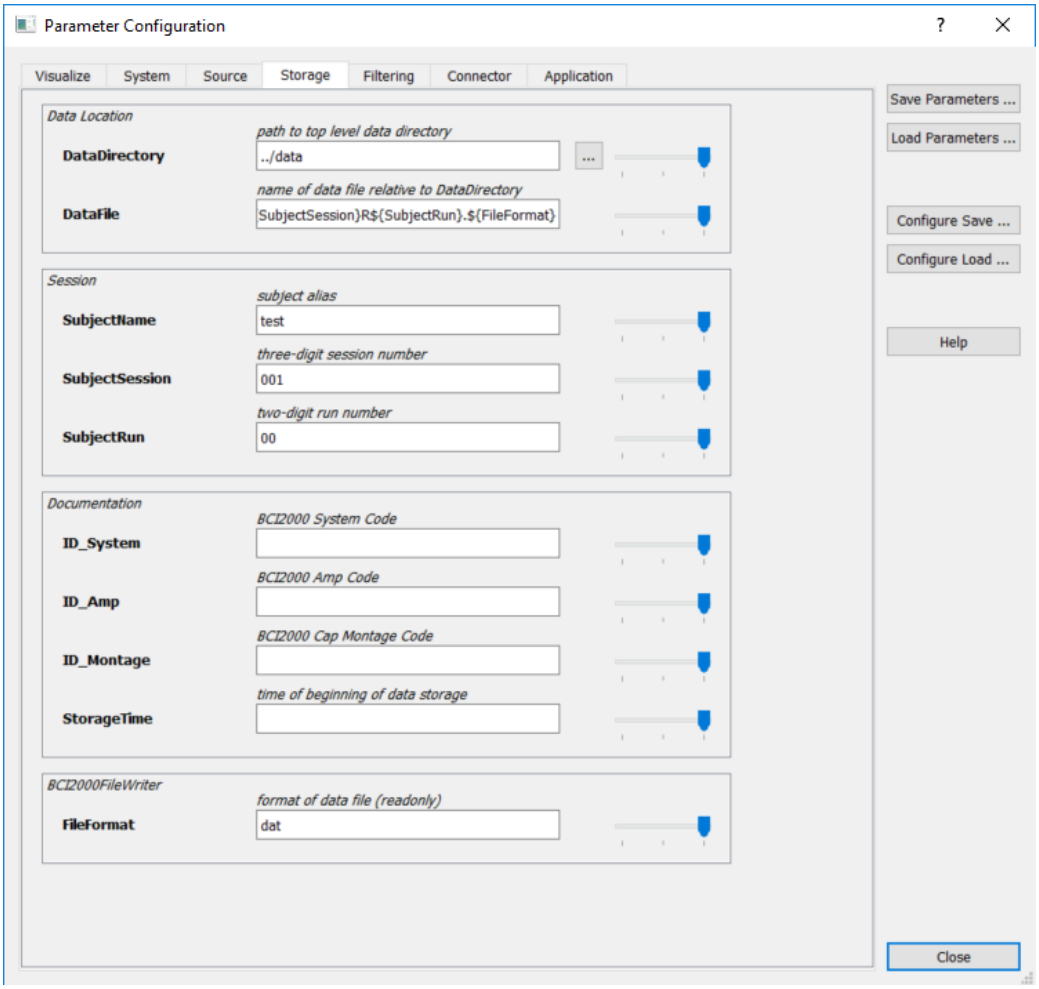

Figura 4-6. Interfaz gráfica de la ventana de configuración de "Operador". Como se puede observar tiene varias pestañas donde se configuran los parámetros correspondientes.

- **PostSequenceDuration: 4s**. Establece la pausa que sigue a la última secuencia del intento.
- **PostRunDuration: 2s**. Establece la pausa que finaliza el intento.
- **NumberOfSequences**. Número de secuencias que compondrán un solo intento. Este valor debe ser el mismo que el del parámetro *EpochsToAverage*, el cual determina el número de épocas que deben considerarse en el promediado sincronizado (la duración de las mismas la establece *EpochLength*). El valor de este parámetro depende de las características individuales del usuario, como veremos mas adelante

En la figura 4-7 podemos observar el desarrollo temporal de cada intento de selección en la matriz, que de ahora en adelante llamaremos *trial*.

En esta pestaña también se modifican los parámetros de las matrices *oddball* de la aplicación P3Speller. La modificación y los valores utilizados de los mismos se verán posteriormente.

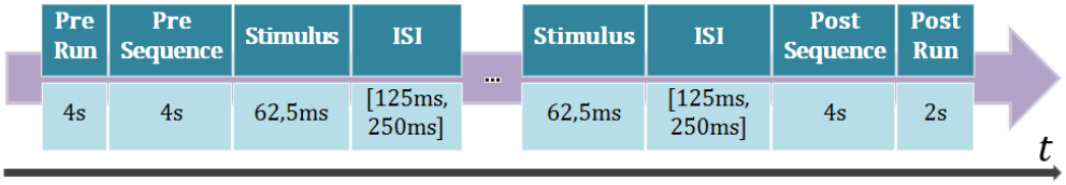

Figura 4-7. Línea de tiempo de cada intento de *P3Speller*.

Pestaña "Filtering"

En esta pestaña podemos configurar los parámetros más importantes respecto al filtrado los cuales son:

- **SpatialFilterType: CAR.** Aplica un filtrado espacial de referencia común, ya introducido en el capítulo 3.
- **EpochLength: 800ms**. Es el periodo de tiempo tras el estímulo que se considera en el procesado de señal para la extracción y traducción de características.
- **Classifier**. Este parámetro es de los más importantes. Consiste en una matriz de pesos que contiene un clasificador individualizado para cada usuario, que se deberá generar antes, para un correcto funcionamiento y precisión de la aplicación.

Pestaña "Source"

En la pestaña source podremos cambiar los parámetros correspondientes con la fuente de la señal y su adquisición. Los más importantes en nuestra aplicación son:

- **ChannelNames: Fz, Cz, Pz, P3, P4, PO7, PO8, Oz.** Nombres de los electrodos utilizados para la adquisición de señal
- **SourceChannels:** 8. Número de canales que utiliza la aplicación.
- **Sampling Rate y BlockSize:** 256 Hz, 8. Frecuencia de muestreo y tamaño de cada bloque de datos.

4.5.3. Matrices *odd-ball*

En este apartado se detallan específicamente los parámetros y el diseño elegidos para las matrices *oddball.* Estos se pueden cambiar la pestaña "Application" de configuración del módulo "Operator", de la que hemos comentado una parte en el punto anterior.

Dada la gran cantidad de tareas y aplicaciones disponibles para controlar en el dispositivo Android y de la necesidad de un teclado para comunicar ciertos datos a la aplicación, se ha decidido diseñar dos matrices separadas.

La primera de ellas se muestra en la figura 4-8. Es una matriz 4x4 que incorpora todas las actividades disponibles para manejar en el terminal móvil. La selección de uno de estos comandos conllevará la ejecución de la tarea consecuente en el terminal después de la comunicación Bluetooth. Los parámetros de esta matriz están pensados para que ocupe toda la pantalla de un ordenador en resolución HD de 1920 x 1080 puntos, distribuyendo los iconos de manera uniforme como podemos ver en la figura 4-8.

Figura 4-8. Matriz de comandos diseñada para la aplicación AppRunner

La segunda matriz es un teclado clásico con las funcionalidades necesarias para la navegación por los formularios de la aplicación móvil que se compone de 6 filas y 9 columnas Cuando se seleccione una letra, número, símbolo o comando se enviará a la aplicación, que decidirá el proceder adecuado. Podemos ver esta matriz en la figura 4-9.

La acción de cada una de las celdas seleccionables se detalla en la siguiente sección, junto con la explicación de la aplicación móvil.

Figura 4-9. Matriz de teclado diseñada para la aplicación AppRunner

4.6. APPRUNNER

En este punto se explicarán en detalle las funcionalidades que implementa la aplicación móvil AppRunner, relacionándolas con cada acción seleccionable en la matriz de comandos de P3Speller (figura 4-8). Además, se explicará brevemente la estructura del código y las herramientas utilizadas en su desarrollo.

4.6.1. Funcionalidades disponibles

Comando "Llamar"

Con la selección de *llamar* la aplicación abre el formulario mostrado en la figura 4-10a que el usuario deberá rellenar con la matriz de teclado.

El campo del formulario se puede rellenar con un nombre de contacto guardado en el teléfono, del cual la aplicación se encargará de averiguar el teléfono, o directamente un número telefónico válido. Para rectificar la información introducida se pueden utilizar las funciones CLR, que limpia el campo por completo, y DEL, que borra el ultimo dígito introducido.

Si por cualquier motivo el usuario no quiere llamar y quiere volver a la matriz de comandos, deberá seleccionar el comando MTX para volver a la actividad principal de la aplicación y poder seleccionar otra acción. Cuando se haya rellenado satisfactoriamente, el usuario deberá seleccionar ENTER en la matriz de teclado para iniciar la llamada.

Comando "SMS"

Con la selección de *SMS* la aplicación abre el formulario de la figura 4-10b. Se deberán rellenar los dos campos para el envío del mensaje, o este será erróneo. Para desplazarse por los campos de un formulario con varias entradas se utilizan los comandos ENTER y TAB_BACK de la matriz de teclado. Cuando se selecciona ENTER se pasa al campo siguiente, cuando se selecciona TAB_BACK se pasa al campo anterior, permitiendo una navegación fluida y la corrección de errores.

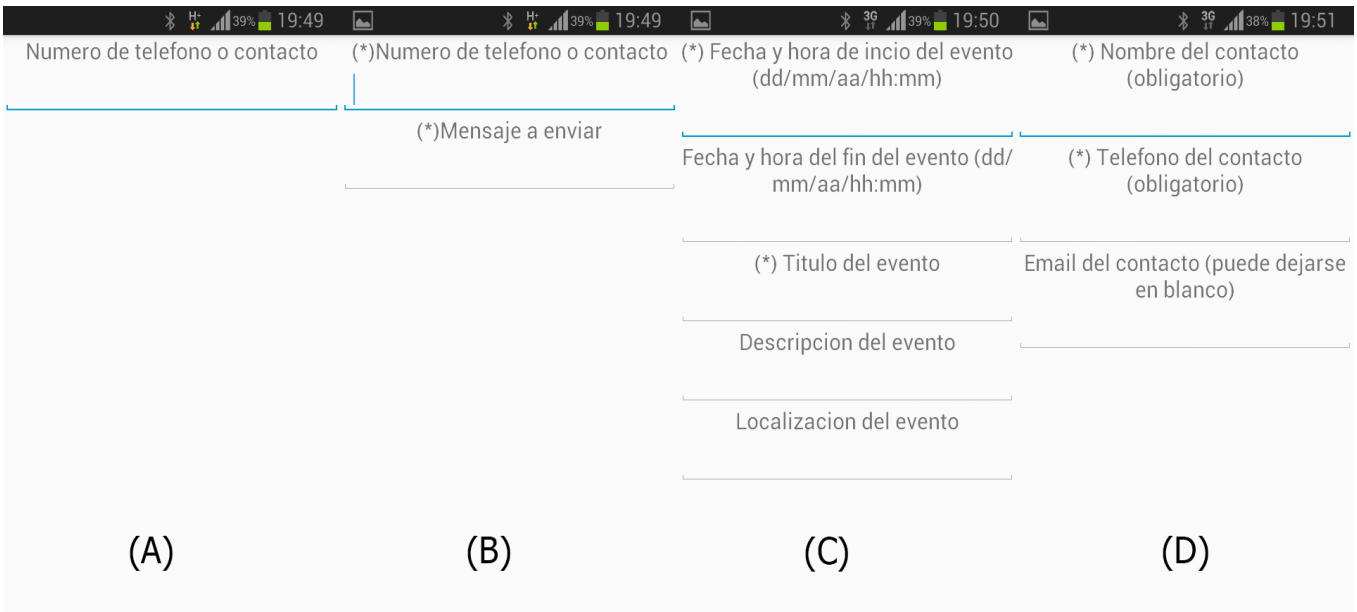

Figura 4-10. Ejemplos de formularios de la aplicación AppRunner para: (A) Llamar, (B) SMS, (C) Añadir evento y (D) Añadir contacto

Como en el caso de una llamada, el campo del contacto se puede rellenar directamente con un teléfono o con el nombre de un contacto almacenado en el teléfono.

Comandos "Añadir evento" y "Alarma"

Estas acciones permiten el manejo de las aplicaciones nativas de calendario y alarmas.

Para añadir un evento los se ha de rellenar el formulario que aparece en la figura 4-10c, estando marcados con el símbolo (*) los que son obligatorios. Una vez rellenado en el formulario, pulsaremos ENTER estando seleccionado el ultimo campo. Se abrirá la aplicación nativa de calendario con todos los datos introducidos ya rellenados. Para guardar el evento y volver a la actividad principal de la aplicación se seleccionará MTX.

Para añadir una alarma el proceso es similar, variando los datos que hay que introducir en el formulario.

Comando "Whatsapp"

Con esta acción podemos rellenar el campo de texto de un mensaje de Whatsapp, pero será necesario seleccionar el contacto al que se lo enviaremos a mano. Esto es debido al carácter privativo de la aplicación, que no está diseñada para interactuar con otras aplicaciones móviles ni sistemas BCI.

Comandos "Ver Contacto", "Añadir Contacto", "Eliminar Contacto"

Estos comandos nos permitirán manejar la libreta de contactos nativa del dispositivo móvil, rellenando los formularios correspondientes (ej. Eliminar contacto, figura 4-10d) a cada acción que se abrirán al seleccionarlas en la matriz de P3Speller. La navegación por los formularios es completamente análoga a los casos ya explicados.

Comandos "Galería", "Sig_Imagen", "Ant_imagen" y "Rotar imagen"

Estas acciones no abrirán un formulario que se deberá rellenar. La acción principal es la de "Galería" que abrirá la galería de las imágenes realizadas con la aplicación a pantalla completa. Una vez abierta, podremos desplazarnos por las imágenes cargadas con los comandos "Sig_Imagen" y "Ant_imagen" o rotarlas 90º hacia la derecha seleccionando "Rotar imagen".

Los comandos "Sig_Imagen", "Ant_imagen" y "Rotar imagen" solo tendrán un efecto si se ha abierto antes la galería.

Para salir de la galería se deberá seleccionar el comando "Volver".

Comandos "Abrir Cámara" y "Sacar foto"

Con estas dos opciones controlaremos una sencilla aplicación de cámara integrada en la aplicación. Para realizar una fotografía, primeramente, deberemos seleccionar "Abrir cámara", lo que abrirá una visualización de lo que se está captando en tiempo real por la cámara del dispositivo. Posteriormente, cuando se considere oportuno, se seleccionará "Sacar foto" para guardar una imagen en la memoria del teléfono. Para volver a la actividad principal de la aplicación seleccionaremos "Volver".

Comandos "Volver" y "Pausa"

El comando "Volver" se utilizará para cerrar una actividad secundaria en la aplicación móvil y volver a la actividad principal.

El botón "Pausa" permite pausar la aplicación, quedando en *stand by* hasta que se vuelva a seleccionar el comando "Pausa". Esto permite la visualización de las fotografías o simplemente descansar del manejo de la aplicación.

4.6.2. Guía de programación de la aplicación

En este punto se explica de manera breve la estructura interna de la aplicación desde el punto de vista de la programación, los ficheros con código generados y las herramientas utilizadas para el desarrollo. El código generado se adjunta en el Anexo B.

a) Modificaciones en P3Speller

En primer lugar, se comentarán las modificaciones realizadas en BCI2000 y concretamente en P3Speller, para la realización de la aplicación. El lenguaje en el que está programado BCI2000 es C++, y por consiguiente la parte de AppRunner introducida en él también lo está. En concreto, se han modificado los archivos "KeystrokeFilter.cpp" y "KeystrokeFilter.h" mediante la herramienta de programación *Microsoft Visual Studio 2010.*

"KeystrokeFilter.cpp" y "KeystrokeFilter.h" son los archivos que contienen el código necesario para traducir los cambios de estado en BCI2000 (por ejemplo, el cambio de estado que se produce cuando se selecciona un comando de la matriz) en pulsaciones de teclado [17]. En el archivo "KeystrokeFilter.cpp" se hicieron los cambios necesarios en la función "`KeystrokeFilter::Process`" para que en vez de traducirse la selección de la matriz en una pulsación de teclado se tradujera en una serie de comandos más complejos que pueda interpretar la parte de la aplicación alojada en el dispositivo móvil.

Además, se introduce un cliente Bluetooth que consta de 4 funciones. La que realiza la funcionalidad de enviar el comando traducido es "`send_through_bluetooth`", que es una función que recibe una cadena de caracteres char* y la envía al dispositivo especificado.

b) Aplicación móvil AppRunner

Para la programación de esta parte se ha utilizado la herramienta para desarrolladores proporcionada por Google, *Android Studio 2.1,* en la versión para el sistema operativo Windows 10. Es una potente herramienta con un gran soporte y muchas funcionalidades, entre las que se encuentra la simulación de un dispositivo Android en el ordenador y la depuración de la aplicación en tiempo de ejecución en el dispositivo mediante una conexión USB. Además, posee una gran cantidad de información que permite corregir errores de lógica y sintácticos durante la generación del código.

La aplicación se compone de 12 ficheros que contienen toda la lógica y la interfaz gráfica de la aplicación. Como ya se ha comentado, una aplicación Android se compone de diferentes actividades que realizan una determinada acción. Cada actividad lleva asociada un archivo .xml que especifica la interfaz gráfica, y uno o varios archivos .java con la lógica de la actividad. En los siguientes subapartados se detalla la estructura interna de la aplicación.

Actividad principal

El código de esta actividad está contenido en "`MainActivity.java`". mientras que la interfaz gráfica está diseñada en "`activity_main.xml`".

Para comprender mejor esta actividad, conviene explicar lo que es el hilo de interfaz de usuario (UI Thread, *User Interface Thread*) que de aquí en adelante llamaremos hilo UI.

Este hilo es el proceso padre de toda la aplicación y en el corre la actividad principal. Cualquier otro proceso que se inicie en paralelo al proceso UI heredará sus atributos, pasando a ser un proceso hijo de este. Esto es importante porque la aplicación AppRunner se compone de, como mínimo, 3 procesos simultáneos, que son el UI y dos procesos hijos.

A continuación, se explica a grandes rasgos el flujo principal del programa y el papel de los 3 procesos involucrados. El programa comienza con la creación del servidor Bluetooth, que corre en los dos procesos hijos. Estos procesos se crean en las clases "`AcceptThread`" y "`ConnectedThread`" contenidas en "`MainActivity.java`". La necesidad de crear dos hilos diferentes del UI viene dada por la naturaleza bloqueante de los métodos usados para aceptar conexiones Bluetooth. El funcionamiento del servidor es el siguiente:

1. EL proceso UI instancia la clase "`AcceptThread`", que hereda de la clase "`Thread`", y pone en marcha un nuevo proceso hijo que ejecuta el código contenido en el método "`run()`" . Para manejar el dispositivo Bluetooth es necesario realizar una serie operaciones las cuales no se detallarán en este documento. Una vez realizadas, la clase "`AcceptThread`" está preparada para recibir conexiones Bluetooth mediante el método bloqueante "`accept()`" de la clase "`BluetoothServerSocket`". Esta operación crea un punto de acceso al programa mediante un socket Bluetooth que está identificado por una UUID (*Universally Unique Identifier*) y el proceso queda bloqueado en este punto hasta que un dispositivo se conecte a este acceso.
2. Cuando llega una petición de conexión al servidor debido a que el usuario ha seleccionado un comando en el ordenador, esta se acepta y se instancia la clase "`ConnectedThread`" que también hereda de "`Thread`" y que crea un nuevo proceso hijo.
3. En este punto, el proceso que corre en "`AcceptThread`" continúa su curso y queda inmediatamente disponible para aceptar otra conexión mientras que el nuevo hilo creado por "`ConnectedThread`" se encarga de recibir y procesar los paquetes de datos que llegan al punto de acceso. Una vez procesados y convertidos en cadenas de texto, los envía al hilo principal UI para traducirlos en comandos de la aplicación. Hay que añadir que la comunicación entre distintos procesos no es trivial y es necesario instanciar una clase que herede de una herramienta proporcionada por Google para tal efecto llamada "`Handler`", de la cual hay que sobrescribir varios métodos.

Una vez recibido el comando, el programa sigue su curso. En el hilo principal se llama al método "`StartActivity()`". Esta función recoge el comando recibido del servidor Bluetooth implementado y decide la siguiente acción a realizar. Es la función que contiene el algoritmo de traducción de los comandos en los pasos a seguir para realizar la acción requerida. Hay dos acciones disponibles:

1. Iniciar la actividad formulario para pedir más datos al usuario. La información de esta actividad se explica a continuación.
2. Abrir una de las otras actividades de la aplicación: La cámara o la galería

En la figura 4.11 podemos ver el flujo de la actividad principal durante la ejecución de la aplicación.

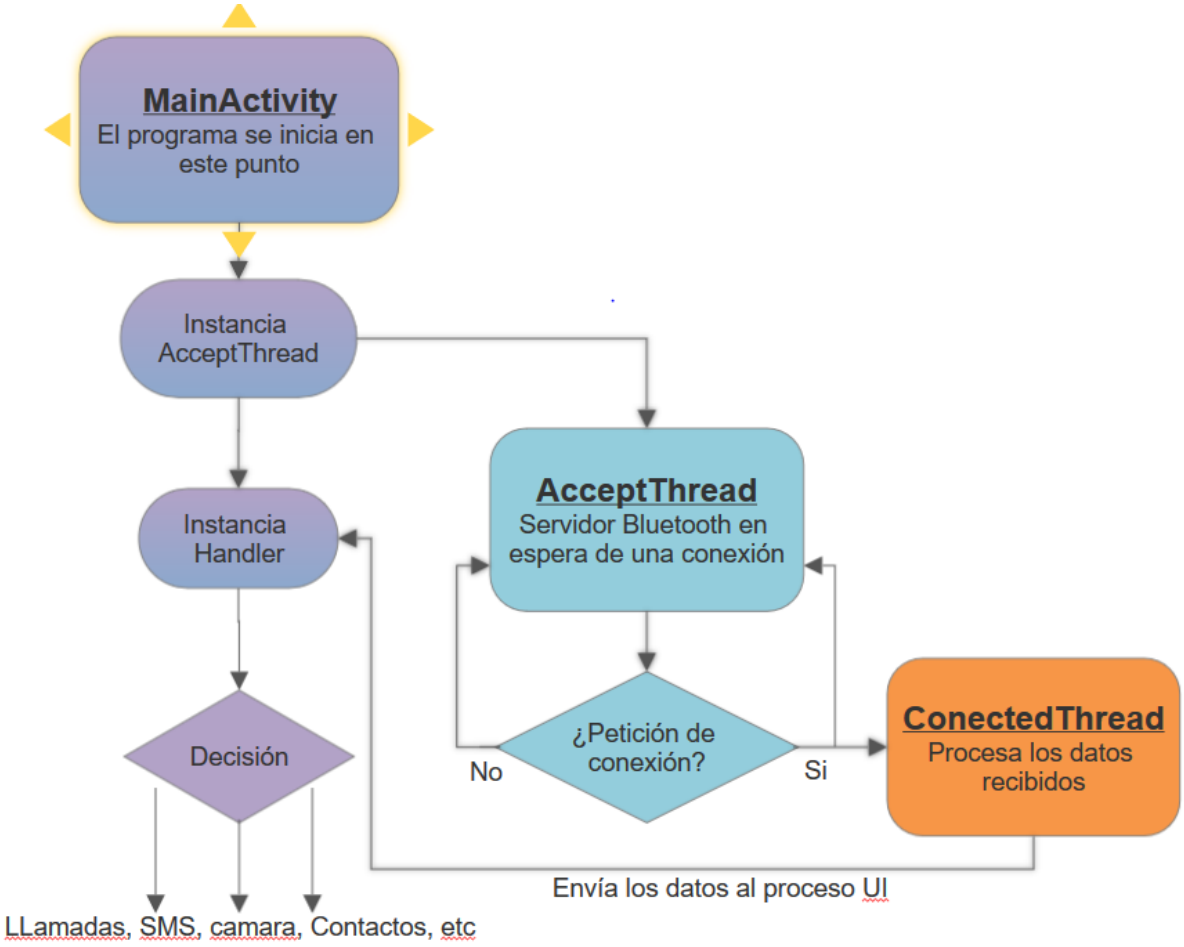

Figura 4-11. Flujo de la actividad principal de AppRunner

Actividad de cámara

Esta actividad lleva asociado dos archivos ("`CameraActivity.java`" y "`CameraPreview.java`") con el código que implementa una sencilla aplicación de cámara para controlar el Hardware óptico del dispositivo. Se divide en dos ficheros, ya que la lógica del programa necesita dos clases diferentes, una para captar imágenes continuamente y conseguir así una previsualización de lo que "ve" la cámara, y otra para captar la imagen en el instante que se toma la fotografía. No se utiliza la aplicación de cámara nativa porque no está preparada para interactuar con otras aplicaciones sin mediar ninguna pulsación real.

Actividad de galería

Esta actividad lleva asociado el archivo ("`ImgViewActivity.java`" con la lógica de la aplicación de galería.

Actividad de formulario

Si hay una actividad de este tipo corriendo en paralelo al hilo UI (que sigue recibiendo comandos como hemos visto hasta ahora), el proceso principal le envía todas las cadenas de texto que lleguen desde el servidor Bueltooth. Además, contiene el código necesario para generar dinámicamente la actividad formulario en función de la operación que quiera realizar el usuario

y retornar a la actividad principal los valores introducidos con la matriz de teclado. Este envío de información se realiza añadiendo datos de resultado al final de actividad. Cuando el proceso que ejecuta la actividad formulario termina, se llama en la actividad principal al método "`onActivityResult`", donde se reciben los resultados.

Resto de actividades

Como ya se ha explicado, para el resto de acciones, se interacciona con las aplicaciones nativas ya instaladas, que debe tener el dispositivo Android, desde la actividad principal. Estas aplicaciones son las correspondientes al calendario, alarmas, gestión de contactos, Whatsapp, llamadas y SMS. La clase "Intent" y sus parámetros permiten esta comunicación

La figura 4-11 contiene un esquema de la estructura interna de la aplicación resumiendo la información contenida en este apartado de una manera visual.

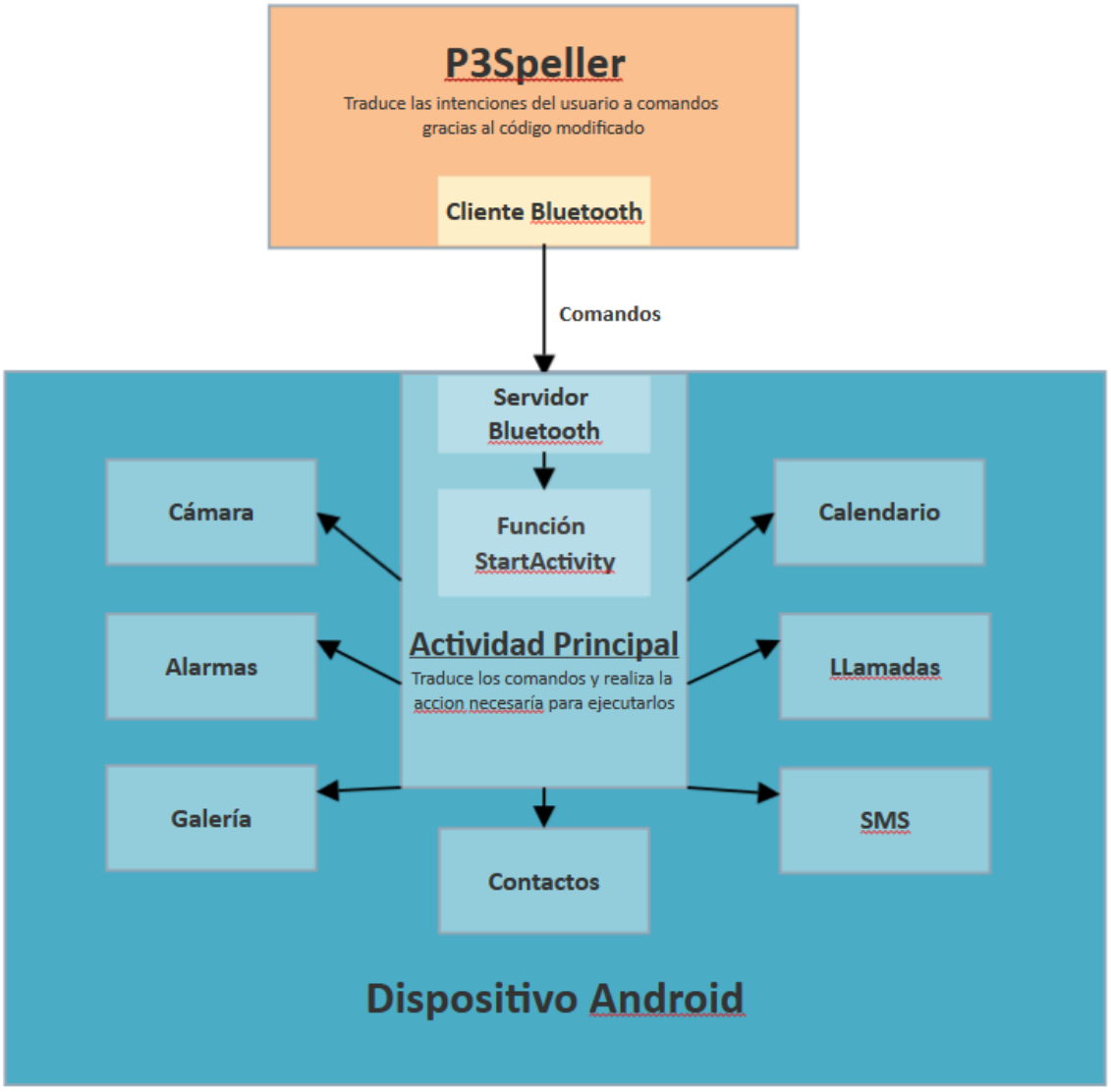

Figura 4-12. Estructura interna del sistema BCI desde el punto de vista de la lógica de programación

CAPÍTULO 5

EVALUACIÓN DE LA APLICACIÓN

5.1. SUJETOS Y TAREAS

La aplicación se ha evaluado en dos sesiones distintas con 5 cinco sujetos sanos, cuatro varones y una mujer, cuyas edades comprenden desde los 23 años hasta los 31 años. Cuatro de los sujetos tenían experiencia en sistemas BCI con señal de control P300, mientras que uno de ellos no había manejado nunca una aplicación de estas características.

La primera sesión fue orientada a calcular el clasificador para el sujeto y a realizar una primera demostración de la aplicación, que servirá de entrenamiento para la segunda sesión.

La segunda sesión se realizó una demostración más exhaustiva de las funcionalidades de la aplicación, con tareas más largas y de más dificultad.

Los electrodos utilizados aparecen en la Figura 4-2 y su localización tampoco ha sido modificada entre sesiones. Los parámetros de P3Speller se han dejado constantes según lo explicado en el anterior capítulo, variando únicamente el número de estímulos necesarios para una selección dependiendo de los resultados del clasificador del usuario.

A continuación, se describe el objetivo y las especificaciones de la aplicación para cada sesión, dispuestas en días distintos.

5.1.1. Primera sesión

Esta primera sesión se divide en dos partes bien diferenciadas. Una primera parte con una duración estimada de unos 26 minutos, orientada a calcular el clasificador con los pesos específicos del usuario, y una segunda parte orientada a introducir al usuario en el manejo de la aplicación AppRunner con una duración de otros 20 minutos. El tiempo total es mayor debido de la necesidad de preparar las tareas, colocación del equipo, descanso, etc.

El objetivo de la primera parte de esta sesión es, como se ha comentado, calibrar el clasificador para cada usuario utilizando el algoritmo SWLDA. Para ello se realizaron 4 rondas de 6 intentos cada una. Para realizar esta primera parte de la sesión se utilizó la matriz de teclado, con unas dimensiones de 6 x 9 debido a que es la de mayor tamaño.

La tarea de los sujetos consistía en mirar fijamente a la letra o número que tocara en cada momento sin perder la concentración. El número de secuencias para esta primera parte de la sesión se estableció en 15, lo que significa que cada fila y cada columna se iluminaban 15 veces, lo que hace un total de 30 iluminaciones de una celda concreta de la matriz.

Una vez realizada esta primera tarea, se calculaba la matriz de pesos del clasificador mediante la herramienta *P3Classifier,* incluida en la aplicación BCI2000. La entrada de esta aplicación son los archivos generados por el módulo “Operator” con los datos y la señal recibida de cada una de las rondas ejecutadas por los sujetos. La salida consiste en un archivo con extensión *prm,* que contiene la información del clasificador generado por *P3Classifier* que podemos cargar en la aplicación *P3Speller* antes de cada sesión. En la figura 5-1 podemos observar la interfaz gráfica de esta herramienta.

La segunda parte de la sesión consistió en realizar 3 tareas diferentes manejando la aplicación. En esta parte se utiliza el clasificador obtenido anteriormente.

La primera tarea propuesta consistía en el manejo de la actividad *galería* que incluye la aplicación AppRunner. Primero, se abre la galería, apareciendo una fotografía precargada en la memoria. Esta fotografía esta girada, de manera que el usuario debe rotarla para una correcta visualización y pausar la aplicación para poder fijar su atención en el dispositivo. Se recomendó que el usuario pausara la aplicación el tiempo correspondiente a dos secuencias de selección de P3Speller. Posteriormente, se visualiza otra imagen precargada en memoria y se cierra la galería.

La segunda tarea de esta parte de la sesión consistió en poner una alarma para la próxima vez que fueran las 14:30 horas. Además del desarrollo normal de la tarea, se propusieron para su selección otro tipo de comandos de la matriz de teclado para instruir al usuario sobre la navegación por los campos de los formularios de la aplicación usando los comandos “ENTER” y “TAB_BACK”.

La tercera tarea consistió en una llamada telefónica a un contacto ya guardado con el nombre de Aa. Esta tarea secuencia permite introducir la funcionalidad de llamadas, además de introducir el funcionamiento del comando “MAYUS”.

Las tablas 5-1 y 5-2 resumen el procedimiento de evaluación de esta primera sesión.

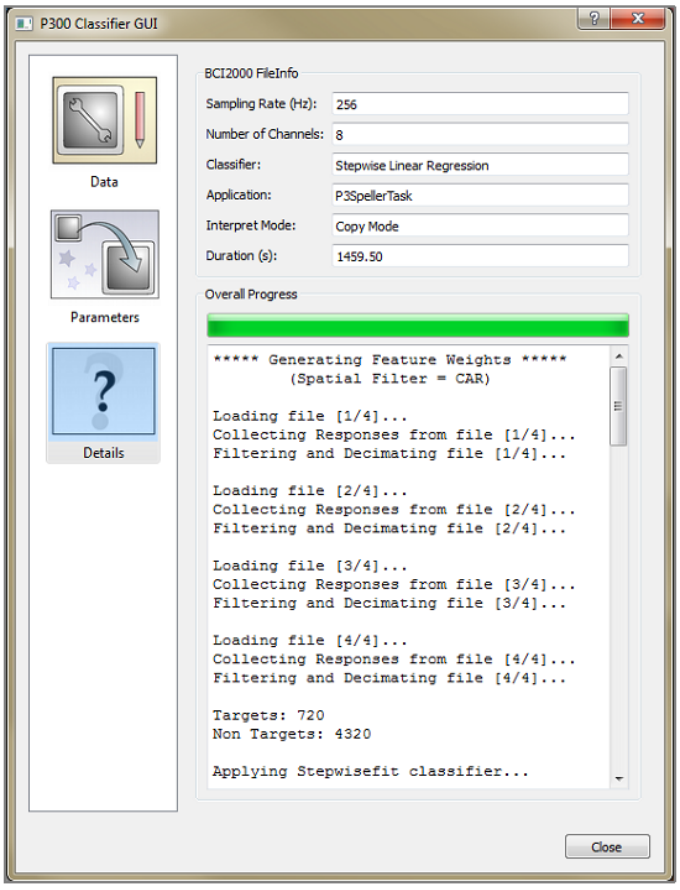

Figura 5-1. Interfaz gráfica de la herramienta P3Classifier, utilizada para hallar el vector de pesos adaptado para cada usuario a partir de una serie de muestras de entrenamiento

CALIBRACIÓN DEL CLASIFICADOR	
CONFIGURACIÓN	15 SECUENCIAS con MATRIZ DE TECLADO 9x6
PROCEDIMIENTO	4 RONDAS de 6 TRIALS (4 palabras de 6 letras)
DURACIÓN APROX.	~26 min (~6 min/ronda + ~2 min SWLDA)
DESCRIPCIÓN	Se ordena al usuario escribir con la matriz de teclado cuatro palabras de seis letras cada una con el fin de obtener muestras de entrenamiento suficientes para hallar su clasificador óptimo. Posteriormente, con SWLDA se calculan los pesos y el número de secuencias óptimos (X) mediante la aplicación P300Classifier.

Tabla 5-1. Calibración del clasificador. Primera parte de la primera sesión.

TAREA 1	**MANEJO DE GALERÍA**
SELEC. MÍNIMAS	6 selecciones
SELEC. ÓPTIMAS	IC.GALERIA IC.ROTAR_IMAGEN PAUSA (mirar foto) PAUSA IC.SIG_IMAGEN IC.VOLVER
DURACIÓN MÍNIMA.	~4 min (X=15 secuencias)
MATRICES A UTILIZAR	Matriz de aplicaciones
DESCRIPCIÓN	El usuario probará las funcionalidades de la galería de imágenes, así como del modo pausa de la aplicación mientras visualiza la fotografía.
TAREA 2	**MANEJO DE ALARMAS**
SELEC. MÍNIMAS	10 selecciones
SELEC. ÓPTIMAS	IC.ALARMA 1 4 : ENTER PREV_EDIT 3 0 ENTER ENTER
DURACIÓN MÍNIMA.	~7 min (X=15 secuencias)
MATRICES A UTILIZAR	Ambas
DESCRIPCIÓN	El usuario probará a activar una alarma a las 14:30. Además se introducirá la navegación entre campos del formulario (ENTER, PREV_EDIT).
TAREA 3	**MANEJO DE LLAMADA**
SELEC. MÍNIMAS	11 selecciones
SELEC. ÓPTIMAS	IC.LLAMAR 6 5 3 2 CLEAR MAYUS A MAYUS a ENTER
DURACIÓN MÍNIMA.	~9 min (X=15 secuencias)
MATRICES A UTILIZAR	Ambas
DESCRIPCIÓN	El usuario probará las funcionalidades de la aplicación de llamadas, así como la opción limpiar el campo de formulario (CLEAR), o escribir con mayúsculas (MAYUS). El contacto Aa será un contacto guardado previamente con el número del usuario, que vera recibir la llamada en su teléfono.

Tabla 5-2. Segunda parte de la primera sesión, dividida en 3 tareas diferentes.

5.1.2. Segunda sesión

La segunda sesión tiene como objetivo una evaluación completa de las funcionalidades de la aplicación. Al igual que la primera, también se divide en dos partes.

La primera consiste en una evaluación del clasificador calculado en la anterior sesión, y si fuera necesario, actualizarlo. Esto se debe a las diferencias en las características de los P300 de cada sujeto entre las dos sesiones, que recordemos, se realizan en días diferentes, lo que puede afectar en la precisión del clasificador. Para ello se realizarán 2 rondas de entrenamiento de 6 *trials* cada una. Si la precisión del clasificador no fuera la suficiente, este se actualizará calculando los pesos de nuevo con los nuevos archivos generados. El tiempo estimado para esta parte son 8 minutos

Una vez comprobada la precisión del clasificador, pasamos a la segunda parte de la sesión. En esta parte se realiza un manejo completo de la aplicación AppRunner aumentando la dificultad de las tareas realizadas respecto a la primera sesión. Se proponen 5 tareas con duración y dificultad crecientes. La duración aproximada de la segunda parte de la sesión son 40 minutos.

La primera tarea consiste en realizar una fotografía con la aplicación cámara incorporada en AppRunner.

La segunda tarea consiste en ver la fotografía realizada anteriormente en la aplicación galería incorporada en AppRunner. Además, se tendrá que rotar para una correcta visualización y también se verá la siguiente imagen.

La tercera tarea consiste en añadir un contacto al teléfono. Se sugiere a los usuarios que añadan el contacto con el numero propio de cada uno, para seguir probando más funcionalidades posteriormente y poder ver los resultados en el momento. El nombre de contacto debe tener 4 letras y puede ser el que quieran los usuarios. Esta tarea es una de las más complicadas, ya que tiene 17 selecciones y un tiempo estimado de 13 minutos.

La cuarta tarea propuesta consistió en llamar al contacto recién añadido, con el número de teléfono del sujeto utilizando la función “LLAMAR”.

La quinta y última tarea fue la de mandar un SMS al contacto añadido anteriormente con el mensaje “hola mundo”.

La información de la sesión, así como las secuencias concretas propuestas, se resume en las tablas 5-3 y 5-4.

COMPROBACIÓN DEL CLASIFICADOR	
CONFIGURACIÓN	15 SECUENCIAS CON MATRIZ DE TECLADO 6X9
PROCEDIMIENTO	1 o 2 RONDAS de 6 TRIALS
DURACIÓN APROX.	~4 min o ~8 min
DESCRIPCIÓN	Puesto que ésta es la segunda sesión, es necesario comprobar el comportamiento del clasificador hallado previamente. Probablemente, las variaciones inter-sesión hayan causado un mal comportamiento del clasificador, así que se realizan 1 o 2 rondas de 6 trials, según el caso particular, y se crea un nuevo clasificador a partir del anterior con estas nuevas muestras de entrenamiento.

Tabla 5-3. Comprobación del clasificador para actualizarlo si fuera necesario.

TAREA 1	REALIZAR FOTOGRAFÍA
SELEC. MÍNIMAS	3 selecciones
SELEC. ÓPTIMAS	IC.CAMARA IC.HACER_FOTO IC.VOLVER
DURACIÓN MÍNIMA	~2 min (si X = 15 secuencias)
MATRICES A UTILIZAR	Matriz de aplicaciones
DESCRIPCIÓN	El objetivo del usuario es realizar una fotografía utilizando AppRunner. Para ello se parte del estado principal de la aplicación con el servidor Bluetooth ya funcionando. Posteriormente el usuario deberá seleccionar el icono CÁMARA, que abrirá la aplicación de cámara. A continuación, seleccionará HACER_FOTO para realizar la fotografía. Para volver a la actividad principal de la aplicación seleccionará VOLVER.
TAREA 2	**VISUALIZAR LA FOTOGRAFÍA REALIZADA ANTERIORMENTE**
SELEC. MÍNIMAS	5 selecciones
SELEC. ÓPTIMAS	IC.GALERÍA IC.SIG_IMAGEN IC.ANT_IMAGEN IC.ROTAR_FOTO IC.VOLVER
DURACIÓN MÍNIMA	~4 min (si X = 15 secuencias)
MATRICES A UTILIZAR	Matriz de aplicaciones
DESCRIPCIÓN	Partiendo de la actividad principal, el usuario seleccionará la aplicación galería, donde visualizará la fotografía realizada anteriormente. Para ello, pasará a través de dos fotografías precargadas con el botón siguiente imagen. Además, para una mejor visualización (Se tomará la foto von el móvil en vertical, por lo que la imagen aparecerá girada), el usuario seleccionará el icono de rotar fotografía. A continuación seleccionará el icono volver para regresar a la actividad principal.
TAREA 3	**AÑADIR UN CONTACTO**
SELEC. MÍNIMAS	17 selecciones
SELEC. ÓPTIMAS	IC.AÑADIR_CONTACTO P E P E ENTER 6 6 6 6 6 6 6 6 6 ENTER ENTER
DURACIÓN MÍNIMA	~13 min (si X = 15 secuencias)
MATRICES A UTILIZAR	Ambas
DESCRIPCIÓN	Partiendo de la actividad principal, el usuario seleccionará la aplicación agregar contacto. Se abrirá un formulario donde introducirá el nombre del contacto y un número de teléfono. Con objeto de homogeneizar la tarea se ha determinado que el nombre de contacto sea Pepe, y el número de teléfono el del propio del usuario.
TAREA 4	**MANEJO DE LLAMADA**
SELEC. MÍNIMAS	6 selecciones
SELEC. ÓPTIMAS	IC.LLAMAR P E P E Enter
DURACIÓN MÍNIMA.	~6 min (X=15 secuencias)
MATRICES A UTILIZAR	Ambas
DESCRIPCIÓN	El usuario probará las funcionalidades de la aplicación de llamadas con el contacto añadido durante la sesión en la tarea anterior (deberá tener un teléfono valido de una persona presente en el laboratorio), que vera recibir la llamada en su teléfono.
TAREA 5	**MANDAR SMS**
SELEC. MÍNIMAS	17 selecciones
SELEC. ÓPTIMAS	IC.SMS P E P E ENTER H O L A SPACE M U N D O ENTER
DURACIÓN MÍNIMA	~12 min (si X = 15 secuencias)
MATRICES A UTILIZAR	Ambas
DESCRIPCIÓN	Partiendo de la actividad principal, el usuario selecciona el icono SMS. Se abrirá en el móvil un formulario para introducir el contacto o número de teléfono y el texto a enviar. Cuando estén rellenados satisfactoriamente, se enviará el SMS pulsando ENTER y volverá a la matriz de selecciones pulsando MTX.

Tabla 5-4. Tareas de la segunda sesión.

Al término de la sesión se entregó un cuestionario a los sujetos para conocer su opinión sobre la aplicación.

El cuestionario entregado se adjunta a continuación. Se compone de 14 afirmaciones a las que se dará una puntuación del 1 al 7 según sea el grado de conformidad del sujeto con la cuestión. Así, el número uno se corresponde con "totalmente en desacuerdo", el cuatro con "ni mucho/ni poco" y el 7 con "totalmente de acuerdo".

Para evitar una respuesta automática de los sujetos, se alternó una cuestión con connotaciones positivas con otra cuestión con connotaciones negativas, evitando así que el sujeto dejara de pensar las respuestas

En el cuestionario fue incluida una última cuestión con el objetivo de que los sujetos pudieran realimentar el proyecto con alguna sugerencia, las cuales se recogen en esta sección.

CUESTIONARIO DE SATISFACCIÓN DE LA APLICACIÓN AppRunner

Valore las siguientes afirmaciones sobre la aplicación BCI "AppRunner":

1. Me ha resultado interesante conocer y usar esta aplicación BCI.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

2. La aplicación me ha resultado difícil de usar.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

3. Se han cumplido mis expectativas respecto a la funcionalidad de la aplicación.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

4. Las sesiones de pruebas me han parecido aburridas.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

5. Me imagino utilizando esta aplicación en mi vida diaria.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

6. Las sesiones requieren demasiada concentración, lo que me ha terminado cansando.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

7. La aplicación ofrece unas funcionalidades suficientes para cubrir mi utilización diaria de un dispositivo móvil.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

8. Me ha resultado difícil seleccionar los comandos deseados.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

9. La navegación por la aplicación es sencilla e intuitiva.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

10. La duración de las sesiones me ha parecido excesiva.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

11. La aplicación responde con suficiente velocidad y fluidez.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

12. Creo que el manejo de las funcionalidades implementadas es complicado y podría simplificarse

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

13. Me encantaría participar de nuevo en un estudio de estas características.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

14. Los efectos visuales de la matriz de selección me han causado fatiga

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

15. ¿Qué aspectos de la aplicación mejoraría y por qué?

CAPÍTULO 6

RESULTADOS

En este capítulo se explicarán los resultados de las sesiones de evaluación propuestas desglosándolas por sujetos y tareas. Además, se realizará un breve comentario del desempeño de cada sujeto y se presentará una tabla resumen de los resultados que obtuvo.

6.1. TAREAS PROPUESTAS

Los siguientes resultados a presentar son los obtenidos de la evaluación de la aplicación por parte de los sujetos. Para ello se realizaron cuatro tareas distintas: visualización de una web, envío de un e-mail, búsqueda en *Google* y lectura de una página web.

Estos resultados se presentan organizados por sujetos, tal y como se ha hecho en el apartado anterior con los resultados sobre la determinación del umbral y organizados por tablas con los datos necesarios para realizar correctamente la evaluación. Estos datos son los siguientes:

- **Secuencias recomendadas:** Numero de secuencias para las que estadísticamente, según el clasificador calculado por P3Classifier, la precisión debería de ser del 100%. Este número de secuencias fue el especificado en los parámetros correspondientes de P3Speller para la realización de las pruebas.
- **Selecciones óptimas**: número de selecciones mínimas necesarias para terminar la tarea propuesta.
- **Selecciones finales**: número de selecciones finales realizadas por el usuario para terminar la tarea propuesta. Obviamente, este número debe ser mayor o igual que las selecciones óptimas, concordando con la fórmula (5.1).
- **Selecciones adicionales**: número de selecciones que fueron necesarias para solventar un error cometido por el usuario.
- **Errores**: número de selecciones que el clasificador y el umbral discriminaron erróneamente.
- **Duración**: como su nombre indica, la duración de la prueba para ese usuario en concreto.
- **Precisión**: porcentaje de selecciones correctas sobre el total de selecciones realizadas para una tarea concreta. La precisión se calcula atendiendo a la fórmula (5.2).

$$Precisión = \frac{Selec.Finales - Errores}{Selec.Totales} \quad (5.2)$$

- **Precisión total**: porcentaje de selecciones correctas sobre el total de selecciones realizadas para todas las tareas, atendiendo a la fórmula (5.3).

$$Precisión\ Total = \frac{\sum_{Tarea} Selec.Finales - \sum_{Tarea} Errores}{\sum_{Tarea} Selec.Totales} \quad (5.3)$$

6.1.1. Sujeto S1

El sujeto S1 desarrolló las tareas propuestas con gran precisión. Sus porcentajes del 96.55% para la primera sesión y 95.31% para la segunda sesión son un excelente resultado. Tan solo cometió 4 errores en el transcurso de todas las pruebas.

El primer error lo cometió en la primera sesión, durante la tarea 2. Esta consistía en poner una alarma, como ya hemos visto en el anterior capítulo. La equivocación vino en la primera selección, quizá por estar desconcentrado, seleccionando el comando de "AÑADIR CONTACTO" en vez de seleccionar "ALARMA". El sujeto solvento la equivocación seleccionando "MTX" en la matriz de teclado y siguiendo con la tarea con normalidad.

En la segunda sesión tuvo 3 errores. En la primera tarea y la más fácil, cometió 2 errores seguidos en la primera selección. El comando seleccionado fue "HACER FOTO" por dos veces, en vez de "CAMARA", que era el correcto. No hizo falta ninguna selección extra para solventar la equivocación ya que el comando "HACER FOTO" está bloqueado hasta que se abra la cámara.

Las tablas 6-1 y 6-2 contienen los resultados completos de las sesiones.

SUJETO S1	SESIÓN 1		
	TAREA 1	TAREA 2	TAREA 3
Sec. Recomendadas	12		
Sec. Óptimas	6	10	11
Sec. Finales	6	12	11
Sec. Adicionales	0	1	0
Errores	0	1	0
Duración	4:50 min	7:50 min	8:00 min
Precisión	100%	91.66%	100%
Precision total	96.55%		

Tabla 6-1. Datos recogidos durante la primera sesión durante la parte de evaluación de la aplicación para el sujeto S1

SUJETO S1	SESIÓN 2					
	EVALUACION CLASIFICADOR	TAREA 1	TAREA 2	TAREA 3	TAREA 4	TAREA 5
Sec.Recomendadas	12					
Actualización clasificador	No					
Sec. Óptimas	12	3	5	17	6	17
Sec. Finales	12	5	5	17	6	19
Sec. Adicionales	0	0	0	0	0	1
Errores	0	2	0	0	0	1
Duración	9:10 min	3:50 min	6:30 min	12:20 min	5:00 min	13:00 min
Precisión	100%	60%	100%	100%	100%	94.7%
Precision total	95.31%					

Tabla 6-2. Datos recogidos en la segunda sesión para el sujeto S1

6.1.2. Sujeto S2

Las precisiones del sujeto 2 son muy parecidas a las del primer sujeto. En la primera sesión alcanzo un 96.55%, es decir, la misma que el sujeto S1 y 95.16% para la segunda sesión. Se cometieron un total de 4 errores, los mismo que el sujeto anterior.

El error cometido en la primera sesión se da en la segunda tarea, en la que se tiene que poner una alarma a las 14:30. El usuario introduce una h en el campo de la hora, de manera que tiene que borrar la letra con el comando "DEL", resultando en una selección adicional.

De los 3 errores cometidos en la segunda sesión, dos se dan en la evaluación del clasificador. Por ello, se decide actualizar el vector de pesos del usuario y recalcular el clasificador con los nuevos datos aportados por las dos rondas de entrenamiento de evaluación del clasificador. Gracias a esta actualización, el usuario consigue un gran porcentaje en el resto de tareas de la sesión, cometiendo un único error en la tercera tarea, una de las más difíciles.

Cabe destacar que durante esta sesión hubo que reparar un electrodo, concretamente el P4 debido a que estaba defectuoso. El efecto fue que el clasificador no podía asignar los pesos correctamente. Para solucionarlo se repitieron las rondas de clasificación una vez reparado.

SUJETO S2	SESIÓN 1		
	TAREA 1	TAREA 2	TAREA 3
Sec. Recomendadas	10		
Sec. Óptimas	6	10	11
Sec. Finales	6	12	11
Sec. Adicionales	0	1	0
Errores	0	1	0
Duración	4:10 min	6:40 min	7:40 min
Precisión	100%	91.66%	100%
Precision total	96.55%		

Tabla 6-3. Datos recogidos durante la primera sesión durante la parte de evaluación de la aplicación para el sujeto S2

SUJETO S2	SESIÓN 2					
	EVALUACION CLASIFICADOR	TAREA 1	TAREA 2	TAREA 3	TAREA 4	TAREA 5
Sec.Recomendadas	12					
Actualización clasificador	Si					
Sec. Óptimas	12	3	5	17	6	17
Sec. Finales	12	3	5	19	6	17
Sec. Adicionales	0	0	0	1	0	0
Errores	2	0	0	1	0	0
Duración	9:10 min	2:40 min	6:30 min	12:20 min	5:00 min	12:00 min
Precisión	83.33%	100%	100%	94.73%	100%	100%%
Precision total	95.16%					

Tabla 6-4. Datos recogidos en la segunda sesión para el sujeto S2

6.1.3. Sujeto S3

El sujeto tres consiguió unas también unas buenas precisiones para ambas sesiones, siendo especialmente buena la primera, con el 100%. En la segunda sesión cometió 4 errores, alcanzando una precisión del 93.91%.

El primer error cometido fue en la segunda tarea, tras seleccionar el comando "GALERIA". Dado que ya estaba abierta, el comando seleccionado no produjo ninguna acción en la aplicación de modo que no hizo falta corregirlo.

El segundo error, en la tarea siguiente, consistió en añadir al teléfono una letra, que hubo que borrar provocando una selección adicional para provocar el error

Los dos últimos errores se dieron en la última tarea, quizás fruto del cansancio del sujeto

SUJETO S3	SESIÓN 1		
	TAREA 1	TAREA 2	TAREA 3
Sec. Recomendadas	12		
Sec. Óptimas	6	10	11
Sec. Finales	6	10	11
Sec. Adicionales	0	0	0
Errores	0	0	0
Duración	4:10 min	6:40 min	7:40 min
Precisión	100%	100%	100%
Precision total	100%		

Tabla 6-5. Datos recogidos durante la primera sesión durante la parte de evaluación de la aplicación para el sujeto S3

SUJETO S3	SESIÓN 2					
	EVALUACION CLASIFICADOR	TAREA 1	TAREA 2	TAREA 3	TAREA 4	TAREA 5
Sec.Recomendadas	8					
Actualización clasificador	Si					
Sec. Óptimas	12	3	5	17	6	17
Sec. Finales	12	3	6	19	6	20
Sec. Adicionales	0	0	0	1	0	1
Errores	0	0	1	1	0	2
Duración	9:00 min	1:40 min	4:30 min	10:20 min	3:20 min	11:20 min
Precisión	100%%	100%	83.3%	94.73%	100%	90%
Precision total	93.91%					

Tabla 6-6. Datos recogidos en la segunda sesión para el sujeto S3

6.1.4. Sujeto S4

La diferencia entre las dos sesiones para el sujeto S4 fue muy grande. En la primera, se alcanzó un 87.5% de precisión debido a los cuatro errores cometidos, mientras que en la segunda sesión obtuvo un 98.3% y 3 errores, una cifra mucho mejor.

El transcurso de la primera sesión, con dos errores en la primera tarea y uno para cada una de las dos restantes, hizo pensar que quizá el clasificador podía mejorarse para este sujeto.

En la segunda sesión y a pesar de que se obtuvo un 100% de precisión en la evaluación del clasificador de la primera sesión, se actualizó el clasificador con un gran resultado, permitiendo incluso bajar el número de secuencias para cada selección a 9. El resto de la sesión transcurrió con buenos resultados con tan solo 3 errores cometidos en las dos tareas más difíciles.

Otro factor que pudo intervenir en la baja precisión de la primera sesión puede haber sido que era la primera vez que este sujeto manejaba un sistema de características parecidas, ya que nunca había participado en un experimento sobre BCI.

SUJETO S4	SESIÓN 1		
	TAREA 1	TAREA 2	TAREA 3
Sec. Recomendadas	10		
Sec. Óptimas	6	10	11
Sec. Finales	8	12	12
Sec. Adicionales	0	1	0
Errores	2	1	1
Duración	5:50 min	6:40 min	7:40 min
Precisión	75%	91.6%	91.6%
Precision total	87.5%		

Tabla 6-7. Datos recogidos durante la primera sesión durante la parte de evaluación de la aplicación para el sujeto S4

SUJETO S4	SESIÓN 2					
	EVALUACION CLASIFICADOR	TAREA 1	TAREA 2	TAREA 3	TAREA 4	TAREA 5
Sec.Recomendadas	9					
Actualización clasificador	Si					
Sec. Óptimas	12	3	5	17	6	17
Sec. Finales	12	3	5	19	6	21
Sec. Adicionales	0	0	0	1	0	2
Errores	0	0	0	1	0	2
Duración	14:10 min	2:20 min	4:30 min	10:20 min	3:50 min	12:00 min
Precisión	100%	100%	100%	94.73%	100%	90.47%
Precision total	98.3%					

Tabla 6-8. Datos recogidos en la segunda sesión para el sujeto S4

6.1.5. Sujeto S5

El sujeto S5 mostró una gran habilidad en la primera sesión, obteniendo un clasificador estadísticamente bueno a partir de la 5 secuencia de selección. Además, el 96.55% de precisión alcanzada con un solo error, demuestran la gran concentración que mostró el sujeto durante la primera sesión.

En la segunda sesión el sujeto cometió dos fallos durante la evaluación del clasificador, lo que sugería que era necesaria una actualización del clasificador. Esta vez, el número de secuencias recomendado era de 6.

Durante la sesión el sujeto afirmó estar cansado y tener sueño, lo que quizás produjo la disminución en la precisión respecto al primer día. En esta sesión alcanzó un 91.17% de precisión cometiendo 6 errores, que, a pesar de ser la peor cifra de este estudio, se encuentra dentro de unos límites razonablemente buenos.

SUJETO S5	SESIÓN 1		
	TAREA 1	TAREA 2	TAREA 3
Sec. Recomendadas	5		
Sec. Óptimas	6	10	11
Sec. Finales	6	10	13
Sec. Adicionales	0	0	1
Errores	0	0	1
Duración	2:50 min	4:20 min	5:00 min
Precisión	100%	100%	92.23%
Precision total	96.55%		

Tabla 6-9. Datos recogidos durante la primera sesión durante la parte de evaluación de la aplicación para el sujeto S5

SUJETO S5	SESIÓN 2					
	EVALUACION CLASIFICADOR	TAREA 1	TAREA 2	TAREA 3	TAREA 4	TAREA 5
Sec.Recomendadas	6					
Actualización clasificador	Si					
Sec. Óptimas	12	3	5	17	6	17
Sec. Finales	12	3	7	19	6	21
Sec. Adicionales	0	0	1	1	0	2
Errores	2	0	1	1	0	2
Duración	7:10 min	2:40 min	5:30 min	9:10 min	3:40 min	12:00 min
Precisión	83.33%	100%	85.71%	94.73%	100%	90.4%
Precision total	91.17%					

Tabla 6-10. Datos recogidos en la segunda sesión para el sujeto S5

6.2. CUESTIONARIO DE SATISFACCIÓN

Como se ha comentado en apartados anteriores, al finalizar la segunda sesión se entregó al usuario un cuestionario de satisfacción. Los resultados del cuestionario se pueden ver en la tabla 6-11.

Cuestiones planteadas	S1	S2	S3	S4	S5	Media
Me ha resultado interesante conocer esta aplicación BCI	7	6	6	7	5	6.2
La aplicación me ha resultado difícil de usar	2	3	1	1	3	2
Se han cumplido mis expectativas respecto a la funcionalidad de la aplicación	7	6	6	6	5	6
Las sesiones de pruebas me han parecido aburridas	3	2	5	2	5	3.4
Me imagino utilizando esta aplicación en mi vida diaria	4	4	4	4	2	3.6
Las sesiones requieren demasiada concentración, lo que me ha terminado cansando	4	6	6	5	5	5.2
La aplicación ofrece suficientes funcionalidades para cubrir mis necesidades de utilización de mi Smartphone	6	3	7	5	6	5.4
Me ha resultado difícil seleccionar los comandos deseados	2	3	4	2	4	3
La navegación por la aplicación es sencilla e intuitiva	6	4	7	7	5	5.8
La duración de las sesiones me ha parecido excesiva	2	4	5	2	4	3.4
La aplicación responde con fluidez y velocidad	7	5	5	5	5	5.4
Creo que el manejo de las funcionalidades implementadas es coplicado y podría simplificarse	2	2	2	2	3	2.2
Me encantaría participar de nuevo en un estudio de estas características	4	5	5	7	2	4.6
Los efectos visuales de selección me han causado fatiga	5	2	4	1	4	3.2

Tabla 6-11. Datos recopilados en los cuestionarios

Sugerencias de S1

- Añadir más funcionalidades, como un manejo completo de Whatsapp
- Diseñar la aplicación de manera que se pueda prescindir del ordenador convirtiéndola en una aplicación realmente móvil.

Sugerencias de S2

- Añadir una detección de umbral que permita mirar el móvil sin que seleccione nada en la matriz *oddball*

Sugerencias de S3

- Poder pausar la aplicación estando situados en la matriz de teclado
- Desarrollar una herramienta de depuración para poder controlar toda la aplicación manualmente además de mediante BCI.
- Implementar alguna herramienta asíncrona que permita ignorar la matriz si el usuario lo desea

Sugerencias de S4

- Implementar más funcionalidades, como leer un email o Whatsapp.
- Incrementar la pausa entre la selección de un comando y el siguiente para poder observar el móvil y ver la anterior instrucción se ha ejecutado correctamente

Sugerencias de S5

- Mejorar la capacidad de Whatsapp

CAPÍTULO 7

DISCUSIÓN

7.1. TAREAS PROPUESTAS

La respuesta de la aplicación durante las sesiones ha sido completamente satisfactoria. La selección de tareas con progresiva dificultad ha resultado ser adecuada, ya que el porcentaje de error conseguido durante las sesiones de pruebas consigue superar en varios casos a la mayoría de los estudios analizados, que se sitúan en torno al 92% de acierto.

El bajo porcentaje de error en la clasificación sumado a la fluidez y facilidad en el manejo de la aplicación, que permiten mantener toda la atención en el sistema de matrices *oddball,* han permitido que las sesiones se desarrollen correctamente y los sujetos involucrados en el estudio hayan valorado positivamente la experiencia.

La figura 7-1 nos muestra una comparativa de la precisión alcanzada por cada sujeto en la segunda sesión de evaluación desglosada por tareas.

En la figura 7-2 se presentan unos gráficos circulares con los datos referentes a la precisión de los sujetos en la segunda sesión, sin contar la evaluación del clasificador.

Un resultado inesperado que resulta de la observación de la gráfica de la figura 7-1 es que la tarea 3, siendo la más difícil junto con la 5 con 17 selecciones, alcanza la segunda mejor precisión media. Esto puede explicarse desde el punto de vista psicológico de los sujetos, que quizá afrontaron esta tarea con más concentración debido a su mayor dificultad.

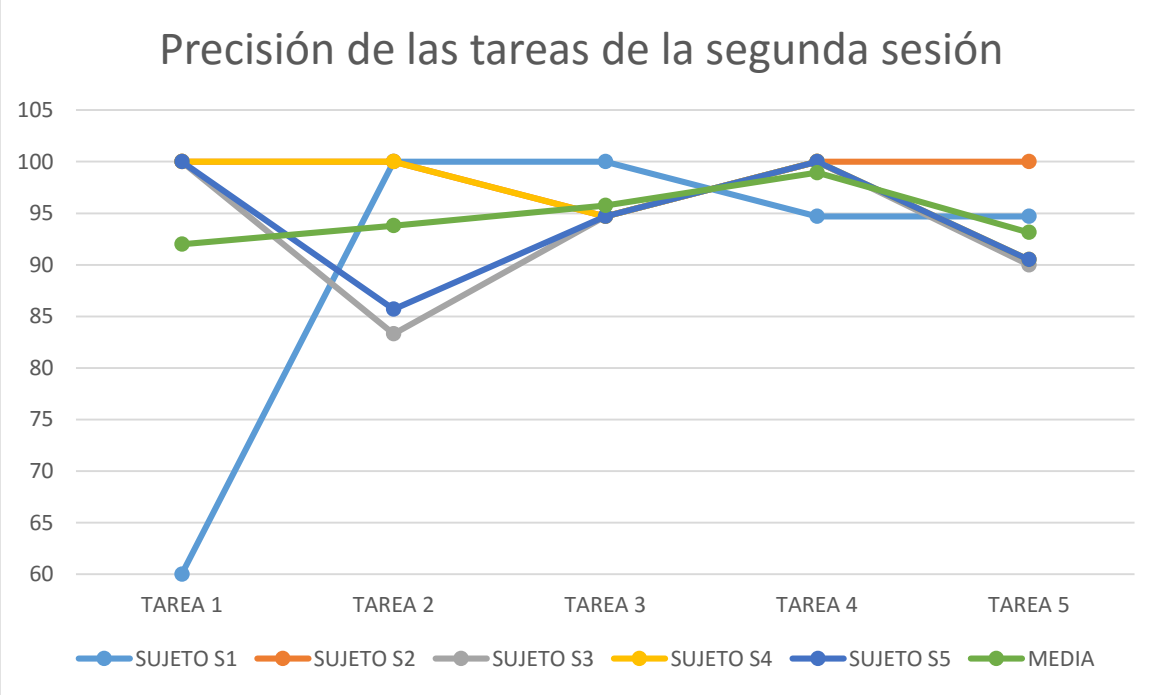

Figura 7-1. Gráfica comparativa de la precisión de las tareas de la segunda sesión

La tarea 5, sin embargo, tiene la segunda peor media a pesar de tener las mismas selecciones que la tarea tres. Quizás el hecho de que se realizaba al final de la sesión, donde los sujetos estaban más cansados, afectó en la precisión alcanzada.

Otro resultado que llama la atención es que la primera tarea, la más fácil con tres selecciones, tiene la peor media. Sin embargo, esto no es significativo, ya que todos los sujetos menos el uno, que tuvo dos errores, alcanzaron el 100% de precisión. La diferencia radica en el porcentaje que significa un error en cada tarea, siendo más grande este cuantas menos selecciones requiera.

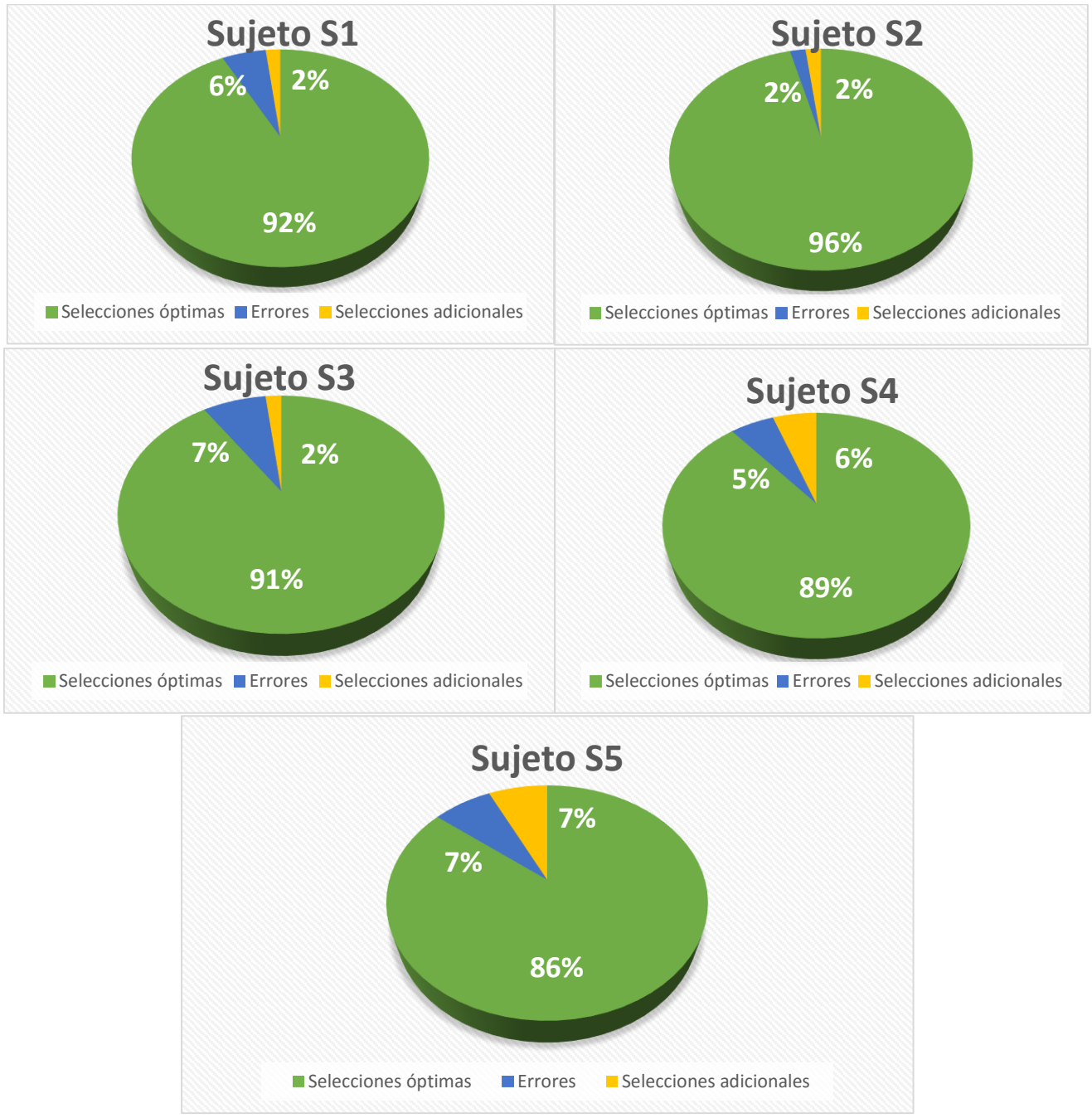

Figura 7-2. Gráfica con los datos de la segunda sesión para los cinco sujetos

7.2. CUESTIONARIO DE SATISFACCIÓN

También se puede extraer bastantes conclusiones del cuestionario que los sujetos rellenaron para dar a conocer su opinión de la aplicación. En este punto se realizará un simple análisis estadístico y se realizarán unos comentarios teniendo en cuenta los datos calculados.

En la figura 7-3 podemos observar un gráfico de barras con la media obtenida en cada pregunta, mientras que en la figura 7-4 podemos observar la varianza de las respuestas. En verde se pueden observar las preguntas con connotaciones positivas, donde una mejor nota es teóricamente mejor, y en rojo las que tienen connotaciones negativas, donde el objetivo es alcanzar la nota más baja.

La media de la cuestión C1 (*Me ha resultado interesante conocer y usar esta aplicación*) es muy buena, lo que sugiere que la aplicación ha generado interés en los sujetos. Otra de las notas más altas se la lleva la cuestión C3 (*se han cumplido mis expectativas respecto a la funcionalidad de la aplicación*), por tanto, la aplicación ha cumplido con lo que los usuarios esperaban de ella.

La media obtenida en la cuestión C6 (*las sesiones requieren demasiada concentración, lo que me ha terminado cansando*) hace pensar que los sujetos estuvieron muy concentrados durante las sesiones, y de ahí los buenos resultados en la precisión. Sin embargo, esto les ha terminado cansando y posiblemente con un uso más continuado de la aplicación no mantendrían tanto la concentración.

Los usuarios también coincidieron en que la aplicación era fácil de usar valorando positivamente y con pequeña varianza las preguntas C2 (*la aplicación me ha resultado fácil de usar)*, C9 (*La*

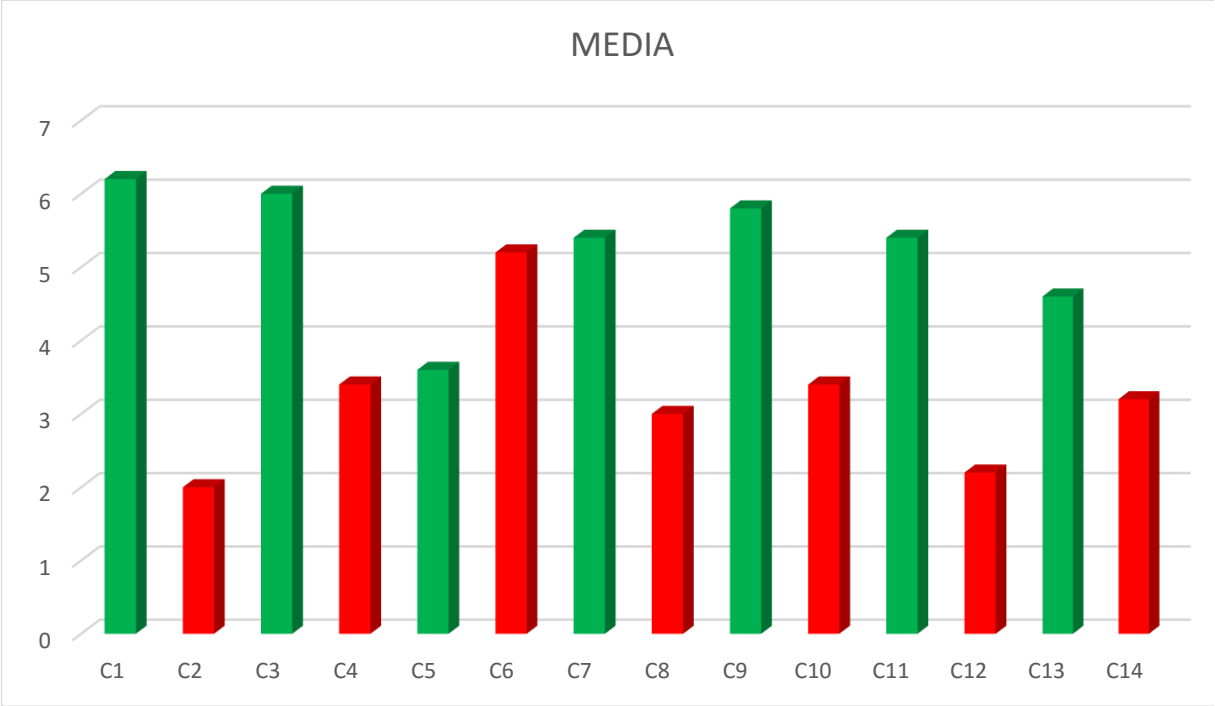

Figura 7-3. Gráfico de barras para los resultados medios del cuestionario por pregunta. En verde se muestran las preguntas con una connotación positiva, donde más nota es mejor. En rojo las que tienen una connotación negativa, donde menor nota es mejor.

navegación por la aplicación es sencilla e intuitiva) y C12 (*El manejo de las funcionalidades implementadas es complicado y podría simplificarse).*

La media obtenida en las respuestas a la pregunta C5 (*me imagino utilizando esta aplicación en la vida diaria*) se explica desde el hecho de que la aplicación está diseñada para personas con grave discapacidad, mientras que los sujetos de pruebas eran personas sanas.

Desde el punto de vista de la homogeneidad de las respuestas de los sujetos, se pueden hacer también algunos comentarios interesantes.

Como vemos en el gráfico de la varianza, las preguntas con respuestas más dispares son C4 (*las sesiones me han parecido aburridas),* C7 (*la aplicación ofrece unas funcionalidades suficientes para cubrir mi utilización diaria de un dispositivo móvil*) lo que sugiere que, aunque la media de la pregunta sea buena, las funcionalidades se quedan escasas según el tipo de usuario, C13 (*Me encantaría participar en un estudio de estas características*) con algunos sujetos que si volverían a participar y otros que no, pudiendo ser la causa la mayor experiencia en aplicaciones parecidas de algunos sujetos, habiendo participado ya en varios estudios similares, y C14 (*Los efectos visuales de la matriz me han causado fatiga*). La disparidad en la última pregunta puede deberse a las horas en las que se hayan realizado las pruebas o las condiciones particulares cada usuario en los días de las sesiones.

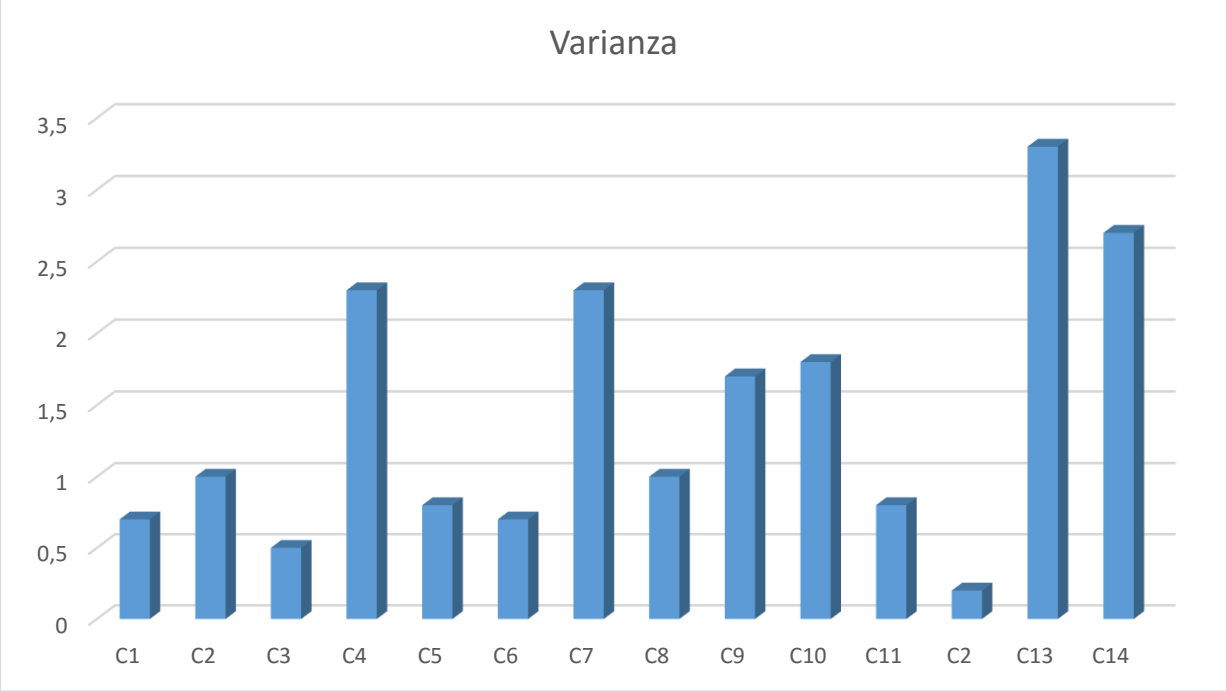

Figura 7-4. Gráfico con la varianza obtenida en las respuestas a las preguntas

7.3. COMPARACIÓN CON OTROS ESTUDIOS

En el capítulo dos podemos encontrar un estudio sobre distintos los proyectos que han implementado aplicaciones funcionales BCI sobre plataformas móviles.

A pesar de que es un campo con grandes aplicaciones prácticas, aún no se han desarrollado la variedad de aplicaciones BCI que se han desarrollado sobre otras plataformas. Esto es debido, quizás, a la reciente irrupción de dispositivos móviles con la capacidad de procesamiento suficiente para ejecutar aplicaciones BCI.

Seguramente, la aplicación con más similitudes con la desarrollada en el presente trabajo es la de A. S. Elsawy and S. Eldawlatly [27] *et al*. Este sistema BCI con señal de control P300 consistía en dos aplicaciones instaladas en un dispositivo Android. La primera permitía abrir cualquiera de las aplicaciones instaladas, sin poder controlarlas o realizar acciones sobre ellas, mientras que la segunda consistía en una sencilla galería de imágenes como la presentada en este trabajo.

A pesar de que las funcionalidades del sistema de A. S. Elsawy and S. Eldawlatly *et al* son muy limitadas en comparación con las presentadas en este proyecto, su gran logro es realizar todo el procesado de señal en el dispositivo móvil, convirtiendolo así en un verdadero sistema BCI portable. Sin duda este debe ser uno de los objetivos primordiales en la mejora de la aplicación AppRunner presentada en este proyecto. Sin embargo, las precisiones que se alcanzaron, por debajo del 90%, hacen pensar que aún queda mucho trabajo por realizar en este sentido.

Otro sistema muy interesante es el desarrollado por Yu-Te Wang *et al* [24], el cual utiliza potenciales evocados visuales para su control en vez de P300, y una pantalla mas grande que la del dispositivo para la estimulación. Como el anterior estudio, realiza el procesado de la señal en el teléfono móvil, al que llegan los datos desde el amplificador directamente por Bluetooth. Su funcionalidad es limitada, pudiendo solamente realizar llamadas. Sin embargo, es otro ejemplo de que es posible realizar el procesado de señales EEG directamente en el dispositivo. Este estudio consiguió además una gran precisión, cercana al 96%.

La aplicación de Scott Vernon *et al* [26] utiliza ritmos sensoriomotores para mover un cursor en una y dos dimensiones sobre un dispositivo móvil. Lo curioso de este sistema es que realiza el procesado de señal en el móvil, convirtiéndola al dominio digital directamente con la entrada de audio y alcanzando grandes precisiones. Los resultados publicados en su artículo afirman que se ha obtenido una precisión del 100% para el caso 1D y de alrededor del 70% para el 2D. Las similitudes con AppRunner son escasas, pero es un ejemplo más de que se puede realizar el procesado de señales EEG en un Smartphone.

La arquitectura que ha servido de referencia en este proyecto ha sido la propuesta por Yu Zhou *et al* [25]. A pesar de que esta aplicación se basaba en señales de control de imagen motora, el concepto es el mismo. La señal EEG se procesaba en un ordenador, que traducía las caracteristicas de la señal EEG en comandos que se enviaban por Bluetooth al dispositivo móvil que los ejecutaba. En este caso, el objetivo era era controlar un juego instalado en el movil mediante tres comandos, por lo que en cuanto a funcionalidades, no se pueden comparar los estudios.

Así pues, de las aplicaciones BCI estudiadas de otros autores la presente es la que tiene mas funcionalidades con gran diferencia. Sin embargo, no ha logrado otros objetivos que si han alcanzado otros estudios, como un sistema realmente móvil, sin la necesidad de un ordenador para el procesado de la señal EEG.

La tabla 7-1 resume los puntos clave de la comparación de AppRunner con las otras aplicaciones móviles aquí expuestas

	Procesado de señal en el dispositivo	Señal de control	Funcionalidades	Sujetos de estudio	Precisión media
AppRunner	No	P300	LLamadas, sms, calendario, alarmas, contactos, cámara, galería, pausa	5 sujetos de control sanos	94.7 %
A.S. Elsawy *et al* [27]	Si	P300	Abrir aplicaciones (sin interactuar con ellas), galería	6 sujetos de control sanos	79.17% Apps. 87.5% Gallery
Yu-Te Wang *et al* [24]	Si	VEP	LLamadas	10 sujetos de control sanos	96%
Yu Zhou *et al* [25]	No	SCP	Control de un cursor con 3 movimientos, izquierda, derecha y no movimiento	1 sujeto de control sano	75%
S. Vernon *et al* [26]	Si	Ritmos μ y β	Movimiento de un cursor en una y dos dimensiones	1 sujeto con avanzado estado de parálisis (Atrofia Muscular Espinal)	100% 1D 70% 2D

Tabla 7-1. Comparación de 5 aplicaciones móviles basadas en BCI

7.4. LIMITACIONES DE LA APLICACIÓN

Debido a la ingente cantidad de tareas que se pueden realizar hoy en día con un Smartphone Android, las limitaciones en el control de un dispositivo de estas características aún son muchas.

Algunas de las funciones más importantes que no se incluyen son la navegación web o el manejo completo de aplicaciones de terceros, como puedan ser Whatsapp, Facebook o Twiter, las cuales no están diseñadas para ser controladas mediante una aplicación externa, debiendo estudiar el modo de interactuar con ellas.

Otra de las limitaciones consiste en que el sistema no es realimente móvil, ya que se necesita un ordenador para procesar la señal EEG y traducirla a comandos que se puedan enviar al móvil.

El estudio realizado de la aplicación ha mostrado muy buenos resultados en cuanto a precisión. Sin embargo, se necesita un estudio más amplio con sujetos gravemente discapacitados para obtener más datos y poder extraer conclusiones más generales.

Otro factor a tener en cuenta es que las pruebas se realizaron en un entorno controlado de laboratorio. Sería interesante también examinar la respuesta del sistema en entornos más cotidianos, con más distracciones.

A pesar de que la precisión del estudio ha sido muy buena, aún se puede mejorar. Se debe estudiar el desempeño de la aplicación con algoritmos de procesado más avanzados que puedan dar mejor resultado en cuanto precisión y número de iluminaciones necesarias para la clasificación.

Las respuestas del cuestionario para ver qué aspectos de la aplicación mejorarían los sujetos estudiados también arrojó ideas interesantes. El sujeto S2 señaló la importancia de una herramienta de umbralización para determinar si el sujeto está o no atendiendo a los estímulos visuales. En caso de que no estuviera prestando atención, el sistema no seleccionaría ningún comando de la matriz. Esta es claramente una funcionalidad a desarrollar.

Otras ideas recogidas en esta parte del cuestionario son la necesidad de implementar una funcionalidad completa de Whatsapp o lectura de emails, aumentar el tiempo entre selección de comandos para poder mirar el móvil, y poder pausar la aplicación en la matriz de teclado, además de en la de navegación.

CAPÍTULO 8

CONCLUSIONES Y LÍNEAS FUTURAS

8.1. CONCLUSIONES

En este trabajo se ha llevado a cabo un estudio completo sobre los sistemas BCI, especialmente aquellos que tienen como señal de control los P300.

En el primer capítulo se hace una introducción a las señales biomédicas, en especial la señal EEG, que ha sido la utilizada para el desarrollo de la aplicación por su simplicidad, resultados, y bajo coste. Posteriormente se han introducido, de manera general, los sistemas BCI y se han establecido los objetivos y la estructura del trabajo. Al final de este punto, se analiza cuáles de los objetivos presentados en este punto se han conseguido y cuáles no.

En el segundo capítulo se han analizado en profundidad los sistemas BCI. Tras un estudio exhaustivo, se concluyó que la mejor señal de control para el sistema a implementar eran los potenciales P300 por su aplicación en el uso del paradigma *oddball*. También se realizó un informe sobre el estado del arte de las aplicaciones BCI en plataformas móviles. Este estudio aportó muchas conclusiones respecto a la arquitectura de la aplicación a desarrollar, los métodos a seguir y los objetivos en cuanto a funcionalidades que se podían alcanzar.

De esta manera se decidió que la mejor manera de implementar el sistema actualmente es mediante una aplicación distribuida entre un ordenador, que recoge la señal EEG y la procesa, y el dispositivo móvil que queremos controlar.

En el tercer capítulo se realiza un examen más a fondo sobre los potenciales P300, elegidos como señal de control de la aplicación. Se estudian las variables que pueden afectar a estos potenciales, como adquirirlos y como procesarlos. Se concluye que un buen método para la extracción y traducción de las características de los P300 es el promediado-sincronizado de varias épocas de señal sumado a un algoritmo de clasificación SWLDA.

En cuanto al diseño y la arquitectura del sistema BCI, se desarrollan en el capítulo cuatro. Primeramente, se proponen una serie de funcionalidades objetivo, basándose en diferentes estudios sobre las características que más se usan de los Smartphones actuales. Se concluye que estos objetivos cubren, de manera suficiente, aproximadamente el 60% de las interacciones de una persona media con su teléfono móvil.

En cuanto a la arquitectura, se diferencian dos partes a implementar. La parte del ordenador debe procesar la señal EEG y enviar los comandos al dispositivo móvil. Se concluye que la mejor manera de implementarla es modificar el código de P3Sepller, una aplicación incluida dentro de la herramienta de propósito general BCI2000, incluyendo un algoritmo de traducción de la celda de la matriz *oddball* seleccionada al comando que quiere ejecutar el usuario y un cliente Bluetooth que lo envíe.

En la parte de la aplicación alojada en el dispositivo móvil se creará un servidor Bluetooth que reciba los comandos y se implementará la lógica para su ejecución. En este capítulo también se deciden los parámetros más adecuados del sistema para un correcto funcionamiento del mismo.

Los capítulos de resultados y discusión analizan el desempeño de la aplicación durante las pruebas realizadas tras ser terminada con cinco sujetos sanos. Los resultados alcanzan una gran precisión, superior en todos los casos al 90%. El análisis del cuestionario de satisfacción que cumplimentaron los usuarios muestra que los usuarios piensan que la aplicación es sencilla, fácil de usar y con funcionalidades suficientes. Como aspecto negativo encontramos la duración de las sesiones de entrenamiento y cálculo del clasificador, que se hicieron tediosas para algunos usuarios, y el cansancio que provocó el mantener la concentración tanto tiempo.

Respecto a los objetivos del trabajo fin de grado desarrollados en el capítulo uno, se han conseguido alcanzar todos, como vemos en el siguiente resumen de las conlusiones:

- Se han estudiado los sistemas de adquisición de señales EEG que registran la actividad cerebral.
- Se han estudiado las diferentes posibilidades de diseño de un sistema BCI en función de las señales gracias a las cuales se traducen las intenciones del usuario, además de los métodos de procesado de señal necesarios para ese objetivo.
- Se ha realizado una revisión del estado del arte de los sistemas BCI en dispositivos móviles desarrollados con anterioridad.
- Se han establecido los requisitos de la aplicación en cuestión, la señal de control a utilizar y las herramientas que se necesitan en el desarrollo y la arquitectura del sistema.
- Se ha evaluado la aplicación una vez finalizada en cinco sujetos sanos en dos sesiones distintas.
- Se han discutido los resultados obtenidos y comparación con otros estudios.
- Se han extracción de conclusiones a partir de dichos resultados

Las conclusiones extraídas del cumplimiento de los anteriores objetivos se resumen en la siguiente lista numerada:

1. La aplicación desarrollada adquiere la señal de control mediante EEG por ser este un método robusto, no invasivo, fácil y de bajo coste.
2. La señal de control utilizada son los P300, por ser esta sobre la que se desarrolla el paradigma *oddball*, implementado en la aplicación desarrollada.
3. La extracción de características se realiza mediante promediado-sincronizado, utilizando para su traducción el algoritmo SWLDA, por ser algoritmos robustos que ofrecen buenas prestaciones para el procesado en tiempo real, tanto en precisión como en coste computacional.
4. La arquitectura de la aplicación está distribuida entre un ordenador que realiza el procesado de señal y traduce las intenciones del usuario en comandos, y un dispositivo móvil que los ejecuta. La elección de esta estructura se basa en la mayor capacidad de procesamiento del ordenador, permitiendo analizar mejor la señal. Además, no existe posibilidad de conectar el amplificador g.USBamp a un dispositivo Android por la inexistencia de los *drivers* adecuados.
5. Las funcionalidades que implementa la aplicación son: Llamar, SMS, Manejar contactos, Realizar y ver fotografías, poner alarmas, añadir eventos a la aplicación calendario, precargar un texto para enviarlo mediante Whatsapp y pausar la aplicación. Esta elección se basa en el análisis de diversos estudios, concluyendo que estas funcionalidades son suficientes para cubrir satisfactoriamente el 60% de las interacciones con un Smartphone de un usuario medio.

6. Los resultados de la evaluación por 5 sujetos sanos han sido satisfactorios, con un 95% de precisión para las tareas propuestas.
7. La comparación con otros estudios sobre aplicaciones BCI implementadas sobre plataformas móviles nos dice que este es el sistema con más funcionalidades y la segunda mejor precisión. Como contrapartida, la señal se procesa en un ordenador.

8.2. LÍNEAS FUTURAS

Las líneas futuras que debe seguir este proyecto se extraen de sus limitaciones, las cuales debe subsanar.

La principal de todas ellas es convertir el sistema AppRunner en un sistema totalmente móvil, implementando la extracción y traducción de características directamente en el Smartphone Android donde esté instalada la aplicación.

También debe ser un objetivo aumentar las funcionalidades implementadas para cubrir las necesidades de todos los usuarios, desarrollando un navegador web y pudiendo utilizar y manejar aplicaciones de terceros. Otra línea futura es implementar la aplicación en otros sistemas operativos, como Windows 10 Phone o IOS.

Desde el punto de vista del procesado de la señal EEG, se podrían implementar técnicas de procesado diferentes para determinar cuáles son las adecuadas y ver si se puede mejorar tanto la precisión como la velocidad en la selección de la aplicación. También sería interesante implementar un sistema de umbralización para determinar si el usuario está atendiendo a los estímulos o está mirando el móvil, de manera que no se seleccione ningún comando.

Desde el punto de vista de la evaluación de la aplicación, debe realizarse un estudio mayor, con más sujetos y pertenecientes a la población objetivo de la aplicación, que son las personas con grave discapacidad.

REFERENCIAS

[1] Joseph D. Bronzino, *The Biomedical Engineering Handbook. Medical Devices and Systems*, vol. 53, no. 9, 2013.

[2] C. Alberola López, *Probabilidad, variables aleatorias y procesos estocásticos: una introducción orientada a las Telecomunicaciones*, 2004.

[3] L. Sörnmo, P. Laguna, *Bioelectrical Signal Processing in Cardiac and Neurological Applications*, 1st ed. Sweden: Academic Press, pp. 1-24, June 2005.

[4] S. M. Landau, "Comparing rates of change in cognitive measurements and regional glucose metabolism in Alzheimer's disease and mild cognitive impairment: Data from the Alzheimer's disease neuroimaging initiative," *Neurology*, vol. 70, no. 11, Philadelphia: Lippincott Williams & Wilkins 2008.

[5] H. H. Jasper, "The ten-twenty electrode system of the International Federation," *Electroencephalogr. Clin. Neurophysiol.*, vol. 10, no. 2, pp. 371–375, 1958.

[6] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, "Brain Computer Interfaces for communication and control," *Front. Neurosci.*, vol. 4, no. 113, pp. 767–791, 2002.

[7] A. Fallis, "Brain-Computer Interfaces," *Clim. Chang. 2013 - Phys. Sci. Basis*, vol. 53, no. 9, pp. 1–30, 2010.

[8] J. J. Vidal, "Real-time detection of brain events in EEG," *Proc. IEEE*, vol. 65, no. 5, pp. 633–641, 1977.

[9] D. J. McFarland, C. W. Anderson, K.R. Müller, A. Schlögl, D. J. Krusienski, "BCI Meeting 2005–Workshop on BCI Signal Processing: Feature Extraction and Translation", *IEEE Trans. Rehab. Eng.*, *vol. 14*, no. 2, pp. 135–138, 2006.

[10] V. Jurcak, D. Tsuzuki, and I. Dan, "10/20, 10/10, and 10/5 systems revisited: Their validity as relative head-surface-based positioning systems," *Neuroimage*, vol. 34, no. 4, pp. 1600–1611, 2007.

[11] O. Friman, I. Volosyak, and A. Gräser, "Multiple Channel Detection of Steady-State Visual Evoked Potentials for Brain-Computer Interfaces.Pdf (Application/Pdf Object)," vol. 54, no. 4, pp. 742–750, 2007.

[12] G. E. Fabiani, D. J. McFarland, J. R. Wolpaw, and G. Pfurtscheller, "Conversion of EEG activity into cursor movement by a brain-computer interface (BCI)," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 12, no. 3, pp. 331–338, 2004.

[13] N. Birbaumer, "Slow cortical potentials: their origin, meaning, and clinical use," in *Brain and behavior past, present, and future*, 1st ed., he Netherlands: Tilburg University Press, pp. 25–39, 1997.

[14] T. Hinterberger, N. Weiskopf, R. Veit, B. Wilhelm, E. Betta, and N. Birbaumer, "An EEG-driven brain-computer interface combined with functional magnetic resonance imaging (fMRI)," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 6, pp. 971–974, 2004.

[15] G. Pfurtscheller and C. Neuper, “Motor imagery and direct brain- computer communication,” *Proc. IEEE*, vol. 89, no. 7, pp. 1123–1134, 2001.

[16] J. Hu, J. Si, B. P. Olson, and J. He, “Principle component feature detector for motor cortical control.,” *Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. IEEE Eng. Med. Biol. Soc. Conf.*, vol. 6, pp. 4021–4, 2004.

[17] J. Mellinger, *A Practical Guide To Brain-Computer Interface with BCI2000*. Springer London, 2010.

[18] F. L.A and E. Donchin, “Talking of the top of your head: toward a mental prosthesis utilizing event-related brain potentials.”, 1988.

[19] K.-R. Müller, C. W. Anderson, and G. E. Birch, “Linear and nonlinear methods for brain-computer interfaces.,” *IEEE Trans. neural Syst. Rehabil. Eng.*, vol. 11, no. 2, pp. 165–169, 2003.

[20] M. Thulasidas, C. Guan, and J. Wu, “Robust classification of EEG signal for brain-computer interface.,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 14, no. 1, pp. 24–29, 2006.

[21] M. Bensch, A. A. Karim, J. Mellinger, T. Hinterberger, M. Tangermann, M. Bogdan, W. Rosenstiel, and N. Birbaumer, “Nessi: An EEG-controlled web browser for severely paralyzed patients,” *Comput. Intell. Neurosci.*, 2007.

[22] H. Zhao, H. Wang, C.-S. Li, and Y.-G. Li, “Brain-computer interface design based on slow cortical potentials using matlab/simulink,” *2009 Int. Conf. Mechatronics Autom.*, pp. 1044–1048, 2009.

[23] J. A. Pineda, D. S. Silverman, A. Vankov, and J. Hestenes, “Learning to control brain rhythms: making a brain computer interface possible,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 11, no. 2, pp. 181–184, 2003.

[24] Y. Te Wang, Y. Wang, and T. P. Jung, “A cell-phone based brain-computer interface for communication in daily life,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6320, no. 2, pp. 233–240, 2010.

[25] Y. Z. - and J. Z. -, “A Novel Platform of Brain Computer Interface Based on Android,” *J. Converg. Inf. Technol.*, vol. 8, no. 4, pp. 108–116, 2013.

[26] S. Vernon and S. S. Joshi, “Brain-muscle-computer interface: Mobile-phone prototype development and testing,” *IEEE Trans. Inf. Technol. Biomed.*, vol. 15, no. 4, pp. 531–538, 2011.

[27] A. S. Elsawy and S. Eldawlatly, “P300-based Applications for Interacting with Smart Mobile Devices,” pp. 22–24, 2015.

[28] S. J. Luck and E. S. Kappenman, *The Oxford Handbook of Event-Related Potential Components*. 2012.

[29] T. W. Picton, “The P300 wave of the human event-related potential.,” *Journal of clinical neurophysiology : official publication of the American Electroencephalographic Society*, vol. 9, no. 4. pp. 456–479, 1992.

[30] R. M. Rangayyan, “Filtering for Removal Artifacts,” in *Biomedical Signal Analysis: A Case-Study Approach*, IEEE Press., pp. 73–176, 2002.

[31] N. K. Squires, K. C. Squires, and S. A. Hillyard, “Two varieties of long-latency positive waves evoked by unpredictable auditory stimuli in man,” *Electroencephalogr. Clin. Neurophysiol.*, vol. 38, no. 4, pp. 387–401, 1975.

[32] R. M. Stelmack, M. Houlihan, and P. A. McGarry-Roberts, “Personality, reaction time, and event-related potentials,” *J. Pers. Soc. Psychol.*, vol. vol. 65 no, p. 399, 1993.

[33] C. Doucet and R. M. Stelmack, “An event-related potential analysis of extraversion and individual differences in cognitive processing speed and response execution.,” *J. Pers. Soc. Psychol.*, vol. 78, no. 5, pp. 956–64, 2000.

[34] J. Polich and C. J. Ochoa, “Alcoholism risk, tobacco smoking, and P300 event-related potential,” *Clin. Neurophysiol.*, vol. 115, no. 6, pp. 1374–1383, 2004.

[35] D. J. McFarland, L. M. McCane, S. V. David, and J. R. Wolpaw, “Spatial filter selection for EEG-based communication,” *Electroencephalogr. Clin. Neurophysiol.*, vol. 103, no. 3, pp. 386–394, 1997.

[36] K. Li, R. Sankar, Y. Arbel, and E. Donchin, “Single trial independent component analysis for P300 BCI system,” *Proc. 31st Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. Eng. Futur. Biomed. EMBC 2009*, pp. 4035–4038, 2009.

[37] R. C. T. Lee, Y. H. Chin, and S. C. Chang, “Application Of Principal Component Analysis To Multikey Searching,” *Tse*, vol. 2, no. 3, pp. 185–193, 1976.

[38] E. Donchin, K. M. Spencer, and R. Wijesinghe, “The mental prosthesis: Assessing the speed of a P300-based brain- computer interface,” *IEEE Trans. Rehabil. Eng.*, vol. 8, no. 2, pp. 174–179, 2000.

[39] D. J. Krusienski, E. W. Sellers, F. Cabestaing, S. Bayoudh, D. J. McFarland, T. M. Vaughan, and J. R. Wolpaw, “A comparison of classification techniques for the P300 Speller.,” *J. Neural Eng.*, vol. 3, no. 4, pp. 299–305, 2006.

[40] IDC, “Smartphone OS Market Share, 2015 Q2.” [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp. [Accessed: 08-Jul-2016].

[41] Google, “TOP aplicaciones descargadas.” [Online]. Available: https://play.google.com/store/apps/collection/topselling_free. [Accessed: 08-Jul-2016].

[42] Aol., BBDO, and InsightsNOW!, “Seven Shades of Mobile: The Hidden Motivations of Mobile Users,” p. 10, 2012.

[43] Medical engineering G.tec, “Advanced Biosignal Acquisition, Processing and Analysis Products 2013-2014,” pp. 1–96, 2013.

[44] Google, “Bluetooth API.” [Online]. Available: https://developer.android.com/guide/topics/connectivity/bluetooth.html. [Accessed: 08-Jul-2016].

ANEXO A

ÍNDICE DE ACRÓNIMOS

A continuación, se recogen todos los acrónimos empleados a lo largo de este trabajo de fin de grado por orden alfabético, acompañados de su descripción tanto en inglés como en español.

ALS *Amyotrophic Lateral Sclerosis*
Esclerosis Lateral Amiotrófica

ANN *Artificial Neural Networks*
Redes Neuronales Artificiales

AR *Autoregressive Models*
Modelos Autorregresivos

AUC *Area Under Curve*
Área Bajo la Curva

BCI *Brain Computer Interface*
Interfaz Cerebro-Computadora

CAR *Common Average Reference*
Referencia de Media Común

CSP *Common Spatial Patterns*
Patrones Espaciales Comunes

CWT *Continous Wavelet Transform*
Transformada Wavelet Continua

DWT *Discrete Wavelet Transform*
Transformada Wavelet Discreta

EEG *Electroencefalography*
Electroencefalografía

ECG *Electrocardiography*
Electrocardiografía

ECoG *Electrocorticography*
Electrocorticografía

EMG *Electromyography*
Electromiografía

EOG *Electrooculography*
Electrooculografía

ERD *Event-Related Desynchronization*
Desincronización del evento relacionado

ERP *Event-Related Potential*
Potencial relacionado a un evento

ERS *Event-Related Synchronization*
Sincronización del evento relacionado

FLD *Fisher's Linear Discriminant*
Discriminante Lineal de Fisher

fMRI *Functional Magnetic Resonance Imaging*
Imagen por Resonancia Magnética Funcional

HMM *Hidden Markov Model*
Modelo Oculto de Markov

ICA *Independent Component Analysis*
Análisis de Componentes Independientes

LDA *Linear Discriminant Analysis*
Análisis Discriminante Lineal

LMS *Least Mean Square*
Mínimos Cuadrados Promediados

LVQ *Linear-Vector Quantization*
Aprendizaje por Cuantificación Vectorial

MEG *Magnetoencephalography*
Magnetoencefalografía

MSE *Mean Square Error*
Error Cuadrático Medio

mVEP *Motion-onset Visual Evoked Potentials*
Potenciales Evocados Visuales de Movimiento

NPV *Negative Predictive Value*
Valor Predictivo Negativo

PCA *Principal Component Analysis*
Análisis de Componentes Principales

PCM *Pearson's Correlation Method*
Método de Correlación de Pearson

PET *Positron Emission Tomography*
Tomografía por Emisión de Positrones

PLS *Partial Least Squares*
Mínimos Cuadrados Parciales

PPV *Positive Predictive Value*
Valor Predictivo Positivo

QDA *Quadratric Discriminant Analysis*
Análisis Discriminante Cuadrático

RLS *Recursive Least Squares*
Mínimos Cuadrados Recursivos

RT *Reaction Time*
Tiempo de Reacción

SCP *Slow Cortical Potentials*
Potenciales Corticales Lentos

STFT *Short-Time Fourier Transform*
Transformada de Fourier de Tiempo Corto

SWLD A *StepWise Linear Discriminant Analysis*
Análisis Discriminante Lineal Paso-a-paso

SVM *Support Vector Machines*
Máquinas de Vectores de Soporte

TTD *Thought Translation Device*
Dispositivo de Interpretación del Pensamiento

VEP *Visual Evoked Potentials*
Potenciales Evocados Visuales

WPT *Wavelet Packet Transform*
Transformada de Paquetes Wavelet

ANEXO B

CÓDIGO GENERADO EN C++ Y ANDROID

B.1. MODIFICACIONES EN P3SPELLER

Solo se introduce el código original generado para la realización de la aplicación y no los archivos completos de KeystrokeFilter.cpp y KeystrokeFilter.h

```
/*--------------------------------------------------------------------*/
/*----------------------BLUETOOTH CLIENT------------------------------*/
/*--------------------------------------------------------------------*/
// Cliente BLuetooth para el envío de cadenas de texto aldispositivo//
// con nombre Bluetooth especificado en la variable g szRemoteNam   //
// Versión: 1.5                                                     //
// Fecha: 27/04/20146                                               // //
                                                          //
//                                                                  //
//                                Universidad de Valladolid(Spain)//
//                                Grupo de Ingeniería Biomédica   //
//                                Eduardo SantamariaVazquez       //
//__________________________________________________________________/


/*********************************************************************
Parametros de FindingDevices
          *
*********************************************************************/
#define MAX_NAME 248
typedef ULONGLONG bt_addr, *pbt_addr, BT_ADDR, *PBT_ADDR;

union {
        // Storage for returned struct
        CHAR buf5000];
        // properly align buffer to BT_ADDR requirements
        SOCKADDR_BTH _Unused_;
} butuh;

/*****************************
Parametros de RunClient      *
*****************************/
// UUID del programa destino {B62C4E8D-62CC-404b-BBBF-BF3E3BBB1374}
DEFINE_GUID(g_guidServiceClass, 0xb62c4e8d, 0x62cc, 0x404b, 0xbb, 0xbf, 0xbf, 0x3e,
0x3b, 0xbb, 0x13, 0x74);

#define CXN SUCCESS                         0
#define CXN ERROR                           1
const char *g_szRemoteName = ("AppRunner");  // 1 extra for trailing NULL character
int MaxCxnCycles=1;
/*****************************
Parametros de nameToBthAddr  *
*****************************/
#define CXN MAX INQUIRY RETRY               3
#define CXN_DELAY_NEXT_INQUIRY              15


/************************************************
Funcion principal del programa. Recibe una cadena *
que manda al dispositivo especificado             *
************************************************/
void send_through_bluetooth(char *command){

    WSADATA wsd;
        SOCKADDR_BTH RemoteBthAddr = {0};
        ULONG ulRetCode = CXN_SUCCESS;
        char *CXN_TEST_DATA_STRING = command;
        SIZE_T CXN_TRANSFER_DATA_LENGTH = sizeof(CXN_TEST_DATA_STRING) *
strlen(CXN_TEST_DATA_STRING);
        if(WSAStartup(MAKEWORD(2,2), &wsd) != 0) bciout << "WSAStartup() failed with
error code" << WSAGetLastError();
```

```
        ulRetCode = NameToBthAddr(g_szRemoteName, &RemoteBthAddr);

        if ( CXN_SUCCESS != ulRetCode ) {
                bciout << "-FATAL- | Unable to get address of the remote radio having
name %s\n" << g_szRemoteName;
    }

    if ( CXN_SUCCESS == ulRetCode) {
            ulRetCode = runClient(RemoteBthAddr, MaxCxnCycles, CXN_TEST_DATA_STRING,
CXN_TRANSFER_DATA_LENGTH);
    }

        if (CXN_SUCCESS == ulRetCode) bciout << "Comando transmitido via Bluetooth a
AppRunner_server";
        if(WSACleanup() != 0) bciout << "WSACleanup() failed with error code" <<
WSAGetLastError();
}

/*********************************************************
Funcion runClient. Esta funcion pone en marcha el cliente  *
*********************************************************/

ULONG runClient (_In_ SOCKADDR_BTH RemoteAddr, _In_ int iMaxCxnCycles, _In_ char*
CXN_TEST_DATA_STRING, _In_ SIZE_T CXN_TRANSFER_DATA_LENGTH){
    ULONG           ulRetCode = CXN_SUCCESS;
    int             iCxnCount = 0;
        char                *pszData = NULL;
    SOCKET          LocalSocket = INVALID_SOCKET;
    SOCKADDR_BTH    SockAddrBthServer = RemoteAddr;
    HRESULT         res;

        pszData = (char *) HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY,
CXN_TRANSFER_DATA_LENGTH);
        //pszData = (LPWSTR ) HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY,
CXN TRANSFER DATA LENGTH);
    if ( NULL == pszData ) {
        ulRetCode = STATUS_NO_MEMORY;
        bciout << "=CRITICAL= | HeapAlloc failed | out of memory, gle = " <<
GetLastError();
    }

    if ( CXN_SUCCESS == ulRetCode ) {
        //
        // Setting address family to AF_BTH indicates winsock2 to use Bluetooth sockets
        // Port should be set to 0 if ServiceClassId is spesified.
        //
        SockAddrBthServer.addressFamily = AF_BTH;
        SockAddrBthServer.serviceClassId = g_guidServiceClass;
        SockAddrBthServer.port = 0;

        //
        // Create a static data-string, which will be transferred to the remote
        // Bluetooth device
        //
              res = StringCbCopyN(pszData, CXN_TRANSFER_DATA_LENGTH,
CXN_TEST_DATA_STRING, CXN_TRANSFER_DATA_LENGTH);
        if ( FAILED(res) ) {
            bciout << "=CRITICAL= | Creating a static data string failed";
            ulRetCode = CXN_ERROR;
        }

    }

    if ( CXN_SUCCESS == ulRetCode ) {

        pszData[(CXN_TRANSFER_DATA_LENGTH/sizeof(wchar_t)) - 1] = 0;

        //
        // Run the connection/data-transfer for user specified number of cycles
        //
        for ( iCxnCount = 0; (0 == ulRetCode) && (iCxnCount < iMaxCxnCycles ||
iMaxCxnCycles == 0);iCxnCount++ ) {
            //
            // Open a bluetooth socket using RFCOMM protocol
            //
            LocalSocket = socket(AF_BTH, SOCK_STREAM, BTHPROTO_RFCOMM);
```

```
            if ( INVALID_SOCKET == LocalSocket ) {
                bciout << "=CRITICAL= | socket() call failed. WSAGetLastError = " <<
WSAGetLastError();
                ulRetCode = CXN_ERROR;
                break;
            }

            //
            // Connect the socket (pSocket) to a given remote socket represented by
address (pServerAddr)
            //
            if ( SOCKET_ERROR == connect(LocalSocket, (struct sockaddr *)
&SockAddrBthServer, sizeof(SOCKADDR_BTH)) ) {
                bciout << "=CRITICAL= | connect() call failed. Target Device is not
available. WSAGetLastError= " << WSAGetLastError();
                ulRetCode = CXN_ERROR;
                break;
            }

            //
            // send() call indicates winsock2 to send the given data
            // of a specified length over a given connection.
            //
            if ( SOCKET_ERROR == send(LocalSocket, pszData,
(int)CXN_TRANSFER_DATA_LENGTH, 0) ) {
                bciout << "=CRITICAL= | send() call failed w/socket = [0x%I64X], szData
= [%p], dataLen = [%I64u]. WSAGetLastError=[%d]\n", (ULONG64)LocalSocket, pszData,
(ULONG64)CXN_TRANSFER_DATA_LENGTH, WSAGetLastError();
                ulRetCode = CXN_ERROR;
                break;
            }

            //
            // Close the socket
            //
            if ( SOCKET_ERROR == closesocket(LocalSocket) ) {
                bciout << "=CRITICAL= | closesocket() call failed w/socket = [0x%I64X].
WSAGetLastError=[%d]\n", (ULONG64)LocalSocket, WSAGetLastError();
                ulRetCode = CXN_ERROR;
                break;
            }

            LocalSocket = INVALID_SOCKET;

        }
    }

    if ( INVALID_SOCKET != LocalSocket ) {
        closesocket(LocalSocket);
        LocalSocket = INVALID_SOCKET;
    }

    if ( NULL != pszData ) {
        HeapFree(GetProcessHeap(), 0, pszData);
        pszData = NULL;
    }
       return(ulRetCode);
}

//
// nameToBthAddr converts a bluetooth device name to a bluetooth address,
// if required by performing inquiry with remote name requests.
// This function demonstrates device inquiry, with optional LUP flags.
//
ULONG NameToBthAddr(_In_ const char *pszRemoteName, _Out_ PSOCKADDR_BTH pRemoteBtAddr)
{
    INT             iResult = CXN_SUCCESS;
    BOOL            bContinueLookup = FALSE, bRemoteDeviceFound = FALSE;
    ULONG           ulFlags = 0, ulPQSSize = sizeof(WSAQUERYSET);
    HANDLE          hLookup = NULL;
    PWSAQUERYSET    pWSAQuerySet = NULL;

    ZeroMemory(pRemoteBtAddr, sizeof(*pRemoteBtAddr));

    pWSAQuerySet = (PWSAQUERYSET) HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY,
ulPQSSize);
    if ( NULL == pWSAQuerySet ) {
```

```
        iResult = STATUS_NO_MEMORY;
        bciout << "!ERROR! | Unable to allocate memory for WSAQUERYSET\n";
    }

    //
    // Search for the device with the correct name
    //
    if ( CXN_SUCCESS == iResult) {

        for ( INT iRetryCount = 0; !bRemoteDeviceFound && (iRetryCount <
CXN_MAX_INQUIRY_RETRY); iRetryCount++ ) {
            //
            // WSALookupService is used for both service search and device inquiry
            // LUP_CONTAINERS is the flag which signals that we're doing a device
inquiry.
            //
            ulFlags = LUP_CONTAINERS;

            //
            // Friendly device name (if available) will be returned in
lpszServiceInstanceName
            //
            ulFlags |= LUP_RETURN_NAME;

            //
            // BTH_ADDR will be returned in lpcsaBuffer member of WSAQUERYSET
            //
            ulFlags |= LUP_RETURN_ADDR;

            if ( 0 == iRetryCount ) {

            } else {
                //
                // Flush the device cache for all inquiries, except for the first
inquiry
                //
                // By setting LUP FLUSHCACHE flag, we're asking the lookup service to do
                // a fresh lookup instead of pulling the information from device cache.
                //
                ulFlags |= LUP_FLUSHCACHE;

                //
                // Pause for some time before all the inquiries after the first inquiry
                //
                // Remote Name requests will arrive after device inquiry has
                // completed.  Without a window to receive IN_RANGE notifications,
                // we don't have a direct mechanism to determine when remote
                // name requests have completed.
                //
                bciout << "*INFO* | Unable to find device.  Waiting for %d seconds
before re-inquiry...\n", CXN_DELAY_NEXT_INQUIRY;
                Sleep(CXN_DELAY_NEXT_INQUIRY * 1000);

                bciout << "*INFO* | Inquiring device ...\n";
            }

            //
            // Start the lookup service
            //
            iResult = CXN_SUCCESS;
            hLookup = 0;
            bContinueLookup = FALSE;
            ZeroMemory(pWSAQuerySet, ulPQSSize);
            pWSAQuerySet->dwNameSpace = NS_BTH;
            pWSAQuerySet->dwSize = sizeof(WSAQUERYSET);
            iResult = WSALookupServiceBegin(pWSAQuerySet, ulFlags, &hLookup);

            //
            // Even if we have an error, we want to continue until we
            // reach the CXN_MAX_INQUIRY_RETRY
            //
            if ( (NO_ERROR == iResult) && (NULL != hLookup) ) {
                bContinueLookup = TRUE;
            } else if ( 0 < iRetryCount ) {
                bciout << "=CRITICAL= | WSALookupServiceBegin() failed with error code
%d, WSAGetLastError = %d\n", iResult, WSAGetLastError();
                break;
```

```
            }

            while ( bContinueLookup ) {
                //
                // Get information about next bluetooth device
                //
                // Note you may pass the same WSAQUERYSET from LookupBegin
                // as long as you don't need to modify any of the pointer
                // members of the structure, etc.
                //
                // ZeroMemory(pWSAQuerySet, ulPQSSize);
                // pWSAQuerySet->dwNameSpace = NS_BTH;
                // pWSAQuerySet->dwSize = sizeof(WSAQUERYSET);
                if ( NO_ERROR == WSALookupServiceNext(hLookup, ulFlags, &ulPQSSize,
pWSAQuerySet) ) {

                    //
                    // Compare the name to see if this is the device we are looking for.
                    //

                    if ( ( pWSAQuerySet->lpszServiceInstanceName != NULL ) &&
                                                ( CXN_SUCCESS == strcmp(pWSAQuerySet-
>lpszServiceInstanceName, pszRemoteName) ) ) {
                        //
                        // Found a remote bluetooth device with matching name.
                        // Get the address of the device and exit the lookup.
                        //
                        CopyMemory(pRemoteBtAddr, (PSOCKADDR_BTH) pWSAQuerySet-
>lpcsaBuffer->RemoteAddr.lpSockaddr, sizeof(*pRemoteBtAddr));
                        bRemoteDeviceFound = TRUE;
                        bContinueLookup = FALSE;
                    }
                } else {
                    iResult = WSAGetLastError();
                    if ( WSA_E_NO_MORE == iResult ) { //No more data
                        //
                        // No more devices found.  Exit the lookup.
                        //
                        bContinueLookup = FALSE;
                    } else if ( WSAEFAULT == iResult ) {
                        //
                        // The buffer for QUERYSET was insufficient.
                        // In such case 3rd parameter "ulPQSSize" of function
"WSALookupServiceNext()" receives
                        // the required size.  So we can use this parameter to
reallocate memory for QUERYSET.
                        //
                        HeapFree(GetProcessHeap(), 0, pWSAQuerySet);
                        pWSAQuerySet = (PWSAQUERYSET) HeapAlloc(GetProcessHeap(),
HEAP_ZERO_MEMORY, ulPQSSize);
                        if ( NULL == pWSAQuerySet ) {
                            bciout << "!ERROR! | Unable to allocate memory for
WSAQERYSET\n";
                            iResult = STATUS_NO_MEMORY;
                            bContinueLookup = FALSE;
                        }
                    } else {
                        bciout << "=CRITICAL= | WSALookupServiceNext() failed with error
code %d\n", iResult;
                        bContinueLookup = FALSE;
                    }
                }
            }

            //
            // End the lookup service
            //
            WSALookupServiceEnd(hLookup);

            if ( STATUS_NO_MEMORY == iResult ) {
                break;
            }
        }
    }

    if ( NULL != pWSAQuerySet ) {
        HeapFree(GetProcessHeap(), 0, pWSAQuerySet);
```

```
        pWSAQuerySet = NULL;
    }

    if ( bRemoteDeviceFound ) {
        iResult = CXN_SUCCESS;
    } else {
        iResult = CXN_ERROR;
    }

    return iResult;
}
```

```
/*-------------------------------------------------------------------------*/
/*----------------------KeystrokeFilter::process---------------------------*/
/*-------------------------------------------------------------------------*/

void KeystrokeFilter::Process( const GenericSignal& Input, GenericSignal& Output ) {
       //En funcion de la seleccion en una matriz u otra el valor del comando cambiara
       char *comando = "";
       //Switch de la primera matriz
       if(mCurrentMatrix == 1) {
              switch (State("SelectedRow")){
              case 1:
                     switch (State("SelectedColumn")){
                            case 1:
                                   comando = ("call");
                                   mCurrentMatrix = 2;
                                   break;
                            case 2:
                                   comando = ("sms");
                                   mCurrentMatrix = 2;
                                   break;
                            case 3:
                                   comando = ("calendar");
                                   mCurrentMatrix = 2;
                                   break;
                            case 4:
                                   comando = ("whatsapp");
                                   mCurrentMatrix = 2;
                                   break;
                     }
                     break;
              case 2:
                     switch (State("SelectedColumn")){
                            case 1:
                                   comando = ("viewContact");
                                   mCurrentMatrix = 2;
                                   break;
                            case 2:
                                   comando = ("deleteContact");
                                   mCurrentMatrix = 2;
                                   break;
                            case 3:
                                   comando = ("newContact");
                                   mCurrentMatrix = 2;
                                   break;
                            case 4:
                                   comando = ("setAlarm");
                                   mCurrentMatrix = 2;
                                   break;
                     }
                     break;
              case 3:
                     switch (State("SelectedColumn")){
                            case 1: comando = ("imageView");
                                   break;
                            case 2: comando = ("next_image");
                                   break;
                            case 3: comando = ("prev_image");
                                   break;
                            case 4: comando = ("rotate_image");
```

```
                                        break;
                                }
                                break;
                        case 4:
                                switch (State("SelectedColumn")){
                                        case 1: comando = ("camera");
                                                break;
                                        case 2: comando = ("take_picture");
                                                break;
                                        case 3: comando = ("sleep");
                                                break;
                                        case 4: comando = ("back");
                                                break;
                                }
                                break;
                        }
                } else {
                        switch (State("SelectedRow")){
                        case 1:
                                switch (State("SelectedColumn")){
                                        case 1:
                                                if (mayus == 1) comando = "A";
                                                else comando = ("a");
                                                break;
                                        case 2:
                                                if (mayus == 1) comando = "B";
                                                else comando = ("b");
                                                break;
                                        case 3:
                                                if (mayus == 1) comando = "C";
                                                else comando = ("c");
                                                break;
                                        case 4:
                                                if (mayus == 1) comando = "D";
                                                else comando = ("d");
                                                break;
                                        case 5:
                                                if (mayus == 1) comando = "E";
                                                else comando = ("e");
                                                break;
                                        case 6:
                                                if (mayus == 1) comando = "F";
                                                else comando = ("f");
                                                break;
                                }
                                break;
                        case 2:
                                switch (State("SelectedColumn")){
                                        case 1:
                                                if (mayus == 1) comando = "G";
                                                else comando = ("g");
                                                break;
                                        case 2:
                                                if (mayus == 1) comando = "H";
                                                else comando = ("h");
                                                break;
                                        case 3:
                                                if (mayus == 1) comando = "I";
                                                else comando = ("i");
                                                break;
                                        case 4:
                                                if (mayus == 1) comando = "J";
                                                else comando = ("j");
                                                break;
                                        case 5:
                                                if (mayus == 1) comando = "K";
                                                else comando = ("k");
                                                break;
                                        case 6:
                                                if (mayus == 1) comando = "L";
                                                else comando = ("l");
                                                break;
                                }
                                break;
                        case 3:
                                switch (State("SelectedColumn")){
                                        case 1:
```

```
                        if (mayus == 1) comando = "M";
                        else comando = ("m");
                        break;
                case 2:
                        if (mayus == 1) comando = "N";
                        else comando = ("n");
                        break;
                case 3:
                        if (mayus == 1) comando = "Ñ";
                        else comando = ("ñ");
                        break;
                case 4:
                        if (mayus == 1) comando = "O";
                        else comando = ("o");
                        break;
                case 5:
                        if (mayus == 1) comando = "P";
                        else comando = ("p");
                        break;
                case 6:
                        if (mayus == 1) comando = "Q";
                        else comando = ("q");
                        break;
        }
        break;
case 4:
        switch (State("SelectedColumn")){
                case 1:
                        if (mayus == 1) comando = "R";
                        else comando = ("r");
                        break;
                case 2:
                        if (mayus == 1) comando = "S";
                        else comando = ("s");
                        break;
                case 3:
                        if (mayus == 1) comando = "T";
                        else comando = ("t");
                        break;
                case 4:
                        if (mayus == 1) comando = "U";
                        else comando = ("u");
                        break;
                case 5:
                        if (mayus == 1) comando = "V";
                        else comando = ("v");
                        break;
                case 6:
                        if (mayus == 1) comando = "W";
                        else comando = ("w");
                        break;
        }
        break;
        case 5:
                switch (State("SelectedColumn")){
                        case 1:
                                if (mayus == 1) comando = "X";
                                else comando = ("x");
                                break;
                        case 2:
                                if (mayus == 1) comando = "Y";
                                else comando = ("y");
                                break;
                        case 3:
                                if (mayus == 1) comando = "Z";
                                else comando = ("z");
                                break;
                        case 4:
                                comando = "0";
                                break;
                        case 5:
                                comando = "1";
                                break;
                        case 6:
                                comando = "2";
                                break;
                }
```

```
break;
case 6:
        switch (State("SelectedColumn")){
                case 1:
                        comando = "3";
                        break;
                case 2:
                        comando = "4";
                        break;
                case 3:
                        comando = "5";
                        break;
                case 4:
                        comando = "6";
                        break;
                case 5:
                        comando = "7";
                        break;
                case 6:
                        comando = "8";
                        break;
        }
break;
case 7:
        switch (State("SelectedColumn")){
                case 1:
                        comando = "9";
                        break;
                case 2:
                        comando = ",";
                        break;
                case 3:
                        comando = ".";
                        break;
                case 4:
                        comando = "@";
                        break;
                case 5:
                        comando = "-";
                        break;
                case 6:
                        comando = ":";
                        break;
        }
break;
case 8:
        switch (State("SelectedColumn")){
                case 1:
                        comando = "/";
                        break;
                case 2:
                        comando = "¿";
                        break;
                case 3:
                        comando = "?";
                        break;
                case 4:
                        comando = "(";
                        break;
                case 5:
                        comando = ")";
                        break;
                case 6:
                        if (mayus==0) {
                                mayus = 1;
                        }else{
                                mayus = 0;
                        }
                        break;
        }
break;
case 9:
        switch (State("SelectedColumn")){
                case 1:
                        comando = " ";
                        break;
                case 2:
```

```
                                        comando = "del";
                                        break;
                                case 3:
                                        comando = "clear";
                                        break;
                                case 4:
                                        comando = "prev_edit";
                                        break;
                                case 5:
                                        comando = "enter";
                                        break;
                                case 6:
                                        comando = "back";
                                        mCurrentMatrix = 1;
                                        break;
                                }
                        break;
                }
        }

        //Evitamos que el envio del comando se repita, ya que el programa pasa por aqui
muchas veces
        //en cada seleccion. Tampoco enviamos comandos vacios.
        if ((strcmp(comando,prevCommand) != 0) && (strlen(comando)!=0)){
                send_through_bluetooth(comando);
        }
        prevCommand = comando;
        State("SelectedRow")=0;
        State("SelectedColumn")=0;
        SendKeystroke( mKeystrokeExpression.Evaluate( &Input ) );
        Output = Input;
}
```

B.2. APLICACIÓN ANDROID APPRUNNER

En esta sección se adjunta el código de los ficheros mas importantes generado para la aplicación AppRunner.

```
/*********************************************************************************
************ * Aplicación creada por Eduardo Santamaria Vazquez *************
*
Funciones importantes para añadir funcionalidades a la aplicacion:
*  -inciarActividad() -> Es un switch donde se realiza una accion en función
                         del comando recibido. Se debe meter un case para el
                         comando que se quiera añadir y realizar la accion
                         necesaria dentro.
*  -onActivityRsult() -> Cuando acaba una actividad iniciada con el método
                            startActivityForResult se llama a esta funcion donde
                            se reciben los resultados. Se devuelve el valor del
                            REQUEST CODE con el que se inicio la actividad, para poder
                            realizar una accion distinta dependiendo de la
                            actividad que haya terminado En esta funcion se recogen los
                            datos metidos por el usuario en la actividad FormActivity.
*********************************************************************************/

package com.bci.eduardo.apprunnerthreads;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothServerSocket;
import android.bluetooth.BluetoothSocket;
import android.content.ContentProviderOperation;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.media.AudioManager;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.os.Message;
import android.provider.AlarmClock;
import android.provider.CalendarContract;
import android.provider.ContactsContract;
```

```
import android.support.v4.content.LocalBroadcastManager;
import android.support.v7.app.AppCompatActivity;
import android.telephony.PhoneStateListener;
import android.telephony.SmsManager;
import android.telephony.TelephonyManager;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import com.bci.eduardo.apprunner.R;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.UUID;
import java.lang.String;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class MainActivity extends AppCompatActivity {

    //Nombre y UUID para identificar el puerto Bluetooth que utiliza la aplicacion
    private static String NAME = "AppRunner";
    private static UUID MY_UUID = UUID.fromString("B62C4E8D-62CC-404b-BBBF-
BF3E3BBB1374");

    //Variables globales de la aplicacion
    private static TextView cmd; //Caja de texto donde se va actualizando la niformacion
de la aplicacion
    private static TelephonyManager manager; //Se utiliza para detectar cambios en la
aplicacion de telefonia
    private static int CURRENT_ACTIVITY = 0; //Actividad actual de la aplicacion
    private static int STATIC_ACTIVITY = 0; //Actividad estatica (si esta a 1 no permite
selecciones en la matriz de navegacion
    private static int SLEEP_ACTIVITY = 0; //Flag que indica si el usuario ha
seleccionado el boton sleep o no;
    //Metodo startServer. Hay que tener en cuenta que el dispositivo ha de haber sido
vinculado con
    //anterioridad al dispositivo bluetooth. Esta aplicacion no requiere que el
dispositivo este
    //en modo discoverable
    private static BluetoothAdapter mBluetoothAdapter;
    private final static int BLUETOOTH_START = 1; //Codigo para la actividad Bluetooth

    //Variables para manejar las opciones de las actividades correspondientes
    //En esta version el maximo de opciones es 3. Si se necesitaran mas, declarar aqui
    private static int numOpt = 0;
    private static String option1, option2, option3, option4, option5;

    //Codigos de las actividades. Permitiran recibir resultados y cerrarlas cuando sea
preciso
    private static final int REQUEST_CAMERA = 100;
    private static final int REQUEST_SEND_WHATSAPP = 101;
    private static final int REQUEST_CALL = 102;
    private static final int REQUEST_IMGVIEW = 103;
    private static final int REQUEST_CALENDAR = 104;
    private static final int REQUEST_VIEWCONTACT = 105;
    private static final int REQUEST_SETALARM = 106;
    private static final int REQUEST_HELP = 106;

    //Codigos para el control del formulario de opciones
    private static final int REQUEST_FORM_CALL = 201;
    private static final int REQUEST_FORM_SMS = 202;
    private static final int REQUEST_FORM_WHATSAPP = 203;
    private static final int REQUEST_FORM_SETALARM = 204;
    private static final int REQUEST_FORM_VIEWCONTACT = 205;
    private static final int REQUEST_FORM_DELETECONTACT = 206;
    private static final int REQUEST_FORM_NEWCONTACT = 207;
    private static final int REQUEST_FORM_CALENDAR = 208;

    //Con el método onCreate cargamos el xml con la interfaz gráfica y realizamos las
    primeras
    //operaciones, como mostrar los dispositivos emparejados o iniciar el adaptador
    bluetooth
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
            //Iniciamos un servicio que nos permite detectar cambios en el estado del
        servicio de telefonia
        //Si cambia activamos el manos libres (Valido para llamadas entrantes o
        salientes mientras la
        //aplicacion este activa)
        StatePhoneReceiver statePhoneReceiver = new StatePhoneReceiver();
        manager = ((TelephonyManager) this.getSystemService(Context.TELEPHONY_SERVICE));
        manager.listen(statePhoneReceiver, PhoneStateListener.LISTEN_CALL_STATE);

        //Instanciamos el cuadro de texto donde veremos la linea de comandos
        cmd = (TextView) findViewById(R.id.list);

        //Instanciamos el adaptador bluetooth por defecto del dispositivo para poder
        manejarlo
        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

        //Comprobacion de que el dispositivo tiene un adaptador bluetooth por defecto
        if (mBluetoothAdapter == null) {
            cmd.append(">> No se encuentra adaptador bluetooth por defecto");
        }

        //Ponemos el dispositivo en modo visible para que se pueda iniciar una conexion
        con el servidor desde el exterior
        if (mBluetoothAdapter.getScanMode() != BluetoothAdapter.SCAN_MODE_CONNECTABLE) {
                Intent discoverableIntent = new
                Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(discoverableIntent, BLUETOOTH_START);
        }

        cmd.append(">> Aplicacion cargada\n");
    }

    protected void startServer(View view) {
        //Solo iniciamos el servidor si el Bluetooth esta encendido
        if (mBluetoothAdapter.getScanMode() != BluetoothAdapter.SCAN_MODE_CONNECTABLE) {
                Intent discoverableIntent = new
                Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                 startActivityForResult(discoverableIntent, BLUETOOTH_START);
        }
        //Se crea un hilo nuevo para recibir mensajes cuando pulsamos el boton
        //Creamos un Handler para que pueda haber comunicacion entre los dos procesos,
        //de manera que cuando se reciba un mensaje, el hilo servidor lo ponga en la
        //pila del handler y se recoja desde el hilo principal.
        AcceptThread acceptThread = new AcceptThread();
        acceptThread.start();
        cmd.append(">> Servidor a la espera\n");
        Button boton = (Button) findViewById(R.id.start);
        boton.setClickable(false);
    }

    //El handler nos permite obtener mensajes de los hilos que corren en pararelo
    //al programa principal. De esta manera no tenemos que tener el programa principal
    //bloqueado mientras se espera a recibir un mensaje.
    private Handler handler = new Handler() {;
        @Override
        public void handleMessage(Message msg) {
            String comando = (String) msg.obj;
            iniciarActividad(comando);
            //Una vez recibido el comando, iniciamos un nuevo servidor para seguir
escuchando.
            AcceptThread acceptThread = new AcceptThread();
            acceptThread.start();
        }
    };

    //Esta funcion se llama cuando se recibe un comando, aqui es donde podemos añadir
funciones a
    //la aplicacion. El comando se recibe en un string que es comando.
    public void iniciarActividad(String comando) {

        //Comprobamos que el comando no sea "sleep", en caso de serlo se conmuta el
estado SLEEP_ACTIVITY
        //Si esta a uno, no se ejecutara ningun comando de los recibidos ahsta que
vuelva a conmutar a cero
        if (comando.equals("sleep")) {
            cmd.append(">> Comando recibido: " + comando + "\n");
            if (SLEEP_ACTIVITY == 0){
```

```
                SLEEP_ACTIVITY = 1;
            }else {
                SLEEP_ACTIVITY = 0;
            }
        }
        //Si el programa no esta en pausa, ejecutamos la accion del comando recibido.
STATIC ACTIVITY
        //es un flag que indica que una actividad estatica esta activa. Las actividades
estaticas son
        //la camara y la galeria, ya que una vez abiertas, solo podemos realizar las
acciones pertinentes
        //de esas actividades y volver atras. No se hace caso a otras selecciones
posiblemente erroneas
        if(SLEEP_ACTIVITY == 0) {
            switch (comando) {
                case "camera":
                    if (STATIC_ACTIVITY == 0) {
                        Intent camera = new Intent(this, CameraActivity.class);
                        if (camera.resolveActivity(getPackageManager()) != null) {
                            startActivityForResult(camera, REQUEST_CAMERA);
                            CURRENT_ACTIVITY = REQUEST_CAMERA;
                            STATIC_ACTIVITY = 1;
                            cmd.append(">> Comando recibido: " + comando + "\n");
                        }
                    }
                    break;
                case "help":
                    if (STATIC_ACTIVITY == 0) {
                        Intent help = new Intent(this, HelpActivity.class);
                        if (help.resolveActivity(getPackageManager()) != null) {
                            startActivityForResult(help, REQUEST_HELP);
                            CURRENT_ACTIVITY = REQUEST_HELP;
                            STATIC_ACTIVITY = 1;
                            cmd.append(">> Comando recibido: " + comando + "\n");
                        }
                    }
                    break;
                case "take_picture":
                    Intent take_picture_intent = new Intent();
                    take_picture_intent.setAction("take_picture");

LocalBroadcastManager.getInstance(this).sendBroadcast(take_picture_intent);
                    cmd.append(">> Comando recibido: " + comando + "\n");
                    break;
                case "whatsapp":
                    if (STATIC_ACTIVITY == 0) {
                        numOpt = 1;
                        option1 = "(*)Texto a enviar";
                        Intent formWhatasppIntent = new Intent(this,
FormActivity.class);
                        formWhatasppIntent.putExtra("numOpt", numOpt);
                        formWhatasppIntent.putExtra("option1", option1);
                        startActivityForResult(formWhatasppIntent,
REQUEST_FORM_WHATSAPP);
                        CURRENT_ACTIVITY = REQUEST_FORM_WHATSAPP;
                        cmd.append(">> Comando recibido: " + comando + "\n");
                    }
                    break;
                case "call":
                    if (STATIC_ACTIVITY == 0) {
                        numOpt = 1;
                        option1 = "Numero de telefono o contacto";
                        Intent formCallIntent = new Intent(this, FormActivity.class);
                        formCallIntent.putExtra("numOpt", numOpt);
                        formCallIntent.putExtra("option1", option1);
                        startActivityForResult(formCallIntent, REQUEST_FORM_CALL);
                        CURRENT_ACTIVITY = REQUEST_FORM_CALL;
                        cmd.append(">> Comando recibido: " + comando + "\n");
                    }
                    break;
                case "sms":
                    if (STATIC_ACTIVITY == 0) {
                        numOpt = 2;
                        option1 = "(*)Numero de telefono o contacto";
                        option2 = "(*)Mensaje a enviar";
                        Intent formSMSIntent = new Intent(this, FormActivity.class);
                        formSMSIntent.putExtra("numOpt", numOpt);
```

```
                        formSMSIntent.putExtra("option1", option1);
                        formSMSIntent.putExtra("option2", option2);
                        startActivityForResult(formSMSIntent, REQUEST_FORM_SMS);
                        CURRENT_ACTIVITY = REQUEST_FORM_SMS;
                        cmd.append(">> Comando recibido: " + comando + "\n");
                    }
                    break;
                case "imageView":
                    if (STATIC_ACTIVITY == 0) {
                        Intent imageView = new Intent(this, ImageViewActivity.class);
                        if (imageView.resolveActivity(getPackageManager()) != null) {
                            startActivityForResult(imageView, REQUEST_IMGVIEW);
                            CURRENT_ACTIVITY = REQUEST_IMGVIEW;
                            STATIC_ACTIVITY = 1;
                            cmd.append(">> Comando recibido: " + comando + "\n");
                        }
                    }
                    break;
                case "next_image":
                    Intent next_picture_intent = new Intent();
                    next_picture_intent.setAction("next_image");

LocalBroadcastManager.getInstance(this).sendBroadcast(next_picture_intent);
                    cmd.append(">> Comando recibido: " + comando + "\n");
                    break;
                case "prev_image":
                    Intent previus_picture_intent = new Intent();
                    previus_picture_intent.setAction("previus_image");

LocalBroadcastManager.getInstance(this).sendBroadcast(previus_picture_intent);
                    cmd.append(">> Comando recibido: " + comando + "\n");
                    break;
                case "rotate_image":
                    Intent rotate_image_intent = new Intent();
                    rotate_image_intent.setAction("rotate_image");

LocalBroadcastManager.getInstance(this).sendBroadcast(rotate_image_intent);
                    cmd.append(">> Comando recibido: " + comando + "\n");
                    break;
                case "calendar"://Inicia la actividad formulario con las opciones
necesarias para añadir un evento a la app calendario por defecto
                    if (STATIC_ACTIVITY == 0) {
                        cmd.append(">> Comando recibido: " + comando + "\n");
                        numOpt = 5;
                        option1 = "(*) Fecha y hora de incio del evento
(dd/mm/aa/hh:mm)";
                        option2 = "Fecha y hora del fin del evento (dd/mm/aa/hh:mm)";
                        option3 = "(*) Titulo del evento";
                        option4 = "Descripcion del evento";
                        option5 = "Localizacion del evento";
                        Intent formCalendarIntent = new Intent(this,
FormActivity.class);
                        formCalendarIntent.putExtra("numOpt", numOpt);
                        formCalendarIntent.putExtra("option1", option1);
                        formCalendarIntent.putExtra("option2", option2);
                        formCalendarIntent.putExtra("option3", option3);
                        formCalendarIntent.putExtra("option4", option4);
                        formCalendarIntent.putExtra("option5", option5);
                        startActivityForResult(formCalendarIntent,
REQUEST_FORM_CALENDAR);
                        CURRENT_ACTIVITY = REQUEST_FORM_CALENDAR;
                    }
                    break;
                case "setAlarm"://Inicia la actividad formulario con las opciones
necesarias para poner una alarma
                    if (STATIC_ACTIVITY == 0) {
                        cmd.append(">> Comando recibido: " + comando + "\n");
                        numOpt = 2;
                        option1 = "(*)Hora de la alarma (hh:mm)";
                        option2 = "Día de la semana (ANDROID 5.0 O SUPERIOR)";
                        Intent formAlarmIntent = new Intent(this, FormActivity.class);
                        formAlarmIntent.putExtra("numOpt", numOpt);
                        formAlarmIntent.putExtra("option1", option1);
                        formAlarmIntent.putExtra("option2", option2);
                        startActivityForResult(formAlarmIntent, REQUEST_FORM_SETALARM);
                        CURRENT_ACTIVITY = REQUEST_FORM_SETALARM;
                    }
```

```
                    break;
                case "viewContact"://Inicia la actividad formulario con las opciones
necesarias para ver un contacto
                    if (STATIC_ACTIVITY == 0) {
                        numOpt = 1;
                        option1 = "(*) Nombre del contacto a visualizar";
                        Intent formViewCntIntent = new Intent(this, FormActivity.class);
                        formViewCntIntent.putExtra("numOpt", numOpt);
                        formViewCntIntent.putExtra("option1", option1);
                        startActivityForResult(formViewCntIntent,
REQUEST_FORM_VIEWCONTACT);
                        CURRENT_ACTIVITY = REQUEST_FORM_VIEWCONTACT;
                        cmd.append(">> Comando recibido: " + comando + "\n");
                    }
                    break;
                case "deleteContact"://Inicia la actividad formulario con las opciones
necesarias para borrar un contacto
                    if (STATIC_ACTIVITY == 0) {
                        numOpt = 1;
                        option1 = "(*) Contacto a borrar";
                        Intent formEditCntIntent = new Intent(this, FormActivity.class);
                        formEditCntIntent.putExtra("numOpt", numOpt);
                        formEditCntIntent.putExtra("option1", option1);
                        startActivityForResult(formEditCntIntent,
REQUEST_FORM_DELETECONTACT);
                        CURRENT_ACTIVITY = REQUEST_FORM_DELETECONTACT;
                        cmd.append(">> Comando recibido: " + comando + "\n");
                    }
                    break;
                case "newContact"://Inicia la actividad formulario con las opciones
necesarias para añadir un contacto
                    if (STATIC_ACTIVITY == 0) {
                        numOpt = 3;
                        option1 = "(*) Nombre del contacto (obligatorio)";
                        option2 = "(*) Telefono del contacto (obligatorio)";
                        option3 = "Email del contacto (puede dejarse en blanco)";
                        Intent formNewCntIntent = new Intent(this, FormActivity.class);
                        formNewCntIntent.putExtra("numOpt", numOpt);
                        formNewCntIntent.putExtra("option1", option1);
                        formNewCntIntent.putExtra("option2", option2);
                        formNewCntIntent.putExtra("option3", option3);
                        startActivityForResult(formNewCntIntent,
REQUEST_FORM_NEWCONTACT);
                        CURRENT_ACTIVITY = REQUEST_FORM_NEWCONTACT;
                        cmd.append(">> Comando recibido: " + comando + "\n");
                    }
                    break;
                case "back"://Cierra la actividad actual
                    cmd.append(">> Comando recibido: " + comando + "\n");
                    finishActivity(CURRENT_ACTIVITY);
                    STATIC_ACTIVITY = 0;
                    break;
                default: //Letras, numeros y demas opciones del teclado
                    Intent sendText = new Intent();
                    sendText.setAction("key_received");
                    sendText.putExtra("key", comando);
                    LocalBroadcastManager.getInstance(this).sendBroadcast(sendText);
                    break;
            }
        }
    }

    //Aqui se recbien los resultados de la actividad formActivity o cualquier otra
actividad iniciada
    //en el hilo MainActivity cuando esta ha acabado
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        String contactInfo[];
        String opt1, opt2, opt3, opt4, opt5;
        switch (requestCode) {
            case REQUEST_FORM_CALENDAR:
                if (resultCode == RESULT_OK) {
                    opt1 = data.getExtras().getString("edit1"); //Fecha inicio
dd/mm/aa/hh:mm
                    opt2 = data.getExtras().getString("edit2"); //Fecha final
dd/mm/aa/hh:mm
                    opt3 = data.getExtras().getString("edit3"); //Titulo del evento
                    opt4 = data.getExtras().getString("edit4"); //Descripcion del eventp
```

```
                    opt5 = data.getExtras().getString("edit5"); //Localizacion

                    String beginDate[] = opt1.split("/");
                    String beginTime[] = beginDate[3].split(":");
                    Calendar begin = Calendar.getInstance();
                    begin.set(Integer.parseInt(beginDate[2]),
Integer.parseInt(beginDate[1]) - 1, Integer.parseInt(beginDate[0]),
Integer.parseInt(beginTime[0]), Integer.parseInt(beginTime[1]));
                    Intent Calendarintent = new Intent(Intent.ACTION_INSERT);
                    Calendarintent.setData(CalendarContract.Events.CONTENT_URI);
                    Calendarintent.putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME,
begin.getTimeInMillis());
                    Calendarintent.putExtra(CalendarContract.Events.TITLE, opt3);
                    Calendarintent.putExtra(CalendarContract.Events.DESCRIPTION, opt4);
                    Calendarintent.putExtra(CalendarContract.Events.EVENT_LOCATION,
opt5);
                    Calendarintent.putExtra(CalendarContract.Events.AVAILABILITY,
CalendarContract.Events.AVAILABILITY_BUSY);
                    if (!opt2.equals("")) {
                        String endDate[] = opt2.split("/");
                        String endTime[] = endDate[3].split(":");
                        Calendar end = Calendar.getInstance();
                        end.set(Integer.parseInt(endDate[2]),
Integer.parseInt(endDate[1]) - 1, Integer.parseInt(endDate[0]),
Integer.parseInt(endTime[0]), Integer.parseInt(endTime[1]));
                        Calendarintent.putExtra(CalendarContract.EXTRA_EVENT_END_TIME,
end.getTimeInMillis());
                    }
                    startActivityForResult(Calendarintent, REQUEST_CALENDAR);
                    CURRENT_ACTIVITY = REQUEST_CALENDAR;
                }
                break;
            case REQUEST_FORM_CALL:
                if (resultCode == RESULT_OK) {
                    opt1 = data.getExtras().getString("edit1");
                    //Si comando_separado[1] es un contacto lo convertimos en numero de
telefono
                    contactInfo = getPhoneNumberFromContact(opt1);
                    if (contactInfo[1] == null) contactInfo[1] = opt1;
                    Intent callIntent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:"
+ contactInfo[1]));
                    if (callIntent.resolveActivity(getPackageManager()) != null) {
                        startActivityForResult(callIntent, REQUEST_CALL);
                        CURRENT_ACTIVITY = REQUEST_CALL;
                    }
                }
                break;
            case REQUEST_FORM_SMS:
                if (resultCode == RESULT_OK) {
                    opt1 = data.getExtras().getString("edit1");
                    opt2 = data.getExtras().getString("edit2");
                    //Si comando_separado[1] es un contacto lo convertimos en numero de
telefono
                    contactInfo = getPhoneNumberFromContact(opt1);
                    if (contactInfo[1] == null) contactInfo[1] = opt1;
                    try {
                        SmsManager smsManager = SmsManager.getDefault();
                        smsManager.sendTextMessage(contactInfo[1], null, opt2, null,
null);
                        Toast.makeText(getApplicationContext(), "Mensaje enviado
correctamente", Toast.LENGTH_LONG).show();
                    } catch (Exception ex) {
                        Toast.makeText(getApplicationContext(), ex.getMessage(),
Toast.LENGTH_LONG).show();
                    }
                }
                break;
            case REQUEST_FORM_WHATSAPP:
                if (resultCode == RESULT_OK) {
                    opt1 = data.getExtras().getString("edit1");
                    //Send Whatsapp
                    Intent sendIntent = new Intent();
                    sendIntent.setAction(Intent.ACTION_SEND);
                    sendIntent.setType("text/plain");
                    sendIntent.putExtra(Intent.EXTRA_TEXT, opt1);
                    sendIntent.setPackage("com.whatsapp");
                    if (sendIntent.resolveActivity(getPackageManager()) != null) {
```

```
                        startActivityForResult(sendIntent, REQUEST_SEND_WHATSAPP);
                        CURRENT_ACTIVITY = REQUEST_SEND_WHATSAPP;
                    }
                }
                break;
            case REQUEST_FORM_SETALARM:
                if (resultCode == RESULT_OK) {
                    opt1 = data.getExtras().getString("edit1"); //Hora en formato hh:mm
                    opt2 = data.getExtras().getString("edit2"); //Dia de la semana
(opcional)
                    String hour_minutes[] = opt1.split(":");
                    //Set alarm
                    int day_of_week = 0;
                    if (opt2.equals("lunes")) day_of_week = Calendar.MONDAY;
                    if (opt2.equals("martes")) day_of_week = Calendar.TUESDAY;
                    if (opt2.equals("miercoles")) day_of_week = Calendar.WEDNESDAY;
                    if (opt2.equals("jueves")) day_of_week = Calendar.THURSDAY;
                    if (opt2.equals("viernes")) day_of_week = Calendar.FRIDAY;
                    if (opt2.equals("sabado")) day_of_week = Calendar.SATURDAY;
                    if (opt2.equals("domingo")) day_of_week = Calendar.SUNDAY;
                    Intent setAlarm_intent = new Intent(AlarmClock.ACTION_SET_ALARM)
                            .putExtra(AlarmClock.EXTRA_DAYS, day_of_week)
                            .putExtra(AlarmClock.EXTRA_HOUR,
Integer.parseInt(hour_minutes[0]))
                            .putExtra(AlarmClock.EXTRA_MINUTES,
Integer.parseInt(hour_minutes[1]));
                    if (setAlarm_intent.resolveActivity(getPackageManager()) != null) {
                        startActivityForResult(setAlarm_intent, REQUEST_SETALARM);
                        CURRENT_ACTIVITY = REQUEST_SETALARM;
                        cmd.append(">> Alarma activada\n");
                    } else {
                        cmd.append(">> No se ha podido activar la alarma\n");
                    }
                }
                break;
            case REQUEST_FORM_VIEWCONTACT:
                if (resultCode == RESULT_OK) {
                    opt1 = data.getExtras().getString("edit1");
                    // New contact
                    contactInfo = getPhoneNumberFromContact(opt1);
                    if (contactInfo[0] != null) {
                        Uri viewContacturi =
Uri.withAppendedPath(ContactsContract.Contacts.CONTENT_URI,
String.valueOf(contactInfo[0]));
                        Intent viewContact_intent = new Intent(Intent.ACTION_VIEW,
viewContacturi);
                        if (viewContact_intent.resolveActivity(getPackageManager()) !=
null) {
                            startActivityForResult(viewContact_intent,
REQUEST_VIEWCONTACT);
                            CURRENT_ACTIVITY = REQUEST_VIEWCONTACT;
                        }
                    } else {
                        cmd.append(">> No existe el contacto " + opt1 + "\n");
                    }
                }
                break;
            case REQUEST_FORM_DELETECONTACT:
                if (resultCode == RESULT_OK) {
                    opt1 = data.getExtras().getString("edit1");
                    //Edit contact
                    contactInfo = getPhoneNumberFromContact(opt1);
                    if (contactInfo[0] != null) {
                        Uri contactUri =
Uri.withAppendedPath(ContactsContract.PhoneLookup.CONTENT_FILTER_URI,
Uri.encode(contactInfo[1]));
                        Cursor cur = this.getContentResolver().query(contactUri, null,
null, null, null);
                        try {
                            if (cur.moveToFirst()) {
                                do {
                                    if
(cur.getString(cur.getColumnIndex(ContactsContract.PhoneLookup.DISPLAY_NAME)).equalsIgno
reCase(opt1)) {
                                        String lookupKey =
cur.getString(cur.getColumnIndex(ContactsContract.Contacts.LOOKUP_KEY));
```

```
                                        Uri uri =
Uri.withAppendedPath(ContactsContract.Contacts.CONTENT_LOOKUP_URI, lookupKey);
                                        this.getContentResolver().delete(uri, null,
null);
                                    }

                                } while (cur.moveToNext());
                                cmd.append(">> Contacto eliminado correctamente\n");
                            }

                        } catch (Exception e) {
                            System.out.println(e.getStackTrace());
                            Toast.makeText(this, "El contacto no se ha podido borrar",
Toast.LENGTH_LONG);
                        } finally {
                            cur.close();
                        }
                    } else {
                        cmd.append(">> No existe ese contacto\n");
                    }
                }
                break;
            case REQUEST_FORM_NEWCONTACT:
                if (resultCode == RESULT_OK) {

                    String DisplayName = data.getExtras().getString("edit1");
                    String MobileNumber = data.getExtras().getString("edit2");
                    String emailID = data.getExtras().getString("edit3");

                    ArrayList< ContentProviderOperation > ops = new ArrayList <
ContentProviderOperation > ();

                    ops.add(ContentProviderOperation.newInsert(
                            ContactsContract.RawContacts.CONTENT_URI)
                            .withValue(ContactsContract.RawContacts.ACCOUNT_TYPE, null)
                            .withValue(ContactsContract.RawContacts.ACCOUNT_NAME, null)
                            .build());

                    //------------------------------------------------------ Names
                    if (!DisplayName.equals("")) {
                        ops.add(ContentProviderOperation.newInsert(
                                ContactsContract.Data.CONTENT_URI)

.withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, 0)
                                .withValue(ContactsContract.Data.MIMETYPE,

ContactsContract.CommonDataKinds.StructuredName.CONTENT_ITEM_TYPE)
                                .withValue(

ContactsContract.CommonDataKinds.StructuredName.DISPLAY_NAME,
                                        DisplayName).build());
                    }

                    //------------------------------------------------------ Mobile
Number
                    if (!MobileNumber.equals("")) {
                        ops.add(ContentProviderOperation.
                                newInsert(ContactsContract.Data.CONTENT_URI)

.withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, 0)
                                .withValue(ContactsContract.Data.MIMETYPE,

ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE)

.withValue(ContactsContract.CommonDataKinds.Phone.NUMBER, MobileNumber)
                                .withValue(ContactsContract.CommonDataKinds.Phone.TYPE,

ContactsContract.CommonDataKinds.Phone.TYPE_MOBILE)
                                .build());
                    }

                    //------------------------------------------------------ Email
                    if (!emailID.equals("")) {

ops.add(ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_URI)

.withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, 0)
```

```
                                .withValue(ContactsContract.Data.MIMETYPE,
ContactsContract.CommonDataKinds.Email.CONTENT_ITEM_TYPE)
                                .withValue(ContactsContract.CommonDataKinds.Email.DATA,
emailID)
                                .withValue(ContactsContract.CommonDataKinds.Email.TYPE,
ContactsContract.CommonDataKinds.Email.TYPE_WORK)
                                .build());
                    }

                    // Asking the Contact provider to create a new contact
                    try {
                        getContentResolver().applyBatch(ContactsContract.AUTHORITY,
ops);
                        cmd.append(">> Contacto añadido correctamente\n");
                    } catch (Exception e) {
                        e.printStackTrace();
                        Toast.makeText(this, "Exception: " + e.getMessage(),
Toast.LENGTH_SHORT).show();
                    }
                }
                break;
        }
    }

    //Funcion que convierte el numbre de contacto contactName en el numero de telefono
correspondiente
    public String[] getPhoneNumberFromContact(String contactName) {
        String[] contactInfo = new String[2];
        ContentResolver cr = getContentResolver();
        Cursor cursor = cr.query(ContactsContract.Contacts.CONTENT_URI, null,
"DISPLAY_NAME = '" + contactName + "'", null, null);
        if (cursor.moveToFirst()) {
            contactInfo[0] =
cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts._ID));
            Cursor phones = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
null, ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = " + contactInfo[0], null,
null);
            while (phones.moveToNext()) {
                String number =
phones.getString(phones.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));
                int type =
phones.getInt(phones.getColumnIndex(ContactsContract.CommonDataKinds.Phone.TYPE));
                contactInfo[1] = number;
            }
            phones.close();
        }
        cursor.close();
        return contactInfo;
    }

    //Cuando el estado del telefono cambia a descolgado se activa el altavoz para poner
el manos libres
    public class StatePhoneReceiver extends PhoneStateListener {
        @Override
        public void onCallStateChanged(int state, String incomingNumber) {

            if (state != TelephonyManager.CALL_STATE_IDLE) {
                AudioManager audioManager = (AudioManager)
getSystemService(Context.AUDIO_SERVICE);
                audioManager.setMode(AudioManager.MODE_IN_CALL);
                audioManager.setSpeakerphoneOn(true);
            }

        }
    }


    //Clase acceptThread: Corre en un hilo diferente al hilo principal (UI). Se encarga
de recibir
    //peticiones de conexion y crear un socket para ellas.
    public class AcceptThread  extends Thread {

        private final BluetoothServerSocket mmServerSocket;
        public AcceptThread() {
            // Use a temporary object that is later assigned to mmServerSocket,
            // because mmServerSocket is final
```

```
            BluetoothServerSocket tmp = null;
            try {
                // MY UUID is the app's UUID string, also used by the client code
                tmp = mBluetoothAdapter.listenUsingRfcommWithServiceRecord(NAME,
MY_UUID);
            } catch (IOException e) {
                cmd.append(">> ERROR " + e.getMessage());
            }
            mmServerSocket = tmp;
        }

        public void run() {
            BluetoothSocket socket = null;
            // Keep listening until exception occurs or a socket is returned
            while (true) {
                try {
                    socket = mmServerSocket.accept();
                } catch (IOException e) {
                    break;
                }
                // If a connection was accepted
                if (socket != null) {
                    // Do work to manage the connection (in a separate thread)
                    ConnectedThread connectedThread = new ConnectedThread(socket);
                    connectedThread.start();
                    try {
                        mmServerSocket.close();
                    }catch (Exception e) {
                        cmd.append(">> ERROR: "+ e.getMessage());
                    }
                    break;
                }
            }
        }

        /** Will cancel the listening socket, and cause the thread to finish */
        public void cancel() {
            try {
                mmServerSocket.close();
            } catch (IOException e) { }
        }

    }

    public class ConnectedThread  extends Thread {

        BluetoothSocket mmSocket;
        InputStream mmInStream;

        public ConnectedThread(BluetoothSocket socket){

            mmSocket = socket;
            InputStream tmpIn = null;
            // Get the input and output streams, using temp objects because
            // member streams are final
            try {
                tmpIn = mmSocket.getInputStream();
            } catch (IOException e) {
                cmd.append(">> ERROR: " + e.getMessage());
            }
            mmInStream = tmpIn;
        }

        //En esta funcion recibimos los datos en el socket mmSocket
        public void run() {
            // buffer de 1024 caracteres (La aplicacion cliente manda caracteres de 16
bits cada uno)
            byte[] buffer = new byte[2048];
            int bytes; // bytes returned from read()
            String msg="";

            Pattern p = Pattern.compile("\u0000");
               // Escuchamos en el buffer de entrada del socket hasta que se recibe la
               cadena closeSocket o  ocurre una excepción.
            while (true) {
                try {
                    // Read from the InputStream
```

```
                    bytes = mmInStream.read(buffer);
                    msg = new String(buffer);
                    //Eliminamos del mensage los caracteres no utilizados (por
                defecto el mensaje tiene 1024 bytes. Los que no se utilizan
                      se rellenan con el carater 0000 en hexadecimal
                    Matcher m = p.matcher(msg);
                    if (m.find()) {
                        msg = m.replaceAll("");
                    }
                    //Enviamos la cadena recibida y tratada al hilo principal
                      para hacer la accion correspondiente.
                    //Si llamaramos directamente a iniciarActividad() desde
                      aqui la accion se procesaria en este hilo
                    //quedando inactivo el hilo principal y siendo
                      ineficiente.
                    Message handlerMessage = new Message();
                    handlerMessage.obj = msg;
                    handler.sendMessage(handlerMessage);
                    break;
                } catch (IOException e) {
                    cmd.append(">> ERROR: " + e.getMessage()+ ". Restart
                      bluetooth server");
                    break;
                }
            }
            cancel();
        }

        /* Call this from the main activity to shutdown the connection */
        public void cancel() {
            try {
                mmSocket.close();
            } catch (IOException e) { }
        }
    }
}


//Archivo CameraActivity
package com.bci.eduardo.apprunnerthreads;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.hardware.Camera;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.support.v4.app.FragmentActivity;
import android.support.v4.content.LocalBroadcastManager;
import android.util.Log;
import android.view.KeyEvent;
import android.widget.FrameLayout;
import android.widget.Toast;

import com.bci.eduardo.apprunner.R;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class CameraActivity extends FragmentActivity {

    Take_picture_receiver take_picture_receiver;
    private static Camera mCamera;
    private static CameraPreview mPreview;
    public static final int MEDIA_TYPE_IMAGE = 1;
    public static final String TAG = CameraActivity.class.getSimpleName();
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_camera);
```

```
        //Añadimos los receivers para los distintos mensajes que podra manejar la
actividad CameraActivity
        take_picture_receiver = new Take_picture_receiver();
        LocalBroadcastManager.getInstance(this).registerReceiver(take_picture_receiver,
new IntentFilter("take_picture"));

        // Create an instance of Camera
        mCamera = getCameraInstance();
        // Create our Preview view and set it as the content of our activity.
        mPreview = new CameraPreview(this, mCamera);
        FrameLayout preview = (FrameLayout) findViewById(R.id.camera_preview);
        preview.addView(mPreview);
    }
    //Cuando recibimos el intento con la action take_picture desde el objeto
Take picture receiver y
    //tomamos la foto.
    public class Take_picture_receiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context arg0, Intent intent) {
            final Camera.PictureCallback mPicture = new Camera.PictureCallback() {
                @Override
                public void onPictureTaken(byte[] data, Camera camera) {

                    File pictureFile = getOutputMediaFile(MEDIA_TYPE_IMAGE);
                    if (pictureFile == null) {
                        Log.d(TAG, "Error creating media file, check storage
permissions: ");
                        return;
                    }

                    try {
                        FileOutputStream fos = new FileOutputStream(pictureFile);
                        fos.write(data);
                        fos.close();
                    } catch (FileNotFoundException e) {
                        Log.d(TAG, "File not found: " + e.getMessage());
                    } catch (IOException e) {
                        Log.d(TAG, "Error accessing file: " + e.getMessage());
                    }
                }
            };
            mCamera.takePicture(null, null, mPicture);
        }
    }

    public Camera getCameraInstance() {
        Camera c = null;
        try {
            c = Camera.open(); // attempt to get a Camera instance
        } catch (Exception e) {
            // Camera is not available (in use or does not exist)
        }
        return c; // returns null if camera is unavailable
    }

    /**
     * Create a file Uri for saving an image or video
     */
    private Uri getOutputMediaFileUri(int type) {
        return Uri.fromFile(getOutputMediaFile(type));
    }

    /**
     * Create a File for saving an image or video
     */
    private File getOutputMediaFile(int type) {
        // To be safe, you should check that the SDCard is mounted
        // using Environment.getExternalStorageState() before doing this.

        File mediaStorageDir = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES),
"BCI_Camera");
        // This location works best if you want the created images to be shared
        // between applications and persist after your app has been uninstalled.

        // Create the storage directory if it does not exist
        if (!mediaStorageDir.exists()) {
```

```
            if (!mediaStorageDir.mkdirs()) {
                return null;
            }
        }

        // Create a media file name
        String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
        File mediaFile;
        if (type == MEDIA_TYPE_IMAGE) {
            mediaFile = new File(mediaStorageDir.getPath() + File.separator + "IMG_" +
timeStamp + ".jpg");
        } else {
            return null;
        }
        return mediaFile;
    }

    //Esta funcion se llama al cerrar la actividad
    protected void onDestroy () {
        super.onDestroy();

LocalBroadcastManager.getInstance(this).unregisterReceiver(take_picture_receiver);
        if (mCamera != null) {
            mCamera.stopPreview();
            mCamera.release();
            mCamera = null;
        }
    }
}

//Archivo CamaraPreview.javapackage com.bci.eduardo.apprunnerthreads;

import android.content.Context;
import android.hardware.Camera;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import java.io.IOException;

public class CameraPreview extends SurfaceView implements SurfaceHolder.Callback {
    private SurfaceHolder mHolder;
    private Camera mCamera;
    public static final String TAG = CameraActivity.class.getSimpleName();

    public CameraPreview(Context context, Camera camera) {
        super(context);
        mCamera = camera;

        // Install a SurfaceHolder.Callback so we get notified when the
        // underlying surface is created and destroyed.
        mHolder = getHolder();
        mHolder.addCallback(this);
        // deprecated setting, but required on Android versions prior to 3.0
        mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    }

    public void surfaceCreated(SurfaceHolder holder) {
        // The Surface has been created, now tell the camera where to draw the preview.
        try {
            mCamera.setPreviewDisplay(holder);
            mCamera.startPreview();
        } catch (IOException e) {
            Log.d(TAG, "Error setting camera preview: " + e.getMessage());
        }
    }

    public void surfaceDestroyed(SurfaceHolder holder) {
        // empty. Take care of releasing the Camera preview in your activity.
    }

    public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {
        // If your preview can change or rotate, take care of those events here.
        // Make sure to stop the preview before resizing or reformatting it.

        if (mHolder.getSurface() == null){
            // preview surface does not exist
            return;
```

```
        }

        // stop preview before making changes
        try {
            mCamera.stopPreview();
        } catch (Exception e){
            // ignore: tried to stop a non-existent preview
        }

        // set preview size and make any resize, rotate or
        // reformatting changes here

        // start preview with new settings
        try {
            mCamera.setPreviewDisplay(mHolder);
            mCamera.startPreview();

        } catch (Exception e){
            Log.d(TAG, "Error starting camera preview: " + e.getMessage());
        }
    }
}

//Archivo FormActivitypackage com.bci.eduardo.apprunnerthreads;
import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.graphics.drawable.BitmapDrawable;
import android.net.Uri;
import android.os.Bundle;
import android.support.v4.content.LocalBroadcastManager;
import android.view.Gravity;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.bci.eduardo.apprunner.R;
public class FormActivity extends Activity {

    Key_received key_received;
    Bundle extras;
    int numOpt;
    Intent formIntent;
    TextView text1,text2,text3, text4, text5;
    EditText edit1, edit2, edit3, edit4, edit5, currentEditText;
    int selectedEdit=1;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.form);

        //Añadimos los receivers para los distintos mensajes que podra manejar la
actividad FormActivity
        key_received = new Key_received();
        LocalBroadcastManager.getInstance(this).registerReceiver(key_received, new
IntentFilter("key_received"));

        LinearLayout layout = (LinearLayout) findViewById(R.id.form);
        formIntent = getIntent();
        extras = formIntent.getExtras();

        numOpt = extras.getInt("numOpt");
        if (numOpt == 1){
            edit1 = new EditText(this);
            edit1.setTextSize(20);
            text1 = new TextView(this);
            text1.setGravity(Gravity.CENTER);
            text1.setTextSize(20);
            text1.setText(extras.getString("option1"));
            layout.addView(text1);
            layout.addView(edit1);
            currentEditText = edit1;
        }
```

```
if (numOpt == 2){
    edit1 = new EditText(this);
    edit1.setTextSize(20);
    edit2 = new EditText(this);
    edit2.setTextSize(20);
    text1 = new TextView(this);
    text1.setGravity(Gravity.CENTER);
    text1.setTextSize(20);
    text1.setText(extras.getString("option1"));
    text2 = new TextView(this);
    text2.setGravity(Gravity.CENTER);
    text2.setTextSize(20);
    text2.setText(extras.getString("option2"));
    layout.addView(text1);
    layout.addView(edit1);
    layout.addView(text2);
    layout.addView(edit2);
    currentEditText = edit1;
}

if (numOpt == 3){
    edit1 = new EditText(this);
    edit1.setTextSize(20);
    edit2 = new EditText(this);
    edit2.setTextSize(20);
    edit3 = new EditText(this);
    edit3.setTextSize(20);
    text1 = new TextView(this);
    text1.setGravity(Gravity.CENTER);
    text1.setTextSize(20);
    text1.setText(extras.getString("option1"));
    text2 = new TextView(this);
    text2.setGravity(Gravity.CENTER);
    text2.setTextSize(20);
    text2.setText(extras.getString("option2"));
    text3 = new TextView(this);
    text3.setGravity(Gravity.CENTER);
    text3.setTextSize(20);
    text3.setText(extras.getString("option3"));
    layout.addView(text1);
    layout.addView(edit1);
    layout.addView(text2);
    layout.addView(edit2);
    layout.addView(text3);
    layout.addView(edit3);
    currentEditText = edit1;
}

if (numOpt == 5){
    edit1 = new EditText(this);
    edit1.setTextSize(20);
    edit2 = new EditText(this);
    edit2.setTextSize(20);
    edit3 = new EditText(this);
    edit3.setTextSize(20);
    edit4 = new EditText(this);
    edit4.setTextSize(20);
    edit5 = new EditText(this);
    edit5.setTextSize(20);
    text1 = new TextView(this);
    text1.setGravity(Gravity.CENTER);
    text1.setTextSize(20);
    text1.setText(extras.getString("option1"));
    text2 = new TextView(this);
    text2.setGravity(Gravity.CENTER);
    text2.setTextSize(20);
    text2.setText(extras.getString("option2"));
    text3 = new TextView(this);
    text3.setGravity(Gravity.CENTER);
    text3.setTextSize(20);
    text3.setText(extras.getString("option3"));
    text4 = new TextView(this);
    text4.setGravity(Gravity.CENTER);
    text4.setTextSize(20);
    text4.setText(extras.getString("option4"));
    text5 = new TextView(this);
    text5.setGravity(Gravity.CENTER);
```

```
            text5.setTextSize(20);
            text5.setText(extras.getString("option5"));
            layout.addView(text1);
            layout.addView(edit1);
            layout.addView(text2);
            layout.addView(edit2);
            layout.addView(text3);
            layout.addView(edit3);
            layout.addView(text4);
            layout.addView(edit4);
            layout.addView(text5);
            layout.addView(edit5);
            currentEditText = edit1;
        }
    }

    public class Key_received extends BroadcastReceiver {
        @Override
        public void onReceive(Context arg0, Intent intent) {
            String key = intent.getExtras().getString("key");
            switch (key){
                case "enter":
                    selectedEdit++;
                    if (selectedEdit <= numOpt) {
                        if (selectedEdit == 2){
                            edit2.requestFocus();
                            currentEditText = edit2;
                        }
                        if (selectedEdit == 3){
                            edit3.requestFocus();
                            currentEditText = edit3;
                        }
                        if (selectedEdit == 4){
                            edit4.requestFocus();
                            currentEditText = edit4;
                        }
                        if (selectedEdit == 5){
                            edit5.requestFocus();
                            currentEditText = edit5;
                        }
                    }else{
                        exitActivity();
                    }
                    break;
                case "del":
                    int length = currentEditText.getText().length();
                    if (length > 0) {
                        currentEditText.getText().delete(length - 1, length);
                    }
                    break;
                case "clear":
                    currentEditText.setText("");
                    break;
                case "prev_edit":
                    if(selectedEdit > 1) selectedEdit--;
                    if (selectedEdit == 1){
                        edit1.requestFocus();
                        currentEditText = edit1;
                    }
                    if (selectedEdit == 2){
                        edit2.requestFocus();
                        currentEditText = edit2;
                    }
                    if (selectedEdit == 3){
                        edit3.requestFocus();
                        currentEditText = edit3;
                    }
                    if (selectedEdit == 4){
                        edit4.requestFocus();
                        currentEditText = edit4;
                    }
                    break;
                default:
                    currentEditText.append(key);
                    break;
            }
        }
```

```
    }

    public void exitActivity(){

        if (numOpt==1){
            setResult(Activity.RESULT_OK, new Intent().
                    putExtra("edit1",edit1.getText().toString()));
            finish();
        }

        if (numOpt == 2){
            setResult(Activity.RESULT_OK, new Intent().
                    putExtra("edit1",edit1.getText().toString()).
                    putExtra("edit2", edit2.getText().toString()));
            finish();
        }

        if(numOpt == 3){
            setResult(Activity.RESULT_OK, new Intent().
                    putExtra("edit1",edit1.getText().toString()).
                    putExtra("edit2", edit2.getText().toString()).
                    putExtra("edit3", edit3.getText().toString()));
            finish();
        }

        if(numOpt == 4){
            setResult(Activity.RESULT_OK, new Intent().
                    putExtra("edit1",edit1.getText().toString()).
                    putExtra("edit2", edit2.getText().toString()).
                    putExtra("edit3", edit3.getText().toString()).
                    putExtra("edit4", edit4.getText().toString()));
            finish();
        }

        if(numOpt == 5){
            setResult(Activity.RESULT_OK, new Intent().
                    putExtra("edit1",edit1.getText().toString()).
                    putExtra("edit2", edit2.getText().toString()).
                    putExtra("edit3", edit3.getText().toString()).
                    putExtra("edit4", edit4.getText().toString()).
                    putExtra("edit5", edit5.getText().toString()));
            finish();
        }
    }

    //Esta funcion se llama al cerrar la actividad
    protected void onDestroy () {
        super.onDestroy();
        LocalBroadcastManager.getInstance(this).unregisterReceiver(key_received);
        setResult(Activity.RESULT_CANCELED);
    }
}


//Archivo ImgViewActivity.java
package com.bci.eduardo.apprunnerthreads;

import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.graphics.drawable.BitmapDrawable;
import android.net.Uri;
import android.os.Bundle;
import android.support.v4.content.LocalBroadcastManager;
import android.view.Display;
import android.widget.ImageView;
import android.widget.Toast;
import com.bci.eduardo.apprunner.R;
import java.io.File;

public class ImageViewActivity extends Activity {
```

```
    Next_image_receiver next_image_receiver;
    Previus_image_receiver previus_image_receiver;
    Rotate_image_receiver rotate_image_receiver;
    File[] listFiles;
    File file = new File("/storage/sdcard0/Pictures/BCI_Camera");
    ImageView imageView;
    static Bitmap bitmap;
    int i=0;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_imageview);

        imageView = (ImageView) findViewById(R.id.imageView);
        listFiles = file.listFiles();
        //Añadimos los receivers para los distintos mensajes que podra manejar la
actividad ImageViewActivity
        next_image_receiver = new Next_image_receiver();
        LocalBroadcastManager.getInstance(this).registerReceiver(next_image_receiver,
new IntentFilter("next_image"));

        previus_image_receiver = new Previus_image_receiver();
        LocalBroadcastManager.getInstance(this).registerReceiver(previus_image_receiver,
new IntentFilter("previus_image"));

        rotate_image_receiver = new Rotate_image_receiver();
        LocalBroadcastManager.getInstance(this).registerReceiver(rotate_image_receiver,
new IntentFilter("rotate_image"));

        //LLamamos a la funcion loadImageOnView para cargar la imagen 0
        loadImageOnView(0);
    }

    public class Next_image_receiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context arg0, Intent intent) {
            i = i+1;
            if(i < listFiles.length) {
                bitmap.recycle();
                loadImageOnView(i);
            }else{
                Toast.makeText(ImageViewActivity.this, "No hay mas imagenes hacia
adelante", Toast.LENGTH_LONG).show();
                i = listFiles.length;
            }
        }
    }

    public class Previus_image_receiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context arg0, Intent intent) {
            i = i-1;
            if (i == listFiles.length-1) i = i-1;
            if(i >= 0) {
                bitmap.recycle();
                loadImageOnView(i);
            }else{
                Toast.makeText(ImageViewActivity.this, "No hay mas imagenes hacia
atras", Toast.LENGTH_LONG).show();
                i = 0;
            }
        }
    }

    public class Rotate_image_receiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context arg0, Intent intent) {
            try{
                //Cargamos la imagen actual
                //Rotacion de 90 grados
                Matrix matrix = new Matrix();
                matrix.postRotate(90);
                bitmap = Bitmap.createBitmap(bitmap, 0, 0, bitmap.getWidth(),
bitmap.getHeight(), matrix, true);
                imageView.setImageBitmap(bitmap);

            }catch (Exception e){
```

```
                Toast.makeText(ImageViewActivity.this, e.getMessage(),
Toast.LENGTH_LONG).show();
            }
        }
    }

    public void loadImageOnView(int i){

        try {
            Display d = getWindowManager().getDefaultDisplay();
            int screenWidth = d.getWidth();
            int screenHeight = d.getHeight();
            BitmapFactory.Options options = new BitmapFactory.Options();
            options.inSampleSize = 2;
            bitmap = BitmapFactory.decodeFile(listFiles[i].getPath(),options);
            bitmap = Bitmap.createScaledBitmap(bitmap, screenHeight, screenWidth, true);
            imageView.setImageBitmap(bitmap);
        }catch (Exception e){
            Toast.makeText(ImageViewActivity.this, e.getMessage(),
Toast.LENGTH_LONG).show();
        }
    }

    //Esta funcion se llama al cerrar la actividad
    protected void onDestroy () {
        super.onDestroy();
        LocalBroadcastManager.getInstance(this).unregisterReceiver(next_image_receiver);

LocalBroadcastManager.getInstance(this).unregisterReceiver(previus_image_receiver);

LocalBroadcastManager.getInstance(this).unregisterReceiver(rotate_image_receiver);
        bitmap.recycle();
        imageView.setImageBitmap(null);
    }
}

//Archivo AndroidManifiest.xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.bci.eduardo.apprunner">
    <uses-feature android:name="android.hardware.camera" android:required="true" />
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.CALL_PHONE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.SEND_SMS" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.WRITE_CONTACTS" />
    <uses-permission android:name="android.permission.READ_CALENDAR" />
    <uses-permission android:name="android.permission.WRITE_CALENDAR" />
    <uses-permission android:name="com.android.alarm.permission.SET_ALARM"/>
    <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="AppRunner"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name="com.bci.eduardo.apprunnerthreads.MainActivity"
            android:configChanges="orientation|screenSize">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="com.bci.eduardo.apprunnerthreads.CameraActivity"
            android:screenOrientation="landscape"
            android:theme="@android:style/Theme.NoTitleBar.Fullscreen">
        </activity>
        <activity android:name="com.bci.eduardo.apprunnerthreads.ImageViewActivity"
            android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
            android:configChanges="keyboardHidden|orientation|screenLayout|screenSize">
        </activity>
        <activity android:name="com.bci.eduardo.apprunnerthreads.FormActivity"
            android:configChanges="orientation|screenSize"/>
```

```
        <activity android:name="com.bci.eduardo.apprunnerthreads.HelpActivity"/>
    </application>
</manifest>

//Archivo MainActivity.xml<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.bci.eduardo.apprunnerthreads.MainActivity">

    <Button
        android:id="@+id/start"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="startServer"
        android:text="PULSAR AQUI PARA CREAR SERVIDOR BLUETOOTH" />

    <Button
        android:id="@+id/stop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="stopServer"
        android:visibility="invisible"
        android:text="PULSAR AQUI PARA CANCELAR SERVIDOR BLUETOOTH" />

    <TextView
        android:id="@+id/title"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="\nCommand Line"
        android:layout_below="@id/start"
        android:textSize="18dp" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/title">
        <TextView
            android:id="@+id/list"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textSize="14dp" />
    </ScrollView></RelativeLayout>
```

ANEXO C

TAREAS Y CUESTIONARIO

En este anexo se adjuntan los dos documentos que se les entregaron a los sujetos para la realización de las pruebas. El primero describe las tareas propuestas, mientras que el segundo es el cuestionario de satisfaccón que deben cumplimentar al termino de la segunda sesión

C.1. TAREAS PROPUESTAS

C.1.1 Procedimiento resumido

SESIÓN 1 – CALIBRACIÓN Y MANEJO DE LA APLICACIÓN **~75 min**

- Calibración del clasificador ~26 min
- Tarea 1: Manejo de galería ~4 min
- Tarea 2: Manejo de alarmas ~7 min
- Tarea 3: Manejo de llamadas ~9 min

SESIÓN 2 – EVALUACIÓN **~60 min**

- Comprobación del clasificador ~4-8 min
- Tarea 1: Realizar una fotografía ~2 min
- Tarea 2: Visualizar la fotografía anterior ~4 min
- Tarea 3: Añadir un contacto ~12 min
- Tarea 4: Visualizar el contacto ~5 min
- Tarea 5: Mandar SMS ~12 min
- Cuestionario de satisfacción ~2 min

Nota(1): se añaden unos 25 minutos extra para cubrir los tiempos de preparación de cada parte, problemas con el usuario o variación del número de secuencias.

Nota(2): si fuese necesario, se podrían incluir registros basales de 2 min antes y después de la realización de cada sesión.

C.1.2 Procedimiento detallado

a) SESIÓN 1 – CALIBRACIÓN Y MANEJO DE LA APLICACIÓN

CALIBRACIÓN DEL CLASIFICADOR	
CONFIGURACIÓN	15 SECUENCIAS con MATRIZ DE TECLADO 9x6
PROCEDIMIENTO	4 RONDAS de 6 TRIALS (4 palabras de 6 letras)
DURACIÓN APROX.	~26 min (~6 min/ronda + ~2 min SWLDA)
DESCRIPCIÓN	Se ordena al usuario escribir con la matriz de teclado cuatro palabras de seis letras cada una con el fin de obtener muestras de entrenamiento suficientes para hallar su clasificador óptimo. Posteriormente, con SWLDA se calculan los pesos y el número de secuencias óptimos (X) mediante la aplicación P300Classifier.

TAREA 1	**MANEJO DE GALERÍA**
SELEC. MÍNIMAS	6 selecciones
SELEC. ÓPTIMAS	IC.GALERIA IC.ROTAR_IMAGEN PAUSA (mirar foto) PAUSA IC.SIG_IMAGEN IC.VOLVER
DURACIÓN MÍNIMA.	~4 min (X=15 secuencias)
MATRICES A UTILIZAR	Matriz de aplicaciones
DESCRIPCIÓN	El usuario probará las funcionalidades de la galería de imágenes, así como del modo pausa de la aplicación mientras visualiza la fotografía.
TAREA 2	**MANEJO DE ALARMAS**
SELEC. MÍNIMAS	10 selecciones
SELEC. ÓPTIMAS	IC.ALARMA 1 4 : ENTER PREV_EDIT 3 0 ENTER ENTER MTX
DURACIÓN MÍNIMA.	~7 min (X=15 secuencias)
MATRICES A UTILIZAR	Ambas
DESCRIPCIÓN	El usuario probará a activar una alarma a las 14:30. Además se introducirá la navegación entre campos del formulario (ENTER, PREV_EDIT) y el regreso a la matriz de aplicaciones con el botón MTX.
TAREA 3	**MANEJO DE LLAMADA**
SELEC. MÍNIMAS	11 selecciones
SELEC. ÓPTIMAS	IC.LLAMAR 6 5 3 2 CLEAR MAYUS A MAYUS a ENTER
DURACIÓN MÍNIMA.	~9 min (X=15 secuencias)
MATRICES A UTILIZAR	Ambas
DESCRIPCIÓN	El usuario probará las funcionalidades de la aplicación de llamadas, asi como las opciones del teclado para borrar una letra o número (DEL), limpiar el

	campo del formulario (CLEAR), o escribir con mayúsculas (MAYUS). El contacto Aa será un contacto guardado previamente con el número del usuario, que vera recibir la llamada en su teléfono.

b) SESIÓN 2 – EVALUACIÓN DE LA APLICACIÓN

COMPROBACIÓN DEL CLASIFICADOR	
CONFIGURACIÓN	15 SECUENCIAS con MATRIZ DE NAVEGACIÓN 4x4
PROCEDIMIENTO	1 o 2 RONDAS de 6 TRIALS
DURACIÓN APROX.	~4 min o ~8 min
DESCRIPCIÓN	Puesto que ésta es la segunda sesión, es necesario comprobar el comportamiento del clasificador hallado previamente. Probablemente, las variaciones inter-sesión hayan causado un mal comportamiento del clasificador, así que se realizan 1 o 2 rondas de 6 trials, según el caso particular, y se crea un nuevo clasificador a partir del anterior con estas nuevas muestras de entrenamiento.
TAREA 1	**REALIZAR FOTOGRAFÍA**
SELEC. MÍNIMAS	3 selecciones
SELEC. ÓPTIMAS	IC.CAMARA IC.HACER_FOTO IC.VOLVER
DURACIÓN MÍNIMA	~2 min (si X = 15 secuencias)
MATRICES A UTILIZAR	Matriz de aplicaciones
DESCRIPCIÓN	El objetivo del usuario es realizar una fotografía utilizando AppRunner. Para ello se parte del estado principal de la aplicación con el servidor Bluetooth ya funcionando. Posteriormente el usuario deberá seleccionar el icono CÁMARA, que abrirá la aplicación de cámara. A continuación, seleccionará HACER_FOTO para realizar la fotografía. Para volver a la actividad principal de la aplicación seleccionará VOLVER.
TAREA 2	**VISUALIZAR LA FOTOGRAFÍA REALIZADA ANTERIORMENTE**
SELEC. MÍNIMAS	5 selecciones
SELEC. ÓPTIMAS	IC.GALERÍA IC.SIG_IMAGEN IC.ANT_IMAGEN IC.ROTAR_FOTO IC.VOLVER
DURACIÓN MÍNIMA	~4 min (si X = 15 secuencias)
MATRICES A UTILIZAR	Matriz de aplicaciones
DESCRIPCIÓN	Partiendo de la actividad principal, el usuario seleccionará la aplicación galería, donde visualizará la fotografía realizada anteriormente. Para ello, pasará a través de dos fotografías precargadas con el botón siguiente imagen. Además, para una mejor visualización (Se tomará la foto von el móvil en vertical, por lo que la imagen aparecerá girada), el usuario seleccionará el icono de rotar fotografía. A continuación seleccionará el icono volver para regresar a la actividad principal.
TAREA 3	**AÑADIR UN CONTACTO**

SELEC. MÍNIMAS	17 selecciones
SELEC. ÓPTIMAS	IC.AÑADIR_CONTACTO P E P E ENTER 6 6 6 6 6 6 6 6 6 ENTER ENTER
DURACIÓN MÍNIMA	~13 min (si X = 15 secuencias)
MATRICES A UTILIZAR	Ambas
DESCRIPCIÓN	Partiendo de la actividad principal, el usuario seleccionará la aplicación agregar contacto. Se abrirá un formulario donde introducirá el nombre del contacto y un número de teléfono. Con objeto de homogeneizar la tarea se ha determinado que el nombre de contacto sea Pepe, y el número de teléfono el del propio del usuario. Para volver a la matriz de aplicaciones se selecciona MTX.
TAREA 4	**MANEJO DE LLAMADA**
SELEC. MÍNIMAS	6 selecciones
SELEC. ÓPTIMAS	IC.LLAMAR P E P E Enter
DURACIÓN MÍNIMA.	~6 min (X=15 secuencias)
MATRICES A UTILIZAR	Ambas
DESCRIPCIÓN	El usuario probará las funcionalidades de la aplicación de llamadas con el contacto añadido durante la sesión en la tarea anterior (deberá tener un teléfono valido de una persona presente en el laboratorio), que vera recibir la llamada en su teléfono.
TAREA 5	**MANDAR SMS**
SELEC. MÍNIMAS	17 selecciones
SELEC. ÓPTIMAS	IC.SMS P E P E ENTER H O L A SPACE M U N D O ENTER
DURACIÓN MÍNIMA	~12 min (si X = 15 secuencias)
MATRICES A UTILIZAR	Ambas
DESCRIPCIÓN	Partiendo de la actividad principal, el usuario selecciona el icono SMS. Se abrirá en el móvil un formulario para introducir el contacto o número de teléfono y el texto a enviar. Cuando estén rellenados satisfactoriamente, se enviara el SMS pulsando ENTER y volverá a la matriz de selecciones pulsando MTX.

CUESTIONARIO DE SATISFACCIÓN	
DESCRIPCIÓN	Al finalizar la sesión de evaluación, se hará entrega al usuario de un cuestionario que recoja la satisfacción, el confort, la fatiga y/o las sugerencias de mejora de cada participante. Para diseñarlo, se pueden tener en cuenta las recomendaciones de la norma ISO 9241-9 (requirements for non-keyboard input).

C.2. CUESTIONARIO DE SATISFACCIÓN

CUESTIONARIO DE SATISFACCIÓN DE LA APLICACIÓN AppRunner

Valore las siguientes afirmaciones sobre la aplicación BCI "AppRunner":

16. Me ha resultado interesante conocer y usar esta aplicación BCI.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

17. La aplicación me ha resultado difícil de usar.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

18. Se han cumplido mis expectativas respecto a la funcionalidad de la aplicación.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

19. Las sesiones de pruebas me han parecido aburridas.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

20. Me imagino utilizando esta aplicación en mi vida diaria.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

21. Las sesiones requieren demasiada concentración, lo que me ha terminado cansando.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

22. La aplicación ofrece unas funcionalidades suficientes para cubrir mi utilización diaria de un dispositivo móvil.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

23. Me ha resultado difícil seleccionar los comandos deseados.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

24. La navegación por la aplicación es sencilla e intuitiva.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

25. La duración de las sesiones me ha parecido excesiva.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

26. La aplicación responde con suficiente velocidad y fluidez.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

27. Creo que el manejo de las funcionalidades implementadas es complicado y podría simplificarse

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

28. Me encantaría participar de nuevo en un estudio de estas características.

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

29. Los efectos visuales de la matriz de selección me han causado fatiga

1	2	3	4	5	6	7
Totalmente en desacuerdo			Ni mucho/ ni poco			Totalmente de acuerdo

30. ¿Qué aspectos de la aplicación mejoraría y por qué?

ANEXO D

PLIEGO DE CONDICIONES

Este anexo contiene las condiciones legales que guiarán a la realización del trabajo de fin de grado titulado como “Diseño y desarrollo de una aplicación para controlar un teléfono móvil mediante sistemas Brain Computer Interface (BCI) orientada a personas con grave discapacidad”.

En lo que sigue se supondrá que el trabajo ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de utilizar el sistema BCI. Dicha empresa ha debido desarrollar una línea de investigación con el objeto de llevar a cabo el trabajo. Esta línea de investigación, junto con el posterior desarrollo de la aplicación, está amparada por las condiciones particulares del siguiente pliego. Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes condiciones:

Condición 1

La modalidad de contratación será el concurso. La adjudicación se hará por tanto a la proposición más favorable, sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho de declararlo desierto.

Condición 2

El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

Condición 3

En la oferta se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

Condición 4

La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos que sea preciso para el desarrollo de la misma. ANEXO D 142

Condición 5

Aparte del Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no está obligado a aceptarla.

Condición 6

El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

Condición 7

Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le haya comunicado por escrito el Ingeniero

Director de obras siempre que dicha obra se haya ejecutado a los preceptos de los pliegos de condiciones, de acuerdo a los cuales se harán modificaciones y la valoración de las diversas unidades, sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto no podrán servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

Condición 8

Tanto en las certificaciones de la obra como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

Condición 9

Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata, pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero considere justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

Condición 10

Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere, y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento se sujetarán siempre al establecido en el punto anterior.

Condición 11

Cuando el contratista, con la autorización del Ingeniero Director de obras, emplee material de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio, ejecute con mayores dimensiones cualquier otra parte de las obras, o en PLIEGO DE CONDICIONES 143 general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sino a lo que le correspondería si se hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

Condición 12

Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ella se formen, o en su defecto, por lo que resulte de su medición final.

Condición 13

El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras, así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

Condición 14

Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

Condición 15

La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

Condición 16

La forma de pago será por certificaciones mensuales de la obra ejecutada de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

Condición 17

La fecha de comienzo de las obras será a partir de los quince días naturales del replanteo oficial de las mismas, y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna a la reclamación de la fianza.

Condición 18

Si el contratista, al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

Condición 19

El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras o con el delegado que este designe, para todo lo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

Condición 20

Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista la conservación de la obra ejecutada hasta la recepción de la misma, por lo que el deterioro total o parcial de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o construido por su cuenta.

Condición 21

El contratista deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa por retraso en la ejecución siempre que este no sea debido a causas de fuerza mayor. A la terminación de la obra se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

Condición 22

Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas establecidas.

Condición 23

Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución por contrata", y anteriormente llamado "Presupuesto de Ejecución Material" que hoy designa otro concepto. La empresa constructora, que ha desarrollado este proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares.

Condición particular 1

La propiedad intelectual de los procesos descritos y analizados en el presente trabajo pertenece por entero a la empresa constructora representada por el Ingeniero Director del Proyecto.

Condición particular 2

La empresa constructora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos posteriores, para la misma empresa cliente o para otra.

Condición particular 3

Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con la autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

Condición particular 4

En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

Condición particular 5

En todas las reproducciones se indicará su procedencia, explicando el nombre del proyecto, nombre del Ingeniero Director y empresa consultora.

Condición particular 6

Si el proyecto pasa a la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de este, la empresa consultora decidirá aceptar o no la modificación propuesta.

Condición particular 7

Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta al añadirla.

Condición particular 8

Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

Condición particular 9

Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

Condición particular 10

La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

Condición particular 11

La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia de este hecho. En este caso deberá autorizar expresamente los proyectos presentados por otros.

Condición particular 12

El Ingeniero Director del proyecto será responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona asignada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.

Los requisitos, clasificados como documentación, hardware y software, que han sido necesarios para la elaboración del presente trabajo fin de grado son los siguientes.

DOCUMENTACIÓN

- Servicio de la Biblioteca de la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universidad de Valladolid.

HARDWARE

- Equipo Intel® Core™ 2 Duo – CPU E4500, con 2.2GHz, 32 bits y 1GB de RAM.
- Smartphone Android 5.0 o superior con 1.5Ghz y 1.5 GB de RAM
- Equipo BCI: Amplificador g.USBamp de 16 canales. Gorro EEG de 64 canales.
- Electrodos EEG de oro.
- Electrodos de referencia. Gel conductor.

ANEXO D

SOFTWARE

- Microsoft® Windows 7 Professional (SP1). Copyright© 2009, Microsoft Corporation.
- Microsoft® Word (14.0.7116.5000). Microsoft® Office Home and Student 2010. Copyright© 2010, Microsoft Corporation.
- MATLAB® Student R2012a (7.14.0.739). Copyright© 2012, The MathWorks Inc.
- Adobe Photoshop CS5 Extended (Versión 12.0). Copyright© 1990–2010, Adobe Systems Incorporated.
- Adobe® Reader® XI (Versión 11.0.07) Copyright© 1984–2012, Adobe Systems Incorporated.
- Google Chrome (Versión 35.0.1916.114m). Copyright© 2014, Google Inc.
- Microsoft® Visual Studio Professional (Versión 10.0.30319.1). Copyright© 2010, Microsoft Corporation.
- Android Studio (Versión 2.1) Copyright© 2016, Google Inc.
- CMake (2.8.12.1) con Qt (4.6.2). Creative Commons (CC), Kitware Inc.
- g.USBamp Simulink High-Speed Online Processing.
- Plataforma de propósito general BCI2000.

ANEXO E

PRESUPUESTO

El presupuesto estimado de acuerdo a las condiciones expuestas en el anexo de pliego de condiciones es el siguiente.

EJECUCIÓN MATERIAL

Equipo Intel® Core™ 2 Duo – CPU E4500, 2.2GHz, 32 bits y 1GB de RAM 700€
Equipo BCI 13.300€
Sistema Operativo: Microsoft® Windows 7 Professional (SP1) 110€
Software: Microsoft® Office Home and Student 2010 80€
Software: MATLAB® Student R2012a 427€
Software: Adobe Photoshop CS5 Extended 80€
TOTAL 14.697€

GASTOS GENERALES

16% sobe la ejecución material 2.351,52€

BENEFICIO INDUSTRIAL

6% sobe la ejecución material 881,82€

MATERIAL FUNGIBLE

Gastos de Impresión (tinta y papel) 200€
Encuadernación 120€
DVDs 2€
TOTAL 322€

HONORARIOS DEL PROYECTO

450 horas a 30€/hora 13.500€

SUBTOTAL DEL PRESUPUESTO

Subtotal del presupuesto 31.752,34€

I.V.A. APLICABLE

21% sobre el subtotal de presupuesto 6.667,99€

TOTAL DEL PRESUPUESTO

Total del presupuesto 38.420,33€

El total del presupuesto, en euros, asciende a:

TREINTA Y OCHO MIL CUATROCIENTOS VEINTE EUROS CON TREINTA Y TRES CÉNTIMOS DE EURO.

En Valladolid, Julio de 2016.

Fdo. Eduardo Santamaría Vazquez.