Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

LINI
laboratorio de neuroimagenología

Desarrollo de un prototipo inalámbrico de una interfaz cerebro-computadora en un dispositivo móvil

Tesis que presenta
Daniel Martínez Aguilar

Para obtener el grado de
Maestro en ciencias en ingeniería biomédica

Asesor: M. en I. Oscar Yánez Suárez

Jurado calificador:

Presidente: Juan Carlos Echeverría Arjonilla
Secretario: Oscar Yánez Suárez
Vocal: Jose Francisco Rodríguez Arellano

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

LINI
laboratorio de neuroimagenología

Desarrollo de un prototipo inalámbrico de una interfaz cerebro-computadora en un dispositivo móvil

Tesis que presenta

Daniel Martínez Aguilar

Para obtener el grado de

Maestro en ciencias en ingeniería biomédica

Asesor: M. en I. Oscar Yánez Suárez

Jurado calificador:

Presidente: Juan Carlos Echeverría Arjonilla
Secretario: Oscar Yánez Suárez
Vocal: Jose Francisco Rodríguez Arellano

Agradecimientos

Al **CONACyT** por todo el apoyo económico y por impulsar y fortalecer el desarrollo científico y tecnológico en México.

A la Universidad Autónoma Metropolitana por ser una institución tan noble y cuna del conocimiento.

Al Laboratorio de Investigación en Neuroimagenología y a todos sus integrantes por ser un excelente grupo de trabajo.

Miembros del jurado

Presidente - Dr. Juan Carlos Echeverría Arjonilla

Secretario - Oscar Yánez Suárez

Vocal - Jose Francisco Rodríguez Arellano

Agradecimientos personales

A Pao, mi compañera de vida, por su apoyo incondicional en todo momento.

A mis papás, Martha y Miguel, por enseñarme todo en esta vida. Sin ellos, esta tesis no hubiera sido posible.

A mi asesor, Oscar Yáñez, por su apoyo, su enorme aporte de conocimientos y financiamiento para la realización del proyecto.

A mi hermana Pame, por aportar una visión distinta a la forma de ver las cosas.

A la familia Ayala Bribiesca, por acogerme en esos momentos de desvelo.

A mi tía Martha, Maria Eugenia y mi tío Sergio, por su gran apoyo.

A mis padrinos, Georgina y Ernesto, por ser mi guía en los momentos de duda y esos amigos incondicionales.

A las personas del CENETEC, por su apoyo y comprensión incondicional. Y en particular al Dr. Esteban Hernández San Román.

"If you can't explain it simply, you don't understand it well enough".

Albert Einstein

Resumen

El uso de las interfaces cerebro-computadora (BCI) en pacientes y personas sanas ha evolucionado considerablemente en los últimos años. En un principio, los esfuerzos estaban enfocados en validar el concepto, probando diferen tes paradigmas en personas sanas en el laboratorio y no en pacientes en los hospitales. Ahora los esfuerzos están enfocados en mejorar los resultados de estos métodos alternativos de comunicación fuera del laboratorio.El Laboratorio de Investigación en Neuroimagenología (LINI) de la Universidad Autónoma Metropolitana (UAM), ha hecho muchos esfuerzos en la mejora y desarrollo de estas interfaces cerebro computadora. Mucha de esta experiencia, y trabajos previos, pueden ser aplicados en el desarrollo de una interfaz móvil.

Dado el vertiginoso desarrollo tecnológico de los últimos años, estando enfocados los recursos de la industria en la fabricación de dispositivos móviles con capacidades de cómputo equiparables a las de una computadora personal y el desarrollo de amplificadores portátiles de Electroencefalograma (EEG) comerciales, es factible combinar los avances en varias disciplinas para lograr un objetivo común.

El objetivo de esta tesis de maestría fue combinar los avances en estas disciplinas para hacer una interfaz cerebro-computadora usando un teléfono con Android y un amplificador de EEG portátil.

Los esfuerzos realizados por diversos laboratorios dedicados al desarrollo de interfaces cerebro computadora no han podido lograr eliminar el uso de una computadora para la adquisición (y en algunos casos el procesamiento) de las señales de EEG, limitando de esta manera la movilidad y las posibles aplicaciones que se puedan lograr fuera de un laboratorio de investigación.

Se eligió el paradigma de control exógeno denominado Potenciales Provocados Visuales de Estado Estacionario (SSEVP) para el desarrollo de una interfaz cerebro-computadora, porque permite distinguir entre dos estímulos y tomar la decisión de llamar a uno de entre dos contactos almacenados en el teléfono.

Con el desarrollo de esta "Android™ BCI" se logra obtener la movilidad e independencia deseada en el comienzo del proyecto para poder utilizarla en cualquier lugar sin las restricciones de movilidad inherentes a las interfaces cerebro computadora tradicionales. Además, brinda las herramientas necesarias para el desarrollo de más interfaces cerebro computadora, debido a que se puede hacer uso de las señales de EEG disponibles en el dispositivo móvil sólo portando el código.

Abstract

In recent years, the use of Brain Computer Interfaces (BCIs) in patients and healthy people has evolved considerably. Initially, developing efforts where focused in validating the concept, testing different paradigms on healthy people in the lab and not in real patients in the hospital. Nowadays efforts in the design of BCIs are focused in improving the results of these alternative communication methods outside the lab. The Neuroimagenology Research Laboratory (LINI for its acronym in Spanish) of the Metropolitan Autonomous University (UAM for its acronym in Spanish) has mademany efforts in the improvement and the development of BCIs. This experience and previous work can be applied in the development of a mobile Brain Computer Interface.

GivGiven the rapid technological development of recent years, with industry resources focused on the manufacture of mobile devices with computing capacities comparable to those of a personal computer and the development of portable commercial electroencephalogram (EEG) amplifiers, it is feasible to combine advances in various disciplines to achieve a common goal.

The aim of this master's thesis was to combine advances in these disciplines to make a brain-computer interface using an Android phone and a portable EEG amplifier.

The efforts made by various laboratories dedicated to the development of brain-computer interfaces have not been able to eliminate the use of a computer for the acquisition (and in some cases processing) of EEG signals, thereby limiting mobility and Possible applications that can be achieved outside a research laboratory.

The exogenous control paradigm called Steady State Visual Potentials (SSEVP) was chosen for the development of this brain-computer interface because it allows to distinguish between two stimuli and to make the decision to call one of two contacts stored in the phone.

With the development of this "Android™ BCI" it is possible to obtain the desired mobility and independence aimed at the beginning of the project in order to be able to use it anywhere without the restrictions of mobility inherent of traditional computer brain interfaces. In addition, it provides the necessary tools

for the development of more brain computer interfaces, because it is possible to make use of the EEG signals available on the mobile device just porting the code.

Índice general

Índice de figuras

Índice de tablas

Secciones de código

Acrónimos

ADB	Android Debugging Bridge	Utilidad de Depuración de Android
AES	Advanced Encryption Standard	Estándar de cifrado avanzado
API	Application Programming Interface	Interfaz de programación de aplicaciones
BCI	Brain Computer Interface	Interfaz cerebro-computadora
BCIs	Brain Computer Interfaces	Interfaces cerebro-computadora
CMS	Common Mode Sense electrode	Electrodo activo modo de censado común
DRL	Driven Right Leg	Electrodo pasivo para "pierna derecha"
ECG	ElectroCardioGram	Electrocardiograma
ECoG	ElectroCorticoGram	Electrocorticograma
EEG	ElectroEncephaloGram	Electroencefalograma
EOG	ElectroOculoGram	Electrooculograma
ERD	Event Related Desynchronization	Paradigma de Desincronización (Imaginación de movimiento)
ERD/ERS	Event Related Sincronization Desynchronization	Paradigma de Sincronización Desincronización
FFT	Fast Fourier Transform	Transformada rápida de Fourier
fMRI	functional Magnetic Resonance Imaging	Imagenología por resonancia magnética funcional
fNIR	functional Near-InfraRed Sensing	Censado por infrarojo-cercano
fNIRS	functional Near-InfraRed Spectroscopy	Espectroscopía funcional por infrarrojo-cercano

GUI	Graphical User Interface	Interfaz gráfica de usuario
ICA	Independent Component Analysis	Análisis por componentes independientes
IDE	Integrated Development Environment	Entorno de Desarrollo Integrado
LSB	Least Significant Bit	bit menos significativo
MEG	MagnetoEncephaloGraphy	Magnetoencefalografia
MFi	Made For iPhone/iPad/iPod	Hecho para iPhone/iPad/iPod
PnP	Plug and Play	Conectar y usar
SCP	Slow Cortical Potentials	Potenciales corticales lentos
SDK	Software Development Kit	Kit de Desarrollo de Software
SSVEP	Steady-State Visual Evoked Potentials	Potenciales provocados visuales de estado estacionario
UI	User Interface	Interfaz de usuario
USB-IF	USB Implementers Forum	Foro de implementación USB
VEP	Visually Evoked Potential	potencial visual provocado

Para evitar traducciones incorrectas de palabras provenientes del lenguaje tecnológico y computacional algunas de ellas se conservaron en el idioma original, debido a que no existe una traducción adecuada en la Real Academia de la Lengua Española

Capítulo 1

Introducción ¿Qué es una interfaz cerebro-computadora?

1.1. Un poco de historia

La investigación en interfaces cerebro-computadora (BCI por su siglas en inglés) comenzó en los años 70 en la Universidad de Los Ángeles California (UCLA por sus siglas en inglés) con el apoyo de la National Science Foundation. Se continuó posteriormente mediante un contrato con la Agencia de Proyectos de Investigación Avanzada del Departamento de Defensa de los Estados Unidos. En los documentos generados por esa asociación, se utiliza por primera vez el término interfaz cerebro-computadora; desde entonces, el campo de las interfaces cerebro-computadora ha avanzado considerablemente [4].

Los avances en las neurociencias cognitivas y las técnicas de neuroimagenología han provisto a los neurocientíficos de habilidades para identificar estructura y función del cerebro humano, examinar la manera en la que ambas varían en función de las diferencias genéticas, de personalidad, emoción y cognición. Esta habilidad se ha hecho posible a través del uso de sensores que pueden monitorear algunos de los procesos fisiológicos que ocurren en el cerebro y que corresponden a ciertas formas de señales relacionadas con tareas cognitivas [5]. Algunos autores refieren estos avances en las técnicas de neuroimagenología como habilidades que han brindado a los neurocientíficos la capacidad de "ver" el cerebro desde un punto de vista funcional, al saber cuáles son las zonas que se irrigan al hacer ciertas tareas específicas,

con ayuda de la resonancia magnética funcional o la medición de la actividad cerebral asociando un compuesto radioactivo a una molécula de glucosa (fluorodeoxyglucose "FDG") en una tomografía por emisión de positrones (PET), lo que ha permitido a los neurocientíficos crear mapas topográficos del cerebro de zonas relacionadas con ciertas tareas cognitivas con mucha precisión. [5] Las interfaces cerebro-computadora son ampliamente consideradas como la aplicación más exitosa de las neurociencias [6] y su investigación ha llamado la atención de múltiples disciplinas.

1.1.1. Un poco de anatomía y fisiología

De forma contraria a las simplificaciones populares, el cerebro no es una computadora de propósito general con una unidad central de procesamiento. Es, por otro lado, un ensamble de subsistemas altamente especializados para tareas precisas, que están en constante competencia [5]. Sin importar de que parte del cerebro se hable, cada una de ellas está formada por células nerviosas llamadas neuronas (aproximadamente 100 000 millones) y cada una de ellas se comunica con miles de otras neuronas en un proceso fisiológico autorregulado que genera el pensamiento. Hay dos tipos de comunicación neuronal: a través de señales eléctricas mediante conexiones físicas o mediante el intercambio químico usando neurotransmisores.

A grandes rasgos el cerebro puede ser dividido en dos partes principales, la corteza cerebral y regiones subcorticales. Las regiones subcorticales son filogenéticamente más antiguas e incluyen áreas asociadas con el control de funciones básicas y funciones vitales como la respiración, la frecuencia cardíaca, la regulación de la temperatura, emociones básicas, y respuestas instintivas como el miedo, el aprendizaje y la memoria. La corteza cerebral, hablando en términos de evolución es más reciente, siendo ésta la parte más grande y más compleja del cerebro humano.[7] En la corteza se lleva a cabo la mayor parte del procesamiento sensorial y motor, así como funciones de alto nivel incluyendo al razonamiento, la planeación, el procesamiento del lenguaje y el reconocimiento de patrones; ésta es la región en la que la mayoría de las interfaces cerebro-computadora se enfocan en la actualidad.

El cerebro, y en particular la corteza cerebral, se puede estudiar desde diversos puntos de vista, una forma general de clasificarla divide la corteza cerebral en dos hemisferios, que generalmente tienen funciones muy distintas. Por ejemplo, la mayor parte de las funciones del lenguaje están ubicadas en el hemisferio izquierdo, mientras que el hemisferio derecho controla muchas tareas del razonamiento y ubicación espacial[7]. La mayoría de las señales motoras y sensoriales del cuerpo terminan en hemisferios cruzados, es decir, el hemisferio derecho se vincula y controla el lado izquierdo del cuerpo y viceversa.

Otra forma de estudio, divide a la corteza por regiones (o lóbulos) independientes, especializadas para diferentes funciones (Ver la figura 1.1 en la página 4). [7]

- Lóbulo occipital:
  Ubicado en la parte posterior del cerebro, está dedicado en gran parte al procesamiento de la información visual (sección de color rosa en la figura 1.1 en la página 4).
- Lóbulo temporal:
  Ubicado a los lados y en la parte baja del cerebro, tiene funciones relacionadas con la memoria, reconocimiento de patrones, el procesamiento del lenguaje y la audición (sección de color verde en la figura 1.1 en la página 4).
- Lóbulo parietal:
  Ubicado en la parte superior del cerebro, en esta región de la corteza cerebral es en donde se llevan a cabo las funciones relacionadas con el pensamiento y la realización de cálculos matemáticos. Esta región está encargada de recibir las sensaciones de tacto, calor, frío, presión, dolor, y coordinar el equilibrio (sección de color amarillo en la figura 1.1 en la página 4).
- Lóbulo frontal:
  Ubicado en la parte anterior del cerebro, es considerada la región filogenéticamente más avanzada[7] en donde se llevan a cabo tareas que requieren del uso de la concentración, la producción lingüística y oral (sección de color azul en la figura 1.1 en la página 4).

En el proceso histórico de estudio del cerebro, los primeros acercamientos fueron de origen anatómico y posteriormente fisiológico, tratando de asociar el funcionamiento a regiones específicas del cerebro, pero es importante mencionar que el entendimiento de la estructura de la corteza cerebral y las funciones asociadas por regiones es un tema aún en desarrollo y no es conocimiento comprobado. Estos mapas topográficos no son asignaciones definitivas de la relación localización-función. De hecho, algunas regiones procesan múltiples funciones y muchas funciones son procesadas en más de una área. [8]

1.1.2. El origen del EEG

Cuando no existía una ciencia que estudiara el cerebro, el pensamiento era (y sigue siendo) una incógnita. Ahora parece obvio que los primeros acercamientos de la neuroimagenología asumían que los principios de la comunicación neuronal necesitan oxígeno y glucosa, requiriendo un incremento en la irrigación sanguínea en las regiones activas del cerebro. Esta

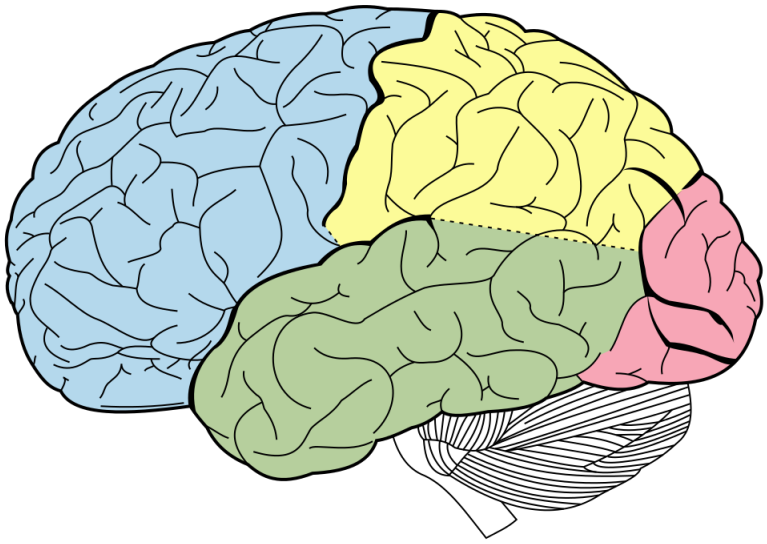


Figura 1.1: Lóbulos de la corteza cerebral

hipótesis dio origen a técnicas para profundizar en las observaciones eléctricas, químicas y de flujo sanguíneo y generar relaciones con la actividad cerebral en el procesamiento de información y la respuesta a todo tipo de estímulos ,y de esta manera, inferir procesos cognitivos y respuestas fisiológicas [5].

El EEG representa una diferencia de potencial eléctrico, generalmente en varios puntos del cuero cabelludo. El potencial eléctrico es generado como resultado de la actividad neuronal que ocurre en un arreglo distribuido de fuentes de corriente en un volumen conductor (la cabeza). [8] Estos potenciales provienen de la comunicación entre un gran número de neuronas que han generado potenciales de acción, en la forma de cambios de estado eléctrico y producidos a gran escala por un proceso de intercambio de neurotransmisores llamado sinapsis. Esta sincronización de un gran número de neuronas en la corteza cerebral da origen a un *potencial de campo local* que genera una diferencia de potencial que es posible medir con amplificadores ubicados en el cuero cabelludo, y las señales adquiridas son producto de una gran mezcla de señales generadas por fuentes distribuidas en toda la cabeza. Esa es una de las razones por las que es tan difícil su interpretación. La reconstrucción de la posición y la geometría de estas fuentes de información son problemas complejos, definido como el problema inverso, y es un reto en las neurociencias en la actualidad[8].
A pesar de que la información de la posición y geometría de las fuentes no está resuelto, el EEG conserva información temporal precisa que está relacionada con patrones de activación

en el tiempo [9].

1.1.3. Tipos de amplificadores de EEG

El EEG es sólo una forma de estimar la actividad neuronal que ocurre en la corteza cerebral, existen también la magnetoencefalografía (MEG), la tomografía por emisión de positrones (PET), la resonancia magnética funcional (fMRI) y la espectroscopia por infrarrojos cercanos (fNIRS) entre otras. Sin embargo, de esta gama de opciones la más económica y fácil de adquirir es el EEG. [8]

Para poder adquirir el EEG del cuero cabelludo existen tres tipos de amplificadores:

1. Los amplificadores de grado médico

   Son amplificadores dedicados al diagnóstico de enfermedades o padecimientos, tienen una buena calidad de señal, generalmente no se pueden modificar parámetros relacionados con el procesamiento de las señales porque están pre configurados para que el neurólogo haga la interpretación.

2. Los amplificadores de investigación

   Son amplificadores especializados que permiten configurar las variables relacionadas a la adquisición y el procesamiento. Permiten configuraciones de adquisición de hasta 128 canales de manera simultánea, alta impedancia, gran amplificación y de alta resolución.

3. Los amplificadores comerciales

   Recientemente desarrollados, permiten la adquisición de señales de EEG con menos precisión que los dos anteriores. Por otro lado, son amplificadores de bajo costo con una relación señal a ruido menor en pro de la portabilidad y la facilidad de uso.

1.2. ¿Qué es una interfaz cerebro-computadora?

No existe una definición formal del término interfaz cerebro-computadora. Esta tesis hace una recopilación de cuatro definiciones escritas por autores en artículos de investigación en el tema:

> "Una interfaz cerebro-computadora es un sistema de control y de comunicación no muscular, una vía directa para entregar mensajes y comandos al mundo externo" [10].

> "Una interfaz cerebro-computadora es un sistema de comunicación en el que los mensajes y comandos que una persona envía al mundo exterior no pasan a través de las vías normales de comunicación ni tampoco a través de los nervios periféricos y músculos. Este sistema infiere la intención del usuario a través de la medición directa de la actividad cerebral" [11].

> "Traducir pensamientos en acciones sin involucrar un acto físico es ahora posible gracias a los recientes desarrollos en interfaces cerebro-computadora" [12].

> "Las interfaces cerebro-computadora brindan un canal de comunicación adicional el cual funciona, usando la actividad neuronal para controlar dispositivos adicionales; por ejemplo, para restaurar la función motriz de alguna extremidad del cuerpo. La actividad cerebral de unas cuantas neuronas o de un gran conjunto de neuronas es muestreada, procesada y convertida en comandos de control para una aplicación (e.g. un brazo robótico o un programa de comunicación)" [12].

Esta última definición de interfaz cerebro-computadora se ajusta muy bien al objetivo de esta tesis, que es la elaboración de una interfaz cerebro-computadora móvil que pretende acercar el uso de este tipo de vías de comunicación a la vida cotidiana, ya que en los últimos años el uso de interfaces cerebro-computadora ha estado atado a los laboratorios, a un tiempo largo de preparación de los pacientes y al uso de equipamiento especializado, lo que ha limitado su uso a los laboratorios en el mundo en donde son desarrolladas [13].

1.2.1. ¿Para qué son usadas las interfaces cerebro-computadora?

Generalmente, las interfaces cerebro-computadora son utilizadas para asistir, aumentar o reparar las funciones cognitivas, sensoriales o motoras del ser humano, usando canales alternativos de comunicación, es decir, canales eferentes que no hacen uso de los nervios periféricos y los músculos [10]; e.g. escribir una palabra sin usar un teclado o hacer una llamada telefónica a un contacto sin pulsar la pantalla del teléfono celular, construyendo un canal de comunicación alternativo entre el cerebro y el mundo, mediante el uso de una computadora.

1.2.2. Tipos de BCIs

La división más general se hace en dos grandes grupos, interfaces cerebro computadora invasivas y no invasivas.

1.2.2.1. BCIs invasivas

Las BCIs invasivas usan la técnica de electrocorticografía (ECoG) que consiste en colocar electrodos en contacto directo con la corteza cerebral para registrar la actividad muy localizada de un grupo de neuronas, o bien, colocando electrodos conectados a una sola neurona para medir el potencial de acción ("single response multiunit activity"). Este tipo de técnicas (ECoG) tienen numerosas ventajas: proveen una gran resolución temporal y espacial (a pesar de que se hacen solamente en regiones pequeñas en comparación con los métodos no invasivos, e.g. EEG), el hecho de **no** ser necesaria la colocación de electrodos en cada sesión de uso, el ancho de banda de las señales es mayor, la relación señal a ruido se incrementa y, se puede centrar mejor atención en las regiones de interés al colocar directamente los electrodos en ellas tras conocer a priori su ubicación espacial [12].

Por otro lado, hay que contrastar estas ventajas con el riesgo de la cirugía y el potencial desarrollo de complicaciones como infecciones por el hecho de tener la corteza cerebral expuesta al medio externo [14] en la colocación de los electrodos.

1.2.2.2. BCIs no invasivas

Las BCIs no invasivas sitúan los electrodos en el cuero cabelludo para registrar el electroencefalograma (EEG),o algún otro transductor que convierta las señales eléctricas de la corteza cerebral en otro tipo de señal, para extraer información relacionada con la actividad de la corteza cerebral. Aunque en este tipo de aproximaciones las señales son más débiles que las obtenidas por contacto directo con la corteza cerebral, éstas son más usadas debido a que son un procedimiento no invasivo y que no pone en riesgo la salud del usuario.

Existen tres formas no invasivas más comunes de obtener señales de la actividad cortical. La primera, que ya se comentó, usando el EEG, es la más usada[8]; otra, es usando magnetoencefalografía, provee de señales más acertadas de la actividad cerebral, pero utiliza equipamiento muy avanzado y requiere un cuarto con aislamiento magnético y magnetos superconductores enfriados por helio, lo que dificulta su utilización en aplicaciones cotidianas; y la tercera, es usando espectroscopia funcional por infrarrojos cercanos (fNIRS) que correlacionan la actividad neuronal con una compleja respuesta a la actividad hemodinámica de

los vasos sanguíneos del cerebro y los agentes químicos involucrados en ella. Algunas de las ventajas que tiene la fNIRS es que la luz infrarroja sólo penetra a una profundidad en la que se registra la actividad cortical (eliminando otras fuentes más profundas que no son de interés en este tipo de aplicaciones en particular), además de que también es portátil.

1.2.3. Elementos básicos de una BCI

Los componentes de una BCI genérica descritos por [11] son cuatro (ver figura 1.2 en la página 9, integrados por:

1. La adquisición de la señal es el principal componente de una BCI. Hay muchas formas de adquirir las diversas señales que reflejan la actividad en la corteza cerebral (no sólo pueden ser señales eléctricas) como se vio en la sección anterior.

2. El módulo de procesamiento de las señales es el segundo elemento que conforma una BCI. Este módulo es en donde existe el mayor reto, ya que en éste es en donde se 'interpretan" las intenciones del usuario.

3. El tercero es el dispositivo de salida, que emite mensajes o comandos de control generados por el algoritmo que interpreta las señales. La salida más común de una interfaz cerebro-computadora es un monitor, aunque se han desarrollado otro tipo de salidas como brazos robóticos, robots móviles, controles para aparatos electrodomésticos y estimuladores eléctricos funcionales.

4. El cuarto componerte es el "paradigma" de operación, el cual determina cómo interactuarán la interfaz cerebro-computadora y el usuario. Generalmente el dispositivo de salida es hardware y el protocolo de operación es software que indica como el usuario puede modificar la salida en un sistema que retroalimenta al usuario.

1.3. Paradigmas de operación de una interfaz cerebro-computadora

Una tarea de control consiste en un esfuerzo mental realizado por el usuario de una interfaz cerebro-computadora para producir de manera voluntaria un cambio en alguna señal del cerebro. Hay diversos paradigmas de control, como pueden ser: imaginación de movimiento,

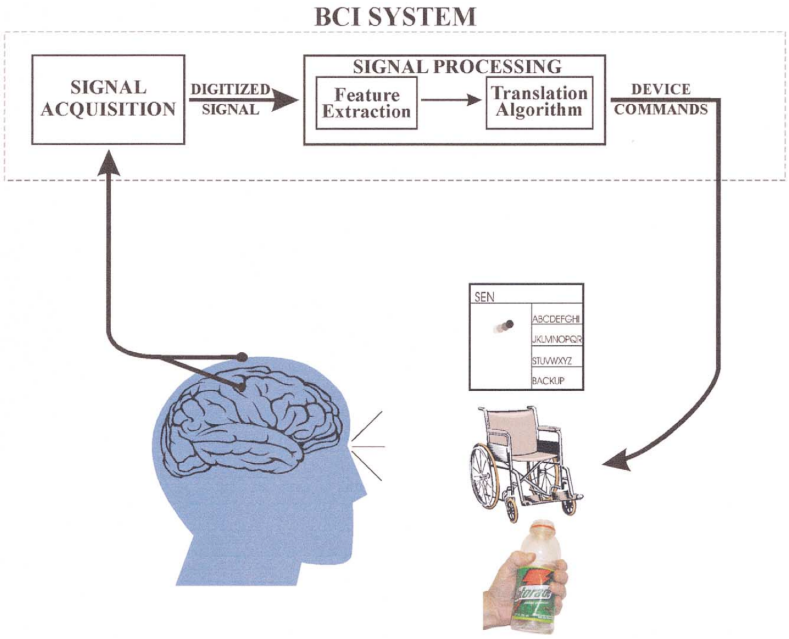


Figura 1.2: Elementos básicos de una BCI

Fuente: Imagen adaptada de Wolpaw 2002 [10]

visualización de objetos, prestar atención, imaginación de música, contar o realizar alguna operación matemática, y algún pensamiento relajante, entre otros.

Pueden ser divididos de manera general en dos grupos [5]:

1. Interfaces cerebro-computadora con paradigma de control endógeno o independientes
2. Interfaces cerebro-computadora con paradigma de control exógeno o dependientes

1.3.1. Interfaces cerebro-computadora con paradigma de control endógeno o independientes

En este tipo de interfaz cerebro-computadora, el usuario a través de la realización de una "tarea mental" voluntaria tal como imaginar movimiento o una operación matemática imaginada, genera cambios en la actividad de la corteza cerebral en una región particular que son detectados por la interfaz cerebro-computadora y clasificados para la toma de decisiones

[5]. No dependen de un estímulo externo, sino que el usuario es el que tiene el control de la interfaz cerebro-computadora. A pesar de ser llamados paradigmas endógenos, estos hacen uso de la retroalimentación del sistema para reforzar la respuesta y así, mejorar el desempeño de la interfaz cerebro-computadora. Dos ejemplos de este tipo de interfaces cerebro-computadora hacen uso de los potenciales corticales lentos y los ritmos "mu" que se explican a detalle a continuación.

1.3.1.1. Potenciales corticales lentos

También llamados Slow Cortical Potentials (o SCP por sus siglas en inglés), es un tipo de interfaz cerebro-computadora que depende del condicionamiento del usuario para cambiar la polaridad (positiva o negativa) de sus potenciales corticales lentos [15]. Son cambios en las características de la señal de EEG que se producen voluntariamente. Los cambios que se presentan en este paradigma son cambios en el nivel promedio de la amplitud EEG y como su nombre lo indica, estos cambios ocurren lentamente y requieren de un alto nivel de concentración para lograr su control. Una desventaja de este paradigma es que solo un tercio de los sujetos pueden aprender a controlar los SCP [16].

1.3.1.2. Ritmos "mu" (o imaginación de movimiento)

También son llamados mu-rhythm response, mu-based systems o Event Related Desynchronization/synchronization (ERD/ERS por sus siglas en inglés). Actúan registrando el movimiento real o imaginado reflejado en la corteza cerebral; miden la amplitud de la señal en la banda mu como mecanismo de control de la interfaz cerebro-computadora [17]. Se hace uso del hecho de que: toda actividad se mapea al cerebro, principalmente a la corteza; y de los cambios en frecuencia para tareas específicas. En particular se observa el fenómeno de sincronía/desincronía (ERS/ERD) que ocurre en la corteza motora al imaginar el movimiento de alguna extremidad. De la misma forma puede realizarse a voluntad y para sujetos sanos requiere una fase de entrenamiento en la cual se aprende a imaginar la realización de un movimiento sin realmente ejecutarlo [16].

1.3.2. Interfaces cerebro-computadora con paradigma de control exógeno o dependientes

El paradigma de control exógeno es también llamado control de respuesta provocada porque es necesario un estímulo externo para producir que la actividad en la corteza cerebral

cambie (respuesta autónoma) y que ésta sea detectada por la interfaz cerebro-computadora. En este tipo de paradigma, el desempeño se ve afectado por el grado de atención incesante que el usuario debe prestar al estímulo externo [5]. En el presente trabajo, se desarrolla a detalle la teoría de este tipo de interfaz cerebro-computadora porque que es el tipo de paradigma que se usará más adelante en el diseño. Dos ejemplos de interfaces cerebro-computadora dependientes son las interfaces cerebro computadora que generan potenciales provocados visuales de estado estacionario mediante estímulos visuales y las interfaces cerebro-computadora que generan potenciales provocados P300 mediante la presentación de un evento extraño. [9]

1.3.2.1. Potenciales provocados visuales de estado estacionario (PEVEE)

Este tipo de estímulos son un caso particular del paradigma de potenciales provocados llamados Steady State Visual Evoked Potencials (SSVEP por sus siglas en inglés o PP) ya que hace uso de estímulos visuales. Este es el paradigma que se explica con más detalle en esta tesis ya que será el paradigma a usar en la interfaz cerebro-computadora. *Este paradigma consiste en generar y presentar al usuario un estímulo visual (de entre al menos uno o más, para poder discernir entre ellos) que oscila a una cierta frecuencia fija, el cual genera una respuesta autónoma en la parte occipital de la corteza cerebral, que corresponde a la corteza visual, con el simple hecho de observar el estímulo visual presentado.*

Son dos los tipos más comunes de generar los estímulos visuales [18] que son:

1. Diodos emisores de luz (LED)

   Éstos consisten en un hardware dedicado que enciende y apaga un arreglo de N (al menos dos) LED a una frecuencia fija distinta, procurando que los armónicos de estas frecuencias se traslapen lo menos posible en el dominio de la frecuencia. Esta solución requiere de un dispositivo adicional además del dispositivo móvil, por lo que fue descartado de origen en el proyecto y porque reduce la portabilidad de la implementación. Es decir, si alguna persona quisiera implementar esta interfaz cerebro-computadora requeriría de este dispositivo adicional; sin embargo, es la mejor solución para generar potenciales visuales de estado estacionario porque el control de las frecuencias de estimulación se puede hacer con gran precisión.

2. Pantalla de cristal líquido (LCD)

   Esta solución es implementada y configurada mediante la programación de los estimuladores, alternando un estado de encendido (color blanco en la pantalla) y un estado apagado (color negro en la pantalla) en algún dispositivo electrónico programable al

que se le pueda conectar una pantalla LCD. Por no necesitar el diseño del arreglo de LED y ser mucho más portable, por tratarse de un programa de cómputo, ésta es la solución que se adoptó en este proyecto.
Uno de los problemas de esta solución es que las frecuencias disponibles a ser usadas como estímulos visuales están limitadas por la tasa de actualización de la pantalla, generando un incremento en el número de errores de despliegue debido a dos principales causas [18]:

a) La interfaz gráfica del dispositivo es afectada por la carga general del sistema.

b) La tasa de actualización de la pantalla genera una combinación de fases.

Los algoritmos para la detección de características de los SSVEP generalmente hacen uso de los armónicos de las frecuencias de estimulación, y la tasa de actualización de la pantalla limita la selección de las frecuencias fundamentales.

Los SSVEP son potenciales cuasi-senoidales que se generan en la corteza visual al presentar un estímulo que oscila. Por el simple hecho de observar a estos estímulos visuales de frecuencia fija se genera una respuesta autónoma en la que las neuronas de la corteza visual ya que éstas se sincronizan y empiezan a oscilar a la misma frecuencia que la del estímulo visual al que el usuario está prestando atención, y generalmente también genera una respuesta a uno o más armónicos de la frecuencia fundamental [19]. Este fenómeno fisiológico se puede observar en la figura 1.3 en la página 13 en la que 300 ms después de la presentación del estímulo se observa como las neuronas se sincronizan a la frecuencia del estímulo (15 Hz en este caso).
En una interfaz cerebro-computadora con paradigma SSVEP, el EEG crudo es traducido en potencia en las diferentes bandas de las frecuencias conocidas de estimulación o el cálculo de un espectro amplio en donde los picos resultantes a frecuencias específicas son usados para determinar cuál es el estímulo visual al que el usuario está prestando atención.

1.3.2.2. Potenciales provocados P300

En este tipo de interfaz cerebro-computadora se hace uso de la respuesta autónoma generada en la región parietal del cerebro 300 milisegundos después de la presentación de un estímulo visual o auditivo inesperado (e.g. la intensificación de una letra en una matriz de letras). La activación de la respuesta P300 depende de la atención del usuario a una letra en particular en una matriz de letras presentadas. Esta matriz de estimulación fue presentada por Farwell y Donchin en 1988 [7] y es la que comúnmente se usa en los experimentos hasta la fecha.

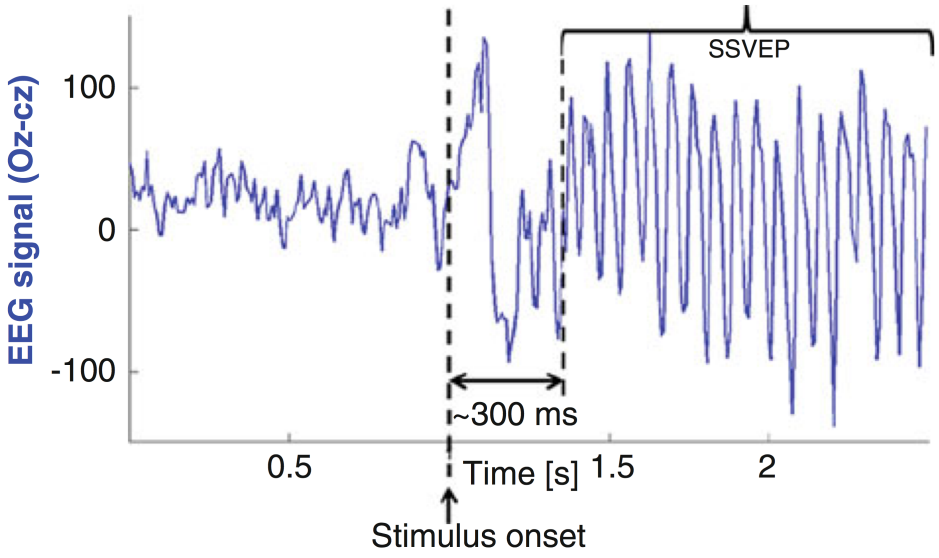


Figura 1.3: Sincronización de las neuronas de la corteza visual

Fuente: [9]

1.4. ¿Cualquier sujeto puede usar una BCI?

Una BCI genérica que funcione para cualquier sujeto no ha sido desarrollada. Por el contrario, cerca del 20 % de los usuarios no han sido capaces de usar una interfaz cerebro-computadora típica. [5] Algunas posibles soluciones han sido exploradas, como mejorar el procesamiento de las señales, el entrenamiento, y nuevas tareas o instrucciones. Sin embargo, estos acercamientos no han resultado en una BCI que funcione para cualquier usuario. Esto puede ser debido a que una pequeña minoría de usuarios de una BCI no genera patrones de actividad cerebral en alguno de los paradigmas de control que puedan ser detectados por el amplificador de EEG o los algoritmos de procesamiento de las señales. [5]

A través de los años, y de acuerdo a la experiencia generada en los laboratorios que se dedican al diseño de interfaces cerebro computadora, un gran problema se está volviendo aparente: *Hasta ahora, no existe una BCI universal que funcione en todos los usuarios.* [5]

1.4.1. ¿Por qué *a veces* las BCIs no funcionan?

Las interfaces cerebro-computadora más populares son implementaciones no invasivas, hechas principalmente con tres paradigmas de operación: potenciales P300, SSVEP y ERD/ERS. Idealmente, estos paradigmas deberían funcionar para todas las personas, sin embargo, numerosos laboratorios en el mundo reportan que un 20 % de los usuarios no pueden lograr el control de una BCI particular [5] [12]. En la literatura, a este problema se le ha llamado incapacidad de uso de una BCI (o "BCI illiteracy" en inglés). En el que a pesar de los esfuerzos, empleando mecanismos de corrección de errores y haciendo un entrenamiento más exhaustivo o probando la estimulación con otros dispositivos (e.g. con LED), algunas personas siguen sin poder usar un paradigma especifico de control de una BCI.

En algunos usuarios, los sistemas neuronales necesarios para el control de la BCI parecen producir una actividad eléctrica detectable en el cuero cabelludo[20]. Esto no se debe a un problema con el usuario ya que las regiones neuronales están sanas y activas, pero la actividad que producen no es detectable por la técnica empleada. En el caso del EEG, es posible que las regiones neuronales de estos usuarios estén ubicadas en un surco del cerebro, muy profundas para ser detectadas por los electrodos de EEG o muy cerca de otro grupo de neuronas con gran actividad.

Algunos factores temporales pueden ser relevantes e influir en el desempeño de las interfaces cerebro-computadora como pueden ser la hora del día, la ingesta reciente de alimentos, alcohol, cafeína o drogas. Por ejemplo, en un estudio particular se encontró que la gente que reportó dormir menos la noche previa al estudio, tuvo un mejor desempeño en la generación de potenciales P300 por lo que incluso proponen este hecho como una forma de mejora para la BCI [20].

1.4.2. Incapacidad de uso en BCIs con paradigma de control de SSVEP

Las causas de que una BCI con paradigma SSVEP no funcione, son debidas en gran medida a que este paradigma requiere acumulaciones claras en regiones del espectro de potencia, y el hecho de que éstos no sean aparentes o demasiado débiles para ser distinguidos del ruido de fondo, genera una identificación poco precisa de la frecuencia a la que el usuario está prestando atención. Esta explicación supone que el usuario sí esta prestando atención al estímulo visual y que la respuesta no puede ser adquirida con los electrodos de EEG situados en el cuero cabelludo, lo que hace pensar, que los grupos de neuronas que generan la respuesta autónoma en estas personas están situados a mayor profundidad que en el promedio de la población y requieren amplificadores de EEG especiales.

1.5. ¿Cómo mejorar la funcionalidad de una BCI?

Brendan Z. Allison y Christa Neuper en [5] proponen cuatro posibles soluciones para mejorar el rendimiento de las interfaces cerebro-computadora:

1. Mejorar la selección o clasificación de las señales de EEG disponibles a través de algoritmos optimizados:

   El procesamiento digital de señales es el componente de una BCI que se puede mejorar más fácilmente porque no requiere equipamiento especial, o desarrollo de algún dispositivo. Sin embargo, por más que se mejore el algoritmo de procesamiento, los resultados no serían buenos si el sujeto no produce una señal detectable por los electrodos de EEG o con baja relación señal a ruido en las señales que permita distinguir entre diferentes estados mentales.

2. Usar sensores que brinden mejor información.

   a) Diferentes técnicas de neuroimagenología.
   Otros métodos no invasivos distintos al EEG para obtener un estimado de la actividad neuronal pueden ser más efectivos, pero se ve perjudicada la portabilidad.

   b) Más o mejores sensores (amplificadores o electrodos)

3. Incorporar corrección o reducción de errores.

   a) Interfaces mejoradas que minimicen los errores posibles o permitan su corrección.
   Estas técnicas son muy usadas actualmente en el uso de la tecnología de los teléfonos; a la hora de escribir en el teclado hay algoritmos que corrigen la ortografía y hasta predicen las posibles palabras que siguen en una frase y aprenden la forma en que el usuario escribe.

   b) Señales adicionales al EEG que ayuden a minimizar los errores, e.g. algunos amplificadores portátiles de EEG tienen incorporados sensores adicionales. Usar las señales que provienen del giroscopio proveen una idea del entorno y permiten dar un menor peso a los intervalos en los que se detectan movimientos abruptos en el usuario y así, eliminar artefactos de movimiento.

4. Generar señales cerebrales que sean fácilmente categorizables:

   a) De entre los paradigmas de control actuales.

   b) Usando paradigmas de control novedosos.

c) Cambiando de paradigma de control si el actual no funciona con el usuario.

d) Combinando diferentes paradigmas de control.

En mi opinión también se deben enfocar los esfuerzos en la mejora del hardware que adquiere las señales así como en integrar los algoritmos de procesamiento a sistemas embebidos que puedan ser colocados dentro del cuerpo humano cerca de donde se generan las señales, para crear un sistema que adquiera y procese la información.

Capítulo 2

Objetivo del proyecto

2.1. Interfaces cerebro-computadora en el laboratorio

En gran medida, las interfaces cerebro-computadora que han sido desarrolladas desde su aparición en la década de 1970, son dispositivos que están destinados para su uso en el laboratorio dada la complejidad de su implementación y uso [21]. En un laboratorio, gran parte de las variables que en el mundo real no podemos despreciar, son eliminadas (o controladas) para aumentar la probabilidad de éxito de un experimento. Por ejemplo, cuando el rendimiento de un experimento depende de la atención que el usuario de una interfaz cerebro-computadora presta a los estímulos visuales que se presentan en una pantalla, se eliminan las posibles distracciones sentando al usuario enfrente del monitor en un espacio aislado en el laboratorio; esto mejora el rendimiento (la calidad de la atención) de la prueba.

2.1.1. ¿Cuál es la necesidad de usar las interfaces cerebro computadora fuera de un laboratorio? y ¿Cuáles son sus complicaciones?

El hecho de que el rendimiento de las interfaces cerebro-computadora disminuya al salir del laboratorio, ha limitado su uso y su popularidad en las personas que las necesitan. Sin embargo, una intención particular de las interfaces cerebro-computadora es ayudar a personas que han perdido las vías de comunicación normales con el mundo que les rodea y brindar un medio de comunicación alternativo para mejorar su calidad de vida, en un medio que no es dentro de un laboratorio.

En el mundo real se ve muy limitado el desempeño de las interfaces cerebro computadora por lo que es necesaria la construcción de hardware dedicado, algoritmos robustos y paradigmas que discriminen las intenciones del usuario en el día a día, porque de lo contrario se produce un rechazo a una solución que no es efectiva.

En el intento de utilizar las interfaces cerebro-computadora fuera del laboratorio se han hecho grandes esfuerzos de expertos en las diversas disciplinas que involucran a la tecnología:

1. Mejoras en los amplificadores

   Las mejoras en los amplificadores comerciales, que son los más actuales, han hecho que estos dispositivos sean menos costosos, portátiles y con una calidad de la señal que es aceptable para aplicaciones no profesionales. Están en su gran mayoría enfocados al segmento de juegos de video o al entretenimiento.

   La tendencia en los fabricantes de amplificadores de EEG es hacerlos cada vez "más portátiles" y más asequibles, basta con observar la evolución de la tecnología a través de los años para darse cuenta. También es importante mencionar que la miniaturización de los amplificadores, las mejoras en la adquisición, el filtrado de las señales y la eliminación de ruido, han mejorado considerablemente con los años. Todos estos adelantos abren la puerta para la realización de aplicaciones que utilizando la instrumentación disponible puedan usarse en la vida real.

2. Mejoras en los electrodos.

   Por experiencia propia, uno de los pasos que más consumen tiempo para poder hacer uso de una interfaz cerebro-computadora en el laboratorio es el tiempo de colocación de los electrodos en la que de manera tradicional, se hace uso de una gorra que los ubica espacialmente, para posteriormente colocar pasta o gel como un medio para mejorar la conducción y disminuir la impedancia con respecto a la referencia en cada uno de los electrodos colocados. Hay varias innovaciones en este campo, ya existen gorras que tienen los electrodos embebidos y solamente se usa gel como medio de acoplamiento. También existen electrodos húmedos que usan solución salina y electrodos secos que evitan el uso de gel [22].

3. Mejoras en el procesamiento de las señales

   En un esfuerzo por incrementar la transferencia de información (*bitrate*) en las BCIs el área más activa para la investigación es en la creación de algoritmos para el procesamiento digital de las señales de EEG ya que no es necesario hacer las adquisiciones porque hay bases de señales de EEG disponibles de varios laboratorios en estos campos

,como la base de señales del Laboratorio de Investigación en Neuroimagenología (LINI) de la Universidad Autónoma Metropolitana (UAM) de Ledesma et. al. disponible en http://akimpech.izt.uam.mx, la base de datos de una competencia de clasificación y discriminación de EEG BCI competition IV data set, o la base de datos del BNCI Horizon 2020, que es un consorcio de laboratorios que consiguió fondos en Europa y tiene disponible información para fomentar la investigación en el campo BNCI Horizon 2020 data set.

Además, los dispositivos móviles como son teléfonos, en la actualidad tienen los puertos de comunicación inalámbrica necesarios, capacidad de almacenamiento, pantallas de alta resolución para el despliegue y el poder de computo necesario para ejecutar el algoritmos diseñados para procesar las señales de EEG adquiridas del amplificador inalámbrico sin hacer uso de una computadora tradicional.

4. Mejoras en los paradigmas de control.

   Ésta es también una disciplina que genera mucha investigación, se está en la búsqueda de paradigmas de control que mejoren la tasa de éxito de las interfaces cerebro-computadora, e.g. la creación de BCIs híbridas que usan SSVEP e información implícita del estado mental del usuario [23].

2.2. Objetivo del proyecto

El objetivo del proyecto fue, *diseñar una BCI que funcione fuera del laboratorio haciendo uso de las tecnologías disponibles, realizar una llamada telefónica utilizando un dispositivo móvil como un teléfono, una tableta, o algún dispositivo portátil a uno de dos posibles usuarios seleccionados* .

En el capítulo previo se desarrolló el contexto acerca del desarrollo de algunas interfaces cerebro-computadora no invasivas, interfaces en las que a pesar de que la relación señal a ruido es menor (cuestión que complica su desarrollo), no existe la necesidad de realizar un procedimiento quirúrgico para realizar el implante de electrodos en la corteza cerebral. Uno de los retos es trabajar con señales de EEG de mala calidad (en comparación con la señales que se obtienen de los amplificadores especializados del laboratorio) provenientes de un amplificador comercial. Este tipo de amplificadores no son tan precisos, porque no es su propósito, en virtud de su portabilidad y su bajo costo. Es por eso que *el tipo de interfaz cerebro-computadora a diseñar debe tomar en cuenta el hardware, el software y el algoritmo de procesamiento del EEG, factores que se consideraron fundamentales para el éxito del proyecto.*

2.2.1. Estado del arte

Se realizó una búsqueda sistemática de evidencia en el metabuscador de la Universidad Autónoma Metropolitana, que incluye a las principales bases de datos indexadas relacionadas con la investigación en diversas áreas en enero de 2012 rastreando trabajos similares para la obtención de evidencia que justificara la realización del proyecto, además de la experiencia de otros laboratorios alrededor del mundo que pudieran coadyuvar a su desarrollo y apoyar en la toma de decisiones en los principales buscadores con la siguiente cadena de búsqueda:

TITLE-ABS-KEY ((Mobile OR Android OR iPhone OR smartphone) AND ("Brain computer interface" OR bci OR "Brain Machine Interface"))

La búsqueda se realizó en los títulos, resúmenes y palabras clave, con una cadena de búsqueda muy general porque la cantidad de resultados fue muy escasa y no todos los artículos que arrojaba la pesquisa eran útiles para el proyecto, e.g. en Scopus se encontraron más de 40 artículos de los cuales se hizo una revisión del título y del resumen para evaluar la pertinencia con el proyecto de los cuales se recuperaron solamente 3 artículos en texto completo que se enlistan a continuación:

Autor	Título	Año	Enlace
Vernon, S., Joshi, S.S.	Multidimensional control using a mobile-phone based brain-muscle-computer interface	2011	Scopus
De Oliveira Miguel, P.V., Muhammad Ismail, S., Barreto, G.	A mobile learning that uses ECOLIG and a brain computer interface with Android	2011	Scopus
Vernon, S., Joshi Sr., S.S.	Brain-muscle-computer interface: Mobile-phone prototype development and testing	2011	Scopus

Tabla 2.1: Búsqueda del estado del arte

En la búsqueda de literatura útil para el proyecto, se encontró un intento por hacer una BCI portátil usando un teléfono para realizar una llamada con el paradigma de control exógeno P300. En el artículo de Campbell et. al. [24] se usa un iPhone™, y parafraseando al autor *"se presenta el diseño del sistema Neurophone, una interfaz cerebro-teléfono móvil basada en el amplificador inalámbrico EPOC de Emotiv y un iPhone"*. Sin embargo, en este artículo explican que debido a que encontraron muchos problemas en el diseño, la transmisión y procesamiento del EEG en el iPhone™, la solución fue adquirir el EEG con una

computadora, filtrarlo y enviarlo en paquetes IP a través de Wi-Fi al teléfono para ejecutar un algoritmo de clasificación. En conclusión terminan usando la información del parpadeo (lo que no es una BCI) cada vez que se ilumina la foto de la persona que se quiere llamar, con la justificación de que mejora el desempeño.

Se actualizó la búsqueda al término del proyecto en el año 2015 y arrojó 51 resultados de los cuales se revisaron todos los títulos y los resúmenes en busca de los artículos relacionados con el proyecto de los cuales se seleccionaron 10 artículos (incluidos los 3 previos) que se recuperaron en texto completo para su análisis.

En una segunda actualización en marzo de 2017, se ejecutó el mismo algoritmo de búsqueda. Encontrando que del año 2015 a la fecha se generaron 20 nuevos artículos relacionados con el tema. De estos 20 artículos, se hizo una eliminación al leer el título de 15 de ellos por no tener relación con el tema buscado. De los 5 restantes se obtuvo y se leyó el resumen, y como resultado de esta lectura se eliminaron 3 artículos que claramente no usaban dispositivos móviles. Los dos artículos restantes se obtuvieron en texto completo de los cuales sólo uno tiene relevancia con el proyecto (mostrado en la Tabla 2.2 en la página 21). Éste se abordará en la sección de Capítulo 7 en la página 115.

Autor	Título	Año	Enlace
Amr S. Elsawy, Seif Eldawlatly, Mohamed Taher, Gamal M. Aly	A mobMindEdit: A P300-based text editor for mobile devices	2017	ELSEVIER

Tabla 2.2: Búsqueda actualizada en 2017

2.2.2. Justificación del proyecto

Los autores del Neurophone [24] no parecían haber abordado el problema de manera rigurosa, porque en el diseño de la BCI hay que tomar en cuenta las complicaciones que conlleva usar un dispositivo móvil, así como elegir el paradigma y el procesamiento de las señales, tomando en cuenta que las señales provenientes del amplificador, al no tratarse de un amplificador de investigación, sino un amplificador comercial contienen mucho ruido, lo que convierte este proyecto en un reto.

La siguiente pregunta de investigación fue planteada: *¿Es posible diseñar una interfaz cerebro-computadora en un dispositivo móvil sin hacer uso de la computadora como medio de adquisición y procesamiento del EEG?*

2.2.3. Características deseables de la BCI a desarrollar

1. Que no sea necesario el uso de una computadora tradicional para la utilización de la BCI.
2. Que el usuario no necesite más de 10 segundos para realizar una llamada telefónica.
3. Que el amplificador de EEG sea un amplificador comercial que se pueda conectar al dispositivo móvil.
4. Que pueda ser implementada por cualquiera que disponga del hardware adecuado.
5. Que no necesite entrenamiento previo a su utilización y dependa de la respuesta autónoma del usuario.

2.3. ¿Qué tipo de interfaz cerebro-computadora diseñar?

Para cumplir con las características deseables de la BCI a desarrollar se tomó la decisión de realizar una interfaz cerebro-computadora dependiente, con el paradigma de potenciales evocados de estado estacionario dicotómica con dos estímulos visuales y con frecuencias de estimulación de 7 y 11 Hz en la que el usuario pueda seleccionar entre dos contactos a llamar, con un botón que inicie la estimulación y realice la adquisición y la toma de decisión de forma automática. La selección de las frecuencias de estimulación se hizo de acuerdo a la experiencia previa en proyectos del LINI considerando que no existiera traslape de las frecuencias fundamentales y el primer armónico.

2.4. ¿Qué algoritmo de procesamiento de EEG usar?

Una mala calidad de la señal proveniente del amplificador puede producir resultados de baja precisión [22], es por eso, que la selección del algoritmo es fundamental. Debe ser un método que compense la mala calidad de la señal proveniente del amplificador. *Se propone el cálculo de la potencia en la banda de paso de la frecuencia de los 2 estimuladores visuales en pantalla y dos armónicos, con un tiempo de adquisición de 4 segundos posteriores y una estimulación visual de 5 segundos que comienza 1 segundo antes de comenzar la adquisición de EEG. Esta decisión fue, tomando en cuenta las recomendaciones de una aplicación similar basada en [6] con una ligera modificación en el tiempo de adquisición por experiencia en los procedimientos realizados en el LINI.*

2.5. Antecedentes

El antecedente principal es el proyecto realizado por este autor y otros colaboradores durante la licenciatura, llamado *Desarrollo de una interfaz cerebro-computadora no invasiva portátil*, que se enfocó al desarrollo de hardware como una primera aproximación para usar una interfaz cerebro-computadora fuera del laboratorio, e.g. para poder usarla en un coche o una silla de ruedas. La realización de este proyecto ocurrió el año 2009, momento en el que los dispositivos móviles no eran aún lo que son hoy en día (año 2015). Integrar en un hardware todas las herramientas necesarias para "montar" una interfaz cerebro-computadora en un único dispositivo que fuera portátil y pudiera adaptarse a una silla de ruedas en ese momento era un reto. Para poner esta situación en un contexto histórico se debe mencionar que el primer iPad fue anunciado el 27 de enero de 2010 [25] y la idea de tener todo integrado en una sola pantalla era muy atractiva en aquella época en la que el poder de procesamiento y los puertos de comunicaciones de los dispositivos portátiles eran muy limitados; no existían sistemas operativos para dispositivos móviles lo suficientemente robustos para la creación de aplicaciones que requirieran establecer una comunicación inalámbrica (o por USB) con periféricos (e.g. amplificadores de EEG), y los entornos de desarrollo para plataformas móviles no eran lo accesibles que son hoy en día.

Capítulo 3

Metodología 1 - Toma de decisiones

3.1. Introducción al Desarrollo de un prototipo inalámbrico de una interfaz cerebro-computadora en una plataforma móvil

Debido a la naturaleza de este proyecto, que conjunta herramientas de diversas disciplinas para el desarrollo, éste fue dividido en tres metodologías: Toma de decisiones, Desarrollo del hardware de la BCI y Diseño de la aplicación en Android; para de esa manera dar el espacio requerido por cada área y facilitar al lector la lectura de la tesis.

3.2. Selección del amplificador de EEG

La selección del amplificador es un punto clave para el desarrollo del proyecto, es por eso que se dedicó buena parte del tiempo a realizar una búsqueda de información y experiencias previas para corroborar que se pudiera usar para el propósito deseado.

3.2.1. Amplificadores de EEG comerciales

Con el propósito de hacer una interfaz cerebro-computadora que hiciera posible usarla fuera del laboratorio en aplicaciones de la vida diaria usando un teléfono como sistema

principal de procesamiento de las señales de EEG y dispositivo de retroalimentación (Ver la figura 1.2 en la página 9) los siguientes criterios de selección del amplificador de EEG fueron tomados en cuenta:

1. Que fuera un amplificador portátil.
2. Que preferentemente fuera inalámbrico para poder conectarlo al teléfono.
3. Que pudiera se adquirido por cualquier usuario que quisiera hacer uso de la aplicación.
4. Que se pudiera obtener el EEG crudo como señal en el tiempo (RAW EEG).
5. Que tuviera varios canales de adquisición, principalmente en la parte occipital de la cabeza, debido a que el paradigma a utilizar es SSVEP. Esto es considerado un punto a favor pero no fue determinante en la toma de decisión.
6. Que la colocación del amplificador fuera lo más fácil posible (comparado con la colocación de un sistema tradicional de laboratorio).

3.2.1.1. Características generales de los amplificadores de EEG que cumplen con los criterios de selección.

En el momento en el que se tomó la decisión de compra del amplificador, dos compañías tenían amplificadores de EEG disponibles que cumplían con las características deseadas (de los que se tenía conocimiento):

1. NeuroSky™ Mindset™ (ver figura 3.1a en la página 27).

   Características generales[1]:

   a) Un canal de adquisición con un electrodo seco que no necesita ningún material conductor intermedio.
   b) Comunicación Bluetooth (incluye el "dongle" para dispositivos que no tengan Bluetooth integrado).
   c) Transmisión de audio estéreo vía Bluetooth.
   d) Compatible con Windows™ , Mac™ y Android™.

[1] http://support.neurosky.com/kb/general-21/what-is-the-difference-between-the-mindset-mindwave-mindwave-mobile-and-xwave | Consultado por última vez en octubre de 2014

(a) NeuroSky^{TM} Mindset^{TM} (www.neurosky.com)

(b) Emotiv^{TM} EPOC^{TM} (www.emotiv.com)

Figura 3.1: Amplificadores comerciales

e) Micrófono.

f) Se puede usar como "manos libres" con el teléfono.

2. Emotiv^{TM} EPOC^{TM} "Neuro headset" (figura 3.1b en la página 27):

   Características generales:

   En ese momento, Emotiv^{TM} tenía disponibles cinco versiones de SDK y dos versiones del amplificador de EEG EPOC^{TM}(de acuerdo a la tabla 3.1 en la página 27) que compartían las siguientes características[2]:

Emotiv SDK		Amplificador		Acceso							Licencia de distribución	
	Development Tools	Emotiv EPOC	Emotiv EEG	Raw EEG	Emo Composer	Control Panel	Expressive Suite	Affectiv Suite	Single Seat License	Multi Seat License	Emo Store	Sin restricción
Lite Edition	✓				✓	✓	✓	✓			✓	
Developer Edition	✓	✓			✓	✓	✓	✓	✓		✓	
Research Edition	✓		✓	✓	✓	✓	✓	✓	✓		✓	
Enterprise Edition	✓	✓			✓	✓	✓	✓	✓		✓	✓
Enterprise Plus Edition	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓
Education Edition	✓		✓	✓	✓	✓	✓	✓		✓	✓	

Tabla 3.1: Características del SDK y versiones del amplificador de EEG EPOC^{TM}

a) Amplificador de EEG que usa electrodos que se humedecen con solución salina.

[2]Software Development Kit User Manual for Release 1.0.0.5 [26]

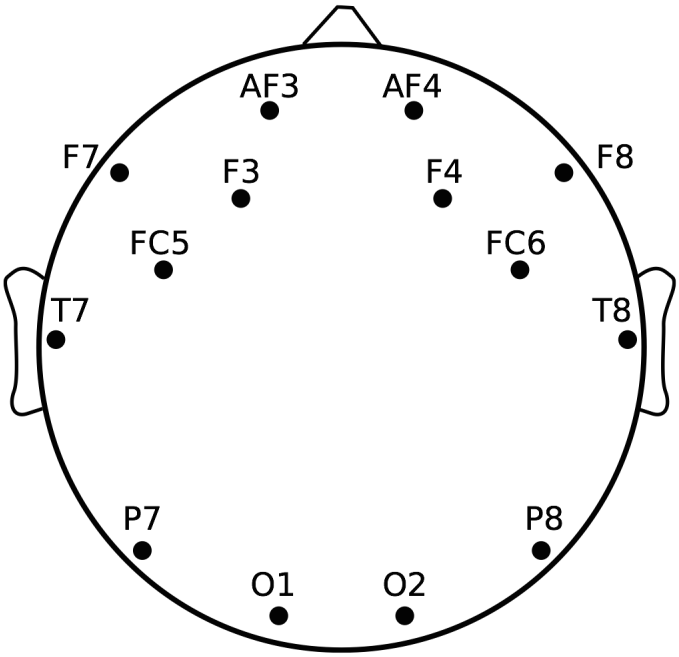


Figura 3.2: Ubicación de los electrodos del EPOC^{TM} en el sistema 10-20

b) Catorce electrodos (ver figura 3.2 en la página 28)con la siguiente nomenclatura en el sistema 10-20: AF3, AF4, F3, F4, F7, F8, FC5, FC6, O1, O2, P7, P8, T7 y T8.

c) Dos electrodos adicionales como referencia, intercambiables entre P3 y P4 (por defecto) o T7 y T8.

d) Giroscopio.

e) "Dongle" USB inalámbrico que no necesita "drivers" y funciona a 2.4Ghz (banda de frecuencia a la que funciona el Bluetooth).

f) Batería de litio de 12 horas.

3.2.2. Búsqueda de la literatura que sustentó la toma de decisión

Al buscar las opciones disponibles de amplificadores de EEG comerciales, se hizo una búsqueda en la literatura disponible en ese momento esperando encontrar experiencias previas en el uso de amplificadores de EEG comerciales en interfaces cerebro-computadora.

Se encontró un artículo de particular importancia en la toma de la decisiones escrito por [27] en que se plantea la pregunta de investigación *P300 y el EPOC de Emotiv ¿El EPOC de Emotiv captura EEG real?*.

En el artículo, usan la versión *research edition* del EPOC^{TM} de Emotiv^{TM} (más adelante se discute acerca de las versiones del amplificador) y concluyen que el EPOC^{TM} de Emotiv^{TM} ***Sí*** *captura EEG real* ya que ellos lograron extraer señales de potenciales provocados P300. Sin embargo, es pertinente hacer la aclaración de que tuvieron que hacer más repeticiones (épocas) de las que se hacen con un amplificador de laboratorio, debido a que las señales provenientes del EPOC^{TM} son muy ruidosas y no se puede comparar el desempeño de este amplificador con uno de uso médico o profesional.

Debido a esa experiencia previa, se tomó la decisión de comprar la version *Research Edition* del EPOC^{TM}, siendo esta versión del hardware con la que en teoría se podía tener acceso al EEG crudo de alguna manera en el "dongle".

Selección del amplificador de EEG inalámbrico

Se eligió el amplificador de EEG inalámbrico EPOC^{TM} de Emotiv^{TM} como el amplificador a usar en el proyecto, porque se consideró que cubría con las características necesarias para diseñar una interfaz cerebro-computadora de control exógeno usando el paradigma de potenciales provocados visuales de estado estacionario (SSVEP), que generan una respuesta automática en la parte occipital del cerebro, y el EPOC^{TM} tiene electrodos en O1 y O2.

Razones por las que se eligió el amplificador de EEG EPOC^{TM} de Emotiv^{TM}

- Posición y número de electrodos.

  Los electrodos O1 y O2 son muy importantes para la adquisición de las señales de EEG en la región occipital del cerebro. También se valoró la opción de usar el Mindset^{TM} (ver figura 3.1b en la página 27) en la posición contraria a la que fue diseñado (es decir con el electrodo frontal en la parte trasera de la cabeza).

- Por la disposición de las señales de EEG:

  Para cumplir el propósito de este proyecto se necesita tener acceso al EEG crudo, característica que de acuerdo a la tabla 3.1 (en la página 27) está disponible en la version *Research Edition, Enterprise Plus Edition y Education Edition* de los amplificadores de EEG de Emotiv^{TM}. De esta tabla, se concluyó que hay dos versiones del hardware

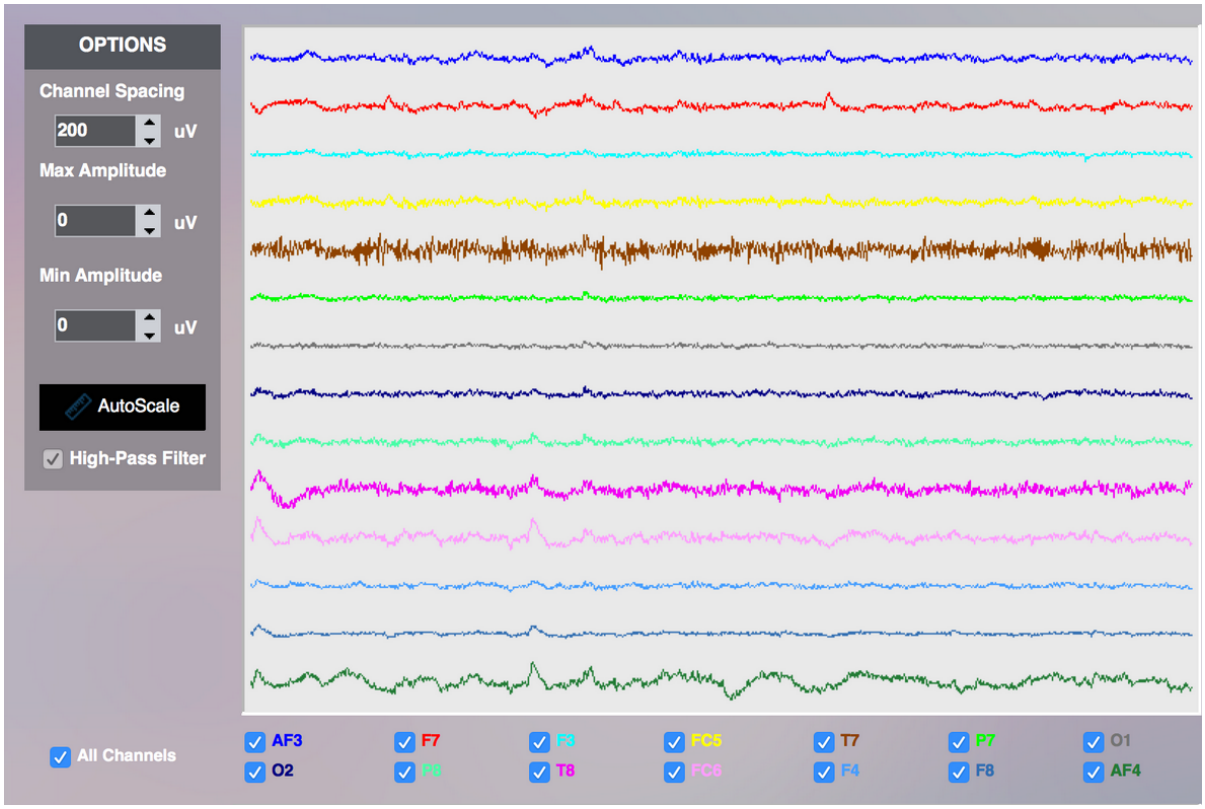


Figura 3.3: Adquisición de EEG del EPOC^{TM}

Fuente: www.emotiv.com

del amplificador, una llamada *Emotiv*^{TM} *EPOC*^{TM} y otra llamada *Emotiv*^{TM} *EEG* que manda las señales de EEG crudo al "dongle". Esto se asumió debido a que en la documentación [26] se pueden ver las señales de EEG graficadas en la computadora (Ver la figura 3.3 en la página 30)

3.2.3. Características detalladas del EPOC™

A continuación se mencionan las características que fueron consideradas relevantes para el proyecto. La información completa que incluye instrucciones de colocación y manejo del amplificador se pueden encontrar en el manual del equipo "Software Development Kit User Manual for Release 1.0.0.5 [26]"

3.2.3.1. Características relevantes para el proyecto

Especificaciones técnicas del amplificador:

Características	Especificaciones (de acuerdo al manual del fabricante)
Número de canales	14 (más dos referencias CMS/DRL)
Etiquetas de los canales (locaciones internacionales 10-20)	AF3, AF4, F3, F4, F7, F8, FC5, FC6, P3 (CMS), P4 (DRL), P7, P8, T7, T8, O1, O2
Método de muestreo	Muestreo secuencial, con un solo convertidor analógico digital (ADC)
Tasa de muestreo	128 Hz (Frecuencia interna 2048 Hz)
Resolución	16 bits (14 bits efectivos) 1 LSB = 0.51μV
Ancho de banda	0.2 - 45Hz, filtros digitales notch a 50Hz y 60Hz
Rango dinámico (con la referencia)	8400 μV (pp)
Modo de acoplamiento	AC
Conectividad	Banda de 2.4 GHz, protocolo inalámbrico cerrado
Tipo de batería	Polímero de Litio
Duración de la batería	12 horas
Medición de la impedancia	Sistema patentado de medición de la calidad de la señal en los electrodos

Tabla 3.2: Especificaciones del EPOC™

3.2.3.2. Cómo funcionan los electrodos y estimación de la calidad de la señal

- Electrodos

  Son electrodos húmedos, en los que para disminuir la impedancia de contacto entre el cuero cabelludo y el material conductor del que está hecho el electrodo (oro en

este caso), se aplica una solución salina a un fieltro y esto sirve como un medio de acoplamiento entre los dos materiales. Los electrodos se guardan en una caja que tiene un material absorbente para que en caso de que se guarden los fieltros aún húmedos, éstos se sequen (ver la figura 3.4 en la pagina 32).

Figura 3.4: Paquete de sensores de hidratación

- Calidad de la señal

  La calidad de la señal es un parámetro que de acuerdo al manual [26] es calculado con un algoritmo patentado. Este indicador de calidad de señal es enviado como un número escalar al "dongle" en donde se relaciona con un color en el SDK de Windows^{TM}: verde para buen contacto; naranja para un contacto adecuado; amarillo para un contacto deficiente; rojo para un mal contacto, y negro para indicar que no hay señal de EEG. En este algoritmo se hace uso de la eliminación activa de ruido para estimar la calidad de la señal.

- Electrodos de referencia

Los electrodos de referencia son P3 y P4 (por defecto) o T7 y T8 (ver figura 3.5) en la pagina 34. Son intercambiables en pares si no se tiene una buena calidad de la señal. Los electrodos de referencia son el par que no tiene colocadas las gomas (se encuentran "al aire"), que son los que tienen los fieltros humedecidos con la solución salina (ver figura 3.4 en la página 32). Estos electrodos son los utilizados para la eliminación activa de ruido.

- Eliminación activa del ruido

  La eliminación activa del ruido se hace con los electrodos de referencia (ver la 3.5 en la página 34), también llamados:

  - Electrodo CMS (P3 o TP9)
    Es un electrodo activo en modo de censado común (Common Mode Sense electrode) o ruido de fondo.
  - Electrodo DRL (P4 o TP10)
    Se le conoce también como electrodo pasivo para "pierna derecha" (Driven Right Leg). El electrodo DRL es el único punto de retorno de corriente entre el usuario y la caja analógico-digital (AD-box). La corriente es controlada electrónicamente a un máximo de 50 μA, lo que hace que el usuario esté protegido contra de un flujo excesivo de corriente debido a un amplificador o un electrodo defectuoso.

  Estos dos electrodos forman un circuito de soporte (ver la figura 3.6 en la página 35) que calcula el potencial promedio del usuario en modo de voltaje común (CM) lo más cercano posible a la referencia del convertidor analógico-digital (ADC) en la caja analógica digital (AD-box). La referencia del convertidor analógico-digital puede ser considerada como el "cero" en el amplificador. El uso combinado del circuito CMS/DRL provee beneficios que no son obtenidos fácilmente en una configuración estándar de tierra única presente en los amplificadores de investigación.

3.2.3.3. Comunicación inalámbrica del EPOC™

Se asumió que la comunicación inalámbrica entre el EPOC™ y la computadora o dispositivo móvil, al leer la documentación oficial, correspondía al protocolo Bluetooth. La verdadera forma en la que la computadora recibe los datos es a través del "dongle" USB 2.0 que transmite la información del amplificador a la computadora, en donde son interpretados por el SDK, se hace con un protocolo propietario cerrado a 2.4 GHz, misma banda en la que funciona el Bluetooth, aunque éste no lo es.

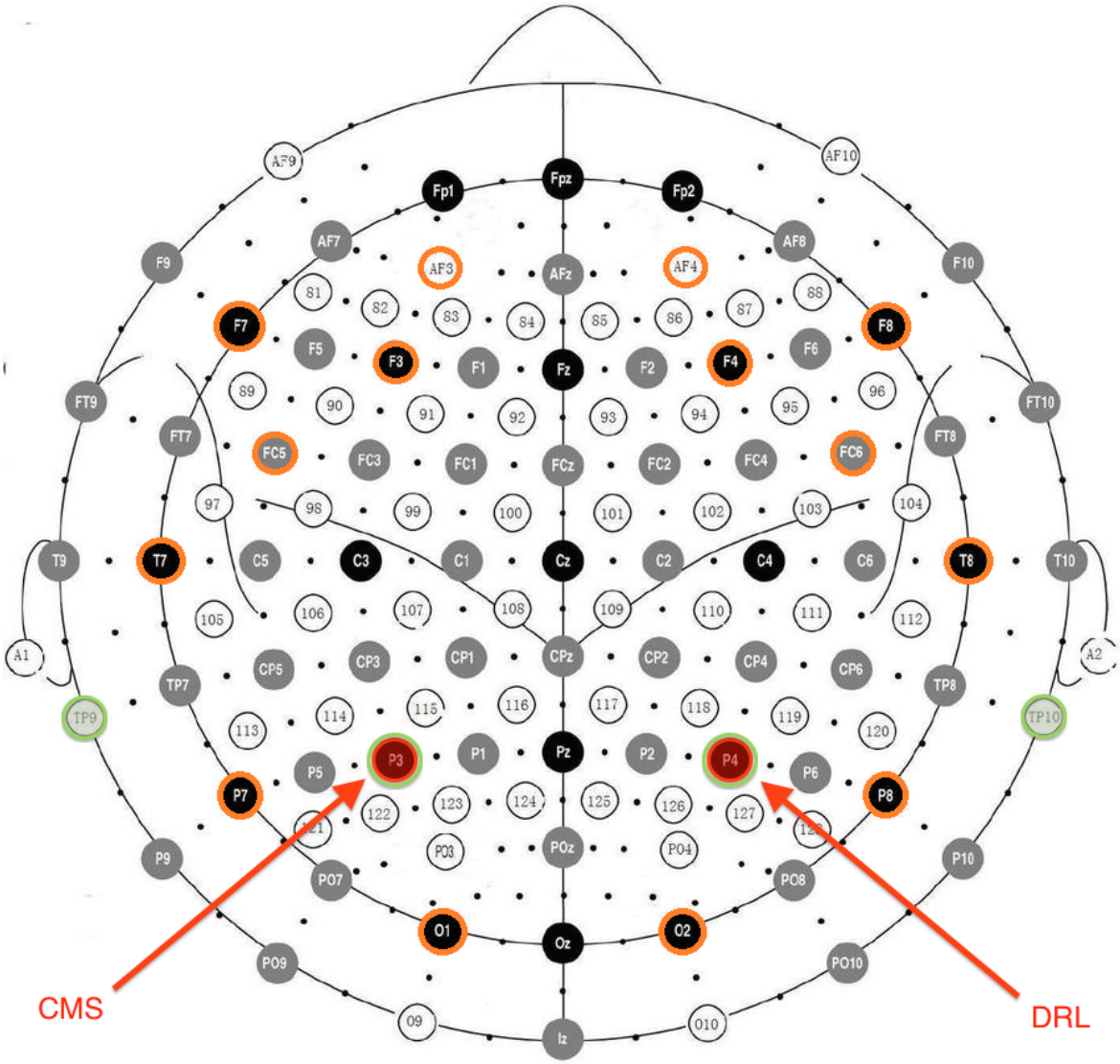


Figura 3.5: Electrodos de referencia

3.2.3.4. Características del software de Emotiv™ (SDK) y por qué no se puede usar para el proyecto

La versión del amplificador que se compró para el desarrollo del proyecto fue la versión de investigación (Research Edition), se eligió así porque de acuerdo a la documentación, esta versión permitía extraer el EEG crudo del "dongle" del EPOC™, característica que resultó

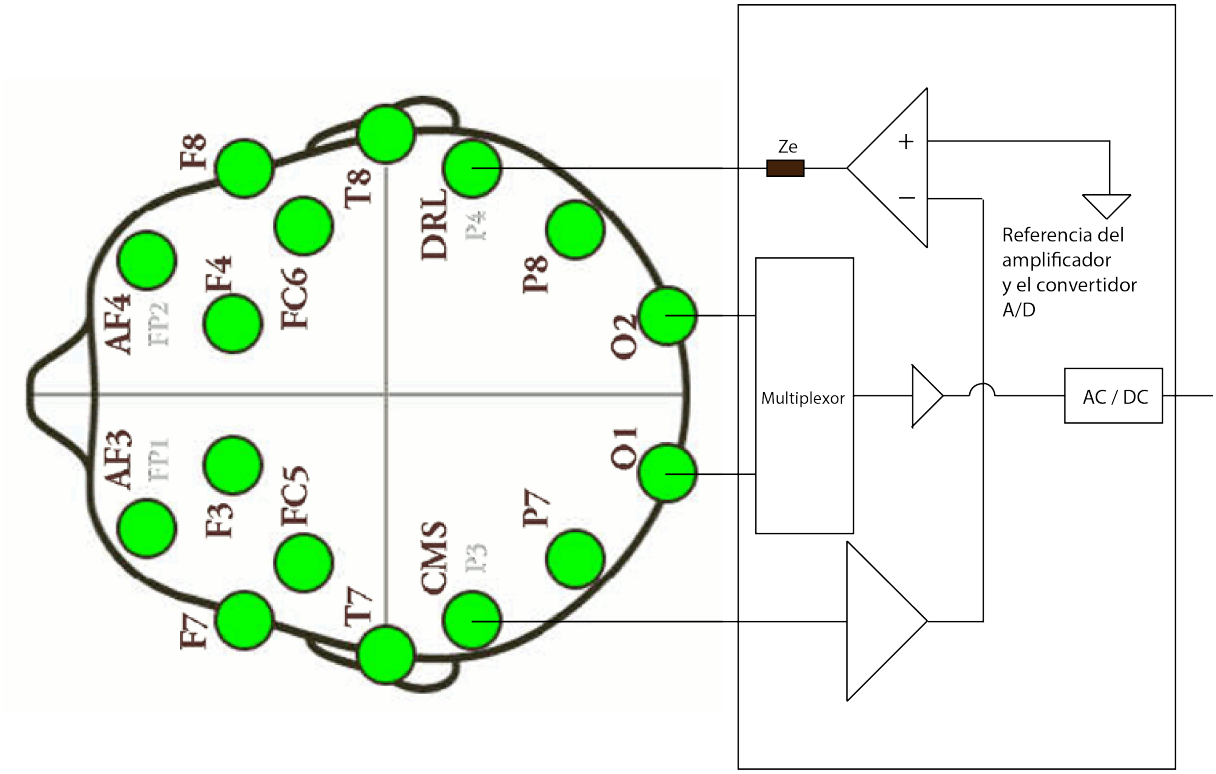


Figura 3.6: Eliminación activa del ruido

no ser cierta, o por lo menos, no para el objetivo de este proyecto como se explica en el punto 3.4 en la página 39.

El desarrollo de una interfaz cerebro-computadora no es el propósito principal del EPOC™ ya que está enfocado al uso de señales de EEG en actividades lúdicas. Por esta razón las herramientas disponibles en el SDK para el desarrollo de aplicaciones se limita al sistema operativo Windows™. Con la finalidad de evidenciar que ninguna de las herramientas desarrolladas por Emotiv™ para hacer uso de las señales de EEG provenientes del EPOC sirve para el objetivo, se enlistan a continuación los componentes de SDK de Emotiv™:

- Expresiones (Expresiv Suite™ [26])

  A través de umbrales detecta las siguientes expresiones faciales que en su mayoría son producto de la acción de los músculos de la cara y no del EEG. En el panel de SDK se pueden configurar umbrales para la detección de las expresiones y así programar controles.

1. Parpadeo bilateral
2. Parpadeo izquierdo
3. Parpadeo derecho
4. Ver hacia la izquierda o derecha
5. Levantar la cejas
6. Fruncir la ceja
7. Sonreír
8. Apretar los dientes
9. Reír

- Emociones subjetivas (Affective suite™ [26])

  De acuerdo al manual del EPOC™ [26], este paquete del SDK reporta cambios en tiempo real de tres emociones subjetivas:

  1. Concentración / Aburrimiento (Engagement / Boredom)

     Se estima mediante un aumento o disminución (según sea el caso) de la potencia en la banda de frecuencias Beta del EEG.
  2. Emoción instantánea (Instantaneous Excitement)

     El manual no dice como lo estiman, sólo menciona que es una emoción que está caracterizada por la activación del sistema nervioso simpático que como resultado, tiene una gama de respuestas fisiológicas.
  3. Emoción a largo plazo (Long term Excitement)

     Es lo mismo que la emoción instantánea, pero en una ventana de mayor tiempo (minutos).

- Procesos relacionados con el pensamiento (Cognitiv Suite™ [26])

  En este paquete del SDK se habla de procesos relacionados con la actividad cerebral en tiempo real para discernir cuando el usuario de manera consciente intenta realizar trece diferentes acciones físicas sobre un objeto virtual, siendo éstas:

  1. Seis movimientos direccionales:
     - Empujar
     - Jalar
     - Mover a la izquierda

- Mover a la derecha
- Mover hacia arriba
- Mover hacia abajo

2. Seis rotaciones:
   - En el sentido de las manecillas del reloj
   - En el sentido contrario al de las manecillas del reloj
   - A la izquierda
   - A la derecha
   - Hacia arriba
   - Hacia abajo
3. Una acción adicional que existe sólo en la imaginación: *desaparecer*.

De la misma manera que con los otros paquetes, todas estas tareas mentales se pueden relacionar a eventos que generan una acción cuando se supera un cierto umbral determinado. Es importante mencionar que este tipo de tareas mentales requieren de entrenamiento, de acuerdo al manual, y es difícil mantener la tarea mental por periodos de tiempo prolongados.

Todas estas son instrucciones de control que el conjunto de herramientas para el desarrollo de software (SDK por sus siglas en inglés) de Emotiv™ provee al programador; comandos que son el resultado del procesamiento del EEG crudo. En la version del SDK que está incluida con el amplificador que se compró para el proyecto es posible extraer el EEG crudo como señal haciendo la solicitud a la API de Windows™.

3.3. Selección del dispositivo móvil

La selección del dispositivo móvil para la realización del proyecto fue una decisión fundamental para el éxito del desarrollo, porque son muchas variables que se tienen que tomar en cuenta para que se pueda concluir con una aplicación funcional distribuible que pueda ser instalada por cualquier persona que la necesite y disponga del hardware necesario, como se expresa en una de las características deseables de la BCI a desarrollar en la sección 2.2.3 en la página 22.

3.3.1. ¿Qué dispositivo móvil usar para el desarrollo de la BCI?

En este sección se discuten cuáles son la opciones disponibles para el desarrollo de la interfaz cerebro-computadora y se justifica la elección.

3.3.1.1. Requisitos para la selección del dispositivo móvil

1. Requisitos de hardware:

   - Dispositivo portátil con conexión Bluetooth[3] para recibir la señal del amplificador.
   - Con capacidad para realizar llamadas telefónicas.
   - Con pantalla a color con resolución HD de al menos 4 pulgadas.
   - Con la capacidad de procesamiento necesario para realizar las siguientes tareas de forma simultánea:
     - Mostrar los estimuladores.
     - Adquirir y procesar la señal.
     - Hacer lo cálculos necesarios para la clasificación.
     - Ejecutar la acción proveniente de la toma de decisión del algoritmo.
   - Que funcione con batería o conectado a la corriente eléctrica

2. Requisitos de software:

   - Que exista un entorno de desarrollo integrado (IDE por sus siglas en inglés) que permita interactuar con el dispositivo móvil.
   - Que el sistema operativo del dispositivo móvil tenga soporte para leer los datos provenientes del amplificador, e.g. leer los datos del "buffer" de comunicación Bluetooth[3].
   - Que la interfaz gráfica de la aplicación permita el diseño y el manejo del tiempo para la creación de los estimuladores.
   - Que se disponga de las herramientas para el procesamiento digital de señales en el dispositivo móvil para procesar el EEG.

[3]En la sección 3.4 en la página 39 se explica que este punto no es más un requisito indispensable; sin embargo, surgieron otros requisitos

3.3.1.2. Selección del hardware

Las prestaciones de los dispositivos portátiles de gama media y alta disponibles de manera comercial en 2012 (momento en el que se hizo la planeación del proyecto), se consideraron suficientes para su realización.

3.3.1.3. Selección del software

La opciones disponibles de software para hacer el desarrollo de la BCI en un dispositivo móvil se redujeron a dos: Android™ e iOS™.

Dados los requerimientos del hardware, la selección del software pudo ser por cualquiera de las dos tecnologías disponibles. En la sección 3.4.3 en la página 40 se describe cuál fue la decisión y por qué.

3.4. Modificaciones al proyecto debido al problema de la extracción de los datos de EEG

Debido a que el desarrollo de una interfaz cerebro-computadora no es el objetivo del EPOC™ sino el desarrollo de aplicaciones haciendo uso de los comandos de control que proporciona el SDK, usando Windows™ como sistema operativo. Cuando se usó el EPOC™ por primera vez, surgieron algunos puntos esenciales que en la documentación oficial no eran claros.

3.4.1. Confusión con la documentación oficial del EPOC™

En la sección 3.2.1.1 en la subsección de características generales del EPOC™ (punto 2*e* de la página 28) se encuentra la información oficial disponible en la página web de Emotiv™ y en los manuales descargables, información que llevó a pensar que el "dongle" USB funciona en la banda de frecuencias en las que trabaja el estándar Bluetooth (2.4 GHz). *Así, se asumió que, la transmisión de la información entre el "dongle" y el amplificador era a través del estándar Bluetooth, y que ambos tenían el hardware necesario para poder "emparejarse" con cualquier dispositivo Bluetooth. Pero no fue así.* Como bien lo dice la documentación (de forma confusa o inclusive engañosa) el "dongle" sí funciona en la banda de 2.4 GHz pero no utiliza el protocolo Bluetooth.

Se identificó entonces que la comunicación del EPOC™ con el "dongle" es a través de un protocolo propietario y es estrictamente necesario el uso del "dongle" USB para poder recibir los datos del amplificador.

Como se mencionó en la Subsección 3.2.3 en la página 31, la documentación oficial menciona que es posible tener acceso a las señales de EEG de cada uno de los electrodos del EPOC™. Así que se contactó al soporte técnico de Emotiv para preguntar ¿de qué manera se podía obtener el EEG crudo del EPOC™ como lo mencionaban en la documentación? Y la respuesta fue la siguiente: Para ello es necesario hacer la solicitud de las señales con un método programado al API de Emotiv™ que está instalada en Windows™ y tener instalado el SDK. El propósito de este proyecto era usar un dispositivo móvil como un teléfono, pero en el momento en el que se desarrolló el proyecto ningún dispositivo tenía instalada una version de Windows™ compatible con el SDK de Emotiv™ (no es el propósito de este sistema operativo), y no había un SDK o una API disponible para Android™ o iOS™ que eran los sistemas operativos para dispositivos móviles en aquel momento.

Esta limitación no justificaba cambiar de amplificador de EEG (además de que ya se había recibido), así que se tomó la decisión, para no retrasar el proyecto, de hacer las siguientes modificaciones.

3.4.2. Modificaciones en la selección del hardware

Debido a que la comunicación del EPOC™ con el "dongle" se realiza a través de un protocolo propietario y es estrictamente necesario el uso del "dongle" USB para poder recibir los datos del amplificador, *se tuvo que hacer la selección del hardware, para adecuarse a este nuevo requerimiento.* Era indispensable que el dispositivo móvil que se eligiera para el proyecto, tuviera la capacidad de conectar el "dongle", no sólo el puerto físico sino el hardware necesario para establecer una conexión de dispositivo maestro-esclavo (teléfono-"dongle"). Por lo tanto un nuevo requerimiento de hardware fue un dispositivo móvil con un puerto "USB host" capaz de aceptar la conexión del dongle del EPOC™.

3.4.3. Modificaciones en la selección del software

Se tomó una decisión en base a los requerimientos de hardware explicados anteriormente, ya que los dispositivos de Apple (iPhone™ e iPad™) tienen un conector propio que no es USB y no existe un adaptador oficial de la marca, por lo cual sería necesario un adaptador fabricado por un tercero que forme parte del programa MFi (Made For iPhone/iPad/iPod).

Así este adaptador tendría que ser desarrollado por un fabricante que brinde soporte a desarrolladores, porque la Interfaz de programación de aplicaciones (API) de Apple para USB muy limitada. En la documentación de Apple se argumenta que "por seguridad" el acceso a muchos puertos de comunicación no está abierto al programador de forma nativa debido a que los protocolos de comunicación tienen que estar implementados en el hardware y estos están reservado para el sistema. En estos casos el fabricante debe proveer un API para el desarrollo de aplicaciones que usen este hardware específico para la comunicación USB.

Por las razones expuestas en el párrafo anterior, que podían haber puesto en riesgo el éxito del proyecto, se tomó la decisión de usar un dispositivo con Android porque es un sistema operativo de código abierto con acceso a todas las funciones nativas.

Capítulo 4

Metodología 2 - Desarrollo del hardware de la BCI

Llegado a este punto, estaba claro que no se iba a hacer uso del SDK de Emotiv™ porque la finalidad del proyecto era tener las señales de EEG en el dispositivo con Android. También se sabe que las señales provenientes del amplificador inalámbrico se transmiten usando un protocolo no estándar elaborado por Emotiv™. Por lo tanto, el paso siguiente era extraer los datos del "dongle" en el dispositivo móvil para poder usarlos sin hacer uso de la API de Emotiv™ (utilidad que no existe en Android). El "dongle" del EPOC™ tiene un conector USB, así que el primer paso para poder entenderlo, fue investigar con detalle el estándar USB.

4.1. Estándar USB

El protocolo USB refiere a un canal de comunicación serial universal (Universal Serial Bus), desarrollado para manejar comunicaciones entre computadoras personales (PC) y periféricos con necesidades variables. Toda comunicación USB se hace entre un "USB host" (el que controla la comunicación) y un dispositivo USB ("USB device"), que es controlado por el "USB host" y sólo responde al mismo. Estas dos propiedades están habilitadas por un hardware específico y no pueden ser emuladas por software, como se explicará más adelante.

4.1.1. Hardware USB

Para poder hacer del USB un estándar universal, es necesario que desde el diseño se utilice un hardware específico que lo categorice como un dispositivo "USB host" o "USB device", esto está implementado en una serie de circuitos integrados que controlan el intercambio de las señales en el más bajo nivel en donde hay varios algoritmos para establecer las comunicaciones bidireccionales, de control, y corrección de errores de las mismas.

De acuerdo a la intención con la que está diseñado el dispositivo, existen una serie de conectores para el "USB host" y el "USB device" como se muestra en la figura 4.1 en la página 45 en la que podemos apreciar que existen conectores para los USB devices y receptáculos para los "USB host", de acuerdo al finalidad de cada dispositivo.

Los dispositivos USB deben tener uno o más "endpoints", que son "buffers" que almacenan la información recibida o que está lista para ser transmitida. Cada dirección "endpoint" tiene asociado un número de "endpoint", una dirección (entrada o salida), un tipo de transferencia, y el número máximo de bytes de datos que puede enviar o recibir en una transacción. Una transferencia USB consiste en una o más transacciones que acarrean datos desde un "endpoint" o hacia un "endpoint" [3]. En la tabla 4.1 en la pagina 44 se explican la tres fases para el intercambio de datos entre un "USB host" y un "USB device" en el estándar USB 2.0. Hay transacciones que no llevan a cabo las tres fases porque en el paquete de datos se incluyen bits para la detección de errores.

Fase	Objetivo
Token	El "USB host" especifica un número de dispositivo, un número de "endpoint" y una dirección para la fase Data.
Data	El "USB host" envía datos al dispositivo o el dispositivo envía datos al "USB host".
Handshake	El receptor de los datos en la fase Data envía información acerca del éxito o el error en la transacción

Tabla 4.1: Fases el intercambio de datos en el estándar USB 2.0 [3]

El USB soporta cuatro tipos de transferencias: control, bulk, por interrupción y asincrónica. Las transferencias de control tienen asignadas una serie de etapas en procesos que se llevan a cabo en el hardware; las otras tres (bulk, por interrupción y asincrónica) no tienen etapas definidas en hardware, es software de alto nivel el que define como realizar la transferencia de datos [3]. Las transferencias por interrupción tienen garantizada una latencia máxima o

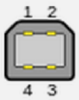
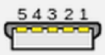
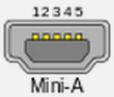
Receptáculo	Conector					
	 Type A	 Type B	 Mini-A	 Mini-B	 Micro-A	 Micro-B
 Type A	Si	No	No	No	No	No
 Type B	No	Si	No	No	No	No
 Mini-A	No	No	Descontinuado	No	No	No
 Mini-AB	No	No	Descontinuado	Descontinuado	No	No
 Mini-B	No	No	No	Si	No	No
 Micro-AB	No	No	No	No	Si	Si
 Micro-B	No	No	No	No	No	Si

Figura 4.1: Matriz de conectores USB con su complemento [1]

Fuente: Modificado de https://en.wikipedia.org/wiki/USB

un tiempo determinado entre intentos de transacción, y están diseñadas para dispositivos que necesitan enviar o recibir información sin retrasos en las que el "USB host" programa las transacciones en el tiempo con un cierto intervalo entre cada una de ellas.

4.1.1.1. ¿Cómo se comunica el "USB host" con los dispositivos USB?

Para comunicarse con un dispositivo, el "USB host" debe usar una clase de comunicación predefinida en los estándares, o usar protocolos definidos por el fabricante del dispositivo. Los dispositivos USB (USB devices) pueden usar también protocolos definidos por la industria, o propietarios, para implementar funciones de alto nivel (interfaces). El hardware que define a un "USB host" tiene implementadas más de 15 clases definidas para periféricos comunes [3], sin embargo, las interfaces dependen del diseño que haya hecho el fabricante del dispositivo. En las computadoras, el sistema operativo tiene definidos muchos controladores ("drivers" en inglés) para acceder a muchas clases, con sus interfaces correspondientes, de dispositivos populares. No obstante, en un dispositivo móvil esta lista de clases e interfaces puede ser muy limitada; es decir, no existen "drivers" para una gran cantidad de dispositivos USB, e.g. un dispositivo móvil con Android puede tener la opción de conectar una impresora al puerto USB y poder imprimir, porque muchas de las impresoras usan protocolos de comunicación estándar para enviar los trabajos de impresión. Por otro lado, la información que proporciona datos, como el nivel de tinta en los cartuchos, usa interfaces propias establecidas por cada fabricante, que dependen de un controlador para su interpretación.

Un ejemplo de una clase definida en el estándar USB es la del dispositivo de interfaz humana (HID por sus siglas en inglés), cuya descripción general se observa en la tabla 4.2 en la página 46.

Clase USB	Código de clase (hexadecimal)	Descriptor que declara la clase
HID (teclado, ratón o función específica del fabricante)	03	Interfaz

Tabla 4.2: Clase USB HID

4.1.1.2. Descriptores USB

La forma en la que un "USB host" aprende acerca del dispositivo que se conecta a él, es examinando los descriptores (estructuras de datos) que son obtenidos durante la enumeración (proceso en el que el hardware USB intercambia datos antes de hacer la conexión de alto nivel). Por estar así normado, cada dispositivo que usa el estándar USB debe tener la capacidad de responder a las peticiones de descriptores (se muestran algunos en la tabla 4.3 en la página 47) hechas por el "USB host". En algunos casos estas solicitudes de información

son estrictamente necesarias de acuerdo al diseño de cada dispositivo USB. Los tres descriptores que se van a encontrar en todos los dispositivos USB, sin importar de que tipo son, representan la mínima información para que la comunicación se pueda realizar de manera efectiva; éstos son el descriptor del dispositivo (01h), el descriptor de la configuración (02h) y el descriptor de interfaz (04h). Con esa información es probable que se pueda descifrar como está hecho el "dongle" y poder leer los datos de EEG.

bDescriptorType (valor que identifica al descriptor)	Tipo de descriptor	¿Es requerido?
01h	Dispositivo	Si
02h	Configuración	Si
03h	Texto (String)	No, sólo en caso de que un "driver" lo requiera. Texto descriptivo opcional
04h	Interfaz	Si
05h	"endpoint"	Si, en caso de que se use uno diferente al "endpoint" cero

Tabla 4.3: Tabla de los descriptores que conforman a un dispositivo USB

Estos descriptores contienen mucha información en cada uno de ellos, la cual define la forma en la que el dispositivo USB fue diseñado, permitiéndole al "USB host" saber qué tipo de conexión establecer a nivel de hardware, ya que la comunicación de alto nivel se hace con los controladores. La información que pueden entregar los descriptores (entre otra) puede ser:

- "id" del fabricante (asignado por el Foro de implementación USB (USB-IF).
- "id" del producto.
- Versión del dispositivo.
- Tamaño del paquete máximo en el "endpoint" cero.
- Número de configuraciones.

De forma opcional, el dispositivo puede especificar muchos otros descriptores, como:

- Clase.
- Subclase.
- Protocolo.
- Uno o más descriptores de interfaz.

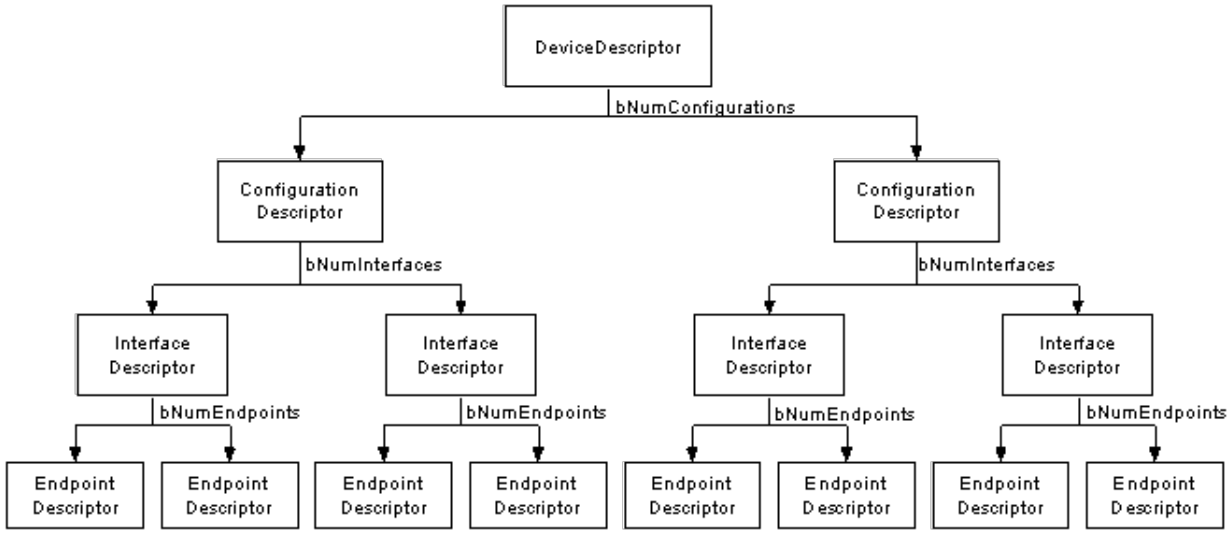


Figura 4.2: Árbol genérico de descriptores USB

Un dispositivo que especifique su función en descriptores de interfaz puede tener múltiples funciones, e.g. un dispositivo puede funcionar como impresora y como dispositivo de almacenamiento masivo, teniendo cada función su propia interfaz. Por otra parte, algunas clases, como la clase audio, necesita varias interfaces para implementar una sola función.

Una interfaz puede tener cero o más *"endpoints"* subordinados, el descriptor de un "endpoint" especifica lo siguiente:

- El número del "endpoint".
- La dirección del flujo de datos (entrada o salida).
- El tipo de transferencia.
- El tamaño máximo del paquete.

- El intervalo de sondeo (polling interval).

Existen dos maneras para conocer los descriptores de un dispositivo USB, en ambas, el primer paso es conectar el dispositivo a un "USB host"; después, en la primera alternativa se observa el tráfico entre ambos con un analizador de protocolos en hardware que capture el tráfico de la enumeración; en la segunda, se usa un software que capture el tráfico de la enumeración, disponible en los sistemas operativos más populares en las computadoras. En la figura 4.2 en la página 48 se puede observar un árbol genérico de un dispositivo USB con dos descriptores de clases y a su vez con dos descriptores de interfaz y dos descriptores de "endpoint" cada uno.

4.1.1.3. Alimentación en el puerto USB

Una característica muy conveniente del puerto USB (de hecho tiene dos cables específicos para hacerlo) es poder obtener energía del mismo puerto (5V). Sin embargo hay una limitación en corriente en el estándar USB 2.0 a 500 mA, lo cual acarrea una responsabilidad por parte de los fabricantes, reducir el consumo energético en un estado de suspensión [28]. Para dispositivos que requieren más corriente de la que puede entregar el bus de alimentación, es necesario conectar una fuente de energía externa al dispositivo USB. El "dongle" del EPOC™ se ajusta a la corriente que le puede entregar el bus, y es por eso que no tiene un conector de alimentación externo, está pensado como un dispositivo Plug and Play.

4.1.2. El "dongle" del EPOC™

El hardware del "dongle" del EPOC™ está conformado por un dispositivo que tiene 2 indicadores tipo LED (uno que indica que está recibiendo alimentación del puerto "USB host" y el otro indica la conexión con el EPOC™). Está diseñado para conectarse a la computadora por medio de un conector USB tipo A con cuatro terminales físicas, dos para la transmisión de datos y dos para la alimentación de energía por parte del "USB host" (ver figura 4.1 en la página 45).

A nivel de "firmware", para conocer cómo está diseñado el "dongle" necesitamos conectarlo a una computadora y usar un software que capture el tráfico de la enumeración para poder darse una idea de cuáles son las clases que tiene implementadas.

El resultado del software que captura el tráfico de la enumeración al conectar el "dongle" a la computadora, arroja los siguientes datos (ver figura 4.4 en la página 51) que fueron ordenados en la estructura de un árbol para facilitar el entendimiento.

Figura 4.3: "dongle" del EPOC™

Fuente: www.emotiv.com

Descriptor de clase

Todos los dispositivos USB deben tener un descriptor de dispositivo, en este caso (como se puede ver en la figura 4.4 en la página 51), el primer descriptor de dispositivo, indica que el "dongle" del EPOC™ es un dispositivo USB 2.0 de clase 0. Esto significa que el descriptor de interfaz es el que especifica la clase y que la función no usa un descriptor de asociación [28]. El descriptor de dispositivo también indica que es un dispositivo de subclase 0, con un tamaño de paquete en el "endpoint" 0 de 8 Bytes (bMazPacketSize0 = 8), señalando el "id" del fabricante, 0x1234; el "id" del producto, 0xed02; y la versión, 0.03 (bcdDevice). La descripción iSerial indica el índice que apunta al lugar que contiene el número de serie del "dongle" (índice 3), y por último, el descriptor de dispositivo indica que sólo hay una posible configuración que es el descriptor que se muestra en el nivel inferior siguiente del árbol.

Descriptor de configuración

Después de obtener el descriptor de clase, el "USB host" puede solicitar el descriptor de configuración porque todos los dispositivos USB deben tener un descriptor de configuración que especifique las habilidades del dispositivo (cuántas funciones tiene). El descriptor de configuración del "dongle" de EPOC™ (ver figura 4.4 en la página 51) muestra que es un dispositivo con un único propósito ya que sólo existe un descriptor de configuración (en caso de que haya más, solamente una configuración puede estar activa a la vez). Los primeros tres parámetros (bLength, bDescriptorType, wTotalLength) nos dan información acerca del propio descriptor. La información relevante está en la sección donde se indica que se implementan 2 interfaces y que sólo tiene un valor de configuración (bConfigurationValue). Este descriptor identifica las peticiones *Get Configuration y Set Configuration* y muestra que no hay descripción de la configuración (iConfiguration = 0). El atributo "bmAttributes" es más

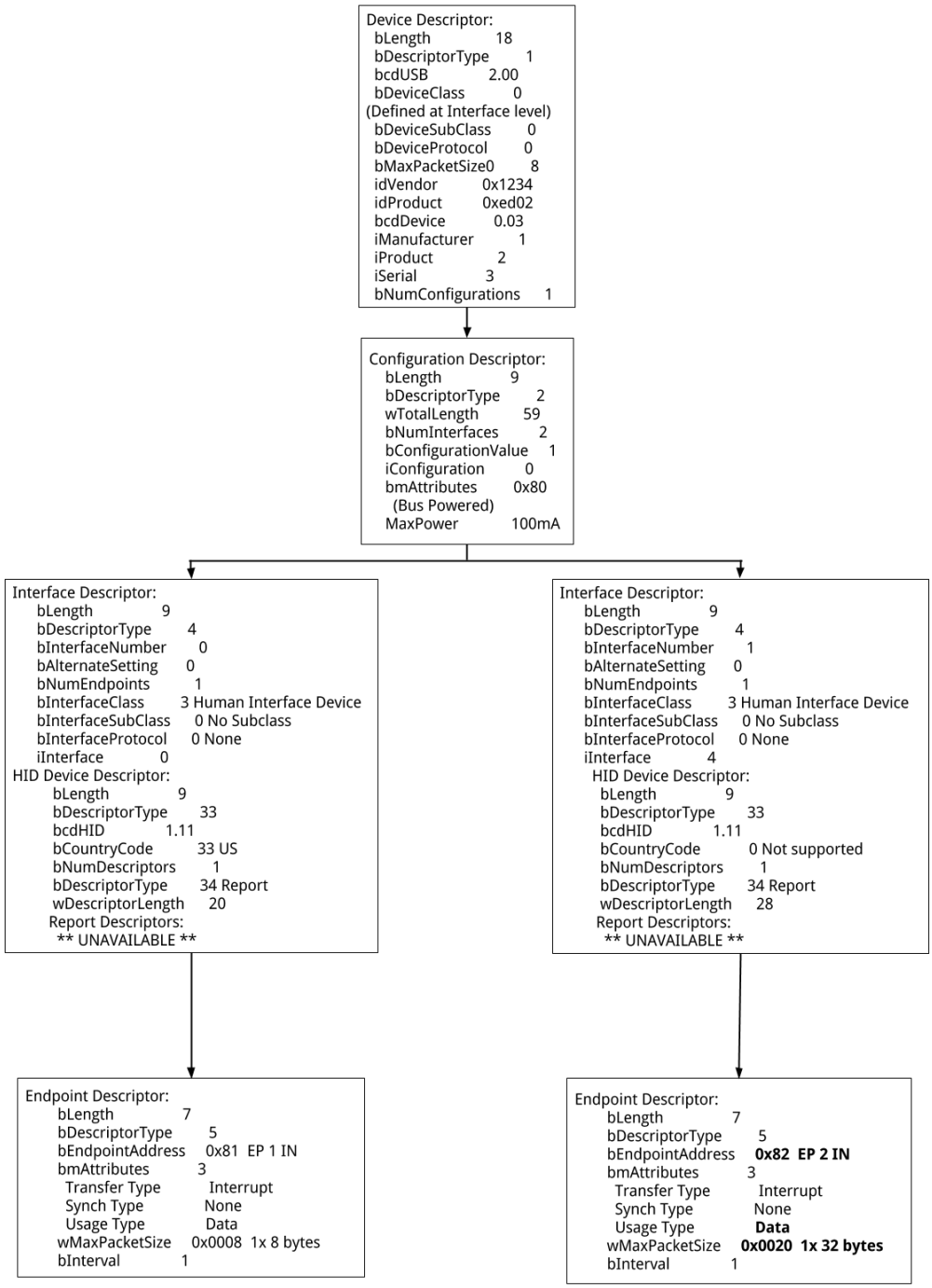


Figura 4.4: Árbol descriptor del "dongle" USB del EPOC^{TM}

complicado, hecho por el que se explica en la figura 4.5 en la página 52. Por último, expresa el consumo energético máximo del "dongle", 100 mA .

0x80							
1	0	0	0	0	0	0	0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Compatible con USB 1.0	0 = recive la alimentación del bus 1 = alimentación externa	Función para avisar si se quiere mandar información al estar en modo de suspensión	Siempre debe ser cero	Siempre debe ser cero	Siempre debe ser cero	Siempre debe ser cero	Siempre debe ser cero

Figura 4.5: bmAttributes

Descriptores de las interfaces

El "dongle" tiene habilitadas dos interfaces como lo muestran los descriptores de interfaz de la figura 4.4 en la página 51. Ambas son interfaces con un "endpoint" de clase HID; prácticamente no hay diferencias entre ellas, el dato relevante es que ambas tienen un descriptor de "endpoint" al que se puede acceder para recuperar la descripción.

Descriptores de "endpoint"

Los "endpoints" son los descriptores más relevantes para el proyecto porque es en estos en los que se hace la descripción de los paquetes de datos que el "dongle" envía al "USB host" para poder ser recuperados, con el "driver" o software, en lenguaje de alto nivel desde la computadora.

Direcciones de los "endpoint" 1 y 2:

- "endpoint" 1 (bEndpointAddress 0x81 EP 1 IN)
- "endpoint" 2 (bEndpointAddress 0x82 EP 2 IN)

4.1.3. Resumen del "firmware" del "dongle" del EPOC™

El "dongle" del EPOC™ es un dispositivo USB que se alimenta a través de los conectores de alimentación del "USB host" con un consumo máximo de 100 mA. Es un dispositivo HID genérico con dos interfaces con un "endpoint" de salida en cada uno de ellos, en los que el "USB host" recibe paquetes de datos por interrupción y sin sincronía de 8 y 32 bytes respectivamente, por lo que no hay forma de enviar información al "dongle" y éste no puede entrar en modo de reposo y notificar al "USB host" cuando recibe datos; sin embargo, está el LED que indica conexión con el amplificador. En general este tipo de dispositivos (HID) no requiere de un "driver", simplemente es necesario saber cómo está ordenada la información que envían al "endpoint" para poder interpretarla en la computadora.

Las particularidades relevantes del "dongle" del EPOC™ se expresan en la siguiente tabla:

Característica	Valor hexadecimal	Valor decimal
"id" del fabricante	0x1234	4660
"id" del producto	0xed02	60674
Apuntador al número de serie		3
Versión del dispositivo		0.03
Numero de "endpoints" de entrada		2
	Bytes	bits
Tamaño del paquete de datos "endpoint" 1	8 Bytes	64 bits
Tamaño del paquete de datos "endpoint" 1	32 Bytes	256 bits

Tabla 4.4: Características relevantes del "dongle" del EPOC™

4.1.4. El proceso de adquisición del EEG en el EPOC™

Para poder extraer la información del EEG digitalizado del EPOC™. Primero, es importante conocer el proceso completo de adquisición de las señales. La descripción del proceso comienza a partir de que el amplificador ya está colocado en el usuario (los pasos para la colocación del EPOC™ se pueden consultar en el manual del [26]). *La descripción se enfoca en*

el sistema de adquisición que tiene como resultado el EEG digitalizado en la computadora[1]*, tomando la información disponible en la Subsección 3.2.3 en la página 31:*

> El conocimiento seguro sobre el amplificador era que existían 14 canales de adquisición, con 14 bits efectivos cada uno de ellos (14x14 bits = 196 bits); el muestreo es en forma secuencial con un sólo convertidor analógico digital con un multiplexor que va cambiando de canal adquirido y manteniendo el orden para de esta manera formar un paquete de datos que es envidado al "endpoint" 2 (tamaño del paquete = 256 bits) del "dongle" del EPOC™ cada 7 ms aproximadamente (128 Hz).
>
> También es conocido que estas señales vienen previamente filtradas por el hardware del amplificador en una banda de paso de 0.2-45 Hz y filtros digitales notch a 50 y 60 Hz.

Descifrar cómo están ordenados los paquetes de datos en el "endpoint" del "dongle" USB, fue el proceso equivalente a la creación del "driver" pero sin tener el apoyo de la compañía que fabricó el amplificador, por lo tanto, fue necesario realizar una exploración poniendo en corto circuito cada uno de los canales con las referencias, y al leer los datos, identificar que secciones del paquete estaban en cero. Estos son los pasos que estaban planeados en el proceso para descifrar los datos del EPOC™, pero tras realizar una búsqueda en Internet se encontró que un grupo de personas [29] ya habían extraído los datos de EEG, siguiendo una serie de pasos similares a la explicada, en el lenguaje de programación C y Python en una computadora con Windows™ y Linux™. Lo rescatable para este proyecto del trabajo de [29] fue, descifrar la forma en la que está estructurada la información en los paquetes de datos en el "endpoint" 2 del "dongle"; sin embargo, no se encontró ninguna aproximación en un dispositivo móvil con Android.

4.1.5. ¿Cómo son enviados los datos al "dongle" USB del EPOC™?

Las señales de EEG son adquiridas y amplificadas por los electrodos del amplificador EPOC™ situados en el cuero cabelludo, y de ahí son transmitidas de forma inalámbrica al "dongle". En el trabajo realizado por [29] se explica que la información de la adquisición de EEG en el amplificador, se transmite (no codificada) en el aire en una frecuencia de 2.4 GHz

[1] La descripción de estos pasos está basada en la información oficial disponible por parte de Emotiv™,con base a toda la investigación de información no oficial alrededor del EPOC™ y suposiciones personales para tratar de explicar cómo funciona el sistema de adquisición.

para ser recibida por el "dongle" USB (emparejado de fábrica con el amplificador). En éste son cifradas y transmitidas al circuito integrado de control USB que pasa la información a través del protocolo USB para comunicación humana (HID) a algún dispositivo "USB host" que haya establecido una conexión con el "dongle" y vacíe los paquetes de datos del "buffer" del "endpoint" 2. Los paquetes de datos extraídos del "endpoint" 2 están cifrados, interpretar estos paquetes de datos es el equivalente a la creación de un *"driver"* que descifre la información.

4.1.5.1. ¿Por qué está cifrado el EEG en el EPOC™?

La información contenida en las señales de EEG es preocupación de algunos científicos porque creen que podría extraerse información sensible de cuentas de bancos o contraseñas, usando técnicas específicas en una BCI [30]. Esta podría ser la principal razón por parte de los fabricantes para cifrar la información proveniente de un amplificador de EEG comercial; sin embargo, los amplificadores de investigación entregan la información sin cifrar, lo que me hace pensar que esto está más relacionado con las licencias en las diferentes versiones del EPOC™ y obedece más a una razón comercial, ya que la respuesta al contactar a Emotiv™ para solicitarle la llave y así poder descifrar la información de EEG del "dongle", fue que, ",la forma para tener acceso al EEG crudo (raw EEG) tenía que ser a través de la API de Emotiv™ en el SDK para Windows™". Eso le permite a Emotiv™ cumplir con la promesa de obtener el EEG crudo como lo especificó en las características del amplificador, pero en el ambiente en el que el fabricante decide y no en el que al desarrollador le permita crear su proyecto.

4.1.6. Alternativas para extraer la información del "dongle" del EPOC™ de otra manera:

1. Interceptar la señal inalámbrica

   Como se mencionó anteriormente, la información del amplificador al "dongle" se transmite de forma no segura (sin cifrar), es decir, no está cifrada. Aunque esta primera opción pudiera parecer la más inmediata, no era la más factible ya que hay muchas variables que son prácticamente imposibles de controlar como descubrir cuál es el protocolo de comunicación inalámbrica, además, de que requería hardware especializado e instalaciones libres de ruido electromagnético para poder aislar la señal y así, facilitar el trabajo.

2. Descifrar la información cifrada por el "dongle" una vez extraídos en el "USB host".

   En esta aproximación, ya se dispone del hardware necesario para leer los datos y se eliminan las interferencias de otras fuentes de señal (proceso hecho por el hardware) porque el "dongle" y el EPOC™ vienen vinculador de fábrica y al ser una comunicación digital la pérdida de paquetes de datos está minimizada desde el diseño del amplificador. El proceso de descifrar la información está resulto, en parte, en la documentación de [29]. Es por eso que adaptar esto a Android resultaba una opción más práctica. La comprobación de la obtención del EEG deseado se puedo observar en la sección Subsección 6.1.6 en la página 82 a través de una maniobra que desencadena la generación de ondas alfa en la corteza cerebral.

4.2. Extracción de la información de los paquetes de datos del "dongle" del EPOC™ en una PC

En la sección 4.1.5 en la página 54 se hace una primera aproximación de cómo se pueden extraer los datos del "dongle" USB del EPOC™; ubicando dos posibles opciones, una de ellas requiere la modificación del hardware del amplificador para extraer las señales antes de que estas fueran transmitidas y cifradas. Sin embargo, esta acción restringe la portabilidad de la aplicación, ya requiere ciertas habilidades en electrónica por parte de los usuarios que tengan el hardware disponible y quieran hacer uso de la aplicación. La otra opción fue descifrar las señales disponibles en el "endpoint". Para lograr esto existen diversos métodos que no fueron encontrados en la literatura proveniente de la búsqueda sistemática realizada con anterioridad, por lo que se inició una búsqueda en Internet. En esta búsqueda se encontró información de un proyecto llamado ***emokit***, desarrollado por [29] y colaboradores, los cuales lograron descifrar las señales de EEG del "dongle" del EPOC™ en un proyecto de colaboración en linea en el cual participé.

4.2.1. Cifrado del paquete de datos de 32 Bytes en el "dongle"

En esta sección se explica el flujo *normal* de cómo se protegen los datos hasta llegar a la API de Emotiv™ en Windows™, desde la cual se pueden solicitar como EEG crudo.

El cifrado de los datos es un proceso que se lleva a cabo en el hardware del "dongle" del EPOC™ que recibe la información del amplificador y se encarga de concatenar todos los valores de las lecturas del convertidor analógico digital, armar el paquete de datos, cifrarlo y

enviarlo al “endpoint” 2 del “dongle” en donde es recibido por el “host”, leído por la API en Windows^{TM}, descifrado. Desde allí el desarrollador puede acceder a los datos (ver figura 4.6 en la página 57).

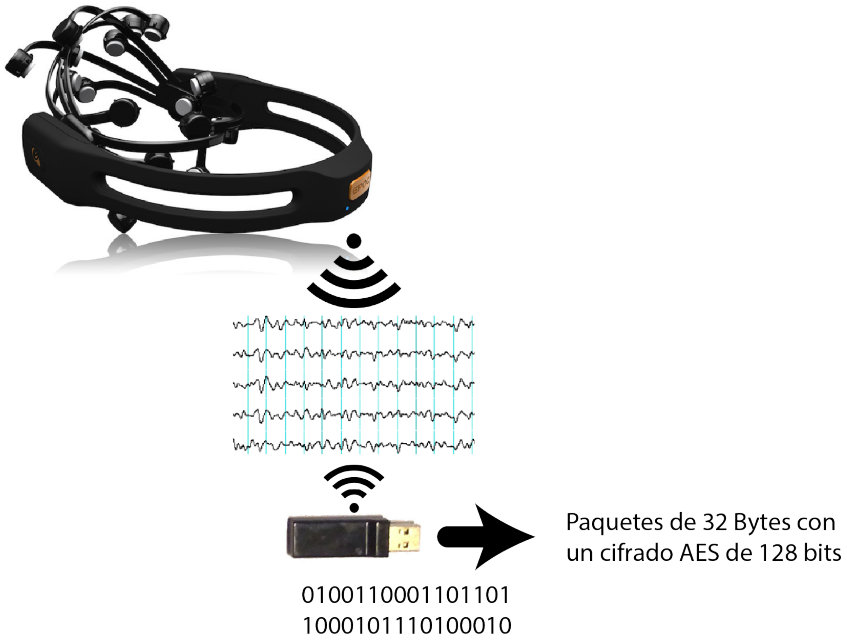


Figura 4.6: Cifrado AES en el “dongle”

4.2.2. Cifrado AES

El cifrado AES (Advanced Encryption Standard por sus siglas en inglés) es el estándar actual en el Departamento de Defensa de Estados Unidos en cuanto a cifrado [31], el que, mediante llaves secretas usa una versión estandarizada del algoritmo de Rijndael. Lo primero, antes de aplicar el algoritmo de cifrado, es definir el tamaño de los bloques de datos; los tamaños permitidos para los bloques de datos son, 128, 168, 192, 224, y 256 bits; y los tamaños aceptados para las llaves son: 128, 192 y 256 bits. El algoritmo más usado de AES usa bloques de datos y de la llave de 128 bits ambos [31]. Este algoritmo está muy optimizado en tiempo, el paso adicional que hay que hacer es partir los datos en bloques de 128 bits.

4.2.3. Cifrado AES en el EPOC™

Los paquetes de datos de 32 Bytes (256 bits) que se pueden leer en el "endpoint" 2 del "dongle" del EPOC™ vienen cifrados en dos partes; es decir, se tiene que partir el paquete a la mitad (128 bits) y usar una llave de 128 bits en cada uno de ellos para poder descifrar el contenido por separado [29] y después volver a concatenar los datos en un paquete de 32 Bytes para poder interpretar la información contenida en ellos.

La razón por la que la información está cifrada de esa manera se puede deber a que el algoritmo de cifrado y descifrado para paquetes y llaves de 128 bits está optimizado, y existen algoritmos que lo hacen en muy poco tiempo.

4.2.4. Cálculo de la llave de descifrado

Hay dos posibles opciones genéricas para el cálculo de la llave de descifrado. La primera es que sea una misma llave para descifrar la información en todos los dongles fabricados por Emotiv™. La segunda, que resultó ser la correcta, es que la llave de descifrado se calcule usando información presente en el "dongle". En este caso el número de serie, información que es única para cada "dongle", solicitado por el "host" mediante una instrucción, debido a que se conoce en donde está almacenado, para así poder calcular la llave de descifrado. El proceso para descubrir como está cifrada la información fue el resultado de la colaboración coordinada por [29] en Github en donde se puede encontrar la explicación y el código en Python para Windows™ o Linux™.

Hay dos tipos de llaves para descifrar la información que se recibe del "dongle", porque hay dos tipos de dongles. Uno para la versión comercial del amplificador y otro para la versión de investigación (que es la que se compró para el desarrollo de este proyecto). Ambos cálculos son muy similares con sutiles diferencias que se describen a continuación.

El número de serie del "dongle" es una palabra de 16 Bytes con la siguiente estructura:

Número de serie SN[]	S	N	X	X	X	X	X	X	X	X	X	X	Y	Y	Y	Y
Numero de Byte	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Tabla 4.5: ¿Cómo está compuesto el número de serie del "dongle"?

En la tabla 4.5 en la página 58 podemos ver la estructura del número de serie donde las X y las Y son usualmente números, sólo los últimos cuatro Bytes son usados para el cálculo de la llave de descifrado [29]. La representación de los números en el número de serie está

en ASCII y se tiene que usar el valor en hexadecimal para construir la llave de descifrado como se muestra en la tabla 4.6 en la página 59. Esos son los valores que hay que poner en la cadena que forma a la llave de descifrado.

ASCII	Hex
0	0x30
1	0x31
2	0x32
3	0x33
4	0x34
5	0x35
6	0x36
7	0x37
8	0x38
9	0x39

Tabla 4.6: Valores en ASCII para armar el número de serie

La llave de 128 bits para descifrar la información del "dongle" de la versión de investigación del amplificador está compuesta como se explica en la tabla 4.7 en la página 59.

Llave de descifrado	SN[0]	0x00	SN[1]	0x54	SN[2]	0x10	SN[3]	0x42
Numero de Byte	15	14	13	12	11	10	9	8
Llave de descifrado	SN[0]	0x00	SN[1]	0x48	SN[2]	0x00	SN[3]	0x50
Numero de Byte	7	6	5	4	3	2	1	0

Tabla 4.7: Estructura de la llave para descifrar la información del "dongle" de la versión de investigación del amplificador

Y para el "dongle" de la versión comercial del amplificador la llave está compuesta como se explica en la tabla 4.8 en la página 60.

4.2.5. Estructura del paquete de 32 Bytes del "dongle"

Ahora que se tiene la información descifrada, el siguiente paso es analizar la estructura de la información de cada paquete recibido en el "USB host" a una frecuencia de 129 paquetes

Llave de descifrado	SN[0]	0x00	SN[1]	0x48	SN[2]	0x00	SN[3]	0x54
Numero de Byte	15	14	13	12	11	10	9	8
Llave de descifrado	SN[0]	0x10	SN[1]	0x42	SN[2]	0x00	SN[3]	0x50
Numero de Byte	7	6	5	4	3	2	1	0

Tabla 4.8: Estructura de la llave para descifrar la información del "dongle" de la versión comercial del amplificador

por segundo. Cada paquete de 32 Bytes contiene la siguiente información (ver tabla 4.9 en la página 61) que es más de la que se había contemplado en el análisis de la sección 4.1.5 en la página 54. Cada sección del paquete de datos se explica con detalle a continuación:

- Un contador de paquete (primeros 8 bits).

  El contador de cada paquete (los primeros 8 bits del paquete) van de cero (00000000 en binario) a 127 (011111111 en binario) y se reinicia la cuenta cada segundo, recibiendo un total de 129 paquetes por segundo.

- Información de la carga de la batería.

  La información de la batería es enviada al "dongle" una vez cada segundo (1 Hz) embebida en el contador de los primeros 8 bits, es decir, cada vez que el bit más significativo del contador está en 1 (1XXXXXXX en binario) los 7 bits menos significativos contienen la información acerca de la carga de la batería como se muestra en la tabla 4.10 en la página 62.

- La lectura de cada uno de los electrodos.

  La información correspondiente a la lectura de cada uno de los electrodos (14 bits cada uno) está en los índices del paquete como se puede observar en la tabla 4.9 en la página 61 y cada lectura tiene la interpretación de un número entero de entre cero a 16383 en una interpretación positiva sin signo o de -8192 a 8191 en una interpretación con signo, considerando que el cero de referencia del amplificador es cuando los 14 bits están en 0. Está interpretación se discutirá más adelante en la sección 6.1.6 en la página 82.

- Calidad del contacto de los electrodos.

Índice de Bits	Información	Tamaño en bits
0:7	Contador / Batería	8
8:21	Datos de F3	14
22:35	Datos de FC5	14
36:49	Datos de AF3	14
50:63	Datos de F7	14
64:77	Datos de T7	14
78:91	Datos de P7	14
92:105	Datos de O1	14
106:120	Calidad de contacto del electrodo*	15
121:133	???	13
134:147	Datos de 02	14
148:161	Datos de P8	14
162:175	Datos de T8	14
176:189	Datos de F8	14
190:203	Datos de AF4	14
204:217	Datos de FC6	14
218:231	Datos de F4	14
232:239	Datos de Gyro X	8
240:247	Datos de Gyro Y	8
248:255	???	8
	Total de bits	256

* Alternando la lectura en relación con el contador

Tabla 4.9: Estructura del paquete de datos del EPOC™

Valor	Nivel de la batería (%)	Valor (continua)	Nivel de la batería (%)
>= 248	100	236	45.93
247	99.93	235	32.34
246	97.02	234	20.43
245	93.40	233	12.37
244	89.45	232	5.08
243	85.23	231	3.63
242	81.89	230	2.80
241	76.77	229	2.05
240	71.54	228	1.42
239	66.59	227	0.88
238	61.92	226	0.42
237	55.37	225	0

Tabla 4.10: Valores para estimar el nivel de carga de la batería

Fuente: Esta información proviene de descifrar los paquetes de datos [29] por lo que puede ser aproximada, ya que no está sustentada por el fabricante del amplificador

La información acerca de la calidad del contacto de los electrodos del amplificador, se encuentra en los bits 106 a 120, pero, como explica [29] la información relevante está solamente en los primeros 13 bits menos significativos (LSB). Sin embargo, esto es considerado un "bug" en la sección de discusión de Github, ya que se hace una aproximación lineal para estimar el cálculo de la calidad del contacto, lo cual en lo particular parece incorrecto, porque de acuerdo a las observaciones en el foro, el máximo valor alcanzado es 8192, pero de acuerdo al esquema del paquete se tienen 15 bits reservados. Además lo correcto sería que estuviera de alguna manera, relacionado con la impedancia. Todos estos temas se abordan más adelante con detalle en la creación de la aplicación para Android. La información relevante para el desarrollo de la aplicación es la información de la calidad de contacto de los electrodos O1 y O2, que como se explicó anteriormente, se envía de manera secuencial varias veces por segundo controlada por medio del valor del contador de los primeros 8 bits del paquete de datos. La relación de los valores de interés se puede observar en la tabla 4.11 en la página 63, lo que indica que hay dos instantes para poder leer la calidad de contacto por electrodo en cada segundo de adquisición.

- Información del giroscopio.

Valor del contador	Calidad de contacto de
6_{10} (00000110_2)	O1
7_{10} (00000111_2)	O2
70_{10} (01000110_2)	O1
71_{10} (01000111_2)	O2

Tabla 4.11: Valor de contador que indica que la información de la calidad de contacto pertenece a O1 y O2.

Fuente: Esta información proviene de descifrar los paquetes de datos [29] por lo que puede ser aproximada, ya que no está sustentada por el fabricante del amplificador

La información acerca del giroscopio, que está integrado en el amplificador, se puede encontrar para el plano XY, dividida en dos vectores de 8 bits, información descrita en cada eje como se muestra en la figura 4.1.5 en la página 54.

Capítulo 5

Diseño de la aplicación en un dispositivo con sistema operativo Android - Metodología 3

Esta fase fue en donde se empezaron a tomar decisiones finales ya que a lo largo de estos capítulos se fueron solucionando los problemas que se presentaron para poder comenzar con el desarrollo del proyecto. En este punto se sabía con seguridad qué tipo de dispositivo se tenía que usar y qué requerimientos de hardware y de software eran indispensables. Por consiguiente, se comenzó con el diseño de la aplicación que por la naturaleza del proyecto se podía programar de diferentes maneras, tratando que fuera lo más sencilla posible para cubrir los requerimientos deseables.

5.1. Requisitos para extraer la información del "dongle" USB en Android

Hay dos tipos de requisitos (hardware y software) para poder extraer la información del "dongle" del EPOC™ con un dispositivo móvil con sistema operativo Android. Requisitos que es indispensable cumplir para que el proceso de extracción de datos se realice con éxito.

5.1.1. Requisitos de hardware para extraer la información del "dongle" USB en Android

Para poder extraer los datos del "dongle" en un dispositivo con Android se tenían dos opciones:

5.1.1.1. Hacer uso de la función *"USB Accessory"*.

Como se discutió en la sección 4.1, los circuitos integrados que controlan la comunicación USB entre dos dispositivos, tienen un propósito específico: son "USB device" o "USB host". No pueden tener ambos roles de forma simultánea.

La gran mayoría de los teléfonos que tienen instalado Android son dispositivos USB que no tienen capacidades de "USB host". Teniendo esto en cuenta, Google en colaboración con Arduino diseñaron un accesorio llamado *Android Development Kit*[1] (ADK ver figura 5.1 en la página 67) basado en la tarjeta de desarrollo *Arduino Mega* que tiene habilitado un "USB host" para comunicarse con teléfonos con Android, y éste a su vez a los puertos de comunicaciones digitales del ADK en donde se pueden conectar otros dispositivos USB. Lo que significa que el ADK funciona como un puente para poder conectar dispositivos USB a un teléfono con Android.

La función *Accesory* está disponible a partir de la versión 3.1 de Android; esta API dispone de una lista definida de instrucciones que se pueden ejecutar desde el dispositivo con Android debido a que el hardware del teléfono no cambia. Por consiguiente, las instrucciones se tienen que enviar como paquetes de respuesta al ADK para que éste a su vez las mande al dispositivo USB conectado. *Esta sería la solución adecuada para poder usar la aplicación en cualquier dispositivo con Android ya que todos los teléfonos tienen un circuito integrado que los habilita como "USB device". Sin embargo, esta solución requiere otra capa de hardware adicional (el ADK) y sería la solución adecuada, si la función USB On The Go no existiera.*

5.1.1.2. USB "On The Go" (OTG)

Como se puede observar en la figura 4.1 en la página 45, a partir de los conectores y receptáculos *Mini* en adelante, todos tienen cinco terminales cuando el estándar tipo A y B sólo tienen cuatro. La razón por la que tienen cinco terminales es para el uso de la función USB OTG, pero para esto, el receptor tiene que tener habilitada esta función en hardware.

[1] http://arduino.cc/en/Main/ArduinoBoardMegaADK

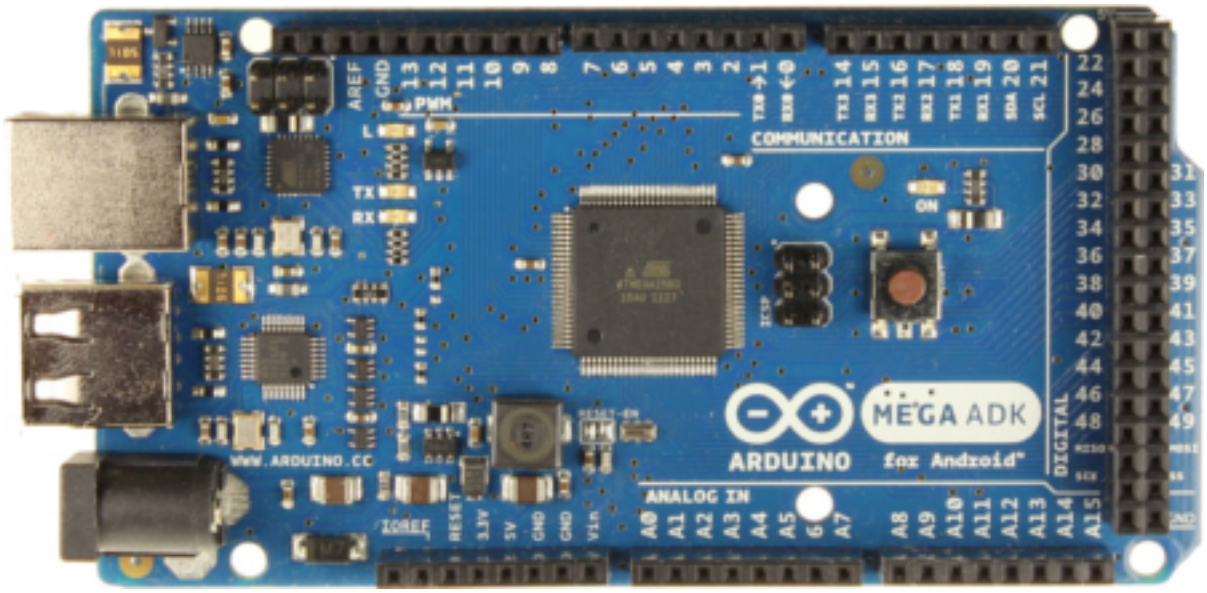


Figura 5.1: ADK de Google y Arduino

Fuente: arduino.cc

Asumiendo que el teléfono tiene la función OTG integrada, la función para poder usar el puerto del teléfono como "USB host", se activa por hardware mediante el uso de un cable especial. Este cable tiene un conector con el estándar Micro-A o Micro-B (ver figura 4.1 en la página 45) de un lado y un receptáculo tipo A en el otro extremo (ver la figura 5.2 en la página 68).

Hay dos tipos de cables USB OTG y se pueden identificar por el lado del conector:

- Los cables con el conector Micro-A.

  Estos cables conectan la terminal 4 (ID) con la terminal 5 (tierra) entre si en corto circuito, es decir conecta la terminal 4 (ID) a tierra, y esto es lo que permite al receptáculo en el teléfono, cambiar su función a "USB host". El hecho de tener la terminal 4 aterrizada lo convierte en un dispositivo OTG A, que le permite recibir energía del receptáculo del teléfono o suministrar energía al mismo, es decir, conectar un cargador externo. Pero la función de "USB host" siempre la tiene el teléfono [3] (ver figura 5.3 en la página 68).

- Los cables con el conector Micro-B.

Figura 5.2: Cable USB OTG micro-B certificado

Estos cables conectan la terminal 4 (ID) con la terminal 5 (tierra) con una resistencia (los valores posibles son 124 KΩ, 68 KΩ, y 36.5 KΩ), esto permite al dispositivo que se conecta en el extremo del cable (receptáculo Tipo A) adquirir el rol de "USB host" al momento de proveer de energía al teléfono, y así controlar la comunicación [3] (ver figura 5.3 en la página 68). Esta configuración está diseñada para alternar el control de la comunicación entre los dos dispositivos conectados solamente controlando la alimentación del puerto.

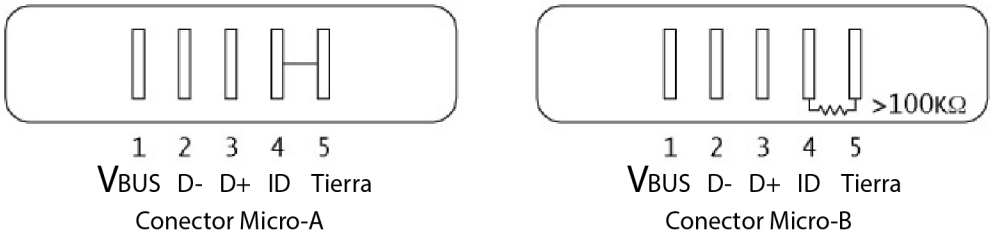


Figura 5.3: Terminales del cable USB OTG

Una vez que se han explicado las características del protocolo USB OTG, es evidente la razón de su uso para el desarrollo del proyecto, ya que no fue necesaria de una capa adicional de hardware que sirva como puente entre la comunicación del teléfono con Android y el

"dongle" del EPOC™. Sólo se necesita un cable, que por un lado tenga un conector Micro-A y del otro lado un receptáculo tipo A para poder conectar el "dongle" y unir la terminal 4 y 5 del lado del conector micro USB. Se utiliza esta configuración debido a que el teléfono siempre controla la comunicación, el "dongle" no tiene la capacidad de funcionar como "USB host" y es por eso que no puede controlar la comunicación. Para que esto funcione, es necesario tener un teléfono que tenga la función USB OTG implementado en hardware.

Sólo una serie de teléfonos con Android, en su mayoría teléfonos de gama alta (con altas prestaciones), tienen integrado el hardware necesario para poder funcionar como "USB host" a través del puerto micro USB disponible en todos los terminales con Android. Esto con ciertas limitaciones, ya que no implementan todas las funciones de un "USB host", pero sí el protocolo HID que es el que se necesita para comunicarse con el "dongle" del EPOC™.

5.1.2. Requisitos de software para extraer la información del "dongle" del EPOC™

Para poder hacer uso de la función USB OTG, además del hardware especial, se debe tener soporte de las funciones en el sistema operativo del teléfono. Android tiene soporte para USB OTG a partir de la versión 3.1 Honeycomb (API level 12) en donde está disponible el API para poder acceder a las funciones del hardware USB OTG en los dispositivos que dispongan de él [32].

Otro de los requisitos de software para poder extraer la información del "dongle", es que el sistema operativo en el que se va a programar tenga la función para descifrar paquetes de datos de 128 bits cifrados usando el algoritmo de Rijndael (o AES) cuando se tiene una llave de descifrado de 128 bits. Este punto no es estrictamente necesario, porque se puede hacer la implementación del algoritmo, pero los algoritmos necesarios para descifrar los paquetes de datos cifrados con AES, están disponibles a partir del API level 1 (la primera versión disponible de Android) y optimizados en lenguaje de bajo nivel, cuestión que puede ser muy importante para hacer el procesamiento de las señales lo más rápido posible.

5.1.3. Dispositivos Android disponibles para el proyecto

Para la realización del proyecto se tenían disponibles dos dispositivos, un teléfono y una tableta, con Android que tienen la funcionalidad USB OTG para conectar un periférico USB mediante el uso de adaptadores. Las característica técnicas de cada uno de ellos se pueden observar con detalle en el Apéndice A en la página 126 de esta tesis.

5.2. Primeros pasos para el desarrollo de la aplicación en Android

Para el desarrollo de una aplicación en Android, es necesario preparar todo el ambiente de desarrollo (IDE) en una computadora, instalando los controladores y todas las herramientas necesarias, que aunque pareciera ser trivial, el no hacerlo de la manera correcta puede resultar contraproducente. No es el objetivo de esta tesis hablar de cómo integrar el ambiente de desarrollo. Sin embargo, es necesario comentar que se comenzó el desarrollo en Eclipse (un IDE para Java), que con la instalación de "plugins" y el SDK de Android, permitía desarrollar aplicaciones. A la mitad del proyecto se migró al entorno de desarrollo oficial para Android, Android Studio, que antes no estaba disponible. Esta aclaración es pertinente porque las imágenes que se mostrarán son imágenes de Android Studio[2]. En la figura 5.4 en la página 70 se muestra el logo de Android Studio.

Figura 5.4: Logo Android Studio

5.2.1. Depuración USB y a través de Wi-Fi

La depuración de comandos en la programación es un paso muy importante porque se pueden vigilar desde el entorno de desarrollo las variables, los procesos que ocurren en segundo plano, las interfaces gráficas y la memoria. La depuración de código normalmente se hace usando una herramienta del ambiente de desarrollo llamado ADB (Android Debugging Bridge) con el teléfono conectado a través del cable USB a la computadora. Pero para esta aplicación en particular, el puerto USB está habilitado como "USB host" con el "dongle" conectado, lo que hace imposible la depuración USB. El ADB tiene la opción de configurar

[2] https://developer.android.com/tools/studio/index.html Consultado en noviembre de 2014

mediante el *shell* la depuración a través de Wi-Fi[3] sin tener el dispositivo con Android conectado con el cable USB, asignando el ADB a la dirección IP del dispositivo que debe estar conectado a la misma red local (se pueden ver los pasos necesarios siguiendo la liga en el pie de página).

Esta alternativa resultó muy útil para el desarrollo de la aplicación en este proyecto. Sin embargo, la depuración es más lenta, y observar los valores de los datos que se recibían del "dongle" se volvió muy complicado, razón por la que se optó por probar el debugging de la aplicación con el "dongle" conectado al puerto USB del *dock* con el teclado de la tableta ASUS, ya que en ésta queda disponible el puerto para conectar la tableta a la computadora y poder hacer debugging en tiempo real.

5.3. Aplicaciones en Android

No es el propósito de esta tesis profundizar en la forma en la que están estructuradas las aplicaciones en Android, en las interfaces de la APIs o en los lenguajes de programación que fueron conocimientos adquiridos por este autor para desarrollar la Android BCI. Sin embargo, sí hay conceptos que son fundamentales para entender la forma en la que está realizada la aplicación, conceptos que dan una idea general de la estructura y fundamentan el por qué la aplicación fue realizada de esa manera. Todo esto se puede observar en el Apéndice B en la página 132 en donde se abordan de manera muy práctica los conocimientos necesarios para la creación de la aplicación en Android.

[3] WirelessADB - https://developer.android.com/tools/help/adb.html#wireless Consultado en noviembre 2014

Capítulo 6

Resultados

El resultado del trabajo del proyecto es la creación de una aplicación en Android que presenta los estímulos SSVEP al usuario, extrae las señales de EEG del "dongle" del EPOC™, procesa la información y determina a qué contacto realizar la llamada de entre dos posibles.

Todos los comentarios del código están en ingles debido a que este código de la aplicación se hizo público en Github para que cualquier persona que quiera hacer uso de él y contribuir al proyecto pueda hacerlo.

6.1. Android™ BCI

El nombre de la aplicación desarrollada es: Android™ BCI, un nombre que trata de explicarse solo. En un inicio, se pensó en nombre de EPOC™ BCI, pero al ser EPOC™ una marca registrada, no se podía hacer uso de ella para una aplicación distribuida de manera libre. Sin embargo, la palabra Android se puede usar en el nombre de la aplicación siempre y cuando se agregue el símbolo ™ después de ella, como lo indica el aviso legal de la marca[1] El ícono de la aplicación se puede ver en la figura 6.1 en la página 74.

Por sí solo, el ícono de la figura 6.1 en la página 74, trata de reflejar el nombre de la aplicación al estar compuesto por un cerebro en color verde, queriendo hacer referencia a Andy, el robot característico del sistema operativo Android, y el acrónimo BCI de *Brain Computer Interface* que en la lista de aplicaciones del dispositivo móvil aparece con el nombre

[1] Legal Notice: https://developer.android.com/legal.html Consultado en noviembre de 2014

Figura 6.1: Ícono de la Android™ BCI

"Android™ BCI" debajo del ícono. Se tenían otras alternativas; sin embargo, este ícono fue producto de una colaboración con *Pamela Martínez Aguilar* (artista plástica a quien otorgo los créditos del ícono) resultando un diseño más profesional que los íconos diseñados por el autor de esta tesis.

6.1.1. Inicio de la Android™ BCI

Existen dos maneras de iniciar la Android™ BCI, que se explican en la figura 6.2 en la página 75, que llevan a la misma "activity" principal y en ambas se verifica que el "dongle" esté conectado. De lo contrario, se manda un mensaje en pantalla al usuario avisándole que la aplicación no puede funcionar sin él. Si esto sucede al dar click en el botón "OK" del mensaje, se cierra la aplicación.

6.1.2. "Manifest" de la Android™ BCI

El "manifest" de la Android™ BCI es el lugar en donde se registran todas las actividades y se establecen los permisos que se deben solicitar al usuario para utilizar la información que está restringida hasta que se cuente con la autorización para su uso. La mejor manera de explicar el "manifest" es explicar el código, omitiendo las secciones que son comunes a todas las aplicaciones; es decir, solamente las secciones que corresponden a permisos propios de la Android™ BCI (ver la Sección de código 14 en la página 141 en el Apéndice C).

En el "manifest", además de obtener los permisos para, acceder a los contactos, poder utilizar el puerto USB, fijar la pantalla a modo horizontal ("landscape"), almacenar datos

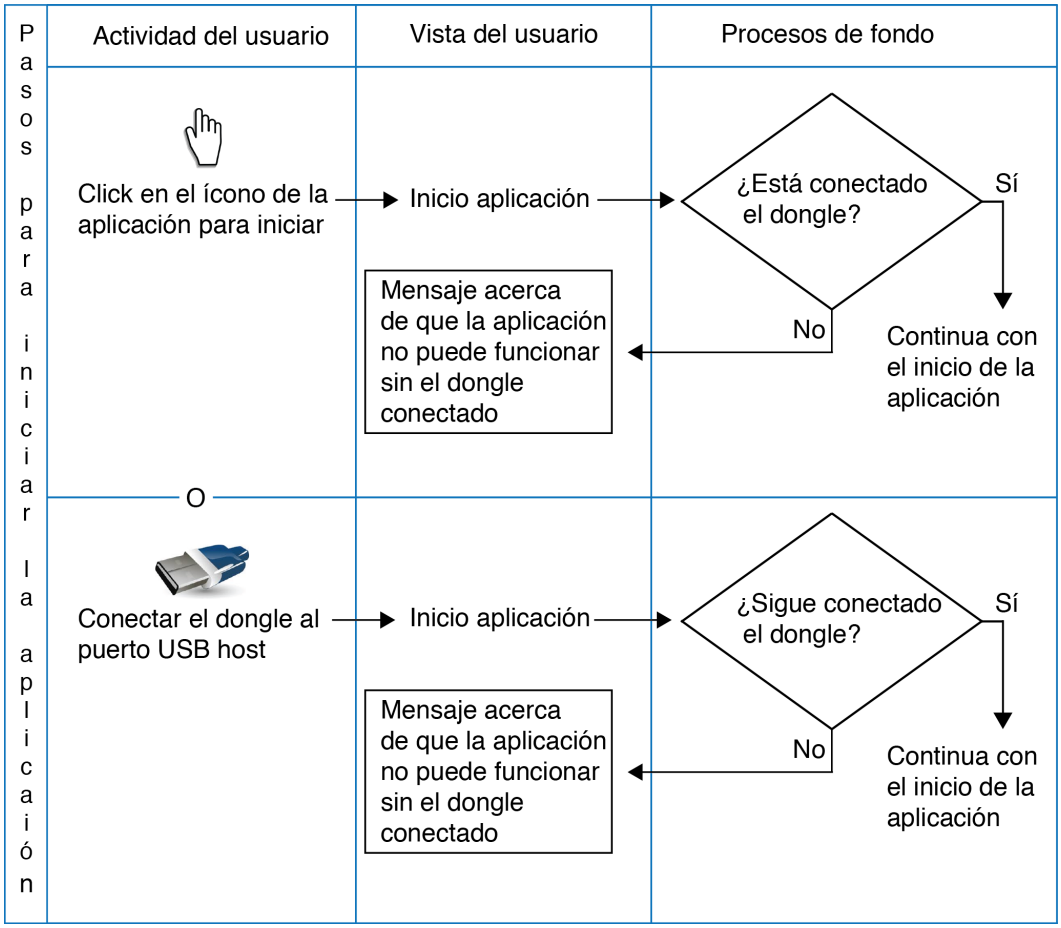


Figura 6.2: Formas de iniciar la Android™ BCI

en el teléfono, y poder realizar llamadas telefónicas, se encuentran registrados dos "intent-filters", uno, que cuando el sistema operativo detecte que se ha conectado un dispositivo USB que coincida con el "id" del fabricante y el "id" del dispositivo del "dongle" del EPOC™, la "activity" principal de la aplicación Android™ BCI se ejecute de manera automática; y otro que notifique de su desconexión.

Estas líneas de código y la información en el archivo de meta-data *epoc_ dongle.xml*, permiten cumplir con uno de los requerimientos de diseño de la Sección B.3 en la página 135. El contenido del archivo *epoc_ dongle.xml* se puede observar en la Sección de código 1 en la página 76 . El contenido es simple porque sólo contiene el "id" del fabricante y el "id" del dispositivo, que son identificadores de hardware presentes en todos los dongles de EPOC™. Esto permite que la aplicación no se ejecute cada vez que se conecta cualquier dispositivo USB al puerto del teléfono; y es muy útil, debido a que si se está conectando el "dongle" al teléfono es porque se quiere usar la aplicación y evita el tener que buscarla en el menú de aplicaciones.

```
<?xml version="1.0" encoding="utf-8"?>

<resources>
    <usb-device vendor-id="4660" product-id="60674" />
</resources>
```

Sección de código 1: Archivo epoc_dongle.xml asociado al "manifest"

Como se puede ver en la Sección de código 1 en la página 76, los valores no corresponden con los valores descritos en el capítulo 4 en la página 43, esto es porque los valores allí descritos están en hexadecimal y en el archivo epoc_dongle.xml se tienen que poner forzosamente en su valor decimal.

6.1.3. Desconexión física del "dongle" por el usuario

Es muy importante detectar cuando el "dongle" se conecta, pero es más importante aún saber cuándo el "dongle" se desconecta de manera repentina, porque se puede interferir con el desempeño del sistema en aplicaciones que dependen de la respuesta de hardware. Es por eso que una de las secciones más importantes del código es un "listener" que permite escuchar las notificaciones del sistema que informan acerca de las desconexiones. El sistema, cuando ocurre la desconexión repentina de un dispositivo USB, no sabe cual es el dispositivo USB que se desconectó, y es por eso, que ante una desconexión hay que hacer una búsqueda de cuales son los dispositivos que siguen conectados, y en caso de que no se encuentre el EPOC™, la aplicación inmediatamente manda un mensaje al usuario notificando de la desconexión para posteriormente cerrar la aplicación. Las consecuencias de no hacerlo, son que se puede provocar un volcado de memoria y reiniciar el teléfono, suceso que se ha comprobado en la experiencia en el desarrollo de esta aplicación. El código asociado a esta acción se puede ver en la Sección de código 16 en la página 143 en el Apéndice C.

6.1.4. Estado de conexión del "dongle"

En la Sección de código 16 en la pagina 143 se hace uso de un método creado para la aplicación que hace uso del aviso ("broadcast") de sistema que nos permite avisar al usuario si el "dongle" está conectado. Este método regresa un valor lógico (verdadero o falso) en el mensaje que a su vez permite modificar el estado actual de la conexión del "dongle" en la aplicación y fue diseñado de esta manera, para que el saber si el "dongle" está conectado, sea tan fácil como hacer una pregunta en el estado actual del "dongle". La descripción del método se puede ver en la Sección de código 21 en la página 148 en donde están todos los pasos necesarios para buscar el "dongle" en la tabla "hash" de dispositivos e iniciar la conexión. Para que esto fuera posible se tuvo que hacer uso de un recurso en Java™ llamado *enumeración* que define, en este caso, estados de una variable de forma personalizada; es decir, se definieron estados del "dongle" como: conectado, "pending" y desconectado (ver la Sección de código 17 en la página 144). Todas las secciones de código mencionadas se pueden consultar en el Apéndice C.

De la misma manera, se diseñaron enumeraciones para el estado de la interfaz USB y la conexión de datos del "dongle", debido a que son estados que se deben preguntar constantemente en otras secciones del código de la aplicación (ver Sección de código 18 en la página 145 en el Apéndice C).

Las enumeraciones por sí solas están incompletas, por lo que es indispensable hacer métodos llamados *getters y setters* para poder acceder y establecer el valor del estado en el que se encuentran. En este caso, el estado general del "dongle", el estado de la interfaz y el estado de la conexión de datos. Por esta razón, se diseñaron los métodos para actualizar los valores de las enumeraciones en la Sección de código 19 en la pagina 146 se describen los "getters" de las enumeraciones, y en la Sección de código 20 en la pagina 147 se describen los "setters" de las mismas.

6.1.5. Lectura de los datos del "dongle" en la Android™ BCI

Hasta ahora, se han establecido todos los pasos previos para hacer la lectura de los paquetes de datos del "dongle" con la mayor seguridad posible y con la creación de métodos para evitar la confusión del estado general en el que se encuentra, que está conformado por diversas variables que permiten iniciar la lectura de los datos en el "endpoint".

La lectura de los datos del "dongle" se tiene que hacer en las dos actividades de las que está compuesta la Android™ BCI, porque en la "activity" principal se tiene que mostrar la calidad del contacto de los electrodos O1 y O2, información que proviene del "dongle", pero cuando la "activity" que hace la estimulación visual pasa al primer plano de la pantalla, la "activity" principal pasa al estado On Stop, lugar en el que hay que liberar la conexión USB con el "dongle" y permitir la lectura de los paquetes de datos para hacer el cálculo de la potencia en la "activity" que hace la estimulación visual.

Los "intent-filters" del "manifest" permiten a la aplicación escuchar los mensajes del sistema que avisan de la conexión y desconexión de dispositivos USB, pero escuchar los mensajes no es suficiente, hay que hacer algo cuando sean recibidos. En el código en Java™ de la Sección de código 16 en la página 143, se muestran los pasos que se deben seguir para recibir los" intents" de la conexión del "dongle" al puerto USB, siendo necesario colocar estas secciones de código en las activities que van a extraer los paquetes de datos del "dongle".

Los pasos que empiezan la comunicación con el "dongle" se explican en la Sección de código 21 en la página 148. Esta serie de pasos se hacen en diversas secciones de la "activity" o en diferentes estados; no se ejecutan necesariamente de manera secuencial y sólo sirven para ejemplificar la forma en la que se leen los datos del "dongle". Lo primero que se hace para leer los datos del "dongle" es, crear un objeto de la Clase *USBmanager*, el cual permite al "intent" detectar la conexión del "dongle" del EPOC™ dentro de la aplicación. A este "intent" se le hace un cast a *UsbDevice* para guardar el objeto en la variable *epocUSBDevice*. Hay dos formas de obtener este objeto:

```
//Implicit Intents to communicate with the system or external applications
//Intent made to be able to detect the object that represents the epoc dongle
//when connected
Intent epocConnectedIntent = new Intent();
UsbDevice epocUSBDeviceIntent =
(UsbDevice) epocConnectedIntent.getParcelableExtra(UsbManager.EXTRA_DEVICE);
```

Sección de código 2: Código que detecta la conexión del "dongle" en el dispositivo móvil

1. Obtener la referencia a este objeto del listener global que ejecuta la aplicación de manera automática cuando el "dongle" es conectado al dispositivo (ver Sección de código 2 en la página 79). La existencia de esta referencia (en caso de que exista) es asignada al UsbDevice en el estado onResume de la aplicación.

2. Obtener la referencia a este objeto de la búsqueda en la tabla hash de dispositivos USB conectados al teléfono como se muestra en la Sección de código 21 en la página 148. Este escenario se puede dar en el caso de que se haya conectado previamente el "dongle" y la aplicación se inicie del menú de aplicaciones del teléfono.

Una vez que se define si el "dongle" está conectado al dispositivo, se extrae el número de serie, se genera la llave de descifrado y se inicia la lectura de datos de la calidad de contacto de los electrodos O1 y O2 para actualizar los cuadros que muestran la calidad de la señal con un código de colores (esto se explica con detalle más adelante), de lo contrario (en caso de que el "dongle" no se encuentre conectado al dispositivo), se muestra un mensaje al usuario informando que el "dongle" no está conectado, pero se permite la interacción con la pantalla principal de la aplicación.

Todas las secciones de código referenciadas están disponibles para su consulta en el Apéndice C.

6.1.5.1. Generación de la llave de descifrado

El primer paso, cuando se conecta o se busca el "dongle" y se encuentra, es extraer el número de serie como se muestra en la Sección de código 3 en la página 80 en donde se inicia la conexión con el dispositivo, se obtiene el número de serie y se cierra la conexión. Esto se hace para que cuando se quiera usar el puerto éste se encuentre disponible.

```
epocUsbConnection= USBmanager.openDevice(epocUSBDeviceIntent);
Log.i(DEBUG_TAG, "Epoc dongle found and change status to connected and USB assigned");
setSerial(epocUsbConnection.getSerial());//Extract and set the serial number
epocUsbConnection.close();//Just extract the serial and then close the connection
```

Sección de código 3: Código que extrae el número de serie del "dongle".

Posteriormente a la obtención del número de serie, éste se envía al método que calcula la llave de descifrado y regresa un valor lógico que indica si el proceso se realizó con éxito. Este es un método de comprobación que no permite avanzar en el código si la llave no se genera de manera exitosa, porque de eso depende que la interpretación de los datos de EEG sea la correcta. Lo anterior se puede observar en el "Sí" de la Sección de código 23 en la página 150 en el Apéndice C donde también se hace la asignación a la bandera que establece si la llave ya fue generada con éxito.

6.1.5.2. Extracción de los datos del "dongle" en la Android™ BCI

La extracción de los datos del "dongle" tiene dos propósitos: mostrar la calidad de contacto de los electrodos O1 y O2, y extraer el EEG para procesarlo; ambos requieren de varios pasos previos para iniciar la lectura descritos a continuación.

Para que se pueda iniciar la lectura de los datos se tiene que reservar un espacio de memoria para almacenar los paquetes de 32 Bytes provenientes del "buffer" de datos cifrados del "dongle" y un espacio de memoria igual para los paquetes descifrados, además, debe reservarse el acceso a la interfaz que manda los paquetes de datos. Esto se puede observar en la Sección de código 24 en la página 151 en el Apéndice C.

Cuando la interfaz está reservada, la lectura de los datos se hace con un ciclo while que extrae los datos del "buffer" cuando éstos se encuentran disponibles, en donde son descifrados para poder hacer uso de ellos cuando sea necesario. Todos estos pasos pueden producir muchas excepciones en tiempo de ejecución, y es por eso que es necesario manejar cada una de ellas con un arreglo de try-catch para cada caso que pueda ocurrir. Esto se puede observar en la Sección de código 25 en la página 152 en el Apéndice C.

6.1.5.3. Descifrar los datos del "dongle" en la Android™ BCI

El cálculo de la llave de descifrado se lleva a cabo en el estado OnCreate de la "activity" principal y es la única vez que se calcula, ya que ésta no cambia nunca. Para poder calcularla

es necesario pedir el número de serie del "dongle" una vez que ya se está establecida la conexión y antes de recibir los paquetes de datos. Estos pasos ya han sido explicados anteriormente, por consiguiente, descifrar la información del "buffer" se resume a las lineas de código dentro del *try* de la Sección de código 25 en la página 152 en donde se tiene que crear un objeto de tipo *Cipher* indicando el tipo de cifrado que tienen los datos, pasar la llave de descifrado, previamente generada a éste, y por último recibir el paquete de datos descifrados de 32 Bytes como resultado de enviar el paquete cifrado con la instrucción cipher.doFinal(). En este paso pueden ocurrir muchas excepciones en tiempo de ejecución que es importante atender.

6.1.5.4. Interpretación de los paquetes de datos en la Android™ BCI

La importancia de saber el orden de los datos en los paquetes ya descifrados y recibidos en el "buffer" del "dongle", es necesaria para poder hacer la correcta interpretación de la información contenida en ellos. El orden en el que viene empaquetada la información se puede ver en la tabla Tabla 4.9 en la página 61, para ello se tuvieron que crear máscaras que extraen los datos que están separados en segmentos de 1 Byte (8 bits) mediante corrimientos en los Bytes que contienen la información. Para la extracción de un valor se hace uso de un máximo de 3 Bytes de origen, que es el máximo número de Bytes en los que puede estar contenido un dato de 14 bits. Es importante mencionar que este es un método optimizado creado en este proyecto, y es un aporte a la comunidad que usa el EPOC™ porque en el código original de Cody Brocious [29] extraen los datos con ciclos *if* que recorren bit a bit la información, y en este caso, en un máximo de 3 operaciones lógicas optimizadas se obtiene el mismo resultado.

Información de la señal de O1 y O2 en el paquete de datos

En la Sección de código 4 en la página 82 se extrae la información de los Bytes 11, 12 y 13 del paquete ya descifrado de 32 Bytes proveniente del "buffer" del "dongle" en un instante de tiempo, se hace corrimientos a la derecha y a la izquierda (según sea necesario) de los Bytes originales y después se concatenan con la operación lógica **"OR"** en un tipo de dato "short" (tiene que ser así porque si no se genera una incompatibilidad de tipos de datos). Posteriormente, se convierte el valor entero de la medida del electrodo en microvolts sólo para darle una interpretación de señal eléctrica con base en la resolución del convertidor analógico digital, y finalmente se almacenan en una matriz de datos.

Lo mismo sucede para los datos del electrodo O2 con los Bytes 16, 17 y 18 como se puede observar en la Sección de código 5 en la página 82. Éstas son las dos lecturas importantes

```
//O1
short epocO11= (short)((decryptedData[11]<<10) & 0x00003C00);
short epocO12= (short)((decryptedData[12]<<2)  & 0x000003FC);
short epocO13= (short)((decryptedData[13]>>>6) & 0x00000003);

epocO1= (short) (epocO11|epocO12|epocO13);//OR to concatenate then in one short
//Convert to micro volts and store in the RAW EEG matrix
EEG[6][N]=((double)(epocO1))*0.00000051;
```

Sección de código 4: Extracción de datos del electrodo O1 en el "dongle" en la Android™ BCI

para el proyecto. *Sin embargo en el código se almacenan en la matriz de datos las lecturas correspondientes a los catorce electrodos del EPOC™ para que se pueda hacer uso del código en aplicaciones que requieran información de otros electrodos en otros proyectos*, como se puede observar en la figura 6.1 en la página 84 en donde se muestran los paquetes de datos de 32 Bytes descifrados correspondientes a 33 lecturas del "buffer" de datos del "dongle".

```
//O2
short epocO21= (short)((decryptedData[16]<<12) & 0x00003000);
short epocO22= (short)((decryptedData[17]<<4)  & 0x00000FF0);
short epocO23= (short)((decryptedData[18]>>>4) & 0x0000000F);

epocO2= (short) (epocO21|epocO22|epocO23); //OR to concatenate then in one short
//Convert to micro volts and store in the RAW EEG matrix
EEG[7][N]=((double)(epocO2))*0.00000051;
```

Sección de código 5: Extracción de datos del electrodo O2 en el "dongle" en la Android™ BCI

6.1.6. Resultado de la lectura de los datos del "dongle" en una aplicación con Android

Antes de comenzar con el diseño de la aplicación, el paso más importante es corroborar que se puedan leer las señales de EEG crudo en el dispositivo con Android y comprobar que

los paquetes de datos provenientes del amplificador estaban siendo descifrados e interpretados de manera correcta.

Como primera aproximación de prueba rápida para leer los datos del "endpoint" 2 del "dongle" se escribió una aplicación con Android sin interfaz gráfica con el único propósito de comprobar que se podían recibir datos del "dongle", extraerlos y descifrarlos con el algoritmo diseñando. Comprobar que estos datos son el EEG proveniente del EPOC™ es una tarea más compleja para la que se diseñó una prueba que se explica más adelante.

En esta primera prueba se coloqué el EPOC™ en mi cabeza siguiendo las instrucciones del fabricante, verifiqué en la aplicación de Windows™ proporcionada por el fabricante que la calidad de la señal de los electrodos fuera adecuada y observé las bandas de potencia de EEG para determinar de acuerdo a mi experiencia si existía algo inusual. Al no ser así, sin hacer cambios desconecté el "dongle" de la computadora con Windows™ y lo conecté en el dispositivo móvil para después de ejecutar la aplicación de prueba que lee los datos del "dongle", los descifra, los ordena y los guarda en un arreglo matricial un fragmento de 33 muestras del resultado se puede observar en la tabla 6.1 en la pagina 84. Los datos que se muestran en la tabla ya han sido ordenados y etiquetados de acuerdo al canal correspondiente del amplificador, sin ningún procesamiento. Los datos se encuentran en un valor esperado para la conversión analógico digital de 14 bits, y algo que hay que resaltar de la tabla es que en la primera columna se puede observar el contador, que en este caso comienza en el valor 100 y se reinicia al llegar al valor 127, después da un valor negativo (valor de la carga de la batería), a cero. Esto corresponde perfectamente a la descripción que se había hecho del hardware del "dongle" hasta ahora. A pesar de que los datos fueron ordenados por canal, es necesario hacer otro tipo de análisis para poder comprobar que es el EEG que se busca, el dato del contador no es suficiente para asegurar que los datos están descifrados.

Ya que resulta prácticamente imposible saber si las lecturas correspondientes a los electrodos corresponden a EEG sólo de observar los datos crudos. Por tal motivo, se diseñó un protocolo de adquisición para comprobar las señales provenientes del "dongle" fueran el EEG que se buscaba.

El protocolo consistió en lo siguiente:

1. Colocar el amplificador en el sujeto y comprobar la calidad de contacto en una computadora con Windows y con el SDK de Emotiv™ instalado (que muestra la calidad de contacto de los electrodos en la pantalla).

2. Una vez verificada la calidad de la señal desconectar el "dongle" la computadora y conectarlo al dispositivo con Android.

Contador	F3	FC5	AF3	F7	T7	P7	O1	O2	P8	T8	F8	AF4	FC6	F4	giroX	giroY
100	8332	8158	8547	8637	9012	8236	8834	9094	8039	8135	9266	8131	8549	8805	102	108
101	8331	8157	8548	8639	9019	8236	8834	9084	8045	8139	9267	8136	8545	8805	102	108
102	8333	8159	8558	8647	9034	8245	8849	9091	8051	8142	9277	8150	8552	8796	102	108
103	8334	8161	8559	8653	9031	8251	8855	9102	8056	8138	9276	8145	8556	8792	102	108
104	8333	8163	8554	8650	9029	8259	8858	9106	8058	8142	9261	8143	8556	8803	102	108
105	8336	8162	8558	8651	9025	8255	8856	9106	8051	8166	9265	8155	8556	8805	102	108
106	8341	8162	8564	8656	9018	8249	8843	9102	8052	8181	9290	8155	8562	8799	102	108
107	8346	8163	8566	8651	9038	8258	8838	9097	8063	8169	9296	8148	8570	8805	102	108
108	8343	8162	8563	8642	9047	8263	8845	9099	8065	8152	9279	8145	8565	8812	102	108
109	8333	8159	8557	8639	9008	8255	8845	9099	8050	8154	9276	8147	8549	8806	102	108
110	8335	8158	8559	8635	8989	8243	8833	9090	8035	8155	9283	8148	8547	8800	102	108
111	8340	8161	8561	8635	9024	8242	8830	9083	8037	8144	9285	8143	8565	8799	102	108
112	8340	8164	8559	8639	9042	8257	8848	9093	8053	8153	9284	8143	8574	8800	102	108
113	8335	8163	8560	8640	9009	8269	8859	9111	8066	8170	9281	8145	8561	8806	102	108
114	8330	8161	8562	8639	9006	8265	8844	9107	8063	8163	9282	8143	8550	8803	102	108
115	8336	8160	8560	8638	9042	8266	8847	9100	8057	8153	9289	8149	8556	8797	102	108
116	8344	8160	8559	8640	9030	8269	8860	9111	8052	8157	9291	8159	8566	8805	102	108
117	8338	8160	8560	8646	8996	8258	8850	9110	8042	8156	9285	8154	8568	8812	102	108
118	8332	8158	8561	8642	9017	8256	8845	9098	8035	8150	9280	8147	8562	8802	102	108
119	8337	8159	8561	8626	9037	8266	8859	9100	8036	8148	9273	8146	8553	8799	102	108
120	8339	8161	8562	8625	9018	8267	8860	9110	8035	8142	9267	8142	8548	8805	102	108
121	8336	8160	8562	8643	9014	8263	8844	9111	8032	8144	9273	8147	8554	8799	102	108
122	8333	8160	8561	8647	9014	8265	8845	9096	8029	8159	9282	8150	8558	8794	102	108
123	8330	8161	8559	8637	9000	8271	8860	9083	8027	8158	9280	8141	8549	8798	102	108
124	8335	8161	8558	8643	9011	8279	8864	9096	8026	8134	9274	8145	8547	8800	102	108
125	8335	8160	8560	8652	9040	8279	8860	9115	8034	8118	9274	8152	8556	8798	102	108
126	8330	8161	8559	8649	9047	8264	8857	9108	8042	8133	9280	8148	8556	8797	102	108
127	8336	8162	8561	8644	9033	8261	8856	9096	8047	8148	9285	8149	8558	8795	102	108
-23	8344	8162	8562	8638	9014	8266	8845	9095	8044	8144	9278	8151	8566	8805	102	108
0	8339	8164	8563	8627	8990	8250	8836	9083	8030	8144	9274	8146	8560	8817	102	108
1	8338	8162	8561	8622	8989	8239	8838	9077	8020	8150	9280	8144	8548	8811	102	108
2	8341	8159	8558	8618	9011	8250	8845	9092	8022	8143	9272	8140	8549	8801	102	108
3	8338	8157	8558	8612	9015	8248	8838	9093	8023	8126	9257	8133	8548	8801	102	108

Tabla 6.1: Segmento de datos crudos del "endpoint" 2 del "dongle" del EPOC^{TM}

3. Comenzar la adquisición de EEG en el dispositivo con Android.

4. El sujeto con el amplificador colocado debe hacer un movimiento lateral de izquierda a derecha para observar los datos del giroscopio en el eje X.

5. De igual manera posteriormente al paso previo, hacer un movimiento vertical de arriba a abajo para observar los datos del giroscopio en el eje Y.

6. Una vez que el sujeto regresa la cabeza a la posición original (la posición central) se hacen diez segundos de registro con los ojos abiertos, sin moverse.

7. Cerrar los ojos para aumentar la actividad de las ondas alfa en el EEG y registrar otros 10 segundos para tener un tiempo total de adquisición (así programado) de aproximadamente 30 segundos.

8. Hacer un análisis en frecuencia de los datos a dquiridos (que es la medida que se va a usar en la BCI) para ver que evidentemente esté aumentada la actividad de las ondas alfa (aproximadamente 10 Hz).

Los resultados del ejercicio anterior fueron exitosos como se muestran en la figura 6.3 en la página 86 en donde se pueden observar cuatro gráficas; en las dos primeras gráficas de

potencia de los electrodos O1 y O2, elegidos porque son los que se van a usar en la adquisición final, se puede ver claramente que cuando los ojos están abiertos (línea verde), el espectro de frecuencias es más plano, y cuando los ojos están cerrados se incrementa notablemente la potencia alrededor de los 10 Hz (banda de las ondas alfa). Este fenómeno no se puede apreciar muy bien, aunque sí está presente en el electrodo T8 que está más alejado de la zona occipital. Por último, podemos ver 6 segundos del registro de la variable del contador (los primeros 8 bits del paquete de datos) que va de 0 a 127 y se reinicia. Con esto, se puede confirmar que la información que está siendo extraída del "dongle" del EPOC^{TM} es el EEG que se buscaba. De no haber obtenido los resultados buscados en el primer ejercicio con el protocolo anterior, posiblemente se hubiera tenido que modificar y hacer de nuevo, pero para el propósito de hacer una verificación del contenido de EEG en las señales, se consideró que no era necesario hacer más experimentos.

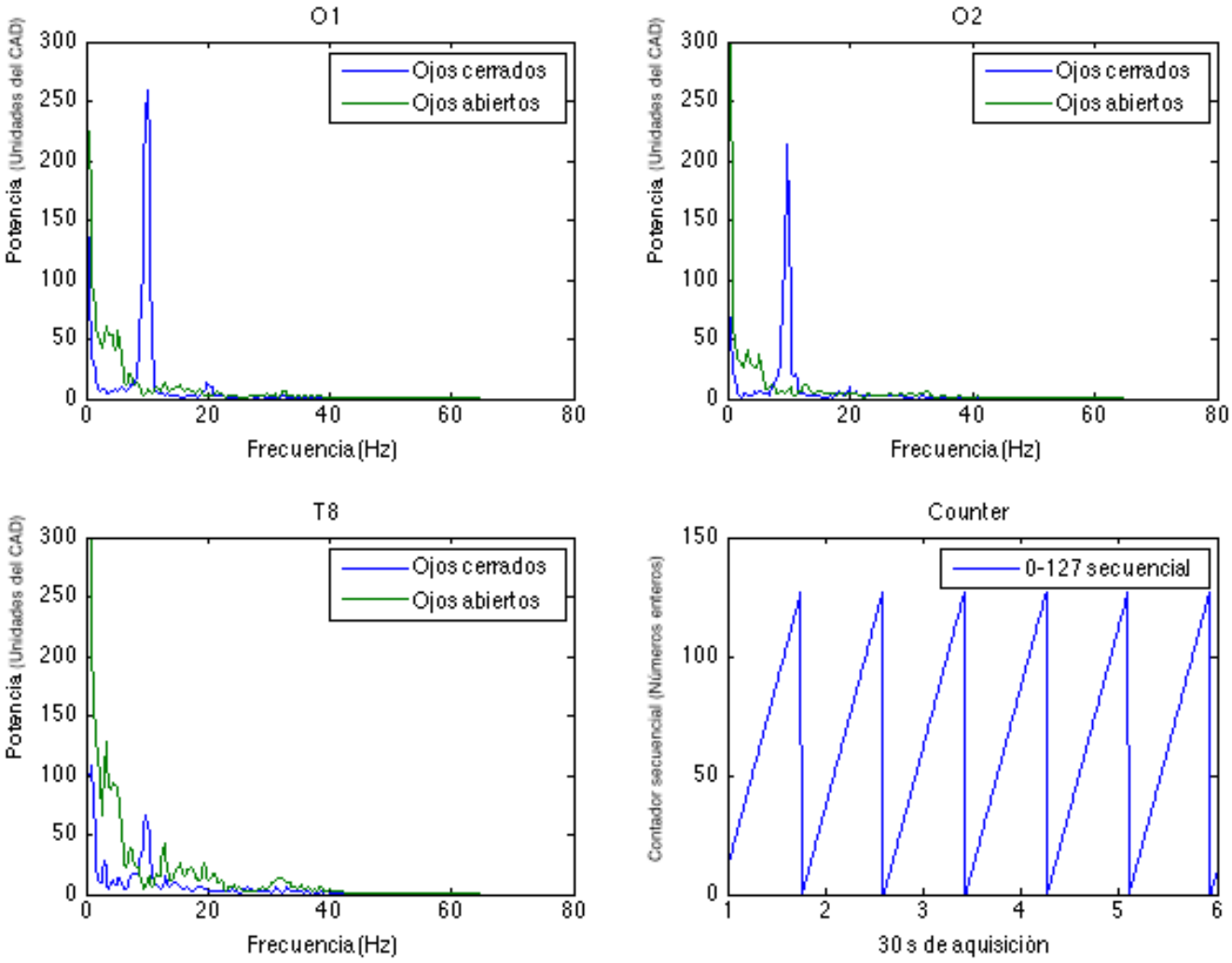


Figura 6.3: Análisis en frecuencia y contador del "dongle"

6.2. Resumen del flujo de acciones en la Android™ BCI

Hasta ahora, se ha mostrado cómo realizar varios de los procesos necesarios para completar el objetivo de la aplicación, independientemente de cómo se realicen estas acciones el flujo de instrucciones de forma simplificada en forma de algoritmo se reduce a los pasos en las siguientes dos subsecciones del ciclo de vida de las dos actividades que componen a la Android™ BCI.

6.2.1. "Activity" principal de la Android™ BCI

En esta "activity" es en donde se lleva a cabo la lectura de los datos del "dongle" para mostrar la calidad del contacto de los electrodos O1 y O2, también aquí es en donde el usuario tiene la capacidad de interactuar con la pantalla y seleccionar a cuáles son los contactos que desea llamar haciendo click sobre el cuadro *search contact*, iniciando una lista desplegable de selección de los contactos almacenados en el teléfono.

El resumen detallado del flujo de las acciones que se llevan a cabo en cada estado por el que pasa la "activity" principal se puede ver en el Apéndice D

6.2.2. Diseño de la "activity" principal

La "activity" principal fue pensada como la pantalla principal de la aplicación, desde la cual el usuario interactúa con todos los elementos que permiten hacer la llamada telefónica a uno de dos contactos. Contiene los recuadros para seleccionar los contactos a los que se va a llamar, además de que muestra los números de teléfono asociados con el contacto. En los recuadros grises aparece la foto del contacto que el usuario haya seleccionado (en caso de que exista). En la parte inferior de la "activity" principal en dos recuadros se muestra la calidad de contacto de los electrodos O1 y O2 relacionada con la calidad de señal. En la figura 6.4 en la página 88 se puede ver la distribución de los elementos en el teléfono Galaxy Nexus que se usó para el proyecto y se pueden apreciar todos los elementos con claridad. Cabe mencionar que no se muestra el botón de inicio de la estimulación visual, tema que se discutirá más adelante.

El diseño tuvo como inspiración un entorno simple en el que entre menos elementos en la pantalla mejor, debido a que es un aplicación con un uso muy específico y el usuario esperado no debiera interactuar con los elementos en pantalla (debido a una discapacidad). Sin

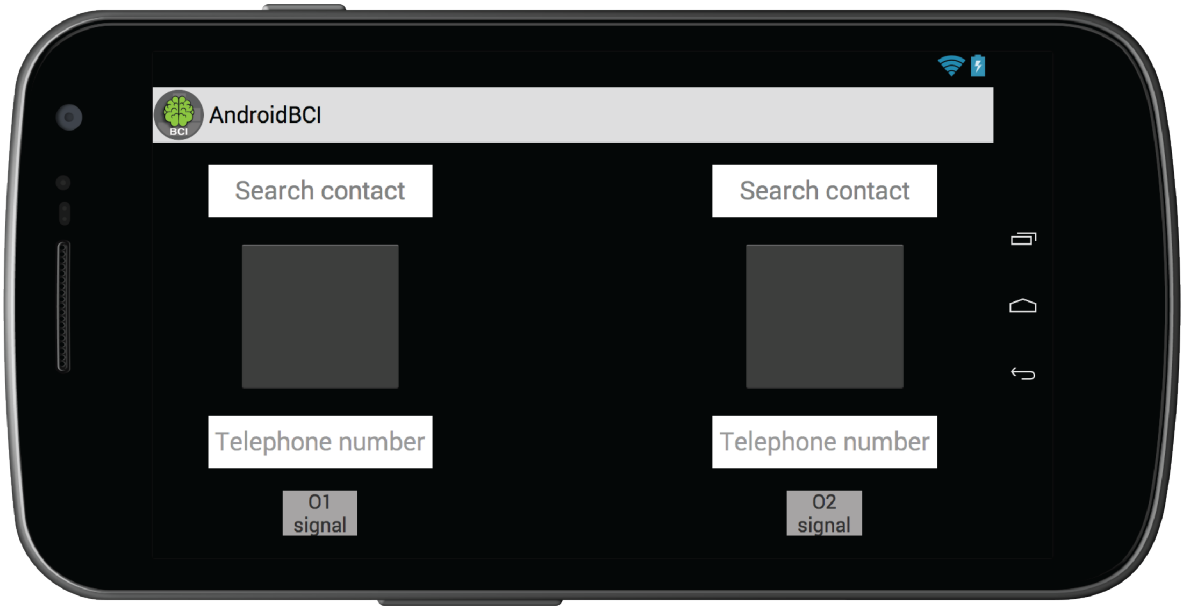


Figura 6.4: "Activity" principal de la Android™ BCI en el Galaxy Nexus

embargo, se muestra la información que se consideró útil para mantener al usuario informado del estado en el que se encuentra la aplicación, como, quiénes son los contactos a los que se va a llamar, cuál es el número telefónico asociado y cómo es la calidad de contacto de los electrodos O1 y O2; información fundamental para iniciar la estimulación visual.

6.2.2.1. Distribución visual del "activity" principal

La "activity" principal está compuesta por diseños de distribución lineal ("linear layout") anidados y graduados en porciones de la pantalla para hacer la aplicación compatible con todos los tamaños de pantalla; ningún valor está fijo y son independientes a la densidad de píxeles. Esta distribución se logra anidando los elementos y ajustando la orientación entre vertical y horizontal. Ésta es la mejor forma para diseñar el ambiente gráfico de la Android™ BCI ya que no son una gran cantidad de subniveles anidados. De acuerdo a las recomendaciones de diseño de Android no es recomendable tener una gran cantidad de layouts anidados porque esta acción reduce el desempeño gráfico de las aplicaciones. [2] Sin embargo, esto ocurre cuando son una gran cantidad de "layouts" anidados, a pesar de que la recomendación para los desarrolladores es mantener la interfaz gráfica lo más simple posible, en este caso

[2] Common Layouts - Consultado en noviembre de 2014

son sólo 4 “layouts” anidados los cuales no representan una carga para la interfaz gráfica del sistema. Esto se puede observar en el desempeño real al probar la aplicación. En la figura 6.5 en la página 89 se muestran los marcos de los “layouts” anidados con algunas proporciones enumeradas en las que se puede ver cómo cada sección anidada está en proporción a su “layout” padre (o el que lo contiene). El texto que contienen algunos elementos se ajusta automáticamente de acuerdo al tamaño de los mismos y a la densidad de píxeles de la misma.

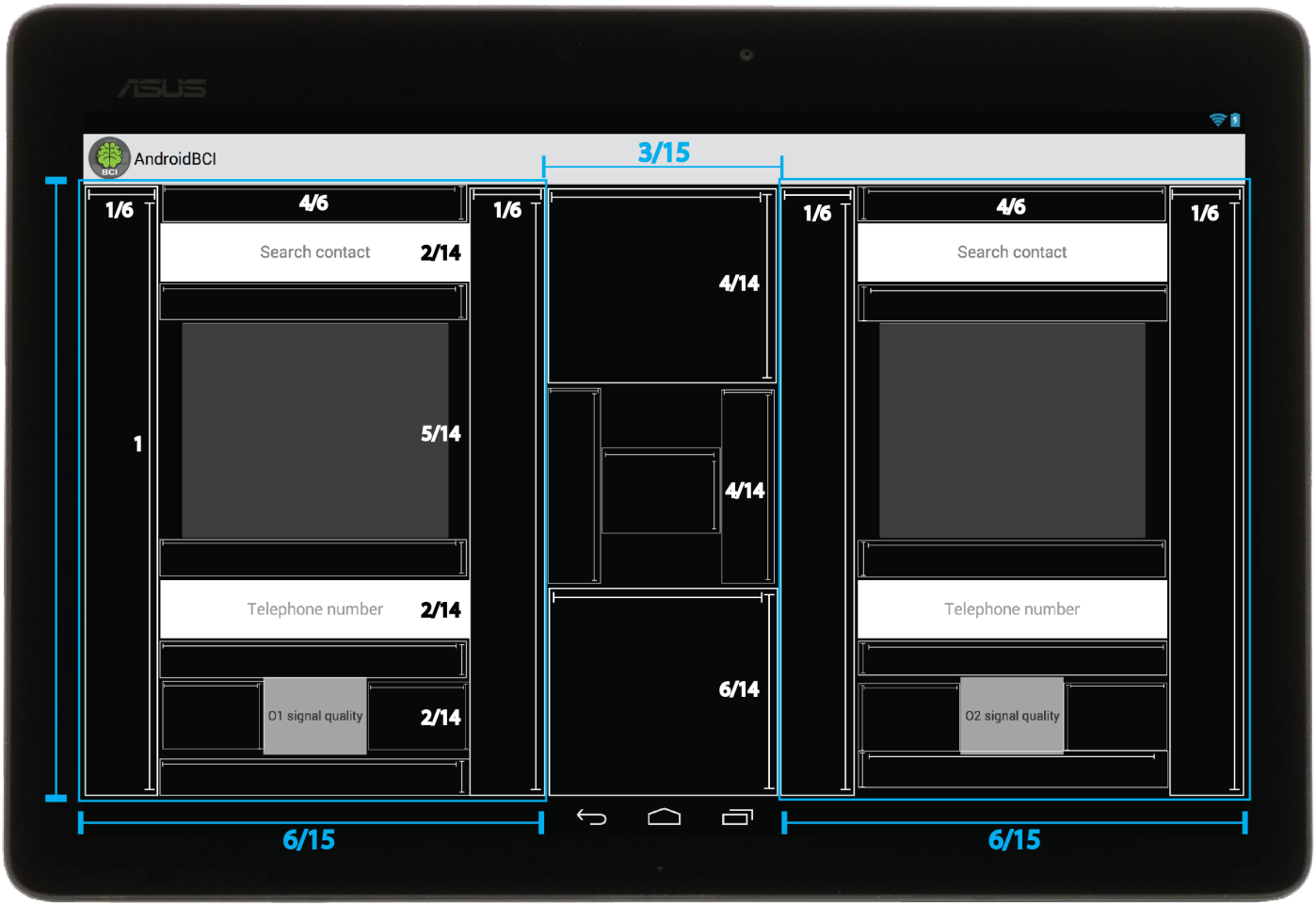


Figura 6.5: “Activity” principal de la Android™ BCI en la tableta

Como referencia se muestra la sección de código en XML de una de las secciones verticales de los extremos laterales, pudiéndose observar la relación que existe entre la indentación y los “layouts” anidados que contienen las secciones para la selección del contacto a llamar, el espacio para la imagen del contacto, la casilla que muestra el número de teléfono a llamar

y un recuadro que con un código de colores muestra la calidad de contacto del electrodo de adquisición. El código de estas secciones se puede consultar en el Apéndice C en la página 153 en la Sección de código 26, Sección de código 27, Sección de código 28 y la Sección de código 29.

6.2.2.2. Calidad de contacto de O1 y O2

Para evaluar la calidad del contacto de los electrodos, se diseñaron unas enumeraciones especiales que tienen los valores especificados de la calidad de contacto a un valor lógico; es decir, hay cuatro estados posibles en la calidad de contacto de los electrodos: *buena, aceptable, mala y ninguna (GOOD, FAIR, BAD y NONE respectivamente en el código)*. El único estado que tiene asociado un valor verdadero es el estado *bueno* muy útil porque definir el estado de contacto de los electrodos se hace mediante comparadores que le asignan uno de los cuatro estados posibles. Pero, cuando surge la pregunta ¿cómo es la calidad del contacto? el método que pregunta recibe como respuesta un valor verdadero o falso que puede ser usado para tomar una decisión. En la Sección de código 6 en la página 91 se puede observar la estructura que tiene la enumeración para evitar confusiones en el estado de la calidad de contacto de los electrodos. El usuario recibe una retroalimentación visual de la calidad de contacto de los electrodos O1 y O2 en pantalla en los objetos "O1 signal quality" y "O2 signal quality" (elementos que se pueden observar en la figura 6.5 en la página 89 mediante un código de colores (que coincide con el del SDK del EPOC) como se puede observar en la tabla 6.2 en la página 90.

Calidad de contacto	Código de color
Buena	Verde
Aceptable	Naranja
Mala	Rojo
Ninguna	Gris

Tabla 6.2: Código de colores asociado a la calidad de contacto de los electrodos.

Información de la calidad de contacto en el paquete de datos

Como se explicó en la sección 4.2.5 la calidad de contacto se puede obtener en dos instantes de tiempo por cada electrodo (O1 y O2 en este caso); estos instantes se pueden observar en la tabla 4.11 en la página 63. En la Sección de código 7 en la página 92 se muestran las

```
public static enum SignalQuality{
  GOOD(true),
  FAIR(false),
  BAD(false),
  NONE(false);

  private  boolean value;

  private SignalQuality(boolean value){
    this.value=value;
  }

  public boolean getSignalQuality(){
    return value;
  }
}
```

Sección de código 6: Enumeración para la calidad de contacto de los electrodos

máscaras que extraen la información de los Bytes del paquete de datos para la calidad de la señal o de contacto de los electrodos O1 y O2 en el ciclo que lee los paquetes de datos del "buffer" del "dongle", se hace un cálculo del estimado de la calidad de la señal y se cambia el valor del elemento visual que informa al usuario en la "activity" principal con el método *publishProgress* que se ejecuta en un hilo distinto al de la GUI para evitar la sensación de retrasos en la interfaz gráfica de la aplicación; este método a su vez llama al método que hace la asignación en la interfaz gráfica de la "activity" como se ve en la Sección de código 8 en la página 93.

El método que evalúa la calidad de la señal empieza preguntando si la calidad es baja y termina preguntando si es buena. Se diseñó de esta manera para optimizar el número de veces que se evalúa la calidad del contacto partiendo del supuesto de que al colocar el EPOC™ en la cabeza siempre se empieza con una calidad de señal baja hasta que se hacen ajustes en la colocación de los electrodos, y una vez que la calidad de contacto es buena ya no se actualiza el estado a menos que ésta cambie.

El método que actualiza los valores del objeto que retroalimentan al usuario en la "activity" y que muestra la calidad de contacto mediante el código de colores previamente mencionado, se puede observar en la Sección de código 9 en la página 95 y en la parte final del código en

```
if(decryptedDataQualitySignal[0]==0x0006){//O1 signal quality
  short sq1 = (short) ((decryptedDataQualitySignal[13] << 9) & 0x00003E00);
  short sq2 = (short) ((decryptedDataQualitySignal[14] << 2) & 0x000003FC);
  short sq3 = (short) ((decryptedDataQualitySignal[15] >> 6) & 0x00000003);
  epocO1SignalQuality= ((double)(sq1|sq2|sq3));
  //Divide it by the maximum posible value to normalize it
  epocO1SignalQuality=epocO1SignalQuality / maxValue;
  Log.i(DEBUG_TAG,"epocO1SignalQuality= "+epocO1SignalQuality);
  if(epocO1SignalQuality<0.2){
    String result= "O1NONE";
    publishProgress(result);
    setO1ContactQualityState(SignalQuality.NONE);
  }else if(epocO1SignalQuality<0.5){
    String result= "O1BAD";
    publishProgress(result);
    setO1ContactQualityState(SignalQuality.BAD);
  }else if(epocO1SignalQuality<0.7){
    String result= "O1FAIR";
    publishProgress(result);
    setO1ContactQualityState(SignalQuality.FAIR);
  }else{
    String result= "O1GOOD";
    publishProgress(result);
    setO1ContactQualityState(SignalQuality.GOOD);
  }
```

Sección de código 7: Extracción de datos de la calidad de contacto del electrodo O1 en el "dongle" en la Android™ BCI

```
protected void onProgressUpdate(String showQuality){
  super.onProgressUpdate(showQuality);
  if(showQuality=="O1NONE"){
    setO1ContactQualityState(SignalQuality.NONE);
  }else if(showQuality=="O1BAD"){
    setO1ContactQualityState(SignalQuality.BAD);
  }else if(showQuality=="O1FAIR"){
    setO1ContactQualityState(SignalQuality.FAIR);
  }else if(showQuality=="O1GOOD"){
    setO1ContactQualityState(SignalQuality.GOOD);
  }else if(showQuality=="O2NONE"){
    setO2ContactQualityState(SignalQuality.NONE);
  }else if(showQuality=="O2BAD"){
    setO2ContactQualityState(SignalQuality.BAD);
  }else if(showQuality=="O2FAIR"){
    setO2ContactQualityState(SignalQuality.FAIR);
  }else if(showQuality=="O2GOOD"){
    setO2ContactQualityState(SignalQuality.GOOD);
  }else{
    Log.i(DEBUG_TAG,"Update setContactQualityState not reached");
  }
}
```

Sección de código 8: Método que llama a la función para actualizar la calidad de contacto del electrodo O2 en la Android™ BCI

la instrucción *finally* se evalúan ciertas condiciones que se tienen que cumplir antes de iniciar la estimulación visual (esta sección de código se explicará más adelante).

6.2.2.3. El botón de *inicio de la estimulación ("start")*

En la Sección de código 9 se pueden ver los requisitos que se deben cumplir para que el botón de *inicio ("start")* start se muestre, esta verificación se hace cada vez que se actualiza la calidad de contacto de los electrodos O1 y O2.

Los requisitos que se deben cumplir para que se muestre el botón de start son los siguientes:

1. Que la calidad de contacto de O1 esté en verde (buena).
2. Que la calidad de contacto de O2 esté en verde (buena).
3. Que ambos contactos a llamar hayan sido seleccionados.

Existe un método que monitorea en cada actualización de la calidad de contacto (dos veces por segundo), y que cuando estas condiciones se cumplan automáticamente se muestra el botón de inicio ("start") para así, poder iniciar la estimulación visual que genera el potencial visual de estado estacionario (SSVEP por sus siglas en inglés) en el usuario que esté observando la pantalla. Este mismo método se encarga de ocultar dicho botón en caso de que las condiciones no se cumplan en algún momento. En la Sección de código 10 en la página 97 se muestra la verificación de las condiciones, esta sección de código se encuentra dentro del método que actualiza la calidad de contacto que es llamado dentro de la lectura de los datos del "buffer".

6.2.2.4. Selección de los contactos

Para que se pueda mostrar el botón de inicio ("start") uno de los requisitos es el hecho de haber seleccionado previamente a los dos contactos a los que el usuario desea llamar; para eso, el usuario tiene que pulsar sobre el botón de selección de los contactos, el cual desplegará una lista con los contactos almacenados en el teléfono y así poder seleccionar un número telefónico para llamar. Si desea llamar a un número telefónico que no esté en la lista de contactos primero hay que almacenarlo en el teléfono.

El algoritmo que resume los tres pasos de verificación para que se pueda lleva a cabo la estimulación y la generación de la llamada telefónica se resumen en el diagrama de flujo de la figura 6.6 de la página 96.

```
public static boolean setO1ContactQualityState(SignalQuality newSignalQuality){
try{
  if(newSignalQuality==SignalQuality.GOOD){// newSignalQuality is GOOD
    if(getO1ContactQualityState()==SignalQuality.GOOD){
      //Do nothing it's already GOOD
      }else{//(getO1ContactQualityState()==SignalQuality.FAIR){
        contactQualityO1.setBackgroundColor(goodColor);
        o1ContactQualityState =SignalQuality.GOOD;}
    }else if(newSignalQuality==SignalQuality.FAIR) {// newSignalQuality is FAIR
      if (getO1ContactQualityState() == SignalQuality.FAIR) {
        //Do nothing it's already FAIR
      }else {
        contactQualityO1.setBackgroundColor(fairColor);
        o1ContactQualityState = SignalQuality.FAIR;}
    }else if(newSignalQuality==SignalQuality.BAD){// newSignalQuality is BAD
      if(getO1ContactQualityState()==SignalQuality.BAD){
        //Do nothing it's already BAD
      }else{
        contactQualityO1.setBackgroundColor(badColor);
        o1ContactQualityState =SignalQuality.BAD;}
    }else{// newSignalQuality is NONE
      if(getO1ContactQualityState()==SignalQuality.NONE){
        //Do nothing it's already NONE
      }else{
        contactQualityO1.setBackgroundColor(noneColor);
        o1ContactQualityState =SignalQuality.NONE;}
    }return true;
  }catch (Exception e){
  Log.i(DEBUG_TAG, "Error en O1changeBackground: "+e.getMessage());
  return false;
  }finally {
    //If certain conditions are granted SHOW START BUTTON
  }
}
```

Sección de código 9: Método que actualiza la calidad de contacto del electrodo O1 en la Android™ BCI

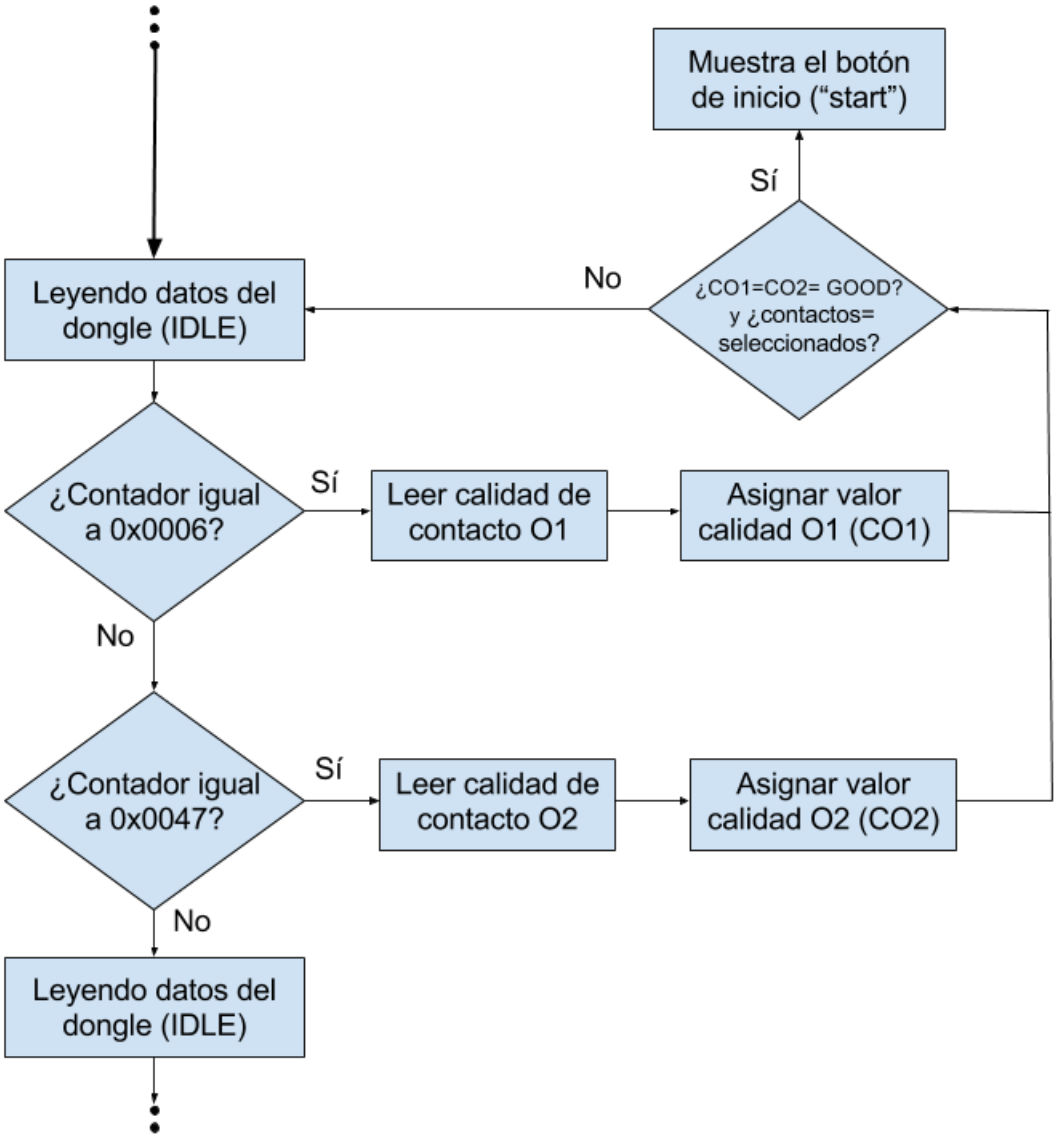


Figura 6.6: Diagrama de flujo para mostrar el botón de inicio (“start”)

```
if(getO1ContactQualityState()==SignalQuality.GOOD && getO2ContactQualityState()==
    SignalQuality.GOOD && contact7HzPickedFlag && contact11HzPickedFlag){
      startButton.setVisibility(View.VISIBLE);
}
```

Sección de código 10: Método que habilita el botón de inicio (*Start*) para comenzar la estimulación en la Android™ BCI

6.2.3. Generación de la llamada telefónica

Una vez que se han verificado los requisitos para mostrar el botón de "start" y este se despliega en la pantalla, la estimulación visual inicia cuando se pulsa sobre él; esto desencadena una serie de acciones en la "activity" de estimulación PPVEE, que se ejecuta con el método startActivityForResult(). Es entonces que la "activity" principal entra en espera del resultado que indica en cual de las dos bandas hay mayor potencia después de la estimulación visual (7 u 11 Hz). Este resultado es el que permite tomar la decisión para hacer la llamada telefónica, en caso de que el resultado sea cero esto significa que no se pudo tomar una decisión debido a que la potencia en una banda no es por lo menos 10 % mayor con respecto a la potencia de la otra banda. El código de este proceso se puede ver en la Sección de código 11 en la página 98.

6.3. "Activity" de estimulación PPVEE

Al presionar el botón de *Start* se llama a la ejecución de una "activity" de pantalla completa que sólo contiene los estimuladores visuales lo más separados posible debajo de la imagen del contacto como referencia. Esta es una "activity" que tiene un tiempo de vida aproximado de 5 segundos, tiempo que corresponde a la duración de la estimulación PPVEE y al procesamiento de las señales en tiempo real, lanzada con una instrucción especial que espera un resultado ya que es invocada con el método *startActivityForResult*.

Existe información que es esencial enviar a la "activity" de los estimuladores para que funcione de manera correcta y así no trabajar doble, e.g. volver a calcular la llave de descifrado, sería repetir trabajo ya realizado.

La mejor manera de hacer los estimuladores, es cambiando la transparencia de un elemento color blanco en pantalla entre cero y cien, ya que este método está acelerado por hardware en el sistema operativo a partir de la versión 11 del API de Android (Android 3.0 Honeycomb).

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data){
if(requestCode == OTROS LISTENERS){
        \\More coding....
}else if(requestCode == START_BUTTON_REQUEST){
  if(resultCode == RESULT_OK){
    int result= data.getIntExtra("result",-1);
    //Returns -1 if the value with the key "result"
    if(result == 7){
      //Make the phonecall to the 7 Hz contact
    }else if(result == 11){
      //Make the phonecall to the 11 Hz contact
    }else if(result == 0){
      //Send the user a message indicating the decision could not be made
    }else{
      Log.i(DEBUG_TAG,"No decision could be made");
    }
  }else{
    Log.i(DEBUG_TAG, "Stimulators canceled THE PHONE CALL CANNOT BE MADE");
    //Toast saying activity cancelled
  }
}
```

Sección de código 11: En espera del resultado de la estimulación visual para hacer la llamada telefónica en la Android™ BCI

De otra manera, habría que actualizar la pantalla de forma manual para avisar acerca de los cambios y esto repercute en el desempeño general del dispositivo móvil además de que se aumenta el gasto de energía.

Esta "activity" tiene funciones muy específicas que se pueden resumir en las siguientes:

1. Generar los estímulos visuales en la pantalla de la forma más precisa posible para generar el PPVEE en la corteza cerebral lo más cercano a las bandas de estimulación (7 y 11 Hz).
2. Extraer las señales del "buffer" del "dongle" del EPOC^{TM} desde el momento en que comienza la estimulación.
3. Cuando hayan pasado dos segundos de estimulación, comienza el cálculo de la potencia muestra tras muestra.
4. Cuando hayan pasado dos segundos de iniciado el cálculo de la potencia, enviar la respuesta acerca de cual banda de potencias es por lo menos 10 % mayor con respecto a la otra (7 u 11) como el resultado de la actividad.
5. En caso de que no se pueda determinar en que banda es mayor la potencia con el umbral, entonces enviar cero (0) como respuesta.
6. En caso de que se presione el botón de atrás del teléfono la respuesta es un *RESULT_CANCELED* y de esta manera se avisa al usuario que fue cancelada la estimulación.

El resumen detallado del flujo de las acciones que se llevan a cabo en cada estado por el que pasa la "activity" de estimulación visual se puede ver en el apéndice E

6.4. Procesos de fondo en la "activity" de estimulación

La lectura de los datos del "dongle" se hace de la misma manera que en la "activity" principal para mostrar la calidad de contacto de los electrodos (ver la sección 6.1.5.4 en la página 81), obteniéndose los datos cifrados con la diferencia de que en el ciclo de lectura de los datos a partir de la muestra 257 se hace el cálculo de la potencia en ese instante de tiempo, y en la muestra 512 se guarda el resultado y se llama al método que cierra la "activity".

6.4.1. Procesamiento de las señales

Para averiguar a qué estímulo está prestando atención el usuario, es necesario determinar a qué frecuencia se sincronizaron las neuronas de la corteza visual en la parte occipital del cerebro. Los electrodos O1 y O2 están situados en la región de interés y han registrado el EEG del usuario desde el momento que se colocó el EPOC™. Sin embargo, la respuesta fisiológica a los estímulos visuales se genera 300 ms después de que el usuario observa el estímulo visual [9] (ver figura 1.3 en la página 13), es por eso, que a partir de que la estimulación visual comienza, la adquisición de los datos de EEG en los electrodos O1 y O2 comienza 45 muestras después.

Existen diversas maneras de calcular la potencia de una señal, algunas de ellas involucran transformaciones al dominio frecuencial de las señales. Una forma de hacer está transformación sería adquirir algunos segundos de señales de O1 y O2, pasar las señales al dominio frecuencial (con la trasformada discreta de Fourier) para calcular el espectro de potencias de las señales de los electrodos como se hizo en el protocolo para desencadenar ondas alfa en la sección 6.1.6 en la página 82. Sin embargo, este tipo de procesamiento necesita tener la señal completa que se va a analizar o hacerlo por ventanas de tiempo, pero siempre es un procesamiento posterior a la adquisición que transforma la señal al dominio de la frecuencia e implica hacer uso de herramientas de procesamiento digital de señales que no están disponibles por defecto en Android, como la transformada rápida de Fourier aunque sí existen bibliotecas (o paquetes de funciones) disponibles para el procesamiento digital de señales en Java™ (algunas con costo) para lograr procesar las señales. Esta alternativa se descartó porque uno de los requerimientos de la aplicación era hacerla lo más rápida posible y hacer el procesamiento en tiempo real.

El teorema de Rayleigh-Parseval

El físico Inglés John William Strutt (Lord Rayleigh) usó la relación de Parseval por primera vez en 1889 mientras estudiaba la radiación de un cuerpo negro en el espacio. Esta relación hace uso del resultado unitario de la transformada de Fourier para demostrar que la suma (o integral) de los cuadrados de la energía de la trasformada de Fourier en la frecuencia, es igual a la suma (o integral) de los cuadrados de su trasformada en el tiempo. A este teorema se le llamó **"el teorema de Rayleigh-Parseval"** y su mayor aplicación reside en demostrar que la energía de ambas ecuaciones es equivalente [33] como se puede ver en la ecuación 6.7 de la página 101 donde se muestra la relación con la transformada discreta de Fourier.

$$\sum_{n=0}^{N-1} |\chi[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |\mathrm{X}[k]|^2$$

Figura 6.7: Teorema de Rayleigh-Parseval

Procesamiento del EEG

Haciendo uso de la propiedad del teorema de Rayleigh-Parseval se evita hacer una transformación al dominio de la frecuencia de las señales de los electrodos O1 y O2 para calcular un estimado de la energía en el tiempo calculando la magnitud de la potencia en un instante de tiempo para después de dos segundos calcular la potencia total en esta ventana de tiempo.

6.4.2. Diseño de los filtros

Para hacer el cálculo de la magnitud de la potencia lo más eficiente posible, se diseñaron dos filtros digitales FIR multibanda de fase lineal para 7 y 14 Hz (la frecuencia fundamental del estímulo visual y un armónico) y para 11 y 22 Hz de orden 256 usando el algoritmo de Parks-McClellan, algoritmo que a su vez usa el algoritmo de intercambio Remez y la teoría de aproximación de Chebyshev dando como resultado un ajuste óptimo porque minimiza el máximo error entre la respuesta en frecuencia deseada y la actual.

El resultado del diseño de los filtros son vectores de 257 coeficientes que fueron cortados a veinte decimales en la implementación. Se hizo una prueba para ver si el hecho de cortar los coeficientes a veinte decimales afectaba la respuesta en frecuencia de los filtros diseñados. En la figura 6.8 en la página 102 se muestra el resultado del diseño del filtro de 7 y 14 Hz y en la figura 6.9 en la página 103 se muestra el resultado del diseño del filtro de 11 y 22 Hz, en ambos se compara la respuesta en frecuencia del filtro con la respuesta en frecuencia ideal y la respuesta en frecuencia del filtro cortado a veinte decimales con la respuesta en frecuencia ideal.

El vector de coeficientes se exportó de Matlab™ en un archivo de valores separados por comas a la "activity" de los estimuladores visuales que es en donde se hace el procesamiento. Para cambiar los coeficientes del filtro, lo único que hay que hacer es cambiar la línea de código que contiene con los valores de los coeficientes en caso de que se requieran hacer ajustes en las bandas de paso. Un ejemplo de esto se puede ver en la Sección de código 12 en la página 104 para los filtros de 7 y 14 Hz y de 11 y 22 Hz en donde se muestran en carácter ilustrativo únicamente los primeros y los últimos coeficientes del vector.

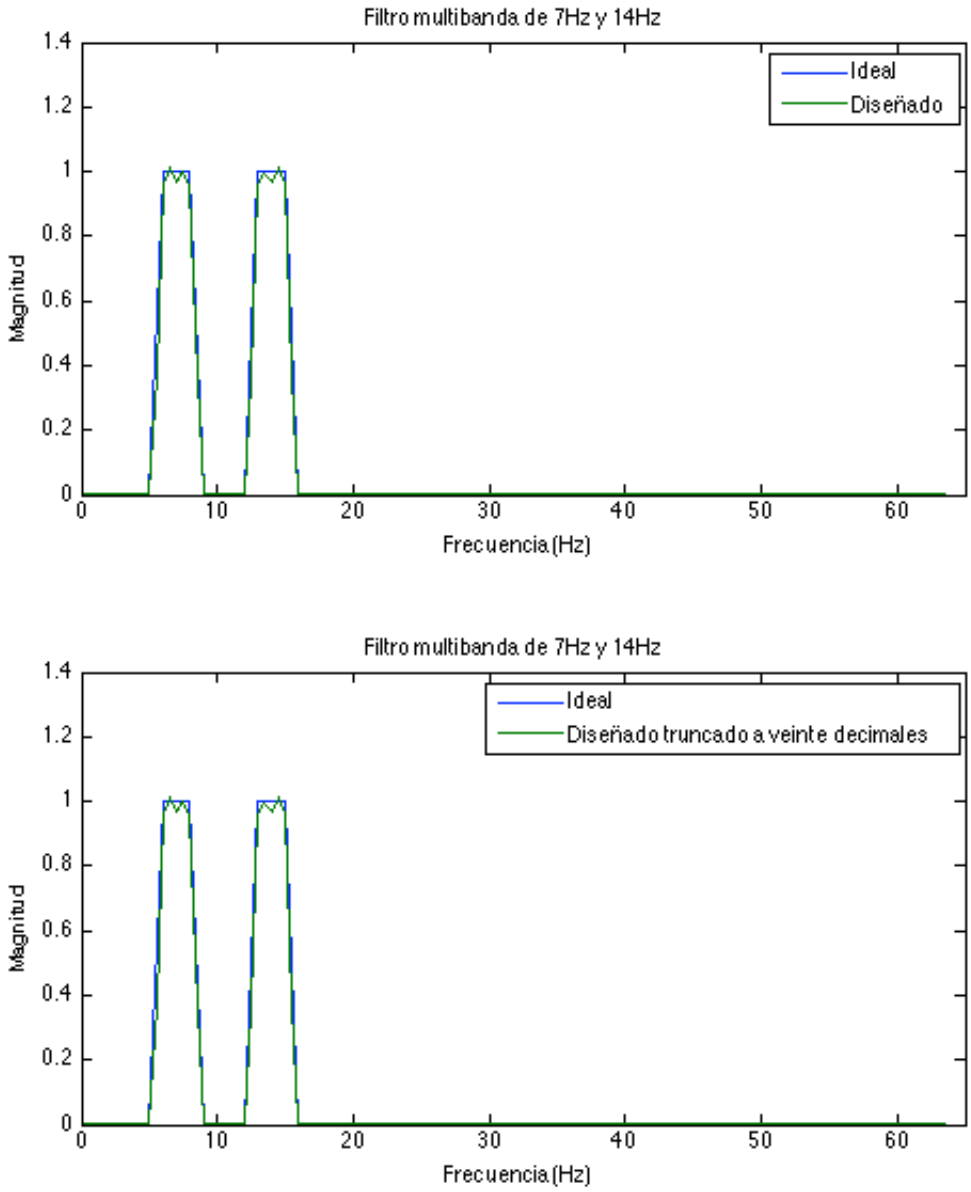


Figura 6.8: Filtro de 7 y 14 Hz.

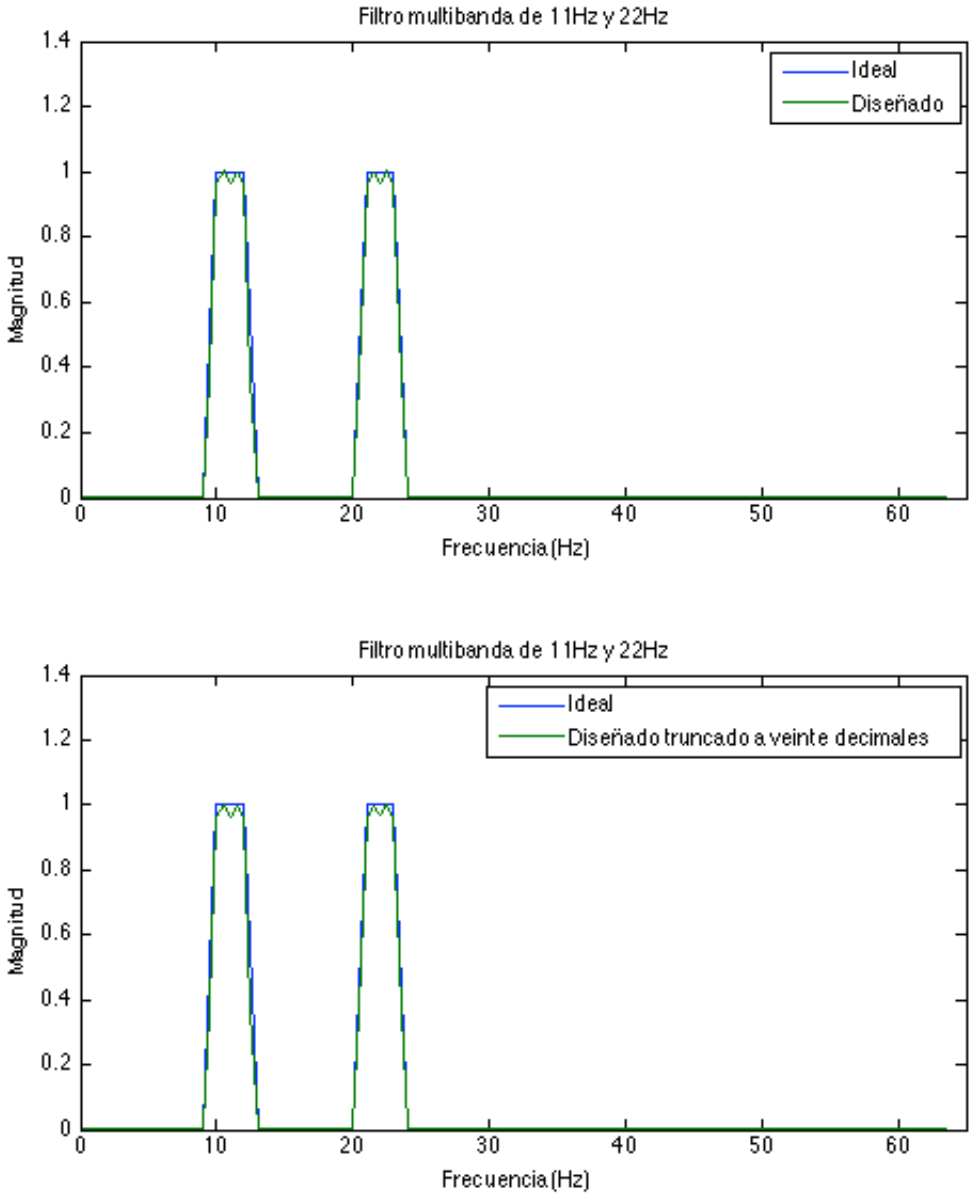


Figura 6.9: Filtro de 11 y 22 Hz.

```
//7 and 14 Hz filter coefficients
private static final double[] FilterCoef7_14 = {0.00239455837587030079,
0.00095425285479180564,0.00059131169302875056,-0.00006821624430425631,
                    ..........................
,-0.00006821624430425631,0.00059131169302875056,0.00095425285479180564,
0.00239455837587030079};

//11 and 22 Hz filter coefficients
private static final double[] FilterCoef11_22 = {0.0.00225503602647888915,
0.00066398697667397291,-0.00020088037549108339,-0.00092938012925806990,
                    ..........................
,-0.00092938012925806990,-0.00020088037549108339,0.00066398697667397291,
0.00225503602647888915};
```

Sección de código 12: Primeros y últimos coeficientes del vector de coeficientes de los filtros de 7 y 14 Hz y de 11 y 22 Hz.

6.4.3. Cálculo de la potencia en la Android™ BCI

De manera nativa, no existen funciones para el filtrado digital de señales en Android que reciban como datos de entrada las señales y los coeficientes filtros y a su vez regresen como respuesta la señal filtrada, es por eso, que para el procesamiento del EEG en la aplicación Android BCI, con la finalidad de optimizar el uso de los recursos y asegurar el procesamiento en tiempo real se escribió la ecuación de diferencias que calcula la potencia en un instante de tiempo con 257 muestras previas.

6.4.3.1. Cálculo de la potencia en un instante de tiempo para cada electrodo

El cálculo de la potencia en el tiempo es controlado con el número de muestra adquirido, éste se hace en el ciclo de lectura de los datos de "buffer" para cada instante de tiempo. Para cada electrodo se calcula en tiempo real la potencia en cada una de las bandas de interés, así que la potencia total para cada banda es el resultado de la potencia calculada en O1 y O2. En la Sección de código 13 en la página 105 se muestra este proceso en el código dentro del ciclo de lectura de datos del "buffer" en la "activity" de estimulación visual para el electrodo O1 en la banda de 7 y 14 Hz como ejemplo, esto se hace también para la otra banda de potencia.

```
//Power at 7 and 14 Hz
Pw_O1_7_14=Pw_O1_7_14 + (FilterCoef7_14[0]*EEG[6][N])^2 +
     (FilterCoef7_14[1]*EEG[6][N-1])^2 + (FilterCoef7_14[2]*EEG[6][N-2])^2 +
     (FilterCoef7_14[3]*  EEG[6][N-3])^2   +......
     ..... + (FilterCoef7_14[254]*EEG[6][N-254])^2 +
     (FilterCoef7_14[255]*EEG[6][N-255])^2 + (FilterCoef7_14[256]*EEG[6][N-256])^2;
```

Sección de código 13: Cálculo de la potencia en las bandas de 7 y 14 Hz en un instante de tiempo.

6.4.4. Resultado de la "activity" de estimulación

Al ser ésta una "activity" invocada con el método *startActivityForResult*, tiene que ser diseñada para que al finalizar envíe un resultado ligado a una respuesta. Las posibles respuestas son:

1. *RESULT_ OK* es el caso en el que la estimulación visual concluyó de manera exitosa y se determinó la respuesta en la que se debe enviar el resultado acerca de cual fue la banda donde la potencia fue por lo menos 10 % mayor con respecto a la otra. Este resultado debe ser enviado en una variable que contenga la información que se quiere transmitir. Este umbral para la toma de decisión puede optimizarse con base en la experiencia y con la recolección de datos mediante el uso continuo de la aplicación y el análisis de la intención del usuario para maximizar la eficiencia de la aplicación, esta acción tendría que ser una personalización para cada usuario con la finalidad de mejorar la eficiencia de la Android™ BCI. Sin embargo, el propósito en esta aplicación genérica es restringir que se tome una decisión por mínima que sea la diferencia entre las bandas de potencia (e.g. el caso en el que se inicia la estimulación con el amplificador sin estar colocado al sujeto, aunque esta condición es prevenida con la aparición del botón de "start" se podría retirar el amplificador de la cabeza del sujeto una vez iniciada la estimulación visual) y de la misma manera tener como otra opción de resultado el escenario en el que no se puede tomar una decisión acerca de cual de las bandas de potencia fue mayor, o por lo menos la diferencia no es significativa.
2. *RESULT_ CANCELED* la cual no tiene que estar ligada con algún resultado, simplemente informa que la estimulación visual concluyó de manera no exitosa.

De esta forma cuando el ciclo de estimulación termina, se envía una respuesta, que está estructurada como se muestra en la Sección de código 30 en la página 157 en la que se puede

ver que se hace una comparación de las potencias totales, conformadas por la potencia en la banda de 7 y 14 Hz del electrodo O1, sumada con la potencia en la banda de 7 y 14 Hz del electrodo O2 y la potencia en la banda de 11 y 22 Hz del electrodo O1, sumada con la potencia en la banda de 11 y 22 Hz del electrodo O2.

Esta comparación valida que alguna de las potencias es por lo menos 10 % mayor a su comparador, si por el contrario no se puede tomar una decisión, la respuesta lo informa de igual manera. Este proceso se puede ver completo en la Sección de código 30 en la página 157 del Apéndice C.

6.5. Validación del diseño de los filtros

Antes de que se implementaran en Java™ las ecuaciones de diferencias que representan a los filtros diseñados, éstos fueron probados con señales conocidas provenientes de un protocolo de adquisiciones desarrollado en otra tesis de maestría (Zamorano 2013 [2]) en la que se tenía control sobre los sujetos y los estímulos.

6.5.1. Objetivo

El objetivo fue 'validar los filtros antes de su implementación en Android con señales conocidas, para poder comprobar su funcionamiento antes de traducir a código en Java™ las ecuaciones. Vale la pena recordar que Android, de manera nativa, no dispone de bibliotecas para el procesamiento digital de señales, o por lo menos, no para el filtrado de las mismas, además algunas de las bibliotecas disponibles para el manejo de señales en Java tienen un costo económico y de tiempo asociado (el tiempo en el que se aprenden a usar las herramientas para procesar las señales). Por lo anterior se tomó la decisión de hacer una implementación propia. Los filtros diseñados debían ser probados en el entorno común de trabajo con señales conocidas de forma previa a su implementación.

6.5.1.1. El origen de las señales

Las señales usadas para probar los filtros diseñados fueron señales provenientes de otro proyecto de maestría del LINI elaborado por Itzel Zamorano [2] que también usó el paradigma de PPVEE en el que se asignaba de forma aleatoria la frecuencia a observar. En estas señales uno de los canales de adquisición contiene marcas elaboradas por la autora con un botón

mecánico para indicar el inicio, término y frecuencia correspondiente y así poder cfacilitar el procesamiento.

Protocolo de adquisición

Las señales de PPVEE de prueba se tomaron de la tesis de maestría de Zamorano 2013 [2] en la que se detalla cada uno de los pasos que se describen a continuación, mismos que se pueden observar en la Figura 6.10 en la página 107.

Se ocupó un tipo de estímulo simple como es de una luz que parpadea. El paradigma consiste en cuatro fases: la primera, el parpadeo de dos luces amarillas, las cuales indican el inicio de la rutina; la segunda, el parpadeo aleatorio de una de las luces amarillas, indica a cual LED se ha de prestar atención en la siguiente fase; la tercera, el parpadeo de dos luces blancas con duración de alrededor de 10 segundos (es aquí donde se genera el estímulo y en la que se ha de prestar atención al LED previamente indicado en la fase anterior), y por último todas las luces se apagan indicando que la rutina ha terminado, dando paso de nuevo al inicio del ciclo, el cual se repite 10 veces. Además, se formularon dos modalidades de rutinas, en una de ellas se le pide al sujeto que sólo se enfoque en poner atención al estímulo, y en la otra, además de lo anterior, tenía que memorizar la secuencia de estimulación de las luces amarillas que prendían aleatoriamente.

Figura 6.10: Esquema del ciclo de estimulación. Tomado de: Zamorano 2013 [2]

Características de las señales

El resultado de la adquisición de acuerdo al protocolo anterior, fueron señales en 4 canales con una frecuencia de muestreo de 256 Hz con marcas temporales en uno de ellos que ubica los eventos ocurridos durante la adquisición, e indica a que frecuencia de estimulación visual corresponde el ciclo correspondiente.

6.5.2. Metodología

A continuación se describen las características del experimento y las consideraciones para el desarrollo de las pruebas hechas a los filtros diseñados.

Requisitos y características del experimento:

- Señales provenientes del registro de EEG de los canales O1 y O2 de por lo menos dos sujetos del experimento de Zamorano 2013 [2] de las que se puedan aislar los diez segmentos que contienen PPVEE y que se conozca la frecuencia real de estimulación visual.
- Un experto que pueda clasificar las señales observando el espectro de potencias del EEG tomando en cuenta la frecuencia fundamental y el primer armónico, siendo éste considerado el estándar de oro para la clasificación.
- Desarrollar un código en Matlab^{TM} que grafique los espectros de potencia sin filtrar de cada uno de los diez segmentos en una hoja para que el experto pueda clasificar cada uno de ellos.
- Desarrollar un código en Matlab^{TM} que grafique los espectros de potencia filtrados de cada uno de los diez segmentos para ambas bandas de paso con sus armónicos en una hoja e imprima el cálculo de la suma de la potencia para cada una de las bandas de paso.

Descripción general del experimento

1. Se usaron señales de tres sujetos para ser procesadas. Cada una de ellas contenía la secuencia de frecuencias de estimulación visual real de los diez ciclos en los que consistió cada una de las pruebas.

2. Se filtraron cada uno de los diez segmentos en la frecuencia fundamental (7 y 11 Hz) con los filtros diseñados, se incluyó la banda de paso del primer armónico de ambos (14 y 22 Hz respectivamente). Se graficó el resultado de este filtrado y la suma de la potencia para ambas bandas de paso en una misma figura (gráficas B) para cada uno de los sujetos de experimentación. El resultado se puede observar en la Figura 6.12 en la página 110 en la que se observa en verde el espectro de potencia correspondiente al segmento filtrado a 11 y 22 Hz y en azul el espectro de potencia correspondiente al filtrado con la banda de paso a 7 y 14 Hz. En la esquina superior derecha de la gráfica se puede observar el cálculo de la suma total de la potencia por banda de filtrado.

3. Se graficó en una hoja el espectro de potencia de cada uno de los diez segmentos sin filtrar (una por cada canal, O1 y O2) y se imprimió para que el experto en detección de PPVEE pudiera clasificar cada uno de los espectros (gráficas A). Las gráficas incluyen tres marcadores: frecuencia fundamental (7 y 11 Hz) marcador rojo, primer armónico (14 y 22 Hz respectivamente) marcador azul, segundo armónico (21 y 33 Hz respectivamente) marcador verde, con el propósito de ayudar al experto. Un ejemplo de esto se puede observar en la Figura 6.11 en la página 110.

4. Posibles resultados de la clasificación:

   - El segmento tuvo una estimulación visual a 7 Hz.
   - El segmento tuvo una estimulación visual a 11 Hz.
   - No es posible determinar la frecuencia de estimulación visual del segmento (sólo en la clasificación del experto).

5. Comparar las clasificaciones hechas por el experto y por el algoritmo que calcula la suma de potencia para obtener los resultados.

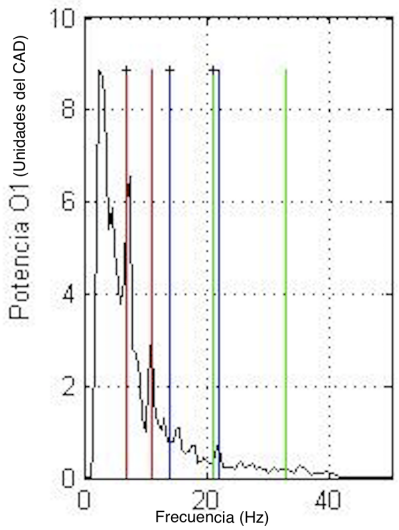


Figura 6.11: Ejemplo del tipo de gráfica que observa el experto considerado como el estándar de oro.

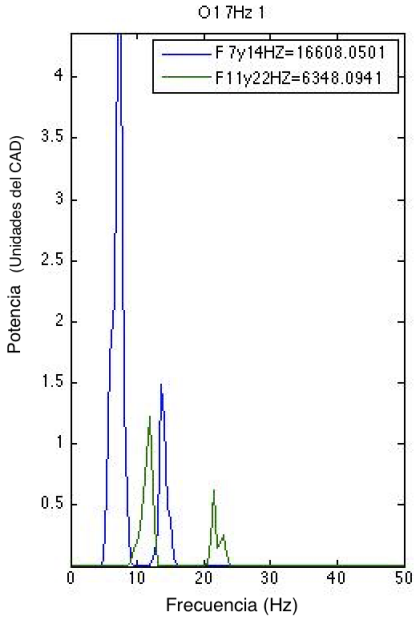


Figura 6.12: Ejemplo del tipo de gráfica que se genera después del filtrado de las señales y que contiene el cálculo de la suma de la potencia.

En la Figura 6.13 en la página 111 se puede observar el tipo de hoja que se le entregaba al experto para que clasificara cada una de las gráficas de potencia de acuerdo al estímulo que creía que el sujeto estaba observando y escribiera sobre ellas el resultado. Es importante mencionar que fue un experimento con doble ciego debido a que como se puede observar en esta misma Figura 6.13 no existe información alguna del sujeto de experimentación así como información alguna de la frecuencia del estímulo visual a la que debería estar observando el mismo. El control estaba dado a través de códigos conocidos solamente por la persona que desarrolló el experimento y que identificaban cada hoja relacionándola con el sujeto correspondiente. El otro cegado se dio debido a que el algoritmo que clasificaba las señales no tenía información "a priori" de cual era la frecuencia real de estimulación a la que debería estar mirando el sujeto, ya que ésta no es una variable en el algoritmo.

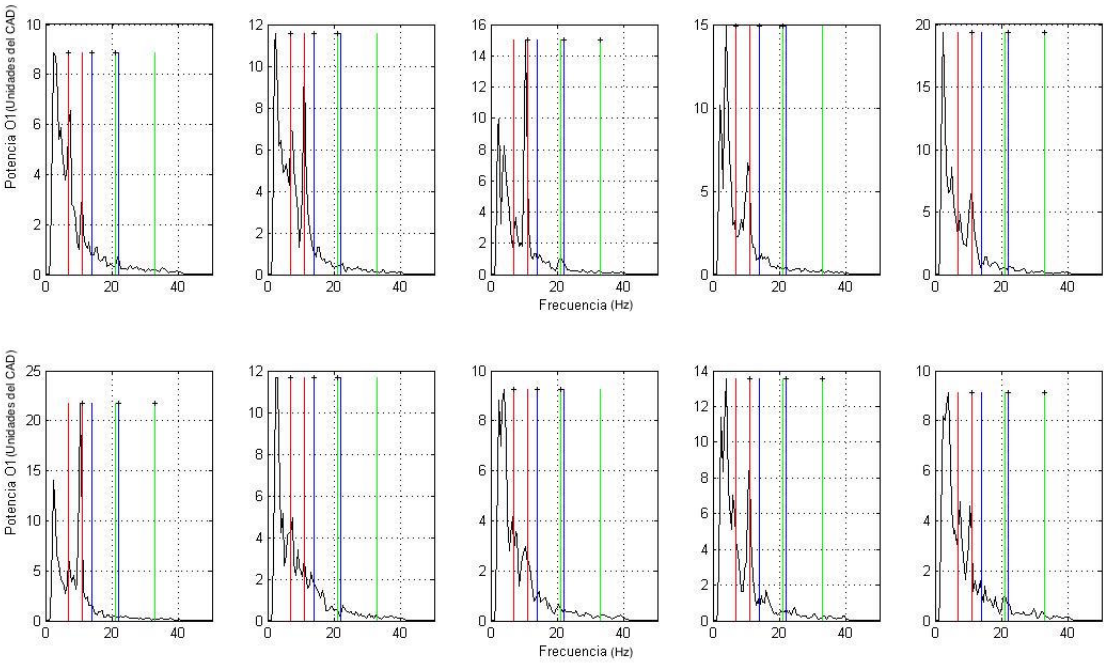


Figura 6.13: Ejemplo del tipo de hoja que se le dio al experto para que clasificara cada una de las gráficas de potencia de O1 de acuerdo al estímulo que el creía que el sujeto estaba observando.

6.5.3. Descripción de resultados

Los resultados del experimento consisten en la combinación de las clasificaciones de los segmentos de acuerdo a la categoría que haya asignado cada clasificador de la siguiente manera:

- Clasificación de acuerdo a la frecuencia real de estimulación.
- Clasificación del experto.
- Clasificación del algoritmo de acuerdo a la suma total de la potencia por bandas de filtrado.

Se desarrolló una medida de probabilidad en la que se le dio más peso a la clasificación hecha por el experto (60 %) y menos peso al conocimiento "a priori" de la frecuencia del estimulo visual al que el sujeto debía prestar atención. Lo anterior, debido a que el sujeto podía no observar el diodo emisor de luz que le indicaba a qué estímulo poner atención o de manera deliberada observar el diodo emisor de luz contrario al indicado generando resultados distintos a los esperados. Con estos pesos a las clasificaciones establecidos se elaboró la tabla de resultados.

6.5.4. Resultados

La tabla completa de resultados se puedo observar en la Tabla F.1 en la página 165. Los resultados para el electrodo O2 y el promedio de ambos fueron similares motivo por el cual no se incluyen en esta tesis.

Los resultados sintetizados se muestran en las tablas de la página 113 en las que se muestran las matrices de confusión de los resultados de la clasificación con respecto al estímulo visual (EV) al que el sujeto debería prestar atención (Tabla 6.3) para evaluar el desempeño del clasificador con respecto a lo ideal. Por otro lado se muestran las matrices de confusión que evalúan el desempeño del clasificador comparándolo con la clasificación del experto (Tabla 6.4). En ambos casos se muestran los resultados generales (con los tres sujetos), los resultados completos del experimento se muestran en el Apéndice F en la página 164.

Con los resultados de las pruebas hechas a los filtros se comprobó el funcionamiento de los mismos. Estos resultados preliminares fueron de gran utilidad ya que permitieron incluir un caso que no se estaba considerando en el desarrollo del cual se hablará en la Subsección 6.5.5 en la página 113.

Sujeto 1		
	Clasificador	
EV	7 Hz	11Hz
7 Hz	3	2
11 Hz	2	3
Sensibilidad: 0.6		
Especificidad: 0.6		

Sujeto 2		
	Clasificador	
EV	7 Hz	11 Hz
7 Hz	5	1
11 Hz	0	4
Sensibilidad: 0.83		
Especificidad: 1		

Sujeto 3		
	Clasificador	
EV	7 Hz	11 Hz
7 Hz	5	0
11 Hz	3	2
Sensibilidad: 1		
Especificidad: 0.4		

Tabla 6.3: Matrices de confusión estímulo visual (EV) v.s. clasificador

Sujeto 1		
	Experto	
EV	7 Hz	11Hz
7 Hz	3	2
11 Hz	2	3
Sensibilidad: 0.6		
Especificidad: 0.6		

Sujeto 2		
	Experto	
EV	7 Hz	11 Hz
7 Hz	4	0
11 Hz	2	4
Sensibilidad: 1		
Especificidad: 0.66		

Sujeto 3		
	Experto	
EV	7 Hz	11 Hz
7 Hz	5	0
11 Hz	0	5
Sensibilidad: 1		
Especificidad: 1		

Tabla 6.4: Matrices de confusión experto v.s. clasificador

6.5.5. Análisis de resultados

En el caso del análisis en el que se evalúa el clasificador en contra del estímulo visual (ver Tabla 6.3) se puede observar que el clasificador no funciona muy bien con el sujeto 1, de forma aceptable con el sujeto 2 y bien con el sujeto 3 en la detección de los estímulos de 7 Hz, y existe un comportamiento similar para la detección del estímulo visual de 11 Hz. En esta versión del clasificador no existía la opción de no poder tomar una decisión debido a que las potencias son muy parecidas y el clasificador siempre toma una decisión por mínima que sea la diferencia.

Por otro lado el escenario en el que se comparan el experto contra el EV, se puede ver que el desempeño en el sujeto 1 es moderado, aceptable en el sujeto 2 y muy bueno en el sujeto 3. Es importante mencionar que el experto tiene más variables a considerar que el algoritmo que sólo mide la potencia en las bandas, como el hecho de saber a priori que el experimento realizaba 5 estimulaciones de cada frecuencia aleatorizadas, sin embargo pudo ser una variable no considerada.

Si se compara el desempeño del clasificador con el del experto se observa que el desempeño del experto es mejor, pero muy similar al del clasificador de forma global. En ambos experimentos se observa una mejor clasificación de los estímulos de 7 Hz, el origen de esto puede

deberse a múltiples factores no determinados desde la estimulación visual, la adquisición o el procesamiento y despliegue.

Como consecuencia de este análisis de resultados se determinó modificar la versión final del algoritmo de clasificación y establecer un umbral de diferencia de 10 % mínimo entre las sumas de la potencia al tomar la decisión (cifra establecida por criterio propio del desarrollador y sin una metodología elaborada para esto). Esto agrega una tercera opción de resultado posible en la aplicación de Android, el comunicarle al usuario que no se pudo determinar que estímulo visual estaba observando.

Capítulo 7

Discusión de resultados

La aplicación está pensada para mejorar la calidad de vida de usuarios que tienen alguna condición de salud que les impide moverse. El usuario puede estar en algún hospital, en casa o en cualquier sitio afuera de un laboratorio de investigación en neurociencias. La pregunta obligada es, si el usuario no se puede mover ¿cómo puede seleccionar los contactos? o peor aún, ¿cómo puede presionar el botón de inicio para comenzar la estimulación visual? La respuesta a estas interrogantes excede los límites de este trabajo ya que el desarrollo de un *"brain switch"* es un reto actualmente para las sujetos involucrados en el desarrollo de interfaces cerebro-computadora. Ya existen diversas aproximaciones a la solución [34, 35, 36, 37], pero ninguna ha logrado la solución óptima. Por esta razón la Android™ BCI se ve limitada debido a que la configuración y el inicio de la estimulación visual debe ser ejecutada por una persona que auxilie al usuario. Se propone una solución al hecho de tener que presionar el botón de inicio en la sección siguiente.

La aplicación puede tener muchas funciones adicionales y se puede mejorar la interfaz gráfica de usuario, se enfocaron los esfuerzos en hacer una aplicación multiprocesos para poder dar una respuesta casi inmediata con procesamiento de las señales de EEG en tiempo real, el desarrollo de este tipo de aplicaciones generalmente es realizado por un equipo multidisciplinario de expertos que desarrollan cada una de las partes de la aplicación de acuerdo a su área de conocimiento; no obstante, se hizo el mejor trabajo posible con los recursos de tiempo y conocimientos limitados del creador. La complejidad de programar la aplicación en un dispositivo móvil, en el que los recursos cambian con el tiempo y debido a que pueden ocurrir eventos externos a los que el sistema da prioridad (como la recepción de una llamada telefónica) complica la programación y retrasa el desarrollo.

Para el procesamiento del EEG se utilizó un algoritmo que tiene una implementación en tiempo real debido a la rapidez con la que se necesitaba saber la respuesta. Son algoritmos que analizan la respuesta fisiológica generada por los estímulos visuales de frecuencia fija que se han usado en otros proyectos del Laboratorio de Investigación en Neuroimagenología y su funcionamiento está probado. [16] Sin embargo se pueden implementar técnicas de procesamiento digital del EEG más robustas que minimicen los errores en la clasificación de las señales como el Análisis por Componentes Independientes; se probó esta aproximación en la computadora con las señales de EEG del EPOC™ con buenos resultados (ver Apéndice G). Existen múltiples herramientas para el uso de los algoritmos de Análisis por Componentes Independientes, entre ellos fastICA en la computadora. [38] No obstante, implementar estos algoritmos en Android significa escribir los algoritmos que procesen el EEG como se hizo para el cálculo de la potencia en tiempo real en la Android™ BCI.

El manejo de hilos de procesamiento que se ejecutan en segundo plano, es complicado, lo cual aumenta el grado de dificultad del desarrollo, y es por eso que se debe usar solamente en tareas específicas porque el manejo de las respuestas no está en función del tiempo, y las tareas se deben diseñar para funcionar de forma asincrónica, empero este tipo de diseño fue necesario para evitar la sensación de lentitud en la aplicación por parte del usuario.

La selección del umbral que compara una banda de frecuencias con respecto a la otra, se hizo para evitar la realización de la llamada telefónica en el supuesto escenario que cuando se inicie la estimulación visual (acción que inicia también la lectura de los datos provenientes de los electrodos) y el amplificador se encuentre *"al aire"*. Aunque este escenario es controlado por la calidad de contacto de los electrodos (que no muestra el botón de inicio hasta tener buena calidad de señal), puede ocurrir que el usuario se retire el amplificador de la cabeza justo cuando inicia la estimulación visual. Por consiguiente, al final de la estimulación visual una de las bandas de potencia puede ser mínimamente mayor con respecto a la otra y de no establecer un umbral, el algoritmo de procesamiento del EEG siempre encontrará una diferencia entre las bandas de potencia, sin importar que tan pequeña sea ésta. Es por eso, que de acuerdo a la experiencia de los proyectos realizados en el Laboratorio de Investigación en Neuroimagenología se tomó la decisión de establecer el umbral de diferencia en 10 %, aunque éste se puede mejorar con métodos de optimización para la selección de umbrales.

El objetivo de desarrollar una interfaz cerebro computadora móvil se cumplió. Se desarrolló una BCI que no hace uso de una computadora para obtener las señales como lo hicieron Campbell et. al. [24] en el *neurophone*. Este avance permite hacer uso de la BCI en cualquier entorno con un dispositivo móvil y un amplificador comercial de EEG. Esto resulta tan relevante que hasta enero de 2017 se encontró publicado un artículo de un proyecto similar elaborado por Elsawy et. al. [39] en el que explican el desarrollo de *MindEdit*, un editor de texto para Android™ que usa un amplificador EPOC™ y un paradigma de evento extraño

("oddball") con potenciales P300 y una matriz de Donchin. Al igual que en esta tesis los autores del proyecto de Elsawy et. al [39] refieren que el *neurophone* de Campbell et. al. [24] requiere una computadora para la adquisición de las señales y que el aporte de el *MindEdit* es la movilidad.

Sin embargo, el aporte de la Android™ BCI no se limita a la movilidad, porque el código estará disponible en GitHub para que cualquier persona que quiera obtener señales de EEG en un dispositivo móvil con Android ,y tenga el hardware requerido, pueda hacerlo utilizando éste código como base eliminando las barreras en la obtención de las señales provenientes del amplificador y no sean una limitante para el desarrollo de nuevas aplicaciones.

7.1. Trabajo futuro

El propósito de esta tesis fue el desarrollo de una aplicación en Android para crear una interfaz cerebro-computadora. Se usó un paradigma y algoritmos ya probados que se usan en los proyectos del Laboratorio de Investigación en Neuroimagenología implementados en Android. Las pruebas de rendimiento y repetitividad de la Android™ BCI en sujetos con ciertos criterios de selección quedan como trabajo futuro debido al tiempo que se ha invertido en el desarrollo del proyecto producto de todas las complicaciones inesperadas en el desarrollo.

El código se puede modificar fácilmente para que una vez que hayan sido seleccionados los contactos a los que se desea llamar y se cumplan las condiciones de la calidad de señal de los electrodos, la estimulación visual esté de forma permanente en espera de que el usuario mueva los ojos (en caso de que pueda hacerlo) para prestar atención a uno de los estimuladores visuales y aguarde a que se desencadene la respuesta fisiológica de las PPVEE. Así que, cuando la Android™ BCI detecte que una de las potencias se incrementó en una banda con respecto a la otra y se pueda concluir que lo que se desea es hacer una llamada telefónica. Para que esto sea posible también hay que modificar el código que calcula y compara la potencia.

La Android™ BCI está limitada solamente para realizar llamadas a contactos previamente almacenados en el teléfono, no obstante, una función deseable es poder hacer la introducción manual de un número de teléfono al que se desea llamar al hacer click en la región en donde se muestra el número de teléfono del contacto seleccionado.

Una característica deseable en la Android™ BCI como trabajo futuro, sería poder diseñar los filtros en una fase de entrenamiento de la BCI para sintonizar las bandas de potencia que generan los potenciales PPVEE de frecuencia fija, de acuerdo a la respuesta natural de cada

usuario, y así poder mejorar el desempeño de la aplicación personalizándola para cada uno. Es decir, se requiere de la implementación de los algoritmos para el diseño de los filtros mediante la selección de las bandas de paso, provenientes del resultado de las pruebas para entrenar a la Android™ BCI y exportar los coeficientes calculados directamente a los algoritmos de procesamiento del EEG.

Otra característica deseable de la Android™ BCI es hacer el tiempo de la estimulación visual variable, haciendo análisis de las bandas de potencia en ventanas de tiempo más cortas que analicen la potencia para determinar si se puede tomar la decisión antes de que ésta termine.

De la misma manera, el código estará disponible en GitHub para que cualquier persona que quiera obtener señales de EEG en un dispositivo móvil con Android ,y tenga el hardware requerido, pueda hacerlo utilizando éste código como base eliminando las barreras en la obtención de las señales provenientes del amplificador y no sean una limitante para el desarrollo de nuevas aplicaciones.

Bibliografía

[1] "Usb," 2014, retreived November 12 2014 url: https://en.wikipedia.org/wiki/USB. [Online]. Available: https://en.wikipedia.org/wiki/USB

[2] I. Z. Hernández, "Uso de potenciales provocados visuales de estado estacionario para el apoyo en el diagnóstico del trastorno por déficit de atención con hiperactividad," Masters Thesis, Universidad Autónoma Metropolitana, 2013.

[3] J. Axelson, *USB Embedded Hosts*. Lakeview Research, 2011.

[4] J.-J. Vidal, "Toward direct brain-computer communication," *Annual review of Biophysics and Bioengineering*, vol. 2, no. 1, pp. 157–180, 1973. [Online]. Available: http://www.cs.ucla.edu/~vidal/BCI.pdf

[5] A. N. Desney S. Tan, *Brain-Computer Interfaces Applying our Minds to Human-Computer Interaction*, D. S. T. A. Nijholt, Ed. Springer, 2010.

[6] N. Chumerin, N. V. Manyakov, A. Combaz, A. Robben, M. van Vliet, and M. M. Van Hulle, "Steady state visual evoked potential based computer gaming - the maze," *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 28–37, 2012. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30214-5_4

[7] L. A. Farwell and E. Donchin, "Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials," *Electroencephalography and clinical Neurophysiology*, vol. 70, no. 6, pp. 510–523, 1988. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0013469488901496

[8] B. Z. Allison, C. Brunner, C. Altstätter, I. C. Wagner, S. Grissmann, and C. Neuper, "A hybrid erd/ssvep bci for continuous simultaneous two dimensional cursor control," *Journal of Neuroscience Methods*, vol. 209, no. 2, pp. 299–307, Aug 2012. [Online]. Available: http://dx.doi.org/10.1016/j.jneumeth.2012.06.022

[9] B. Allison, S. Dunne, R. Leeb, J. R. Millan, and A. Nijholt, "Recent and upcoming bci progress: Overview, analysis, and recommendations," in *Towards Practical Brain-Computer Interfaces: Bridging the Gap from Research to Real-World Applications*, ser. Biological and Medical Physics, Biomedical Engineering, B. Allison, S. Dunne, R. Leeb, J. R. Millan, and A. Nijholt, Eds. Berlin: Springer Verlag, 2012, pp. 1–13, iSBN=978-3-642-29745-8.

[10] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, "Brain-computer interfaces for communication and control," *Clinical neurophysiology*, vol. 113, no. 6, pp. 767–791, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1388245702000573

[11] J. P. B.Z. Allison, Ed., *Brain-Computer Interface: New Directions*. The Scripps Research Institute, 2006, accessed september 2014. [Online]. Available: https://www.scripps.edu/news/scientificreports/sr2006/mind06polich.html

[12] G. Dornhege, *Toward brain-computer interfacing*. MIT press, 2007.

[13] Y. Wang, R. Wang, X. Gao, B. Hong, and S. Gao, "A practical vep-based brain-computer interface," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 14, no. 2, pp. 234–240, Jun 2006. [Online]. Available: http://dx.doi.org/10.1109/TNSRE.2006.875576

[14] M. A. Lebedev and M. A. Nicolelis, "Brain-machine interfaces: past, present and future," *Trends in Neurosciences*, vol. 29, no. 9, pp. 536–546, Sep 2006. [Online]. Available: http://dx.doi.org/10.1016/j.tins.2006.07.004

[15] N. Birbaumer, "Breaking the silence: brain–computer interfaces (bci) for communication and motor control," *Psychophysiology*, vol. 43, no. 6, pp. 517–532, 2006. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1111/j.1469-8986.2006.00456.x/full

[16] T. Solis-Escalante, "Diseño e implementación de una interfaz cerebro-computadora basada en potenciales evocados visuales de estado estacionario," Master's thesis, Universidad Autónoma Metropolitana, 2007.

[17] J. R. Wolpaw, D. J. McFarland, T. M. Vaughan, and G. Schalk, "The wadsworth center brain-computer interface (bci) research and development program," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 11, no. 2, pp. 1–4, 2003. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1214722

[18] C. M. Wong, B. Wang, F. Wan, P. U. Mak, P. I. Mak, and M. I. Vai, "A solution to harmonic frequency problem: Frequency and phase coding-based brain-computer

interface," in *Proc. Int Neural Networks (IJCNN) Joint Conf*, 2011, pp. 2119–2126. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6033490

[19] A. Zani and A. M. Proverbio, *The cognitive electrophysiology of mind and brain*. Academic press, 2003.

[20] C. Guger, S. Daban, E. Sellers, C. Holzner, G. Krausz, R. Carabalona, F. Gramatica, and G. Edlinger, "How many people are able to control a p300–based brain-computer interface (bci)?" *Neuroscience Letters*, vol. 462, no. 1, pp. 94–98, Sep 2009. [Online]. Available: http://dx.doi.org/10.1016/j.neulet.2009.06.045

[21] G. Schalk, D. McFarland, T. Hinterberger, N. Birbaumer, and J. Wolpaw, "Bci2000: A general-purpose brain-computer interface (bci) system," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 6, pp. 1034–1043, Jun 2004. [Online]. Available: http://dx.doi.org/10.1109/TBME.2004.827072

[22] S. Lee, Y. Shin, S. Woo, K. Kim, and H.-N. Lee, "Dry electrode design and performance evaluation for EEG based bci systems," *2013 International Winter Workshop on Brain-Computer Interface (BCI)*, Feb 2013. [Online]. Available: http://dx.doi.org/10.1109/IWW-BCI.2013.6506627

[23] A. Cotrina, A. Benevides, A. Ferreira, T. Bastos, J. Castillo, M. L. Menezes, and C. Pereira, "Towards an architecture of a hybrid bci based on ssvep-bci and passive-bci," *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Aug 2014. [Online]. Available: http://dx.doi.org/10.1109/EMBC.2014.6943847

[24] A. Campbell, T. Choudhury, S. Hu, H. Lu, M. K. Mukerjee, M. Rabbi, and R. D. Raizada, "Neurophone: brain-mobile phone interface using a wireless eeg headset," in *Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds*. ACM, 2010, pp. 3–8. [Online]. Available: http://pac.cs.cornell.edu/pubs/neurophone.pdf

[25] "ipad," 2013, retreived November 12 2013 url: https://es.wikipedia.org/wiki/IPad#Modelos. [Online]. Available: https://es.wikipedia.org/wiki/IPad#Modelos

[26] Emotiv, *Software Development Kit User Manual for Release 1.0.0.5*.

[27] H. Ekanayake, "P300 and emotiv epoc: Does emotiv epoc capture real eeg?" *The Solution Available: http://neurofeedback. visaduma. info/emotivrese arch. htm*, 2011. [Online]. Available: http://neurofeedback.visaduma.info/P300nEmotiv.pdf

[28] J. Axelson, *USB complete: the developer's guide*. Lakeview Research, 2009.

[29] "Emokit," 2012. [Online]. Available: https://github.com/qdot/emokit

[30] I. Martinovic, D. Davies, M. Frank, D. Perito, T. Ros, and D. Song, "On the feasibility of side-channel attacks with brain-computer interfaces." in *USENIX Security Symposium*, 2012, pp. 143–158.

[31] D. Selent, "Advanced encryption standard," *Rivier Academic Journal*, vol. 6, no. 2, pp. 1–14, 2010.

[32] AndroidOpenSourceProject, "Api guide usb host." [Online]. Available: https://developer.android.com/guide/topics/connectivity/usb/host.html

[33] J. N. Howard, "The rayleigh notebooks," *Applied Optics*, vol. 3, no. 10, pp. 1129–1129, 1964.

[34] L. Cao, J. Li, H. Ji, and C. Jiang, "A hybrid brain computer interface system based on the neurophysiological protocol and brain-actuated switch for wheelchair control," *Journal of Neuroscience Methods*, vol. 229, pp. 33–43, May 2014. [Online]. Available: http://dx.doi.org/10.1016/j.jneumeth.2014.03.011

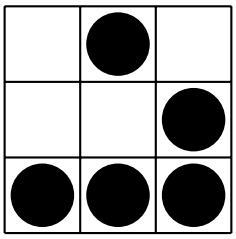
[35] A. Barachant, S. Bonnet, M. Congedo, and C. Jutten, "A brain-switch using riemannian geometry," in *5th International Brain-Computer Interface Conference 2011 (BCI 2011)*, 2011, pp. 64–67. [Online]. Available: http://www-ist.cea.fr/publicea/exl-doc/201100001904.pdf

[36] G. R. Müller-Putz, V. Kaiser, T. Solis-Escalante, and G. Pfurtscheller, "Fast set-up asynchronous brain-switch based on detection of foot motor imagery in 1-channel eeg," *Medical & biological engineering & computing*, vol. 48, no. 3, pp. 229–233, 2010. [Online]. Available: http://www0.cs.ucl.ac.uk/research/vr/Projects/PRESENCCIA/Public/presenccia_pub/sharedDocuments/presenccia_publications/Publications/wp4/tug/0667.pdf

[37] G. Pfurtscheller and T. Solis-Escalante, "Could the beta rebound in the eeg be suitable to realize a "brain switch"?" *Clinical Neurophysiology*, vol. 120, no. 1, pp. 24–29, 2009. [Online]. Available: http://www0.cs.ucl.ac.uk/research/vr/Projects/PRESENCCIA/Public/presenccia_pub/sharedDocuments/presenccia_publications/Publications/wp4/tug/0517.pdf

[38] C. J. James, "On independent component analysis based on spatial, temporal and spatio-temporal information in biomedical signals," pp. 34–37, 2009. [Online]. Available: https://doi.org/10.1007%2F978-3-540-89208-3_10

[39] A. S. Elsawy, S. Eldawlatly, M. Taher, and G. M. Aly, "MindEdit: A p300-based text editor for mobile devices," *Computers in Biology and Medicine*, vol. 80, pp. 97–106, jan 2017. [Online]. Available: https://doi.org/10.1016%2Fj.compbiomed.2016.11.014



Apéndice A

Dispositivos Android disponibles para el proyecto

A.1. Teléfono Samsung Galaxy Nexus

Este teléfono con Android es el teléfono oficial de Google, está diseñado para obtener la mejor experiencia de usuario con el sistema operativo y cumple con todas las características necesarias para la realización del proyecto. Las características detalladas se muestran en la tabla A.1 en la página 127 en la que se puede observar que tiene el tamaño de pantalla adecuado, la capacidad de procesamiento necesaria y el puerto USB OTG requerido para el desarrollo. Para conectar el "dongle" al teléfono es necesario usar un cable como el descrito en la sección 5.1.1.2.

Figura A.1: Galaxy Nexus

Características generales:	Red	- GSM 850 / 900 / 1800 / 1900 - HSDPA 850/ 900 / 1900 / 1700 / 2100	
	Anunciado	2011, Octubre	
	Tamaño	Dimensiones	135.5 x 67.94 x 8.94 mm
		Peso	135 g
	Pantalla	Tipo	- Super AMOLED - Touchscreen capacitivo - 16M colores
		Tamaño	720 x 1280 pixels, 4.65 pulgadas
Memoria y procesador	- 16GB/32GB memoria interna, 1GB RAM - Procesador TI OMAP 4460 dual-core 1.2 GHz, GPU PowerVR SGX540		
	Slot de tarjeta SD	No	
OS	Android OS, v4.3 Jelly Bean		
Características especiales:	- GPS con soporte A-GPS - Brújula digital - 3G HSDPA 21 Mbps/ HSUPA 5.76Mbps - Wi-Fi 802.11 a/b/g/n; DLNA - Bluetooth v3.0 +HS - microUSB 2.0, OTG - Soporte NFC - Salida TV vía MHL - Cancelación activa de ruido con micrófono dedicado - Acelerómetro - Controles sensibles al tacto - Sensor de proximidad - Barómetro - Giroscópico de tres ejes		
	Batería	Standard, Li-Ion 1750 mAh	
	Tiempo de espera	Hasta 290 h (2G) / Hasta 270 h (3G)	

Tabla A.1: Características generales del Samsung Galaxy Nexus

A.2. Tableta Asus Transformer Prime (TF201)

Esta tableta en particular, tiene la capacidad de conectarle un *dock* con un teclado (ver figura A.3 en la página 130), una batería adicional, un puerto USB tipo A y una ranura para tarjeta SD, es por eso, el distintivo transformer en el nombre. Tiene características muy similares a las del teléfono en cuestión de procesamiento, así que la aplicación debería comportarse de manera similar en ambos dispositivos. Hay dos formas de conectar el "dongle" a la tableta, una a través del *dock* con el teclado que tiene el puerto USB tipo A, y la otra es sin usar el *dock*, conectando un adaptador de un puerto propietario de ASUS a USB (ASUS USB kit). Las características generales de la tableta se pueden ver en la tabla A.2 en la página 129.

Figura A.2: Asus USB kit

Diseño	Dimensiones	263 x 180.8 x 8.3 mm
	Peso	586 g
	Sistema operativo	Android 4.1
	Interfaz	ASUS Waveshare UI
	Teclado qwerty	Sí mediante dock
Pantalla	Tamaño pantalla	10.1 pulgadas
	Táctil	Multitáctil
	Tipo	Capacitiva
	Tecnología	Super IPS LCD
	Resolución	1280 x 800 píxeles
	Densidad de píxeles	149 ppi
Hardware	Chipset	Nvidia Tegra 3
	Procesador	4 Núcleos a 1.3 GHz Cortex-A9
	GPU	ULP GeForce
	Memoria RAM	1 GB
	Memoria interna	64 GB
	Slot para tarjeta de memoria	Hasta 32 GB MicroSD
Conectividad	Bluetooth	A2DP
	Wi-Fi	802.11 b / g / n
	GPS	A-GPS
	USB OTG	Sí
Batería	Autonomía	En espera: 300 horas En uso: 12 horas
	Tipo	Li-Po No extraíble

Tabla A.2: Características generales de la tableta TF201

Figura A.3: Tableta Asus TF201

Apéndice B

Aplicaciones en Android

B.1. Conceptos fundamentales

- Las aplicaciones en Android se programan en Java, que es un lenguaje de programación de alto nivel.
- Las interfaces gráficas en Android se programan en XML, que es un lenguaje estructurado.
- Cada proceso o aplicación tiene su propia máquina virtual. Lo que significa que está encapsulada en el entorno en que se ejecuta la aplicación con todas las variables locales a su disposición, sin embargo, el intercambio de información con el sistema operativo y con otras aplicaciones se hace a través de mensajes.
- Las aplicaciones no pueden tomar el control del sistema y éste las puede cerrar cuando sea necesario, ya sea por falta de memoria o porque algún suceso de mayor importancia ocurre en un entorno externo a la aplicación, e.g. una llamada telefónica entrante.
- El acceso a los recursos del sistema se hace a través de permisos en un documento llamado *"manifest"*. En el "manifest" se declaran todos los recursos que la aplicación necesita y éste comunica al usuario la información que la aplicación necesita para poder funcionar. Este paso es muy importante, porque un programador malintencionado puede extraer información personal del usuario almacenada en el teléfono. Si no están declarados los permisos en el "manifest" y el usuario no los aprueba, la aplicación no puede de ninguna manera acceder a esta información. Estos son los permisos que se aprueban antes de instalar una aplicación desde la tienda de aplicaciones.

B.1.1. Componentes de una aplicación

Los componentes de una aplicación son las piezas fundamentales para su creación, cada componente está considerado como un punto distinto desde el cual el sistema puede interactuar con la aplicación. No todos los componentes permiten la interacción con el usuario y algunos dependen entre si, pero cada uno existe como una entidad teniendo su rol específico en un todo que conforma a la aplicación.

- Actividades ("Activities")

  Una actividad representa una única pantalla con una interfaz de usuario teniendo todos los elementos necesarios para llevar a cabo una actividad específica; puede ser invocada por otras aplicaciones si así se especifica.

- Servicios ("Services")

  Un servicio es un componente que se ejecuta en segundo plano para realizar tareas que requieren de mucho tiempo de ejecución o para realizar instrucciones de procesos remotos. Éstos no tienen tienen interfaz de usuario.

- Proveedores de contenido ("Content provider")

  Un proveedor de contenido administra la información que la aplicación comparte, se pueden almacenar datos en un sistema de archivos, en una base da datos SQLite, en la web o en algún otro medio de almacenamiento al que la aplicación tenga acceso. A través del proveedor de contenido, otras aplicaciones pueden hacer solicitudes de información o incluso modificar la información de la aplicación (si ésta así lo permite). Pueden ser también usados para leer o escribir información privada, que no es compartida, dentro de la misma aplicación.

- Receptores de difusión ("Broadcast receivers")

  Estos componentes responden a los anuncios globales que realiza el sistema operativo, e.g. el sistema informa que un dispositivo USB se ha conectado o que se ha apagado la pantalla del dispositivo. La aplicaciones también pueden iniciar un *broadcast*, e.g. informar al sistema que se quiere hacer una llamada telefónica. Estos componentes tampoco tienen interfaz de usuario.

Intents

Tres de los cuatro tipos de componentes de una aplicación, actividades, servicios, y *broadcast receivers*, son activados mediante un mensaje asincrónico llamado intent. Los intents enlazan componentes individuales entre sí, al momento de la ejecución. Se pueden ver como mensajeros que solicitan una acción de otros componentes, no importando si el componente pertenece a la aplicación misma o a otra. Pueden ser implícitos (generales) o explícitos (con nombre y apellido).

El *"manifest"*

El "manifest" es un documento que se encuentra en el directorio raíz de la aplicación, está escrito en XML y su nombre tiene que ser *"AndroidManifest.xml"*. Es el documento maestro que le permite al sistema operativo saber de que componentes está conformada la aplicación y cuales son sus funciones. Además de declarar los componentes de la aplicación, el "manifest" tiene las siguientes funciones:

- Identifica todos los permisos que se deben solicitar al usuario de acuerdo a las funciones de la aplicación.
- Especifica la versión mínima de la API de Android que la aplicación requiere para funcionar. En este caso es la versión 12 de la API, que corresponde a la versión 3.1 de Android (Honeycomb) porque a partir de ella se tiene soporte para USB "host".
- Especifica la versión objetivo de la API utilizada para el diseño. En este caso, la versión 15 de la API que corresponde a la versión 4.0.4 (Ice Cream Sandwich) en base a las guías de diseño de Android.
- Especifica las necesidades de hardware y software que la aplicación requiere, en este caso, la capacidad del teléfono para funcionar como USB "host".

B.2. Procesamiento multitarea en Android

El poder realizar múltiples tareas de forma paralela o simultánea en el dispositivo es una herramienta no sólo útil sino fundamental en el desarrollo de esta aplicación en particular. El manejo de hilos de procesamiento que se ejecutan en segundo plano es complicado, lo

que aumenta el grado de dificultad del desarrollo, y es por eso que se debe usar solamente en tareas específicas ya que el manejo de las respuestas no está en función del tiempo y las tareas se deben diseñar para funcionar de forma asincrónica, sin embargo el no hacerlo de esta forma provoca retrasos en la interfaz gráfica y puede dar la sensación de lentitud al usuario. Hay tareas como el procesamiento de la información y el acceso al puerto USB que se tienen que hacer en un hilo distinto al principal.

IntentService

La forma de hacer multitarea mediante hilos en una "activity" en Android es mediante un recurso llamado *IntentService* que es una estructura para ejecutar operaciones en un hilo distinto al principal, esto permite que procesos largos no afecten la respuesta de la interfaz gráfica de usuario y viceversa. Las limitantes de un *IntentService* son:

- No puede interactuar directamente con la interfaz de usuario, para mostrar el resultado de una acción en pantalla e.g. para cambiar la transparencia de un estimulador se tiene que mandar la instrucción a la actividad correspondiente.
- Las peticiones de acciones (work requests) se ejecutan de manera secuencial. Si una operación está siendo ejecutada en un IntentService, y se manda una petición para realizar otra acción ésta empieza una vez que la anterior finaliza.
- Una operación que está siendo ejecutada en un IntentService no puede ser interrumpida.

B.3. Requerimientos de diseño de la aplicación

La aplicación fue diseñada tomando en cuenta requerimientos enfocados a la funcionalidad, son requisitos mínimos que se deben cubrir para que se pueda cumplir el objetivo de la aplicación. La implementación de estos requerimientos se puede llevar a cabo de muchas maneras, más adelante, se explica en código, la forma que se creyó adecuada para cumplir con el objetivo. Desde la concepción del proyecto se pensó en la portabilidad, aterrizando el concepto del punto 4 de la sección 2.2.3 en la página 22, uno de los pilares del diseño; es decir, poder usar la aplicación en un teléfono o en una tableta.

1. Que la aplicación sea capaz de detectar cuando se conecta el "dongle" de algún EPOC™ y se inicie de forma automática.

2. Que la aplicación sea capaz de detectar cuando se desconecte el "dongle" de manera inesperada, y se le avise al usuario que no puede funcionar sin el "dongle" conectado, y de no hacerlo así, se cierre automáticamente.

3. Que se permita al usuario seleccionar dos personas entre su lista de contactos para llamar, asignando cada uno a una posición distinta en la GUI y que le permita distinguir a cual de ellos desea llamar mediante la estimulación visual de PPVEE.

4. Que la aplicación pueda funcionar en pantallas de diferentes tamaños con una densidad de píxeles distinta, es decir, que se adapte al tamaño de pantalla de múltiples dispositivos e.g. tabletas y teléfonos.

5. Que la aplicación siempre se muestre en modalidad horizontal (landscape), y haga caso omiso a los giros de la pantalla.

6. Que todos los comentarios del código estén en inglés para hacer el código entendible para otras personas que quieran hacer uso de él.

¿Por qué hacer compatible la aplicación con un tableta?

Las tabletas no tienen la capacidad de realizar llamadas de voz usando las redes telefónicas. Entonces, ¿por qué diseñar la aplicación para poder ser usada en una tableta?. Al ser una BCI que usa el paradigma PPVEE, un tamaño de pantalla más grande permite tener más alejados los estimuladores visuales entre sí, hecho que es recomendable para evitar la generación de una respuesta autonómica que contenga información de ambas estimulaciones. Y el problema de la generación de llamadas puede ser solucionado usando alguna aplicación que realice llamadas de voz sobre IP (VoIP). Este es un punto que se soluciona en la implementación, lo único que se hace es lanzar un intent para hacer la llamada y el sistema se encarga de asignar la tarea a cualquier aplicación que sea capaz de realizarla, e.g. Skype.

B.4. Activities

Este componente se explica con más detalle debido a que la aplicación final está conformada por dos activities débilmente ligadas, y todo en ella depende de como se comportan y comunican entre sí. Son componentes que proveen de una interfaz gráfica con la que el usuario puede interactuar para realizar alguna acción. Generalmente, una "activity" en la aplicación, es la pantalla principal, que es la interfaz que el usuario ve al iniciarla, y desde

ahí navega a otras activities. Cada vez que una nueva "activity" entra en acción, la "activity" previa entra en un estado de suspensión (On Stop), pero el sistema operativo la conserva en el *stack* de activities (llamado "back stack"), para que en caso de que el usuario presione el "botón atrás", se pueda regresar a la "activity" previa. Cuando una "activity" entra al estado On Stop debe liberar los objetos que tiene en uso, como en este caso, la conexión USB, para que otra "activity" haga uso de ella, y cuando regresa al estado "On Resume" se puedan restablecer todas las conexiones requeridas.

El diagrama de flujo de la figura B.1 en la página 138 muestra el ciclo completo de vida de una "activity" y las transiciones entre los estados por los que transita.

Con estos conceptos, ahora se tiene una idea clara de como realizar la aplicación.

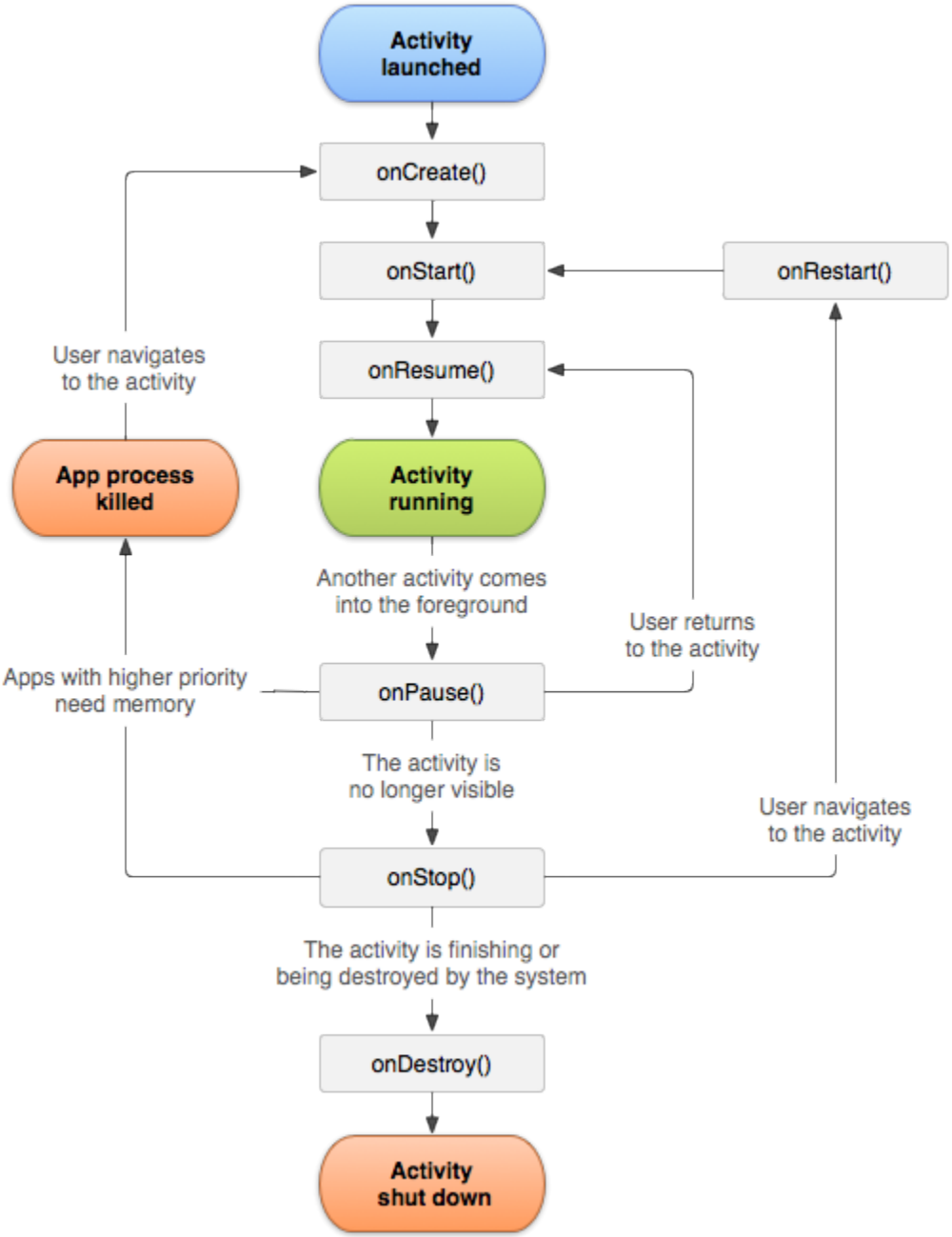


Figura B.1: Ciclo de vida de una "activity".

Fuente www.android.com

Apéndice C

Secciones de código

C.1. “Manifest”

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.uam.android_bci"
    android:versionCode="1"
    android:versionName="1.0" >

<!-- Permission to access contacts list-->
    <uses-permission android:name="android.permission.READ_CONTACTS"/>

    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="18" />

<application

   <activity
        android:name=".MainActivity"
        android:label="AndroidBCI"

        android:screenOrientation="userLandscape"
        android:configChanges="orientation|keyboardHidden|keyboard">
<!--Change to lock the screen orientation to landscape-->

<!--Intent-filter to detect the epoc dongle connection-->
        <intent-filter>
<!-- USB HOST action -->
            <action android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED" />
        </intent-filter>
```

Sección de código 14: AndroidManifest.xml

```
<!--epoc_dongle meta-data file indicates which hardware to detect-->
        <meta-data
            android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"
            android:resource="@xml/epoc_dongle" />

<!-- Intent-filter to detect epoc dongle disconnection -->
        <intent-filter>
            <action android:name="android.hardware.usb.action.USB_DEVICE_DETACHED" />
        </intent-filter>

        <meta-data
            android:name="android.hardware.usb.action.USB_DEVICE_DETACHED"
            android:resource="@xml/epoc_dongle" />
    </activity>

    <activity
        android:name=".StimulatorsFullScreen"
        android:label="@string/title_activity_stimulators_full_screen"
        android:parentActivityName="com.uam.android_bci.MainActivity"
        android:theme="@android:style/Theme.Black.NoTitleBar"
        android:screenOrientation="userLandscape"
        android:hardwareAccelerated="true"
        android:configChanges="orientation|keyboard|keyboardHidden|screenSize">
<!-- Change to lock the screen orientation to landscape-->
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.uam.android_bci.MainActivity" />
    </activity>
</application>

</manifest>
```

Sección de código 15: AndroidManifest.xml

C.2. Métodos y procedimientos implementados

```
/*BroadcastReceiver that notifies when the epoc dongle is DETACHED
or disconnected from the usb hub*/
BroadcastReceiver mUsbReceiver = new BroadcastReceiver(){
  public void onReceive(Context context, Intent intent){
    String action= intent.getAction(); //Catch the action metadata
    //Compare the evaluation
    if(UsbManager.ACTION_USB_DEVICE_DETACHED.equals(action)){
      //Verify that the dongle is really disconnected
      boolean msg= dongle.isDongleConnected(USBmanager);
      if(!msg){//If it was really disconnected
          UsbDevice device =
          (UsbDevice)intent.getParcelableExtra(UsbManager.EXTRA_DEVICE);
           if (device != null) {
           /*Initializes the dialog that notifies that the dongle was
           disconnected and the application will close*/
             DialogFragment newFragment = new ExitDialog();
             newFragment.show(getFragmentManager(), "BeforeExit");
             if(ExitButtonHandle){
               //Close the application
               finish();
             }
           }
       }
   }
};
```

Sección de código 16: Código del listener que detecta la desconexión del "dongle" en la Android™ BCI

```
/* Enum which defines the USB status (Hardware dongle) to connected,
disconnected or pending || Note: Pending validation should be implemented
checking the enum not the boolean value of the enum */
public enum EpocDongleUSBStatus{
  CONNECTED (true),
  DISCONNECTED (false),
  PENDING (false);/*false because it's not connected yet and can cause
  an error if the reading method is started*/
  private boolean value;
  private EpocDongleUSBStatus(boolean value){
    this.value= value;
  }
  public boolean getValue(){
    return value;
  }
}
```

Sección de código 17: Enumeración del estado general del "dongle" en la Android™ BCI

```
/* Enum which defines the interface status to connected or disconnected
|| Note: Once the interface is connected it can't disconnect */
public enum EpocDongleInterfaceStatus{
  CONNECTED (true),
  DISCONNECTED (false);
  private boolean value;
  private EpocDongleInterfaceStatus(boolean value){
    this.value= value;
  }
  public boolean getValue(){
    return value;
  }
}

/* Enum which defines the USB connection status to connected or
disconnected || Note: This is important because trying to access the
dongle with a previous open connection will cause an error */
public enum EpocDongleConnectionStatus{
  CONNECTED (true),
  DISCONNECTED (false);
  private boolean value;
  private EpocDongleConnectionStatus(boolean value){
     this.value= value;
  }
  public boolean getValue(){
     return value;
  }
}
```

Sección de código 18: Enumeración del estado de la interfaz y la conexión de datos del "dongle" en la Android™ BCI

```
public static EpocDongleUSBStatus getUSBStatus(){
  return usbStatus;
}

public static EpocDongleInterfaceStatus getInterfaceStatus(){
  return interfaceStatus;
}

public static EpocDongleConnectionStatus getConnectionStatus(){
  return endpointStatus;
}
```

Sección de código 19: Getters de las Enumeraciones del "dongle" en la Android™ BCI

```
//SETTERS
public static boolean setUSBStatus(EpocDongleUSBStatus status){
  try{
     usbStatus= status;
     return true;
  }catch(Exception e){
    System.err.println("The usbStatus assignation couldn't be made");
    return false;
  }
}

public static boolean setInterfaceStatus(EpocDongleInterfaceStatus status){
  try{
    interfaceStatus= status;
    return true;
  }catch(Exception e){
    System.err.println("The interfaceStatus couldn't be changed");
    return false;
  }
}

public static boolean setConnectionStatus(EpocDongleConnectionStatus status){
  try{
    endpointStatus= status;
    return true;
  }catch(Exception e){
    System.err.println("The endPointStatus couldn't be changed");
    return false;
  }
}
```

Sección de código 20: Setters de las Enumeraciones del "dongle" en la Android™ BCI

```
private UsbManager USBmanager; //Controls all the USB connections

// Otras lineas del programa

//Implicit Intents to communicate with the system or external applications
//Intent made to be able to detect the object that represents the epoc
//dongle when connected
Intent epocConnected = new Intent();
UsbDevice epocUSBDevice =
(UsbDevice) epocConnected.getParcelableExtra(UsbManager.EXTRA_DEVICE);

// Otras lineas del programa

USBmanager = (UsbManager) getSystemService(Context.USB_SERVICE);
//Tabla hash de los dispositivos USB conectados
HashMap<String, UsbDevice> deviceList = USBmanager.getDeviceList();
//Iterator que recorre la tabla hash hasta que encuentra el dongle del epoc
Iterator<UsbDevice> deviceIterator = deviceList.values().iterator();

do{
  //test to make assignations
  boolean test=true;
  //It restarts in every iteration
  UsbDevice temporalUSBDevice=null;
  int tempProductId = 0;
  int tempVendorId = 0;
  String tempName=null;

  try{
    UsbDevice temporalDevice = deviceIterator.next();
    temporalUSBDevice=temporalDevice;
    tempProductId = temporalDevice.getProductId();//try bacuase this can be null
    tempVendorId = temporalDevice.getVendorId();//try bacuase this can be null
    tempName = temporalDevice.getDeviceName();
  }catch (NoSuchElementException e){
    setUSBStatus(usbStatus.DISCONNECTED);
  }
```

Sección de código 21: Código que inicia la conexión del "dongle" en la Android™ BCI parte 1/2

```
  if (tempProductId==productId && tempVendorId==vendorID){//is it the epoc dongle?
    int tempIntents1=0;
    int tempIntents2=0;
    /*Do the assignations with do-while structure to verify them,
    taking advantage to the boolean return of the setters*/
    do{
      //Change the usbStatus to connected to make the do loop work
      test=setUSBStatus(EpocDongleUSBStatus.CONNECTED);
      //Open a USB connection and get the serial
      EpocUSBConnection= manager.openDevice(temporalUSBDevice);
      if(test){
        Epoc=temporalUSBDevice;
        break;
      }
      /*The next security step is made to avoid infinite loop during connection,
      if connection is not made after 5 attempts sends a message to the user*/
      if(tempIntents1>5){
        Toast noConnection = Toast.makeText(context, "5 attempts trying to to
        make the dongle usbConnection exceeded, connection couldn't be made",
        Toast.LENGTH_SHORT);
        noConnection.show();
        break;
      }else{
        try{
          Thread.sleep(500);
        }catch(InterruptedException e){
          Log.e(DEBUG_TAG, "Interruption made during waiting for connection");
      }
    }
  }while(test);
}while(deviceIterator.hasNext());
return usbStatus.getValue();
}
```

Sección de código 22: Código que inicia la conexión del "dongle" en la Android™ BCI parte 2/2

```
if(generateKey(serialNumber))
  {
    generatedKey= true;
  }else{
    generatedKey= false;
  }

///***METHODS***///
public boolean generateKey(String Serial){
try{
  key[0]= (byte)Serial.charAt(15);
  key[1]= 0x00;
  key[2]= (byte)Serial.charAt(14);
  key[3]= 0x54;
  key[4]= (byte)Serial.charAt(13);
  key[5]= 0x10;
  key[6]= (byte)Serial.charAt(12);
  key[7]= 0x42;
  key[8]= (byte)Serial.charAt(15);
  key[9]= 0x00;
  key[10]= (byte)Serial.charAt(14);
  key[11]= 0x48;
  key[12]= (byte)Serial.charAt(13);
  key[13]= 0x00;
  key[14]= (byte)Serial.charAt(12);
  key[15]= 0x50;
  setSerial(Serial);
  return true;
  }catch (Exception e){
    return false;
  }
}
```

Sección de código 23: Generación de la llave de descifrado en la Android™ BCI

```
byte[] encryptedDataQualitySignal= new byte[32];
byte[] decryptedDataQualitySignal= new byte[32];
//This opens the USB port data connection
epocUsbConnection= USBmanager.openDevice(epocUSBDevice);
if(getInterfaceStatus().getValue()) {
  try {
    //HERE IS WHERE THE CONNECTION IS CLAIMED
    tempClaimInterface = epocUsbConnection.claimInterface(epocInterface, true);
    Log.i(DEBUG_TAG,"Interface claimed");
  } catch (Exception e) {
    Log.i(DEBUG_TAG, "Cannot claim the interface");
  } finally {
    if (tempClaimInterface == true) {
      setInterfaceConnectionStatus(EpocDongleInterfaceConnectionStatus.CLAIMED);
    } else {
      setInterfaceConnectionStatus(EpocDongleInterfaceConnectionStatus.NOTCLAIMED);
    }
  }
}
```

Sección de código 24: Reservar la interfaz para establecer la extracción de datos del "dongle" en la Android™ BCI

```
while(MainActivity.showQualityFlag && getInterfaceConnectionStatus().getValue()){
  requestWaitCounter=0;//The counter restarts
  ByteBuffer buffer= ByteBuffer.allocate(32);
  UsbRequest request= new UsbRequest();
  request.initialize(epocUsbConnection,epocEndpoint);//This is data transfer begins
  request.queue(buffer, 32);//Starts a request for data in the buffer
  if(epocUsbConnection.requestWait()==request){
  //The case where the data package is received (notified via requestWait)
    Log.i(DEBUG_TAG,"Receiving data");
    buffer.position(0);
    buffer.get(encryptedDataQualitySignal,0,32);//Data is encrypted here
    requestWaitCounter=0;//Reset the counter every time data is taken
    if(keyGenerated){
      try{
        Cipher cipher= Cipher.getInstance("AES/ECB/NOPADDING");
        cipher.init(cipher.DECRYPT_MODE, AESKey);
        decryptedDataQualitySignal= cipher.doFinal(encryptedDataQualitySignal);
        Log.i(DEBUG_TAG,"DECRYPTED DATA SUCCESFULLY");
      }catch(IllegalBlockSizeException e){
        System.err.println("Error descipher IllegalBlockSizeException");
      }catch(BadPaddingException e){
        System.err.println("Error descipher BadPaddingException");
      }catch(IllegalStateException e){
        System.err.println("Error descipher IllegalStateException");
      }catch(NoSuchPaddingException e){
        System.err.println("Error descipher NoSuchPaddingException");
      }catch(NoSuchAlgorithmException e){
        System.err.println("Error descipher NoSuchAlgorithmException");
      }catch(InvalidKeyException e){
        System.err.println("Error descipher InvalidKeyException");
      }
// Do something with the data
}
```

Sección de código 25: Extracción de datos del “dongle” en la Android™ BCI

C.3. Layouts

```
    <!--Left horizontal layout-->
<LinearLayout
    android:layout_width="0dp"
    android:layout_height="fill_parent"
    android:orientation="horizontal"
    android:layout_weight="6"
    android:visibility="visible"
    >
    <LinearLayout
        android:layout_height="fill_parent"
        android:layout_width="0dp"
        android:orientation="horizontal"
        android:layout_weight="1">
        </LinearLayout>
    <LinearLayout
        android:layout_height="fill_parent"
        android:layout_width="0dp"
        android:orientation="vertical"
        android:layout_weight="4">
         <Space
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1"/>
```

Sección de código 26: "Layout" vertical del extremo izquierdo parte 1/4

```
            <AutoCompleteTextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/searchContact7Hz"
        android:hint="@string/SearchContactSpace"
        android:gravity="center"
        android:singleLine="true"
        android:layout_weight="1"
        android:layout_gravity="center"
        android:focusableInTouchMode="true"
        android:imeOptions="actionDone"
        android:background="@color/white"
        />
    <Space
        android:layout_width="fill_parent"
        android:layout_height="0px"
        android:layout_weight="1"/>
    <ImageButton
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/imageContact7Hz"
        android:layout_gravity="center"
        android:layout_weight="5"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"/>
    <Space
        android:layout_width="fill_parent"
        android:layout_height="0px"
        android:layout_weight="1"/>
```

Sección de código 27: “Layout” vertical del extremo izquierdo parte 2/4

```
<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:inputType="phone"
    android:text="@string/EnterTelephoneNumber"
    android:gravity="center"
    android:id="@+id/editText7Hz"
    android:layout_gravity="center"
    android:layout_weight="1"
    android:background="@color/white"
    android:ellipsize="none"
    android:singleLine="true"
    android:textColor="@color/black_overlay"/>
<Space
    android:layout_width="fill_parent"
    android:layout_height="0dp"
    android:layout_weight="1" />
<LinearLayout
    android:orientation="horizontal"
    android:layout_height="0dp"
    android:layout_width="match_parent"
    android:layout_weight="2">
    <Space
        android:layout_width="0dp"
        android:layout_height="fill_parent"
        android:layout_weight="1" />
```

Sección de código 28: “Layout” vertical del extremo izquierdo parte 3/4

```
        <TextView
            android:layout_width="0dp"
            android:layout_height="fill_parent"
            android:id="@+id/contactQualityO1"
            android:layout_gravity="center"
            android:layout_weight="1"
            android:background="@color/signal_quality_none"
            android:textAlignment="center"
            android:gravity="center"
            android:text="@string/O1QualityIndicator"
            />
        <Space
            android:layout_width="0dp"
            android:layout_height="fill_parent"
            android:layout_weight="1" />
    </LinearLayout>
    <Space
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1" />
</LinearLayout>
```

Sección de código 29: “Layout” vertical del extremo izquierdo parte 4/4

C.4. Resultado del cálculo de la potencia

```
Pw_7_14=Pw_01_7_14+Pw_02_7_14; //7 and 14 Hz total power
Pw_11_22=Pw_01_11_22+Pw_02_11_22; //11 and 22 Hz total power

//More code

if (AquisitionFinished==true){
  if(Pw_7_14>(1.1*Pw_11_22)){//Multiplies by 1.1 to make sure
                             //it is greater by ten percent
    Intent returnIntent =new Intent();
    returnIntent.putExtra("result",7);
    setResult(RESULT_OK,returnIntent);
  }else if (Pw_11_22>(1.1*Pw_7_14)){
    Intent returnIntent =new Intent();
    returnIntent.putExtra("result",11);
    setResult(RESULT_OK,returnIntent);
  }else {
    Intent returnIntent =new Intent();
    setResult(RESULT_CANCELED,returnIntent);
    Log.i(DEBUG_TAG,"onActivityResult CANCELED No power differences");
  }
}else{
  Intent returnIntent = new Intent();
  setResult(RESULT_CANCELED,returnIntent);
  Log.i(DEBUG_TAG,"onActivityResult CANCELED unfinished aquisition");
}
```

Sección de código 30: Envío del resultado del cálculo de la potencia después de la estimulación

Apéndice D

Flujo de acciones en el ciclo de la "activity" principal

Los estados en los que se encuentra la actividad en su ciclo de vida son los correspondientes a la figura B.1 en la página 138. Es muy importante tener bien definidos los roles de cada estado de la aplicación para así poder saber cuales son las acciones que se deben llevar a cabo. A continuación se describen las acciones importantes para el funcionamiento de la "activity" principal:

- onCreate
  Este estado se invoca cuando la aplicación es ejecutada o cuando ésta fue cerrada y se regresó a ella.
  - Inicialización de las variables necesarias en la "activity" principal.
  - Se dan de alta los listeners de los botones de start y los listeners para la selección de los contactos.
  - Se establece el tiempo de adquisición (en caso de querer modificarlo esta es la sección en donde hay que hacerlo).
  - Se reservan los espacios de memoria para la lectura de datos del "buffer" y las variables del hardware.
  - Se registra el broadcast que detecta la desconexión del "dongle".
  - Se inicializan las banderas que marcan el estado de las acciones de control.

- onStart
  Este estado se invoca después de onCreate y cuando se regresa del estado onStop, es decir, después de que la actividad de estimulación PPVEE termina.

  - En este estado la acción más importante es recuperar la conexión de datos en caso de que se haya cerrado la conexión al pasar a la "activity" de estimulación para poder seguir actualizando la información de la calidad de contacto de los electrodos O1 y O2.
  - Además hay que recuperar la información del estado en el que se dejó la "activity" al regresar de la estimulación visual.

- onResume
  Este es el estado en el que se encuentra la aplicación cuando se tiene ocupada toda la pantalla y ya se han mostrado todos los elementos visuales en espera de la interacción con el usuario.

  - Se verifica si la llave de descifrado fue generada previamente para no calcularla cada vez que se regresa a este estado.
  - Inmediatamente se inicia el hilo que muestra la calidad de contacto de los electrodos O1 y O2 en caso de que el "dongle" esté conectado (se verifica previamente).
  - Después de ejecutar el hilo que muestra la calidad de contacto de los electrodos, la aplicación queda en "espera" de la interacción del usuario a través de los listeners.

- onPause
  Este estado se invoca cuando la "activity" principal pasa a segundo plano o ya no se encuentra visible para el usuario, en él se deben poner en "pausa" la actualización de la calidad de contacto de los electrodos y guardar el estado actual.

  - Detener la actualización de la calidad de contacto de los electrodos mediante las banderas de control.
  - Cerrar la comunicación con los puertos USB y liberar la interfaz.
  - Actualizar el estado de las banderas de control de acuerdo al estado correspondiente.

- onDestroy
  Este estado se invoca solamente cuando se ha dado la instrucción de cerrar la aplicación, en él hay que liberar todos los recursos que usa la aplicación para que no se sigan consumiendo recursos que afecten la vida de la batería del teléfono.

- Dar de baja los listeners que pertenecen a la "activity", en este caso, el que detecta la desconexión del "dongle", porque que si este se vuelve a conectar al dispositivo, éste debe ser reconocido.
- Cerrar todas las comunicaciones que se tengan con el hardware para evitar errores si se desconecta el "dongle" una vez que se ha cerrado la aplicación.

- onBackPressed
  Este estado no se encuentra dentro del ciclo normal de una "activity" como se puede ver en la figura B.1 en la página 138, sin embargo, es muy importante saber cómo se debe comportar la "activity" principal cuando el usuario presiona el *botón de atrás*. Se tomó la decisión, que cuando la aplicación se encuentra en la "activity" principal y el usuario presione el botón de atrás se llama a la instrucción finish() que a su vez llama al ciclo onDestroy que cierra la aplicación. Porque de lo contrario, se haría una llamada a la cola (o stack) de actividades del sistema, acción que dejaría la aplicación en un estado de onPause y no se liberarían los recursos de hardware de forma correcta.

Apéndice E

Flujo de acciones en el ciclo de la "activity" de estimulación visual

- onCreate
  Este estado se invoca cuando la aplicación es ejecutada o cuando la aplicación fue cerrada y se regresó a ella.
  - Inicialización de las variables necesarias.
  - Recibir la información necesaria proveniente de la "activity" principal
  - Se reservan los espacios de memoria para la lectura de datos del "buffer" y las variables del hardware.
  - Se inicializan las banderas que marcan el estado de las acciones de control.
- onStart
  Este estado se invoca después de onCreate y cuando se regresa del estado onStop, es decir, después de que la actividad de estimulación PPVEE termina.
  - En este estado la acción más importante es reservar la interfaz del puerto USB para poder leer los datos.
  - Se establecen los intervalos de tiempo para generar la estimulación visual.
- onResume
  Este es el estado en el que se encuentra la aplicación cuando se tiene ocupada toda la pantalla y ya se han mostrado todos los elementos visuales.

- Se espera a que todos los elementos hayan sido mostrados en la pantalla y posteriormente se inicia el hilo que genera los estimuladores visuales.
- En la muestra 257 se inicia el cálculo de la potencia en ese instante de tiempo.
- En la muestra 512 se guarda la respuesta y se manda la instrucción de cerrar la "activity" de los estimuladores visuales.

- onPause
  Este estado se invoca cuando la "activity" de los estimuladores visuales pasa a segundo plano o ya no se encuentra visible para el usuario, en este caso, no se puede entrar en un estado de "pausa", la acción correcta es avisar al usuario que se canceló la estimulación visual.

  - Detener los estimuladores visuales.
  - Cerrar la comunicación con los puertos USB y liberar la interfaz.
  - Actualizar el estado de las banderas de control de acuerdo al estado correspondiente.

- onDestroy
  Este estado se invoca solamente cuando se ha dado la instrucción de cerrar la aplicación. En él hay que liberar todos los recursos que usa la aplicación para que no se sigan consumiendo recursos que afecten la vida de la batería del teléfono.

  - Cerrar todas las comunicaciones que se tengan con el hardware para evitar errores si se desconecta el "dongle" una vez que se ha cerrado la aplicación.

- onBackPressed
  Este estado no se encuentra dentro del ciclo normal de una "activity", en este caso si se accede a este estado se tiene que guardar el valor $RESULT\_CANCELED$ en la respuesta para poder mostrar el mensaje al usuario diciendo que la "activity" de los estimuladores fue cancelada.

Apéndice F

Resultados completos de la validación de los filtros

Resultados completos para tres sujetos con señales del electrodo O1.

. . .	Número de segmento	Clasificación del experto	Estímulo visual real	Clasificación filtrado	Resultado	Obs
	1	11	11	7	Erróneo	**
Sujeto 1	2	7	7	7	Correcto	
	3	11	11	11	Correcto	
	4	7	11	11	Erróneo	**
	5	7	11	7	Correcto	
	6	7	7	7	Correcto	**
	7	7	7	7	Correcto	
	8	11	11	11	Correcto	
	9	11	7	11	Correcto	
	10	11	7	11	Correcto	
Sujeto 2	1	7	7	7	Correcto	
	2	11	7	7	Erróneo	
	3	11	11	11	Correcto	
	4	7	7	11	Erróneo	**
	5	11	11	11	Correcto	**
	6	11	11	11	Correcto	
	7	7	7	7	Correcto	
	8	7	7	7	Correcto	
	9	11	7	7	Erróneo	
	10	11	11	11	Correcto	
Sujeto 3	1	7	7	7	Correcto	
	2	7	7	7	Correcto	
	3	7	7	7	Correcto	
	4	11	11	7	Erróneo	
	5	11	11	7	Erróneo	
	6	11	11	7	Erróneo	**
	7	11	11	11	Correcto	
	8	11	11	11	Correcto	
	9	7	7	7	Correcto	
	10	7	7	7	Correcto	

Tabla F.1: Tabla de resultados electrodo O1 (** Diferencia menor al 10 % entre el cálculo de la potencia de ambas bandas de filtrado)

Apéndice G

Procesamiento del EEG con Análisis por Componentes Independientes

El principio de separación de fuentes se basa en la propiedad en la que dos o más señales no están correlacionadas de origen, siendo fundamental su importancia en el análisis de señales que son consideradas mezclas de señales estadísticamente independientes con respecto a las demás. Un análisis por componentes independientes funciona mejor con grandes cantidades de datos, en el EEG entre mayor sea el número de electrodos con los que se adquiere la señal y mayor sea el tiempo de adquisición, la separación de fuentes será mejor. El resultado del procesamiento, son vectores que representan a los componentes independientes (se asume la no correlación). En la figura G.1 en la página 167 se puede ver el componente independiente resultado del análisis del EEG de todos los electrodos del EPOC en la aplicación Android™ BCI con veinte segundos de señal, producto de la concatenación de cuatro adquisiciones de cinco segundos con el usuario observando la estimulación visual a 11 Hz. El análisis de frecuencia del componente independiente 26 se puede ver en la figura G.2 en la página 167 en el cual se puede observar claramente un pico en el espectro de potencia a 10.5 Hz, producto de la separación de la fuente correspondiente a las neuronas sincronizadas a la frecuencia de la estimulación visual (11 Hz). Este método, tiene resultados prometedores para el análisis de EEG, empero, tiene las complicaciones de la implementación en Android y el hecho de que el desempeño depende de un tiempo de adquisición más largo. Aunque por otro lado, usa toda la información disponible del amplificador de EEG (los 14 electrodos del EPOC™), hecho que mejora los resultados. El objetivo de este apéndice fue mostrar que existen métodos de procesamiento del EEG más robustos, pero también involucran otros retos.

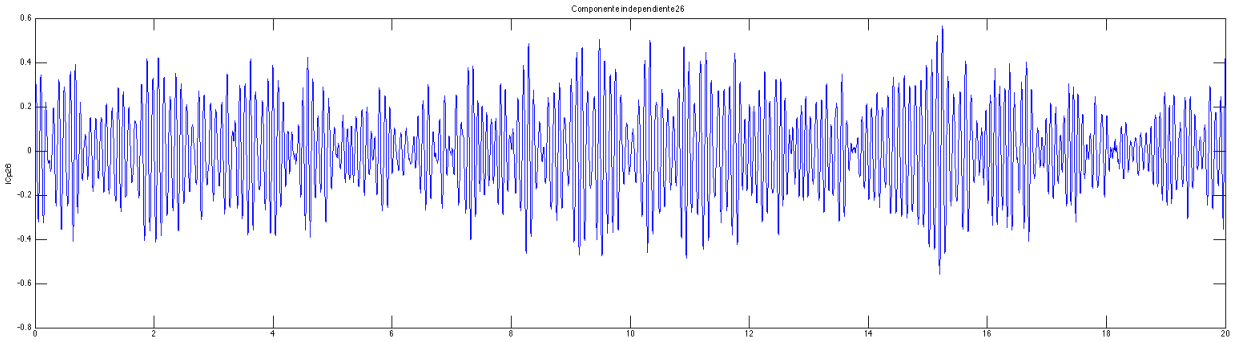


Figura G.1: Componente independiente 26 resultado de ICA.

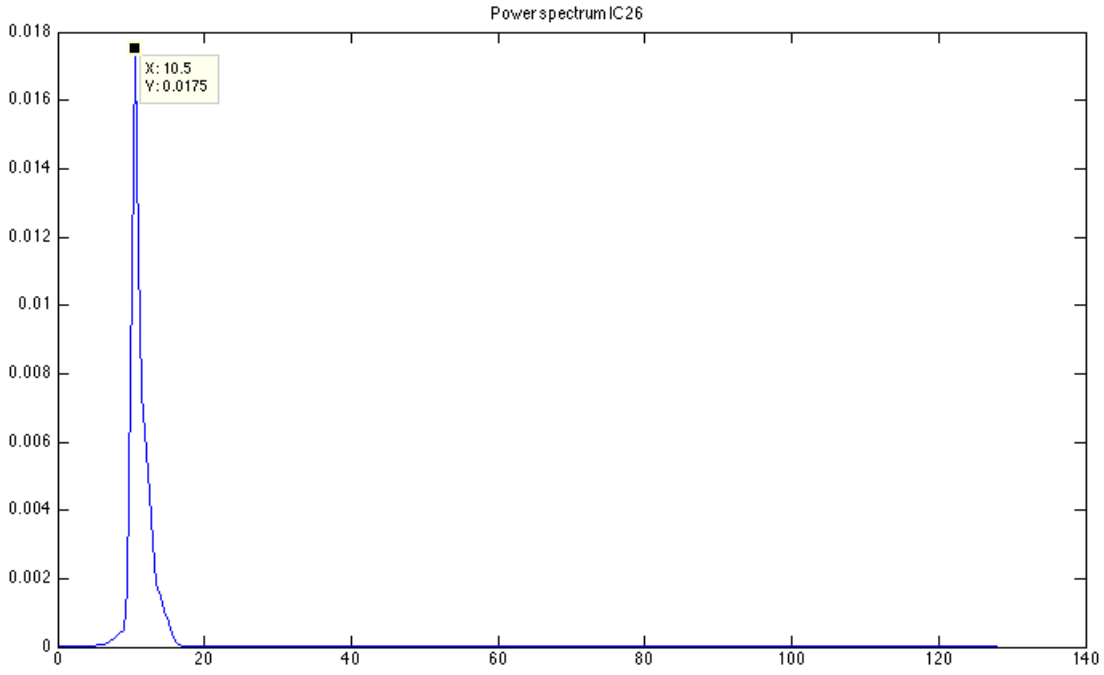


Figura G.2: Potencia del componente independiente 26 resultado de ICA.

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

ACTA DE EXAMEN DE GRADO

No. 00111
Matrícula: 2113800047

Desarrollo de un prototipo inalámbrico de una interfaz cerebro-computadora en un dispositivo móvil

DANIEL MARTINEZ AGUILAR
ALUMNO

REVISÓ

LIC. JULIO CESAR DE LARA ISASSI
DIRECTOR DE SISTEMAS ESCOLARES

En la Ciudad de México, se presentaron a las 12:00 horas del dia 7 del mes de abril del año 2017 en la Unidad Iztapalapa de la Universidad Autónoma Metropolitana, los suscritos miembros del jurado:

DR. JUAN CARLOS ECHEVERRIA ARJONILLA
MTRO. JOSE FRANCISCO RODRIGUEZ ARELLANO
MTRO. OSCAR YAÑEZ SUAREZ

Bajo la Presidencia del primero y con carácter de Secretario el último, se reunieron para proceder al Examen de Grado cuya denominación aparece al margen, para la obtención del grado de:

MAESTRO EN CIENCIAS (INGENIERIA BIOMEDICA)

DE: DANIEL MARTINEZ AGUILAR

y de acuerdo con el articulo 78 fracción III del Reglamento de Estudios Superiores de la Universidad Autónoma Metropolitana, los miembros del jurado resolvieron:

= APROBAR =

Acto continuo, el presidente del jurado comunicó al interesado el resultado de la evaluación y, en caso aprobatorio, le fue tomada la protesta.

DIRECTOR DE LA DIVISIÓN DE CBI

DR. JOSE GILBERTO CORDOBA HERRERA

PRESIDENTE

DR. JUAN CARLOS ECHEVERRIA ARJONILLA

VOCAL

MTRO. JOSE FRANCISCO RODRIGUEZ ARELLANO

SECRETARIO

MTRO. OSCAR YAÑEZ SUAREZ