Facultad de Informática

Grado de Ingeniería Informática

Ingeniería de Computadores

Sistema BCI mediante imaginación motora para control de movimiento en 2D

Aritz Bringas Galarza

Junio 2014

Resumen

Este proyecto trata sobre el control de movimiento en 2D mediante el uso de la imaginación motora, utilizando para ello un sistema con interfaz cerebro-computadora (BCI). En este documento se irá explicando la manera en la que se han utilizado herramientas como BCI2000 y V-Amp 16 a la hora de hacer la adquisición de las señales cerebrales y la interpretación de señales en tiempo real mediante Matlab y Weka.

El proyecto realizado se compone de dos aplicaciones distintas. La primera, se encargará de recoger los datos de la actividad cerebral generada en respuesta a unas indicaciones visuales en la corteza motora. La segunda, realizará una interpretación de los datos que se vayan obteniendo en tiempo real y mostrarlo mediante el movimiento de un cursor en la pantalla.

En este documento se irá explicando la manera en la que se han utilizado y la configuración requerida para el uso de herramientas como BCI2000 y V-Amp 16 a la hora de hacer la captura de las señales cerebrales y el post-tratamiento de la información obtenida de ella mediante Matlab y Weka.

Además, se incluirán dos manuales, una de cara al usuario al que se le realice la prueba y otra de cara al experimentador que quiera realizar alguna prueba con las aplicaciones implementadas.

Palabras clave: BCI, EEG, V-Amp 16, Matlab

Índice de contenido

Índice de imágenes

Índice de tablas

1

1. Introducción

En este capítulo se hace una introducción del contexto en el que se sitúa el proyecto realizado, explicando de manera sencilla cuáles han sido las herramientas que se han utilizado en el desarrollo y cuál es el esquema general del trabajo realizado.

Antes de nada es necesario hacer una breve explicación sobre lo que se desea realizar con este proyecto. Como el título del proyecto indica, se trata de lograr el control de movimiento en 2D (por ejemplo, el cursor del ordenador) usando un sistema BCI (Brain Computer Interface) el cual trabajará con la imaginación motora del usuario en cuestión. La imaginación motora constituye la simulación mental de una acción sin la ejecución de ella mediante un movimiento físico.

BCI es un sistema que permite a una persona controlar dispositivos como el cursor de un ordenador usando solamente su pensamiento. El objetivo es ayudar a personas con necesidades especiales a comunicarse ofreciéndoles una manera en la que no dependan del control muscular, sino que lo hagan mediante el pensamiento. Para construir una BCI hace falta combinar conocimientos en medicina, neurología, psicología, aprendizaje automático, estadística y procesamiento de señales. En los últimos 15 años, ha sido un área de investigación muy activa. Esto se debe a los nuevos conocimientos de funciones cerebrales y señales EEG, disponibilidad de equipos de ordenadores potentes y de bajo coste, y el reconocimiento mayor de las necesidades de personas con problemas neuromusculares.

Los sistemas BCI se basan en la grabación de la actividad cerebral a través de señales EEG y el reconocimiento de patrones asociados a las tareas mentales. Es conocido que dichas tareas como el de imaginar el movimiento de la mano derecha o izquierda, están asociadas con los patrones de actividad EEG en la parte izquierda y derecha de la corteza motora, respectivamente. Estos patrones están asociadas a varios cambios en la actividad EEG. Por ejemplo, se sabe que los valores mu rhythm (8-12 Hz) y beta rhythms (18-26 Hz) decrecen durante el movimiento o preparación del movimiento y crecen después del movimiento. Es posible seleccionar unas tareas pequeñas que activen diferentes partes del cerebro para hacer este reconocimiento más sencillo. Después, utilizar algoritmos clasificadores supervisados para aprender a reconocer

estos patrones de actividad EEG.

Desde el punto de vista de data mining, se trata de una tarea muy desafiante por varias razones. La primera es que los datos EEG son muy ruidosos y cada electrodo posicionado en la superficie del cuero cabelludo mide la actividad de miles de neuronas. Además, la calidad de los datos varía dependiendo del sujeto que lo prueba y de la concentración de éste durante el proceso de grabación, incluyendo de esta manera aún más ruido. Otra razón es que la dimensión de los datos cambia acorde con los canales que se utilizan en el proceso de grabación y se extraen varias características de ellos. De la misma forma, el número de ejemplos de entrenamiento es pequeño, ya que la adquisición de datos etiquetados cuesta mucho tiempo y cognitivamente es un proceso que exige mucho al sujeto.

En este proyecto, a la hora de tratar los datos de BCI se ha hecho una selección de características intentando que el ruido de estos datos afecte de la menor manera posible. De esta forma, se ha conseguido reducir los datos, obteniendo solamente los que resultan más valiosos y quitando las características irrelevantes y redundantes para así poder lograr un resultado bueno con los algoritmos de clasificación. Además, mediante el uso de selección de características se reduce la dimensión de la base de datos, lo cual permite ejecutar los algoritmos de clasificación de forma más rápida.

1.1 Motivación

La comunicación es una parte esencial en cualquier aspecto de la vida. La mayor parte de estas comunicaciones requiere algún movimiento de cualquier tipo, como por ejemplo un gesto facial, lenguaje corporal o la capacidad de mover una mano a la hora de comunicarse de manera escrita. De esta manera, si alguna persona sufre algún tipo de inmovilidad en alguna parte de su cuerpo, perdería alguna manera de poder comunicarse. Esto podría llevar a alguno a no sentirse integrado en la vida cotidiana.

Hoy en día, mucha gente tiene problemas físicos en alguna parte de su cuerpo que les impide utilizar algún dispositivo periférico del ordenador. Es muy importante que ninguna persona sienta que en su condición no pueda hacer ciertas actividades igual que lo haría otro sin su condición. Dado que cada día más personas sufren algún tipo de impedimento, se están realizando cada vez más investigaciones para el desarrollo de tecnologías llamadas *brain-computer interfaces* (BCI). Estos permiten que los individuos se comuniquen mediante la monitorización de la actividad cerebral y sin la necesidad de realizar ningún tipo de movimiento físico, lo cual puede resultar muy interesante para las personas que sufran alguna discapacidad que les impida integrarse en su vida cotidiana. Por ejemplo, controlar el movimiento de una silla de ruedas.

Por esta razón, con este proyecto se pretende implementar una aplicación que ofrecerá un servicio para ayudar a la integración en el día a día de estas personas mediante la interacción con dispositivos que recogen y tratan señales cerebrales.

Como se puede apreciar en la Figura 1.1, la aplicación constará de un sistema que adquiere la actividad cerebral y tras una fase de procesado de la señal ejecutará el comando correspondiente.

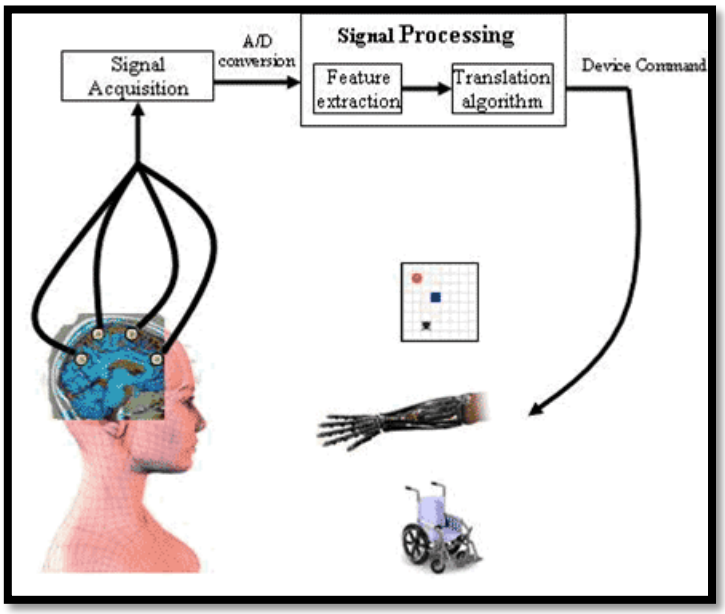

Figura 1.1 Proceso general del proyecto

Para llevar a cabo este experimento, se usará un amplificador llamado V-Amp 16, el cual es capaz de adquirir diferentes señales de una persona como cualquier potencial evocado, ECG (Electrocardiogramas), ECG (Electrooculogramas), EMG (Electromiografía) o EEG (Electroencefalogramas). Estas últimas serán las que se utilizarán en este proyecto.

La electroencefalografía (EEG) consiste en medir la actividad bioeléctrica cerebral. De esta manera, mediante el uso de unos electrodos posicionados en la parte superior de la cabeza se pueden recoger estas señales cerebrales.

Esta memoria seguirá un orden lo largo de la memoria. El Capítulo 2 se trata del Documento de Objetivos del Proyecto (DOP). Dentro de este capítulo primero se explicarán cuáles han sido los objetivos del proyecto, después se mostrará la planificación que se ha llevado a cabo para realizar el proyecto y se contrastará la desviación final. En el Capítulo 3 se explicará detalladamente las herramientas utilizadas para la ejecución del proyecto. El diseño de proyecto realizado y su desarrollo se mostrará en el Capítulo 4. A continuación, en el Capítulo 5, se indicarán las conclusiones extraídas después de haber realizado el proyecto. El Capítulo 6 recogerá toda la bibliografía utilizada para informarse acerca de las herramientas y el uso de éstas. Por último, en el Capítulo 7 se encontrarán dos manuales, el primero de cara al sujeto y el segundo de cara al experimentador que use las aplicaciones realizadas.

2

2. Documento de objetivos del proyecto

2.1 Objetivos

El objetivo de este proyecto es poner en marcha un entorno experimental en el campo de las interfaces cerebro-computador (Brain Computer Interfaces, BCI), consistente en la adquisición de señales de actividad electroencefalográfica correspondiente a la imaginación motora para actuar sobre algún dispositivo mediante el pensamiento. De esta manera puede servir de ayuda a personas que tengan alguna discapacidad que les impida interactuar con su alrededor de forma autónoma (por ejemplo, controlando una silla de ruedas). Por otra parte, cualquier persona no-discapacitada que quiera utilizar este tipo de servicio también podría hacerlo.

A la hora de empezar este proyecto, se planteó la siguiente pregunta: ¿Puede un usuario controlar el movimiento en 2D sin hacer ningún movimiento físico, utilizando sólo los impulsos eléctricos cerebrales?

Para contestar a esta pregunta, se han realizado las siguientes tareas:

- Familiarización con el entorno BCI2000, V-Amp 16 y Matlab.
- Diseñar una secuencia de indicaciones visuales para adquirir las señales eléctricas cerebrales generadas en el sujeto en respuesta a estas.
- Entrenamiento del módulo de comportamiento utilizando la plataforma de aprendizaje Weka.
- Puesta en marcha y pruebas.

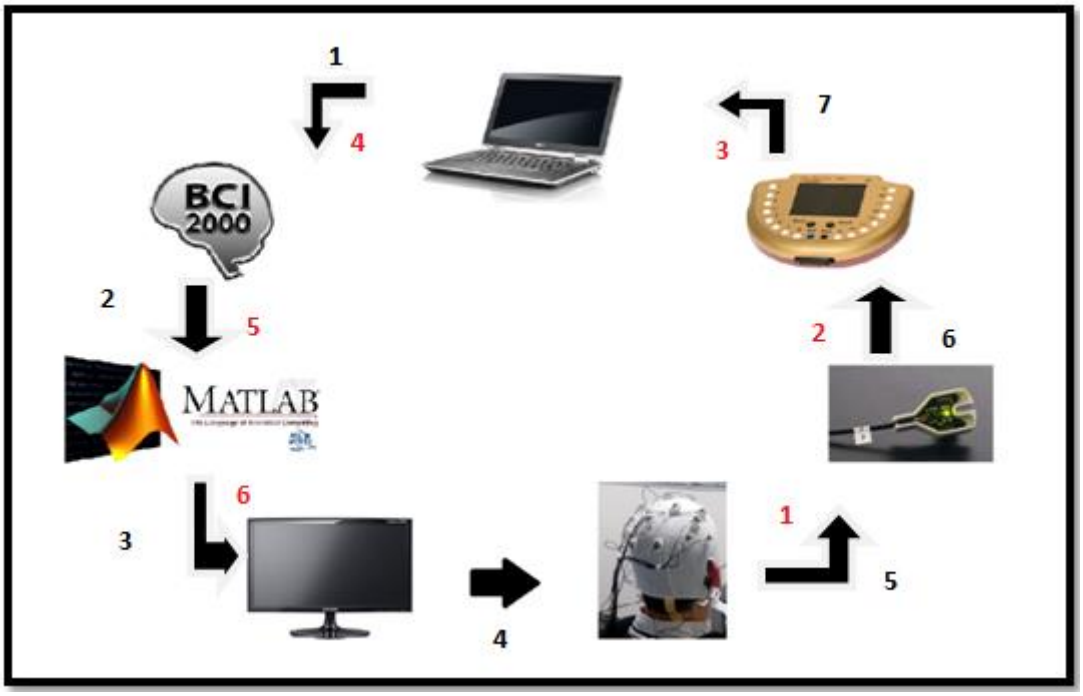

Figura 2.1 Programas utilizados paso a paso en el proyecto

Este proyecto en su totalidad consta de dos aplicaciones que son independientes en su uso. La primera aplicación será el utilizado para adquirir las señales electroencefalográficas de un sujeto en respuesta a unas indicaciones visuales. Los programas utilizados para realizar esta aplicación y el orden que se ha seguido para ello se puede ver en la Figura 2.1 marcado de color negro. La puesta en marcha empieza desde el PC. Mediante un software llamado BCI2000 que sirve para monitorizar las señales cerebrales en conjunto con un programa desarrollado en Matlab se crea una secuencia de eventos que un sujeto tendrá que visualizar. El sujeto contará con una especie de gorro que contiene electrodos que irán captando las señales generadas durante la visualización de la secuencia, llamada actiCAP. Los electrodos irán conectados a un amplificador llamado V-Amp 16 que se encargará de a recoger las señales y mandarlas al PC, en el que se guardarán en un fichero para poder analizarlos en cualquier momento. La segunda aplicación hará una interpretación de las señales cerebrales que le lleguen de entrada en tiempo real. Aunque su ejecución sea independiente al primero, usará el mismo tipo de señal que se genera de salida en la primera aplicación como parámetro de entrada. Por lo tanto, las señales que se adquieran de un sujeto mediante los electrodos del actiCAP y V-Amp 16 se tratarán en el PC mediante el conjunto de BCI2000 y Matlab, haciendo mover un cursor en pantalla en respuesta a las interpretaciones realizadas a las señales. Esto puede verse de color rojo en la Figura 2.1.

2.2 Planificación

2.2.1 Fases del proyecto

En la Figura2.1 podemos ver de manera clara y concisa cuales son las fases que ha tenido el proyecto para poder llevarlo a cabo de manera exitosa.

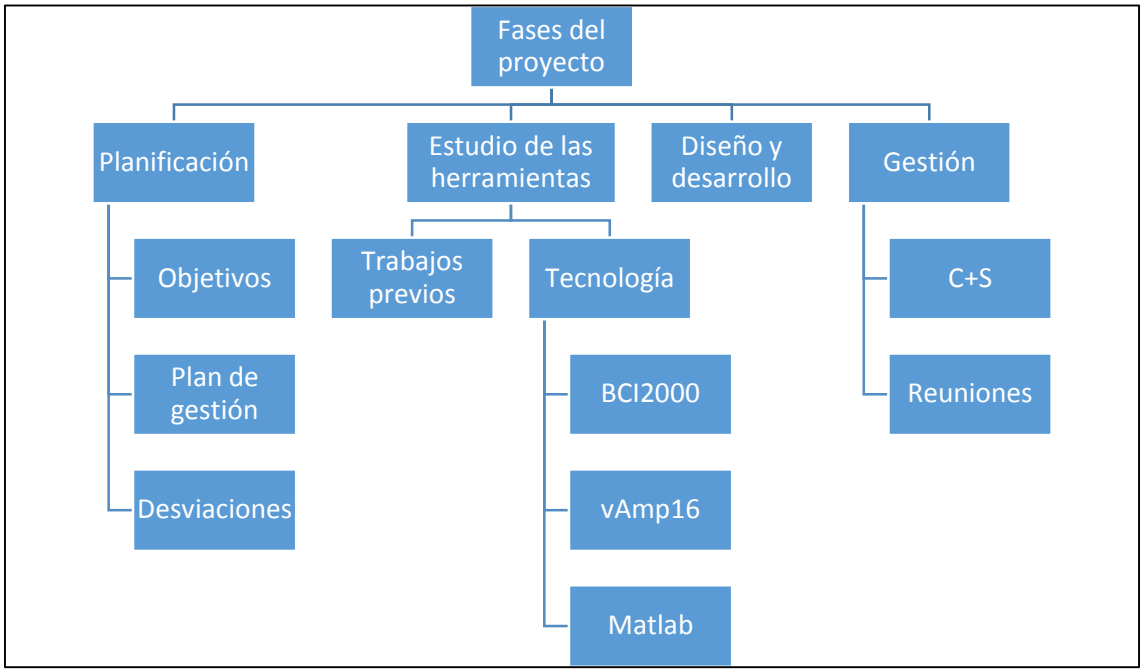

Figura 2.2 Fases del proyecto

Explicación de las fases del proyecto:

- **Planificación.** Se trata de la primera fase, momento en el que se piensa todas las diferentes tareas que hacen falta para llevar a cabo el proyecto. Se definen cuáles serán los objetivos mínimos que debe cumplir el proyecto y cuál será el plan de trabajo a lo largo del proyecto, definiendo la división de horas a lo largo del proyecto. Estas se contrastarán con los tiempos realmente utilizados al final del proyecto.
- **Estudio de las herramientas**: En esta fase se adquieren los conocimientos básicos requeridos para poder llevar a cabo diferentes tareas del proyecto. Esta fase tiene bastante importancia, ya que es necesario saber usar las herramientas antes de empezar a utilizarlos.

  -Trabajos previos: Este proyecto es la continuación de otro proyecto realizado el año anterior por otro alumno. Por lo tanto, para empezar es imprescindible documentarse del avance realizado por el anterior alumno a la hora de empezar y de esta manera entender mejor el funcionamiento de las herramientas que se tendrán que utilizar en el desarrollo de este, como el caso de V-Amp 16.

  -Por lo que respecta a la tecnología, es muy importante saber cómo utilizar las tres principales herramientas que se usarán para llevar a cabo el proyecto. Para ello, conviene leer el manual de BCI2000 para entender el funcionamiento y saber cómo programar.

  -Matlab es un lenguaje muy importante para realizar el tratamiento de señales. No se trata de un leguaje muy complicado y además incluye varias funciones matemáticas y cajas de herramientas implementadas por defecto para tratar las funciones de manera más sencilla.

- **Diseño y desarrollo**: Es la fase más costosa, ya que es la que más tiempo y esfuerzo requiere. Para poder llevar a cabo esta fase, es necesario entender bien las herramientas mencionadas en la fase anterior. En esta fase, como se explicará en el Capítulo 4 de manera más detallada, se ha llevado a cabo la programación en Matlab de los dos módulos (módulo de adquisición de señales de actividad EEG el módulo de interpretación de señales EEG en tiempo real) para BCI2000 que conjuntan este proyecto.
- **Gestión**: Esta fase es para hacer un seguimiento del proyecto. Se les realiza un control y seguimiento (C+S) al proyecto y al desarrollador. Esta fase, ante todo es para llevar el control de los tiempos de cada tarea y ver si se están cumpliendo o se llegarán a cumplir en el plazo indicado. Se realizarán reuniones con los tutores del proyecto en el momento en el que se prevea que es necesario, ya sea para cambiar alguna característica del proyecto o para añadir o quitar una. Además, también se realizarán una memoria que recogerá todo el trabajo realizado y una presentación acorde con lo conseguido en el proyecto.

2.2.2 Estimación de tiempos

Al tratarse de un Proyecto de Fin de Grado, la realización de éste es equivalente a dos asignaturas. Por lo tanto se trata de 12 créditos que en horas son 300 como mínimo. Teniendo en cuenta esto, se ha resumido la estimación de los tiempos en función de las fases del proyecto como se puede ver en la Tabla 2.1.

Tareas	Estimación
Planificación	10 horas
Estudio de las herramientas	25 horas
Diseño	10 horas
Desarrollo	200 horas
Control y seguimiento	10 horas
Documentación	65 horas
Total	320 horas

Tabla 2.1 Gestión del tiempo del proyecto

Como se puede ver en la Tabla 2.1, el desarrollo es la tarea que más tiempo requiere junto con la documentación.

En la Figura 2.2 se puede ver de manera gráfica la estimación de las tareas planificadas mediante el uso de un diagrama de GANTT.

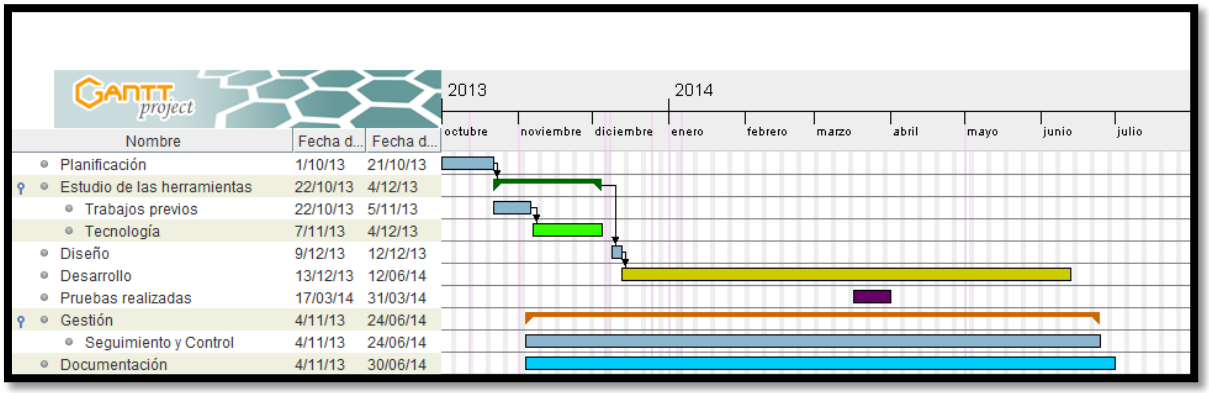

Figura 2.3 Diagrama de GANTT

El inicio del proyecto es en el mes de octubre, aprovechando de esta manera las horas libres del curso para llevar poco a poco el proyecto hacia adelante. Durante el primer mes se realizan un par de reuniones en las que se me asignan unos cuantos documentos para leer acerca del tema escogido en el proyecto. En este periodo, se llevan a cabo las tareas de investigación acerca del tema de trabajo, leyendo el trabajo de un compañero que trabajo en este tema el año anterior y ojeando varios manuales y trabajos similares para ir progresivamente entendiendo mejor cuales son objetivos que se quieren realizar con este proyecto. Después de la puesta al día con lo que previamente se había conseguido realizar, comienza el periodo de planificación. Es en este momento en el que se hace un esquema de los objetivos mínimos que se quieren realizar y escribir el primer borrador de lo que luego será la planificación. Esto lleva un par de semanas en total para planificar todo.

La documentación, se ha iniciado a la vez que la fase de planificación. Sin embargo, esta tiene como límite la tercera semana de Junio, ya que es cuando hay que entregar este documento con todas las correcciones terminadas.

2.2.3 Metodología de trabajo

Para gestionar el proyecto de manera adecuada, es obligatorio establecer la forma de gestión y las condiciones de gestión.

Las horas de trabajo diarias irán cambiando a lo largo del curso. En el primer cuatrimestre al contar con menos horas de trabajo debido a las clases, se ha trabajado utilizando las horas libres. En el segundo cuatrimestre, al tener más horas libres, se dispondrá de más tiempo para dedicar al proyecto. De esta manera, se prevé meter unas 12 horas semanales y así terminar respetando lo especificado en el diagrama de GANTT. Estas 12 horas serán empleadas siempre en los días de entre semana, dejando siempre los fines de semana para descansar.

Por lo que respecta a las reuniones, no se prevé planificar de antemano una reunión, sino que las reuniones se realizarán en el momento que se vea necesario. Como se ha explicado anteriormente, se han realizado reuniones el primer mes, para explicar el punto de vista de los miembros del grupo de tutores y repartir algunos documentos para entender la base de los que se desea realizar. Además, se han hecho reuniones para discutir cuáles serán los objetivos mínimos que se quieren cumplir. En la fase de desarrollo, en el momento en que surja una duda

o se vea que una tarea no se puede hacer de la manera prevista, también se realizarán reuniones, si estos no se pueden solucionar mediante correos electrónicos.

Como el desarrollo del proyecto depende del ordenador del despacho de la Facultad de Informática de la UPV/EHU de San Sebastián, la mayor parte de éste se realizará allí.

2.2.4 Viabilidad

Para que el proyecto sea viable, es necesario cumplir las siguientes condiciones de gestión:

- Establecer una planificación del proyecto y respetarlo.
- Definir los servicios que el proyecto debe ofrecer.
- Escoger y aprender a utilizar las herramientas necesarias para poder llevar a cabo el proyecto, ya sea para la hora de implementarlo o para la documentación.

2.2.5 Análisis de riesgos

A lo largo del proyecto, pueden aparecer varios riesgos. Para evitarlos, conviene primero identificar los riesgos posibles y después realizar un plan de contingencia para hacerles frente. De esta manera, de ocurrir algún problema, se pueden minimizar los daños de estos. Estos son los posibles riesgos previstos:

- Pérdida de datos. Para evitar que se sufra algún tipo de pérdida de datos se ha decidido trabajar siempre en un sistema que trabaja en la nube, Google Drive. De esta manera, siempre se tendrá un fichero en el PC y la copia de ésta en un servidor. Además, Google Drive incluye un editor web con el que se han trabajado los diferentes apartados de este documento.
- Retraso en el desarrollo del proyecto. Es posible que por alguna razón el proyecto sufra un retraso en el desarrollo. Para hacer frente a este tipo de problemas se contemplan tres opciones dependiendo del tamaño del retraso del proyecto:
    - Establecer un margen para terminar las tareas con retraso al final de mayo.
    - Si aun así parece que no hay suficiente tiempo para terminarlo a tiempo, se trabajará también en los fines de semana.
    - Si a pesar de las dos anteriores se ve que no dará tiempo para terminar, se contemplará la opción de entregar el proyecto en septiembre.
- Funcionamiento de los módulos. Como todo el desarrollo del proyecto se ha realizado en el ordenador del despacho de la Facultad de Informática, puede que haya que instalar algún software adicional en el portátil del tutor del proyecto al hacer una demostración de los módulos.

2.2.6 Comunicación

Para hacer frente a los problemas y dudas que van surgiendo a lo largo del desarrollo del proyecto, es obligatorio establecer un plan de comunicación con los tutores del proyecto. En la primera reunión con todo el grupo de tutores del proyecto, se decidió que en caso de necesitar cualquier tipo de ayuda, la principal manera de comunicación sería el del correo electrónico o acudir en horas de tutoría al despacho de cualquiera de ellos. Cada uno de los tutores dispone de su propio horario de trabajo, por lo tanto, no es complicado encontrar a alguno de ellos en el despacho en caso de necesitarlos. En el caso de que todos estén ocupados, se intentará solucionar el problema mediante emails o si no se organizará una reunión cuando sea posible de la misma manera.

2.3 Desviaciones

En este apartado, se estimará el tiempo para el desarrollo del proyecto. La planificación, en general ha ido de acuerdo con lo estimado en un inicio, quitando una tarea hacia el final, la cual ha hecho que haya una pequeña desviación en la última parte. Este desvío no ha generado muchos problemas, ya que había bastante margen previsto por si ocurría algún incidente.

Tareas	Estimación	Desviación
Planificación	10 horas	-
Estudio de las herramientas	25 horas	-
Diseño	10 horas	-
Desarrollo	200 horas	+ 20 horas
Control y seguimiento	10 horas	-
Documentación	65 horas	+ 4 horas
Total	320 horas	24 horas

Tabla 2.2 Desviaciones del proyecto por fases

Hacia el final del proyecto ha habido un cambio en el servidor de licencias de la Universidad seguido por una caída del mismo, la cual ha hecho que durante dos días no se haya podido trabajar en el desarrollo del segundo módulo del proyecto. Consecuentemente, la documentación respectiva a este desarrollo se ha atrasado ligeramente.

2.4 Entregables

Este proyecto tiene los siguientes como entregables:

- Este documento, en el que se explica todo lo que se ha realizado para llevar a cabo el proyecto entero.
- Dos manuales, uno para el sujeto de las aplicaciones desarrolladas y otro para el experimentador, que irán como anexos al final de este documento.
- Código desarrollado para realizar las dos aplicaciones de este proyecto.

3

3. Herramientas utilizadas

3.1 Adquisición de señales

En este capítulo se explicará cada herramienta utilizada en el proyecto en la aplicación que se encargará de realizar la adquisición de señales. Para cada herramienta, se indicarán las características que tienen cada uno y todo lo relacionado con el uso de ellos en este proyecto.

3.1.1 V-Amp 16

Para realizar la adquisición de las señales cerebrales del sujeto se ha utilizado V-Amp 16. Éste es un sistema que sirve para capturar las señales electroencefalográficas y unido con BCI2000 permite guardar estas señales en ficheros que pueden ser tratados de manera offline. Se trata de un producto de Brain Products GmbH en colaboración con Vision LLC.

El sistema se alimenta mediante un conector USB y como el nombre indica trabaja con 16 electrodos. En la Figura 3.1 se puede ver el amplificador que ha sido utilizado en el proyecto.

Figura 3.1 Imagen del V-Amp 16

Como se puede ver en la Figura 3.1, este sistema cuenta con una pantalla TFT, un medidor de

impedancia, un acoplamiento AD/DC controlado mediante software y 16 canales. Cada uno de estos canales lleva conectado un electrodo, que serán los que permitirán recoger las señales deseadas. Estos electrodos irán posicionados en el lugar deseado por el experimentador dependiendo del experimento que desee realizar y en qué zona cerebral se quiera ajustar para que la recepción de las señales sea la más idónea. Pero aparte de estos 16 electrodos también hay uno que será el de tierra (*Ground*) y un último para referencias (*Reference*).

Este hardware es capaz de medir distintas señales cerebrales, entre otras las EEG (Electroencefalogramas), las ECG (Electrooculogramas) y las ECG (Electrocardiogramas).

V-Amp 16 funciona como módulo de BCI2000. Para usar V-Amp 16 junto a BCI, se ha utilizado un módulo llamado vAmpSource.exe, el cual permite la comunicación de ambos. Solamente hay que conectar el dispositivo al ordenador y conectar los electrodos en los agujeros deseados del gorro para empezar con la recogida de señales.

3.1.2 actiCAP

El actiCAP lo conforman unos electrodos que están destinados a aumentar las investigaciones de EEG. Al igual que el amplificador utilizado, el actiCAP se trata de un producto de *Brain Products GmbH*. Como anteriormente se ha mencionado, cada uno de los canales de V-Amp 16 tendrá un electrodo asociado, como el de la Figura 3.2. Al otro extremo de la entrada a V-Amp 16, estos electrodos se posicionarán en unos orificios de un gorro. Dependiendo en la zona en la que se quiere focalizar a la hora de adquirir la actividad cerebral, estos se colocarán en un orificio o en otro. En total actiCAP tiene 32 posiciones en las que se pueden poner estos electrodos. En este experimento, al tener 16 electrodos asociados a sus respectivos canales, se han posicionado todos ellos en los orificios cercanos a donde se encuentra la corteza motora como se puede apreciar en la Figura 3.3.

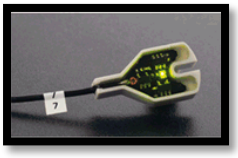

Figura 3.2 Ejemplo de un electrodo de v-Amp 16

Figura 3.3 Electrodos colocados en el actiCAP

En la pantalla TFT de V-Amp 16, se irán mostrando la impedancia en tiempo real de cada uno de los electrodos conectados en el sistema. Para que los resultados obtenidos sean los más exactos posibles y no perjudiquen a la hora de tratarlos, conviene que la impedancia de los electrodos no sea grande. Para ello, este hardware viene equipado con un medidor de impedancia como el que podemos ver en la Figura 3.4.

La caja de impedancias mide la impedancia de los electrodos conectándolos entre ellos directamente. De esta manera, cuando la impedancia sea la correcta se encenderá el LED correspondiente al electrodo en la pantalla de V-Amp 16. Para poder bajar la impedancia de cada uno de los electrodos, se aplica un gel mediante una jeringa en un agujero que tiene cada uno de los electrodos en la parte trasera, como se puede apreciar en la Figura 3.5.

La caja de impedancia no es de un tamaño grande y se alimenta de una batería, lo cual hace que sea muy conveniente por si los experimentos cambian de lugar.

Figura 3.4 Caja de impedancia

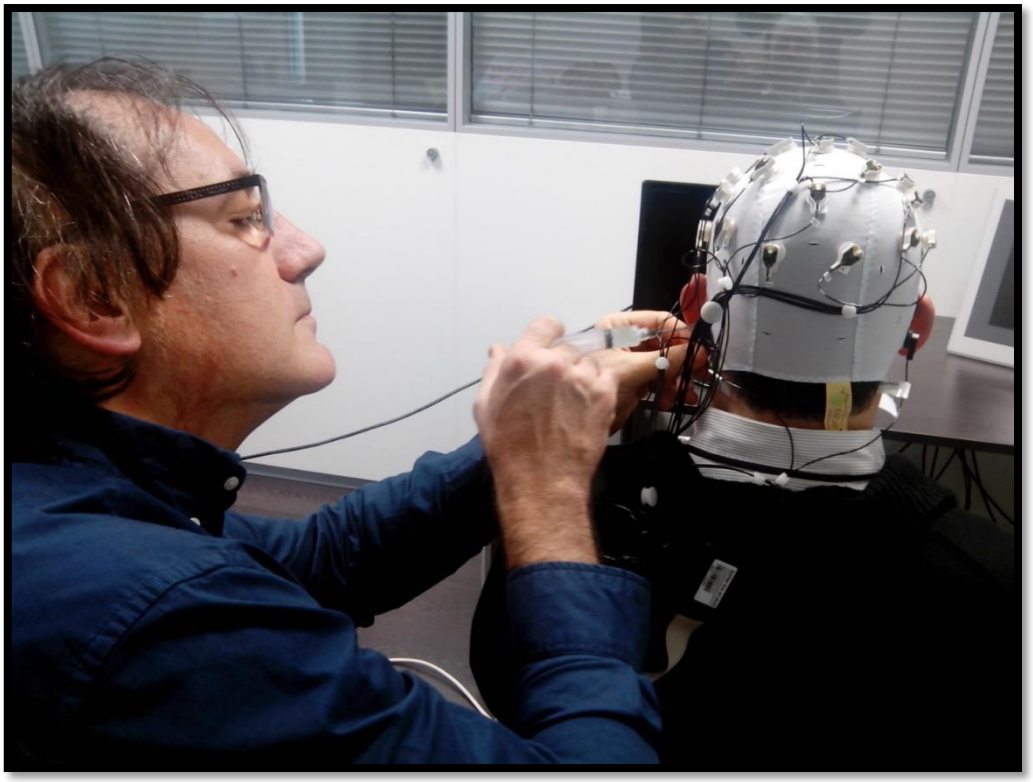

Figura 3.5 Aplicación del gel en los electrodos para bajar la impedancia

Es conocido que dependiendo de un pensamiento se activan diferentes zonas de actividad cerebral, por lo tanto dependiendo de lo que se le plantea hacer pensar al sujeto durante el experimento se podrá monitorizar las zonas de actividad de manera controlada. Las investigaciones previas en esta área permiten saber que al imaginar el movimiento de la mano derecha o izquierda, estos pensamientos están asociados con los patrones de actividad EEG en la parte izquierda y derecha de la corteza motora, respectivamente. Estos patrones varían dependiendo de los cambios en la actividad EEG. Es decir, se sabe que los valores mu rhythm (8-12 Hz) y beta rhythms (18-26 Hz) decrecen durante el movimiento o preparación del movimiento y crecen después del movimiento. Por lo tanto, es posible seleccionar unas tareas pequeñas que activen diferentes partes del cerebro para hacer este reconocimiento más sencillo. En este experimento, se ha centrado en la captura de la actividad EEG posicionando los electrodos en la superficie del cuero cabelludo en la que se encuentra la corteza motora. En la Figura 3.6 se puede ver el lugar cerebral en la que se encuentra la corteza motora.

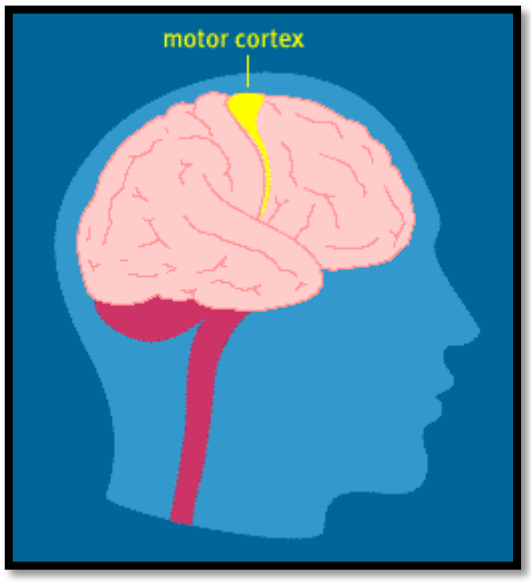

Figura 3.6 Localización cerebral de la corteza motora

La corteza motora es la región de la corteza cerebral responsable de los procesos de planificación, control y ejecución de los movimientos voluntarios. Fijándose solamente en la corteza motora, es posible ver cómo se distinguen las zonas que se activan dependiendo de la simulación mental de la acción del individuo. Para que los datos que se obtienen del experimento sean más fáciles para trabajar y los indicadores de actividad motora se diferencien de manera más sencilla entre sí y el reconocimiento de cada uno sea más simple, se han elegido estas tareas para que se activen partes del cerebro en distintas zonas: la primera tarea hará al sujeto que simule mentalmente la acción de mover la lengua, otra tarea le hará pensar en mover los pies y otras dos pedirán que el sujeto quiera mover el brazo derecho o izquierdo dependiendo de la indicación visual.

Como se puede apreciar en la Figura 3.7, en la parte central de la corteza motora se encuentra la actividad de la imaginación motora de mover el pie, más hacia la parte exterior se encuentra la actividad que indica el deseo de mover el brazo y casi al extremo de la corteza motora se encuentra la actividad relacionada al deseo de mover la lengua.

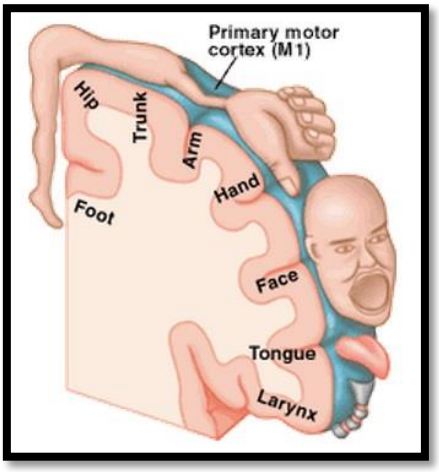

Figura 3.7 Actividad en la corteza motora

Teniendo en cuenta que la actividad que se quiere adquirir se encuentra en la corteza motora, la colocación de los electrodos en el actiCAP será en la superficie del cuero cabelludo superior a la zona en la que se encuentra la corteza motora.

3.2 BCI2000

BCI (*Brain Computer Interface*), se trata de una interfaz que une el cerebro con el ordenador. Esto nos permite monitorizar los impulsos eléctricos generados y obtenidos desde el cerebro y guardarlos en el propio ordenador. Para realizar esta acción, es necesario el uso de V-Amp 16.

Como se ha indicado en el segundo capítulo de este documento, uno de los objetivos de este trabajo ha sido investigar y entender el funcionamiento de BCI2000 y ver las funcionalidades que ofrece, ya que es la base de este proyecto.

El software BCI2000 es uno de los softwares más utilizados en el entorno de investigación de BCI. Esto se debe a que BCI2000 es de código abierto y por lo tanto cualquiera puede ayudar a reparar errores o desarrollar e incluir compatibilidades con otros softwares, como pueden ser sensores o sistemas de adquisición de datos.

El objetivo básico de BCI2000 ha sido querer ser el principal soporte para la investigación de interfaces brain-computer (BCI) desde el inicio. Se creó en el año 2000 en la empresa Schalklab en un proyecto de colaboración y desde entonces sigue desarrollándose.

Este software se puede descargar desde la página http://www.bci2000.org de manera gratuita con el único requisito de registrarse en ella. Los paquetes de ejecutables que se ofrecen están disponibles solamente para Windows, aunque también existe la opción de compilarlo para otro sistema operativo a partir del código fuente.

BCI2000 ha sido la herramienta central del proyecto. Este programa funciona de manera modular, es decir, consta de distintos módulos a la hora de ejecutarse. BCI2000 tiene la capacidad de poder trabajar con otros softwares. A parte de los módulos propios de BCI2000, se ha utilizado un módulo complementario de Matlab para la programación y otro módulo de V-Amp 16 para la adquisición de las señales.

BCI2000 está compuesto por cuatro módulos dependiendo de su funcionalidad: *Source*, *Signal Processing*, *User Application* y *Operator*. Estos cuatro módulos están conectados entre sí como se puede observar en la Figura 3.8. Los tres primeros cogen como nombre módulo *Core* y usan el mismo protocolo de comunicación con el módulo *Operator*. A pesar de que se pueda usar cualquier protocolo de transmisión, se utiliza el protocolo TCP, debido a la fiabilidad de este protocolo. Esta comunicación entre el *Core* y el módulo *Operator* se hace de manera asíncrona y bidireccional. Sin embargo, la comunicación entre los módulos que componen el *Core* es solamente unidireccional.

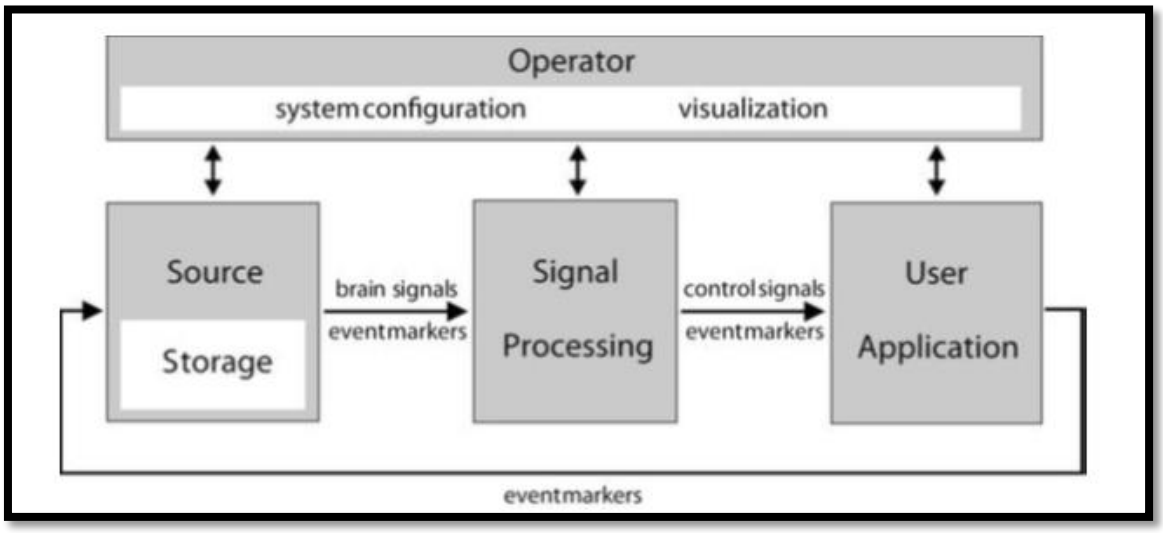

Figura 3.8 Módulos de BCI2000

Cada uno de estos módulos tiene su propio ejecutable para ponerlo en marcha. La puesta en marcha de estos módulos puede hacerse uno a uno mediante los ejecutables respectivos o si no también es posible hacerlo mediante el uso de un *Launcher* (véase Figura 3.9) o interfaz gráfica en la que se pueden elegir manualmente los ejecutables que se quieren lanzar en cada módulo. Una tercera opción sería crear un fichero *.batch* ejecutable (véase Figura 3.10), en la que se definan los ejecutables que se desean lanzar en cada módulo.

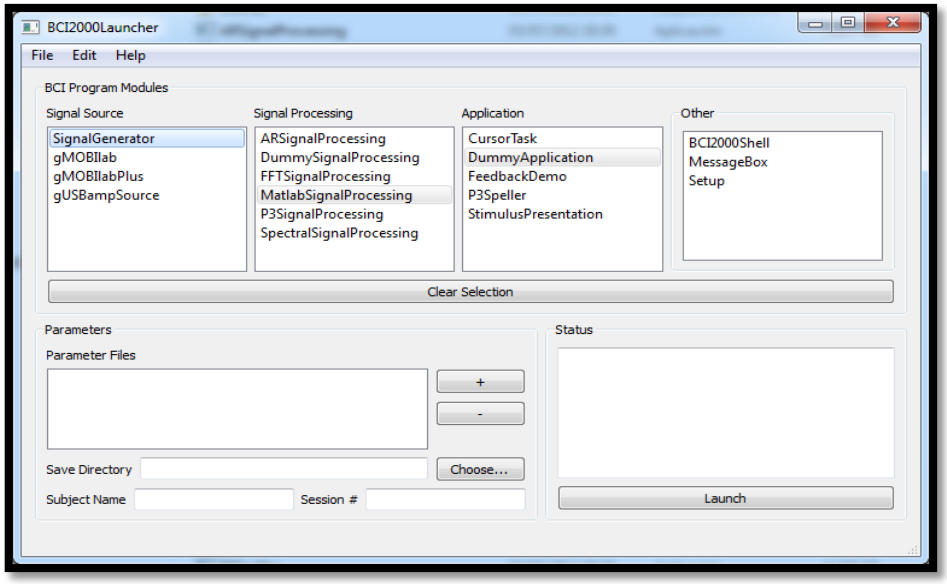

Figura 3.9 Launcher de BCI2000

Como se puede apreciar en la Figura 3.10, utilizando el Shell del propio BCI2000 es posible lanzar los tres módulos que desean mediante la función *Start executable.* En el caso del módulo de Matlab es necesario especificar el directorio en el que se va a trabajar (en este caso *matlabfilter*).

```
#! ../prog/BCI2000Shell
@cls & ..\prog\BCI2000Shell %0 %* #! && exit /b 0 || exit /b 1\n
Change directory $BCI2000LAUNCHDIR
Show window; Set title ${Extract file base $0}
Startup system localhost
Start executable SignalGenerator --local
Start executable MatlabSignalProcessing  --local --MatlabWD=../matlabfilter
Start executable DummyApplication --local
Wait for Connected
#Load parameterfile "../parms/examples/MatlabDemo_SignalGenerator.prm"
```

Figura 3.10 Batch Matlab_SignalGenerator de BCI2000

3.2.1 Módulo Operator

Este módulo consiste de la interfaz que ve el experimentador cuando ejecuta BCI2000 como se puede apreciar en la Figura 3.11. Desde esta interfaz se pueden realizar varias tareas, tales como ver/modificar los valores de los parámetros del sistema, guardar/cargar ficheros de parámetros o iniciar/terminar la ejecución del sistema.

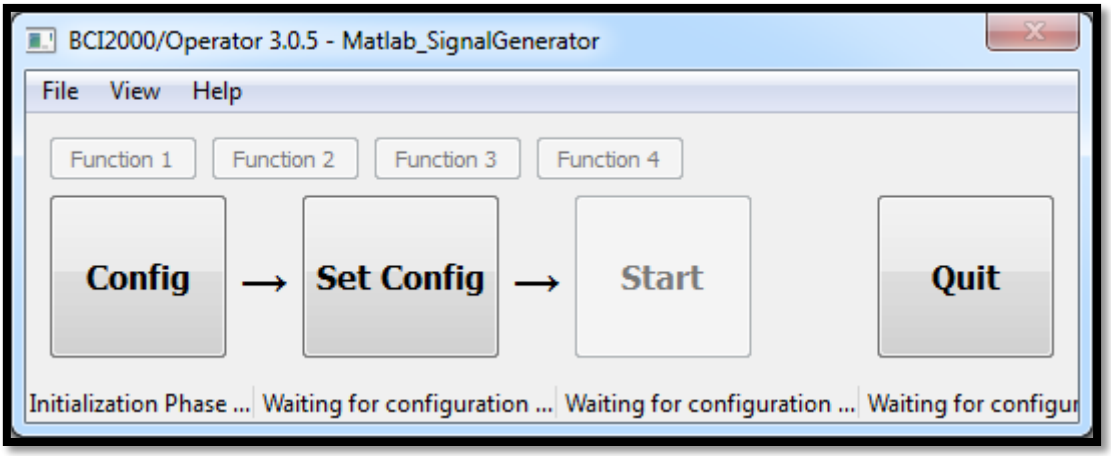

Figura 3.11 Ventana inicial de BCI2000

Como puede observar, en esta ventana principal aparecen cuatro botones: *Config, Set Config, Start* y *Quit*.

- El botón *Config* abre una nueva ventana de configuración. En ella se encuentran todos los parámetros configurables de BCI2000 (véase Figura 3.12).
- El botón *Set Config* realiza las configuraciones elegidas en el apartado anterior (*Config*).
- El botón *Start* sirve para poner en marcha la ejecución. Esta ejecución tendrá los valores previamente elegidos en la configuración.
- Mediante el botón *Quit* se termina la ejecución y se cierra BCI2000.

3.2.1.1 Config

Entre los parámetros de configuración hay varias opciones divididas en distintas pestañas como se pueden apreciar en la Figura 3.12. A continuación se explicarán los parámetros que se han configurado en este proyecto pestaña a pestaña.

En la pestaña *Visualize* se pueden encontrar las opciones dentro de lo que representa la visualización de las señales. A pesar de ofrecer varias opciones, como ayuda para la visualización de las señales se han activado las opciones de visualizar el tiempo (*VisualizeTiming*) y visualización de la señal cerebral (*VisualizeSource*).

La pestaña *System* se ha dejado tal y como venía por defecto. Lo único que hay que tener en cuenta es fijarse que el directorio de trabajo de Matlab (*MatlabWD*) sea el correcto.

Los cambios realizados en la pestaña *Source* han sido muy importantes en el proyecto. El número de canales (*SourceCh*) que se utilizarán en el proyecto son 16, ya que se dispone de un amplificador de señales con 16 canales. El tamaño de bloque o número de muestras que se envían cada vez son 32 (*SampleBlockSize*) y la frecuencia de muestreo (*SamplingRate*) 256Hz. Además, es muy importante especificar cuáles de los canales serán los que transmitan la señal (en este caso todos los canales).

En la pestaña *Storage* se puede configurar el lugar donde se quieren guardar los datos obtenidos, cambiar el nombre del sujeto del experimento, el número de sesión y el número de ejecución del experimento. De esta manera, es más sencillo y práctico ya que se dispone de todos los datos obtenidos de forma ordenada.

En la pestaña *Filtering* se debe quitar el filtro espacial (*SpatialFilterType*) mediante la opción *None*, ya que en este proyecto no se hará uso de esta.

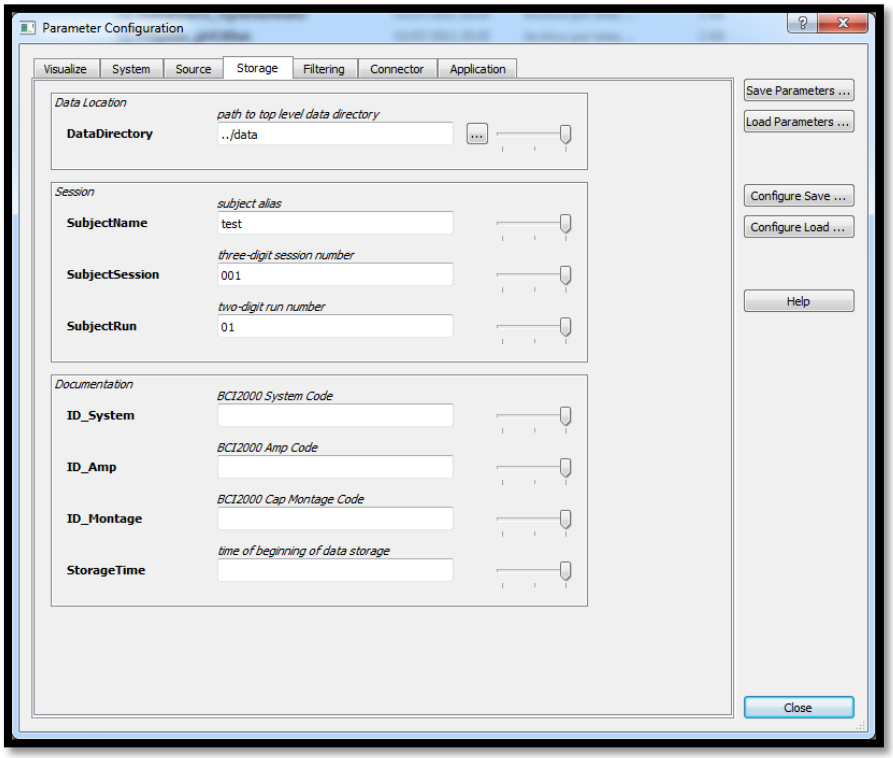

Figura 3.12 Parámetros configurables de BCI2000

Dependiendo del módulo que se ejecute aparecerá otra pestaña al final de éstas. Esa pestaña será la implantada mediante la función *bci_construct* de Matlab.

3.2.1.2 Set Config

Una vez que se hayan elegido los valores que los parámetros del programa, éstos se aplicarán en la aplicación antes de ponerla en marcha. Para ello, solamente hay que pulsar el botón *Set Config*. Al pulsarlo BCI2000 se encargará de validar todos los campos parametrizables y ver si los datos insertados son correctos. Si no es así, dará un error de compilación y mostrará qué parámetro es el que es incorrecto y dará opción de volver a cambiarlo. Cuando los datos insertados en los parámetros de BCI2000 sean los adecuados, el programa realizará los cambios en el programa y lo dejará listo para ejecutarlo.

3.2.1.3 Start

Una vez correctamente configurado, BCI2000 dejará poner en ejecución la aplicación mediante el botón *Start*, que estará desactivado hasta ese momento. Como en cualquier otro programa, si existe algún fallo en el código implementado, devolverá un error y dejará de ejecutarse. Si todo funciona correctamente, a la hora de ejecutarse se abrirá una ventana como ayuda para el usuario para el tratamiento de señales de manera visual. Esta ventana lleva como nombre *Source Signal* y mostrará las señales que se recogen desde los canales de entrada. Además de esta ventana, se abrirá otra llamada *Timing* en la que se irá mostrando el uso de los recursos del CPU en tiempo de ejecución (véase Figura 3.13). En este proyecto al trabajar con una secuencia de imágenes visuales en el primer módulo del proyecto, también aparecerá una tercera ventana a la hora de ejecutarse. Esta se trata de una ventana de Matlab, en la que se irá mostrando la secuencia.

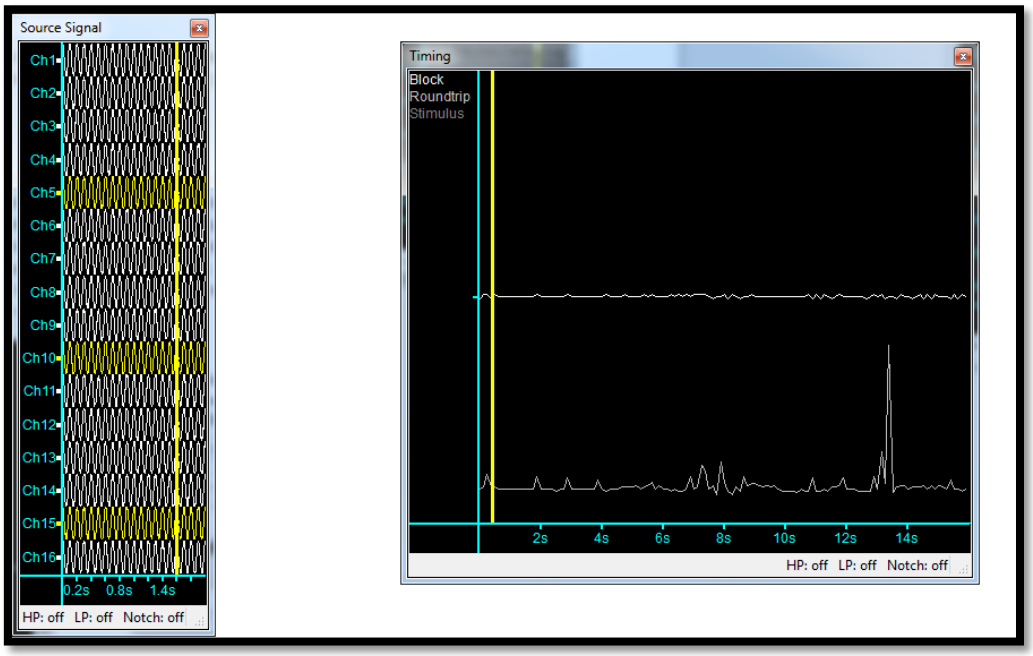

Figura 3.13 Ventanas Source Signal y Timing

Source Signal se puede mover o cambiar de tamaño y siempre se mantiene en la última posición y el mismo tamaño de la última ejecución. Esta ventana ofrece varias opciones en su menú contextual:

- Ajustar la amplitud de la señal.
- Ajustar el tiempo que se muestra en la ventana.
- Elegir los colores de cada una de las señales.
- Elegir los canales de las señales mostradas.
- Aplicar filtros paso bajo, paso alto y paso banda a las señales mostradas.
- Cambiar el modo de visualización de las señales mostradas (*display mode*).
- Activar/Desactivar la elección de mostrar la unidad de cada señal.

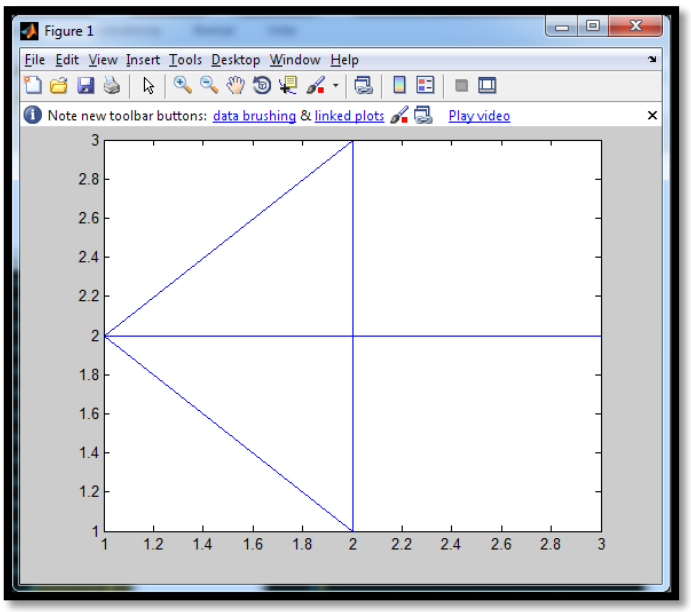

Figura 3.14 Ventana de Matlab

3.2.2 Módulos del Core

El *Core* lo componen estos tres módulos: *Source*, *Signal Processing* y *User Application*. La transmisión entre ellos es como en un bucle cerrado y la velocidad la marca el hardware usado en la adquisición de datos.

3.2.2.1 Signal Source

El módulo *Signal Source* se encarga de esperar los bloques de datos que vienen desde el hardware A/D, para enviárselos al módulo *Signal Processing*. Este envío está sincronizado con el reloj del hardware. Mientras tanto, recoge del módulo *Application* la información del vector de estado y lo guarda en formato .dat a la vez que con los datos digitalizados sin procesar. El funcionamiento de este módulo se puede ver en la Figura 3.15.

```
1: While Running
2:    Save state vector to file
3:    Wait for A/D data
4:    Send A/D data to Signal Processing
5:    Wait for state vector from Application
6:    Save A/D data to file
```

Figura 3.15 Funcionamiento del módulo signal Source

El paso 3 y el 5 son operaciones de bloqueo, el módulo esperará a que lleguen los datos A/D y a que le llegue el vector de estados desde el módulo *Application*.

Este funcionamiento impone una reducción en tiempo real en el procesado de la señal, por lo que necesita un sistema que trabaje de manera rápida. Para ello se debe cumplir lo siguiente:

- Operaciones síncronas de entrada y salida. Se requiere que la duración de un bloque de datos sea menor que el tiempo de ejecución. El tiempo por bloque se definirá mediante la relación de entre la frecuencia de muestreo y el tamaño de
- El tiempo de procesamiento del paso 4 en los módulos *Signal Processing* y *Application* debe ser menor que la duración de un bloque.

3.2.2.2 Signal Processing

Este módulo trabaja como si fuese una caja negra para los demás módulos del sistema, recibe desde *Signal Source* las señales de actividad cerebral y envía señales de control a la aplicación.

La base del módulo la conforman 4 filtros conectados entre ellos en forma de tubo: el filtro espacial, el filtro temporal, el clasificador y el normalizador. La salida de cada filtro va a la entrada del siguiente y trabajan de manera secuencial con el bloque de datos. De esta manera, por cada entrada de bloque de datos hay una salida.

Los canales escogidos en la adquisición de datos van a la entrada del filtro espacial. Después, la señal filtrada con el filtro espacial irá a la entrada del filtro temporal. La señal filtrada temporalmente es enviada a un clasificador y por último se aplica la normalización. Esta última señal de control neutralizada se mandará al módulo de aplicación.

3.2.2.3 User Application

Uno de los objetivos principales del diseño de BCI2000 es la independencia entre módulos. Sin embargo, algunos de los módulos basan su funcionamiento en el feedback del usuario. La dependencia entre los módulos *Signal Processing* y *User Application* hace que el diseño del sistema cree varios problemas, ya que no pueden separase los dos módulos entre sí completamente. Con el objetivo de que se minimicen las dependencias entre estos dos módulos, se utiliza el módulo *User Application*.

3.3 Matlab

Matlab (*Matrix Laboratory*) es la herramienta de software que se ha utilizado a la hora de programar los módulos de captura de BCI2000 y los de tratamiento e interpretación de la señal. Para ello, se ha utiliza el lenguaje propio de Matlab, el lenguaje M. Este lenguaje es de alto nivel e incluye un entorno interactivo para realizar cálculos numéricos, para la visualización y para la programación. De esta forma, es posible crear modelos o aplicaciones y analizar datos. Matlab es muy común en el uso de procesamiento de señales y comunicaciones, es por esta razón que es muy valioso para el desarrollo de este proyecto.

A diferencia de BCI2000, Matlab no es una herramienta gratuita, ya que se debe pagar la licencia para hacer uso de ésta. Además, hay paquetes que tienen como nombre Toolbox que añaden funcionalidades a Matlab, pero que tienen otra licencia distinta. Algunos de estos Toolbox pueden resultar valiosos para el tratamiento de imágenes y sonidos. En este proyecto, la licencia utilizada para la instalación ha sido la que tiene la propia Facultad. Esta licencia incluye un Toolbox llamado *Signal Processing Toolbox*, que ha resultado ser de ayuda a la hora de hacer el filtrado de algunas señales.

La versión de Matlab que se ha utilizado ha sido R2011a 32 bits, ya que esta versión está preparada para hacer la comunicación de manera más sencilla con el software BCI2000. Cualquier otra versión implicaría tener que descargar el código fuente de BCI2000 y tener que compilarlo manualmente para que se pueda conectar con la versión deseada de Matlab.

Como se ha explicado anteriormente, BCI2000 tiene implementado un mecanismo para usar el motor de Matlab con su pipeline. De esta manera, toda la programación realizada con Matlab está hecha para que se pueda ejecutar desde el propio BCI2000 y utilizar su interfaz. Este módulo tiene como nombre *MatlabFilter* y llama al motor de Matlab para que actúe sobre señales, parámetros y estados.

Cuando BCI2000 está ejecutándose, cada bloque de datos se envía al motor de Matlab y se ejecuta la función correspondiente (véase Tabla 3.1). Además, *MatlabFilter* copia la señal de entrada y los estados al motor de Matlab y después de procesar la señal copia la señal de salida y estados del motor.

BCI2000 function	Matlab equivalent
Constructor()	bci_Construct
Preflight()	bci_Preflight
Initialize()	bci_Initialize
StartRun()	bci_StartRun
Process()	bci_Process
StopRun()	bci_StopRun
Resting()	bci_Resting
Halt()	bci_Halt
Destructor()	bci_Destruct

Tabla 3.1 Equivalencias de las funciones de BCI2000 y Matlab

No es necesario que estén cada una de las funciones de la Tabla 3.1. Para determinar si la función dada está disponible Matlab usará el comando '*exists*' y no llamará al motor de Matlab cuando no lo esté. A continuación se explicara las funciones más importantes que se han utilizado a la hora de llevar a cabo la programación del proyecto.

3.3.1 bci_Construct

En este fichero se especifican los parámetros y estados que se quieren parametrizar y de esta manera puedan ser introducidos desde la pestaña deseada en la ventana de configuración de BCI2000. Los valores de los parámetros y estados pueden obtenerse desde el mismo código de Matlab mediante *bci_Parameters* y *bci_States* respectivamente.

3.3.2 bci_Preflight

Esta función comprueba si se cumplen las condiciones requeridas para el funcionamiento. Cada vez que se aplican cambios en los parámetros de BCI2000, se le llama a esta función.

3.3.3 bci_Initialize

Este fichero es el utilizado para inicializar las variables globales que se vayan a utilizar. Esto es importante porque las comunicaciones entre todas estas funciones se hacen mediante estas variables globales.

3.3.4 bci_Process

La función más importante de todas es la función *bci_Process*, ya que se ejecuta cada vez que llega un bloque de datos nuevo. Cada vez que entra en este fichero el programa, se procesa un bloque de la señal recibida. En este fichero se encuentra el código que se quiere ejecutar.

3.3.5 bci_StartRun

El código que se encuentre en este fichero se ejecutará cada vez que se indique desde BCI2000 que se ponga en marcha una ejecución (*run*).

3.3.6 bci_StopRun

El código respectivo a este fichero se ejecutará solamente cuando termine de ejecutarse un *run*.

3.4 Weka

Es un software libre de licencia GNU-GPL que es usado para minería de datos y aprendizaje automático. Desarrollado en la Universidad de Waikato y escrito en Java.

Weka tiene una interfaz gráfica de usuario sencilla, la cual permite acceder a sus funcionalidades de manera fácil. Además incluye varias herramientas de visualización y algoritmos para el análisis de datos y modelado predictivo.

Entre sus puntos fuertes destaca que:

- Está disponible libremente bajo la licencia pública general de GNU.
- Es muy portable porque está completamente implementado en Java y puede correr en casi cualquier plataforma.
- Contiene una extensa colección de técnicas para pre-procesamiento de datos y modelado.
- Es fácil de utilizar por un principiante gracias a su interfaz gráfica de usuario.

En el proyecto, se ha utilizado Weka gestionar el apartado de aprendizaje automático. Por una parte, se ha entrenado un clasificador SVM (*Support Vector Machine*) utilizando una base de datos de entrenamiento para poder aprender un modelo de comportamiento en función de las 5 clases del sistema: movimiento de la lengua, movimiento de los pies, movimiento de la mano derecha, movimiento de la mano izquierda y descanso. Posteriormente, se ha utilizado este clasificador para poder reconocer las intenciones del sujeto que ha realizado el experimento.

4

4. Diseño y desarrollo

Este proyecto se divide en dos aplicaciones independientes entre sí. En la primera, se recoge la actividad cerebral en respuesta a unas indicaciones visuales. Estas indicaciones visuales están programadas en Matlab para que se pueda ejecutar junto con la adquisición de la actividad EEG de V-Amp 16. BCI2000 tiene implementado un mecanismo para usar el motor de Matlab con su pipeline. De esta manera, llama al motor de Matlab para que actúe según las señales, parámetros y estados. En la segunda aplicación, se interpretarán en tiempo real los impulsos EEG de un sujeto mediante el uso de técnicas de aprendizaje automático. Los dos módulos son independientes entre sí a la hora de ejecutarlos, pero existe una conexión entre ambas, ya que el segundo módulo depende de los datos adquiridos en el primero.

De esta manera, mediante el desarrollo de estas dos aplicaciones se quiere ofrecer una ayuda a personas que sufren algún tipo de discapacidad, dándoles la oportunidad de actuar sobre un dispositivo de forma autónoma.

Los dos módulos pueden ser monitorizados mediante el experimentador utilizando la interfaz gráfica, que servirá para cambiar los parámetros de entrada de las dos aplicaciones y para ver los datos en tiempo real de los canales del sistema.

4.1 Adquisición de señales cerebrales

Como anteriormente se ha explicado, el primer módulo se utilizará para recoger la actividad cerebral de la corteza motora del sujeto. Para ello, primero se ha tenido que programar una secuencia de indicaciones visuales a las que el sujeto tendrá que responder a la hora de realizar el experimento. Estas indicaciones las conformarán cuatro flechas en distintas direcciones en dos coordenadas (arriba, abajo, izquierda y derecha) y un símbolo de suma (‘+’). Cada una de las indicaciones hará que el sujeto deba utilizar su imaginación motora para distintas acciones. Después de la programación de la secuencia se llevará a cabo la adquisición de la actividad

cerebral acorde con lo imaginado por el sujeto. En la Figura 4.1 se pueden ver cuáles son los programas que se han utilizado y cuándo se ejecuta cada uno de ellos.

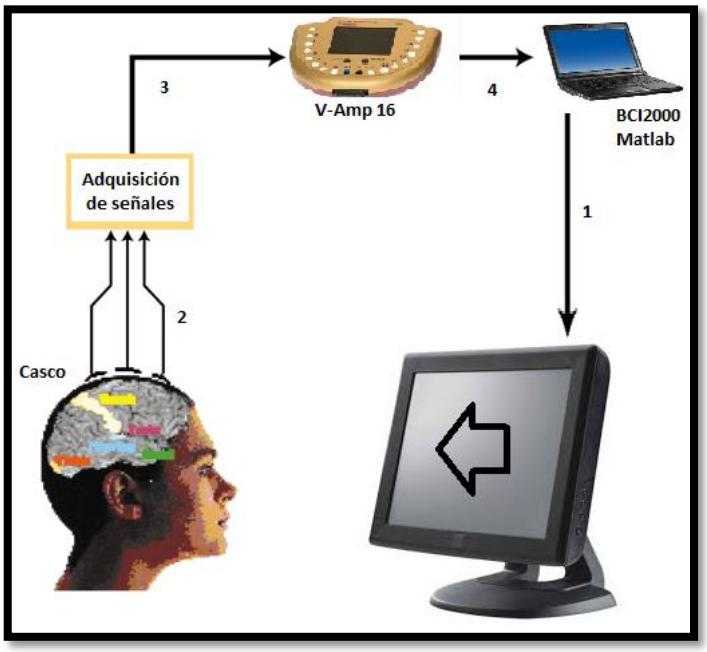

Figura 4.1 Programas utilizados en la adquisición de datos

Como se puede apreciar en la Figura 4.1 este primer módulo conforma un ciclo que empieza y termina en el PC que se utilice. Primero se pone en marcha el software BCI2000 y Matlab en el PC. Cuando estén cargados, se realizarán las configuraciones deseadas en los parámetros de BCI2000 desde su ventana de configuración. Tras ello, se aplicarán los cambios en el programa y se podrá poner la aplicación en marcha. En un monitor externo, se mostrará una secuencia de indicaciones visuales al sujeto del experimento, quién llevará puesto el actiCAP con los respectivos 16 electrodos en la zona superficial cercana a la corteza motora. Estos electrodos irán conectados al amplificador que se utilizará en el proyecto, V-Amp 16, que se encargará de recoger la actividad correspondiente a la zona en la que se encuentren los electrodos y llevarlo al PC. Una vez que llegue al PC, BCI2000 se encargará de guardar la señal recibida cuando la ejecución de la aplicación haya terminado de ejecutarse. BCI2000 guardará en un fichero con formato .dat una matriz con los valores recogidos de las señales electroencefalográficas de manera automática. Esta matriz será de 16 filas (una por cada canal de V-Amp 16) x número de muestras recogidas en el experimento.

Para poder llevar a cabo la adquisición de datos hacen falta las siguientes herramientas:

- Pantalla: Se utilizará un monitor para mostrar unas indicaciones visuales. Las indicaciones visuales de este proyecto serán cuatro flechas y un símbolo de suma, las cuales harán que el sujeto del experimento tenga que pensar en el pensamiento de la acción asociada a dicha indicación. Las flechas aparecerán en cuatro direcciones distintas. La flecha hacia arriba indica que el sujeto del experimento debe de pensar en mover la lengua sin llevar acabo la acción física. La flecha hacia abajo hará que el sujeto piense en mover las piernas. Las flechas hacia la derecha y hacia la izquierda indicarán que el sujeto debe pensar en mover el brazo derecho o izquierdo respectivamente. El

símbolo de suma se utilizará como un aviso para el sujeto de que una de las cuatro flechas hará aparición en breve.

- actiCAP: Como se ha explicado en el segundo capítulo de este documento, el actiCAP tendrá 16 electrodos que servirán para recoger la actividad que se genera a lo largo de la ejecución del experimento.
- Amplificador V-Amp 16: Los 16 electrodos del actiCAP estarán conectados a V-Amp 16. Estos se encargarán de recoger la actividad cerebral y llevarlo al PC.
- Computador: El PC se encargará de poner en marcha el experimento, enviando a la pantalla externa las indicaciones visuales. Mientras que el experimento esté en marcha, las señales llegarán al PC gracias a V-Amp 16. En este punto mediante el uso de BCI2000 existe la posibilidad de guardar las señales obtenidas en ficheros externos. De esta forma, habrá una opción de tratar los datos de manera offline.

La duración del experimento dependerá de los parámetros de entrada configurables por el experimentador. El experimentador tendrá la opción de controlar la duración de todas las partes que compone la secuencia de indicaciones visuales, como cuánto tiempo se desea mantener la imagen de la flecha o el de la suma. Además, también dispondrá de la elección del número de secuencias que se desean realizar en el experimento.

Para que el uso de este módulo sea posible en distintos experimentos, los tiempos en los que aparecerá cada indicación es parametrizable. De esta forma, se puede cambiar al gusto del experimentador cualquier parte que componga la secuencia.

La secuencia que se va a mostrar a lo largo del experimento se calculará al inicio de la ejecución y se guardará en una matriz. Las flechas de las indicaciones visuales aparecerán en la misma cantidad a lo largo del experimento y de manera aleatoria, es decir, la secuencia de las indicaciones visuales irá cambiando en cada ejecución. Esto se debe a que no se desea que el sujeto sepa en ningún momento cuál será la imagen que aparecerá a continuación para que no influya en sus acciones imaginativas. Por lo que en cada iteración se mostrará la figura correspondiente a la matriz en la posición del momento. El símbolo de suma servirá como aviso de que a continuación aparecerá una de las cuatro flechas para el sujeto que esté realizando el experimento. En la Figura 4.2 se puede ver la forma que seguirá la secuencia.

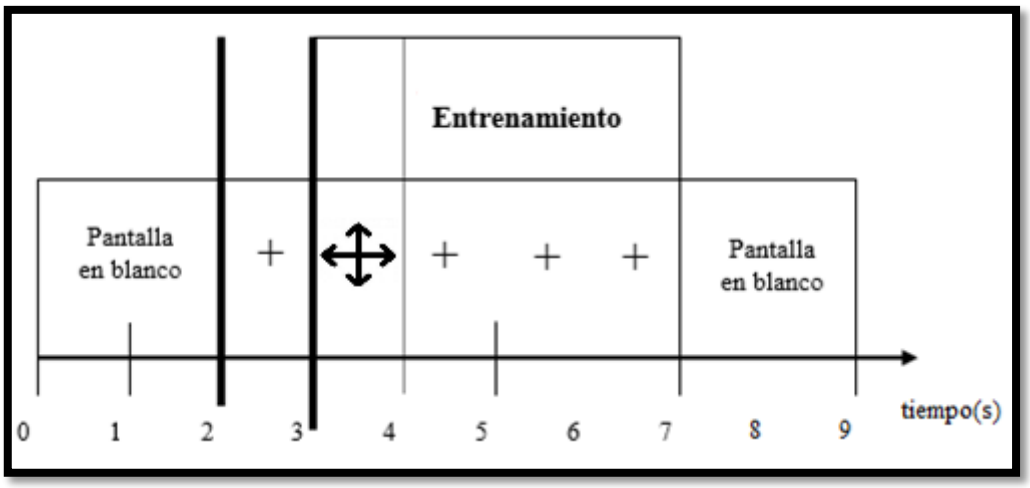

Figura 4.2 Secuencia de indicaciones visuales

4.1.1 BCI2000

BCI2000 ha sido la herramienta principal del proyecto, ya que todas las demás herramientas utilizadas para llevarlo a cabo están preparadas para trabajar con ésta. BCI2000 consta de una interfaz gráfica que lo hace sencillo de usar. Nada más inicializarlo, se carga un módulo llamado *Operator* como se puede ver en la Figura 4.3. Esta interfaz ofrece las opciones de abrir una ventana de configuración mediante el botón *Config*, aplicar la configuración mediante *Set Config* y cerrar la aplicación mediante *Quit*. Se puede apreciar que hay una cuarta opción llamada *Start*, pero ésta permanecerá bloqueada hasta haber realizado la configuración. Una vez que se realicen las configuraciones, se podrá poner en marcha la aplicación mediante el botón *Start*.

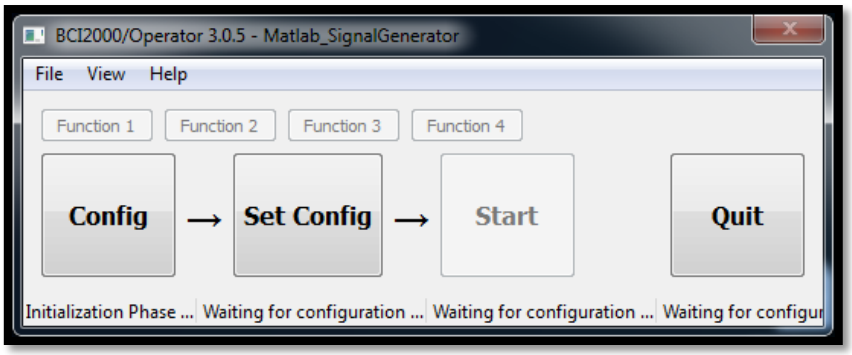

Figura 4.3 Botones Config y Set Config de BCI2000

A continuación se explicará la configuración que se ha utilizado cuando se ha ejecutado esta aplicación. Para configurar los parámetros de la aplicación basta con pulsar el botón *Config* que se ha mencionado anteriormente. Al pulsarlo, se abrirá la ventana de configuración de BCI2000. Los parámetros de BCI2000 están distribuidos en esta ventana en distintas pestañas. La mayoría de ellas se han mantenido tal y como venían por defecto. Para cada ejecución se recomienda cambiar el nombre del sujeto, el número de la sesión y el de la ejecución para que los resultados finales estén ordenados. Esto se hace a través de la pestaña *Storage*, como se puede observar en la Figura 4.4.

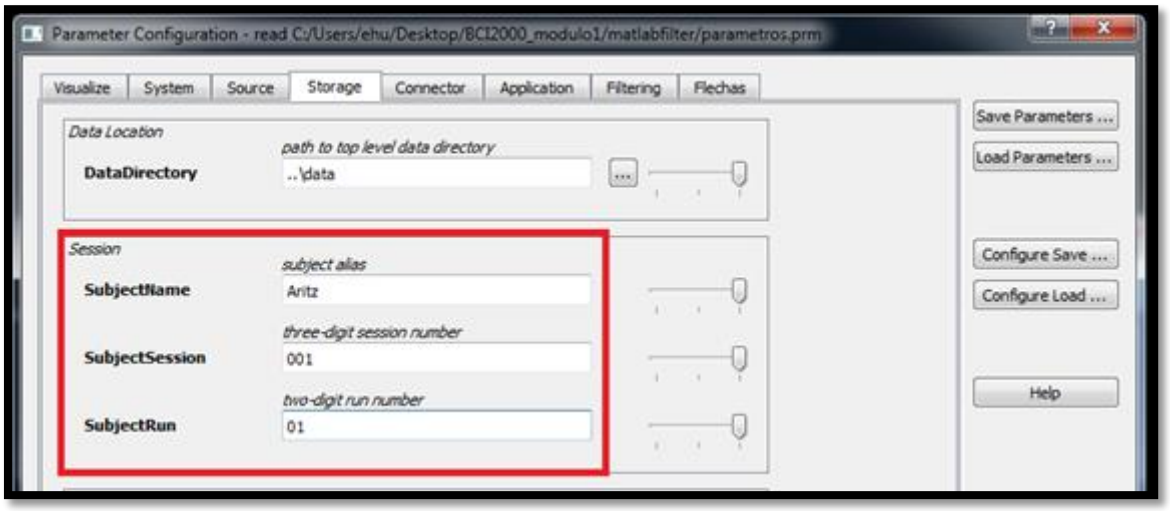

Figura 4.4 Configuración del sujeto que realiza el experimento

La adquisición de datos se realiza entre la combinación de BCI2000 y Matlab. BCI2000 recoge las señales electroencefalográficas que le llegan desde V-Amp 16 y la agrupa hasta que haya una cantidad de capturas (bloques) y las envía a Matlab para que ésta las procese. El número de capturas que es enviado es configurable desde la interfaz de BCI2000 como se puede ver en la Figura 4.5.

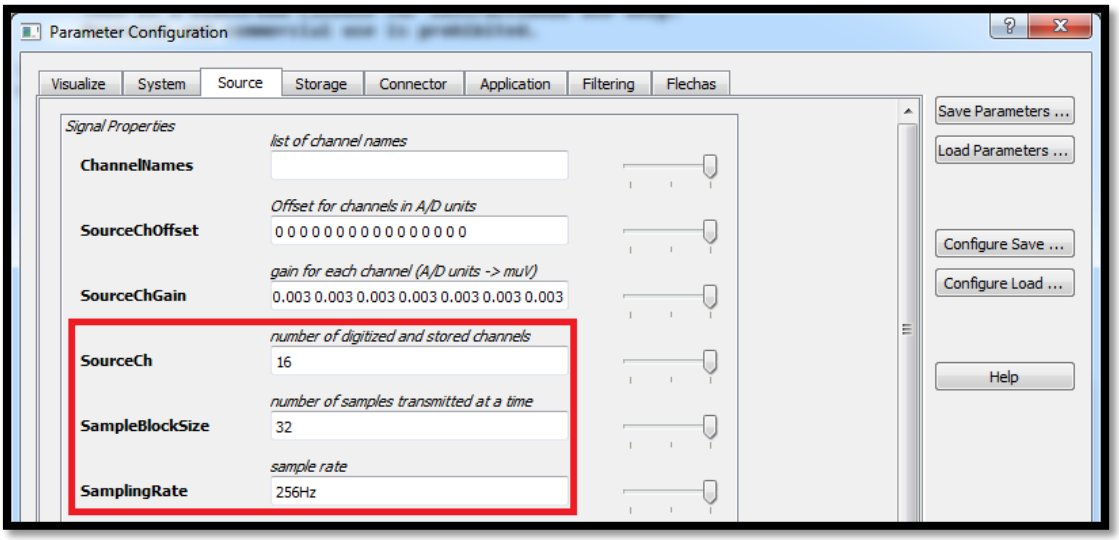

Figura 4.5 Configuración de la fuente de las señales

SourceCh es el parámetro correspondiente a los canales que se quieren utilizar en la aplicación. En este proyecto al disponer del amplificador V-Amp 16 que recoge las señales desde 16 canales, se han utilizado todas ellas en la adquisición.

SampleBlockSize indica el número de muestras que se transmiten a la vez y *SamplingRate* corresponde a la frecuencia de muestreo de la aplicación. Teniendo en cuenta el tamaño de del número de muestras que se transmite cada vez y dividiéndolo con la frecuencia de muestreo de cada muestra se puede conseguir el tiempo que transcurre para ejecutar un boque, que ha servido para controlar los tiempos de ejecución en el proceso.

El uso de Matlab junto con BCI2000 ofrece la posibilidad de añadir una nueva pestaña en la ventana de configuración de BCI2000 para definir los valores de los parámetros de entrada que se vayan a necesitar en el programa principal (véase Figura 4.6). Esto se puede realizar desde el código de Matlab como se explicará en el Capítulo 5.1.2. Como se puede ver en la Figura 4.6, los parámetros corresponden a los tiempos de visualización de la secuencia de indicaciones visuales. Por lo tanto, la duración del experimento dependerá de los valores escogidos. Los valores introducidos corresponderán al tiempo en segundos en los que se mostrará cada una de las indicaciones que forman la secuencia.

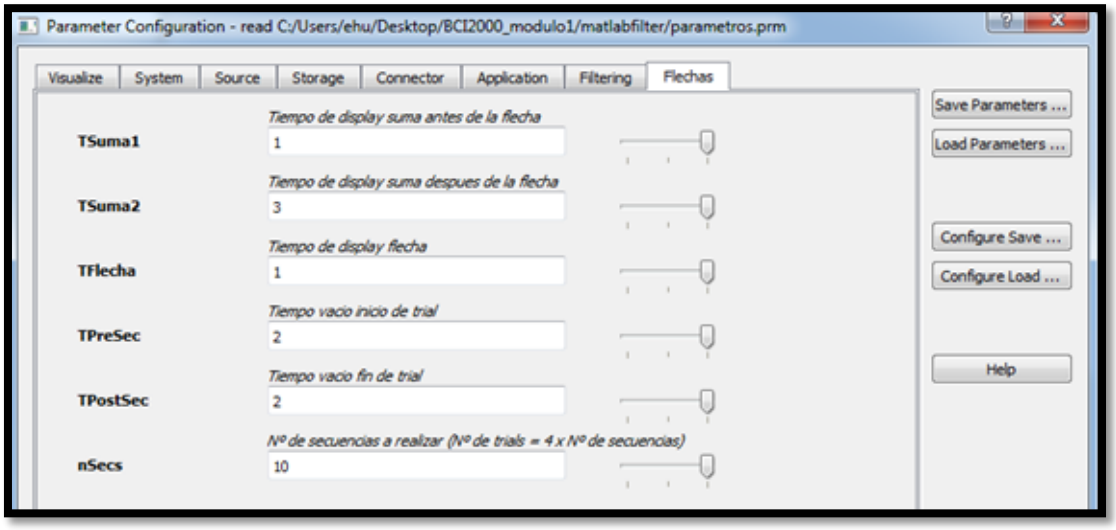

Figura 4.6 Pestaña añadida desde el código de Matlab a BCI2000

4.1.2 Matlab

Aunque la herramienta central del proyecto haya sido BCI2000, para la programación de las dos aplicaciones que componen el proyecto se ha utilizado Matlab. Como anteriormente se ha explicado en el Capítulo 3, se ha utilizado un módulo llamado *matlabfilter* conectado con BCI2000. De esta manera, el software de BCI2000 y Matlab pueden trabajar conectados entre sí. Como se ha explicado con anterioridad, BCI2000 y Matlab tienen una serie de funciones equivalentes (véase Tabla 4.1). Cuando la ejecución de BCI2000 esté en marcha, se envía cada bloque de datos al motor de Matlab y éste ejecuta la función equivalente.

BCI2000 function	Matlab equivalent
Constructor()	bci_Construct
Preflight()	bci_Preflight
Initialize()	bci_Initialize
StartRun()	bci_StartRun
Process()	bci_Process
StopRun()	bci_StopRun
Resting()	bci_Resting
Halt()	bci_Halt
Destructor()	bci_Destruct

Tabla 4.1 Tabla de funciones equivalentes

Mediante el uso de estas funciones equivalentes se ha llevado a cabo la programación entera de la aplicación. A continuación se explicarán las funciones importantes de cada una de estas funciones de *matlabfilter*.

4.1.2.1 bci_Construct

En la función *bci_Construct* pueden definirse los parámetros de entrada y estados que después se podrán configurar desde la ventana de configuración de BCI2000. De esta manera, como se puede ver en la Figura 4.7 se han definido los parámetros correspondientes al tiempo de visualización de cada parte de la secuencia y el número de secuencias que se quieren realizar. Además, también se ha definido *Flechas* que constará de 5 estados que se utilizarán en *bci_Process*.

```
parameters = { ...
  [ 'Flechas float TSuma1= 1
  [ 'Flechas float TSuma2= 3
  [ 'Flechas float TFlecha= 1
  [ 'Flechas float TPreSec= 1
  [ 'Flechas float TPostSec= 1
  [ 'Flechas int nSecs= 10
};

states = { ...
  'Flechas 5 0 0 0' ...
};
```

Figura 4.7 Función bci_Construct

La columna de números correspondiente a los parámetros indica el valor que tendrán por defecto éstos parámetros cuando se inicie BCI2000. A continuación se explica la función de cada parámetro:

- *TPreSec*: en los dos primeros segundos la pantalla aparecerá vacía.
- *Tsuma1*: durante este segundo aparecerá en pantalla el símbolo de suma. Esto servirá para avisar al sujeto que pronto aparecerá una de las flechas.
- *Tflecha*: en este segundo será cuando una de las cuatro flechas aparezca. En cada iteración aparecerá una de las cuatro. Desde este momento el sujeto deberá pensar en la acción de pensar correspondiente a la flecha.
- *Tsuma2*: a pesar de que la flecha para este momento se quite, el sujeto deberá de seguir pensando en la acción correspondiente hasta que esta desaparezca. Esta indicación durará tres segundos, que sumados al anterior hacen cuatro como el tiempo total en el que el sujeto debe esforzarse en pensar en la acción a realizar.
- *TpostSec*: en este momento la indicación anterior desaparecerá y quedará la pantalla en blanco. Este momento será el que el sujeto tendrá como descanso entre secuencias.

En este proyecto se ha utilizado una secuencia que tiene definidos los tiempos de cada uno de los parámetros que conforman la secuencia de manera estricta. Desde un inicio se pensó en la distribución que llevaría la secuencia para saber en qué momentos fijarse cuando se adquiriesen las señales. Estos son los valores que tendrán los parámetros en esta aplicación y el orden que seguirán a la hora de crear la secuencia de indicaciones visuales:

TpreSec (2s)	*Tsuma1 (1s)*	*Tflecha (1s)*	*Tsuma2 (3s)*	*TpostSec (2s)*

Esta secuencia irá repitiéndose durante toda la ejecución de la aplicación. La duración de todo el experimento dependerá de los parámetros indicados en BCI2000.

4.1.2.2 bci_Initialize

La función *bci_Initialize* irá utilizando variables globales a lo largo de la aplicación (véase Figura 4.8).

```
global bci_Parameters bci_States;

% Variables globales
global nSecs;
global numTrial;
global numBloq;
global nBSuma1;
global nBFlecha;
global nBSuma2;
global nBPost;
global nBFin;
global x1;
global x2;
global x3;
global x4;
global x5;
global x6;
global v2;
```

Figura 4.8 Variables globales

La secuencia de indicaciones visuales tiene sus propios tiempos de visualización como se ha podido observar antes. Para controlar el tiempo dentro del programa, se han calculado el tiempo por bloque mediante la relación del tamaño del bloque y la frecuencia de muestreo (véase Figura 4.9).

```
tamBloq = str2double(bci_Parameters.SampleBlockSize); %tamaño de bloque
fs = str2double(strrep(bci_Parameters.SamplingRate,'Hz','')); %frecuencia de muestreo
tiempoBloq = tamBloq/fs;
```

Figura 4.9 Tiempo que transcurre por bloque

Después de saber el tiempo por bloque, se ha calculado el número de bloques para cada uno de los parámetros introducidos desde la interfaz de BCI2000 correspondientes a la función *bci_Construct* (véase Figura 4.10).

```
nBSuma1 = round(str2double(bci_Parameters.TPreSec)/tiempoBloq);
nBFlecha = round(str2double(bci_Parameters.TSuma1)/tiempoBloq) + nBSuma1;
nBSuma2 = round(str2double(bci_Parameters.TFlecha)/tiempoBloq) + nBFlecha;
nBPost = round(str2double(bci_Parameters.TSuma2)/tiempoBloq) + nBSuma2;
nBFin = round(str2double(bci_Parameters.TPostSec)/tiempoBloq) + nBPost;

nSecs = str2double(bci_Parameters.nSecs); %cuantas secuencias (de 4 trials) se quieren realizar
```

Figura 4.10 Obtención de los parámetros introducidos desde la interfaz de BCI2000

En esta función también se han definido las propiedades de las indicaciones visuales que se mostrarán a la hora de ejecutarlo. Las indicaciones visuales de las flechas y el símbolo de suma en conjunto forma una sola imagen. En la Figura 4.11 se pueden ver todas las propiedades de la figura utilizada en la secuencia.

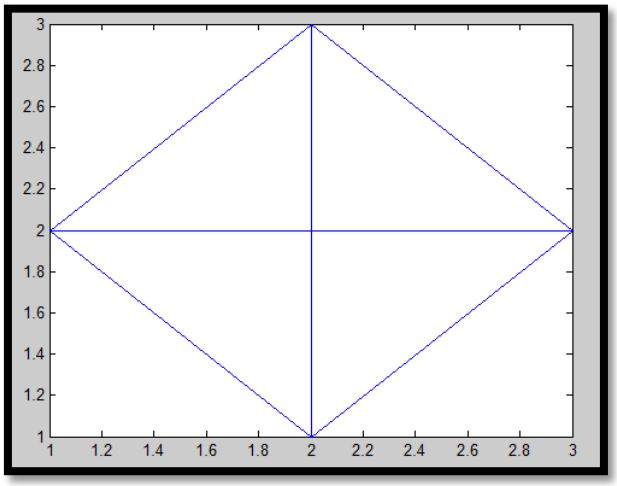

Figura 4.11 Propiedades de la figura

Cada una de las líneas se considera como una propiedad de la figura global. Por lo tanto, dependiendo de la flecha que se quiera mostrar en cada momento, algunas de las propiedades se harán visibles o invisibles. De esta manera, al inicio se cargará en pantalla una figura con todas sus propiedades de visualización desactivadas, por lo que solo se verá una figura vacía. Una vez que se ponga en ejecución el módulo, algunas de las propiedades se irán activando y desactivando dependiendo de la indicación que se quiera mostrar en cada momento. Esto se puede ver explicado de manera más detallada en el Capítulo 4.1.2.4.

4.1.2.3 bci_StartRun

Esta función se ejecutará al pulsar el botón *Start* de BCI2000. Será en este momento en el que se ejecutará una función llamada *vectorFlechas* la cual creará un vector con el orden en el que las flechas irán visualizándose en la secuencia. Como se puede ver en la Figura 4.12 el resultado de esta función está definida también como una variable global.

```
global v2;

v2 = vectorFlechas(nSecs);
```

Figura 4.12 Función bci_StartRun

La función *vectorFlechas,* como se puede ver en la Figura 4.13, tiene como parámetro de entrada el número de secuencias a realizar en la aplicación. Dentro de la función se utiliza la función *randperm(4)* de Matlab, que devuelve una permutación de cuatro enteros. Por lo tanto nunca se repetirá el orden de aparición de las flechas en distintas ejecuciones de la aplicación. Esto tiene una gran importancia, ya que se evita que el sujeto sepa de antemano el orden de aparición de las flechas en la secuencia.

```
function vector = vectorFlechas(numSecs)
vector = zeros(1, 4*numSecs);
j = 1;
for iter=1:numSecs
    x = randperm(4);
    for p=1:4
        vector(j)= x(p);
        j=j+1;
    end
end
```

Figura 4.13 Función vectorFlechas

4.1.2.4 bci_Process

Cada vez que se ejecute la función *bci_Process*, se incrementará el número de bloque como se puede ver en la Figura 4.14.

```
numBloq = numBloq +1;
```

Figura 4.14 Variable que controla el número de bloques

De esta manera, dependiendo del número de bloques se irán ejecutando distintas fases de la secuencia de indicaciones visuales (véase las Figuras 4.15-4.19).

```
if (numBloq == nBSuma1) % CRUZ
    bci_States.Flechas = 5;
    set(x5,'Visible','on')
    set(x6,'Visible','on')
    drawnow
```

Figura 4.15 Visualización de la suma previa a la flecha

Cómo se puede ver en la Figura 4.15, cuando el número de bloques sea igual que el número de bloques de la suma previa a la flecha, se harán visibles dos propiedades de la figura (x5 y x6). Estas dos propiedades corresponden a la línea horizontal y a la vertical respectivamente. De esta manera, se visualizará en la pantalla el símbolo de suma. Además, se actualizará el estado de *Flechas* a 5. Esto servirá para identificar la secuencia que se ha llevado a cabo en el fichero que BCI2000 cree con la adquisición de señales viendo a qué momento en la secuencia corresponden las señales recogidas.

Los demás casos se han llevado a cabo de la misma manera, es decir, dependiendo del momento en la secuencia se activarán/desactivarán las propiedades que conforman la figura que se mostrará.

Cada vez que se haya realizado una secuencia, se comprobará si se han realizado la cantidad de secuencias insertadas desde el parámetro de entrada *nSecs, como se puede ver en la Figura 4.19.* En el caso de que no se haya cumplido se procederá a iniciar una nueva secuencia y se actualizará el contador de las secuencias realizadas. Cuando este contador sea igual que el número de secuencias definidas, se terminará la ejecución de la aplicación (*bci_States.Running*=0).

```
elseif (numBloq == nBFlecha) %FLECHA
    bci_States.Flechas = v2(numTrial);
    switch v2(numTrial)
        case 1,
            set(x3,'Visible','on')
            set(x4,'Visible','on')
        case 2,
            set(x1,'Visible','on')
            set(x2,'Visible','on')
        case 3,
            set(x1,'Visible','on')
            set(x3,'Visible','on')
        case 4,
            set(x2,'Visible','on')
            set(x4,'Visible','on')
    end
    drawnow
```

Figura 4.16 Visualización de la flecha

```
elseif (numBloq == nBSuma2) % CRUZ
    bci_States.Flechas = 5;
    set(x1,'Visible','off')
    set(x2,'Visible','off')
    set(x3,'Visible','off')
    set(x4,'Visible','off')
    drawnow
```

Figura 4.17 Símbolo de suma después de la aparición de la flecha

```
elseif (numBloq == nBPost) % BLANCO
    bci_States.Flechas = 0;
    set(x5,'Visible','off')
    set(x6,'Visible','off')
    drawnow
```

Figura 4.18 Descanso dentro de la secuencia

```
elseif (numBloq == nBFin)
    if(numTrial < 4*nSecs)
        numBloq = 0;
        numTrial = numTrial+1;
    else
        bci_States.Running = 0;
    end
end
```

Figura 4.19 Control de final de secuencia

4.1.2.5 bci_StopRun

Cuando finalice la ejecución de las secuencias en esta aplicación, aparecerá un aviso en la pantalla indicando el fin de ésta (véase Figura 4.20) y BCI2000 guardará en un fichero todas las señales adquiridas durante todo el proceso.

```
text(0.5, 0.5, 'RUN acabado','HorizontalAlignment','center', ...
               'FontSize', 28, 'EdgeColor','blue', 'LineWidth',3, ...
               'BackgroundColor',[.7 .9 .7],'Margin',10,'Tag' , 'txtFin');
```

Figura 4.20 Función bci_StopRun

4.1.3 V-Amp 16

Durante todo el proceso de visualización de la secuencia, el sujeto del experimento tendrá colocado el actiCAP con 16 electrodos conectados a V-Amp 16. Estos estarán recibiendo constantemente la señal cerebral que se genere mientras el sujeto realice la prueba. V-Amp 16 estará conectado con BCI2000 y una vez que se termine la ejecución de la aplicación, BCI2000 se encargará de guardar las señales recibidas desde los 16 canales de V-Amp 16. En la Figura 4.21 pueden verse las conexiones realizadas entre V-Amp y el resto de herramientas en el experimento realizado. En la Figura 4.22 en cambio se puede ver al sujeto aguardando el inicio de la secuencia de indicaciones visuales con el actiCAP colocado.

Figura 4.21 Conexiones para la adquisición de datos

Figura 4.22 El sujeto del experimento aguardando la llegada de las indicaciones visuales en el monitor

4.2 Interpretación de la actividad cerebral

En el momento en el que se pone en marcha este módulo, comenzará la captura de la actividad cerebral mediante V-Amp 16. Cuando el experimento termine su ejecución, los datos adquiridos desde el amplificador se guardarán en un fichero externo que podrá ser tratado en el momento deseado.

Como se puede ver en la Figura 4.23, el objetivo es interpretar los datos registrados para conseguir mover el cursor en la pantalla.

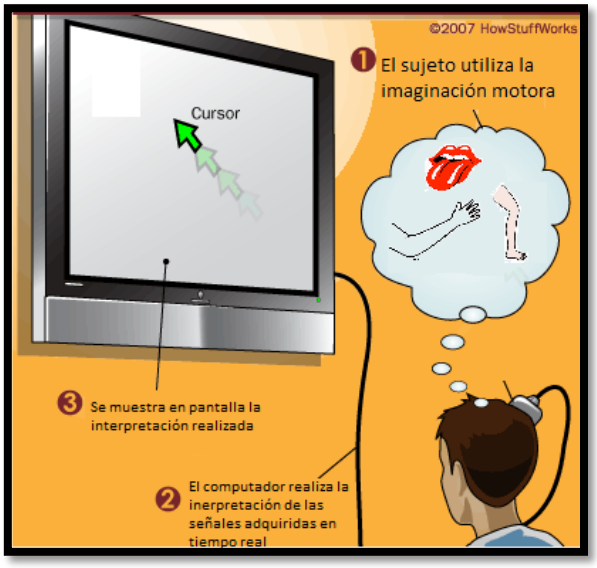

Figura 4.23 Interpretación de la acción que imagina el sujeto

El objetivo de esta parte del módulo consiste en interpretar correctamente los datos que se adquieren en tiempo real desde el actiCAP, es decir, adivinar mediante el análisis de los datos lo que el sujeto del experimento está pensando. De esta manera, dependiendo de la interpretación que se haga, un cursor en la pantalla irá moviéndose a la dirección deseada.

Para la realización de esta aplicación se han vuelto a utilizar BCI2000 y Matlab (véase Figura 4.24). BCI2000 se encarga de enviar las señales EEG de V-Amp16 recogidas en tiempo real como parámetro de entrada a las funciones de Matlab. Por lo tanto, cada vez que un bloque de señales llegue a Matlab se irá cargando en un buffer. El tamaño de bloque es configurable como en el caso de la aplicación de adquisición de datos. Cuando este buffer esté lleno se llevará a cabo una elección de movimiento. El tamaño de este buffer es configurable desde los parámetros de BCI2000. Esta función de elección será la que dirá hacia donde se debe mover el cursor en la pantalla. Para ello, interpretará los datos recogidos en el buffer y los tratará. Una vez haya devuelto la respuesta, una parte de los datos se desechará del buffer y se meterán nuevos datos provenientes del actiCAP (véase Capítulo 4.2.2.3). De esta manera, los datos obtenidos se irán analizando en tiempo real.

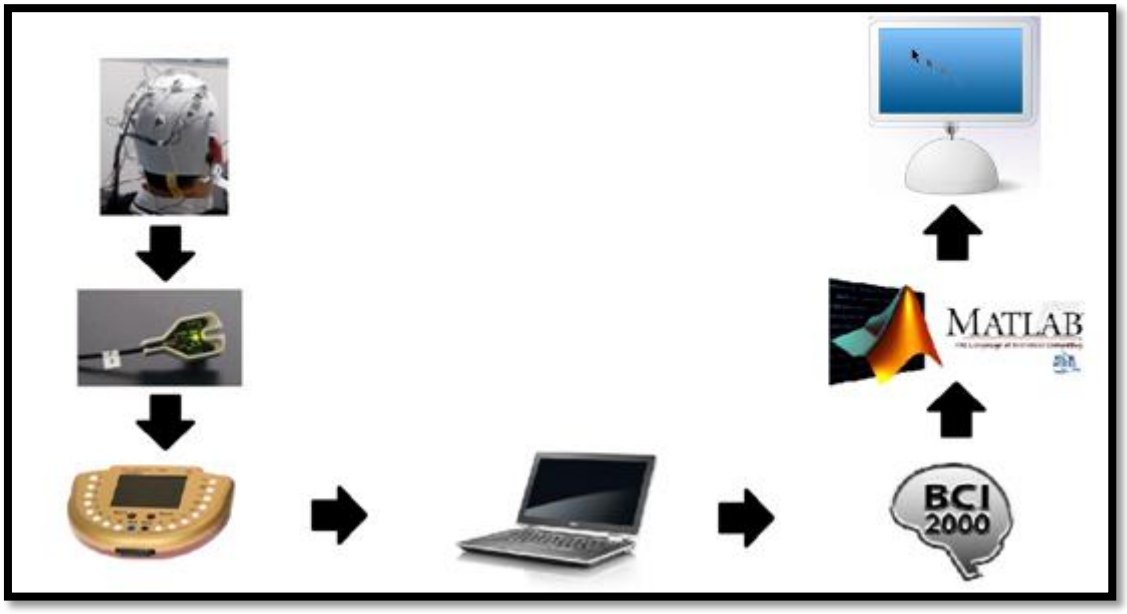

Figura 4.24 Programas utilizados en orden en esta aplicación

4.2.1 BCI2000

Las configuraciones realizadas en esta aplicación han sido distintas a la aplicación anterior. Para ello, como en el caso anterior pulsando el botón *Config* del módulo *Operator* de BCI2000 se podrán realizar las configuraciones deseadas. Los parámetros utilizados en este proyecto con esta aplicación pueden cargarse como se puede ver en la Figura 4.25.

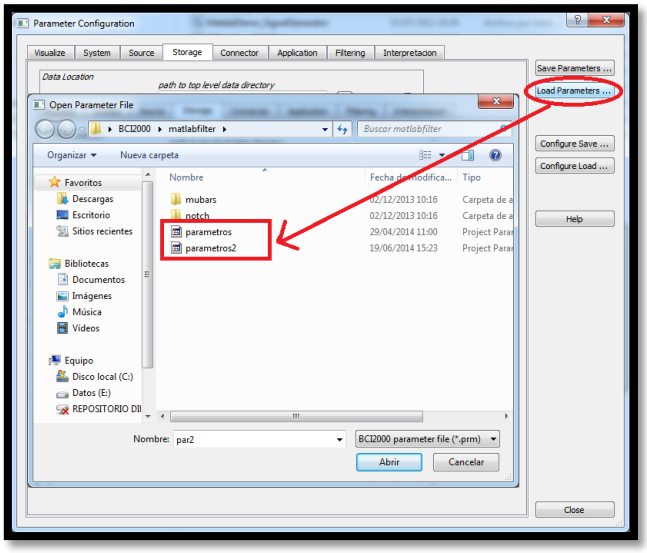

Figura 4.25 Cargar parámetros previamente guardados

Los cambios que se han realizado en los parámetros son los siguientes:

- En la pestaña *Source* se han añadido la lista de canales de transmisión que se va a usar para realizar esta aplicación (véase Figura 4.26).
- En esta aplicación, como en la anterior, se han pedido desde la función de Matlab *bci_Construct* en una pestaña llamada *Interpretación* dos parámetros de entrada configurables para el experimentador (véase Figura 4.27). El primero, T1, será el tiempo entre análisis o desplazamiento de la ventana cada vez que se procese las señales. El segundo, T2, es el correspondiente al tamaño de la ventana de análisis. Esto se explicará más detalladamente cuando se explique el uso de ellos en Matlab.
- En la pestaña *Filtering*, es muy importante quitar el filtro espacial (*SpatialFilterType*) y ponerlo en la opción *None*, como se puede observar en la Figura 4.28, ya que en este proyecto no se hace uso de ello.

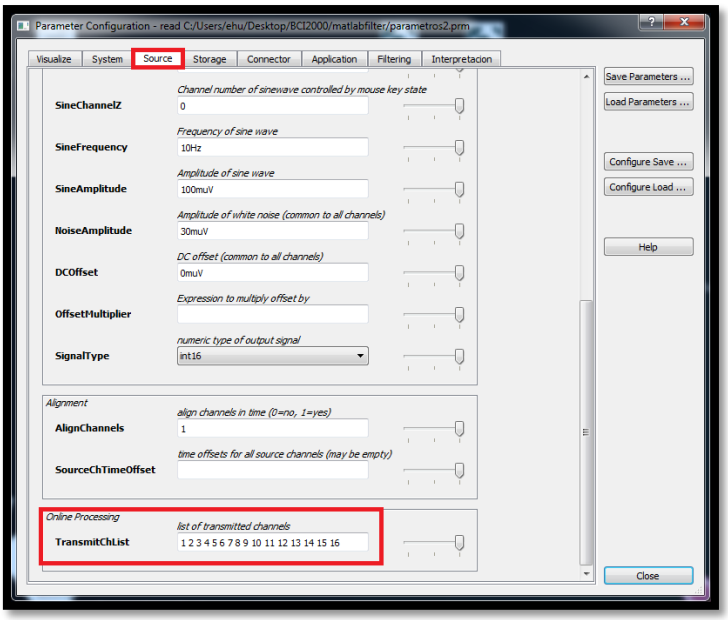

Figura 4.26 Configuraciones en la pestaña Source

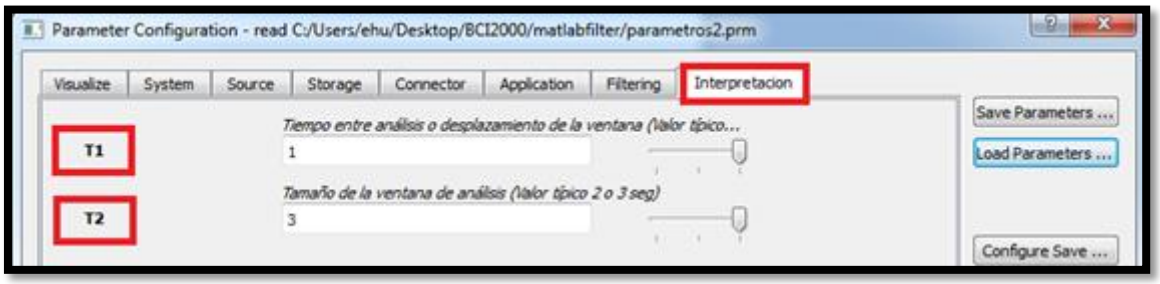

Figura 4.27 Configuraciones en la pestaña Interpretación

Figura 4.28 Configuraciones en la pestaña Filtering

4.2.2 Matlab

Como en la aplicación anterior, la programacion de la aplicación se ha realizado mediante el módulo matlabfilter de Matlab.

4.2.2.1 bci_Construct

En esta función se han definido los parámetros T1 y T2 correspondientes al tiempo entre análisis y tamaño de la ventana de análisis respectivamente (véase Figura 4.29). Estos parámetros podrán ser configurados desde la interfaz de BCI2000 en la pestaña Interpretación. Los valores que se introduzcan desde la interfaz de BCI2000 serán en segundos.

```
parameters = { ...
    [ 'Interpretacion int T1= 3 3 0
    [ 'Interpretacion int T2= 3 3 0
};

states = { ...
  'Interpretacion 4 0 0 0' ...
};
```

Figura 4.29 Definición de los parámetros utilizados dentro de la aplicación

4.2.2.2 bci_Initialize

En la función de inicialización, se han cargado algunos parámetros que se utilizarán en el programa principal mediante el comando *load*, como se puede ver en la Figura 4.30.

```
load bci_ol_parms_16.mat
```

Figura 4.30 Cargar algunos parámetros que se usarán en la aplicación

Los parámetros que se cargan desde el fichero *bci_ol_parms_16.mat* son los que serán utilizados para realizar la interpretación de las señales recibidas en el programa principal. Además, se han definido las variables globales que se usarán a lo largo de la aplicación en las distintas funciones de Matlab como se puede ver en la Figura 4.31.

```
global T1;
global T2;
global N1;
global N2;
global punt;
global x;
global y;
%-------
global W;
global filters;
global features;
global svm_model;
```

Figura 4.31 Variables globales definidas

Después de definir las variables globales, se recogen los valores de los parámetros definidos en la función *bci_Construct* y se calcula el número de bloques correspondiente a cada uno para luego usarlo en el programa principal (véase Figura 4.32). Para ello, se han tenido en cuenta los parámetros introducidos desde la interfaz de BCI2000 para T1 y T2 (segundos) y la frecuencia de muestreo que tiene cada bloque. De esta manera, como ya se verá en el programa principal, se ha podido trabajar los tiempos mediante el número de bloques.

```
T1= str2double(bci_Parameters.T1);
T2= str2double(bci_Parameters.T2);

N1= T1*fs;
N2= T2*fs;
```

Figura 4.32 Cálculo del número de bloques en T1 y T2

Además, se crea y muestra la figura en pantalla en la que el cursor (en este experimento el símbolo '+') se irá moviendo dependiendo de las señales que recoja y la interpretación que realice sobre éstas como se puede ver en la Figura 4.33.

```
x=0;
y=0;

cruz=figure, plot(x,y,'+','DisplayName', 'puntero');
axis([-10,10,-10,10]);
punt = findobj(cruz, 'DisplayName', 'puntero');
```

Figura 4.33 Inicialización de la figura con el símbolo de suma

4.2.2.3 bci_Process

La función *bci_Process* es la función principal de la aplicación. En esta función cada vez que llegue un bloque cómo parámetro de entrada, éste se irá guardando en un buffer como se puede ver en la Figura 4.34.

```
buffer=[buffer, in_signal];
```

Figura 4.34 Cargar señales de entrada en el buffer

Mientras que el tamaño del buffer no sea del tamaño de la ventana de análisis T2 se seguirá recogiendo en el buffer las señales que llegan de entrada. T2 como anteriormente se ha mencionado es un parámetro de entrada que será introducido en segundos, que medirá el tamaño de la ventana de análisis. Por lo tanto, el control del tamaño de la ventana de análisis se realizará mediante el número de bloques que puede entrar en el tiempo introducido como parámetro desde la interfaz. Esta cantidad de bloques se conocerá como N2.

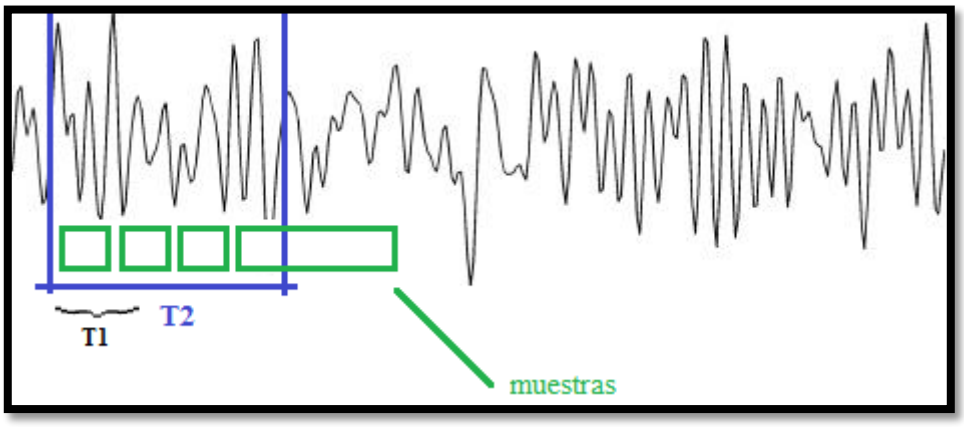

Figura 4.35 Se recogen las señales en el buffer hasta que tenga tamaño N2

Un buffer irá recogiendo bloques a través de la señal de entrada (véase Figura 4.35). Cada vez que el buffer sea del tamaño de la ventana de análisis, se realizará la interpretación de las señales recogidas en el buffer. Tras ello, se desecharán del buffer los bloques correspondientes a los T1 primeros segundos y se volverá a recoger más bloques de señal desde la entrada solapándolo con los anteriores, como se ve en la Figura 4.36.

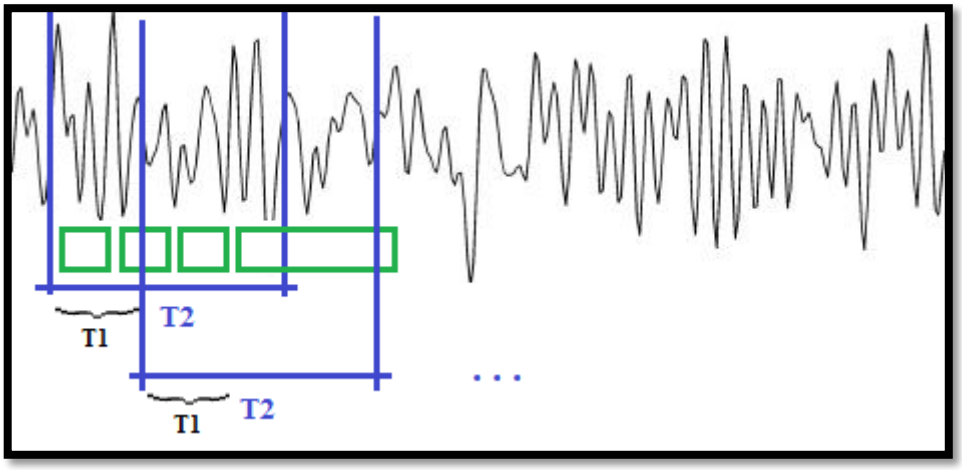

Figura 4.36 Desechar las muestras correspondientes a T1

Cuando el tamaño del buffer supere a la ventana de análisis (véase Figura 4.37) se llamará a la función *bci_ol_test* (véase Figura 4.38) la cual interpretará la señal pasada desde el buffer después de pasar unos filtros y características en función a un modelo clasificador mediante Weka. Esta función devolverá un valor del 1 al 5 dependiendo de la clase a la que interprete que pertenece. Después se desecharán del buffer que se ha tratado las muestras correspondientes al desplazamiento (T1) para que el tratamiento de la señal de una ejecución a otra sea incremental y la interpretación de ésta no se haga de manera brusca. (véase Figura 4.39). Finalmente, en función del valor devuelto por *bci_ol_test* se realizará el movimiento del cursor en la figura (véase Figura 4.40).

La función *bci_ol_test* tiene definidas 5 clases interpretables correspondientes a las acciones que el sujeto debería haber pensado querer realizar mediante el uso de su imaginación motora: movimiento de la mano derecha, movimiento de la mano izquierda, movimiento de la lengua, movimiento de los pies y un quinto estado correspondiente al momento en el que no se realiza nada. De esta forma, dependiendo del valor que devuelva la función *bci_ol_test* se moverá el cursor hacia la posición correspondiente:

- Clase=1, el cursor se moverá hacia la izquierda una posición.
- Clase=2, el cursor se moverá hacia la derecha una posición.
- Clase=3, el cursor se moverá hacia arriba una posición.
- Clase=4, el cursor se moverá hacia la abajo una posición.
- Clase=5, el cursor mantendrá su posición.

```
if(size(buffer,2)>=N2)
```

Figura 4.37 Buffer mayor que la ventana de análisis

```
cursor=bci_ol_test(buffer(:,1:N2),W,filters,features,svm_model);
```

Figura 4.38 Función de interpretación de la señal

```
buffer=[buffer(:,N1+1:end)]
```

Figura 4.39 Desechar las muestras de T1 segundos

```
if (cursor==1) %izq
    x=x-1;
    set(punt,'XData',x, 'YData', y)
    drawnow
elseif (cursor==2) %dcha
    x=x+1;
    set(punt,'XData',x, 'YData', y)
    drawnow
elseif(cursor==3) %up
    y=y+1;
    set(punt,'XData',x, 'YData', y)
    drawnow
elseif(cursor==4) %down
    y=y-1;
    set(punt,'XData',x, 'YData', y)
    drawnow
elseif(cursor==5) %no hacer nada
    set(punt,'XData',x, 'YData', y)
    drawnow
end
```

Figura 4.40 Realizar el movimiento del cursor en pantalla

Esta función tiene como entrada cinco parámetros:

- *Una matriz*: se trata de la variable buffer, que como anteriormente se ha explicado será donde se irán cargando las señales EEG cerebrales procedentes de BCI2000 como parámetro de entrada de la función *bci_Process* (*in_signal*). Como el buffer será de mayor tamaño que el tamaño de la ventana de análisis, en la función se enviarán solamente las muestras de señales que han entrado en la ventana de análisis, dejando algunas para que sean tratadas en la siguiente ejecución.
- *W*: consta de cinco matrices de proyección (1 por clase), de tamaño 5x16, calculadas utilizando un método llamado CSP (*Common Spatial Patterns*). Proyectan los canales de entrada en 25 señales con varianza entre clases maximada.
- *Filters*: contiene los coeficientes de tres filtros FIR paso banda (8-12 Hz, 12-20 Hz y 20-30Hz) que se aplican a las señales antes de calcular las características.
- *Features*: contiene una lista de las características a extraer, que han sido seleccionadas a partir de los datos del entrenamiento. Se parte de hasta 525 características distintas que podrían utilizarse (25 señales producidas por CSP x 3 filtros FIR aplicados x 7 parámetros estadísticos calculados), pero tras un proceso de selección realizado con Weka con los datos de entrenamiento, se ha elegido un subconjunto mucho menor con el que se ha entrenado el clasificador SVM (*Support Vector Machine*).
- *Svm_model*: en este parámetro se define el modelo que se utilizará para clasificar los datos.

Después de aplicar algunos filtros en la señal de entrada buffer para que éstas sean más fáciles de tratar se evalúa la base de datos de las señales en función a un modelo previamente entrenado en Weka (véase Figura 4.41).

```
eval(['!java -Xmx1024m weka.classifiers.functions.SMO', ...
      ' -l ', svm_model, '.model', ...
      ' -T test.arff -p 0 > test.out'])
```

Figura 4.41 Función de evaluación por Weka

4.3 Resultados

Una vez realizada la evaluación mediante Weka, ésta escribirá en el fichero *test.out* la interpretación realizada de las señales recibidas. En el fichero *test.out* aparecerá la predicción realizada con el trozo de señal de la matriz *buffer* como se ve en la Figura 4.42.

```
=== Predictions on test data ===

 inst#     actual  predicted error prediction
     1     5:null   3:tongue   +    0.4
```

Figura 4.42 Resultado obtenido

Como se ha podido ver en la Figura 4.42, ha habido un error en la interpretación de los datos y no se ha predicho correctamente la intención del sujeto. En el instante número 1 el bloque de señal correspondiente a la clase 5 (nulo) se ha predicho como correspondiente a la clase 3 (movimiento de lengua).

El cursor está configurado para que empiece en la posición (0,0) como se puede apreciar en la Figura 4.43. Cuando se ejecute la función *bci_ol_test* y ésta realice la interpretación de la señal, el cursor se moverá una posición dependiendo de la interpretación realizada.

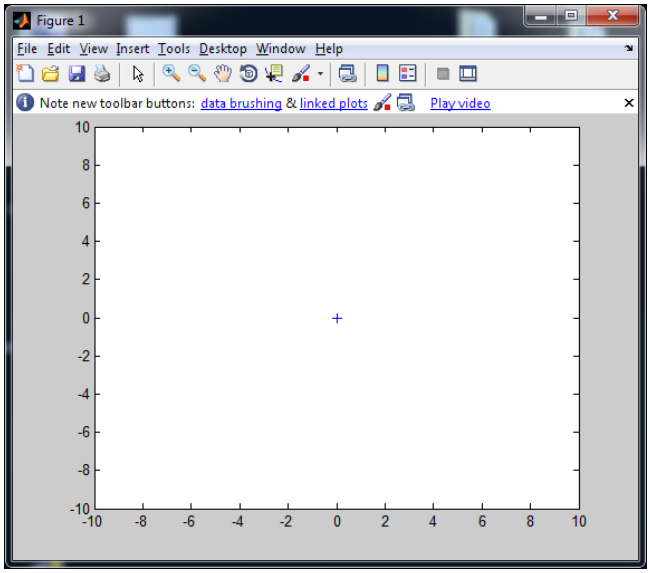

Figura 4.43 Inicio del cursor en (0,0)

Como se puede observar en la Figura 4.44, se ha podido llevar adelante la aplicación de interpretación de datos. Al predecirse que el sujeto pretendía mover la lengua, el cursor se ha ido moviendo hacia arriba.

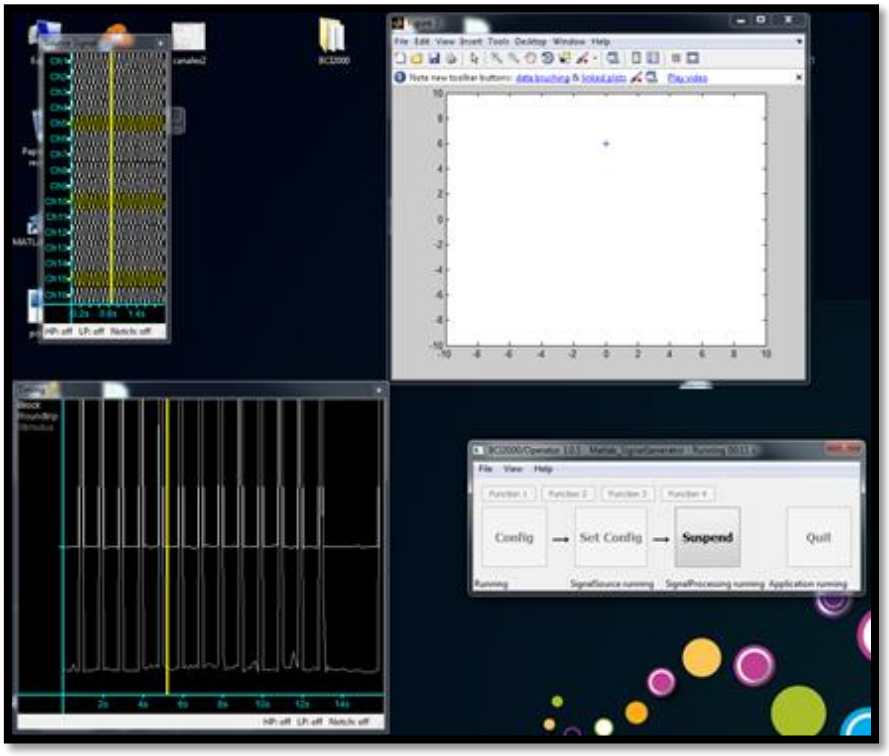

Figura 4.44 Ejecución de la interpretación de señales

Para ver que los resultados obtenidos tienen sentido, se ha realizado una prueba utilizando una base de datos de señales electroencefalográficas como entrada, en la que se saben a qué pensamiento corresponde cada trozo de señal y el resultado conseguido ha sido bastante bueno.

5

5. Conclusiones

En este capítulo se explicarán las conclusiones extraídas de este proyecto, se hará un repaso de los objetivos realizados y las posibles mejoras a realizar en un futuro.

5.1 Resumen del trabajo realizado

Los objetivos planteados en el inicio del proyecto han sido cumplidos. El objetivo principal de este proyecto era poner en marcha un sistema BCI (*Brain Computer Interface*) basado en EEG e imaginación motora, en el que el usuario mueve un cursor en una pantalla, mediante el pensamiento utilizando para ello una plataforma ya en marcha (V-Amp 16 + BCI2000).

Para poder cumplir con los objetivos, se ha estudiado el entorno BCI en general y después, más a fondo, los conceptos y uso de BCI2000 específicamente. También se ha llevado a cabo la familiarización con V-Amp 16 mediante el análisis de la herramienta y su uso junto con BCI2000.

La adquisición de señales EEG correspondientes a la imaginación motora se ha podido realizar gracias a la secuencia de indicaciones visuales implementada en Matlab, ya que dichas señales se generaban en respuesta a estas indicaciones.

Las señales recogidas se han interpretado de acuerdo a un modelo de comportamiento previamente entrenado utilizando la plataforma de aprendizaje Weka. Esta interpretación de la imaginación motora se ha traducido en el movimiento de un cursor en la pantalla en tiempo real.

La aplicación directa del proyecto desarrollado sería ayudar a personas que tengan algún tipo de discapacidad que les impida interactuar con su entorno de forma autónoma.

De cara al futuro, se plantea la realización de un experimento más amplio, que involucre a un mayor número de personas, de cara a validar la plataforma experimental propuesta. Así mismo, el diseño de los indicadores es otro aspecto que se podría mejorar, haciéndolos visualmente más atractivos.

Finalmente, se podría ampliar el objeto del experimento, yendo más allá del movimiento del cursor en una pantalla. Los resultados del experimento podrían utilizarse para interaccionar con un ordenador (por ejemplo, añadiendo la posibilidad de clickar en el sitio deseado del escritorio), para mover una silla de ruedas de forma autónoma, etc. Al fin y al cabo, el objetivo del trabajo realizado está enfocado a la ayuda para personas que tengan algún tipo de discapacidad.

5.2 Conclusiones personales

A continuación pueden verse algunas conclusiones personales extraídas después de haber realizado el proyecto.

- Como se ha podido observar después de realizar el proyecto, lo que se planteaba en la teoría se ha podido llevar a la práctica. Aunque esta área de investigación está aún en proceso de aprendizaje, se puede ver que poco a poco con el paso del tiempo este tipo de proyectos van cogiendo fuerza.
- Proyectos como éste pueden resultar de gran ayuda para que las personas discapacitadas se integren en la sociedad de manera autónoma.
- En el momento en el que el sujeto realice el experimento de adquisición de señales debe estar tranquilo y relajado y concentrarse en la mayor cantidad posible para que las señales sean mejores.

6

6. Bibliografía

[1] Gerwin Schalk, Juergen Mellinger: *A Practical Guide to Brain-Computer Interfacing with BCI2000*. http://books.google.es/books?id=20m_-r9nxgAC&printsec=frontcover&dq=isbn:1849960925&hl=es&sa=X&ei=RyOXU9SHC9K20QWor4B4&ved=0CC0Q6AEwAA#v=onepage&q&f=false

[2] Brain Products GmbH

http://www.brainproducts.com/bci_pack.php

[3] BCI2000 wiki

http://bci2000.org/wiki/index.php/Main_Page

[4] Schalk Lab

http://www.schalklab.org/research/bci2000

[5] Foro de BCI2000

http://www.bci2000.org/phpbb/

[6] How Stuff Works

http://computer.howstuffworks.com/brain-computer-interface.htm

[7] MathWorks Matlab Central
http://www.mathworks.es/matlabcentral/?s_tid=gn_mlc

[8] Irena Koprinska: Feature Selection for Brain-Computer Interfaces

[9] Brain Vision LLC
http://www.brainvision.com

[10] Wikipedia
http://www.wikipedia.org

[11] Josian Santamaria: *Mugimendu-irudimenean oinarritutako garunaren eta konputagailuaren arteko interfazea*.

[12] actiCAP

http://www.brainproducts.com/productdetails.php?id=4&tab=3

Anexos

Anexo A: Manual para el usuario

El experimento constará en utilizar la imaginación motora en función de unas indicaciones visuales que se irán alternando en cada secuencia. En total aparecerán 5 tipos de indicaciones visuales a lo largo de la secuencia: cuatro flechas en diferentes direcciones (en dos dimensiones) y el símbolo de suma. En cada una de estas secuencias, el sujeto solamente debe centrarse en una única indicación que irá siempre a continuación del primer símbolo de la suma. Las imágenes en las que el sujeto de focalizar su atención son unas flechas. Dependiendo de la dirección en la que apunten, el sujeto tendrá que esforzarse en pensar en mover una parte distinta de su cuerpo sin físicamente moverla.

En todas las secuencias se irá repitiendo un mismo patrón de indicaciones, como se puede ver en la Figura A.1.

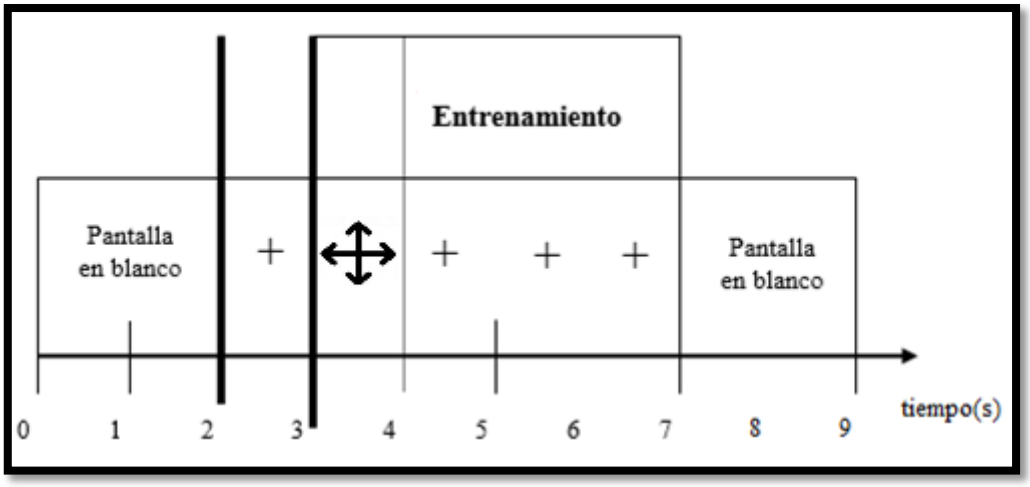

Figura A.1 Secuencia de indicaciones visuales

Explicación sobre la secuencia de indicaciones visuales:

1. **Inicio de la iteración**: La pantalla mostrará mediante un aviso que la iteración va a comenzar. Cuando pasen los dos primeros segundos, mediante el símbolo de suma se advertirá que la indicación en la que el sujeto va a tener que centrarse va a aparecer al de un segundo.
2. **Focalización**: Después de que el símbolo de suma esté durante 1 segundo en pantalla, junto al símbolo anterior de suma se mostrará en pantalla la flecha en la que el sujeto tendrá que focalizarse y hacer uso de su imaginación motora pensando en la acción acorde a la indicación. Esta flecha aparecerá solamente durante 1 segundo y después volverá a desaparecer de la pantalla dejando solamente visible el símbolo de suma. Aunque la flecha no sea visible, el sujeto deberá continuar pensando la acción a realizar hasta que la pantalla vuelva a estar vacía, es decir, hasta que siga la suma el sujeto deberá seguir pensando en la acción que debe pensar realizar. El símbolo de suma seguirá apareciendo durante 3 segundos más y luego la pantalla volverá a estar en blanco. Por lo tanto, el sujeto cuando detecte la imagen en la que debe focalizar su mente, tendrá 4 segundos en total por cada iteración para pensar en el movimiento a realizar. Estas son las acciones que el sujeto de pensar realizar mediante el uso se du imaginación motora:
    - Flecha arriba: El sujeto debe pensar en mover la lengua hacia arriba.
    - Flecha abajo: El sujeto debe pensar en mover sus pies.
    - Flecha izquierda: El sujeto debe pensar en mover su brazo izquierdo.
    - Flecha derecha: El sujeto debe pensar en mover su brazo derecho.
3. **Descanso**: Cuando el símbolo de suma desaparezca y la pantalla se mantenga vacía, el sujeto dispondrá de tiempo para descansar.

Anexo B: Manual para el experimentador

En este documento se explicará lo necesario para que el experimentador pueda poner en marcha los dos módulos de este proyecto.

Módulo de adquisición de datos

El experimentador contará con un PC en el que estarán instalados BCI2000 y Matlab. Además también contará del amplificador que se usará en este apartado llamado V-Amp 16.

Para empezar con el experimento, primero se colocarán los 16 electrodos de V-Amp 16 en los orificios deseados dependiendo de las señales que se quieran obtener. Como se ha explicado anteriormente en el capítulo 3.1, la zona en la que se debe focalizar para adquirir las señales correspondientes a la imaginación motora es la zona superior a la corteza motora. Por lo tanto, los 16 electrodos convienen distribuirlos en dicha zona. Una vez que estén posicionados en el gorro de nombre actiCAP, este se le pondrá al sujeto que realizará el experimento. Esta colocación es muy importante hacerla de modo que el punto central del casco concuerde con el punto central de la cabeza. Por eso, se recomienda medir con un metro para ver que esto se cumple.

Cuando se le coloque el casco al sujeto que realizará el experimento, se llevará a cabo la conexión de V-Amp 16 con el PC. V-Amp 16 está alimentado mediante el USB, por lo que se pondrá en marcha al conectarlo al PC. La conexión de V-Amp 16 con el PC se realizará utilizando el modo impedancia. De esta manera, en la pantalla TFT de V-Amp 16 se podrá ver la impedancia de cada uno de los electrodos en tiempo real. Para que el experimento sea más claro y no haya más ruido conviene que cada electrodo tenga menos de 100 ohmios. Para bajar el valor de cada uno de los electrodos, se deberá aplicar un gel en los electrodos que necesiten mediante el uso de una jeringuilla. Cuando los ohmios bajen de 100 se activará una luz verde en los LEDs de V-Amp 16. Esto indicará que está bien configurado y que se puede poner en marcha el experimento.

A continuación, se realizarán las configuraciones necesarias en los parámetros de BCI2000. Para ello, se arrancará BCI2000 desde el fichero batch deseado y se pulsará el botón *Config* (véase Figura A.1

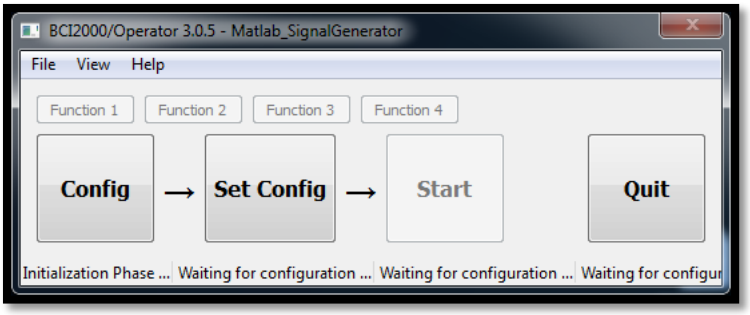

Figura A.1 Botones Config y Set Config de BCI2000

Al pulsarlo, se abrirá la ventana de configuración de BCI2000. Los parámetros de BCI2000 están distribuidos en esta ventana en distintas pestañas. La mayoría de ellas se han mantenido tal y

como venían por defecto. Para cada ejecución se recomienda cambiar el nombre del sujeto, el número de la sesión y el de la ejecución para que los resultados finales estén ordenados. Esto se hace a través de la pestaña *Storage*, como se puede observar en la Figura A.2.

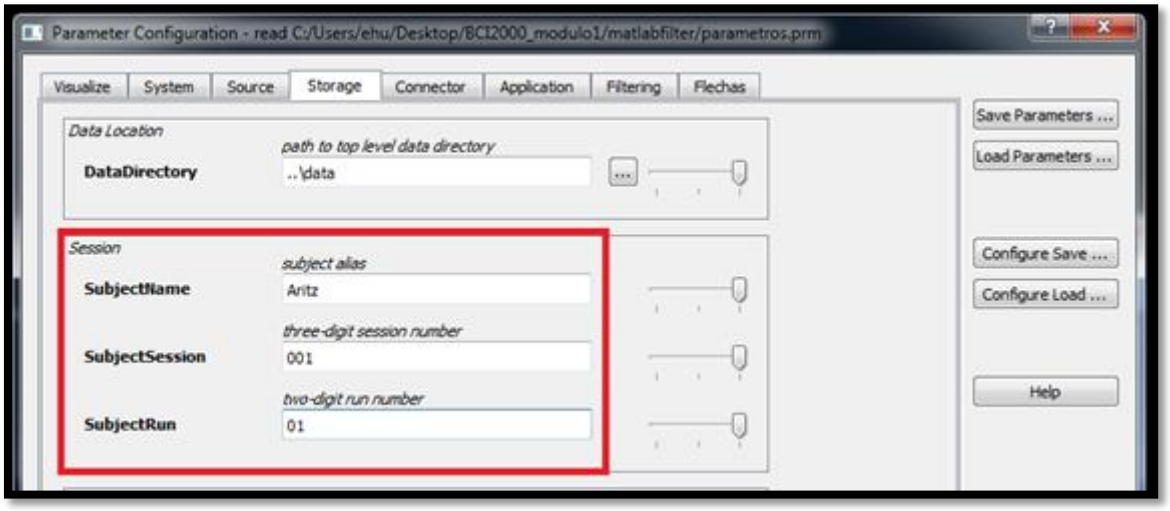

Figura A.2 Configuración del sujeto que realiza el experimento

En la ventana *Flechas* se insertarán los parámetros deseados para la secuencia de indicaciones visuales que se le pondrá al sujeto del experimento. Las medidas utilizadas en el experimento han creado una secuencia como la que se puede ver en la Figura A.3.

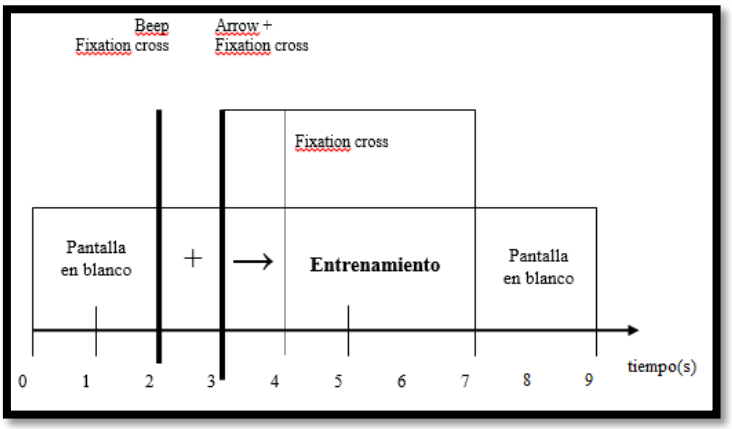

Figura A.3 Secuencia utilizada

Estos tiempos se ha predefinido desde un inicio para poder ver los patrones que se crean en la adquisición de señales y han sido los utilizados en la ejecución, como se puede ver en la Figura A.4.

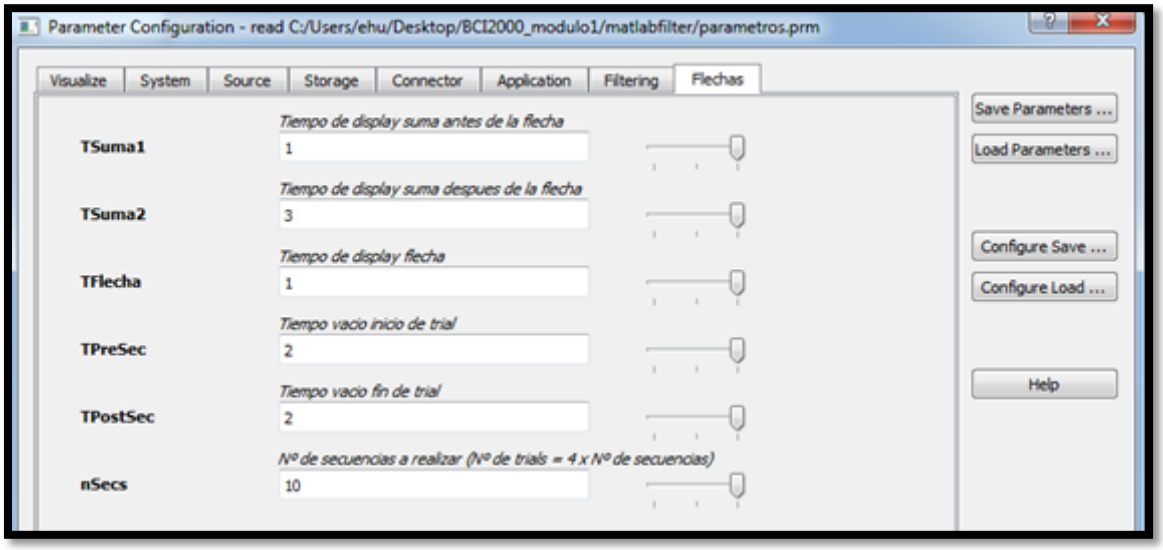

Figura A.4 Configuración de la ventana Flechas

El tiempo del experimento variará dependiendo de la cantidad de repeticiones que el experimentador crea conveniente realizar. Convendría que el experimentador antes de iniciar la ejecución diga al sujeto cuánto tiempo durará el experimento aproximadamente. De esta manera, el sujeto estaría mentalizado a la hora de realizarlo. Hacer experimentos largos sin proporcionar descansos sería perjudicante para los resultados, ya que puede que después de un rato de ejecución del experimento el sujeto comience a estresarse debido a la velocidad de la prueba. Por eso, al experimentador se le ofrece la posibilidad de controlar todos los tiempos de la secuencia mediante distintos parámetros.

Conviene conectar una pantalla externa al PC en el que se ejecutará la aplicación para ir mostrando en ella las indicaciones visuales al sujeto. De esta manera, mientras las indicaciones se van mostrando el experimentador irá adquiriendo todos los datos de actividad cerebral que se genere desde el casco gracias a V-Amp 16. Estos datos se guardarán desde BCI2000 en un fichero cuando finalice la ejecución.

Módulo de interpretación de datos

Es objetivo de este módulo es interpretar las señales electroencefalográficas y hacer mover un cursor en la pantalla. Para ello, como en caso anterior se hará uso de la interfaz de BCI2000.

Después de ejecutar el batch *Matlab_SignalGenerator* se abrirá la interfaz de BCI2000. En ella mediante el botón Config se realizarán las siguientes configuraciones:

- En la pestaña *Source* se han añadido la lista de canales de transmisión que se va a usar para realizar esta aplicación, que han sido 16 ya que son los respectivos a los que tiene el amplificador utilizado (véase Figura A.5).
- En la pestaña *Interpretación,* T1 es el tiempo entre análisis o desplazamiento de la ventana cada vez que se procese las señales y T2 es el correspondiente al tamaño de la ventana de análisis. Los valores correspondientes a estos parámetros son 1 y 3 respectivamente (véase Figura A.6).

- En la pestaña *Filtering*, es muy importante quitar el filtro espacial (*SpatialFilterType*) y ponerlo en la opción *None*, como se puede observar en la Figura A.7, ya que en este proyecto no se hace uso de ello.

Figura 4.5 Configuraciones en la pestaña Source

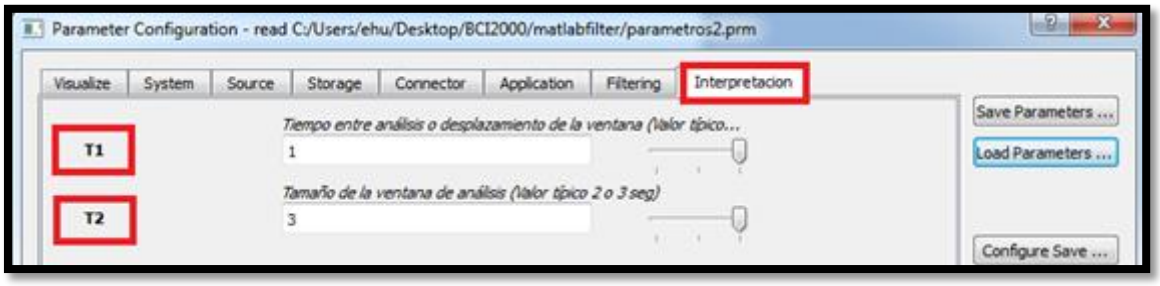

Figura A.6 Configuraciones en la pestaña Interpretación

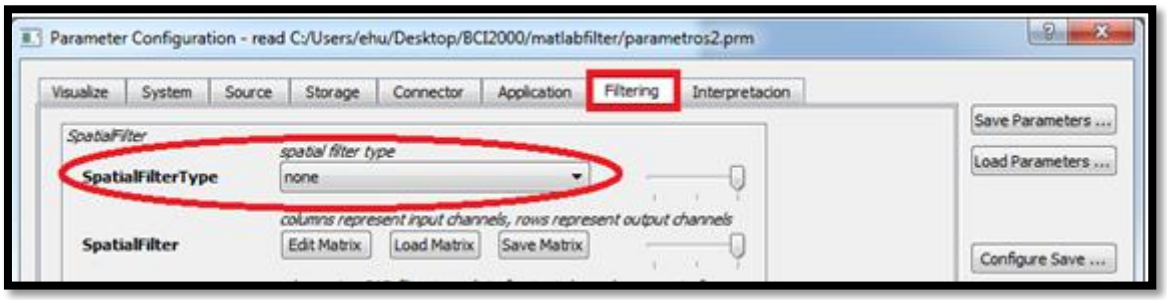

Figura A.7 Configuraciones en la pestaña Filtering

Los cambios realizados tendrán efecto una vez pulsado el botón Set Config de BCI2000. Al ejecutar la aplicación, se ejecutara la interpretación de las señales que lleguen de entrada al sistema. Dependiendo de la señal entrante, se moverá un cursor en la pantalla hacia el sitio interpretado.