Universidad Autónoma Metropolitana
Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Comparativa de Métodos de Extracción de Características para Reducción de Variabilidad Inter-Usuario en una Interfaz Cerebro-Computadora.

Proyecto de Investigación

Trimestre 2017 Invierno

Alumna:
Ana Victoria Cruz Méndez
Matricula: 2123000075
al2123000075@alumnos.azc.uam.mx

Asesor:
Dr. Eduardo Rodríguez Martínez
Departamento de Electrónica
erm@correo.azc.uam.mx

Asesora:
M. en C. Hilda María Chablé Martínez
Departamento de Electrónica
hmcm@correo.azc.uam.mx

Declaratoria

Nosotros, Dr. Eduardo Rodríguez Martínez y M. en C. Hilda María Chablé Martínez, declaramos que aprobamos el contenido del presente Reporte de Proyecto de Integración y damos nuestra autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Asesores:

Rodríguez Martínez E.

Dr. Eduardo Rodríguez Martínez

Hilda Ch

M. en C. Hilda María Chablé Martínez

Yo, Ana Victoria Cruz Méndez doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Alumna:

Ana Victoria Cruz Méndez

Resumen

En el presente reporte se explica el desarrollo de una Interfaz Cerebro-Computadora que permita hacer adquisición de datos a través de una diadema Emotiv Epoc+14. Los datos son procesados y se caracterizan de manera qe se puedan estudiar diferentes tipos de movimiento. Para el caso específico, se distinguen dos movimientos: brazo izquierdo y pierna derecha. La extracción de características se realizó centrada en dos bandas (beta y mu).

Posteriormente, se aplicó una Transformada Wavelet Discreta sobre los datos para poder hacer su caracterización en el dominio de las frecuencias mencionadas anteriormente. Los resultados de dicha interfaz están abiertos a agregar nuevos métodos de extracción así como para implementar diferentes métodos de clasificación de los mismos. Esto permitirá hacer una comparativa de diferentes métodos para comprobar cual es el que arroja un mejor rendimiento.

Índice general

Índice de figuras

1. Introducción

Una interfaz cerebro-computadora (Brain Computer Interface, BCI por sus siglas en inglés) es un sistema que intenta detectar y decodificar señales electroencefalográficas (EEG), que representan la actividad de los potenciales eléctricos cerebrales, para después transformarlas en acciones que pueda ejecutar una máquina – como una silla de ruedas. [1]

El estado del arte ha visto un tremendo crecimiento en las aplicaciones de BCI. Esto ha sido motivado por dos eventos importantes. El primero es la mejora y abaratamiento en la producción de sensores/electrodos electroencefalográficos. El segundo está relacionado con los recientes desarrollos en las ciencias de la computación que permiten procesar la señal adquirida de forma más rápida y precisa.

Las interfaces BCI (BCIs) ofrecen una oportunidad sin precedentes a la comunidad de inteligencia computacional. La relación orgánica entre las BCI y las redes neuronales artificiales (Artificial Neuronal Networks, ANNs por sus siglas en inglés) ha promovido la aplicación de estas últimas en tareas de clasificación y/o control basadas en BCIs. La investigación sobre las BCI ha penetrado en el campo de los sistemas difusos, en donde se han desarrollado aplicaciones de control. El campo de la computación evolutiva ha contribuido al desarrollo de aplicaciones BCI en donde la extracción de características se ha realizado mediante algoritmos evolutivos.

El reto práctico y fundamental en este sistema BCI consiste en combatir la variabilidad que existe entre diferentes usuarios e incluso entre distintas sesiones de un mismo usuario. Esto se debe a que un mismo usuario puede presentar variaciones en las señales electroencefalográficas (EEG) para el mismo proceso mental en diferentes instantes de tiempo. Este fenómeno puede ser causado por cambios fisiológicos, por el impacto de la carga de trabajo, por una demanda física inusual, y/o por efectos secundarios en el consumo de café o alcohol. Estas variaciones deterioran el desempeño de un sistema BCI y reducen su confiabilidad, independientemente de que tan poderoso haya sido el modelo computacional planteado. Al eliminar dicha variabilidad, los sistemas BCI pueden generalizarse a aplicaciones más ambiciosas, con múltiples usuarios en diferentes contextos.

En particular, hablamos de la intención de movimiento. Esta se refiere al intento de determinar que movimiento realizará el individuo o que parte de su cuerpo se moverá, basado en las señales EEG obtenidas de cada individuo mediante el uso de la diadema Emotiv Epoc+14. Sin embargo, no todos los ritmos que se extraen permiten identificar la intención de movimiento, los ritmos asociados para la detección de movimiento en extremidades son: beta y mu [2].

El ritmo mu se encuentra en la banda de 8 a 13 Hz y tiene una amplitud menor a 50 µV; suele asociarsele con el imaginar o anticipar un movimiento. El ritmo beta está en la banda de 13 a 30 Hz, y tiene una amplitud aproximada de 2 a 20 µV. El ritmo mu se enfoca en la anticipación de un movimiento, mientras que el ritmo beta permite realizarlo [3].

2. Justificación

En el presente proyecto se propone comparar el desempeño de al menos dos métodos de extracción de características, y evaluar su impacto en la reducción de la variabilidad inter-usuario, para la tarea de detección de intención de movimiento, por medio de un sistema BCI basado en un sensor de bajo costo.

Actualmente, cuando un usuario desea utilizar una interfaz cerebro-computadora necesita realizar un proceso de adaptación que ajuste un modelo previamente entrenado, a los patrones de la señal EEG característica de ese usuario. Este proceso se necesita repetir al cambiar de usuario, por lo que resulta tedioso y reduce el interés por usar este tipo de tecnología, a estos dos factores lo consideramos como variabilidad.

Las dos fases más importantes dentro de la detección de intención de movimiento son; la extracción de características y el método de clasificación. En la primera fase se extrae de la señal EEG información relevante para caracterizar la intención de movimiento y facilitar la tarea de clasificación. Desafortunadamente esta etapa se ve altamente afectada por la variabilidad inter-usuario. Encontrando un método de extracción de características invariante al cambio de usuario se eliminaría la etapa de adaptación, agilizando el uso de las interfaces cerebro-computadora.

Para el proceso de clasificación de intención de movimiento se ha reportado el uso de distintos clasificadores, entre los más populares se encuentran: el clasificador de vecinos cercanos, el clasificador Bayesiano y la máquina de soportes vectorial. De entre ellos, el clasificador Bayesiano necesita el menor número de parámetros; al ser un clasificador estadístico, su entrenamiento es bastante sencillo; y no se requieren muchos cálculos para clasificar un nuevo punto. Es por esto que en la presente propuesta se eligió como método de clasificación el clasificador Bayesiano.

3. Antecedentes

Entre los trabajos relacionados con interfaces cerebro-computadora se tomaron como referencia los siguientes.

3.1. Proyectos de Integración

- **Interfaz Cerebro-Computadora para el Control de un Cursor Basada en Ondas Cerebrales** [4]. Este proyecto de integración se refiere al desarrollo de una nueva interfaz de comunicación mediante la extracción de características de ondas EEG. Como en nuestra propuesta, se hace la adquisición de señales así como el filtrado y clasificación de las mismas, la diferencia radica en los métodos de clasificación de ondas EEG.
- **Implementación de señal EEG de intención de movimiento para teleoperación de robot móvil diferencial** [5]. En este proyecto terminal, al igual que en nuestra propuesta se realiza la identificación de las señales que realizan la intención de movimiento de la mano izquierda y derecha. La diferencia es que en este no se intenta disminuir la variabilidad inter-usuario.

3.2. Proyectos Externos

- **Real-time EEG based object recognition system using Brain Computer Interface** [6]. En el artículo se explica el proceso de adquisición de señales EEG en tiempo real, además de la implementación de diversos algoritmos para la extracción de características y clasificación de datos. El objetivo, como en nuestra propuesta, es encontrar un algoritmo eficiente de clasificación; sin embargo, en el artículo se usan diferentes algoritmos de clasificación al propuesto en este documento.
- **Methodology for Analysis of Stress Level Based on Asymmetry Patterns of Alpha Rhythms in EEG Signals** [7]. Se describe el proceso de adquisición y procesamiento de señales EEG mediante la diadema Emotiv EPOC además de su correspondiente análisis para determinar el nivel de estrés de una persona basada en los ritmos alfa obtenidos. La similitud con la propuesta es que también se requieren analizar ondas cerebrales; sin embargo el análisis de esta propuesta se hará con los ritmos beta y mu para detectar la intención de movimiento.

3.3. Tesis

- **Human Machine Interaction via Brain Activity Monitoring** [8]. En el artículo se efectua la implementación de un robot móvil diferencial de ruedas que se controla mediante

la identificación de actividad cerebral. La relación que tiene con nuestra propuesta, se usó una BCI para registrar e identificar los patrones de los usuarios, la diferencia es que en el artículo se realiza una aplicación de control.

- **SSVEP-based brain computer interface using the Emotiv EPOC** [9]. Es un proyecto de investigación en el cual se desarrolla una BCI que estudia los potenciales evocados visuales de estado estacionario (SSVEP por sus siglas en inglés) para controlar objetos desplegados en una pantalla. En el presente documento también se propone analizar los ritmos cerebrales para detectar movimiento sin la participación directa de movimiento físico por parte del sujeto con la diferencia de que en nuestra propuesta no se pretende realizar una aplicación de control.

4. Objetivos

4.1. Objetivo General

Diseñar e implementar un sistema que permita comparar al menos dos métodos de extracción de características para evaluar su impacto en la reducción de la variabilidad inter-usuario.

4.2. Objetivos Particulares

- Diseñar e implementar cada uno de los módulos que conforman la BCI: adquisición, extracción de características, entrenamiento y clasificación.
- Diseñar y construir una base de datos de señales EEG para detección de intención de movimiento.
- Seleccionar al menos dos métodos de extracción de características que alimenten el clasificador.
- Implementar un clasificador Bayesiano que permita reconocer la intención y tipo de movimiento.

5. Marco Teórico

5.1. Actividad eléctrica neuronal

El EEG está compuesto por las actividades eléctricas en conjunto de neuronas, con contribución de células gliales. Las neuronas al poseer propiedades eléctricas producen campos eléctricos y magnéticos que alcanzan la superficie del cuero cabelludo.

5.1.1. Neurona

La neurona es la unidad básica del cerebro, es una célula especialmente diseñada para transmitir información por medio de impulsos eléctricos a otras células nerviosas, musculares, o glándulas. Las células especializadas interconectadas forman el sistema nervioso, estas incluyen células nerviosas o neuronas y células gliales o glía. El cerebro mamífero contiene entre 100 millones – 100 billones de neuronas dependiendo de la especie. Cada neurona consiste de tres partes básicas: Cuerpo celular o soma, dendritas o terminales nerviosas y axón (*Figura 5.1)*. [10]

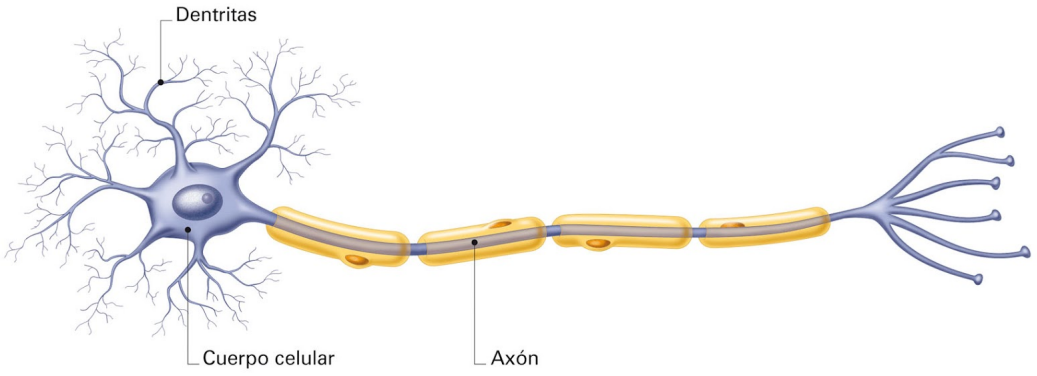

Figura 5.1: Partes de la neurona

.

- **Cuerpo celular o soma** - Considerada como la parte principal, contiene el núcleo (donde se encuentra el ADN), el retículo endoplásmico, el citoplasma, los ribosomas y la mitocondria. Desde el soma nacen dos tipos de prolongaciones, las dendritas y el axón.
- **Dendritas o terminales nerviosas** - Las dendritas son prolongaciones que proceden del soma, tienen una apariencia similar a ramas o puntas. Estas se encargan de recibir la información de otras células. Las dendritas pueden encontrarse en uno o ambos lados de la célula.

- **Axón** - Es una estructura alargada que parte del soma en un área especializada llamada cono axónico. Se encarga de conducir los impulsos nerviosos desde el soma hacia otra célula, las distancias que abarca pueden ir desde una pequeña fracción de centímetro hasta más de un metro. Los axones están cubiertos de una sustancia llamada mielina, que acelera la transmisión del impulso nervioso. La cobertura de mielina está hecha de células especializadas llamadas glía. En su extremo el axón se divide en ramas y desarrolla estructuras bulbosas conocidas como terminales axónicas o terminales nerviosas, que son las que forman las conexiones con las células blanco u objetivo.

5.1.2. Tipos de neuronas

Al ser tan complejo el funcionamiento del cerebro y el sistema nervioso tan extenso las células nerviosas se adaptan a la multiplicidad de tareas especializándose y dividiéndose en distintos tipos de neuronas, todo esto para poder cubrir todas las tareas necesarias. A continuación, se muestra una breve clasificación: [11]

1. Según la transmisión del impulso nervioso
    - Neurona presináptica - La neurona presináptica contiene el neurotransmisor y lo libera al espacio sináptico para que pase a otra neurona.
    - Neurona postsináptica - Es la neurona encargada de recibir el neurotransmisor de la neurona presináptica.
2. Según su función
    - Neuronas sensoriales - Las neuronas sensoriales recopilan información sobre lo que sucede dentro y fuera del cuerpo, y la llevan hacia el SNC (Sistema Nervioso Central) para que sea procesado.
    - Neuronas motoras - Las neuronas motoras obtienen información del SNC y la envían a otras neuronas — tales como las motoneuronas somáticas, las motoneuronas vicerales e incluso al músculo liso — para efectuar movimiento. Es decir, transmiten órdenes a los músculos, órganos y glándulas para que realicen movimientos.
    - Interneuronas - Las interneuronas, también conocidas como neuronas integradoras o de asociación, ubicadas únicamente en el SNC se encargan de la interconexión de las neuronas. Reciben información de neuronas como las sensoriales o interneuronas y la transmiten a neuronas motoras o interneuronas, pero nunca se conectan con receptores sensoriales o fibras musculares. Realizan funciones más complejas y actúan en los actos reflejo.
3. Según la dirección del impulso nervioso
    - Neuronas aferentes - Son neuronas sensoriales ya que se encargan de transportar el impulso nervioso desde los receptores u órganos sensoriales hasta el SNC.
    - Neuronas eferentes -Son neuroReciben información de neuronas como las sensoriales o interneuronas y la transmiten a neuronas motoras o interneuronas, pero nunca se conectan con receptores sensoriales o fibras muscularesnas motoras, encargadas de transportar los impulsos nerviosos fuera del SNC hacia efectores (músculos o glándulas).

4. Según el tipo de sinapsis

    - Neuronas excitatorias - Son aquellas en el que la sinapsis provoca una respuesta excitatoria o excitación, aumenta la posibilidad de producir un potencial de acción.
    - Neuronas inhibitorias - En estas el resultado de la sinapsis provoca una respuesta inhibitoria, reduciendo la posibilidad de producir un potencial de acción.
    - Neuronas moduladoras -Las neuronas moduladoras no generan una señal transmisora, sino que, se encargan de regularla; modulan la respuesta de la célula durante la transmisión sináptica a un neurotransmisor principal. Sus principales neurotransmisores son la dopamina, serotonina y acetilcolina.

5. Según el neurotransmisor

    - Neuronas Serotoninérgicas -Estas neuronas se encargan de transmitir el neurotransmisor llamado Serotonina, relacionado con el estado de ánimo.
    - Neuronas Dopaminérgicas - Este tipo de neuronas transmiten la Dopamina, relacionado con la conducta adictiva.
    - Neuronas GABAérgicas - Transmiten el principal neurotransmisor inhibitorio GABA.
    - Neuronas Glutamatérgicas - Estas transmiten el Glutamato, neurotransmisor excitatorio.
    - Neuronas Colinérgicas - La Acetilcolina, encargada de la memoria a corto plazo y el aprendizaje es transmitida por medio de las neuronas Colinérgicas.
    - Neuronas Noradrenérgicas - Estas neuronas transmiten la Noradrenalina (Norepinefrina).
    - Neuronas Vasopresinérgicas - Como su nombre lo indica, las neuronas Vasopresinérgicas transmiten la Vasopresina.
    - Neuronas Oxitocinérgicas - Transmiten la Oxitocina.

6. Según su morfología externa

    - Neuronas Unipolares o Pseudounipolares - Poseen una sola prolongación de doble sentido que sale del soma y que actúa como dendrita y axón, generalmente son neuronas sensoriales. Neuronas bipolares - Tienen dos extensiones citoplasmáticas o prolongaciones que salen del soma. Una funciona como dendrita y la otra como axón.
    - Neuronas multipolares - Tienen una gran cantidad de dendritas y un solo axón. Estas se localizan en el cerebro o en la médula espinal.

5.1.3. Comunicación neuronal

Sinapsis

Conocemos como sinapsis a la comunicación entre las neuronas y otras células. La comunicación se lleva a cabo mediante el contacto entre ambas células donde la terminal presináptica

de la célula emisora contacta la membrana postsináptica de la célula receptora, aumentando o disminuyendo la probabilidad de que dispare su propio potencial de acción. Es decir, en estas uniones las neuronas que entran en excitación, son inhibidas o moduladas. La transmisión sináptica puede ser eléctrica o química, en algunos casos puede ser ambas durante la misma sinapsis.

La sinapsis eléctrica ocurre cuando la terminal presináptica está en continuidad eléctrica con la postsináptica. Esta continuidad o conexión física se le llama unión en hendidura, que permite que la corriente de iones fluya de una célula a otra. Este tipo de transmisión es más rápido que la sinapsis química. La sinapsis eléctrica permite la actividad sincronizada de grupos de células. También puede llevar corriente en ambas direcciones, de modo que los papeles de la neurona postsináptica y presináptica se invierten. (*Figura 5.2).*

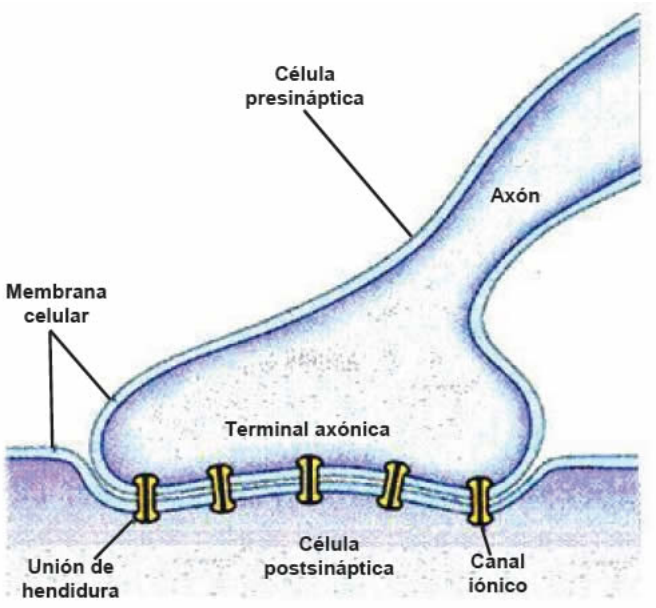

Figura 5.2: Sinápsis eléctrica

.

La sinapsis química es más común y más complicada que la sinapsis eléctrica. En ella ocurre la liberación de los neurotransmisores que llevan información de la neurona presináptica a la postsináptica. Los neurotransmisores son substancias neuroactivas que son liberadas al lado de la unión presináptica. La sinápsis química puede dividirse en dos tipos, El tipo I es excitatoria, ubicada en las dendritas, el cambio ocasionado en la célula blanco es más propenso a que dispare su propio potencial de acción, llamado potencial excitatorio postsináptico (PEPS) y el tipo II inhibitoria localizada en las células del cuerpo, si el cambio provocado en la célula blanco la hace menos propensa a disparar su potencial de acción, se le llama potencial inhibitorio postsináptico (PIPS). Dependiendo del tipo de transmisión las substancias liberadas son diferentes. El PEPS despolariza, ocasionando que el interior de la célula se vuelva positivo, acercando el potencial de membrana a su umbral de disparo de un potencial de acción. Los PIPS generan la reacción opuesta al PEPS, manteniendo el potencial de la membrana postsinaptica debajo del umbral de disparo de potencial de acción. Estos pueden contrarrestar los efectos excitatorios de los PEPS.[12] (*Figura 5.3).*

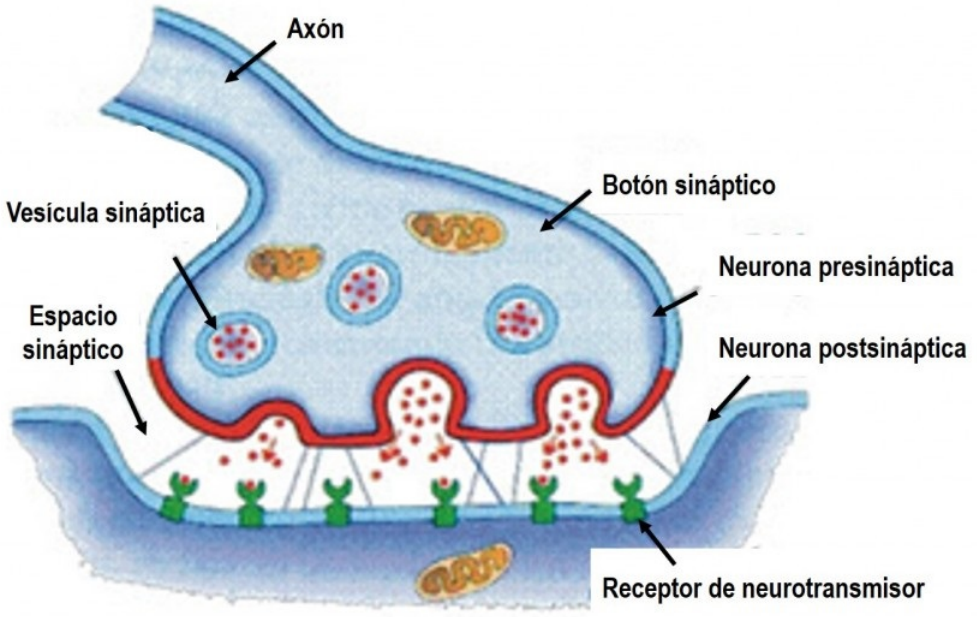

Figura 5.3: Sinápsis química

.

5.1.4. Potenciales de acción

Las señales eléctricas generadas por las neuronas son llamadas potenciales de acción. Se utilizan en el cuerpo para transportar información entre tejidos volviéndolo una característica vital. El potencial de acción es el cambio rápido y temporal en el potencial eléctrico a través de una membrana, de negativo a positivo y de positivo a negativo. [12]

Las variaciones del potencial de la membrana durante el potencial de acción se deben a los cambios en la permeabilidad de la membrana celular a iones de sodio (Na) y potasio(K), ambas proteínas, que en consecuencia, generan modificaciones en las concentraciones iónicas en los comportamientos intracelular y extracelular. Este proceso consta de tres fases:

1. Reposo. Durante esta etapa la membrana se encuentra polarizada con -70 mV.
2. Despolarización. Existe una mayor entrada Na+, aumentando el potencial de la membrana.
3. Repolarización. Existe una mayor entrada de K, y una disminución en el flujo de Na+, provocando una diferencia de voltaje ligeramente menor que en el estado de reposo.

Los potenciales de acción se ocasionan cuando una despolarización inicial alcanza un umbral, que generalmente se encuentra entre -55 y -50 MV sobre el potencial de reposo de la célula. Así la entrada de iones de sodio supera a la salida de iones de potasio, el flujo de los iones de sodio despolariza el potencial de la membrana desencadenando una apertura de los canales de

sodio dependientes del voltaje, aportando en consecuencia un flujo mayor de corrientes iónicas hacia el interior, aumentando la despolarización en una retroalimentación positiva que hace que la membrana llegue a niveles de despolarización elevados. Los potenciales de acción se propagan como una interacción pasiva entre la despolarización que se mueve por la membrana y los canales de sodio, si la membrana celular se despolariza lo suficiente para abrir los canales de sodio dependientes del voltaje, los iones de sodio entran en la célula impulsando a los iones próximos por el axón mediante la repulsión electrostática, atrayendo a los iones negativos desde la membrana continua. Una corriente positiva se desplaza a través del axón. Una vez que la membrana adyacente está despolarizada, sus canales de sodio dependientes del voltaje se abren, reiniciando el ciclo. [12]

5.1.5. Ondas cerebrales

Las ondas cerebrales son producidas por los pulsos eléctricos sincronizados de neuronas comunicándose entre ellas. La combinación de la actividad eléctrica del cerebro es llamada onda cerebral, debido a que es cíclica, parecida a las ondas u olas del mar. Las ondas eléctricas registradas en el cerebro, emiten pequeños impulsos electroquímicos de frecuencias distintas. Estas ondas cerebrales son divididas de acuerdo a su ancho de banda, pero también pueden ser consideradas como partes continuas del espectro consciente; que abarca de lento, ruidoso y funcional a rápido, sutil y complejo. La velocidad de las ondas cerebrales se mide en Hertz (*Figura 5.4).*

1. Ultra bajas. Menores a 0.5 Hz, conocidas también como Potenciales Corticales Lentos, consideradas como los ritmos básicos corticales, tienen un papel importante en la funcionalidad de la red cerebral.

2. Delta. De 0.5 a 3 Hz, son lentas y ruidosas, parecidas a las ondas emitidas por un tambor. Son generadas en la meditación profunda, sueño profundo o estado de inconciencia. La sanación y regeneración son estimuladas en este estado.

3. Theta. De 3 a 8 Hz, estas ocurren en su mayoría durante el sueño, pero también son dominantes en la meditación profunda. En ellas se da el aprendizaje y la memoria.

4. Alpha. De 8 a 13 Hz, son dominantes durante el pensamiento y en algunos estados de meditación. Las ondas alpha ayudan en la coordinación mental, calma, estado de alerta, integración mente-cuerpo y el aprendizaje.

5. Beta. De 14 a 30 Hz, dominan nuestro estado de conciencia cuando la atención está dirigida hacia actividades cognitivas. Las ondas beta son relacionadas con actividades rápidas, estados de alerta, atención, toma de decisiones, juicios y concentración.

6. Gamma. De 38 a 42 Hz, son las ondas cerebrales más rápidas y están relacionadas al procesamiento simultáneo de información en diferentes partes del cerebro. [13]

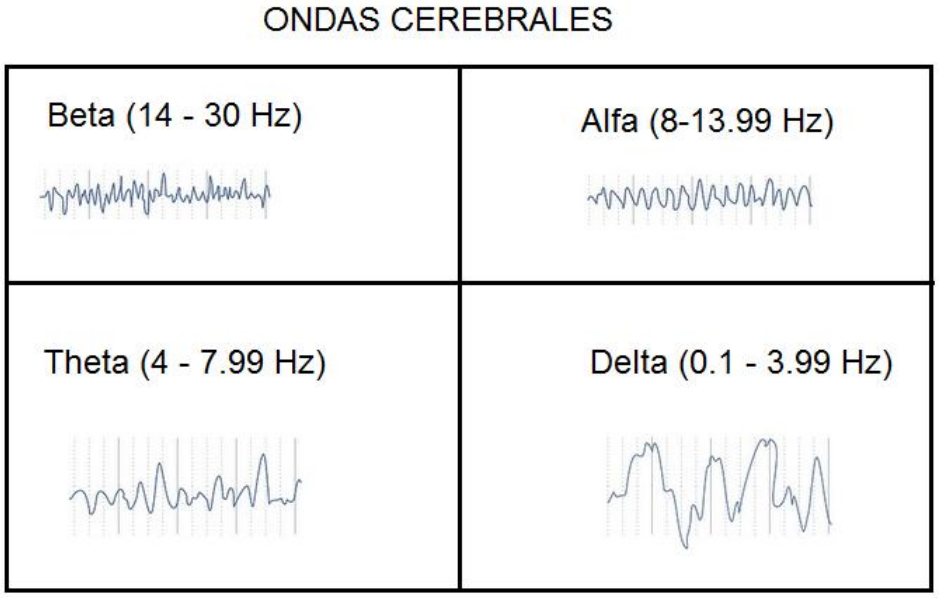

Figura 5.4: Ondas cerebrales

.

5.1.6. Potenciales relacionados a eventos

Un potencial relacionado a evento (ERP) es la medida de la respuesta cerebral como resultado directo a un evento sensorial, cognitivo o motriz, estos son medidos con una electroencefalografía (EEG). Un electroencefalograma se compone de un conjunto de señales eléctricas medidas sobre el cuero cabelludo a través de una serie de electrodos colocados en puntos característicos. Dichos potenciales reflejan simultáneamente miles de procesos cerebrales, y son clasificados de acuerdo a sus características de polaridad, latencia, y distribución. Para poder registrar adecuadamente un ERP es necesario que se lleven a cabo muchas pruebas (100+) y sacar el promedio, para que las actividades aleatorias cerebrales se eliminen y quede únicamente el ERP. La actividad aleatoria del cerebro e interferencia electromagnética forman el ruido en el registro de los ERP. [14]

5.2. Interfaz cerebro-computadora

5.2.1. Definición

Una interfaz cerebro-computadora (Brain Computer Interface, BCI por sus siglas en inglés) es un método de comunicación basado en actividad neuronal que intenta detectar y decodificar señales electroencefalográficas (EEG), que representan la actividad de los potenciales eléctricos

cerebrales, La actividad neuronal usada en una BCI puede ser obtenida por métodos invasivos o no invasivos. el objetivo de una BCI no es determinar las intenciones de una persona al ver su actividad neuronal, sino proveer un nuveo canal de salida de las señales EEG que requiere control adaptativo voluntario por el usuario, dichas señales pueden ser posteriormente transformadas en acciones que pueda ejecutar una máquina. [1] [15]

5.2.2. Estructura de una BCI

El objetivo general de una BCI es permitir que el usuario interactue con un dispositivo, la cual se logra mediante una serie de componentes funcionales intermedios como señales de control, ciclos de retroalimentación, etc. como de muestra en la *F*igura 5.5. Mediante los diversos componentes que conforman la BCI, se intenta convertir la intención del usuario en una acción que pueda ser llevada a cabo por dicho dispositivo. [15]

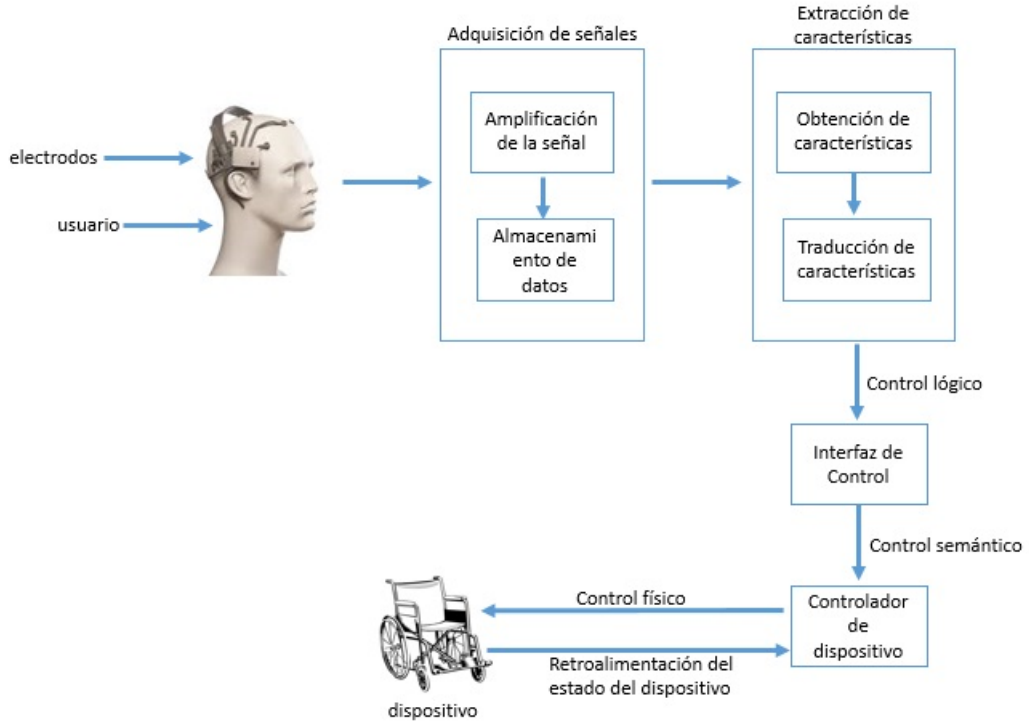

Figura 5.5: Componentes básicos de una BCI.

.

5.2.3. Adquisición de señales

- **Métodos invasivos**
  En los métodos invasivos se utilizan equipos especiales los cuales son insertados directamente en el cerebro mediante cirugía.

La electrocorticografía (ECoG) mide la actividad eléctrica del córtex cerebral mediante electrodos sobre la superficie cerebral, los electrodos pueden ser colocados fuera de la duramadre (epidurales) o bajo ella (subdurales), con este método se obtienen las señales del cerebro de mayor calidad, pero después de un tiempo la estabilidad de las señales puede perderse por la generación de cicatriz en la corteza, impidiendo la lectura clara de las señales.

La grabación intracortical neuronal es una técnica de mapeo neuronal que mide la actividad eléctrica cerebral, requiere de la implantación de arreglos de micro electrodos dentro de la corteza para capturar señales pico y potenciales de campo de las neuronas. Las señales obtenidas mediante esta técnica son: actividad mono-modo (SUA, del inglés single-unit activity), actividad multi-modo (MUA, del inglés multi-unit activity) y potencial de campo local (LFP, del inglés local field potential). Las grabaciones intracorticales tienen una resolución espacial y temporal que los EEG, aunque también puede perder calidad por la generación de cicatrices en la corteza.

- **Métodos parcialmente invasivos**
  Existen otros equipos que pueden capturar las señales cerebrales, son parcialmente invasivos por que requieren ser colocados en el cráneo, sin embargo, pese a obtener señales de menor calidad que las obtenidas por los métodos invasivos, son menos probables a verse afectados por la formación de cicatrices.

- **Métodos no invasivos**
  Estos métodos son los más comunes y de menor costo, pero son los más débiles en cuanto a la adquisición de señales. La detección de señales se da mediante la colocación de electrodos en el cuero cabelludo, tienen una mejor resolución temporal y puede llegar a usar hasta 256 electrodos colocados en todo el cuero cabelludo. Estos equipos utilizan las resonancias magnéticas (fMRI), tomografía positrón electrón (PET), magnetoencefalorama (MEG) y tomografía computarizada de emisión de fotones individualizada (SPECT).

  La resonancia magnética detecta los cambios de volumen sanguíneo en el cerebro, el flujo sanguíneo cerebral y los niveles de oxigenación durante la activación neuronal mediante los campos electromagnéticos emitidos. Posee una alta resolución espacial lo que facilita la lectura de las regiones activas en el cerebro, pero también tiene una baja resolución temporal de 1 a 2 segundos y es altamente susceptible al movimiento de la cabeza [16].

5.2.4. Emotiv Epoc +14

Es una diadema que se conecta mediante bluetooth al ordenador, cuenta con 14 canales de lectura más 2 de referencia para indicar el posicionamiento de la misma. A diferencia de otros sistemas EEG dedicados a la captura de datos, los sensores funcionan a base de solución salina. *Figura 5.7* [17]. Cuenta con su propio SDK para el desarrollo de BCIs sobre diferentes sistemas operativos. Además, a diferencia de otros dispositivos existentes para el desarrollo de este timpo de aplicaciones, cuenta con un sensor giroscópico, mediante el cuál se puede mantener y/o cambiar la posición permitiendo que la diadema este estable al hacer fuerzas compensatorias.

Figura 5.6: Emotiv Epoc +14

.

5.3. Extracción de Características

5.3.1. Definición

De manera general, la extracción de características permite que se determinen las características necesarias y específicas de una señal. Debido a que la actividad neuronal es bastante compleja, se discriminan únicamente aquellas características que son de interés para algún propósito específico; creando así vectores de propiedades que caractericen a dicha señal. [18].

5.3.2. Transformada Wavelet Discreta

Una wavelet es una señal oscilatoria de energía finita en un determinado intervalo de tiempo. En el campo del análisis de datos una Transformada Wavelet Discreta (DWT por sus siglas en inglés). El análisis wavelet para señales discretas utiliza una familia de wavelets orto-normales, es decir, que las wavelets son ortogonales y normalizadas [19].

En la mayoría de las señales son las componentes de baja frecuencia las que le otorgan a la señal la mayor parte de su información, o bien, le dan una especie de identidad a la señal. Mientras que las componentes de alta frecuencia se encargan de incorporar características más particulares. Es por ello que se subdividen las componentes de una señal en dos categorías:

- Aproximaciones (baja frecuencia)
- Detalles (alta frecuencia)

Luego, surge la idea de separar estas dos componentes a través de filtros como se muestra en la *Figura* **??**. Donde S es la señal que se desea analizar, A la salida del pasa-bajos y D la salida del filtro pasa-altos. Naturalmente, los filtros son diseñados de tal manera que sean complementarios, es decir, la suma de A y D debe ser S.

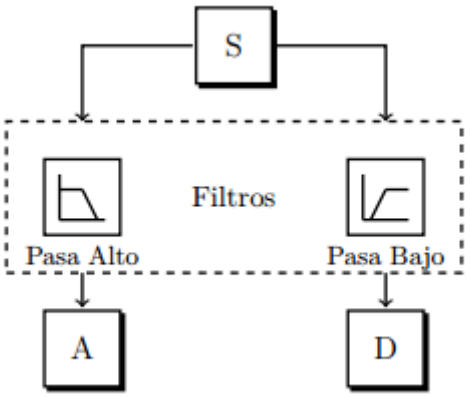

Figura 5.7: Diagrama de descomposición de señales usando bancos de filtros.

Algebraicamente, el objetivo de aplicar la DWT a un vector es obtener un vector transformado que tiene en la mitad, conocida como parte alta, la misma información de alta frecuencia que el vector original y en otra mitad, conocida como parte baja, la información de baja frecuencia.

El éxito de la DWT en el procesamiento de señales reside en el hecho de que usualmente la mayor parte de la información de alta frecuencia es relativamente pequeña y puede descartarse, permitiendo así una compresión eficiente de los datos. Para poder completar esta operación la DWT debe ser invertible además de ser fácil y rápida de calcular $O(n)$ operaciones para un vector que pertenezca a R^n El algoritmo 1 muestra la implementación de la transformada wavelet discreta sobre un vector. Esta implementación se conoce como transformada rápida de wavelet o DWT. [20]

Algoritmo 1 Transformada rápida de k niveles wavelet de un vector de longitud N

n := N
for $l := 1$ to k **do**
  for $i := 0$ to $\frac{n}{2} - 1$ **do**
    for $j := 0$ to $m - 1$ **do**
      $w_i = w_i + h_j v_{(2i+j)_n}$
      $w_i + \frac{n}{2} = w_i + g_j v_{ji+jn}$
    end for
  end for
  $v_{0:n-1} = w$
  $n = \frac{n}{2}$
end for

5.3.3. Análisis de Componentes Independientes

El análisis de componentes independientes (ICA por sus siglas en inglés) asume que se observan d señales $x_i(t)$, $i = 1, 2, 3, \ldots, d$, $t = 1, 2, 3, \ldots, n$, generadas mediante la mezcla de un conjunto de b señales estadísticamente independientes $s_i(t)$, $i = 1, 2, 3, \ldots, b$, $t = 1, 2, 3, \ldots, n$. Sea X la matriz de datos a analizar, compuesta de d columnas y n renglones, cada columna corresponde a una de las d señales observadas, y cada renglón a una de las n muestras. El proceso de generar X se puede describir como $X = SA$, donde S es una matriz cuyas columnas corresponden a las b señales independientes, y A es la matriz de mezcla que contiene los coeficientes a_{ji} tal que

$$x_i(t) = a_{1,i}s_1(t) + a_{2,i}s_2(t) + a_{3,i}s_3(t) + \cdots + a_{k,i}s_k(t) + \cdots + a_{b,i}s_b(t).$$

Dado el modelo anterior, la tarea consiste en estimar A y S a partir de X. La mayoría de los algoritmos que realizan ICA estiman los coeficientes de la matriz $W = A^{-1}$ mediante un proceso iterativo que busca maximizar la no-Gausianidad de los componentes

$$\hat{s}_i(t) = w_{1,i}x_1(t) + w_{2,i}x_2(t) + w_{3,i}x_3(t) + \cdots + w_{k,i}x_k(t) + \cdots + w_{d,i}x_d(t)$$

o minimizar su información mutua. El algoritmo más popular que implementa ICA fué desarrollado por Aapo Hyvärinen [21] y recibe el nombre de FastICA. Como la mayoría de los algoritmos que implementan ICA, FastICA busca una rotación ortogonal de los datos X previamente blanqueados, mediante un esquema iterativo de punto fijo, el cual maximiza la no-Gausianidad de los componentes. La métrica de no-Gausianidad sirve como una representación de la condición de independencia estadística, la cual requiere datos infinitos para ser verificada.

El procedimiento de blanqueo sobre X consiste en restar de cada una de las filas en X la media $\mu = [\mu_1\, \mu_2\, \mu_3\, \ldots\, \mu_d]$, $\mu_k = (1/n)\sum_{i=1}^{n} x_{ik}$, para después aplicar una transformación lineal $\tilde{X} = \bar{X}M$ que decorrelacione las columnas y haga que cada $\tilde{x}_i(t)$ tenga varianza unitaria. La matriz M se obtiene de los vectores y valores propios de la matriz de covarianza de $\bar{X}$. Sea E la matriz donde cada columna $e_j \in \mathbb{R}^d, j = 1, 2, 3, \ldots, d$ corresponde a un vector propio, y D la matriz diagonal con el valor propio asociado al j-esimo vector propio en el elemento d_{jj}, entonces la transformación lineal esta dada por $M = ED^{-1/2}E^T$. Con los datos blanqueados, se procede a estimar cada una de las columnas de W mediante el Algoritmo 2, donde $g(u) = \tanh(u)$ y $g'(u) = 1 - \tanh^2(u)$.

5.4. Clasificación de señales EEG

5.4.1. Definición

El proceso de clasificación se enfoca en determinar cuales son las características que definen a cierto grupo de datos para poder ser clasificados como propios de una clase. Al poder definir los datos dentro de una clase específica, es posible entender la naturaleza del comportamiento que estos tuvieron al generarse y así al tener nuevos datos, será posible predecir a cuál de las clases ya definidas pertenece.

Algoritmo 2 FastICA

Entrada: $\tilde{X} \in \mathbb{R}^{n \times d}$, $b \in \mathbb{Z}^+ \leq n$
Salida: $W \in \mathbb{R}^{d\ timesb}$, $\hat{S} \in \mathbb{R}^{n \times b}$

```
for j = 1 to b do
    w_j(0) ← vector aleatorio
    t = 0;
    repeat
        t = t + 1
        w_j(t) = (1/n) ( X^T g (X w_j(t-1)) - w_j(t-1) 1\!\!1^T g' (X w_j(t-1)) )
        w_j(t) = w_j(t) - Σ_{k=1}^{j-1} w_j^T(t) w_k w_k
        w_j(t) = w_j(t) / ||w_j(t)||
    until w_j(t) = w_j(t-1)
end for
```

5.4.2. Clasificador Bayesiano

Es un método basado en la teoría de la probabilidad, usa frecuencias para calcular probabilidades condicionales para calcular predicciones sobre nuevos casos. Naïve Bayes es una técnica tanto predictiva como descriptiva. [22]

El clasificador Naïve Bayes establece que: Sean E y F eventos. Podemos expresar a E como:

$$E = EF \cup EF^c$$

Debido a que EF y EF_c son mutuamente excluyentes, tenemos que

$$P(E) = P(EF) + P(EF^c) = P(E|F)P(F) + P(E|F^c)(1 - P(F))$$

La ecuación anterior establece que la probabilidad del evento E es una ponderación de la probabilidad condicional de E dado que F ha ocurrido y la probabilidad condicional del evento E dado que F no ha ocurrido. Cada probabilidad condicional proporciona tanta ponderación como el evento condicionado tiende a ocurrir.

De la definición de probabilidad condicional tenemos que:

$$P(EF_i = P(E|F_i)P(F_i)$$

Además, usando el hecho de que los eventos $EF_i, i = 1, ..., n$ son mutuamente excluyentes, obtenemos que

$$P(E) \sum_{i=1}^{n} = P(E|F_i)P(F_i)$$

Así, la ecuación anterior muestra como, para eventos dados $F_1, F_2, ...F_n$ de los cuales uno y solamente uno puede ocurrir, se puede calcular $P(E)$ condicionando a que ocurra F_1,. Esto es,

se establece que $P(E)$ es igual al promedio de las ponderaciones de $P(E|F_i)$ y cada término es ponderado por la probabilidad del evento en el cual es condicionado. Supóngase ahora que E ha ocurrido y que se quiere determinar la probabilidad de que el evento F_j haya ocurrido. Entonces se tiene:

$$P(E_j) = \frac{P(EF_j)}{P(E)} = \frac{P(E|F_j)P(F_j)}{\sum_{i=1}^{n} P(E|F_i)P(F_i)}$$

Esta última ecuación es conocida como la fórmula de Bayes. Así, podemos considerar a E como evidencia de F, y calcular la probabilidad de que F_j ocurra dada la evidencia. La suposición que da origen al adjetivo Naïve (ingenuo) es la independencia entre las variables, lo cual no es siempre cierto. Sin embargo, el método ha sido exitoso en su aplicación debido a que la información relevante está contenida en las magnitudes relativas entre las cantidades y no tanto en los valores de las probabilidades en sí.[22]

6. Desarrollo del Proyecto

6.1. Diseño de la Interfaz

El desarrollo de la interfaz se realizó en Visual Studio 2013. En una computadora portátil con procesador Intel Core i7-4700MQ. Tanto la interfaz como los algoritmos de control interno están diseñados en C++. Para poder ejecutar la aplicación se requiere que todos los archivos **.cpp, *.h* y **.resx* que se van añadiendo estén en la carpeta *EEGRECORD* contenida en el proyecto. Asímismo, las imágenes que están incluidas como recursos dentro del proyecto deberán estar en la carpeta *EEGRecord->Images* del proyecto. Adicionalmente todas las librerías utilizadas del SDK que proporciona la diadema están contenidas en la carpeta *emotiv*. La dependencia de librerías se hace de manera automática en el proyecto. Basta con compilar el proyecto completo con Visual Studio. En dado caso de que se desee agregar nuevas librerías deberan guardarse en las carpetas anteriormente mencionadas y hacer el repectivo *"#include"* de dichos archivos en donde sea que se vayan a utilizar.

6.1.1. Pantalla de Inicio

La interfaz cuenta con varias pantallas de navegación; la primera permite navegar hacia los diferentes módulos con los que cuenta el sistema completo, como se muestra en la *Figura 6.1*.

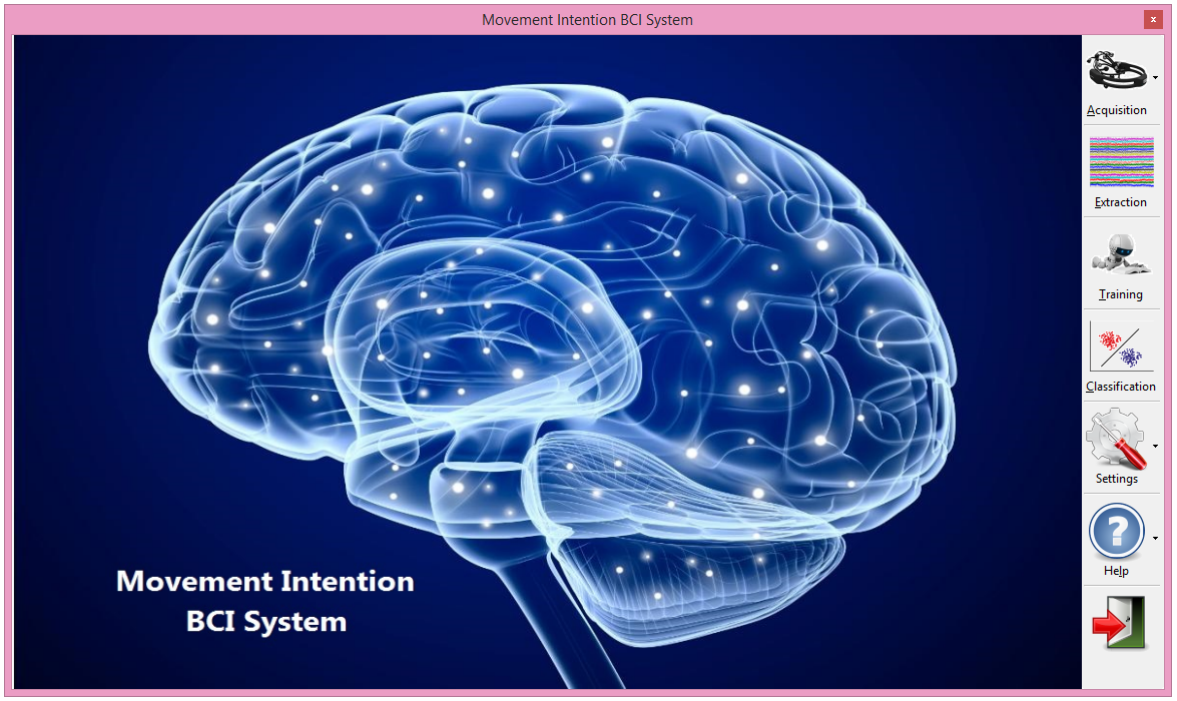

Figura 6.1: Pantalla Principal.

Se cuenta con un panel de navegación con las siguientes opciones:

- Adquisición - Permite realizar la extracción de señales EEG en dos modalidades: En línea (tiempo real) y fuera de línea. Además Permite ver el estado de conexión de cada uno de los electrodos que conforman la diadema así como visualizar las señales leídas de manera gráfica.
- Extracción - Permite seleccionar el método de extracción de características que se aplicará a la señales obtenidas.
- Entrenamiento - Permite la caracterización de cada movimiento y el ingreso de datos al modelo.
- Clasificación - Implementación del clasificador bayesiano para la evaluación de las clases.
- Ajustes - Permite cambiar el idioma en el que se presenta la interfaz así como el directorio donde se almacenarán los archivos que conformarán la base de datos que del sistema.
- Salir - Detiene le ejecución del sistema.

6.1.2. Adquisición de señales

Para poder realizar la adquisición de señales, se creó una pantalla dentro de la interfaz que obligará al usuario a verificar el estado de los electrodos antes de poder hacer la lectura de las señales. Para esto, se creó un menú emergente del módulo de adquisición que inhabilitará las demás opciones como se muestra en la *Figura 6.2*.

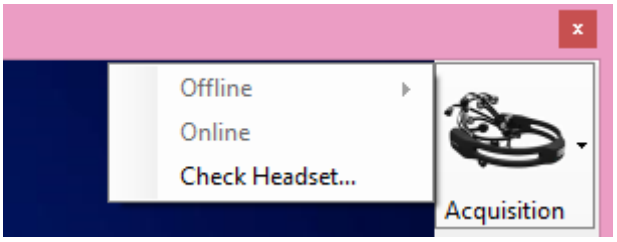

Figura 6.2: Deshabilitación de menú para la adquisición

Es necesario que el usuario revise el estado de la diadema. Al dar click en esta opción, se abrirá la ventana que se muestra en la *Figura 6.3*. En la pantalla se mostrará la intensidad de señal de cada uno de los electrodos, ésta cambia de color dependiendo de su estado:

- Negro - No hay señal.
- Rojo - Muy mala calidad de señal
- Amarillo - Calidad de señal muy pobre
- Naranja - Calidad de señal faorable
- Verde - Buena señal

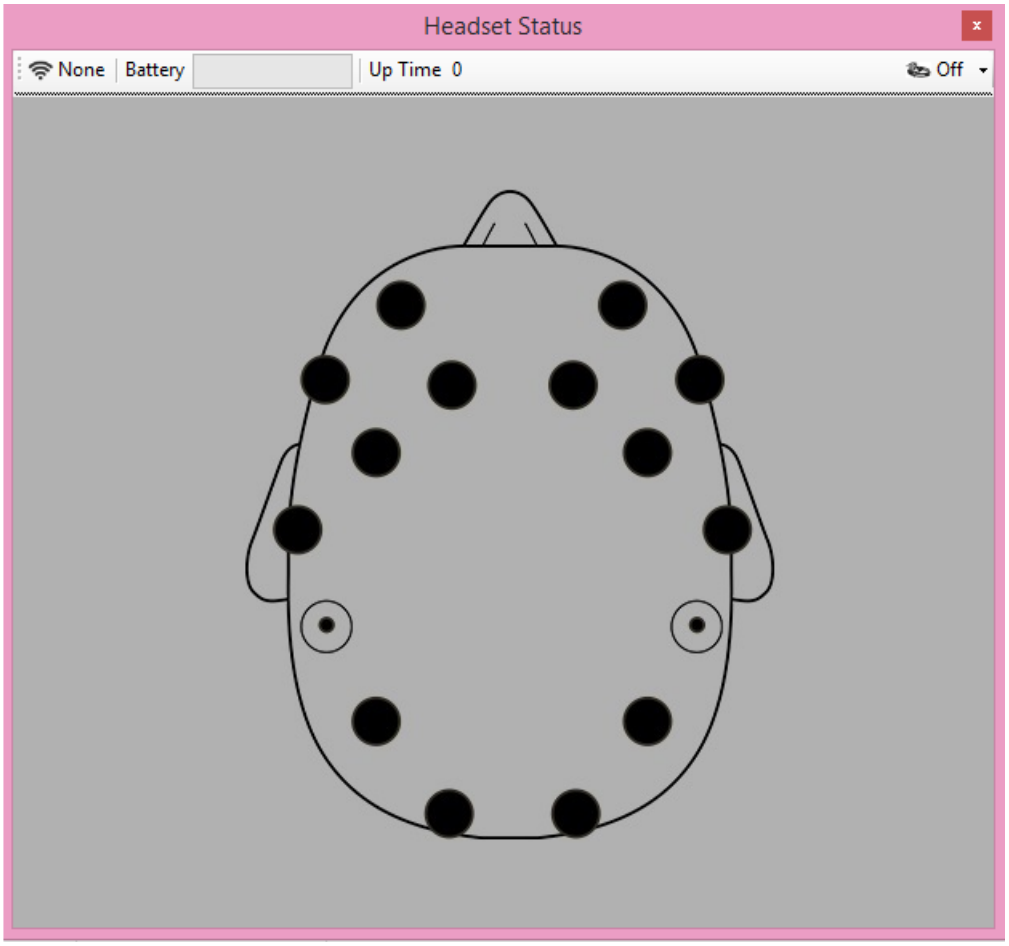

Figura 6.3: Estado de la diadema

Además se muestran datos básicos como el nivel de bateria de la diadema, el tiempo que ha estado conectada,y la intensidad de señal que hay de la diadema hacia la computadora a la que está conectada. Una vez que se haya comprobado su estado, en la pantalla principal se habilitarán las opciones 'En línea' y 'Fuera de línea' como se muestra en la *Figura 6.4*.

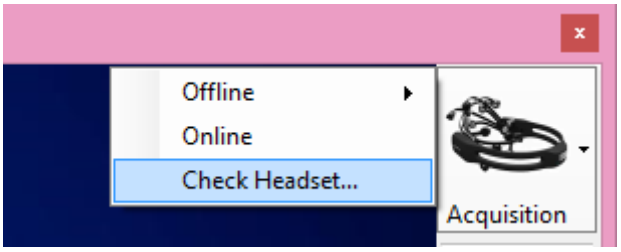

Figura 6.4: Habilitación de menú para la adquisición.

La adquisición fuera de línea es el primer paso a realizar puesto que aquí se concentrarán los archivos que contiene la lectura de los electrodos, mismos que serán usados posteriormente para la clasificación y la adquisición en tiempo real. El procedimiento de adquisición se lleva a cabo en primera instancia mediante la configuración de diversos parámetros como se muestra en la *Figura 6.5*.

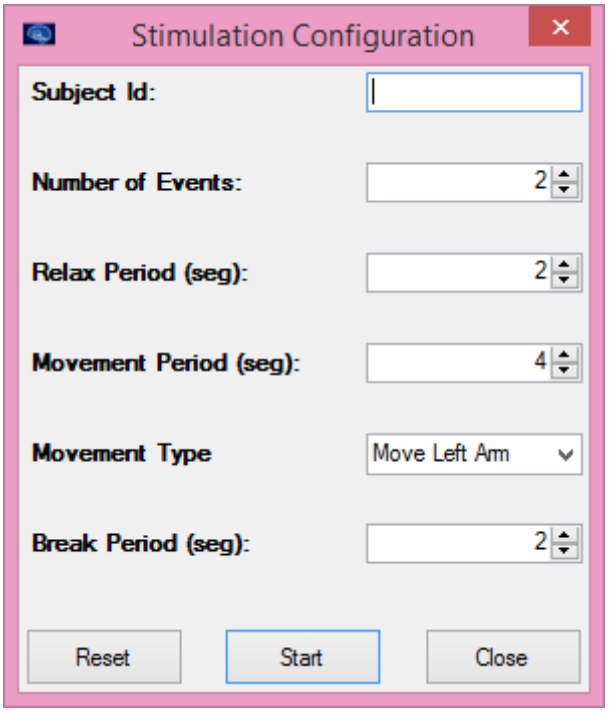

Figura 6.5: Configuración de parámetros para estímulos

Los parámetros a configurar en dicha interfaz son:

- Subject id: Permiten asignarle un id/nombre al individuo que hará uso de la diadema para la adquisición de señales.
- Número de eventos: Permite configurar el número de eventos seguidos que se realizarán durante la fase del estímulo.
- Periodo de relajación: Configura el tiempo que se mostrarán las imágenes que representan el estímulo para la relajación.
- Tiempo de movimiento: Permite seleccionar el tiempo que se mostrarán las imágenes que representan el estímulo para la intención del movimiento deseado.
- Tipo de movimiento: Permite elegir si se mostrarán estímulos visuales para el movimiento de brazo izquierdo o de pierna derecha.
- Tiempo de descanso: Permite configurar el tiempo de descanso que tendrá el sujeto entre cada evento.

Con el botón de 'Reiniciar(reset)' todos los campos regresarán a sus valores predeterminados en caso de querer volver a configurarlos. Con el botón 'Cerrar (close)' se cerrará la ventana. Finalmente, con el botón de 'Inicio (start) se comenzarán a presentar las imágenes que representan cada uno de las fases del evento de acuerdo a los tiempos que se configuraron como se indicó previamente.

Para los estímulos que representan relajación se usaron como base las imágenes que se muestran en la *Figura 6.6*.

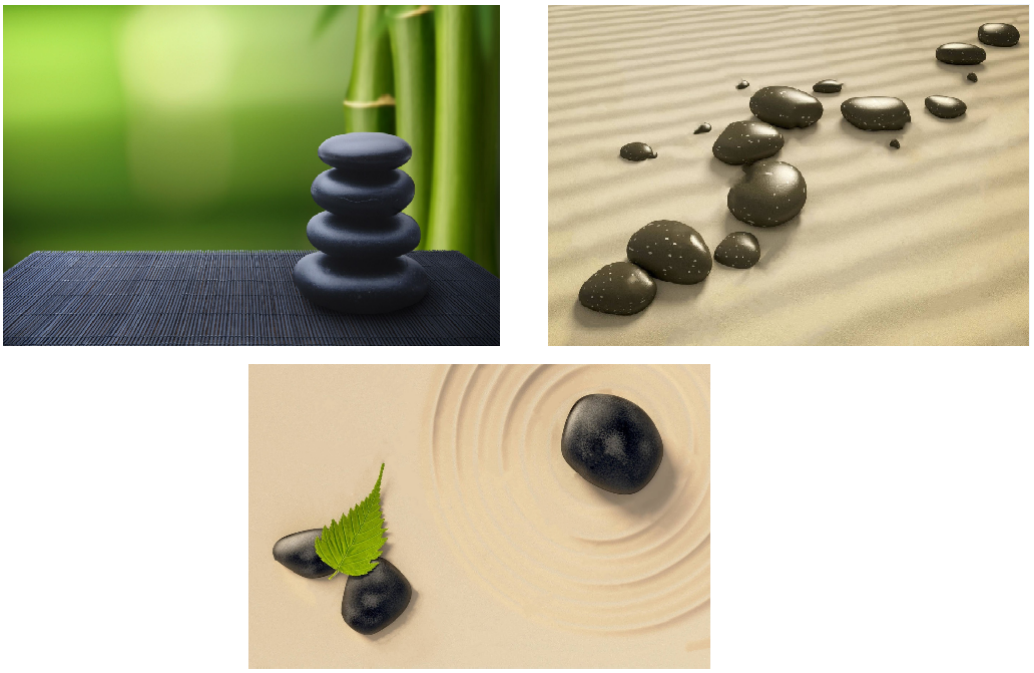

Figura 6.6: Estímulos visuales para relajación

Para los estímulos que representan intención de movimiento se utilizaron las secuencias de imágenes que se muestran en la *Figura 6.7* y *Figura 6.8*.

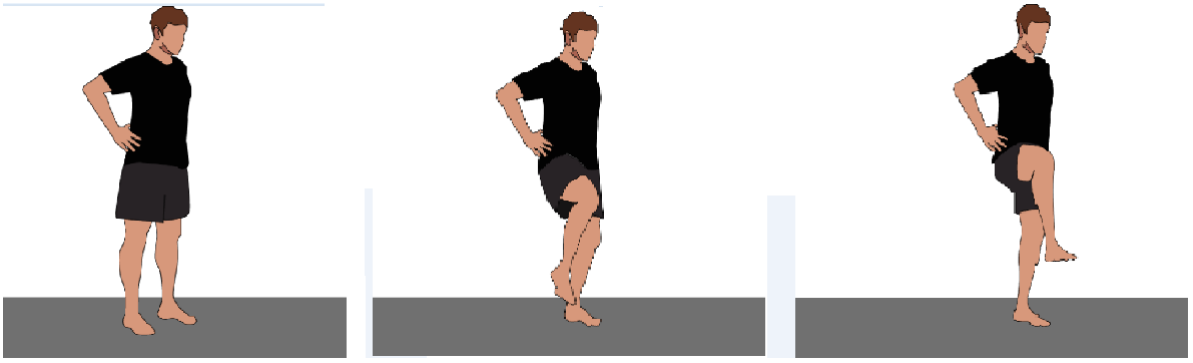

Figura 6.7: Estímulo para intención de movimiento de pierna derecha

Figura 6.8: Estímulo para intención de movimiento de brazo izquierdo

Además, dentro de la adquisición fuera de línea se implementó una pantalla que permite visualizar de manera gráfica las señales extraídas de la diadema como se muestra en la *Figura 6.9*. Dentro de la pantalla se implementó un buscador donde se visualizan los diferentes archivos que se han creado, mismos que corresponden a los diferentes eventos que se han capturado. Asimismo, se cuenta con la opción de visualizar canales específicos en caso de que no se desee ver todos en su representación gráfica.

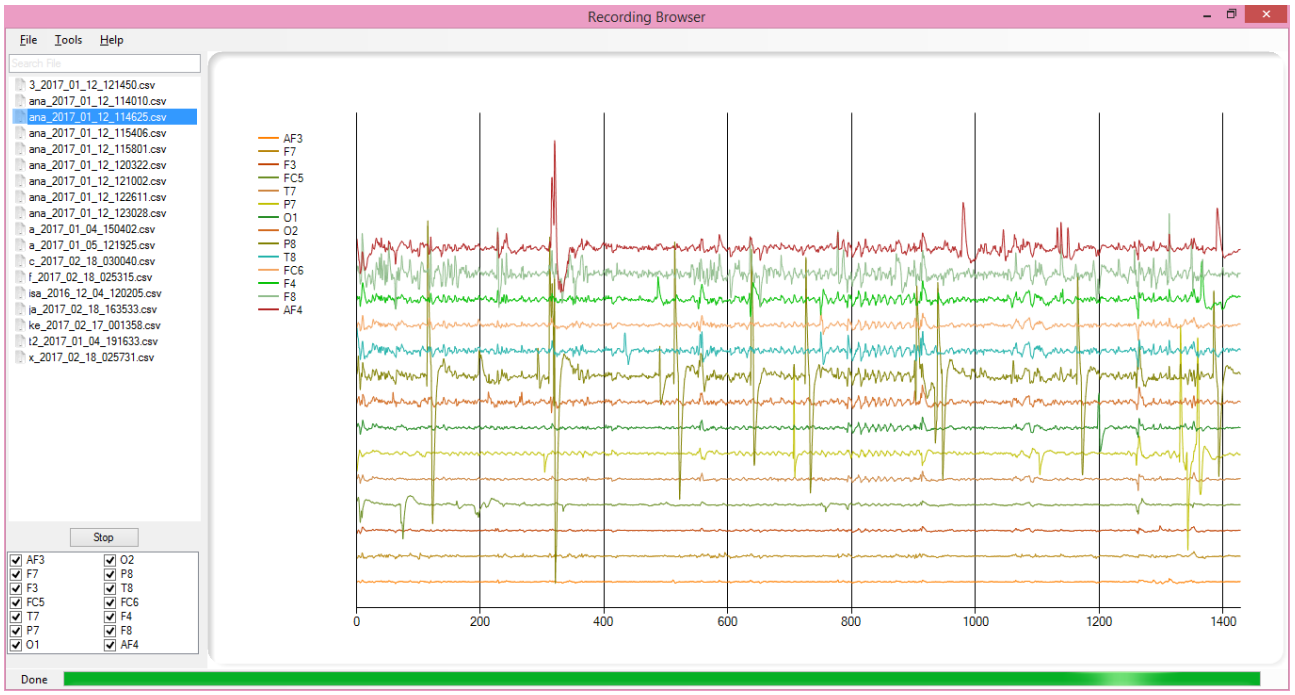

Figura 6.9: Buscador para visualización de señales EEG adquiridas.

6.1.3. Extracción de características

Para el módulo de extracción de características se implementó una pantalla en la que se puede escoger que método de extracción se quiere aplicar a las señales obtenidas, así como el canal que se quiere visualizar; también cuenta con un panel de búsqueda dividido (*Figura 6.10*). En el primero se muestran los archivos que contienen las señales a las que se les aplicará el método escogido, mientras que en la parte de abajo se muestran los archivos que se van generando una vez que se aplicó dicho método. Éstos son los archivos que se usarán para desplegar en pantalla las gráficas correspientes a las bandas de beta y mu de cada uno de los canales que se procesaron.

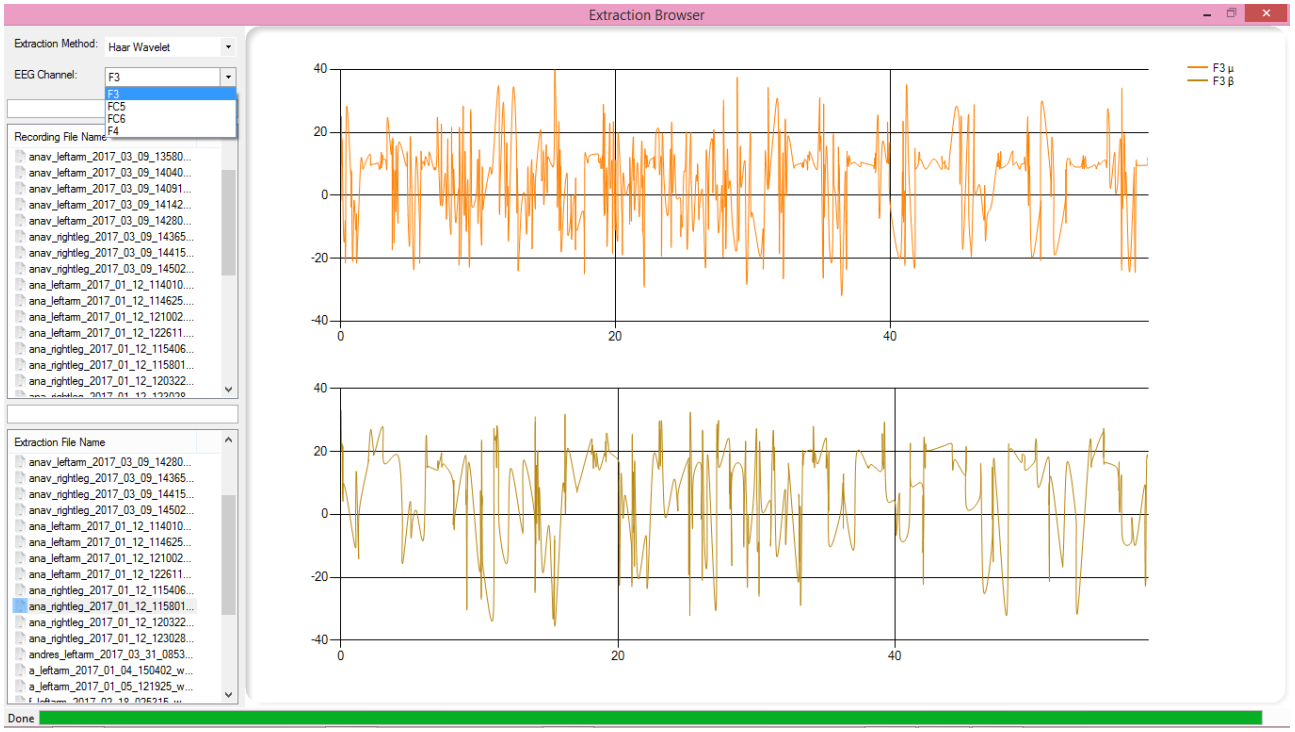

Figura 6.10: Visualización de señales al aplicar los métodos de extracción de características.

6.1.4. Entrenamiento

Para la ventana de entrenamiento se hizo un panel dividido donde se filtran todos los archivos que constituyen la base de datos. En primera instancia se selecciona el id del sujeto, después el tipo de movimiento y finalmente el método de características usado. De esta forma, se muestran en pantalla los archivos que contenga la información correspondiente a lo previamente seleccionado. Se cuenta con botones de selección para escoger los archivos de manera más rápida y poder hacer el entrenamiento mediante el botón 'Train'. Al hacer el paso del entrenamiento se muestra en pantalla el vector de medias de los 4 canales así como la matriz de varianza-covarianza resultante, mismos que fueron obtenidos al procesar todos los archivos seleccionados en el panel izquierdo tal como se muestra en la *Figura 6.11.*

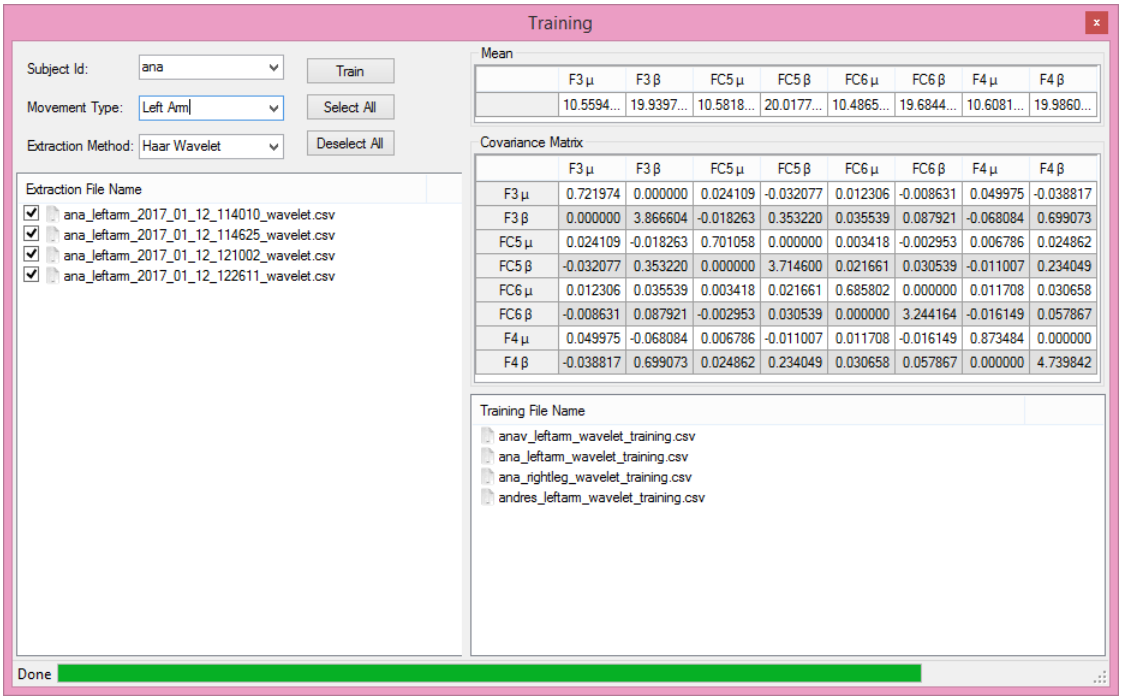

Figura 6.11: Pantalla de módulo de entrenamiento.

6.2. Compilación de la Interfaz

Para poder compilar el proyecto, se requiere instalar el programa Microsoft Visual Studio 2013. Una vez instalado, se abrirá la ventana que se muestra en la *Figura 6.12*.

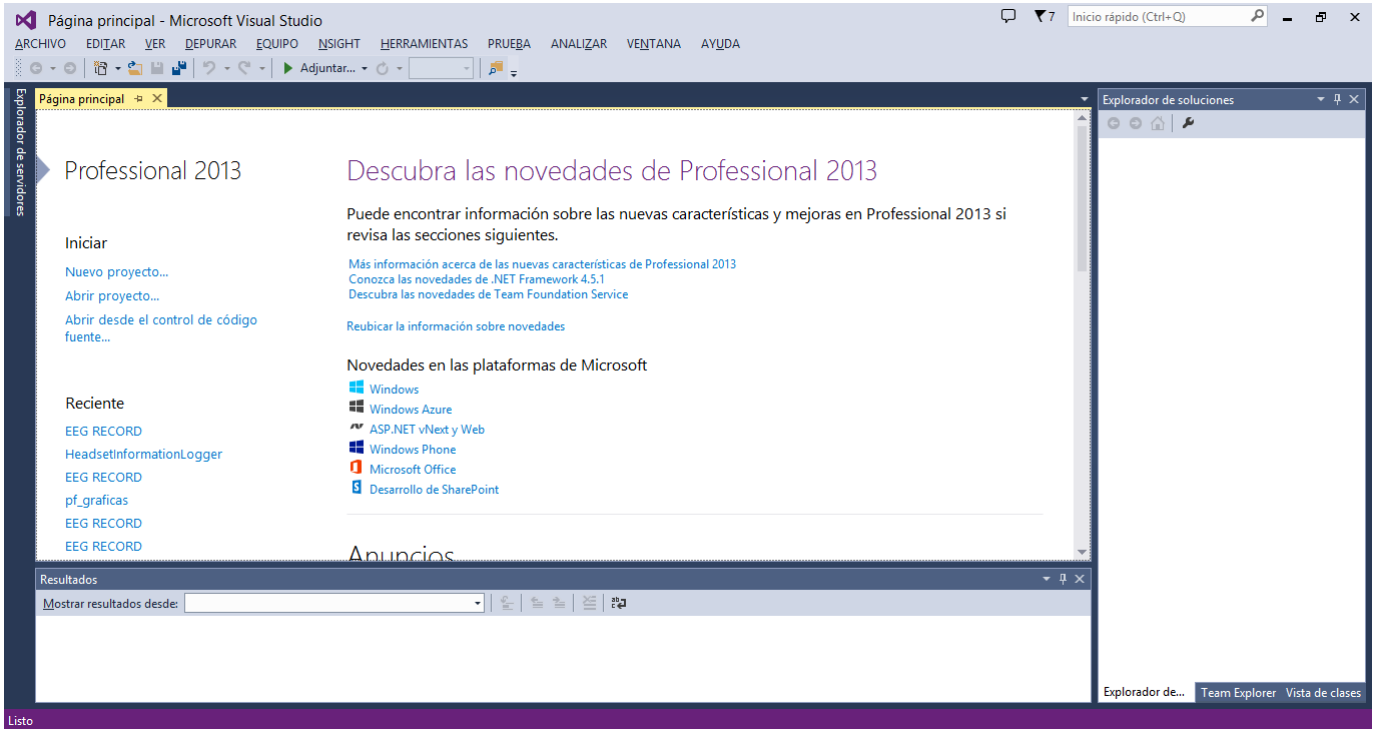

Figura 6.12: Pantalla principal de Visual Studio.

Posteriormente, se selecciona en la barra de herramientas superior (*Figura 6.13*) la opción *Archivo* → *Abrir* → *Proyecto o solución*

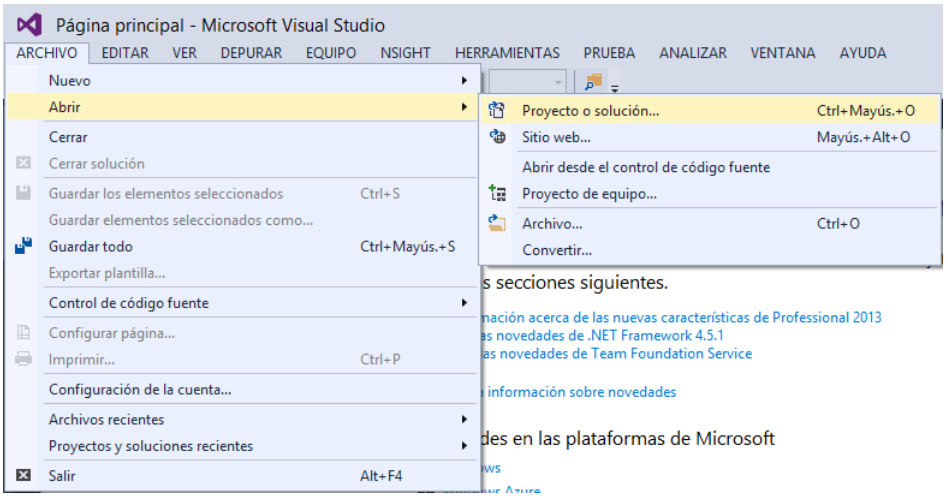

Figura 6.13: Opciones para abrir el proyecto.

Al hacer esto se abrirá una ventana donde abrá que localizar donde está contenida la carpeta *EEG RECORD* que es done se encuentran todos los archivos del proyecto como se muestra en la *Figura 6.14*.

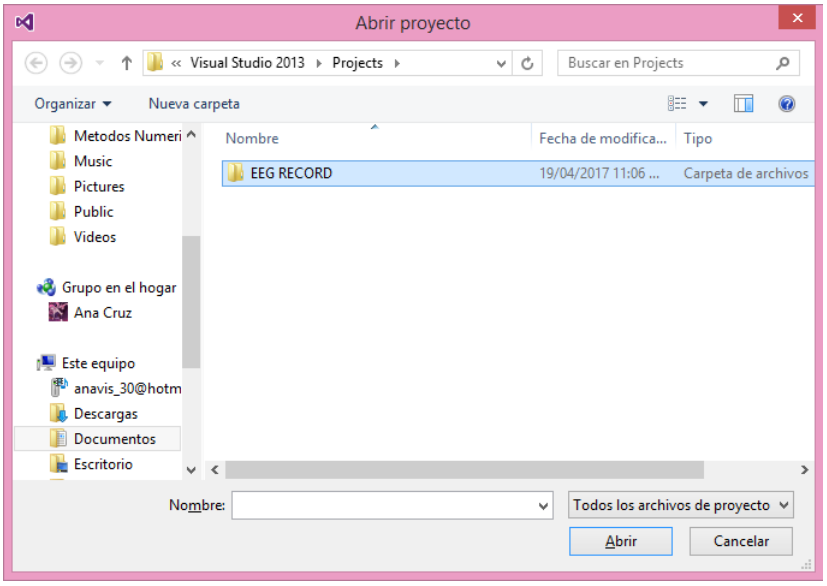

Figura 6.14: Ventana de búsqueda del proyecto.

Una vez localizada la ruta del proyecto, abriremos la carpeta que la contiene. Dentro de está carpeta ubicaremos el archivo *EEG RECORD.sln*. Lo seleccionaremos y daremos click en la opción *Abrir* (*Figura 6.15*).

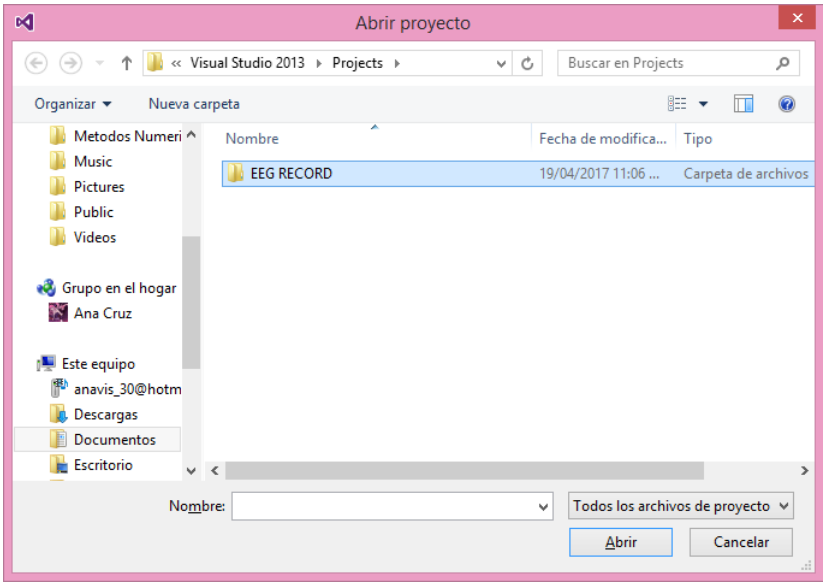

Figura 6.15: Ventana de búsqueda del proyecto.

Al hacer esto, se desplagará un explorador de soluciones en la pantalla (*Figura 6.16*) donde aparecerán todos los archivos que están dentro de la carpeta del proyecto.

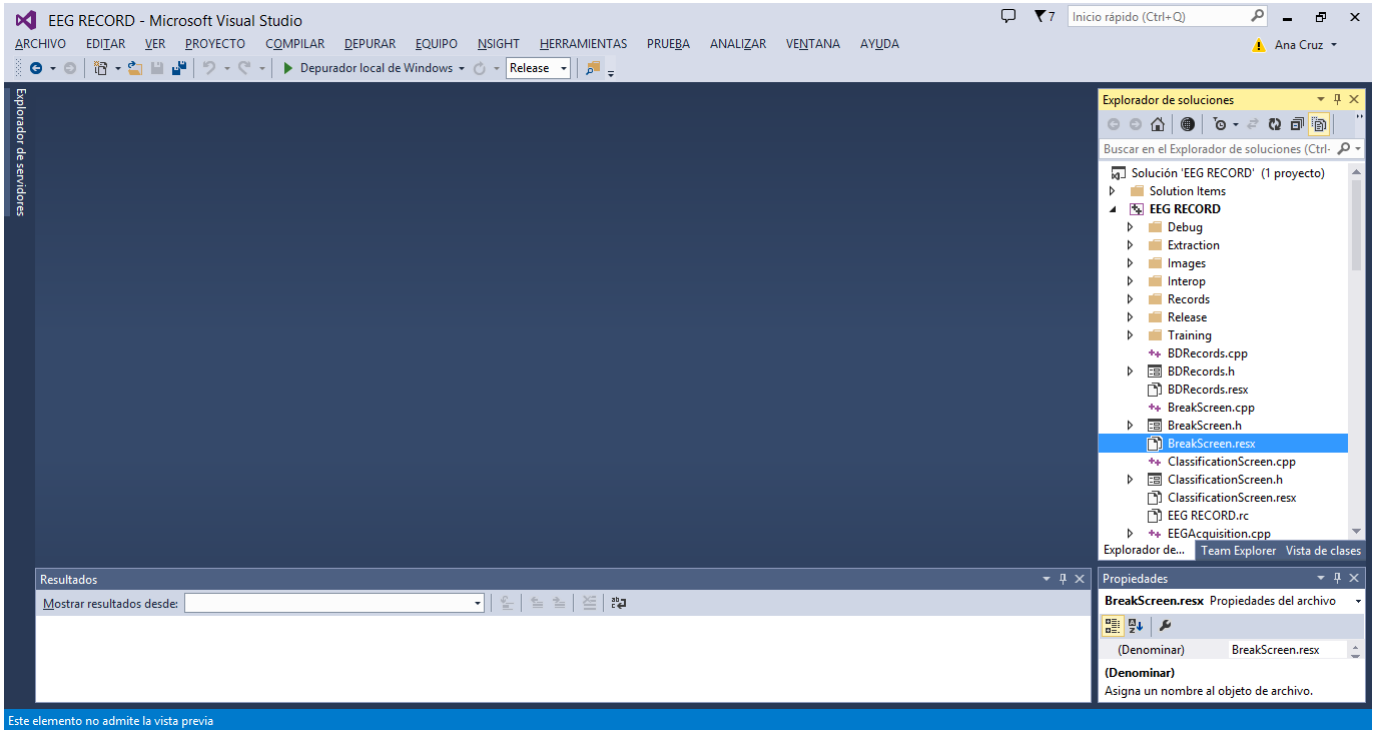

Figura 6.16: Proyecto abierto en Visual Studio.

Para poder compilar todos los archivos basta con seleccionar en el explorador de soluciones, el proyecto y en la barra de herramientas superior de Visual Studio verificar que las opciones remarcadas estén como en la *Figura 6.17* y hacer click en *Depurador local de Windows*

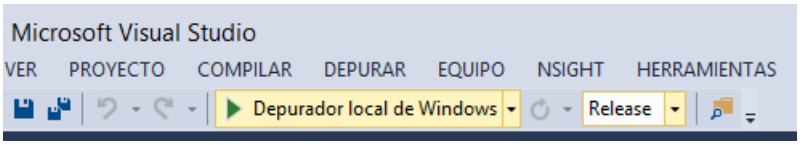

Figura 6.17: Opciones para lanzamiento de la interfaz.

Haciendo esto, aparecerá la ventana mostrada en la *Figura 6.18* donde se hace la confirmación para poder compilar el proyecto.

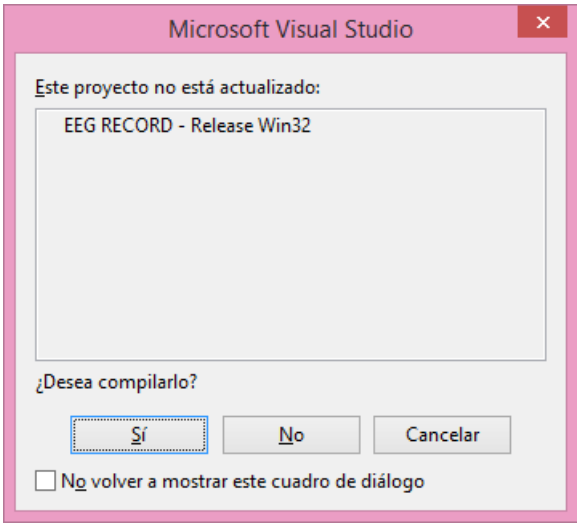

Figura 6.18: Confirmación para compilar el proyecto.

Finalmente, el proyecto comenzará a compilar los archivos contenido en el proyecto (*Figura 6.19*) y en la parte inferior de la pantalla, en la ventana de Resultados se podrá observar el proceso de compilación de todos los archivos. Al finalizar, se lanzará la aplicación y aparecerá la ventana de inicio descrita en la sección anterior.

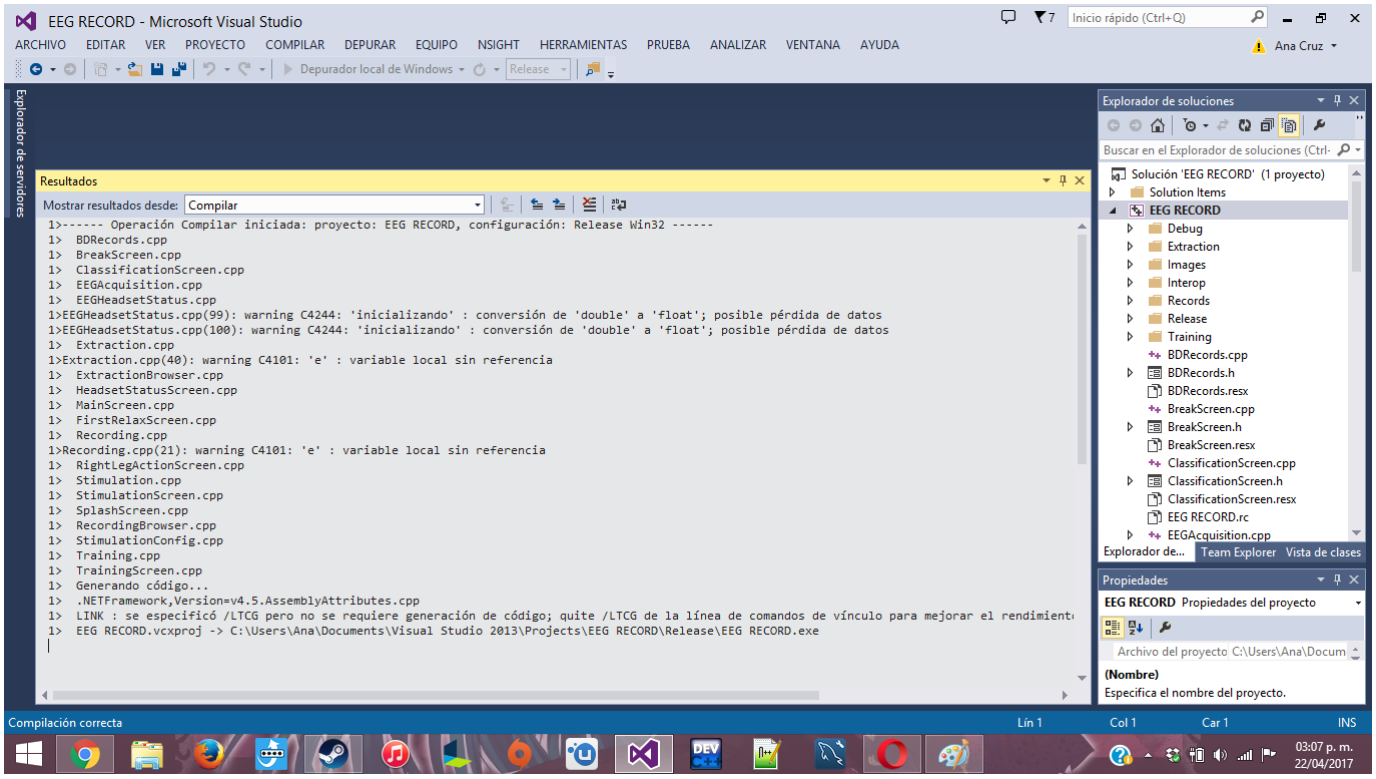

Figura 6.19: Proceso de compilación

6.3. Descripción técnica

6.3.1. Estado de la diadema

Para poder hacer la conexión de la diadema a la propia interfaz fue necesario hacer uso del SDK que proporciona el fabricante Emotiv, para obtener tanto el estado de la diadema así como poder hacer la lectura de datos de cada uno de los canales. Para poder verificar la conexión de los electrodos así como el resto de características de la diadema se adaptó la siguiente porción de código a la interfaz:

Código 6.1: Verificación del estado de la diadema del archivo EEGHeadsetStatus.cpp en líneas 125 a 133

```
if (readytocollect && onStateChanged)
{
    nStateChanged = false;
    systemUpTime = IS_GetTimeFromStart(eState);
    wirelessStrength = IS_GetWirelessSignalStatus(eState);
    if (wirelessStrength != NO_SIG)
    {
        Is_GetBatteryChargeLevel(eState, &batteryLevel,&maxBatteryLevel);
    }
}
```

Dicho código verifica que la diadema esté conectada y lista para recolectar datos mediante las variables booleanas readytocollect y onStateChanged; si se detecta que la diadema está conectada y los electrodos están conectados ambas son verdaderas con lo que se puede proceder a verificar el estado de la bateria y la señal. Adicionalmente, dentro de está misma condición, se verifican en tiempo real el estado de cada uno de los electrodos. El SDK de la diadema cuenta con sus propias variables que hacen referencia a cada uno de los electrodos, mismas que se guardaron en una lista para poder hacer uso de ellas:

Código 6.2: Selección de colores para mapeo de electrodos del archivo EEGHeadseatStatus.h en líneas 37-42

```
list<EE_InputChannels_t> contactChannelList = {{ EE_CHAN_CMS, EE_CHAN_DRL
    ,
    EE_CHAN_AF3, EE_CHAN_F7,EE_CHAN_F3, EE_CHAN_FC5, EE_CHAN_T7,
        EE_CHAN_P7,
    EE_CHAN_O1, EE_CHAN_O2, EE_CHAN_P8, EE_CHAN_T8, EE_CHAN_FC6,
        EE_CHAN_F4,
    EE_CHAN_F8,EE_CHAN_AF4
}};
```

Así, únicamente fue necesario tener una imagen donde se muestre la posición de cada uno de los electrodos para poder mapear a ésta dichas variables para poder hacer el cambio de color de acuerdo a la intensidad de señal mediante el siguiente algoritmo,donde se obtiene la intensidad de señal y se redibuja lo que está en las coordenadas de cada uno de ellos.

Código 6.3: Selección de colores para mapeo de electrodos del archivo EEGHeadsetStatus.cpp en líneas 64 a 94

```
SolidBrush HeadsetStatus::getContactQuality(int channel)
{
    channelIterator = contactQualityMap.find(channel);
    if (channelIterator != contactQualityMap.end())
    {
        int quality = channelIterator->second;
        switch (quality)
        {
            case EEG\_CQ\_NO\_SIGNAL:
            brush = gcnew SolidBrush(Color::Black);
            break;
            case EEG_CQ_VERY_BAD:
                                brush = gcnew SolidBrush(Color::Red);
                                break;
                        case EEG_CQ_POOR:
                                brush = gcnew SolidBrush(Color::Yellow);
                                break;
                        case EEG_CQ_FAIR:
                                brush = gcnew SolidBrush(Color::Orange);
                                break;
                        case EEG_CQ_GOOD:
                                brush = gcnew SolidBrush(Color::Green);
                                break;
                }
                if (brush == nullptr) {
                        brush = gcnew SolidBrush(Color::White);
                }
                return brush;
```

```
    }
}
```

6.3.2. Adquisición de datos

Como se explicó anteriormente, la interfaz muestra una serie de estímulos visuales basándose en los tiempos que se configuraron en cada una de las fases del evento. Mientras que la interfaz presenta los estímulos, la diadema está leyendo los datos de los 16 canales y los almacena en un archivo CSV mediante el siguiente algoritmo:

Código 6.4: Adquisición y escritura de datos del archivo EEGAcquisition.cpp en líneas 69 a 92

```
while (readyToCollect)
{
    EE_DataUpdateHandle(0, hData);
    unsigned int nSamplesTaken = 0;
    EE_DataGetNumberOfSample(hData, \&nSamplesTaken);
    if (nSamplesTaken != 0)
    {
        double* data = new double[nSamplesTaken];
        for (int sampleIdx = 0; sampleIdx<(int)nSamplesTaken; ++sampleIdx
           )
        {
            for (std::list<EE_DataChannel_t>::iterator it =
               targetChannelList.begin(); it != targetChannelList.end();
               it++)
            {
                EE\_DataGet(hData, *it, data, nSamplesTaken);
                outputFile << data[sampleIdx] << ",";
            }
            outputFile << marker << "," << eventNbr << std::endl;
        }
        delete[] data;
    }
    eventNbr;
}
```

Mientras la diadema esté en un estado activo (ready to collect) comenzará a adquirir la señales que manda cada uno de los canales y lo mandará al archivo, dicho archivo contiene el número de muestra, los datos extraídos de los canales, un marcador de tiempo, el tipo de movimiento que se configuró en la pantalla de estimulación y el número de evento al que corresponde dicha lectura de datos. (*Figura 6.20*).

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
COUNTER	AF3	F7	F3	FC5	T7	P7	O1	O2	P8	T8	FC6	F4	F8	AF4	TIMESTAMP	MARKER	EVENT_NBR
62	4138.46	4131.28	4129.74	4132.82	4132.82	4127.69	4124.62	4132.31	4138.46	4135.9	4181.54	4903.59	4134.87	4137.95	16.464	firstRelax	1
63	4137.95	4135.9	4127.69	4131.79	4135.38	4130.77	4127.18	4131.79	4137.44	4135.9	4183.08	4907.69	4134.36	4136.92	16.479	firstRelax	1
64	4133.33	4135.9	4131.28	4131.79	4134.36	4130.77	4128.72	4128.72	4135.9	4135.9	4190.77	4910.26	4134.36	4138.46	16.479	firstRelax	1
65	4131.79	4132.31	4134.87	4136.92	4132.82	4127.18	4127.18	4126.67	4137.44	4135.9	4189.74	4907.69	4134.36	4139.49	16.781	firstRelax	1
66	4134.87	4131.28	4132.31	4136.41	4133.85	4127.18	4125.64	4127.69	4136.92	4135.9	4183.59	4903.08	4134.36	4136.41	16.781	firstRelax	1
67	4134.87	4134.36	4132.31	4131.79	4135.38	4126.15	4125.64	4132.31	4135.38	4135.9	4184.62	4902.05	4134.36	4135.9	16.781	firstRelax	1
68	4132.31	4134.87	4130.77	4130.26	4134.36	4124.62	4127.18	4132.31	4136.41	4135.9	4186.15	4904.1	4134.36	4137.95	16.812	firstRelax	1
69	4132.82	4133.33	4125.13	4130.26	4131.79	4125.64	4128.72	4128.72	4136.92	4135.9	4184.62	4910.26	4134.36	4137.95	16.812	firstRelax	1
70	4133.85	4133.33	4123.59	4130.77	4132.82	4127.69	4127.69	4132.31	4135.9	4135.9	4187.18	4912.82	4133.85	4136.92	16.812	firstRelax	1
71	4133.85	4133.85	4127.69	4131.28	4133.33	4129.23	4128.21	4134.36	4136.41	4135.9	4191.79	4912.82	4134.36	4138.97	16.812	firstRelax	1
72	4134.36	4133.85	4131.28	4129.74	4126.15	4130.77	4126.15	4136.41	4138.46	4136.92	4189.74	4914.36	4134.87	4141.03	16.812	firstRelax	1
73	4133.85	4131.79	4131.79	4130.77	4124.1	4131.28	4125.64	4138.46	4137.44	4137.44	4187.18	4912.31	4135.38	4138.46	16.812	firstRelax	1
74	4134.36	4128.21	4136.41	4132.31	4132.82	4128.72	4129.74	4133.33	4135.38	4136.41	4188.21	4907.69	4135.38	4137.44	16.812	firstRelax	1
75	4133.85	4131.28	4138.46	4130.26	4137.44	4128.72	4129.23	4129.74	4136.41	4135.9	4187.18	4906.67	4134.87	4138.46	16.812	firstRelax	1
76	4133.85	4136.41	4136.41	4129.74	4134.87	4129.74	4128.21	4132.82	4135.38	4136.92	4185.64	4908.72	4133.85	4138.97	16.812	firstRelax	1
77	4133.85	4134.87	4131.79	4129.74	4133.85	4129.74	4134.36	4132.82	4134.36	4137.95	4183.59	4910.26	4135.9	4138.46	16.812	firstRelax	1
78	4133.85	4133.33	4130.77	4128.72	4135.38	4127.69	4132.31	4130.26	4132.82	4137.44	4186.67	4913.85	4135.38	4138.46	16.812	firstRelax	1
79	4133.85	4135.38	4132.82	4129.74	4137.44	4127.18	4124.62	4131.79	4130.26	4137.44	4193.33	4912.82	4131.28	4139.49	16.812	firstRelax	1
80	4133.85	4134.36	4134.87	4133.33	4133.85	4129.23	4126.67	4130.77	4132.82	4137.95	4191.28	4909.23	4132.31	4139.49	16.812	firstRelax	1
81	4133.85	4129.23	4133.85	4132.82	4121.54	4128.72	4127.69	4129.74	4137.95	4137.95	4185.64	4907.69	4135.9	4136.92	16.812	firstRelax	1
82	4133.85	4125.13	4128.21	4130.26	4122.05	4126.67	4126.15	4133.85	4136.41	4137.44	4186.15	4907.18	4135.38	4136.92	16.812	firstRelax	1
83	4133.85	4127.69	4125.13	4128.72	4136.92	4126.15	4126.15	4134.36	4133.85	4137.44	4184.1	4905.64	4134.36	4137.95	16.812	firstRelax	1
84	4133.85	4133.85	4129.23	4128.21	4140.51	4127.69	4125.13	4134.36	4134.87	4137.44	4184.1	4905.13	4134.87	4137.44	16.812	firstRelax	1
85	4133.85	4134.36	4130.26	4130.26	4132.31	4128.72	4130.77	4134.36	4135.9	4137.44	4189.74	4907.18	4134.36	4137.44	16.812	firstRelax	1
86	4133.85	4131.28	4129.74	4130.26	4130.77	4129.23	4134.36	4133.85	4135.38	4137.44	4191.79	4912.82	4135.9	4138.46	16.812	firstRelax	1

Figura 6.20: Archivo CSV obtenido de la adquisición.

.

6.3.3. Visualización de datos

Dependiendo del tiempo que se visualizarán los estímulos así como el número de eventos que se configuraron en la pantalla de captura, el número de datos puede crecer demasiado, por lo que visualizar punto a punto los datos no resulta algo factible. Para poder mejorar la velocidad a la que se muestran, se hizo un ajuste a los datos usando el *Código 6.5*.

Código 6.5: Ajuste de valores para visualización del archivo Recording.cpp en líneas 60 a 70

```
if (i == 0)
{
    double timestamp = atof(CString(columns[15]));
    adjustX = timestamp;
    dp[j - 1] = gcnew  DataPoint();
    dp[j - 1]->XValue = timestamp - adjustX;
    yPoints[j - 1] = gcnew List<double>(datasetSize);
}

double signalValue = atof(CString(columns[j]));
yPoints[j - 1]->Add((signalValue*(adjustY *j)) / 1000);
```

El *código 6.5* realiza un ajuste sobre el eje Y de cada canal para que en lugar de tomar un sólo punto y graficarlo, se tomaran conjuntos de 32 puntos (correspondientes a 32 datos) para sacar su promedio, desplegarlo en la pantalla y mejorar el tiempo de procesamiento de los datos. Este ajuste se hizo de manera independiente para cada canal.

6.3.4. Preparación de los datos

Partiendo del hecho de que el análisis que se planea realizar es la intención de movimiento en brazo izquierdo y brazo derecho; únicamente se seleccionan los canales que son de nuestro interés: FC5 y FC6 que son los canales encargados de recolectar la información del control motriz del cuerpo y F3 y F4 encargados de recolectar los datos relacionados a la planeación motriz [24]. Así el área cerebral que abarcan dichos canales y que será el objeto de estudio es la que se indica en la *Figura 6.21*.

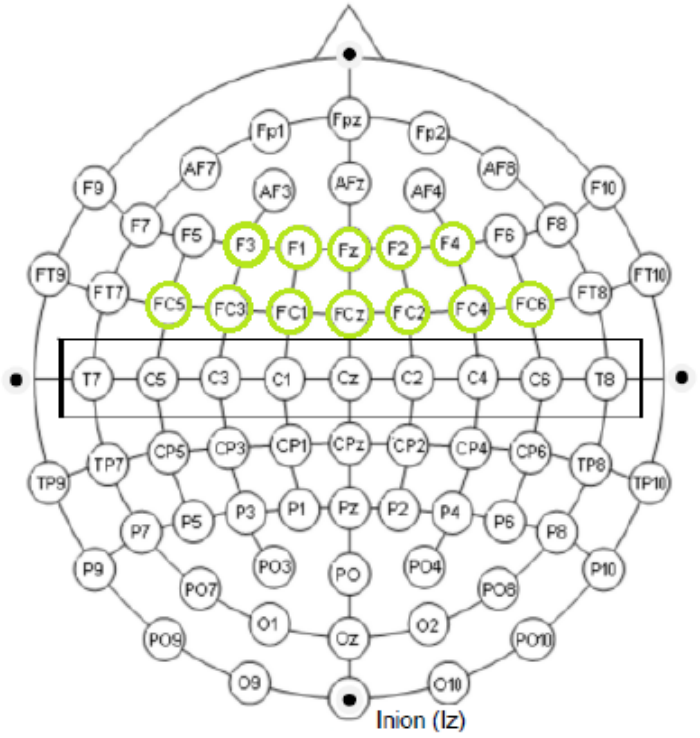

Figura 6.21: Selección de canales para la extracción.

Para poder hacer la reducción de información de los canales que se desea, se leyeron los archivos originales (generados durante la adquisición) y se conservaron solo los datos indispensables. Dicha información se mandó a otro archivo con el mismo nombre y se le agregó la etiqueta "_prep"para poder hacer la diferenciación. Adicionalmente, se descarta toda la información de los datos que correspondían a la fase de relajación y descanso, puesto que los datos de interés son los generados durante la fase en la que se presentó el estímulo de movimiento. El archivo, también en formato .csv, conservó los datos que se muestran en la *Figura 6.22*.

	A	B	C	D	E
1	COUNTER	F3	FC5	FC6	F4
2	0	4232.31	4155.38	4150.26	4153.33
3	1	4228.21	4152.82	4144.62	4149.74
4	2	4220.51	4142.05	4135.38	4144.1
5	3	4214.36	4139.49	4134.87	4138.46
6	4	4217.44	4147.69	4136.92	4137.44
7	5	4221.03	4153.33	4138.97	4145.13
8	6	4220.51	4151.79	4140	4153.33
9	7	4223.08	4148.72	4142.56	4157.44
10	8	4224.1	4151.28	4144.62	4159.49
11	9	4218.46	4151.79	4141.54	4157.95
12	10	4209.74	4144.1	4136.41	4150.77
13	11	4204.62	4139.49	4141.54	4150.26
14	12	4203.59	4146.15	4145.64	4153.33
15	13	4203.59	4152.82	4140	4146.67
16	14	4205.64	4151.79	4140	4142.05
17	15	4209.74	4151.28	4152.82	4150.77
18	16	4209.23	4151.79	4155.9	4157.95
19	17	4203.08	4151.28	4142.05	4151.28
20	18	4202.05	4149.74	4133.85	4144.62
21	19	4204.1	4151.28	4138.46	4150.77
22	20	4203.08	4153.33	4143.59	4157.95
23	21	4201.03	4145.64	4142.56	4153.33

Figura 6.22: Archivo CSV con los datos preparados para la extracción

.

6.3.5. Extracción de características

Al tener los archivos ya preparados con la información necesaria, se procede a hacer la extracción de características.Para esto, primero es necesario seleccionar el método que se va a aplicar al conjunto de datos.

Extracción de características con TWD
Para poder hacer la extracción con la Transformada de Wavelet se usó el siguiente algoritmo:

Código 6.6: Algoritmo de Transformada Wavelet Discreta del archivo Extraction.cpp en líneas 234 a 267

```
//Determina la potencia mayor de K en 2, donde K<=N
k = 1;
while (k * 2 <= n)
{
    k = k * 2;
}

while (1 < k)
{
        k = k / 2;
        for (i = 0; i < k; i++)
```

```
        {
                //F3
                y1[i] = (f3Channel[2 * i] + f3Channel[2 * i + 1]) / s;
                y1[i + k] = (f3Channel[2 * i] - f3Channel[2 * i + 1]) / s
                    ;

                //FC5
                y2[i] = (fc5Channel[2 * i] + fc5Channel[2 * i + 1]) / s;
                y2[i + k] = (fc5Channel[2 * i] - fc5Channel[2 * i + 1]) /
                     s;

                //FC6
                y3[i] = (fc6Channel[2 * i] + fc6Channel[2 * i + 1]) / s;
                y3[i + k] = (fc6Channel[2 * i] - fc6Channel[2 * i + 1]) /
                     s;

                //F4
                y4[i] = (f4Channel[2 * i] + f4Channel[2 * i + 1]) / s;
                y4[i + k] = (f4Channel[2 * i] - f4Channel[2 * i + 1]) / s
                    ;
        }

        for (i = 0; i < k * 2; i++){
                f3Channel[i] = y1[i];
                fc5Channel[i] = y2[i];
                fc6Channel[i] = y3[i];
                f4Channel[i] = y4[i];
        }
}
```

Al aplicar dicho algoritmo, garantizamos que las señales han sido procesadas y además están en el espectro de la frecuencia. Para dicha investigación, las bandas que nos interesan son las caracterizadas en la frecuencia de [8,14) Hz y [14,30] Hz correspondientes a mu y beta respectivamente. Por lo que se aplica un filtro sobre el conjunto de datos de cada canal para descartar todos los valores que están fuera de este rango de frecuencias.

Las características extraídas de las señales de cada uno de los canales de los diferentes eventos se etiquetan para saber a cual banda pertenecen (beta/mu) y se mandan a archivo. Si un vector tiene un valor pertenececiente a alguna de estas bandas, no implica que los demás también lo tengan. Por esto, si esto se presentaba, para no dejar espacios vacíos en el archivo se puso un 'na' para conservar los 8 vectores resultantes de wavelet del mismo tamaño como se muestra en la *Figura 6.23*. Para fines posteriores, el archivo conserva el mismo nombre que se generó durante la preparación de datos y cambiando el sufijo '_prep' por '_wavelet'.

MILLISECONI	F3	FC5	FC6	F4	F3WAVE	FC5WAVE	FC6WAVE	F4WAVE
39.0625	na	na	na	29.820461	na	na	na	BETA
54.6875	14.962379	na	22.666308	na	BETA	na	BETA	na
70.3125	na	11.665	19.0975	na	na	MU	BETA	na
78.125	na	12.44	na	na	na	MU	na	na
85.9375	27.9475	na	10.775	na	BETA	na	MU	na
93.75	na	na	na	10.7675	na	na	na	MU
101.5625	na	na	11.2825	na	na	na	MU	na
109.375	na	na	na	21.6625	na	na	na	BETA
132.8125	27.916576	na	na	na	BETA	na	na	na
140.625	na	na	13.420887	na	na	na	MU	na
148.4375	na	11.600087	na	na	na	MU	na	na
156.25	16.680649	13.420887	na	25.201286	BETA	MU	na	BETA
164.0625	na	15.407857	na	na	na	BETA	na	na
179.6875	16.860961	20.668731	na	18.671155	BETA	BETA	na	BETA
187.5	28.464583	18.851467	na	na	BETA	BETA	na	na
210.9375	22.302148	23.387557	na	16.132641	BETA	BETA	na	BETA
218.75	na	na	13.77444	na	na	na	MU	na
273.4375	na	na	na	10.255	na	na	na	MU
281.25	na	na	12.045	na	na	na	MU	na
296.875	na	14.36	na	na	na	BETA	na	na
312.5	8.21	15.895	na	na	MU	BETA	na	na
320.3125	na	11.795	na	na	na	MU	na	na

Figura 6.23: Archivo *_wavelet.csv resultante de la extracción de características usando TWD

Para hacer la representación gráfica de los datos se utilizó el algoritmo de la Transformada Inversa de Wavelet para ver los datos en su forma de onda y no en el espectro de frecuencia, que es como se tienen hasta este puntos los datos. Al igual que TWD, la Transformada Wavelet Inversa ocupa vectores en potencia de 2 tal como se muestra a continuación.

Código 6.7: Transformada Inversa de Wavelet del archivo Extraction.cpp en líneas 333 a 353

```
for (i = 0; i < n; i++)
{
    y[i] = 0.0L;
}

k = 1;

while (k * 2 <= n)
{
    for (i = 0; i < k; i++)
    {
        y[2 * i] = (frequencyChannel[i] + frequencyChannel[i + k]) / s;
        y[2 * i + 1] = (frequencyChannel[i] - frequencyChannel[i + k])/s;
    }

    for (i = 0; i < k * 2; i++)
    {
        frequencyChannel[i] = y[i];
    }
    k = k * 2;
}
```

Al aplicar dicho algoritmo, se tienen nuevos datos, donde se caracterizó cada onda de beta y mu para cada canal en el tiempo específico en milisegundos para poder generar las gráficas. Los datos generados de aplicar el algoritmo de wavelet inverso se mandaron a un archivo CSV

(*Figura 6.24*) donde se puede ver el tiempo de la muestra, el canal al que pertenece, así como el tipo de onda dentro del que se caracteriza. Dicho archivo sigue conservando el nombre generado durante la adquisición y esta vez cambia el sufijo '_wavelet' por '_wav_extraction'.

MILLISECONI	F3	FC5	FC6	F4	F3WAVE	FC5WAVE	FC6WAVE	F4WAVE
39.0625	na	na	na	29.820461	na	na	na	BETA
54.6875	14.962379	na	22.666308	na	BETA	na	BETA	na
70.3125	na	11.665	19.0975	na	na	MU	BETA	na
78.125	na	12.44	na	na	na	MU	na	na
85.9375	27.9475	na	10.775	na	BETA	na	MU	na
93.75	na	na	na	10.7675	na	na	na	MU
101.5625	na	na	11.2825	na	na	na	MU	na
109.375	na	na	na	21.6625	na	na	na	BETA
132.8125	27.916576	na	na	na	BETA	na	na	na
140.625	na	na	13.420887	na	na	na	MU	na
148.4375	na	11.600087	na	na	na	MU	na	na
156.25	16.680649	13.420887	na	25.201286	BETA	MU	na	BETA
164.0625	na	15.407857	na	na	na	BETA	na	na
179.6875	16.860961	20.668731	na	18.671155	BETA	BETA	na	BETA
187.5	28.464583	18.851467	na	na	BETA	BETA	na	na
210.9375	22.302148	23.387557	na	16.132641	BETA	BETA	na	BETA
218.75	na	na	13.77444	na	na	na	MU	na
273.4375	na	na	na	10.255	na	na	na	MU
281.25	na	na	12.045	na	na	na	MU	na
296.875	na	14.36	na	na	na	BETA	na	na
312.5	8.21	15.895	na	na	MU	BETA	na	na
320.3125	na	11.795	na	na	na	MU	na	na

Figura 6.24: Archivo *_wav_extraction.csv resultante de oplicar TWD inversa.

6.3.6. Entrenamiento

Para poder realizar el entrenamiento es necesario hacer un filtrado de todos los archivos existentes en la base de datos de acuerdo al sujeto, método de extracción realizado y tipo de movimiento seleccionados.

Código 6.8: Filtrado de archivos del archivo TrainingScreen.h en líneas 693 a 722

```
Void fileFilter(String^ subjectId, String^ method, String^ movement)
{
    if (starting) return;
    SrainingListView->Items->Clear();
    SubjectId = subjectId->Concat(subjectId, "_");
    String^ fileFilter = subjectId->Concat(subjectId, movement->Replace("
        ", "")->ToLower());

    if (method->CompareTo("Haar Wavelet") == 0)
        {
            fileFilter = fileFilter->Concat(fileFilter, "_*
                _wav_extraction.csv");
    }
    else
    {
        fileFilter = fileFilter->Concat(fileFilter, "_*_ica_extraction.
            csv");
    }
```

```
    for each (String^ fileName in Directory::EnumerateFiles(
        extractFolderName, extractionFileFilter, IO::SearchOption::
        TopDirectoryOnly))
        {
                String^ shortFileName = fileName->Substring(fileName->
                    LastIndexOf(Path::DirectorySeparatorChar) + 1);
                extractionFileListView->Items->Add(shortFileName)->
                    ImageIndex = 0;
        }

        for each (String^ fileName in Directory::EnumerateFiles(
            trainingFolderName, trainingFileFilter, IO::SearchOption::
            TopDirectoryOnly))
        {
                String^ shortFileName = fileName->Substring(fileName->
                    LastIndexOf(Path::DirectorySeparatorChar) + 1);
                trainingFileListView->Items->Add(shortFileName)->
                    ImageIndex = 0;
        }
}
```

Una vez que se adquiere la lista de archivos que cumplen las características seleccionadas se hace el entrenamiento. Para esto es necesario extraer los datos de todos los archivos que fueron seleccionados para realizar el entrenamiento, dichos datos se mandan a un archivo .csv como se muestra en la *Figura 6.25*, dicho archivo almacena lo extraído de los archivos seleccionados previamente.

F3MU	F3BETA	FC5MU	FC5BETA	FC6MU	FC6BETA	F4MU	F4BETA
10.559483	19.939743	10.581855	20.017704	10.486591	19.684464	10.608149	20.39125
10.559483	19.939743	11.96955	20.017704	10.486591	19.684464	10.608149	19.489631
10.559483	19.939743	10.581855	23.385789	10.486591	14.594684	10.608149	16.318257
10.062129	19.939743	10.581855	20.017704	10.486591	19.684464	10.608149	19.986056
10.559483	29.87	10.581855	20.017704	10.486591	21.155	10.608149	19.986056
10.559483	28.205	10.581855	20.017704	10.486591	17.945	10.608149	23.4575
10.559483	20.9	10.581855	20.017704	10.486591	19.684464	11.0275	19.986056
10.559483	24.4925	10.581855	20.017704	10.486591	19.684464	10.608149	19.986056
10.559483	22.695	10.581855	20.017704	10.486591	19.684464	10.608149	19.986056
9.62	19.939743	10.581855	20.017704	10.486591	19.684464	10.608149	19.986056
8.520637	19.939743	10.581855	20.017704	10.486591	19.684464	10.608149	19.986056
10.559483	19.939743	10.581855	20.017704	8.163548	19.684464	10.608149	19.986056
12.148095	19.939743	10.581855	20.017704	10.486591	19.684464	9.068644	19.986056
10.559483	19.939743	9.245421	20.017704	10.486591	19.684464	10.608149	19.986056
10.559483	19.579787	10.581855	20.017704	10.486591	19.684464	12.328407	19.986056
8.46	19.939743	10.581855	20.017704	11.54	19.684464	10.608149	19.986056
10.559483	19.939743	10.581855	20.017704	12.305	19.684464	11.54	19.986056
10.51	19.939743	10.581855	20.017704	10.77	19.684464	13.85	19.986056
10.559483	19.939743	10.581855	20.017704	12.82	19.684464	10.608149	19.986056
10.559483	19.939743	10.581855	20.017704	10.486591	14.105	10.608149	19.986056
12.825	19.939743	13.33	20.017704	12.315	19.684464	10.255	19.986056
10.559483	14.1	9.74	20.017704	10.486591	19.684464	8.205	19.986056

Figura 6.25: Archivo *.csv usado para realizar el entrenamiento.

Adicionalmente, se hace un ajuste a los datos antes de escribirlos al archivo. Dado que los archivos resultantes de la extracción de características contenían espacios (identificados como "na") donde no había ninguna señal, los vectores resultantes de los canales no eran del mismo tamaño. Por esto, primero se sacó la media de cada vector sin tomar encuenta los espacios vacios

y posteriormente, los valores donde se tenía una "na"se sustituyeron con el valor resultante de la media de cada vector. De esta forma, se logró que los vectores de datos fueran del mismo tamaño para así poder realizar el cálculo de la matriz de varianza-covarianza como se muestra a continuación:

Código 6.9: Cálculo para el vector de medias del archivo Training.cpp en líneas 48 a 63

```
for (int i = 1; i < 5; i++)
    {
            if (columns[i]->CompareTo("na") != 0)
                {
                    if (columns[i+4]->CompareTo("MU") == 0)
                        {
                                means[j] += atof(CString(columns[i]));
                                numSamples[j] += 1;
                        }
                        else
                        {
                        means[j+1] += atof(CString(columns[i]));
                                numSamples[j+1] += 1;
                        }
                }
        j+=2;
}
```

Código 6.10: Cálculo de la matriz de varianza-covarianza del archivo Training.cpp en líneas 174 a 200

```
for (int k = 0; k < 8; k++)
{
        allValues += gcnew String(to_string(values[k]).c_str());
        if(k < 7) allValues +=",";
}
//Multiplica los valores por la combinaci n de variables en el ciclo
for (int m = 0; m < 8; m++)
{
    for (int n = 0; n < 8; n++)
        {
                covariance[m, n] += (values[m]-means[m]) * (values[n] -
                   means[n]);
        }
}
totalSamples++;

// Calcula la media de los productos y resta el producto de las medias
totalSamples--;
for (int m = 0; m < 8; m++)
{
        for (int n = 0; n < 8; n++)
    {
        covariance[m, n] = (covariance[m, n] / totalSamples);
        }
}
```

Por último, al haber realizado los cálculos necesarios, el vector de medias como la matriz de covarianza se guardan en un archivo .csv (*Figura 6.26*), el nombre de este contiene los datos seleccionados para dicho entrenamiento es decir:

- Id del sujeto
- Tipo de movimiento
- Método de extracción

WAVE	MEAN	F3MU	F3BETA	FC5MU	FC5BETA	FC6MU	FC6BETA	F4MU	F4BETA
F3MU	10.574287	0.745353	0	-0.00393	0.050143	0.019549	-0.021673	0.034051	-0.033297
F3BETA	19.390177	0	3.230776	-0.028242	0.205974	-0.034646	0.241836	-0.044929	0.388368
FC5MU	10.533656	-0.00393	-0.028242	0.918273	0	0.006423	-0.00925	0.001268	-0.013602
FC5BETA	19.842849	0.050143	0.205974	0	4.520934	0.034369	0.259804	0.054088	-0.003145
FC6MU	10.569534	0.019549	-0.034646	0.006423	0.034369	0.797644	0	0.020501	-0.029812
FC6BETA	19.771839	-0.021673	0.241836	-0.00925	0.259804	0	3.613739	-0.010337	0.345923
F4MU	10.552951	0.034051	-0.044929	0.001268	0.054088	0.020501	-0.010337	0.644942	0
F4BETA	19.878126	-0.033297	0.388368	-0.013602	-0.003145	-0.029812	0.345923	0	2.965061

Figura 6.26: Archivo training.csv resultante del entrenamiento.

.

7. Conclusiones

Emotiv, fabricante de la diadema posee aplicaciones cerradas para hacer uso de la diadema; sin embargo, estas no realizaban las funciones necesarias para lo que se pretendía. Por esto, la meta principal de la investigación del proyecto era tener una interfaz de fácil manejo y de código libre que pudiera hacer el pesado procesamiento de datos tanto para hacer la extracción de características como el entrenamiento y la clasificación de datos.

De manera general, se puede decir que en principio se extrajeron las señales de la diadema, haciendo uso del SDK proporcionado por el fabricante; sin embargo, una limitante que se tuvo en principio fue la compatibilidad del mismo con los diferentes sistemas operativos. Las librerías que se tenían del SDK eran funcionales únicamente con Linux, cuando se lograron conseguir las librerías para Windows, se pudo observar que la única diferencia en la lógica era el nombre de las variables. Es decir, por ejemplo, una variable que era funcional para Linux tenía el nombre de IEE_EEG_ContactQuality_t mientras que para el ambiente de Windows era simplemente EE_EEG_ContactQuality_. De igual forma, todas las variables tenían esa I al principio del nombre, y ésta era la única diferencia entre la lógica funcional de un sistema operativo y otro.

Posteriormente se aplicó la Transformada Wavelet Discreta a cuatro canales diferentes FC5, FC6, F3, y F4 . Para poder hacer este procesamiento fue necesario hacer un ajuste en el algoritmo de wavelet dado que normalmente, éste se usa para procesamiento de imágenes. El algoritmo trabaja con datos de tamaño 2n̂ por lo que al tener vectores de datos y procesarlos mediante este algoritmo, había algunos que se perdían. El ajuste del algoritmo, consistió en perder la menor cantidad de datos posibles para poder hacer una búsqueda de características que resultara factible para el posterior procesamiento de datos.

Adicionalmente, uno de los mayores retos que se tuvo para el desarrollo de la interfaz, fue el poder vincular las librerías propias de la diadema con lo que ofrece Visual Studio para el mapeo de variables, así como el mapeo para el posicionamiento de los eletrodos de la diadema.

Bibliografía

[1] Herweg, Andreas and Gutzeit, Julian and Kleih, Sonja and Kübler, Andrea, 2016, *Wheelchair control by elderly participants in a virtual environment with a brain-computer interface (BCI) and tactile stimulation*, Elsevier, Biological Psychology.

[2] G. Rodríguez, 2013, *Adquisición, procesamiento y clasificación de señales EEG para diseño de sistemas BCI basados en imaginación de movimiento*, VI Jornadas de introducción a la investigación de la UPCT.

[3] R. Barea Navarro, *Electroencefalografía*, Instrumentación Biomédica, Departamento de Electrónica, Universidad de Alcalá

[4] J.R. De la O Chávez, 2007, *Interfaz cerebro-computadora para el control de un cursor basada en ondas cerebrales*, Proyecto Terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana, México, 2007.

[5] J. Cirilo Cruz, 2015, *Implementación de señal EEG de intención de movimiento para teleoperación de robot móvil diferencial*, Proyecto Terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana, México, 2015.

[6] H. S. Anupama and N. K. Cauvery and G. M. Lingaraju, 2014, *Real-time EEG based object recognition system using Brain Computer Interface*, Contemporary Computing and Informatics (IC3I), 2014 International Conference on.

[7] A. G. Pomer-Escher and M. D. P. de Souza and T. F. B. Filho, 2014, *Methodology for analysis of stress level based on asymmetry patterns of alpha rhythms in EEG signals*, 5th ISSNIP-IEEE Biosignals and Biorobotics Conference (2014): Biosignals and Robotics for Better and Safer Living (BRC).

[8] D. Wijayasekara and M. Manic, 2013, *Human machine interaction via brain activity monitoring*, 2013 6th International Conference on Human System Interactions (HSI).

[9] B. J. Zier, 2012, *SSVEP-based brain computer interface using the Emotiv EPOC*, EWU Masters Thesis Collection, Eastern Washington University.

[10] The Society for Neuroscience. (2012). *Brain Facts: A Primer on the Brain and Nervous System*. Marzo, 2017.

[11] Guillery, 2005, *Observations of synaptic structures: origins of the neuron doctrine and its current status* Marzo, 2017, de Philosophical Transactions of the Royal Society of London B: Biological Sciences. The Royal Society.

[12] Alan Longastaff, *BIOS Instant Notes in Neuroscience*, Taylor Francis Ltd, 2011, ISBN: 9780203808290.

[13] Juri D. Kropotov, *Quantitative EEG, Event-Related Potentials and Neurotherapy*, Academic Press, Elsevier, 2009, ISBN: 978-0-12-374512-5

[14] Steven J. Luck, *An Introduction to the Event-Related Potential Technique*, MIT Press, 2005, ISBN: 0-262-12277-4

[15] Aboul Ella Hassanien, Ahmad Taher Azar, *Brain-computer interfaces*, Springer, 2015, ISBN: 9783319109787

[16] Fernando Nicolas-Alonso, Luis and Gomez-Gil, Jaime., 2012, *Brain Computer Interfaces, a Review*, SENSORS.

[17] Emotiv Epoc, *EPOC+*, http://emotiv.com/epoc/

[18] Claudia Nureibis Henríquez Muñoz,2014, *Estudio de técnicas de análisis y clasificación de señales EEG en el contexto de sistemas BCI*, Universidad Autónoma de Madrid, Departamento de Ingeniería Informática.

[19] González R. (2010). Revisión de la teoría de wavelets. Marzo, 2017, de Universidad de las Américas Puebla. Sitio web: catarina.udlap.mx/u_dl_a/tales/documentos/mel/gonzalez_g_ra/capitulo3.pdf

[20] Liesner Acevedo Martínez. (2009). Computación Paralela de la Transformada Wavelet; Aplicaciones de la transformada Wavelet al álgebra lineal numérica. Marzo, 2017, de Universidad Politécnica de valencia Sitio web: http://www.dsic.upv.es/docs/bib-dig/tesis/etd-05152009-123504/phd.pdf

[21] A. Hyvarinen (1999), *Fast and robust fixed-point algorithms for independent component analysis.* Marzo, 2017, IEEE Transactions on Neural Networks, vol. 10, no. 3, pp. 626-634. Sitio web: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=761722isnumber=16479

[22] Pacheco S. . (2005). *El clasificador Naïve Bayes en la extracción de conocimiento de bases de datos.* Marzo, 2017, de Revista de la Facultad de Ingeniería Mecánica y Eléctrica de la Universidad Autónoma de Nuevo León Sitio web: ingenierias.uanl.mx/27/27_clasificador.pdf

[23] A. N. Bermúdez Cicchino, 2013, *Técnicas de procesamiento de EEG para detección de eventos*, Departamento de Electrotecnia, Universidad Nacional de la Plata.

[24] Sra Sontisirkit. (2013). *Special Study on guildlines to develop brain computer interface.* Marzo, 2017, de Asian Institute of Technology Sitio web: http://sralife.com/workblog/eeg_2014/assets/docs/sra_special_study_eeg.pdf

Anexos

Código Fuente en C++

Código 7.1: Cabecera para pantalla inicial

```
#pragma once
#include "StimulationScreen.h"
#include "RecordingBrowser.h"
#include "HeadsetStatusScreen.h"
#include "ExtractionBrowser.h"
#include "TrainingScreen.h"
#include "ClassificationScreen.h"

namespace EEGRECORD {

        using namespace System;
        using namespace System::ComponentModel;
        using namespace System::Collections;
        using namespace System::Windows::Forms;
        using namespace System::Data;
        using namespace System::Drawing;
        using namespace System::Globalization;
        using namespace System::Threading;
        using namespace System::IO;
        using namespace System::Collections::Generic;
        using namespace System::Text;
        using namespace System::Threading::Tasks;
        using namespace System::Diagnostics;
        using namespace System::Runtime::InteropServices;

        [DllImportAttribute("user32")]
        extern IntPtr SetParent(IntPtr hwc, IntPtr hwp);

        /// <summary>
        /// Summary for MainScreen
        /// </summary>
        public ref class MainScreen : public System::Windows::Forms::Form
        {
        public:
                MainScreen(void)
                {
                        //Thread::CurrentThread->CurrentUICulture = gcnew
                            CultureInfo(L"en-US");
                        InitializeComponent();
                        initializeScreen();
                        //
                        //TODO: Add the constructor code here
                        //
                }

                //private: System::Windows::Forms::
                   RichTextBoxLanguageOptions

        protected:
                /// <summary>
```

```
        /// Clean up any resources being used.
        /// </summary>
        ~MainScreen()
        {
                if (components)
                {
                        delete components;
                }
        }
private: System::Windows::Forms::ToolStripContainer^
    toolStripContainer1;


private: System::Windows::Forms::PictureBox^  pictureBox1;
private: System::Windows::Forms::ToolStrip^  mainScreenToolStrip;

private: System::Windows::Forms::ToolStripSeparator^
    toolStripSeparator1;
private: System::Windows::Forms::ToolStripButton^
    extractionToolStripButton;
private: System::Windows::Forms::ToolStripSeparator^
    toolStripSeparator2;
private: System::Windows::Forms::ToolStripButton^
    trainingToolStripButton;
private: System::Windows::Forms::ToolStripSeparator^
    toolStripSeparator3;
private: System::Windows::Forms::ToolStripButton^
    classificationToolStripButton;

private: System::Windows::Forms::ToolStripPanel^
    BottomToolStripPanel;
private: System::Windows::Forms::ToolStripPanel^
    TopToolStripPanel;
private: System::Windows::Forms::ToolStripPanel^
    RightToolStripPanel;
private: System::Windows::Forms::ToolStripPanel^
    LeftToolStripPanel;
private: System::Windows::Forms::ToolStripContentPanel^
    ContentPanel;
private: System::Windows::Forms::ToolStripSeparator^
    toolStripSeparator4;
private: System::Windows::Forms::ToolStripSeparator^
    toolStripSeparator5;
private: System::Windows::Forms::ToolStripDropDownButton^
    settingsToolStripDropDownButton;
private: System::Windows::Forms::ToolStripMenuItem^
    selectDirectoryToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^
    languageToolStripMenuItem;
private: System::Windows::Forms::ToolStripDropDownButton^
    helpToolStripButton;
private: System::Windows::Forms::ToolStripMenuItem^
    vieHelpToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^
    indexToolStripMenuItem;
private: System::Windows::Forms::ToolStripSeparator^
    toolStripSeparator6;
private: System::Windows::Forms::ToolStripMenuItem^
    aboutToolStripMenuItem;
```

```
        private: System::Windows::Forms::ToolStripDropDownButton^
            acquisitionToolStripButton;
        private: System::Windows::Forms::ToolStripMenuItem^
            offlineToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^
            onlineToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^
            englishToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^
            espa olToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^
            stimulationToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^
            viewRecordingToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^
            headsetToolStripMenuItem;
        private:
                //OpenFileDialog^ openFileDialog1;
                String^ folderName;

        private: System::Windows::Forms::FolderBrowserDialog^
            folderBrowserDialog;
        private: System::Windows::Forms::ToolStripSeparator^
            toolStripSeparator7;
        private: System::Windows::Forms::ToolStripButton^
            exitToolStripButton;


        protected:

        private:
                /// <summary>
                /// Required designer variable.
                /// </summary>
                System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                void InitializeComponent(void)
                {
                        System::ComponentModel::ComponentResourceManager^
                             resources = (gcnew System::ComponentModel::
                            ComponentResourceManager(MainScreen::typeid));
                        this->toolStripContainer1 = (gcnew System::
                            Windows::Forms::ToolStripContainer());
                        this->pictureBox1 = (gcnew System::Windows::Forms
                            ::PictureBox());
                        this->mainScreenToolStrip = (gcnew System::
                            Windows::Forms::ToolStrip());
                        this->acquisitionToolStripButton = (gcnew System
                            ::Windows::Forms::ToolStripDropDownButton());
                        this->offlineToolStripMenuItem = (gcnew System::
                            Windows::Forms::ToolStripMenuItem());
                        this->stimulationToolStripMenuItem = (gcnew
                            System::Windows::Forms::ToolStripMenuItem());
                        this->viewRecordingToolStripMenuItem = (gcnew
```

```
            System::Windows::Forms::ToolStripMenuItem());
        this->onlineToolStripMenuItem = (gcnew System::
            Windows::Forms::ToolStripMenuItem());
        this->headsetToolStripMenuItem = (gcnew System::
            Windows::Forms::ToolStripMenuItem());
        this->toolStripSeparator1 = (gcnew System::
            Windows::Forms::ToolStripSeparator());
        this->extractionToolStripButton = (gcnew System::
            Windows::Forms::ToolStripButton());
        this->toolStripSeparator2 = (gcnew System::
            Windows::Forms::ToolStripSeparator());
        this->trainingToolStripButton = (gcnew System::
            Windows::Forms::ToolStripButton());
        this->toolStripSeparator3 = (gcnew System::
            Windows::Forms::ToolStripSeparator());
        this->classificationToolStripButton = (gcnew
            System::Windows::Forms::ToolStripButton());
        this->toolStripSeparator4 = (gcnew System::
            Windows::Forms::ToolStripSeparator());
        this->settingsToolStripDropDownButton = (gcnew
            System::Windows::Forms::
            ToolStripDropDownButton());
        this->selectDirectoryToolStripMenuItem = (gcnew
            System::Windows::Forms::ToolStripMenuItem());
        this->languageToolStripMenuItem = (gcnew System::
            Windows::Forms::ToolStripMenuItem());
        this->englishToolStripMenuItem = (gcnew System::
            Windows::Forms::ToolStripMenuItem());
        this->espa olToolStripMenuItem = (gcnew System::
            Windows::Forms::ToolStripMenuItem());
        this->toolStripSeparator5 = (gcnew System::
            Windows::Forms::ToolStripSeparator());
        this->helpToolStripButton = (gcnew System::
            Windows::Forms::ToolStripDropDownButton());
        this->vieHelpToolStripMenuItem = (gcnew System::
            Windows::Forms::ToolStripMenuItem());
        this->indexToolStripMenuItem = (gcnew System::
            Windows::Forms::ToolStripMenuItem());
        this->toolStripSeparator6 = (gcnew System::
            Windows::Forms::ToolStripSeparator());
        this->aboutToolStripMenuItem = (gcnew System::
            Windows::Forms::ToolStripMenuItem());
        this->toolStripSeparator7 = (gcnew System::
            Windows::Forms::ToolStripSeparator());
        this->exitToolStripButton = (gcnew System::
            Windows::Forms::ToolStripButton());
        this->BottomToolStripPanel = (gcnew System::
            Windows::Forms::ToolStripPanel());
        this->TopToolStripPanel = (gcnew System::Windows
            ::Forms::ToolStripPanel());
        this->RightToolStripPanel = (gcnew System::
            Windows::Forms::ToolStripPanel());
        this->LeftToolStripPanel = (gcnew System::Windows
            ::Forms::ToolStripPanel());
        this->ContentPanel = (gcnew System::Windows::
            Forms::ToolStripContentPanel());
        this->folderBrowserDialog = (gcnew System::
            Windows::Forms::FolderBrowserDialog());
        this->toolStripContainer1->ContentPanel->
```

```
        SuspendLayout();
    this->toolStripContainer1->RightToolStripPanel->
        SuspendLayout();
    this->toolStripContainer1->SuspendLayout();
    (cli::safe_cast<System::ComponentModel::
        ISupportInitialize^>(this->pictureBox1))->
        BeginInit();
    this->mainScreenToolStrip->SuspendLayout();
    this->SuspendLayout();
    //
    // toolStripContainer1
    //
    //
    // toolStripContainer1.ContentPanel
    //
    resources->ApplyResources(this->
        toolStripContainer1->ContentPanel, L"
        toolStripContainer1.ContentPanel");
    this->toolStripContainer1->ContentPanel->Controls
        ->Add(this->pictureBox1);
    this->toolStripContainer1->ContentPanel->Load +=
        gcnew System::EventHandler(this, &MainScreen::
        toolStripContainer1_ContentPanel_Load);
    resources->ApplyResources(this->
        toolStripContainer1, L"toolStripContainer1");
    this->toolStripContainer1->Name = L"
        toolStripContainer1";
    //
    // toolStripContainer1.RightToolStripPanel
    //
    this->toolStripContainer1->RightToolStripPanel->
        Controls->Add(this->mainScreenToolStrip);
    //
    // toolStripContainer1.TopToolStripPanel
    //
    resources->ApplyResources(this->
        toolStripContainer1->TopToolStripPanel, L"
        toolStripContainer1.TopToolStripPanel");
    this->toolStripContainer1->TopToolStripPanel->
        Click += gcnew System::EventHandler(this, &
        MainScreen::
        toolStripContainer1_TopToolStripPanel_Click);
    //
    // pictureBox1
    //
    resources->ApplyResources(this->pictureBox1, L"
        pictureBox1");
    this->pictureBox1->Name = L"pictureBox1";
    this->pictureBox1->TabStop = false;
    this->pictureBox1->Click += gcnew System::
        EventHandler(this, &MainScreen::
        pictureBox1_Click);
    //
    // mainScreenToolStrip
    //
    this->mainScreenToolStrip->BackColor = System::
        Drawing::SystemColors::Control;
    resources->ApplyResources(this->
        mainScreenToolStrip, L"mainScreenToolStrip");
```

```
this->mainScreenToolStrip->GripStyle = System::
    Windows::Forms::ToolStripGripStyle::Hidden;
this->mainScreenToolStrip->ImageScalingSize =
    System::Drawing::Size(64, 64);
this->mainScreenToolStrip->Items->AddRange(gcnew
    cli::array< System::Windows::Forms::
    ToolStripItem^ >(13) {
         this->acquisitionToolStripButton,
                 this->toolStripSeparator1, this->
                     extractionToolStripButton,
                     this->toolStripSeparator2,
                     this->trainingToolStripButton,
                      this->toolStripSeparator3,
                 this->
                     classificationToolStripButton,
                      this->toolStripSeparator4,
                     this->
                     settingsToolStripDropDownButton
                     , this->toolStripSeparator5,
                 this->helpToolStripButton, this->
                     toolStripSeparator7, this->
                     exitToolStripButton
});
this->mainScreenToolStrip->Name = L"
    mainScreenToolStrip";
this->mainScreenToolStrip->Stretch = true;
this->mainScreenToolStrip->ItemClicked += gcnew
    System::Windows::Forms::
    ToolStripItemClickedEventHandler(this, &
    MainScreen::mainScreenToolStrip_ItemClicked);
//
// acquisitionToolStripButton
//
this->acquisitionToolStripButton->BackColor =
    System::Drawing::SystemColors::Control;
this->acquisitionToolStripButton->DropDownItems->
    AddRange(gcnew cli::array< System::Windows::
    Forms::ToolStripItem^ >(3) {
         this->offlineToolStripMenuItem,
                 this->onlineToolStripMenuItem,
                     this->headsetToolStripMenuItem
});
this->acquisitionToolStripButton->ForeColor =
    System::Drawing::SystemColors::ControlText;
resources->ApplyResources(this->
    acquisitionToolStripButton, L"
    acquisitionToolStripButton");
this->acquisitionToolStripButton->Name = L"
    acquisitionToolStripButton";
this->acquisitionToolStripButton->Click += gcnew
    System::EventHandler(this, &MainScreen::
    acquisitionToolStripButton_Click);
//
// offlineToolStripMenuItem
//
this->offlineToolStripMenuItem->DropDownItems->
    AddRange(gcnew cli::array< System::Windows::
    Forms::ToolStripItem^ >(2) {
         this->stimulationToolStripMenuItem,
```

```
                    this->
                        viewRecordingToolStripMenuItem
            });
            this->offlineToolStripMenuItem->Name = L"
                offlineToolStripMenuItem";
            resources->ApplyResources(this->
                offlineToolStripMenuItem, L"
                offlineToolStripMenuItem");
            this->offlineToolStripMenuItem->Click += gcnew
                System::EventHandler(this, &MainScreen::
                offlineToolStripMenuItem_Click);
            //
            // stimulationToolStripMenuItem
            //
            this->stimulationToolStripMenuItem->Name = L"
                stimulationToolStripMenuItem";
            resources->ApplyResources(this->
                stimulationToolStripMenuItem, L"
                stimulationToolStripMenuItem");
            this->stimulationToolStripMenuItem->Click +=
                gcnew System::EventHandler(this, &MainScreen::
                stimulationToolStripMenuItem_Click);
            //
            // viewRecordingToolStripMenuItem
            //
            this->viewRecordingToolStripMenuItem->Name = L"
                viewRecordingToolStripMenuItem";
            resources->ApplyResources(this->
                viewRecordingToolStripMenuItem, L"
                viewRecordingToolStripMenuItem");
            this->viewRecordingToolStripMenuItem->Click +=
                gcnew System::EventHandler(this, &MainScreen::
                viewRecordingToolStripMenuItem_Click);
            //
            // onlineToolStripMenuItem
            //
            this->onlineToolStripMenuItem->Name = L"
                onlineToolStripMenuItem";
            resources->ApplyResources(this->
                onlineToolStripMenuItem, L"
                onlineToolStripMenuItem");
            //
            // headsetToolStripMenuItem
            //
            this->headsetToolStripMenuItem->Name = L"
                headsetToolStripMenuItem";
            resources->ApplyResources(this->
                headsetToolStripMenuItem, L"
                headsetToolStripMenuItem");
            this->headsetToolStripMenuItem->Click += gcnew
                System::EventHandler(this, &MainScreen::
                headsetToolStripMenuItem_Click);
            //
            // toolStripSeparator1
            //
            this->toolStripSeparator1->Name = L"
                toolStripSeparator1";
            resources->ApplyResources(this->
                toolStripSeparator1, L"toolStripSeparator1");
```

```
            //
            // extractionToolStripButton
            //
            this->extractionToolStripButton->AccessibleRole =
                System::Windows::Forms::AccessibleRole::
                ScrollBar;
            this->extractionToolStripButton->ForeColor =
                System::Drawing::SystemColors::ControlText;
            resources->ApplyResources(this->
                extractionToolStripButton, L"
                extractionToolStripButton");
            this->extractionToolStripButton->Name = L"
                extractionToolStripButton";
            this->extractionToolStripButton->Click += gcnew
                System::EventHandler(this, &MainScreen::
                extractionToolStripButton_Click);
            //
            // toolStripSeparator2
            //
            this->toolStripSeparator2->Name = L"
                toolStripSeparator2";
            resources->ApplyResources(this->
                toolStripSeparator2, L"toolStripSeparator2");
            //
            // trainingToolStripButton
            //
            this->trainingToolStripButton->ForeColor = System
                ::Drawing::SystemColors::ControlText;
            resources->ApplyResources(this->
                trainingToolStripButton, L"
                trainingToolStripButton");
            this->trainingToolStripButton->Name = L"
                trainingToolStripButton";
            this->trainingToolStripButton->Click += gcnew
                System::EventHandler(this, &MainScreen::
                trainingToolStripButton_Click);
            //
            // toolStripSeparator3
            //
            this->toolStripSeparator3->Name = L"
                toolStripSeparator3";
            resources->ApplyResources(this->
                toolStripSeparator3, L"toolStripSeparator3");
            //
            // classificationToolStripButton
            //
            this->classificationToolStripButton->ForeColor =
                System::Drawing::SystemColors::ControlText;
            resources->ApplyResources(this->
                classificationToolStripButton, L"
                classificationToolStripButton");
            this->classificationToolStripButton->Name = L"
                classificationToolStripButton";
            this->classificationToolStripButton->Click +=
                gcnew System::EventHandler(this, &MainScreen::
                classificationToolStripButton_Click);
            //
            // toolStripSeparator4
            //
```

```
this->toolStripSeparator4->Name = L"
    toolStripSeparator4";
resources->ApplyResources(this->
    toolStripSeparator4, L"toolStripSeparator4");
//
// settingsToolStripDropDownButton
//
this->settingsToolStripDropDownButton->
    DropDownItems->AddRange(gcnew cli::array<
    System::Windows::Forms::ToolStripItem^  >(2) {
         this->selectDirectoryToolStripMenuItem,
                 this->languageToolStripMenuItem
});
this->settingsToolStripDropDownButton->ForeColor
    = System::Drawing::SystemColors::ControlText;
resources->ApplyResources(this->
    settingsToolStripDropDownButton, L"
    settingsToolStripDropDownButton");
this->settingsToolStripDropDownButton->Name = L"
    settingsToolStripDropDownButton";
//
// selectDirectoryToolStripMenuItem
//
this->selectDirectoryToolStripMenuItem->Name = L"
    selectDirectoryToolStripMenuItem";
resources->ApplyResources(this->
    selectDirectoryToolStripMenuItem, L"
    selectDirectoryToolStripMenuItem");
this->selectDirectoryToolStripMenuItem->Click +=
    gcnew System::EventHandler(this, &MainScreen::
    selectDirectoryToolStripMenuItem_Click);
//
// languageToolStripMenuItem
//
this->languageToolStripMenuItem->DropDownItems->
    AddRange(gcnew cli::array< System::Windows::
    Forms::ToolStripItem^  >(2) {
         this->englishToolStripMenuItem,
                 this->espa olToolStripMenuItem
});
this->languageToolStripMenuItem->Name = L"
    languageToolStripMenuItem";
resources->ApplyResources(this->
    languageToolStripMenuItem, L"
    languageToolStripMenuItem");
//
// englishToolStripMenuItem
//
resources->ApplyResources(this->
    englishToolStripMenuItem, L"
    englishToolStripMenuItem");
this->englishToolStripMenuItem->Name = L"
    englishToolStripMenuItem";
this->englishToolStripMenuItem->Click += gcnew
    System::EventHandler(this, &MainScreen::
    englishToolStripMenuItem_Click);
//
// espa olToolStripMenuItem
//
```

```
            this->espa olToolStripMenuItem->Name = L"
                espa olToolStripMenuItem";
            resources->ApplyResources(this->
                espa olToolStripMenuItem, L"
                espa olToolStripMenuItem");
            this->espa olToolStripMenuItem->Click += gcnew
                System::EventHandler(this, &MainScreen::
                espa olToolStripMenuItem_Click);
            //
            // toolStripSeparator5
            //
            this->toolStripSeparator5->Name = L"
                toolStripSeparator5";
            resources->ApplyResources(this->
                toolStripSeparator5, L"toolStripSeparator5");
            //
            // helpToolStripButton
            //
            this->helpToolStripButton->DropDownItems->
                AddRange(gcnew cli::array< System::Windows::
                Forms::ToolStripItem^ >(4) {
                    this->vieHelpToolStripMenuItem,
                            this->indexToolStripMenuItem,
                                this->toolStripSeparator6,
                                this->aboutToolStripMenuItem
            });
            this->helpToolStripButton->ForeColor = System::
                Drawing::SystemColors::ControlText;
            resources->ApplyResources(this->
                helpToolStripButton, L"helpToolStripButton");
            this->helpToolStripButton->Name = L"
                helpToolStripButton";
            //
            // vieHelpToolStripMenuItem
            //
            this->vieHelpToolStripMenuItem->Name = L"
                vieHelpToolStripMenuItem";
            resources->ApplyResources(this->
                vieHelpToolStripMenuItem, L"
                vieHelpToolStripMenuItem");
            this->vieHelpToolStripMenuItem->Click += gcnew
                System::EventHandler(this, &MainScreen::
                vieHelpToolStripMenuItem_Click);
            //
            // indexToolStripMenuItem
            //
            this->indexToolStripMenuItem->Name = L"
                indexToolStripMenuItem";
            resources->ApplyResources(this->
                indexToolStripMenuItem, L"
                indexToolStripMenuItem");
            //
            // toolStripSeparator6
            //
            this->toolStripSeparator6->Name = L"
                toolStripSeparator6";
            resources->ApplyResources(this->
                toolStripSeparator6, L"toolStripSeparator6");
            //
```

```
            // aboutToolStripMenuItem
            //
            this->aboutToolStripMenuItem->Name = L"
                aboutToolStripMenuItem";
            resources->ApplyResources(this->
                aboutToolStripMenuItem, L"
                aboutToolStripMenuItem");
            //
            // toolStripSeparator7
            //
            this->toolStripSeparator7->Name = L"
                toolStripSeparator7";
            resources->ApplyResources(this->
                toolStripSeparator7, L"toolStripSeparator7");
            //
            // exitToolStripButton
            //
            this->exitToolStripButton->DisplayStyle = System
                ::Windows::Forms::ToolStripItemDisplayStyle::
                Image;
            resources->ApplyResources(this->
                exitToolStripButton, L"exitToolStripButton");
            this->exitToolStripButton->Name = L"
                exitToolStripButton";
            this->exitToolStripButton->Click += gcnew System
                ::EventHandler(this, &MainScreen::
                exitToolStripButton_Click);
            //
            // BottomToolStripPanel
            //
            resources->ApplyResources(this->
                BottomToolStripPanel, L"BottomToolStripPanel")
                ;
            this->BottomToolStripPanel->Name = L"
                BottomToolStripPanel";
            this->BottomToolStripPanel->Orientation = System
                ::Windows::Forms::Orientation::Horizontal;
            this->BottomToolStripPanel->RowMargin = System::
                Windows::Forms::Padding(3, 0, 0, 0);
            //
            // TopToolStripPanel
            //
            resources->ApplyResources(this->TopToolStripPanel
                , L"TopToolStripPanel");
            this->TopToolStripPanel->Name = L"
                TopToolStripPanel";
            this->TopToolStripPanel->Orientation = System::
                Windows::Forms::Orientation::Horizontal;
            this->TopToolStripPanel->RowMargin = System::
                Windows::Forms::Padding(3, 0, 0, 0);
            //
            // RightToolStripPanel
            //
            resources->ApplyResources(this->
                RightToolStripPanel, L"RightToolStripPanel");
            this->RightToolStripPanel->Name = L"
                RightToolStripPanel";
            this->RightToolStripPanel->Orientation = System::
                Windows::Forms::Orientation::Horizontal;
```

```
            this->RightToolStripPanel->RowMargin = System::
                Windows::Forms::Padding(3, 0, 0, 0);
            //
            // LeftToolStripPanel
            //
            resources->ApplyResources(this->
                LeftToolStripPanel, L"LeftToolStripPanel");
            this->LeftToolStripPanel->Name = L"
                LeftToolStripPanel";
            this->LeftToolStripPanel->Orientation = System::
                Windows::Forms::Orientation::Horizontal;
            this->LeftToolStripPanel->RowMargin = System::
                Windows::Forms::Padding(3, 0, 0, 0);
            //
            // ContentPanel
            //
            resources->ApplyResources(this->ContentPanel, L"
                ContentPanel");
            //
            // folderBrowserDialog
            //
            resources->ApplyResources(this->
                folderBrowserDialog, L"folderBrowserDialog");
            this->folderBrowserDialog->RootFolder = System::
                Environment::SpecialFolder::MyComputer;
            this->folderBrowserDialog->HelpRequest += gcnew
                System::EventHandler(this, &MainScreen::
                folderBrowserDialog_HelpRequest);
            //
            // MainScreen
            //
            resources->ApplyResources(this, L"$this");
            this->AutoScaleMode = System::Windows::Forms::
                AutoScaleMode::Font;
            this->Controls->Add(this->toolStripContainer1);
            this->FormBorderStyle = System::Windows::Forms::
                FormBorderStyle::FixedToolWindow;
            this->Name = L"MainScreen";
            this->Load += gcnew System::EventHandler(this, &
                MainScreen::MainScreen_Load);
            this->toolStripContainer1->ContentPanel->
                ResumeLayout(false);
            this->toolStripContainer1->RightToolStripPanel->
                ResumeLayout(false);
            this->toolStripContainer1->RightToolStripPanel->
                PerformLayout();
            this->toolStripContainer1->ResumeLayout(false);
            this->toolStripContainer1->PerformLayout();
            (cli::safe_cast<System::ComponentModel::
                ISupportInitialize^>(this->pictureBox1))->
                EndInit();
            this->mainScreenToolStrip->ResumeLayout(false);
            this->mainScreenToolStrip->PerformLayout();
            this->ResumeLayout(false);

        }
#pragma endregion
    private: System::Void toolStripContainer1_ContentPanel_Load(
        System::Object^  sender, System::EventArgs^  e) {
```

```
        }
        private: System::Void pictureBox1_Click(System::Object^  sender,
            System::EventArgs^  e) {
        }
        private: System::Void mainScreenToolStrip_ItemClicked(System::
            Object^  sender, System::Windows::Forms::
            ToolStripItemClickedEventArgs^  e) {
        }
        private: System::Void acquisitionToolStripButton_Click(System::
            Object^  sender, System::EventArgs^  e) {
        }
        private: System::Void extractionToolStripButton_Click(System::
            Object^  sender, System::EventArgs^  e) {
                                 ExtractionBrowser^ eb = gcnew
                                    ExtractionBrowser(folderName);
                                 this->Hide();
                                 eb->ShowDialog();
                                 eb->Close();
                                 this->Show();
        }
        private: System::Void classificationToolStripButton_Click(System
            ::Object^  sender, System::EventArgs^  e) {
                                 ClassificationScreen^ cs = gcnew
                                    ClassificationScreen();
                                 this->Hide();
                                 cs->ShowDialog();
                                 cs->Close();
                                 this->Show();
        }
        private: System::Void MainScreen_Load(System::Object^  sender,
            System::EventArgs^  e) {
                                 onlineToolStripMenuItem->Enabled = false
                                    ;
                                 offlineToolStripMenuItem->Enabled =
                                    false;
        }
        private: System::Void settw2sqingsToolStripDropDownButton_Click(
            System::Object^  sender, System::EventArgs^  e) {

        }
        private: System::Void espa olToolStripMenuItem_Click(System::
            Object^  sender, System::EventArgs^  e) {
                                 Thread::CurrentThread->CurrentUICulture
                                    = gcnew CultureInfo(L"es-MX");
                                 this->Controls->Clear();
                                 this->InitializeComponent();
                                 this->toolStripContainer1->Dock =
                                    DockStyle::Fill;
                                 this->Refresh();

        }
        private: System::Void englishToolStripMenuItem_Click(System::
            Object^  sender, System::EventArgs^  e) {
                                 Thread::CurrentThread->CurrentUICulture
                                    = gcnew CultureInfo(L"en-US");
                                 this->Controls->Clear();
                                 this->InitializeComponent();
                                 this->toolStripContainer1->Dock =
                                    DockStyle::Fill;
```

```
                        this->Refresh();
        }
        private: System::Void stimulationToolStripMenuItem_Click(System::
            Object^  sender, System::EventArgs^  e) {

                        StimulationScreen^ stimulationScreen =
                            gcnew StimulationScreen();

                        //stimulationScreen->MdiParent = this;
                        stimulationScreen->Show();
                        //Invalidate();
        }

                    void initializeScreen()
                    {
                        folderName = this->folderBrowserDialog->
                            SelectedPath;
                        if (String::IsNullOrEmpty(folderName))
                        {
                            folderName = Directory::
                                GetCurrentDirectory();
                            this->folderBrowserDialog->
                                SelectedPath = folderName;

                        }
                    }
        private: System::Void offlineToolStripMenuItem_Click(System::
            Object^  sender, System::EventArgs^  e) {
        }
        private: System::Void headsetToolStripMenuItem_Click(System::
            Object^  sender, System::EventArgs^  e) {
                        HeadsetStatusScreen^ hstatus = gcnew
                            HeadsetStatusScreen();
                        this->Hide();
                        onlineToolStripMenuItem->Enabled = true;
                        offlineToolStripMenuItem->Enabled = true
                            ;
                        hstatus->ShowDialog();
                        this->Show();
        }
        private: System::Void openFileDialog1_FileOk(System::Object^
            sender, System::ComponentModel::CancelEventArgs^  e) {
        }
        private: System::Void selectDirectoryToolStripMenuItem_Click(
            System::Object^  sender, System::EventArgs^  e) {
                        System::Windows::Forms::DialogResult
                            result = folderBrowserDialog->
                            ShowDialog();
                        if (result == System::Windows::Forms::
                            DialogResult::OK)
                        {
                            folderName = folderBrowserDialog
                                ->SelectedPath;
                        }
        }
        private: System::Void folderBrowserDialog_HelpRequest(System::
            Object^  sender, System::EventArgs^  e) {
        }
        private: System::Void exitToolStripButton_Click(System::Object^
```

```
            sender, System::EventArgs^  e) {
                                this->Close();
        }
        private: System::Void viewRecordingToolStripMenuItem_Click(System
            ::Object^  sender, System::EventArgs^  e) {
                                RecordingBrowser^ sr = gcnew
                                    RecordingBrowser(folderName);
                                this->Hide();
                                sr->ShowDialog();
                                sr->Close();
                                this->Show();
        }
        private: System::Void toolStripContainer1_TopToolStripPanel_Click
            (System::Object^  sender, System::EventArgs^  e) {
        }
        private: System::Void vieHelpToolStripMenuItem_Click(System::
            Object^  sender, System::EventArgs^  e) {
        }
        private: System::Void trainingToolStripButton_Click(System::
            Object^  sender, System::EventArgs^  e) {
                                TrainingScreen^ ts = gcnew
                                    TrainingScreen(folderName);
                                this->Hide();
                                ts->ShowDialog();
                                ts->Close();
                                this->Show();
        }
};
}
```

Código 7.2: Definición de pantallas en modo de aplicación

```
#include "MainScreen.h"
#include "SplashScreen.h"

using namespace System;
using namespace System::Windows::Forms;

[STAThread]
int main(array <String^>^ arg)
{
        Application::EnableVisualStyles();
        Application::SetCompatibleTextRenderingDefault(false);
        EEGRECORD::MainScreen form;
        EEGRECORD::SplashScreen ss;
        Application::Run(%ss);
        Application::Run(%form);

}
```

Código 7.3: Cabecera para detección de análisis de la diadema

```
#include <iostream>
#include <fstream>
#include <conio.h>
#include <list>
#include <map>
```

```
#include "edk.h"
#include "edkErrorCode.h"
#include "EmoStateDLL.h"

using namespace System;
using namespace System::Drawing;
using namespace System::IO;
using namespace std;

namespace EEGHeadsetStatus{
        public class HeadsetStatus{
                public:
                        HeadsetStatus();
                        ~HeadsetStatus();
                        void connect();
                        void check();
                        void disconnect();

                        String^ getSystemUpTime();

                        String^ getBatteryLevel();
                        String^ getWirelessStrength();
                        SolidBrush^ getContactQuality(int channel);

                private:
                        EmoStateHandle eState;
                        EmoEngineEventHandle eEvent;
                        bool readytocollect = false;
                        bool onStateChanged = false;
                        int state = 0;
                        list<EE_InputChannels_t> contactChannelList
                                = { { EE_CHAN_CMS, EE_CHAN_DRL,
                                    EE_CHAN_FP1, EE_CHAN_AF3, EE_CHAN_F7,
                                       EE_CHAN_F3, EE_CHAN_FC5, EE_CHAN_T7
                                           , EE_CHAN_P7,
                                            EE_CHAN_O1, EE_CHAN_O2,
                                               EE_CHAN_P8, EE_CHAN_T8,
                                            EE_CHAN_FC6, EE_CHAN_F4,
                                               EE_CHAN_F8, EE_CHAN_AF4
                                } };

                        float systemUpTime;
                        int batteryLevel;
                        int maxBatteryLevel;
                        int wirelessStrength;
                        map<int,int> contactQualityMap;
                        map<int, int>::iterator channelIterator;
        };
}
```

Código 7.4: Definición de funciones para detección del estado de la diadema

```
#include "EEGHeadsetStatus.h"
#include <iostream>
//#include <atlstr.h>

using namespace std;
using namespace System::Drawing;
```

```
namespace EEGHeadsetStatus
{
        HeadsetStatus::HeadsetStatus() {}
        void HeadsetStatus::connect()
        {
                eEvent = EE_EmoEngineEventCreate();
                eState = EE_EmoStateCreate();
                int result = EE_EngineConnect();

                if (result != EDK_OK)
                {
                        string message = "Emotiv Engine start up failed.
                           Error code : ";
                        message.append(to_string(result));
                        throw exception(message.c_str());
                }
                int isOn = ES_GetHeadsetOn(eState);
                if (isOn == 1)

                {
                        throw exception("Headset has not been detected.
                           Please correct");
                }
        }

        String^ HeadsetStatus::getBatteryLevel()
        {
                char batteryPercentage[8];
                if (batteryLevel > 0)
                {
                        sprintf_s(batteryPercentage, sizeof(
                           batteryPercentage), "%d", (maxBatteryLevel /
                           batteryLevel * 100));
                }
                else
                {
                        sprintf_s(batteryPercentage, sizeof(
                           batteryPercentage),"%d", batteryLevel);
                }
                String^ result = gcnew String(batteryPercentage);
                return  result;
        }

        String^  HeadsetStatus::getWirelessStrength()
        {
                String^ result;
                switch (wirelessStrength)
                {
                case NO_SIGNAL:
                        "None";
                        break;
                case BAD_SIGNAL:
                        "Bad";
                        break;
                case GOOD_SIGNAL:
                        "Good";
                        break;
                }
                return result;
```

```
}

SolidBrush^ HeadsetStatus::getContactQuality(int channel)
{
        SolidBrush^ brush;
        channelIterator = contactQualityMap.find(channel);
        if (channelIterator != contactQualityMap.end())
        {
                int quality = channelIterator->second;
                switch (quality)
                {
                case EEG_CQ_NO_SIGNAL:
                        brush = gcnew SolidBrush(Color::Black);
                        break;
                case EEG_CQ_VERY_BAD:
                        brush = gcnew SolidBrush(Color::Red);
                        break;
                case EEG_CQ_POOR:
                        brush = gcnew SolidBrush(Color::Yellow);
                        break;
                case EEG_CQ_FAIR:
                        brush = gcnew SolidBrush(Color::Orange);
                        break;
                case EEG_CQ_GOOD:
                        brush = gcnew SolidBrush(Color::Green);
                        break;
                }
        }
        if (brush == nullptr) {
                brush = gcnew SolidBrush(Color::White);
        }
        return brush;
}

String^ HeadsetStatus::getSystemUpTime()
{
        float hours = floor(systemUpTime / 3600);
        float minutes = floor((int)(systemUpTime / 60) % 60);
        float seconds = floor((int)systemUpTime % 60);
        char timeFormat[9];
        sprintf_s(timeFormat, sizeof(timeFormat),"%02d:%02d:%02d"
            , (int) hours, (int) minutes, (int) seconds);
        return gcnew String(timeFormat);
}

void HeadsetStatus::check()
{
        state = EE_EngineGetNextEvent(eEvent);
        if (state == EDK_OK) {

                EE_Event_t eventType = EE_EmoEngineEventGetType(
                    eEvent);

                if (eventType == EE_UserAdded)
                {
                        readytocollect = true;
                }

                if (eventType == EE_EmoStateUpdated)
```

```
                        {
                                onStateChanged = true;
                                EE_EmoEngineEventGetEmoState(eEvent,
                                    eState);
                        }
                }

                if (readytocollect && onStateChanged)
                {
                        onStateChanged = false;
                        systemUpTime = ES_GetTimeFromStart(eState);
                        wirelessStrength = ES_GetWirelessSignalStatus(
                            eState);
                        contactQualityMap.clear();
                        if (wirelessStrength != NO_SIGNAL)
                        {
                                ES_GetBatteryChargeLevel(eState, &
                                    batteryLevel, &maxBatteryLevel);
                                for (std::list<EE_InputChannels_t>::
                                    iterator it = contactChannelList.begin
                                    (); it != contactChannelList.end(); it
                                    ++)
                                {
                                      contactQualityMap.insert(pair<int
                                          ,int>(*it,
                                          ES_GetContactQuality(eState, *
                                          it)));
                                }
                        }
                }
        }

        void HeadsetStatus::disconnect()
        {
                EE_EngineDisconnect();
                EE_EmoStateFree(eState);
                EE_EmoEngineEventFree(eEvent);
        }

        HeadsetStatus::~HeadsetStatus()
        {

        }
}
```

Código 7.5: Detección del estado de la diadema

```
#pragma once
#include "EEGHeadsetStatus.h"
namespace EEGRECORD {

        using namespace System;

        using namespace System::ComponentModel;
        using namespace System::Collections;
        using namespace System::Windows::Forms;
        using namespace System::Data;
        using namespace System::Drawing;
        using namespace EEGHeadsetStatus;
```

```
/// <summary>
/// Resumen de HeadsetStatusScreen
/// </summary>

public ref class HeadsetStatusScreen : public System::Windows::
    Forms::Form
{
public:
        HeadsetStatusScreen(void)
        {
                InitializeComponent();
                headsetGraphics = headsetPictureBox->
                    CreateGraphics();
                headset = new HeadsetStatus();
                sensorArea = gcnew array<RectangleF> {
                        rectCMS, rectDRL, rectAF3, rectAF3,
                            rectF7,
                                rectF3, rectFC5, rectT7, rectP7,
                                rectO1, rectO2, rectP8, rectT8,
                                rectFC6, rectF4, rectF8, rectAF4
                };
                //
                //TODO: agregar c digo de constructor aqu
                //
        }
public:
        void DrawImage(Image^ image, Point point)
        {}

protected:
        /// <summary>
        /// Limpiar los recursos que se est n utilizando.
        /// </summary>
        ~HeadsetStatusScreen()
        {
                if (components)
                {
                        delete components;
                }
                if (headset != nullptr) delete [] headset;
        }
private: System::Windows::Forms::SplitContainer^  splitContainer1
    ;
private: System::Windows::Forms::ToolStrip^
    headsetStatusToolStrip;
protected:

private: System::Windows::Forms::ToolStripLabel^
    signalStrengthToolStripLabel;
private: System::Windows::Forms::ToolStripLabel^
    signalLevelToolStripLabel;

private: System::Windows::Forms::ToolStripSeparator^
    toolStripSeparator1;
private: System::Windows::Forms::ToolStripLabel^
    bateryToolStripLabel;
private: System::Windows::Forms::ToolStripProgressBar^
    batteryToolStripProgressBar;
```

```
private: System::Windows::Forms::ToolStripSeparator^
    toolStripSeparator2;
private: System::Windows::Forms::ToolStripLabel^
    uptimeToolStripLabel;
private: System::Windows::Forms::ToolStripLabel^
    systemUpTimeToolStripLabel;
private: System::Windows::Forms::PictureBox^  headsetPictureBox;


                    // Create rectangles for electrodes
                    RectangleF rectT7 = RectangleF(260.0F - 100.0F,
                        353.0F - 100.0F, 33.0F, 33.0F);
                    RectangleF rectT8 = RectangleF(530.0F - 100.0F,
                        353.0F - 100.0F, 33.0F, 33.0F);
                    RectangleF rectFC5 = RectangleF(310.0F - 100.0F,
                         305.0F - 100.0F, 33.0F, 33.0F);
                    RectangleF rectFC6 = RectangleF(480.0F - 100.0F,
                         305.0F - 100.0F, 33.0F, 33.0F);
                    RectangleF rectO1 = RectangleF(355.0F - 100.0F,
                        530.0F - 100.0F, 33.0F, 33.0F);
                    RectangleF rectO2 = RectangleF(434.0F - 100.0F,
                        530.0F - 100.0F, 33.0F, 33.0F);
                    RectangleF rectAF3 = RectangleF(325.0F - 100.0F,
                         212.0F - 100.0F, 33.0F, 33.0F);
                    RectangleF rectAF4 = RectangleF(463.0F - 100.0F,
                         212.0F - 100.0F, 33.0F, 33.0F);
                    RectangleF rectF3 = RectangleF(358.0F - 100.0F,
                        262.0F - 100.0F, 33.0F, 33.0F);
                    RectangleF rectF4 = RectangleF(432.0F - 100.0F,
                        262.0F - 100.0F, 33.0F, 33.0F);
                    RectangleF rectF8 = RectangleF(512.0F - 100.0F,
                        260.0F - 100.0F, 33.0F, 33.0F);
                    RectangleF rectF7 = RectangleF(278.0F - 100.0F,
                        260.0F - 100.0F, 33.0F, 33.0F);
                    RectangleF rectCMS = RectangleF(279.0F - 100.0F,
                         413.50F - 100.0F, 33.0F, 33.0F);
                    RectangleF rectDRL = RectangleF(510.0F - 100.0F,
                         413.50F - 100.0F, 33.0F, 33.0F);
                    RectangleF rectP7 = RectangleF(310.0F - 100.0F,
                        472.0F - 100.0F, 33.0F, 33.0F);
                    RectangleF rectP8 = RectangleF(480.0F - 100.0F,
                        472.0F - 100.0F, 33.0F, 33.0F);
                    array<RectangleF>^ sensorArea;


                    Graphics^ headsetGraphics;
                    HeadsetStatus* headset;
                    bool isConnected = false;
private: System::Windows::Forms::ToolStripSplitButton^
    connectToolStripButton;
private: System::Windows::Forms::ToolStripMenuItem^
    connectToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^
    disconnectToolStripMenuItem;
private: System::Windows::Forms::Timer^  headsetStatusTimer;

private: System::ComponentModel::IContainer^  components;
```

```
        protected:

        private:
                /// <summary>
                /// Variable del dise ador requerida.
                /// </summary>


#pragma region Windows Form Designer generated code
                /// <summary>
                /// M todo necesario para admitir el Dise ador. No se
                    puede modificar
                /// el contenido del m todo con el editor de c digo.
                /// </summary>
                void InitializeComponent(void)
                {
                        this->components = (gcnew System::ComponentModel
                            ::Container());
                        System::ComponentModel::ComponentResourceManager^
                             resources = (gcnew System::ComponentModel::
                            ComponentResourceManager(HeadsetStatusScreen::
                            typeid));
                        this->splitContainer1 = (gcnew System::Windows::
                            Forms::SplitContainer());
                        this->headsetStatusToolStrip = (gcnew System::
                            Windows::Forms::ToolStrip());
                        this->signalStrengthToolStripLabel = (gcnew
                            System::Windows::Forms::ToolStripLabel());
                        this->signalLevelToolStripLabel = (gcnew System::
                            Windows::Forms::ToolStripLabel());
                        this->toolStripSeparator1 = (gcnew System::
                            Windows::Forms::ToolStripSeparator());
                        this->bateryToolStripLabel = (gcnew System::
                            Windows::Forms::ToolStripLabel());
                        this->batteryToolStripProgressBar = (gcnew System
                            ::Windows::Forms::ToolStripProgressBar());
                        this->toolStripSeparator2 = (gcnew System::
                            Windows::Forms::ToolStripSeparator());
                        this->uptimeToolStripLabel = (gcnew System::
                            Windows::Forms::ToolStripLabel());
                        this->systemUpTimeToolStripLabel = (gcnew System
                            ::Windows::Forms::ToolStripLabel());
                        this->connectToolStripButton = (gcnew System::
                            Windows::Forms::ToolStripSplitButton());
                        this->connectToolStripMenuItem = (gcnew System::
                            Windows::Forms::ToolStripMenuItem());
                        this->disconnectToolStripMenuItem = (gcnew System
                            ::Windows::Forms::ToolStripMenuItem());
                        this->headsetPictureBox = (gcnew System::Windows
                            ::Forms::PictureBox());
                        this->headsetStatusTimer = (gcnew System::Windows
                            ::Forms::Timer(this->components));
                        (cli::safe_cast<System::ComponentModel::
                            ISupportInitialize^>(this->splitContainer1))->
                            BeginInit();
                        this->splitContainer1->Panel1->SuspendLayout();
```

```
this->splitContainer1->Panel2->SuspendLayout();
this->splitContainer1->SuspendLayout();
this->headsetStatusToolStrip->SuspendLayout();
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->headsetPictureBox))
    ->BeginInit();
this->SuspendLayout();
//
// splitContainer1
//
resources->ApplyResources(this->splitContainer1,
    L"splitContainer1");
this->splitContainer1->Name = L"splitContainer1";
//
// splitContainer1.Panel1
//
this->splitContainer1->Panel1->Controls->Add(this
    ->headsetStatusToolStrip);
//
// splitContainer1.Panel2
//
this->splitContainer1->Panel2->Controls->Add(this
    ->headsetPictureBox);
//
// headsetStatusToolStrip
//
this->headsetStatusToolStrip->AllowMerge = false;
this->headsetStatusToolStrip->Items->AddRange(
    gcnew cli::array< System::Windows::Forms::
    ToolStripItem^ >(9) {
        this->signalStrengthToolStripLabel,
                this->signalLevelToolStripLabel,
                    this->toolStripSeparator1,
                    this->bateryToolStripLabel,
                    this->
                    batteryToolStripProgressBar,
                    this->toolStripSeparator2,
                this->uptimeToolStripLabel, this
                    ->systemUpTimeToolStripLabel,
                    this->connectToolStripButton
});
resources->ApplyResources(this->
    headsetStatusToolStrip, L"
    headsetStatusToolStrip");
this->headsetStatusToolStrip->Name = L"
    headsetStatusToolStrip";
//
// signalStrengthToolStripLabel
//
this->signalStrengthToolStripLabel->DisplayStyle
    = System::Windows::Forms::
    ToolStripItemDisplayStyle::Image;
resources->ApplyResources(this->
    signalStrengthToolStripLabel, L"
    signalStrengthToolStripLabel");
this->signalStrengthToolStripLabel->Name = L"
    signalStrengthToolStripLabel";
//
// signalLevelToolStripLabel
```

```
            //
            this->signalLevelToolStripLabel->Name = L"
                signalLevelToolStripLabel";
            resources->ApplyResources(this->
                signalLevelToolStripLabel, L"
                signalLevelToolStripLabel");
            //
            // toolStripSeparator1
            //
            this->toolStripSeparator1->Name = L"
                toolStripSeparator1";
            resources->ApplyResources(this->
                toolStripSeparator1, L"toolStripSeparator1");
            //
            // bateryToolStripLabel
            //
            this->bateryToolStripLabel->Name = L"
                bateryToolStripLabel";
            resources->ApplyResources(this->
                bateryToolStripLabel, L"bateryToolStripLabel")
                ;
            //
            // batteryToolStripProgressBar
            //
            this->batteryToolStripProgressBar->Name = L"
                batteryToolStripProgressBar";
            resources->ApplyResources(this->
                batteryToolStripProgressBar, L"
                batteryToolStripProgressBar");
            this->batteryToolStripProgressBar->Step = 1;
            //
            // toolStripSeparator2
            //
            this->toolStripSeparator2->Name = L"
                toolStripSeparator2";
            resources->ApplyResources(this->
                toolStripSeparator2, L"toolStripSeparator2");
            //
            // uptimeToolStripLabel
            //
            this->uptimeToolStripLabel->Name = L"
                uptimeToolStripLabel";
            resources->ApplyResources(this->
                uptimeToolStripLabel, L"uptimeToolStripLabel")
                ;
            //
            // systemUpTimeToolStripLabel
            //
            this->systemUpTimeToolStripLabel->Name = L"
                systemUpTimeToolStripLabel";
            resources->ApplyResources(this->
                systemUpTimeToolStripLabel, L"
                systemUpTimeToolStripLabel");
            //
            // connectToolStripButton
            //
            this->connectToolStripButton->Alignment = System
                ::Windows::Forms::ToolStripItemAlignment::
                Right;
```

```
this->connectToolStripButton->DropDownItems->
    AddRange(gcnew cli::array< System::Windows::
    Forms::ToolStripItem^  >(2) {
         this->connectToolStripMenuItem,
                 this->disconnectToolStripMenuItem
});
resources->ApplyResources(this->
    connectToolStripButton, L"
    connectToolStripButton");
this->connectToolStripButton->Name = L"
    connectToolStripButton";
//
// connectToolStripMenuItem
//
this->connectToolStripMenuItem->Name = L"
    connectToolStripMenuItem";
resources->ApplyResources(this->
    connectToolStripMenuItem, L"
    connectToolStripMenuItem");
this->connectToolStripMenuItem->Click += gcnew
    System::EventHandler(this, &
    HeadsetStatusScreen::
    connectToolStripMenuItem_Click);
//
// disconnectToolStripMenuItem
//
this->disconnectToolStripMenuItem->Name = L"
    disconnectToolStripMenuItem";
resources->ApplyResources(this->
    disconnectToolStripMenuItem, L"
    disconnectToolStripMenuItem");
this->disconnectToolStripMenuItem->Click += gcnew
     System::EventHandler(this, &
    HeadsetStatusScreen::
    disconnectToolStripMenuItem_Click);
//
// headsetPictureBox
//
resources->ApplyResources(this->headsetPictureBox
    , L"headsetPictureBox");
this->headsetPictureBox->Name = L"
    headsetPictureBox";
this->headsetPictureBox->TabStop = false;
//
// headsetStatusTimer
//
this->headsetStatusTimer->Tick += gcnew System::
    EventHandler(this, &HeadsetStatusScreen::
    headsetStatustimer_Tick);
//
// HeadsetStatusScreen
//
resources->ApplyResources(this, L"$this");
this->AutoScaleMode = System::Windows::Forms::
    AutoScaleMode::Font;
this->Controls->Add(this->splitContainer1);
this->FormBorderStyle = System::Windows::Forms::
    FormBorderStyle::FixedToolWindow;
this->Name = L"HeadsetStatusScreen";
```

```
                        this->FormClosed += gcnew System::Windows::Forms
                            ::FormClosedEventHandler(this, &
                            HeadsetStatusScreen::
                            HeadsetStatusScreen_FormClosed);
                        this->Load += gcnew System::EventHandler(this, &
                            HeadsetStatusScreen::HeadsetStatusScreen_Load)
                            ;
                        this->splitContainer1->Panel1->ResumeLayout(false
                            );
                        this->splitContainer1->Panel1->PerformLayout();
                        this->splitContainer1->Panel2->ResumeLayout(false
                            );
                        (cli::safe_cast<System::ComponentModel::
                            ISupportInitialize^>(this->splitContainer1))->
                            EndInit();
                        this->splitContainer1->ResumeLayout(false);
                        this->headsetStatusToolStrip->ResumeLayout(false)
                            ;
                        this->headsetStatusToolStrip->PerformLayout();
                        (cli::safe_cast<System::ComponentModel::
                            ISupportInitialize^>(this->headsetPictureBox))
                            ->EndInit();
                        this->ResumeLayout(false);

                }
#pragma endregion


        private: System::Void HeadsetStatusScreen_Load(System::Object^
            sender, System::EventArgs^  e)
        {
            headsetStatusTimer->Enabled = true;
            headsetStatusTimer->Stop();
        }

private: System::Void connectToolStripMenuItem_Click(System::Object^
    sender, System::EventArgs^  e) {
                        try
                        {
                            headset->connect();
                                isConnected = true;
                                connectToolStripButton->Text = "On";
                                headsetStatusTimer->Start();
                        }
                        catch (exception e)
                        {
                                MessageBox::Show(gcnew String(e.what()),
                                    "Headset Error", MessageBoxButtons::
                                    OK, MessageBoxIcon::Error);
                                isConnected = false;
                        }
}
private: System::Void disconnectToolStripMenuItem_Click(System::Object^
    sender, System::EventArgs^  e) {
                        if (isConnected)
                        {
                                headsetStatusTimer->Stop();
                                headset->disconnect();
                                isConnected = false;
```

```
                                connectToolStripButton->Text = "Off";
                                for (int i = 0; i < sensorArea->Length;
                                    i++)
                                {
                                        headsetGraphics->FillEllipse(
                                            gcnew SolidBrush(Color::Black
                                            ), sensorArea[i]);
                                }
                        }
}
private: System::Void headsetStatustimer_Tick(System::Object^  sender,
    System::EventArgs^  e) {
                        headsetStatusTimer->Stop();
                        headset->check();
                        signalLevelToolStripLabel->Text = headset->
                            getWirelessStrength();
                        batteryToolStripProgressBar->Text = headset->
                            getBatteryLevel()->Concat(" %");
                        batteryToolStripProgressBar->Step = System::
                            Convert::ToInt16(headset->getBatteryLevel());
                        batteryToolStripProgressBar->PerformStep();
                        systemUpTimeToolStripLabel->Text = headset->
                            getSystemUpTime();
                        for (int i = 0; i < sensorArea->Length; i++)
                        {
                                headsetGraphics->FillEllipse(headset->
                                    getContactQuality(i), sensorArea[i]);
                        }
                        headsetStatusTimer->Start();
}
private: System::Void HeadsetStatusScreen_FormClosed(System::Object^
    sender, System::Windows::Forms::FormClosedEventArgs^  e) {
                        headsetStatusTimer->Stop();
                        if (headset != nullptr)
                        {
                                if (isConnected)
                                {
                                        headset->disconnect();
                                }
                                delete headsetGraphics;
                                delete headset;
                        }
}
};
}
```

public: String^getSubjectId(); intgetEventNbr(); intgetRelaxPeriod(); intgetMovementPeriod(); intge

Código 7.6: Pantalla de adquisición

```
#pragma once
#include "StimulationConfig.h"
#include "EEGAcquisition.h"
using namespace EEGAcquisition;
namespace EEGRECORD {

        using namespace System;
        using namespace System::ComponentModel;
```

```
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace System::IO;
using namespace System::Resources;

/// <summary>
/// Summary for StimulationScreen
/// </summary>
public ref class StimulationScreen : public System::Windows::
   Forms::Form
{
public:
        StimulationScreen(void)
        {
                InitializeComponent();
                stimulationTimer->Stop();

                this->SuspendLayout();
                this->tableLayoutPanel1->SuspendLayout();
                firstRelaxPictureBox->Visible = false;
                secondRelaxPictureBox->Visible = false;
                leftArmPictureBox->Visible = false;
                rightLegPictureBox->Visible = false;
                breakPictureBox->Visible = false;
                this->tableLayoutPanel1->ResumeLayout(false);
                this->ResumeLayout(true);
                acquisition = new Acquisition();
                //
                //TODO: Add the constructor code here
                //
        }

protected:
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        ~StimulationScreen()
        {
                if (components)
                {
                        delete components;
                }
        }
private: System::Windows::Forms::TableLayoutPanel^
   tableLayoutPanel1;
private: System::Windows::Forms::Label^  startLabel;
private: System::Windows::Forms::Label^  relaxLabel;
private: System::Windows::Forms::Label^  rightLegLabel;
private: System::Windows::Forms::Label^  leftArmLabel;
private: System::Windows::Forms::Label^  breakLabel;

private: System::Windows::Forms::Timer^  stimulationTimer;

private: System::Windows::Forms::PictureBox^  startPictureBox;

private: System::ComponentModel::IContainer^  components;
```

```
        protected:

        private:

                /// <summary>
                /// Required designer variable.
                /// </summary>
        private:
                Stimulation^ stimulation;
                String^ step;
                int eventNbr = 0;
                Acquisition* acquisition;
        private: System::Windows::Forms::PictureBox^
            firstRelaxPictureBox;

        private: System::Windows::Forms::PictureBox^  leftArmPictureBox;
        private: System::Windows::Forms::PictureBox^  rightLegPictureBox;
        private: System::Windows::Forms::PictureBox^
            secondRelaxPictureBox;
        private: System::Windows::Forms::PictureBox^  breakPictureBox;
        private: System::ComponentModel::BackgroundWorker^
            acquisitionBackgroundWorker;




                        int totalEventNbr = 0;

#pragma region Windows Form Designer generated code
                        /// <summary>
                        /// Required method for Designer support - do
                            not modify
                        /// the contents of this method with the code
                            editor.
                        /// </summary>
                        void InitializeComponent(void)
                        {
                                this->components = (gcnew System::
                                    ComponentModel::Container());
                                System::ComponentModel::
                                    ComponentResourceManager^  resources
                                    = (gcnew System::ComponentModel::
                                    ComponentResourceManager(
                                    StimulationScreen::typeid));
                                this->tableLayoutPanel1 = (gcnew System
                                    ::Windows::Forms::TableLayoutPanel())
                                    ;
                                this->startPictureBox = (gcnew System::
                                    Windows::Forms::PictureBox());
                                this->firstRelaxPictureBox = (gcnew
                                    System::Windows::Forms::PictureBox())
                                    ;
                                this->leftArmPictureBox = (gcnew System
                                    ::Windows::Forms::PictureBox());
                                this->rightLegPictureBox = (gcnew System
                                    ::Windows::Forms::PictureBox());
                                this->secondRelaxPictureBox = (gcnew
```

```
    System::Windows::Forms::PictureBox())
    ;
this->breakPictureBox = (gcnew System::
    Windows::Forms::PictureBox());
this->startLabel = (gcnew System::
    Windows::Forms::Label());
this->relaxLabel = (gcnew System::
    Windows::Forms::Label());
this->rightLegLabel = (gcnew System::
    Windows::Forms::Label());
this->leftArmLabel = (gcnew System::
    Windows::Forms::Label());
this->breakLabel = (gcnew System::
    Windows::Forms::Label());
this->stimulationTimer = (gcnew System::
    Windows::Forms::Timer(this->
    components));
this->acquisitionBackgroundWorker = (
    gcnew System::ComponentModel::
    BackgroundWorker());
this->tableLayoutPanel1->SuspendLayout()
    ;
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->
    startPictureBox))->BeginInit();
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->
    firstRelaxPictureBox))->BeginInit();
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->
    leftArmPictureBox))->BeginInit();
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->
    rightLegPictureBox))->BeginInit();
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->
    secondRelaxPictureBox))->BeginInit();
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->
    breakPictureBox))->BeginInit();
this->SuspendLayout();
//
// tableLayoutPanel1
//
resources->ApplyResources(this->
    tableLayoutPanel1, L"
    tableLayoutPanel1");
this->tableLayoutPanel1->Controls->Add(
    this->startPictureBox, 0, 0);
this->tableLayoutPanel1->Name = L"
    tableLayoutPanel1";
//
// startPictureBox
//
this->startPictureBox->BackColor =
    System::Drawing::Color::Transparent;
resources->ApplyResources(this->
    startPictureBox, L"startPictureBox");
this->startPictureBox->Name = L"
```

```
    startPictureBox";
this->startPictureBox->TabStop = false;
//
// firstRelaxPictureBox
//
this->firstRelaxPictureBox->BackColor =
    System::Drawing::Color::Transparent;
resources->ApplyResources(this->
    firstRelaxPictureBox, L"
    firstRelaxPictureBox");
this->firstRelaxPictureBox->Name = L"
    firstRelaxPictureBox";
this->firstRelaxPictureBox->TabStop =
    false;
//
// leftArmPictureBox
//
resources->ApplyResources(this->
    leftArmPictureBox, L"
    leftArmPictureBox");
this->leftArmPictureBox->Name = L"
    leftArmPictureBox";
this->leftArmPictureBox->TabStop = false
    ;
//
// rightLegPictureBox
//
resources->ApplyResources(this->
    rightLegPictureBox, L"
    rightLegPictureBox");
this->rightLegPictureBox->Name = L"
    rightLegPictureBox";
this->rightLegPictureBox->TabStop =
    false;
//
// secondRelaxPictureBox
//
resources->ApplyResources(this->
    secondRelaxPictureBox, L"
    secondRelaxPictureBox");
this->secondRelaxPictureBox->Name = L"
    secondRelaxPictureBox";
this->secondRelaxPictureBox->TabStop =
    false;
//
// breakPictureBox
//
resources->ApplyResources(this->
    breakPictureBox, L"breakPictureBox");
this->breakPictureBox->Name = L"
    breakPictureBox";
this->breakPictureBox->TabStop = false;
//
// startLabel
//
resources->ApplyResources(this->
    startLabel, L"startLabel");
this->startLabel->BackColor = System::
    Drawing::SystemColors::Window;
```

```
this->startLabel->ForeColor = System::
    Drawing::Color::Black;
this->startLabel->Name = L"startLabel";
//
// relaxLabel
//
resources->ApplyResources(this->
    relaxLabel, L"relaxLabel");
this->relaxLabel->BackColor = System::
    Drawing::SystemColors::Window;
this->relaxLabel->ForeColor = System::
    Drawing::SystemColors::MenuHighlight;
this->relaxLabel->Name = L"relaxLabel";
//
// rightLegLabel
//
resources->ApplyResources(this->
    rightLegLabel, L"rightLegLabel");
this->rightLegLabel->BackColor = System
    ::Drawing::SystemColors::Window;
this->rightLegLabel->ForeColor = System
    ::Drawing::Color::Black;
this->rightLegLabel->Name = L"
    rightLegLabel";
//
// leftArmLabel
//
resources->ApplyResources(this->
    leftArmLabel, L"leftArmLabel");
this->leftArmLabel->BackColor = System::
    Drawing::SystemColors::Window;
this->leftArmLabel->ForeColor = System::
    Drawing::SystemColors::MenuHighlight;
this->leftArmLabel->Name = L"
    leftArmLabel";
//
// breakLabel
//
resources->ApplyResources(this->
    breakLabel, L"breakLabel");
this->breakLabel->BackColor = System::
    Drawing::SystemColors::Window;
this->breakLabel->ForeColor = System::
    Drawing::Color::LawnGreen;
this->breakLabel->Name = L"breakLabel";
//
// stimulationTimer
//
this->stimulationTimer->Interval = 10;
this->stimulationTimer->Tick += gcnew
    System::EventHandler(this, &
    StimulationScreen::
    stimulationTimer_Tick);
//
// acquisitionBackgroundWorker
//
this->acquisitionBackgroundWorker->
    WorkerSupportsCancellation = true;
this->acquisitionBackgroundWorker->
```

```
                DoWork += gcnew System::
                ComponentModel::DoWorkEventHandler(
                this, &StimulationScreen::
                acquisitionBackgroundWorker_DoWork);
            this->acquisitionBackgroundWorker->
                RunWorkerCompleted += gcnew System::
                ComponentModel::
                RunWorkerCompletedEventHandler(this,
                &StimulationScreen::
                acquisitionBackgroundWorker_RunWorkerCompleted
                );
            //
            // StimulationScreen
            //
            resources->ApplyResources(this, L"$this"
                );
            this->AutoScaleMode = System::Windows::
                Forms::AutoScaleMode::Font;
            this->ControlBox = false;
            this->Controls->Add(this->
                tableLayoutPanel1);
            this->FormBorderStyle = System::Windows
                ::Forms::FormBorderStyle::None;
            this->MaximizeBox = false;
            this->MinimizeBox = false;
            this->Name = L"StimulationScreen";
            this->ShowIcon = false;
            this->WindowState = System::Windows::
                Forms::FormWindowState::Maximized;
            this->Load += gcnew System::EventHandler
                (this, &StimulationScreen::
                StimulationScreen_Load);
            this->Shown += gcnew System::
                EventHandler(this, &StimulationScreen
                ::StimulationScreen_Shown);
            this->tableLayoutPanel1->ResumeLayout(
                false);
            (cli::safe_cast<System::ComponentModel::
                ISupportInitialize^>(this->
                startPictureBox))->EndInit();
            (cli::safe_cast<System::ComponentModel::
                ISupportInitialize^>(this->
                firstRelaxPictureBox))->EndInit();
            (cli::safe_cast<System::ComponentModel::
                ISupportInitialize^>(this->
                leftArmPictureBox))->EndInit();
            (cli::safe_cast<System::ComponentModel::
                ISupportInitialize^>(this->
                rightLegPictureBox))->EndInit();
            (cli::safe_cast<System::ComponentModel::
                ISupportInitialize^>(this->
                secondRelaxPictureBox))->EndInit();
            (cli::safe_cast<System::ComponentModel::
                ISupportInitialize^>(this->
                breakPictureBox))->EndInit();
            this->ResumeLayout(false);

        }
#pragma endregion
```

```
        private: System::Windows::Forms::DialogResult result;
                         StimulationConfig^ sc = gcnew StimulationConfig
                            ();

        private: System::Void StimulationScreen_Load(System::Object^
            sender, System::EventArgs^  e) {


        }

        private: System::Void stimulationTimer_Tick(System::Object^
            sender, System::EventArgs^  e) {
                                 stimulationTimer->Stop();

                                 if (step->CompareTo("firstRelax") == 0)
                                 {
                                         setPictureBox(
                                            firstRelaxPictureBox,
                                            relaxLabel, stimulation->
                                            getRelaxPeriod(), "action");
                                 }
                                 else if (step->CompareTo("action") == 0)
                                 {
                                         int movementType = stimulation->
                                            getMovementType();
                                         if (movementType == 0)
                                         {
                                                 setPictureBox(
                                                    leftArmPictureBox,
                                                    leftArmLabel,
                                                    stimulation->
                                                    getMovementPeriod(),
                                                    "secondRelax");
                                         }
                                         else if (movementType == 1)
                                         {
                                                 setPictureBox(
                                                    rightLegPictureBox,
                                                    rightLegLabel,
                                                    stimulation->
                                                    getMovementPeriod(),
                                                    "secondRelax");
                                         }
                                 }
                                 else if (step->CompareTo("secondRelax")
                                    == 0)
                                 {
                                         setPictureBox(
                                            secondRelaxPictureBox,
                                            relaxLabel, stimulation->
                                            getRelaxPeriod(), "break");
                                 }
                                 else if (step->CompareTo("break") == 0)
                                 {
                                         eventNbr++;
                                         if (eventNbr <= totalEventNbr)
                                         {
                                                 setPictureBox(
                                                    breakPictureBox,
```

```
                                        breakLabel,
                                        stimulation->
                                        getBreakPeriod(), "
                                        firstRelax");
                                }
                                else
                                {
                                        stimulationTimer->Stop()
                                            ;
                                        acquisition->pause();
                                        acquisitionBackgroundWorker
                                            ->CancelAsync();
                                }
                        }

        }

        private: System::Void StimulationScreen_Shown(System::Object^
            sender, System::EventArgs^  e) {

                        stimulationTimer->Stop();
                        result = sc->ShowDialog();
                        if (result == System::Windows::Forms::
                            DialogResult::Cancel)
                        {
                                this->Close();
                        }
                        else if (result == System::Windows::
                            Forms::DialogResult::Yes)
                        {
                                this->Cursor->Hide();
                                stimulation = sc->getStimulation
                                    ();
                                totalEventNbr = stimulation->
                                    getEventNbr();
                                step = "firstRelax";
                                stimulationTimer->Interval = 10;
                                acquisition->setMarker(step);
                                acquisition->setup(stimulation->
                                    getSubjectId(), stimulation->
                                    getMovementType());
                                acquisition->setEventNbr(++
                                    eventNbr);
                                acquisitionBackgroundWorker->
                                    RunWorkerAsync();
                                stimulationTimer->Start();
                        }

        }

        private: Void setPictureBox(PictureBox^ newPb, Label^
            pictureLabel, int interval, String^ nextStep)
        {

                        this->tableLayoutPanel1->SuspendLayout()
                            ;
                        this->tableLayoutPanel1->Controls->
                            RemoveAt(0);
                        newPb->Visible = true;
```

```
                                pictureLabel->Visible = true;
                                pictureLabel->Parent = newPb;
                                pictureLabel->BackColor = Color::
                                    Transparent;

                                this->tableLayoutPanel1->Controls->Add(
                                    newPb, 0, 0);
                                pictureLabel->BringToFront();
                                this->tableLayoutPanel1->ResumeLayout(
                                    true);
                                this->ResumeLayout(true);
                                stimulationTimer->Interval = interval;

                                String^ marker = step;
                                step = nextStep;
                                if (marker->CompareTo("action") == 0)
                                {
                                        if (stimulation->getMovementType
                                            () == 0)
                                        {
                                                marker = "leftArm";
                                        }
                                        else
                                        {
                                                marker = "rightLeg";
                                        }
                                }
                                acquisition->setMarker(marker);
                                if (marker->CompareTo("break") == 0)
                                {
                                        acquisition->setEventNbr(
                                            eventNbr - 1);
                                }
                                else
                                {
                                        acquisition->setEventNbr(
                                            eventNbr);
                                }
                                stimulationTimer->Start();

        }
        private: System::Void acquisitionBackgroundWorker_DoWork(System::
            Object^  sender, System::ComponentModel::DoWorkEventArgs^  e)
            {
                                acquisition->acquire();
        }
        private: System::Void
            acquisitionBackgroundWorker_RunWorkerCompleted(System::Object^
              sender, System::ComponentModel::RunWorkerCompletedEventArgs^
              e) {
                                acquisition->stop();
                                this->Close();
                                this->Cursor->Show();
        }
        };
}
```

Código 7.7: Cabecera para pantalla de configuración de estímulos

```
#include "Stimulation.h"
namespace EEGRECORD
{

        Stimulation::Stimulation()
        {
        }

        Stimulation::Stimulation(String^ subjectId,
                int     eventNbr,
                int     relaxPeriod,
                int     movementPeriod,
                int     movementType,
                int     breakPeriod)
        {
                this->subjectId = subjectId;
                this->eventNbr = eventNbr;
                this->relaxPeriod = relaxPeriod;
                this->movementPeriod = movementPeriod;
                this->movementType = movementType;
                this->breakPeriod = breakPeriod;
        }
        Stimulation::~Stimulation()
        {

        }

    System::String^ EEGRECORD::Stimulation::getSubjectId()
    {
        return subjectId;
    }

    int EEGRECORD::Stimulation::getEventNbr()
    {
        return eventNbr;
    }
    int     EEGRECORD::Stimulation::getRelaxPeriod()
    {
        return relaxPeriod * 1000;
    }
    int     EEGRECORD::Stimulation::getMovementPeriod()
    {
        return movementPeriod * 1000;
    }
    int EEGRECORD::Stimulation::getMovementType()
    {
        return movementType;
    }
    int     EEGRECORD::Stimulation::getBreakPeriod()
    {
        return breakPeriod * 1000;
    }

}
```

Código 7.8: Pantalla de configuración de estímulos

```
#pragma once
#include "Stimulation.h"
namespace EEGRECORD {

        using namespace System;
        using namespace System::ComponentModel;
        using namespace System::Collections;
        using namespace System::Windows::Forms;
        using namespace System::Data;
        using namespace System::Drawing;

        /// <summary>
        /// Summary for StimulationConfig
        /// </summary>
        public ref class StimulationConfig : public System::Windows::
           Forms::Form
        {
        public:
                StimulationConfig(void)
                {
                        InitializeComponent();
                        movementComboBox->SelectedItem = "Move Left Arm";
                        //
                        //TODO: Add the constructor code here
                        //
                }

        protected:
                /// <summary>
                /// Clean up any resources being used.
                /// </summary>
                ~StimulationConfig()
                {
                        if (components)
                        {
                                delete components;
                        }
                }
        private: System::Windows::Forms::SplitContainer^
           StimulationSplitContainer;
        private: System::Windows::Forms::TableLayoutPanel^
           tableLayoutPanel1;
        private: System::Windows::Forms::Button^  resetButton;

        private: System::Windows::Forms::Button^  startButton;
        private: System::Windows::Forms::PictureBox^
           stimulationPictureBox;

        private: System::Windows::Forms::Button^  closeButton;

        private: System::Windows::Forms::TableLayoutPanel^
           tableLayoutPanel2;
        private: System::Windows::Forms::Label^  relaxPeriodLabel;

        private: System::Windows::Forms::NumericUpDown^
           relaxNumericUpDown;
        private: System::Windows::Forms::Label^  movementPeriodLabel;
```

```
        private: System::Windows::Forms::NumericUpDown^
            MovementNumericUpDown;

        private: System::Windows::Forms::Label^  subjectIdLabel;
        private: System::Windows::Forms::TextBox^  subjectIdTextBox;
        private: System::Windows::Forms::Label^  eventsNbrLabel;

        private: System::Windows::Forms::NumericUpDown^  eventNbrUpDown3;




        private: System::ComponentModel::IContainer^  components;

        private:

        private: System::Windows::Forms::Label^  movementTypeLabel;

        private: System::Windows::Forms::Label^  breakPeriodLabel;
        private: System::Windows::Forms::NumericUpDown^
            breakPeriodNumericUpDown;
        private: System::Windows::Forms::ComboBox^  movementComboBox;

        protected:

        protected:

        private:
                /// <summary>
                /// Required designer variable.
                /// </summary>


#pragma region Windows Form Designer generated code
                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                void InitializeComponent(void)
                {
                        System::ComponentModel::ComponentResourceManager^
                              resources = (gcnew System::ComponentModel::
                            ComponentResourceManager(StimulationConfig::
                            typeid));
                        this->StimulationSplitContainer = (gcnew System::
                            Windows::Forms::SplitContainer());
                        this->tableLayoutPanel2 = (gcnew System::Windows
                            ::Forms::TableLayoutPanel());
                        this->relaxPeriodLabel = (gcnew System::Windows::
                            Forms::Label());
                        this->relaxNumericUpDown = (gcnew System::Windows
                            ::Forms::NumericUpDown());
                        this->MovementNumericUpDown = (gcnew System::
                            Windows::Forms::NumericUpDown());
                        this->subjectIdLabel = (gcnew System::Windows::
                            Forms::Label());
                        this->subjectIdTextBox = (gcnew System::Windows::
                            Forms::TextBox());
```

```
this->eventsNbrLabel = (gcnew System::Windows::
    Forms::Label());
this->eventNbrUpDown3 = (gcnew System::Windows::
    Forms::NumericUpDown());
this->movementTypeLabel = (gcnew System::Windows
    ::Forms::Label());
this->breakPeriodLabel = (gcnew System::Windows::
    Forms::Label());
this->breakPeriodNumericUpDown = (gcnew System::
    Windows::Forms::NumericUpDown());
this->movementPeriodLabel = (gcnew System::
    Windows::Forms::Label());
this->movementComboBox = (gcnew System::Windows::
    Forms::ComboBox());
this->stimulationPictureBox = (gcnew System::
    Windows::Forms::PictureBox());
this->tableLayoutPanel1 = (gcnew System::Windows
    ::Forms::TableLayoutPanel());
this->resetButton = (gcnew System::Windows::Forms
    ::Button());
this->startButton = (gcnew System::Windows::Forms
    ::Button());
this->closeButton = (gcnew System::Windows::Forms
    ::Button());
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->
    StimulationSplitContainer))->BeginInit();
this->StimulationSplitContainer->Panel1->
    SuspendLayout();
this->StimulationSplitContainer->Panel2->
    SuspendLayout();
this->StimulationSplitContainer->SuspendLayout();
this->tableLayoutPanel2->SuspendLayout();
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->relaxNumericUpDown)
    )->BeginInit();
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->
    MovementNumericUpDown))->BeginInit();
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->eventNbrUpDown3))->
    BeginInit();
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->
    breakPeriodNumericUpDown))->BeginInit();
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->
    stimulationPictureBox))->BeginInit();
this->tableLayoutPanel1->SuspendLayout();
this->SuspendLayout();
//
// StimulationSplitContainer
//
resources->ApplyResources(this->
    StimulationSplitContainer, L"
    StimulationSplitContainer");
this->StimulationSplitContainer->Name = L"
    StimulationSplitContainer";
//
```

```
            // StimulationSplitContainer.Panel1
            //
            this->StimulationSplitContainer->Panel1->Controls
                ->Add(this->tableLayoutPanel2);
            this->StimulationSplitContainer->Panel1->Controls
                ->Add(this->stimulationPictureBox);
            //
            // StimulationSplitContainer.Panel2
            //
            this->StimulationSplitContainer->Panel2->Controls
                ->Add(this->tableLayoutPanel1);
            //
            // tableLayoutPanel2
            //
            resources->ApplyResources(this->tableLayoutPanel2
                , L"tableLayoutPanel2");
            this->tableLayoutPanel2->Controls->Add(this->
                relaxPeriodLabel, 0, 4);
            this->tableLayoutPanel2->Controls->Add(this->
                relaxNumericUpDown, 1, 4);
            this->tableLayoutPanel2->Controls->Add(this->
                MovementNumericUpDown, 1, 6);
            this->tableLayoutPanel2->Controls->Add(this->
                subjectIdLabel, 0, 0);
            this->tableLayoutPanel2->Controls->Add(this->
                subjectIdTextBox, 1, 0);
            this->tableLayoutPanel2->Controls->Add(this->
                eventsNbrLabel, 0, 2);
            this->tableLayoutPanel2->Controls->Add(this->
                eventNbrUpDown3, 1, 2);
            this->tableLayoutPanel2->Controls->Add(this->
                movementTypeLabel, 0, 8);
            this->tableLayoutPanel2->Controls->Add(this->
                breakPeriodLabel, 0, 10);
            this->tableLayoutPanel2->Controls->Add(this->
                breakPeriodNumericUpDown, 1, 10);
            this->tableLayoutPanel2->Controls->Add(this->
                movementPeriodLabel, 0, 6);
            this->tableLayoutPanel2->Controls->Add(this->
                movementComboBox, 1, 8);
            this->tableLayoutPanel2->Name = L"
                tableLayoutPanel2";
            //
            // relaxPeriodLabel
            //
            resources->ApplyResources(this->relaxPeriodLabel,
                 L"relaxPeriodLabel");
            this->relaxPeriodLabel->Name = L"relaxPeriodLabel
                ";
            //
            // relaxNumericUpDown
            //
            resources->ApplyResources(this->
                relaxNumericUpDown, L"relaxNumericUpDown");
            this->relaxNumericUpDown->Maximum = System::
                Decimal(gcnew cli::array< System::Int32 >(4) {
                 60, 0, 0, 0 });
            this->relaxNumericUpDown->Minimum = System::
                Decimal(gcnew cli::array< System::Int32 >(4) {
```

```
    1, 0, 0, 0 });
this->relaxNumericUpDown->Name = L"
    relaxNumericUpDown";
this->relaxNumericUpDown->Value = System::Decimal
    (gcnew cli::array< System::Int32 >(4) { 2, 0,
    0, 0 });
//
// MovementNumericUpDown
//
resources->ApplyResources(this->
    MovementNumericUpDown, L"MovementNumericUpDown
    ");
this->MovementNumericUpDown->Maximum = System::
    Decimal(gcnew cli::array< System::Int32 >(4) {
     60, 0, 0, 0 });
this->MovementNumericUpDown->Minimum = System::
    Decimal(gcnew cli::array< System::Int32 >(4) {
     1, 0, 0, 0 });
this->MovementNumericUpDown->Name = L"
    MovementNumericUpDown";
this->MovementNumericUpDown->Value = System::
    Decimal(gcnew cli::array< System::Int32 >(4) {
     4, 0, 0, 0 });
//
// subjectIdLabel
//
resources->ApplyResources(this->subjectIdLabel, L
    "subjectIdLabel");
this->subjectIdLabel->Name = L"subjectIdLabel";
//
// subjectIdTextBox
//
resources->ApplyResources(this->subjectIdTextBox,
     L"subjectIdTextBox");
this->subjectIdTextBox->Name = L"subjectIdTextBox
    ";
//
// eventsNbrLabel
//
resources->ApplyResources(this->eventsNbrLabel, L
    "eventsNbrLabel");
this->eventsNbrLabel->Name = L"eventsNbrLabel";
//
// eventNbrUpDown3
//
resources->ApplyResources(this->eventNbrUpDown3,
    L"eventNbrUpDown3");
this->eventNbrUpDown3->Maximum = System::Decimal(
    gcnew cli::array< System::Int32 >(4) { 60, 0,
    0, 0 });
this->eventNbrUpDown3->Minimum = System::Decimal(
    gcnew cli::array< System::Int32 >(4) { 1, 0,
    0, 0 });
this->eventNbrUpDown3->Name = L"eventNbrUpDown3";
this->eventNbrUpDown3->Value = System::Decimal(
    gcnew cli::array< System::Int32 >(4) { 2, 0,
    0, 0 });
//
// movementTypeLabel
```

```
            //
            resources->ApplyResources(this->movementTypeLabel
                , L"movementTypeLabel");
            this->movementTypeLabel->Name = L"
                movementTypeLabel";
            //
            // breakPeriodLabel
            //
            resources->ApplyResources(this->breakPeriodLabel,
                 L"breakPeriodLabel");
            this->breakPeriodLabel->Name = L"breakPeriodLabel
                ";
            //
            // breakPeriodNumericUpDown
            //
            resources->ApplyResources(this->
                breakPeriodNumericUpDown, L"
                breakPeriodNumericUpDown");
            this->breakPeriodNumericUpDown->Maximum = System
                ::Decimal(gcnew cli::array< System::Int32 >(4)
                 { 60, 0, 0, 0 });
            this->breakPeriodNumericUpDown->Minimum = System
                ::Decimal(gcnew cli::array< System::Int32 >(4)
                 { 1, 0, 0, 0 });
            this->breakPeriodNumericUpDown->Name = L"
                breakPeriodNumericUpDown";
            this->breakPeriodNumericUpDown->Value = System::
                Decimal(gcnew cli::array< System::Int32 >(4) {
                 2, 0, 0, 0 });
            //
            // movementPeriodLabel
            //
            resources->ApplyResources(this->
                movementPeriodLabel, L"movementPeriodLabel");
            this->movementPeriodLabel->Name = L"
                movementPeriodLabel";
            //
            // movementComboBox
            //
            resources->ApplyResources(this->movementComboBox,
                 L"movementComboBox");
            this->movementComboBox->FormattingEnabled = true;
            this->movementComboBox->Items->AddRange(gcnew cli
                ::array< System::Object^  >(2) {
                    resources->GetString(L"movementComboBox.
                        Items"),
                            resources->GetString(L"
                                movementComboBox.Items1")
            });
            this->movementComboBox->Name = L"movementComboBox
                ";
            //
            // stimulationPictureBox
            //
            resources->ApplyResources(this->
                stimulationPictureBox, L"stimulationPictureBox
                ");
            this->stimulationPictureBox->Name = L"
                stimulationPictureBox";
```

```
            this->stimulationPictureBox->TabStop = false;
            //
            // tableLayoutPanel1
            //
            resources->ApplyResources(this->tableLayoutPanel1
                , L"tableLayoutPanel1");
            this->tableLayoutPanel1->Controls->Add(this->
                resetButton, 0, 0);
            this->tableLayoutPanel1->Controls->Add(this->
                startButton, 1, 0);
            this->tableLayoutPanel1->Controls->Add(this->
                closeButton, 2, 0);
            this->tableLayoutPanel1->Name = L"
                tableLayoutPanel1";
            //
            // resetButton
            //
            resources->ApplyResources(this->resetButton, L"
                resetButton");
            this->resetButton->Name = L"resetButton";
            this->resetButton->UseVisualStyleBackColor = true
                ;
            this->resetButton->Click += gcnew System::
                EventHandler(this, &StimulationConfig::
                resetButton_Click);
            //
            // startButton
            //
            resources->ApplyResources(this->startButton, L"
                startButton");
            this->startButton->DialogResult = System::Windows
                ::Forms::DialogResult::Yes;
            this->startButton->Name = L"startButton";
            this->startButton->UseVisualStyleBackColor = true
                ;
            this->startButton->Click += gcnew System::
                EventHandler(this, &StimulationConfig::
                startButton_Click);
            //
            // closeButton
            //
            resources->ApplyResources(this->closeButton, L"
                closeButton");
            this->closeButton->DialogResult = System::Windows
                ::Forms::DialogResult::Cancel;
            this->closeButton->Name = L"closeButton";
            this->closeButton->UseVisualStyleBackColor = true
                ;
            this->closeButton->Click += gcnew System::
                EventHandler(this, &StimulationConfig::
                close_Click);
            //
            // StimulationConfig
            //
            this->AcceptButton = this->startButton;
            resources->ApplyResources(this, L"$this");
            this->AutoScaleMode = System::Windows::Forms::
                AutoScaleMode::Font;
            this->CancelButton = this->closeButton;
```

```
                        this->Controls->Add(this->
                            StimulationSplitContainer);
                        this->FormBorderStyle = System::Windows::Forms::
                            FormBorderStyle::FixedDialog;
                        this->MaximizeBox = false;
                        this->MinimizeBox = false;
                        this->Name = L"StimulationConfig";
                        this->ShowInTaskbar = false;
                        this->Load += gcnew System::EventHandler(this, &
                            StimulationConfig::StimulationConfig_Load);
                        this->StimulationSplitContainer->Panel1->
                            ResumeLayout(false);
                        this->StimulationSplitContainer->Panel2->
                            ResumeLayout(false);
                        (cli::safe_cast<System::ComponentModel::
                            ISupportInitialize^>(this->
                            StimulationSplitContainer))->EndInit();
                        this->StimulationSplitContainer->ResumeLayout(
                            false);
                        this->tableLayoutPanel2->ResumeLayout(false);
                        this->tableLayoutPanel2->PerformLayout();
                        (cli::safe_cast<System::ComponentModel::
                            ISupportInitialize^>(this->relaxNumericUpDown)
                            )->EndInit();
                        (cli::safe_cast<System::ComponentModel::
                            ISupportInitialize^>(this->
                            MovementNumericUpDown))->EndInit();
                        (cli::safe_cast<System::ComponentModel::
                            ISupportInitialize^>(this->eventNbrUpDown3))->
                            EndInit();
                        (cli::safe_cast<System::ComponentModel::
                            ISupportInitialize^>(this->
                            breakPeriodNumericUpDown))->EndInit();
                        (cli::safe_cast<System::ComponentModel::
                            ISupportInitialize^>(this->
                            stimulationPictureBox))->EndInit();
                        this->tableLayoutPanel1->ResumeLayout(false);
                        this->ResumeLayout(false);

                }
#pragma endregion

                public:
                Stimulation^ getStimulation()
                {
                                        String^ subjectId = this->
                                            subjectIdTextBox->Text;
                                        int     eventNbr = Decimal::
                                            ToInt32(this->eventNbrUpDown3
                                            ->Value);
                                        int     relaxPeriod = Decimal::
                                            ToInt32(this->
                                            relaxNumericUpDown->Value);
                                        int     movementPeriod = Decimal
                                            ::ToInt32(this->
                                            MovementNumericUpDown->Value)
                                            ;
                                        int     movementType =
                                            movementComboBox->
```

```
                                                SelectedIndex;
                                            int     breakPeriod = Decimal::
                                                ToInt32(this->
                                                breakPeriodNumericUpDown->
                                                Value);

                                            return gcnew Stimulation(
                                                subjectId, eventNbr,
                                                relaxPeriod, movementPeriod,
                                                movementType, breakPeriod);
                    }


            private: System::Void startButton_Click(System::Object^  sender,
                System::EventArgs^  e) {
                                        this->Close();
            }
    private: System::Void close_Click(System::Object^  sender, System::
        EventArgs^  e) {
                                this->Close();
    }

    private: System::Void StimulationConfig_Load(System::Object^  sender,
        System::EventArgs^  e) {
    }
    private: System::Void resetButton_Click(System::Object^  sender, System::
        EventArgs^  e) {
                                subjectIdTextBox->ResetText();
                                eventNbrUpDown3->Value = 2;
                                relaxNumericUpDown->Value = 2;
                                MovementNumericUpDown->Value = 4;
                                movementComboBox->SelectedItem = "Move Left Arm"
                                    ;
                                breakPeriodNumericUpDown->Value = 2;
    }
    };
    }
```

Código 7.9: Cabecera para vista de gráficas de adquisición

```
#pragma once

namespace EEGRecording
{
        using namespace System;
        using namespace System::IO;
        using namespace std;
        using namespace System::Collections::Generic;
        using namespace System::Windows::Forms::DataVisualization::
            Charting;
        ref class Recording
        {
        public:
                Recording();
                ~Recording();
                void start(String^ file);
                List<array<DataPoint^>^>^ getDataPoints() { return
                    dataPoints; }
                void setDataPoints();
```

```
                void stop();
                bool isDone() { return done; }
                bool isStopped() { return stopped; }

                int getDatasetSize() { return datasetSize; }
                int getSampleRate() { return sampleRate; }
                void setSampleRate(int sampleRate) { this->sampleRate =
                    sampleRate; }
        private:
                StreamReader^ inputFile;

                List<array<DataPoint^>^>^ dataPoints;
                array<String^>^ SEPARATOR;
                bool done = false;
                bool stopped = false;

                int datasetSize = 32;
                int sampleRate = 4096;
        };
}
```

Código 7.10: Definición de funciones para visualización de adquisición

```
#include "Recording.h"
#include <stdio.h>
#include <stdlib.h>
#include <atlstr.h>

namespace EEGRecording
{
        using namespace System;

        Recording::Recording() {}
        void Recording::start(String^ file)
        {
                try
                {
                        inputFile = File::OpenText("Records\\" + file);
                        String^ header = inputFile->ReadLine();
                        SEPARATOR = gcnew array<String^>(1);
                        SEPARATOR[0] = ",";
                        stopped = false;
                }
                catch (Exception^ e){
                }

        }

        void Recording::setDataPoints()
        {
                delete[] dataPoints;
                dataPoints = gcnew List<array<DataPoint^>^>();
                double adjustY = atof("2.5");
                double adjustX;
                int k = 0;
                List<String^>^ lines = gcnew List<String^>();
                while (!inputFile->EndOfStream && k++ < sampleRate)
                {
                        lines->Add(inputFile->ReadLine());
```

```
}

array<DataPoint^>^ dp;
array<List<double>^>^ yPoints = gcnew array<List<double
    >^>(14);

int i = 0;

for each(String^ line in lines)
{
        if (i == 0)
        {   // Create a new group of data points
                dp = gcnew array<DataPoint^>(14);
                dataPoints->Add(dp);
        }
        array<String^>^ columns
                = line->Split(SEPARATOR, System::
                    StringSplitOptions::None);

        for (int j = 1; j < 15; j++)
        {
                if (i == 0)
                {   // Initialize one data point that
                    will hold 32 Y Points for each channel
                        double timestamp = atof(CString(
                            columns[15]));
                        adjustX = timestamp;
                        dp[j - 1] = gcnew  DataPoint();
                        dp[j - 1]->XValue = timestamp -
                            adjustX;
                        yPoints[j - 1] = gcnew List<
                            double>(datasetSize);
                }

                double signalValue = atof(CString(columns
                    [j]));
                yPoints[j - 1]->Add((signalValue*(adjustY
                     *j)) / 1000);
        }
        if (i < (datasetSize - 1))
        {
                i++;
        }
        else
        {
                for (int j = 0; j < 14; j++)
                {
                        dp[j]->YValues = yPoints[j]->
                            ToArray();
                        yPoints[j]->Clear();
                        yPoints[j] = nullptr;
                }

                i = 0;
                delete[] dp;

        }
}
```

```
                // Add remaining points if not module of 32
                if (i > 0)
                {
                        for (int j = 0; j < 14; j++)
                        {
                                dp[j]->YValues = yPoints[j]->ToArray();
                                yPoints[j]->Clear();
                                yPoints[j] = nullptr;
                        }
                }
                if (inputFile->EndOfStream)
                {
                        done = true;
                        stopped = true;
                }
        }

        void Recording::stop()
        {
                stopped = true;
                if (inputFile != nullptr)
                {
                        inputFile->Close();
                }
        }

        Recording::~Recording()
        {}
}
```

Código 7.11: Pantalla para visualización de datos de adquisición

```
#pragma once

#using <System.dll>
#using <System.Windows.Forms.dll>
#using <System.Drawing.dll>
#include "Recording.h"

namespace EEGRECORD {

        using namespace System;
        using namespace System::ComponentModel;
        using namespace System::Collections;
        using namespace System::Collections::Generic;
        using namespace System::Windows::Forms;
        using namespace System::IO;
        using namespace System::Data;
        using namespace System::Drawing;
        using namespace System::Windows::Forms::DataVisualization::
           Charting;


        using namespace EEGRecording;

        /// <summary>
        /// Summary for MyForm
        /// </summary>
```

```
public ref class RecordingBrowser : public System::Windows::Forms
    ::Form
{

        MenuStrip^ mainMenu1;
        String^ openFileName;
        String^ folderName;
        Recording^ recording = (gcnew Recording());
        array<Series^>^series = gcnew array<Series^>(14);

private: System::Windows::Forms::MenuStrip^
    recordingBrowserMenuStrip;
private: System::Windows::Forms::ToolStripMenuItem^
    fileToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^
    printToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^
    printPreviewToolStripMenuItem;
private: System::Windows::Forms::ToolStripSeparator^
    toolStripSeparator2;
private: System::Windows::Forms::ToolStripMenuItem^
    closeToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^
    toolsToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^
    customizeToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^
    optionsToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^
    helpToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^
    contentsToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^
    indexToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^
    searchToolStripMenuItem;
private: System::Windows::Forms::ToolStripSeparator^
    toolStripSeparator5;
private: System::Windows::Forms::ToolStripMenuItem^
    aboutToolStripMenuItem;
private: System::Windows::Forms::SplitContainer^  splitContainer1
    ;
private: System::Windows::Forms::ImageList^  csvImageList;
private: System::Windows::Forms::TableLayoutPanel^
    fileListTableLayoutPanel;
private: System::Windows::Forms::TextBox^  searchTextBox;
private: System::Windows::Forms::ListView^  csvFileListView;
private: System::Windows::Forms::ColumnHeader^
    fileNameColumnHeader;
private: System::Windows::Forms::DataVisualization::Charting::
    Chart^  recordingChart;
private: System::Windows::Forms::PrintDialog^
    recordingBrowserPrintDialog;
private: System::ComponentModel::BackgroundWorker^
    recordingBackgroundWorker;
private: System::Windows::Forms::Timer^  recordingTimer;
private: System::Windows::Forms::StatusStrip^  statusStrip;
private: System::Windows::Forms::ToolStripStatusLabel^
    toolStripStatusLabel;
```

```
        private: System::Windows::Forms::ToolStripProgressBar^
            toolStripProgressBar;
        private: System::Windows::Forms::ToolStripContainer^
            toolStripContainer1;
        private: System::Windows::Forms::CheckedListBox^
            eegChCheckedListBox;
        private: System::Windows::Forms::Button^  stopButton;
                         System::Windows::Forms::DataVisualization::
                            Charting::Series^  seriesa = (gcnew System::
                            Windows::Forms::DataVisualization::Charting::
                            Series());

        private: bool fileOpened;
        public:

                RecordingBrowser(String^ folderName)
                {
                        InitializeComponent();
                        fileOpened = false;
                        this->mainMenu1 = gcnew System::Windows::Forms::
                            MenuStrip;


                        this->ClientSize = System::Drawing::Size(296,
                            360);


                        this->MainMenuStrip = this->mainMenu1;
                        this->mainMenu1->ResumeLayout(false);
                        this->mainMenu1->PerformLayout();

                        this->Text = "Recording Browser";
                        String^ recordingDirectory = folderName->Concat(
                            folderName, Path::DirectorySeparatorChar);
                        recordingDirectory = recordingDirectory->Concat(
                            recordingDirectory, "Records");
                        recordingDirectory = recordingDirectory->Concat(
                            recordingDirectory, Path::
                            DirectorySeparatorChar);
                        for each (String^ fileName in Directory::
                            EnumerateFiles(recordingDirectory, "*.csv", IO
                            ::SearchOption::TopDirectoryOnly))
                        {
                                String^ shortFileName = fileName->
                                    Substring(fileName->LastIndexOf(Path::
                                    DirectorySeparatorChar) + 1);

                                csvFileListView->Items->Add(shortFileName
                                    )->ImageIndex = 0;
                                //csvFileListView->Items->
                        }
                        int i = 0;
                        for (Generic::IEnumerator<Series^>^ it = this->
                            recordingChart->Series->GetEnumerator(); it->
                            MoveNext();)
                        {
                                series[i++] = it->Current;
                        }
                }
```

```
        protected:
                /// <summary>
                /// Clean up any resources being used.
                /// </summary>
                ~RecordingBrowser()
                {
                        if (components)
                        {
                                delete components;
                        }
                }
        private: System::ComponentModel::IContainer^  components;
        protected:

        protected:

        protected:

        private:
                /// <summary>
                /// Required designer variable.
                /// </summary>

                // Bring up a dialog to open a file.

#pragma region Windows Form Designer generated code
                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                void InitializeComponent(void)
                {
                        this->components = (gcnew System::ComponentModel
                           ::Container());
                        System::ComponentModel::ComponentResourceManager^
                             resources = (gcnew System::ComponentModel::
                           ComponentResourceManager(RecordingBrowser::
                           typeid));
                        System::Windows::Forms::DataVisualization::
                           Charting::ChartArea^  chartArea1 = (gcnew
                           System::Windows::Forms::DataVisualization::
                           Charting::ChartArea());
                        System::Windows::Forms::DataVisualization::
                           Charting::Legend^  legend1 = (gcnew System::
                           Windows::Forms::DataVisualization::Charting::
                           Legend());
                        System::Windows::Forms::DataVisualization::
                           Charting::Series^  series1 = (gcnew System::
                           Windows::Forms::DataVisualization::Charting::
                           Series());
                        System::Windows::Forms::DataVisualization::
                           Charting::Series^  series2 = (gcnew System::
                           Windows::Forms::DataVisualization::Charting::
                           Series());
                        System::Windows::Forms::DataVisualization::
                           Charting::Series^  series3 = (gcnew System::
                           Windows::Forms::DataVisualization::Charting::
                           Series());
```

```
System::Windows::Forms::DataVisualization::
    Charting::Series^  series4 = (gcnew System::
    Windows::Forms::DataVisualization::Charting::
    Series());
System::Windows::Forms::DataVisualization::
    Charting::Series^  series5 = (gcnew System::
    Windows::Forms::DataVisualization::Charting::
    Series());
System::Windows::Forms::DataVisualization::
    Charting::Series^  series6 = (gcnew System::
    Windows::Forms::DataVisualization::Charting::
    Series());
System::Windows::Forms::DataVisualization::
    Charting::Series^  series7 = (gcnew System::
    Windows::Forms::DataVisualization::Charting::
    Series());
System::Windows::Forms::DataVisualization::
    Charting::Series^  series8 = (gcnew System::
    Windows::Forms::DataVisualization::Charting::
    Series());
System::Windows::Forms::DataVisualization::
    Charting::Series^  series9 = (gcnew System::
    Windows::Forms::DataVisualization::Charting::
    Series());
System::Windows::Forms::DataVisualization::
    Charting::Series^  series10 = (gcnew System::
    Windows::Forms::DataVisualization::Charting::
    Series());
System::Windows::Forms::DataVisualization::
    Charting::Series^  series11 = (gcnew System::
    Windows::Forms::DataVisualization::Charting::
    Series());
System::Windows::Forms::DataVisualization::
    Charting::Series^  series12 = (gcnew System::
    Windows::Forms::DataVisualization::Charting::
    Series());
System::Windows::Forms::DataVisualization::
    Charting::Series^  series13 = (gcnew System::
    Windows::Forms::DataVisualization::Charting::
    Series());
System::Windows::Forms::DataVisualization::
    Charting::Series^  series14 = (gcnew System::
    Windows::Forms::DataVisualization::Charting::
    Series());
System::Windows::Forms::DataVisualization::
    Charting::Title^  title1 = (gcnew System::
    Windows::Forms::DataVisualization::Charting::
    Title());
this->splitContainer1 = (gcnew System::Windows::
    Forms::SplitContainer());
this->fileListTableLayoutPanel = (gcnew System::
    Windows::Forms::TableLayoutPanel());
this->searchTextBox = (gcnew System::Windows::
    Forms::TextBox());
this->csvFileListView = (gcnew System::Windows::
    Forms::ListView());
this->fileNameColumnHeader = (gcnew System::
    Windows::Forms::ColumnHeader());
this->csvImageList = (gcnew System::Windows::
```

```
    Forms::ImageList(this->components));
this->stopButton = (gcnew System::Windows::Forms
    ::Button());
this->eegChCheckedListBox = (gcnew System::
    Windows::Forms::CheckedListBox());
this->recordingChart = (gcnew System::Windows::
    Forms::DataVisualization::Charting::Chart());
this->toolStripContainer1 = (gcnew System::
    Windows::Forms::ToolStripContainer());
this->statusStrip = (gcnew System::Windows::Forms
    ::StatusStrip());
this->toolStripStatusLabel = (gcnew System::
    Windows::Forms::ToolStripStatusLabel());
this->toolStripProgressBar = (gcnew System::
    Windows::Forms::ToolStripProgressBar());
this->recordingBrowserMenuStrip = (gcnew System::
    Windows::Forms::MenuStrip());
this->fileToolStripMenuItem = (gcnew System::
    Windows::Forms::ToolStripMenuItem());
this->printToolStripMenuItem = (gcnew System::
    Windows::Forms::ToolStripMenuItem());
this->printPreviewToolStripMenuItem = (gcnew
    System::Windows::Forms::ToolStripMenuItem());
this->toolStripSeparator2 = (gcnew System::
    Windows::Forms::ToolStripSeparator());
this->closeToolStripMenuItem = (gcnew System::
    Windows::Forms::ToolStripMenuItem());
this->toolsToolStripMenuItem = (gcnew System::
    Windows::Forms::ToolStripMenuItem());
this->customizeToolStripMenuItem = (gcnew System
    ::Windows::Forms::ToolStripMenuItem());
this->optionsToolStripMenuItem = (gcnew System::
    Windows::Forms::ToolStripMenuItem());
this->helpToolStripMenuItem = (gcnew System::
    Windows::Forms::ToolStripMenuItem());
this->contentsToolStripMenuItem = (gcnew System::
    Windows::Forms::ToolStripMenuItem());
this->indexToolStripMenuItem = (gcnew System::
    Windows::Forms::ToolStripMenuItem());
this->searchToolStripMenuItem = (gcnew System::
    Windows::Forms::ToolStripMenuItem());
this->toolStripSeparator5 = (gcnew System::
    Windows::Forms::ToolStripSeparator());
this->aboutToolStripMenuItem = (gcnew System::
    Windows::Forms::ToolStripMenuItem());
this->recordingBrowserPrintDialog = (gcnew System
    ::Windows::Forms::PrintDialog());
this->recordingBackgroundWorker = (gcnew System::
    ComponentModel::BackgroundWorker());
this->recordingTimer = (gcnew System::Windows::
    Forms::Timer(this->components));
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->splitContainer1))->
    BeginInit();
this->splitContainer1->Panel1->SuspendLayout();
this->splitContainer1->Panel2->SuspendLayout();
this->splitContainer1->SuspendLayout();
this->fileListTableLayoutPanel->SuspendLayout();
(cli::safe_cast<System::ComponentModel::
```

```
        ISupportInitialize^>(this->recordingChart))->
        BeginInit();
    this->toolStripContainer1->BottomToolStripPanel->
        SuspendLayout();
    this->toolStripContainer1->ContentPanel->
        SuspendLayout();
    this->toolStripContainer1->TopToolStripPanel->
        SuspendLayout();
    this->toolStripContainer1->SuspendLayout();
    this->statusStrip->SuspendLayout();
    this->recordingBrowserMenuStrip->SuspendLayout();
    this->SuspendLayout();
    //
    // splitContainer1
    //
    resources->ApplyResources(this->splitContainer1,
        L"splitContainer1");
    this->splitContainer1->FixedPanel = System::
        Windows::Forms::FixedPanel::Panel1;
    this->splitContainer1->Name = L"splitContainer1";
    //
    // splitContainer1.Panel1
    //
    this->splitContainer1->Panel1->Controls->Add(this
        ->fileListTableLayoutPanel);
    //
    // splitContainer1.Panel2
    //
    resources->ApplyResources(this->splitContainer1->
        Panel2, L"splitContainer1.Panel2");
    this->splitContainer1->Panel2->Controls->Add(this
        ->recordingChart);
    //
    // fileListTableLayoutPanel
    //
    resources->ApplyResources(this->
        fileListTableLayoutPanel, L"
        fileListTableLayoutPanel");
    this->fileListTableLayoutPanel->Controls->Add(
        this->searchTextBox, 0, 0);
    this->fileListTableLayoutPanel->Controls->Add(
        this->csvFileListView, 0, 1);
    this->fileListTableLayoutPanel->Controls->Add(
        this->stopButton, 0, 2);
    this->fileListTableLayoutPanel->Controls->Add(
        this->eegChCheckedListBox, 0, 3);
    this->fileListTableLayoutPanel->Name = L"
        fileListTableLayoutPanel";
    this->fileListTableLayoutPanel->Paint += gcnew
        System::Windows::Forms::PaintEventHandler(this
        , &RecordingBrowser::tableLayoutPanel1_Paint);
    //
    // searchTextBox
    //
    resources->ApplyResources(this->searchTextBox, L"
        searchTextBox");
    this->searchTextBox->ForeColor = System::Drawing
        ::SystemColors::ControlLight;
    this->searchTextBox->Name = L"searchTextBox";
```

```
            this->searchTextBox->Enter += gcnew System::
                EventHandler(this, &RecordingBrowser::
                searchTextBox_enterFocus);
            this->searchTextBox->KeyDown += gcnew System::
                Windows::Forms::KeyEventHandler(this, &
                RecordingBrowser::searchTextBox_KeyDown);
            this->searchTextBox->KeyPress += gcnew System::
                Windows::Forms::KeyPressEventHandler(this, &
                RecordingBrowser::searchTextBox_KeyPress);
            //
            // csvFileListView
            //
            this->csvFileListView->BorderStyle = System::
                Windows::Forms::BorderStyle::None;
            this->csvFileListView->Columns->AddRange(gcnew
                cli::array< System::Windows::Forms::
                ColumnHeader^ >(1) { this->
                fileNameColumnHeader });
            resources->ApplyResources(this->csvFileListView,
                L"csvFileListView");
            this->csvFileListView->FullRowSelect = true;
            this->csvFileListView->HeaderStyle = System::
                Windows::Forms::ColumnHeaderStyle::None;
            this->csvFileListView->HideSelection = false;
            this->csvFileListView->LargeImageList = this->
                csvImageList;
            this->csvFileListView->MultiSelect = false;
            this->csvFileListView->Name = L"csvFileListView";
            this->csvFileListView->SmallImageList = this->
                csvImageList;
            this->csvFileListView->
                UseCompatibleStateImageBehavior = false;
            this->csvFileListView->View = System::Windows::
                Forms::View::Details;
            this->csvFileListView->SelectedIndexChanged +=
                gcnew System::EventHandler(this, &
                RecordingBrowser::
                csvFileListView_SelectedIndexChanged);
            this->csvFileListView->KeyDown += gcnew System::
                Windows::Forms::KeyEventHandler(this, &
                RecordingBrowser::enter);
            this->csvFileListView->MouseDoubleClick += gcnew
                System::Windows::Forms::MouseEventHandler(this
                , &RecordingBrowser::
                csvFileListView_MouseDoubleClick);
            //
            // fileNameColumnHeader
            //
            resources->ApplyResources(this->
                fileNameColumnHeader, L"fileNameColumnHeader")
                ;
            //
            // csvImageList
            //
            this->csvImageList->ImageStream = (cli::safe_cast
                <System::Windows::Forms::ImageListStreamer^>(
                resources->GetObject(L"csvImageList.
                ImageStream")));
            this->csvImageList->TransparentColor = System::
```

```
            Drawing::Color::Transparent;
        this->csvImageList->Images->SetKeyName(0, L"csv.
            ico");
        //
        // stopButton
        //
        resources->ApplyResources(this->stopButton, L"
            stopButton");
        this->stopButton->Name = L"stopButton";
        this->stopButton->UseVisualStyleBackColor = true;
        this->stopButton->Click += gcnew System::
            EventHandler(this, &RecordingBrowser::
            stopButton_Click);
        //
        // eegChCheckedListBox
        //
        this->eegChCheckedListBox->CheckOnClick = true;
        resources->ApplyResources(this->
            eegChCheckedListBox, L"eegChCheckedListBox");
        this->eegChCheckedListBox->FormattingEnabled =
            true;
        this->eegChCheckedListBox->Items->AddRange(gcnew
            cli::array< System::Object^ >(14) {
                resources->GetString(L"
                    eegChCheckedListBox.Items"),
                        resources->GetString(L"
                            eegChCheckedListBox.Items1"),
                            resources->GetString(L"
                            eegChCheckedListBox.Items2"),
                            resources->GetString(L"
                            eegChCheckedListBox.Items3"),
                        resources->GetString(L"
                            eegChCheckedListBox.Items4"),
                            resources->GetString(L"
                            eegChCheckedListBox.Items5"),
                            resources->GetString(L"
                            eegChCheckedListBox.Items6"),
                        resources->GetString(L"
                            eegChCheckedListBox.Items7"),
                            resources->GetString(L"
                            eegChCheckedListBox.Items8"),
                            resources->GetString(L"
                            eegChCheckedListBox.Items9"),
                        resources->GetString(L"
                            eegChCheckedListBox.Items10"),
                             resources->GetString(L"
                            eegChCheckedListBox.Items11"),
                             resources->GetString(L"
                            eegChCheckedListBox.Items12"),
                        resources->GetString(L"
                            eegChCheckedListBox.Items13")
        });
        this->eegChCheckedListBox->MultiColumn = true;
        this->eegChCheckedListBox->Name = L"
            eegChCheckedListBox";
        this->eegChCheckedListBox->ThreeDCheckBoxes =
            true;
        //
        // recordingChart
```

```
            //
            this->recordingChart->BorderSkin->SkinStyle =
                System::Windows::Forms::DataVisualization::
                Charting::BorderSkinStyle::Sunken;
            this->recordingChart->CausesValidation = false;
            chartArea1->AxisY->LabelStyle->Enabled = false;
            chartArea1->AxisY->MajorGrid->Enabled = false;
            chartArea1->AxisY->MajorTickMark->Enabled = false
                ;
            chartArea1->Name = L"chartArea";
            this->recordingChart->ChartAreas->Add(chartArea1)
                ;
            resources->ApplyResources(this->recordingChart, L
                "recordingChart");
            legend1->DockedToChartArea = L"chartArea";
            legend1->Docking = System::Windows::Forms::
                DataVisualization::Charting::Docking::Left;
            legend1->IsDockedInsideChartArea = false;
            legend1->Name = L"Legend1";
            this->recordingChart->Legends->Add(legend1);
            this->recordingChart->Name = L"recordingChart";
            this->recordingChart->Palette = System::Windows::
                Forms::DataVisualization::Charting::
                ChartColorPalette::EarthTones;
            series1->ChartArea = L"chartArea";
            series1->ChartType = System::Windows::Forms::
                DataVisualization::Charting::SeriesChartType::
                Spline;
            series1->Legend = L"Legend1";
            series1->Name = L"AF3";
            series1->YValuesPerPoint = 32;
            series2->ChartArea = L"chartArea";
            series2->ChartType = System::Windows::Forms::
                DataVisualization::Charting::SeriesChartType::
                Spline;
            series2->Legend = L"Legend1";
            series2->Name = L"F7";
            series2->YValuesPerPoint = 32;
            series3->ChartArea = L"chartArea";
            series3->ChartType = System::Windows::Forms::
                DataVisualization::Charting::SeriesChartType::
                Spline;
            series3->Legend = L"Legend1";
            series3->Name = L"F3";
            series3->YValuesPerPoint = 32;
            series4->ChartArea = L"chartArea";
            series4->ChartType = System::Windows::Forms::
                DataVisualization::Charting::SeriesChartType::
                Spline;
            series4->Legend = L"Legend1";
            series4->Name = L"FC5";
            series4->YValuesPerPoint = 32;
            series5->ChartArea = L"chartArea";
            series5->ChartType = System::Windows::Forms::
                DataVisualization::Charting::SeriesChartType::
                Spline;
            series5->Legend = L"Legend1";
            series5->Name = L"T7";
            series5->YValuesPerPoint = 32;
```

```
series6->ChartArea = L"chartArea";
series6->ChartType = System::Windows::Forms::
    DataVisualization::Charting::SeriesChartType::
    Spline;
series6->Legend = L"Legend1";
series6->Name = L"P7";
series6->YValuesPerPoint = 32;
series7->ChartArea = L"chartArea";
series7->ChartType = System::Windows::Forms::
    DataVisualization::Charting::SeriesChartType::
    Spline;
series7->Legend = L"Legend1";
series7->Name = L"O1";
series7->YValuesPerPoint = 32;
series8->ChartArea = L"chartArea";
series8->ChartType = System::Windows::Forms::
    DataVisualization::Charting::SeriesChartType::
    Spline;
series8->Legend = L"Legend1";
series8->Name = L"O2";
series8->YValuesPerPoint = 32;
series9->ChartArea = L"chartArea";
series9->ChartType = System::Windows::Forms::
    DataVisualization::Charting::SeriesChartType::
    Spline;
series9->Legend = L"Legend1";
series9->Name = L"P8";
series9->YValuesPerPoint = 32;
series10->ChartArea = L"chartArea";
series10->ChartType = System::Windows::Forms::
    DataVisualization::Charting::SeriesChartType::
    Spline;
series10->Legend = L"Legend1";
series10->Name = L"T8";
series10->YValuesPerPoint = 32;
series11->ChartArea = L"chartArea";
series11->ChartType = System::Windows::Forms::
    DataVisualization::Charting::SeriesChartType::
    Spline;
series11->Legend = L"Legend1";
series11->Name = L"FC6";
series11->YValuesPerPoint = 32;
series12->ChartArea = L"chartArea";
series12->ChartType = System::Windows::Forms::
    DataVisualization::Charting::SeriesChartType::
    Spline;
series12->Legend = L"Legend1";
series12->Name = L"F4";
series12->YValuesPerPoint = 32;
series13->ChartArea = L"chartArea";
series13->ChartType = System::Windows::Forms::
    DataVisualization::Charting::SeriesChartType::
    Spline;
series13->Legend = L"Legend1";
series13->Name = L"F8";
series13->YValuesPerPoint = 32;
series14->ChartArea = L"chartArea";
series14->ChartType = System::Windows::Forms::
    DataVisualization::Charting::SeriesChartType::
```

```
    Spline;
series14->Legend = L"Legend1";
series14->Name = L"AF4";
series14->YValuesPerPoint = 32;
this->recordingChart->Series->Add(series1);
this->recordingChart->Series->Add(series2);
this->recordingChart->Series->Add(series3);
this->recordingChart->Series->Add(series4);
this->recordingChart->Series->Add(series5);
this->recordingChart->Series->Add(series6);
this->recordingChart->Series->Add(series7);
this->recordingChart->Series->Add(series8);
this->recordingChart->Series->Add(series9);
this->recordingChart->Series->Add(series10);
this->recordingChart->Series->Add(series11);
this->recordingChart->Series->Add(series12);
this->recordingChart->Series->Add(series13);
this->recordingChart->Series->Add(series14);
title1->IsDockedInsideChartArea = false;
title1->Name = L"Raw EEG ";
this->recordingChart->Titles->Add(title1);
this->recordingChart->Click += gcnew System::
    EventHandler(this, &RecordingBrowser::
    recordingChart_Click);
//
// toolStripContainer1
//
//
// toolStripContainer1.BottomToolStripPanel
//
this->toolStripContainer1->BottomToolStripPanel->
    Controls->Add(this->statusStrip);
//
// toolStripContainer1.ContentPanel
//
resources->ApplyResources(this->
    toolStripContainer1->ContentPanel, L"
    toolStripContainer1.ContentPanel");
this->toolStripContainer1->ContentPanel->Controls
    ->Add(this->splitContainer1);
resources->ApplyResources(this->
    toolStripContainer1, L"toolStripContainer1");
this->toolStripContainer1->
    LeftToolStripPanelVisible = false;
this->toolStripContainer1->Name = L"
    toolStripContainer1";
this->toolStripContainer1->
    RightToolStripPanelVisible = false;
//
// toolStripContainer1.TopToolStripPanel
//
this->toolStripContainer1->TopToolStripPanel->
    Controls->Add(this->recordingBrowserMenuStrip)
    ;
//
// statusStrip
//
resources->ApplyResources(this->statusStrip, L"
    statusStrip");
```

```
this->statusStrip->Items->AddRange(gcnew cli::
    array< System::Windows::Forms::ToolStripItem^
     >(2) {
         this->toolStripStatusLabel,
                 this->toolStripProgressBar
});
this->statusStrip->Name = L"statusStrip";
this->statusStrip->RenderMode = System::Windows::
    Forms::ToolStripRenderMode::Professional;
//
// toolStripStatusLabel
//
resources->ApplyResources(this->
    toolStripStatusLabel, L"toolStripStatusLabel")
    ;
this->toolStripStatusLabel->Name = L"
    toolStripStatusLabel";
//
// toolStripProgressBar
//
this->toolStripProgressBar->Alignment = System::
    Windows::Forms::ToolStripItemAlignment::Right;
this->toolStripProgressBar->MarqueeAnimationSpeed
     = 10;
this->toolStripProgressBar->Maximum = 1000;
this->toolStripProgressBar->Name = L"
    toolStripProgressBar";
resources->ApplyResources(this->
    toolStripProgressBar, L"toolStripProgressBar")
    ;
this->toolStripProgressBar->Step = 1;
//
// recordingBrowserMenuStrip
//
resources->ApplyResources(this->
    recordingBrowserMenuStrip, L"
    recordingBrowserMenuStrip");
this->recordingBrowserMenuStrip->Items->AddRange(
    gcnew cli::array< System::Windows::Forms::
    ToolStripItem^  >(3) {
         this->fileToolStripMenuItem,
                 this->toolsToolStripMenuItem,
                    this->helpToolStripMenuItem
});
this->recordingBrowserMenuStrip->Name = L"
    recordingBrowserMenuStrip";
//
// fileToolStripMenuItem
//
this->fileToolStripMenuItem->DropDownItems->
    AddRange(gcnew cli::array< System::Windows::
    Forms::ToolStripItem^  >(4) {
         this->printToolStripMenuItem,
                 this->
                    printPreviewToolStripMenuItem,
                     this->toolStripSeparator2,
                    this->closeToolStripMenuItem
});
this->fileToolStripMenuItem->Name = L"
```

```
            fileToolStripMenuItem";
        resources->ApplyResources(this->
            fileToolStripMenuItem, L"fileToolStripMenuItem
            ");
        this->fileToolStripMenuItem->Click += gcnew
            System::EventHandler(this, &RecordingBrowser::
            fileToolStripMenuItem_Click_1);
        //
        // printToolStripMenuItem
        //
        resources->ApplyResources(this->
            printToolStripMenuItem, L"
            printToolStripMenuItem");
        this->printToolStripMenuItem->Name = L"
            printToolStripMenuItem";
        //
        // printPreviewToolStripMenuItem
        //
        resources->ApplyResources(this->
            printPreviewToolStripMenuItem, L"
            printPreviewToolStripMenuItem");
        this->printPreviewToolStripMenuItem->Name = L"
            printPreviewToolStripMenuItem";
        //
        // toolStripSeparator2
        //
        this->toolStripSeparator2->Name = L"
            toolStripSeparator2";
        resources->ApplyResources(this->
            toolStripSeparator2, L"toolStripSeparator2");
        //
        // closeToolStripMenuItem
        //
        this->closeToolStripMenuItem->Name = L"
            closeToolStripMenuItem";
        resources->ApplyResources(this->
            closeToolStripMenuItem, L"
            closeToolStripMenuItem");
        this->closeToolStripMenuItem->Click += gcnew
            System::EventHandler(this, &RecordingBrowser::
            closeToolStripMenuItem_Click);
        //
        // toolsToolStripMenuItem
        //
        this->toolsToolStripMenuItem->DropDownItems->
            AddRange(gcnew cli::array< System::Windows::
            Forms::ToolStripItem^  >(2) {
                this->customizeToolStripMenuItem,
                        this->optionsToolStripMenuItem
        });
        this->toolsToolStripMenuItem->Name = L"
            toolsToolStripMenuItem";
        resources->ApplyResources(this->
            toolsToolStripMenuItem, L"
            toolsToolStripMenuItem");
        this->toolsToolStripMenuItem->Click += gcnew
            System::EventHandler(this, &RecordingBrowser::
            toolsToolStripMenuItem_Click);
        //
```

```
            // customizeToolStripMenuItem
            //
            this->customizeToolStripMenuItem->Name = L"
                customizeToolStripMenuItem";
            resources->ApplyResources(this->
                customizeToolStripMenuItem, L"
                customizeToolStripMenuItem");
            //
            // optionsToolStripMenuItem
            //
            this->optionsToolStripMenuItem->Name = L"
                optionsToolStripMenuItem";
            resources->ApplyResources(this->
                optionsToolStripMenuItem, L"
                optionsToolStripMenuItem");
            //
            // helpToolStripMenuItem
            //
            this->helpToolStripMenuItem->DropDownItems->
                AddRange(gcnew cli::array< System::Windows::
                Forms::ToolStripItem^ >(5) {
                    this->contentsToolStripMenuItem,
                            this->indexToolStripMenuItem,
                                this->searchToolStripMenuItem,
                                 this->toolStripSeparator5,
                                this->aboutToolStripMenuItem
            });
            this->helpToolStripMenuItem->Name = L"
                helpToolStripMenuItem";
            resources->ApplyResources(this->
                helpToolStripMenuItem, L"helpToolStripMenuItem
                ");
            this->helpToolStripMenuItem->Click += gcnew
                System::EventHandler(this, &RecordingBrowser::
                helpToolStripMenuItem_Click);
            //
            // contentsToolStripMenuItem
            //
            this->contentsToolStripMenuItem->Name = L"
                contentsToolStripMenuItem";
            resources->ApplyResources(this->
                contentsToolStripMenuItem, L"
                contentsToolStripMenuItem");
            //
            // indexToolStripMenuItem
            //
            this->indexToolStripMenuItem->Name = L"
                indexToolStripMenuItem";
            resources->ApplyResources(this->
                indexToolStripMenuItem, L"
                indexToolStripMenuItem");
            //
            // searchToolStripMenuItem
            //
            this->searchToolStripMenuItem->Name = L"
                searchToolStripMenuItem";
            resources->ApplyResources(this->
                searchToolStripMenuItem, L"
                searchToolStripMenuItem");
```

```
            //
            // toolStripSeparator5
            //
            this->toolStripSeparator5->Name = L"
                toolStripSeparator5";
            resources->ApplyResources(this->
                toolStripSeparator5, L"toolStripSeparator5");
            //
            // aboutToolStripMenuItem
            //
            this->aboutToolStripMenuItem->Name = L"
                aboutToolStripMenuItem";
            resources->ApplyResources(this->
                aboutToolStripMenuItem, L"
                aboutToolStripMenuItem");
            //
            // recordingBrowserPrintDialog
            //
            this->recordingBrowserPrintDialog->UseEXDialog =
                true;
            //
            // recordingBackgroundWorker
            //
            this->recordingBackgroundWorker->
                WorkerSupportsCancellation = true;
            this->recordingBackgroundWorker->DoWork += gcnew
                System::ComponentModel::DoWorkEventHandler(
                this, &RecordingBrowser::
                recordingBackgroundWorker_DoWork);
            this->recordingBackgroundWorker->
                RunWorkerCompleted += gcnew System::
                ComponentModel::RunWorkerCompletedEventHandler
                (this, &RecordingBrowser::
                recordingBackgroundWorker_RunWorkerCompleted);
            //
            // recordingTimer
            //
            this->recordingTimer->Interval = 50;
            this->recordingTimer->Tick += gcnew System::
                EventHandler(this, &RecordingBrowser::
                recordingTimer_Tick);
            //
            // RecordingBrowser
            //
            resources->ApplyResources(this, L"$this");
            this->AutoScaleMode = System::Windows::Forms::
                AutoScaleMode::Font;
            this->Controls->Add(this->toolStripContainer1);
            this->FormBorderStyle = System::Windows::Forms::
                FormBorderStyle::Fixed3D;
            this->Name = L"RecordingBrowser";
            this->ShowIcon = false;
            this->ShowInTaskbar = false;
            this->WindowState = System::Windows::Forms::
                FormWindowState::Maximized;
            this->Load += gcnew System::EventHandler(this, &
                RecordingBrowser::SelectRecording_Load);
            this->Shown += gcnew System::EventHandler(this, &
                RecordingBrowser::RecordingBrowser_Shown);
```

```
                        this->splitContainer1->Panel1->ResumeLayout(false
                            );
                        this->splitContainer1->Panel2->ResumeLayout(false
                            );
                        (cli::safe_cast<System::ComponentModel::
                            ISupportInitialize^>(this->splitContainer1))->
                            EndInit();
                        this->splitContainer1->ResumeLayout(false);
                        this->fileListTableLayoutPanel->ResumeLayout(
                            false);
                        this->fileListTableLayoutPanel->PerformLayout();
                        (cli::safe_cast<System::ComponentModel::
                            ISupportInitialize^>(this->recordingChart))->
                            EndInit();
                        this->toolStripContainer1->BottomToolStripPanel->
                            ResumeLayout(false);
                        this->toolStripContainer1->BottomToolStripPanel->
                            PerformLayout();
                        this->toolStripContainer1->ContentPanel->
                            ResumeLayout(false);
                        this->toolStripContainer1->TopToolStripPanel->
                            ResumeLayout(false);
                        this->toolStripContainer1->TopToolStripPanel->
                            PerformLayout();
                        this->toolStripContainer1->ResumeLayout(false);
                        this->toolStripContainer1->PerformLayout();
                        this->statusStrip->ResumeLayout(false);
                        this->statusStrip->PerformLayout();
                        this->recordingBrowserMenuStrip->ResumeLayout(
                            false);
                        this->recordingBrowserMenuStrip->PerformLayout();
                        this->ResumeLayout(false);

                }
#pragma endregion

        private: System::Void SelectRecording_Load(System::Object^
            sender, System::EventArgs^  e) {
                                recordingTimer->Stop();


        }
        private: System::Void folderBrowserDialog1_HelpRequest(System::
            Object^  sender, System::EventArgs^  e) {
        }
        private: System::Void menuStrip1_ItemClicked(System::Object^
            sender, System::Windows::Forms::ToolStripItemClickedEventArgs^
              e) {
        }
        private: System::Void openToolStripMenuItem_Click(System::Object^
             sender, System::EventArgs^  e) {
        }
        private: System::Void fileToolStripMenuItem_Click(System::Object^
             sender, System::EventArgs^  e) {
        }
        private: System::Void selectDirectoryToolStripMenuItem_Click(
            System::Object^  sender, System::EventArgs^  e) {
        }
        private: System::Void helpToolStripMenuItem_Click(System::Object^
```

```
        sender, System::EventArgs^  e) {
}
private: System::Void toolsToolStripMenuItem_Click(System::Object
    ^  sender, System::EventArgs^  e) {
}
private: System::Void fileToolStripMenuItem_Click_1(System::
    Object^  sender, System::EventArgs^  e) {
}
private: System::Void openToolStripMenuItem_Click_1(System::
    Object^  sender, System::EventArgs^  e) {
}

public:
        Void setFolderName(String^ folderName)
        {
                this->folderName = folderName;
        }

private: System::Void closeToolStripMenuItem_Click(System::Object
    ^  sender, System::EventArgs^  e) {
                        this->Close();
}

private: System::Void tableLayoutPanel1_Paint(System::Object^
    sender, System::Windows::Forms::PaintEventArgs^  e) {
}

private: System::Void csvFileListView_SelectedIndexChanged(System
    ::Object^  sender, System::EventArgs^  e) {
                        recordingTimer->Stop();
                        if (recording != nullptr )
                        {
                                recording->stop();
                                delete recording;
                                toolStripProgressBar->Value = 0;
                                toolStripProgressBar->Visible =
                                    false;
                                toolStripStatusLabel->Text = "";
                                recordingChart->SuspendLayout();
                                for (int i = 0; i < 14; i++)
                                {
                                        series[i]->Points->Clear
                                            ();
                                }
                                recordingChart->ResumeLayout(
                                    true);
                        }

}
Void search()
{
        if (!String::IsNullOrEmpty(searchTextBox->Text) &&
                searchTextBox->Text->CompareTo("Search File") !=
                    0)
        { // Call FindItemWithText with the contents of the
            textbox.
                ListViewItem^ foundItem =
                        csvFileListView->FindItemWithText(
                            searchTextBox->Text, false, 0, true);
```

```
                        if (foundItem != nullptr)
                        {
                                foundItem->Focused = true;
                                foundItem->Selected = true;
                                csvFileListView->TopItem = foundItem;
                                csvFileListView->Focus();
                                openFileName = foundItem->Text;
                                recording->start(openFileName);

                        }
                        this->searchTextBox->ForeColor = System::Drawing
                            ::SystemColors::ControlLight;
                        this->searchTextBox->Text = "Search File";
                }
        }

private: System::Void searchTextBox_enterFocus(System::Object^ sender,
    System::EventArgs^ e){
                         this->searchTextBox->Text = "";
                                this->searchTextBox->ForeColor = System
                                    ::Drawing::SystemColors::WindowText;
        }

private: System::Void enter(System::Object^  sender, System::Windows::
    Forms::KeyEventArgs^  e) {

        }
private: System::Void searchTextBox_KeyPress(System::Object^  sender,
    System::Windows::Forms::KeyPressEventArgs^  e){

}

private: System::Void searchTextBox_KeyDown(System::Object^  sender,
    System::Windows::Forms::KeyEventArgs^  e) {
                        if ((e->KeyCode.CompareTo(System::Windows::Forms
                            ::Keys::Enter)) == 0)
                        {
                                search();
                        }
}

private: System::Void recordingBackgroundWorker_DoWork(System::Object^
    sender, System::ComponentModel::DoWorkEventArgs^  e)
{
                        if (recording != nullptr)
                        {
                                recordingTimer->Stop();
                                recording->setDataPoints();
                                if (recording->isStopped())
                                {
                                        recording->stop();
                                }
                        }
}
private: System::Void recordingTimer_Tick(System::Object^  sender, System
    ::EventArgs^  e) {
                        if (!recordingBackgroundWorker->IsBusy && !
                            recording->isStopped())
```

```
                        {
                                recordingBackgroundWorker->
                                    RunWorkerAsync();
                        }
                        else
                        {
                                recordingTimer->Stop();
                                recordingBackgroundWorker->CancelAsync()
                                    ;
                        }
}
private: System::Void recordingBackgroundWorker_RunWorkerCompleted(System
    ::Object^  sender, System::ComponentModel::RunWorkerCompletedEventArgs
    ^  e) {

                        List<array<DataPoint^>^>^ dataPoints = recording
                            ->getDataPoints();
                        int datasetSize = recording->getDatasetSize();

                        for each (array<DataPoint^>^ samples in
                            dataPoints )
                        {
                                System::Collections::Generic::
                                    IEnumerator<Series^>^ ie =
                                    recordingChart->Series->GetEnumerator
                                    ();
                                if (samples[0]->YValues->Length <
                                    datasetSize)
                                {
                                        while (ie->MoveNext())
                                        {
                                                ie->Current->
                                                    YValuesPerPoint =
                                                    samples[0]->YValues->
                                                    Length;
                                        }
                                }

                                for (int i = 0; i < 14; i++)
                                {
                                        if (eegChCheckedListBox->
                                            GetItemChecked(i))
                                        {
                                                series[i]->Points->Add(
                                                    samples[i]);
                                        }
                                        else
                                        {
                                                series[i]->Points->Clear
                                                    ();
                                        }
                                }
                        }

                        if (!recording->isDone())
                        {
                                toolStripStatusLabel->Text = "Loading...
                                    ";
                                if (toolStripProgressBar->Value >= 1000)
```

```
                                {
                                        toolStripProgressBar->Value =
                                           500;
                                }
                                toolStripProgressBar->PerformStep();
                                recordingTimer->Start();
                        }
                        else
                        {
                                toolStripProgressBar->Value = 1000;
                                toolStripStatusLabel->Text = "Done";
                                recordingTimer->Stop();
                                recordingChart->UseWaitCursor = false;
                                recordingChart->Visible = true;
                        }
}
private: System::Void RecordingBrowser_Shown(System::Object^  sender,
    System::EventArgs^  e) {

                        toolStripProgressBar->ProgressBar->Width =
                           statusStrip->Width - 90;
                        toolStripProgressBar->Visible = false;
                        toolStripStatusLabel->Text = "";
                        for (int i = 0; i < eegChCheckedListBox->Items->
                           Count; i++)
                        {
                                eegChCheckedListBox->SetItemChecked(i,
                                   true);
                        }
}
private: System::Void stopButton_Click(System::Object^  sender, System::
    EventArgs^  e) {
                        recordingTimer->Stop();
                        if (recording != nullptr)
                        recording->stop();
                        toolStripProgressBar->Value = 1000;
                        toolStripStatusLabel->Text = "Stopped";
}
private: System::Void csvFileListView_MouseDoubleClick(System::Object^
    sender, System::Windows::Forms::MouseEventArgs^  e) {

                        recordingChart->SuspendLayout();
                        for (int i = 0; i < 14; i++)
                        {
                                series[i]->Points->Clear();
                        }
                        //recordingChart->ChartAreas->FindByName(""); ?
                           Sigue aqui
                        recordingChart->ResumeLayout(true);
                        recordingChart->Visible = false;
                        recordingChart->UseWaitCursor = true;
                        if (recordingBackgroundWorker->IsBusy)
                           recordingBackgroundWorker->CancelAsync();
                        recordingTimer->Stop();
                        if (recording != nullptr) delete recording;
                        recording = gcnew Recording();
                        Collections::IEnumerator^ it = csvFileListView->
                           SelectedItems->GetEnumerator();
                        it->MoveNext();
```

```
                        ListViewItem^ currFile = (ListViewItem^)it->
                            Current;
                        openFileName = currFile->Text;
                        recording->start(openFileName);

                        recordingTimer->Start();
                        toolStripProgressBar->Visible = true;
                        toolStripStatusLabel->Text = "Loading...";

}
private: System::Void recordingChart_Click(System::Object^  sender,
    System::EventArgs^  e) {
}
};
}
```

Código 7.12: Cabecera con funciones para extracción

```
#pragma once
#include <map>

#include <msclr/marshal_cppstd.h>
namespace EEGExtraction
{
        using namespace System;
        using namespace System::IO;
        using namespace std;
        using namespace System::Collections;
        using namespace System::Collections::Generic;
        using namespace System::Windows::Forms::DataVisualization::
            Charting;
        using namespace System::Drawing;

        ref class Extraction
        {
            public:
                    Extraction();
                        ~Extraction();
                        void start(String^ file, String^ extractionMethod
                            );

                        void stop();
                        bool isDone() { return done; }
                        bool isStopped() { return stopped; }
                        int getDatasetSize() { return datasetSize; }
                        void extract(String^ extractionMethod);
                        void graph(String^ fileName);
                        List<array<DataPoint^>^>^ getDataPoints() {
                            return dataPoints; }
                        String^ getOutFile() { return outFile; }
            private:
                    StreamReader^ inputFile;
                        bool done = false;
                        bool stopped = false;
                        array<String^>^ SEPARATOR;
                        int datasetSize = 1;
                        int sampleRate = 4096;
                        List<double>^ f3Channel;
                        List<double>^ fc5Channel;
```

```
                        List<double>^ fc6Channel;
                        List<double>^ f4Channel;
                        List<String^>^ f3Wave;
                        List<String^>^ fc5Wave;
                        List<String^>^ fc6Wave;
                        List<String^>^ f4Wave;
                        List<String^>^ f3Time;
                        List<String^>^ fc5Time;
                        List<String^>^ fc6Time;
                        List<String^>^ f4Time;
                        StreamWriter^ outputFile;
                        String^ inFile;
                        String^  outFile;
                        int numSecs = 0;

                        List<array<DataPoint^>^>^ dataPoints;

                        void haar(int numSecs);
                        void inverseHaar(List<double>^ frequencyChannel)
                            ;
                        void inverseHaar(array<List<double>^>^
                            frequencies, array<List<String^>^>^ waves,
                            array<List<String^>^>^ sampleTimes);
                        void updateMap(Hashtable^ results, double
                            amplitude, String^ wave, String^ sampleTime,
                            int nChannel);
                        String^ prepareFile();
                        String^ setWaveType(double amplitude);
                        void setDataBlock(array<DataPoint^>^ dp, array<
                            List<double>^>^yPoints, array<List<double>^>^
                            amplitudes, array<List<String^>^>^ waves,
                            array<List<String^>^>^ sampleTimes);
        };
}
```

Código 7.13: Definición de funciones para extracción

```
#include "Extraction.h"
#include <cmath>
#include <atlstr.h>
#include <string>
#include <list>
#include <iostream>


namespace EEGExtraction
{
        using namespace System;

        Extraction::Extraction() {}
        void Extraction::start(String^ file, String^ extractionMethod)
        {
                try
                {
                        inputFile = File::OpenText("Records\\" + file);
                        String^ header = inputFile->ReadLine();
                        SEPARATOR = gcnew array<String^>(1);
                        SEPARATOR[0] = ",";
                        stopped = false;
```

```
                        inFile = file;
                        if (extractionMethod->CompareTo("Haar Wavelet")
                            == 0)
                        {
                                outFile = inFile->Substring(0, inFile->
                                    Length - 4) + "_wavelet.csv";
                        }
                        else if (extractionMethod->CompareTo("ICA") == 0)
                        {
                                outFile = inFile->Substring(0, inFile->
                                    Length - 4) + "_ica.csv";
                        }

                        FileStream^ output = File::Open("Extraction\\" +
                            outFile, FileMode::Create);
                        outputFile = gcnew StreamWriter(output);

                        outputFile->WriteLine("MILLISECONDS,F3,FC5,FC6,F4
                            ,F3WAVE,FC5WAVE,FC6WAVE,F4WAVE");

                }
                catch (Exception^ e)
                {
                        // Que hacer aqui?

                }

        }
        String^ Extraction::prepareFile()
        {
                String^ prepFile = inFile->Substring(0, inFile->Length -
                    4) + "_prep.csv";
                FileStream^ prepOutput = File::Open("Extraction\\" +
                    prepFile, FileMode::Create);
                StreamWriter^ prepOutputFile = gcnew StreamWriter(
                    prepOutput);

                prepOutputFile->WriteLine("COUNTER,F3,FC5,FC6,F4");
                String^ uselessLine = "";
                while (!inputFile->EndOfStream && !uselessLine->
                    StartsWith("0"))
                {
                        uselessLine = inputFile->ReadLine();
                }
                String^ line = uselessLine; //First line found with
                    COUNTER 0
                while (!inputFile->EndOfStream)
                {
                        if (line->Contains("leftArm") || line->Contains("
                            rightLeg"))
                        {
                                array<String^>^ columns
                                        = line->Split(SEPARATOR, System::
                                            StringSplitOptions::None);
                                String^ result =
                                        columns[0] + "," +      //COUNTER
                                        columns[3] + "," +      //F3
                                        columns[4] + "," +      //FC5
```

```
                                        columns[11] + "," +     //FC6
                                        columns[12];            //F4
                                prepOutputFile->WriteLine(result);
                        }
                        line = inputFile->ReadLine();
                }
                inputFile->Close();
                prepOutputFile->Close();
                return prepFile;

        }
        void Extraction::extract(String^ extractionMethod)
        {
                String^ prepFile = prepareFile();
                inputFile = File::OpenText("Extraction\\" + prepFile);
                String^ header = inputFile->ReadLine();
                int i = 0;

                f3Channel = gcnew List<double>();
                fc5Channel = gcnew List<double>();
                fc6Channel = gcnew List<double>();
                f4Channel = gcnew List<double>();

                while (!inputFile->EndOfStream)
                {
                        String^ line = inputFile->ReadLine();
                        array<String^>^ columns
                                = line->Split(SEPARATOR, System::
                                   StringSplitOptions::None);
                        double f3 = atof(CString(columns[1]));
                        double fc5 = atof(CString(columns[2]));
                        double fc6 = atof(CString(columns[3]));
                        double f4 = atof(CString(columns[4]));
                        f3Channel->Add(f3);
                        fc5Channel->Add(fc5);
                        fc6Channel->Add(fc6);
                        f4Channel->Add(f4);
                        i++;
                        if (i > 127)
                        {
                                if (extractionMethod->CompareTo("Haar
                                   Wavelet") == 0)
                                {
                                        haar(numSecs);
                                }
                                i = 0;
                                numSecs++;
                        }
                }
                // Process remaining items that are less than 127 at the
                   end
                if (i > 0)
                {
                        if (extractionMethod->CompareTo("Haar Wavelet")
                           == 0)
                        {
                                haar(numSecs);
                        }
                }
```

```
inputFile->Close();
outputFile->Close();
//7.8125
//Apply inverse Haar
//Input file is now *_wavelet.csv or *_ica.csv
inputFile = File::OpenText("Extraction\\" + outFile);
header = inputFile->ReadLine();
//Output file is now *_<extraction method>_extraction.csv
String^ lastFile;
if (extractionMethod->CompareTo("Haar Wavelet") == 0)
{
        lastFile = inFile->Substring(0, inFile->Length -
           4) + "_wav_extraction.csv";
}
else
{
        lastFile = inFile->Substring(0, inFile->Length -
           4) + "_ica_extraction.csv";
}
FileStream^ output = File::Open("Extraction\\" + lastFile
    , FileMode::Create);
outputFile = gcnew StreamWriter(output);
outFile = lastFile;
outputFile->WriteLine("F3,F3WAVE,F3TIME,FC5,FC5WAVE,
    FC5TIME,FC6,FC6WAVE,FC6TIME,F4,F4WAVE,F4TIME");
f3Wave = gcnew List<String^>();
fc5Wave = gcnew List<String^>();
fc6Wave = gcnew List<String^>();
f4Wave = gcnew List<String^>();
f3Time = gcnew List<String^>();
fc5Time = gcnew List<String^>();
fc6Time = gcnew List<String^>();
f4Time = gcnew List<String^>();
array<List<double>^>^ frequencies = gcnew array<List<
    double>^>{f3Channel, fc5Channel, fc6Channel, f4Channel
    };
array<List<String^>^>^ waves = gcnew array<List<String
    ^>^> {f3Wave, fc5Wave, fc6Wave, f4Wave};
array<List<String^>^>^ sampleTimes = gcnew array<List<
    String^>^>{f3Time, fc5Time, fc6Time, f4Time};

i = 0;
while (!inputFile->EndOfStream)
{
        String^ line = inputFile->ReadLine();
        array<String^>^ columns
                = line->Split(SEPARATOR, System::
                    StringSplitOptions::None);
        if (columns[1]->CompareTo("na") != 0)
        {
double f3 = atof(CString(columns[1]));
                f3Channel->Add(f3);
                f3Wave->Add(columns[5]);
                f3Time->Add(columns[0]);
        }
        if (columns[2]->CompareTo("na") != 0)
        {
                double fc5 = atof(CString(columns[2]));
                fc5Channel->Add(fc5);
```

```
                        fc5Wave->Add(columns[6]);
                        fc5Time->Add(columns[0]);
                }
                if (columns[3]->CompareTo("na") != 0)
                {
                        double fc6 = atof(CString(columns[3]));
                        fc6Channel->Add(fc6);
                        fc6Wave->Add(columns[7]);
                        fc6Time->Add(columns[0]);
                }
                if (columns[4]->CompareTo("na") != 0)
                {
                        double f4 = atof(CString(columns[4]));
                        f4Channel->Add(f4);
                        f4Wave->Add(columns[8]);
                        f4Time->Add(columns[0]);
                }

                i++;
                if (i > 127)
                {
                        if (extractionMethod->CompareTo("Haar
                            Wavelet") == 0)
                        {
                                inverseHaar(frequencies, waves,
                                    sampleTimes);
                        }
                        i = 0;
                }
        }
        if (i > 0)
        {
                inverseHaar(frequencies, waves, sampleTimes);
        }

        stop();

}

void Extraction::haar(int numSecs)
{
        int i, k;
        double s;
        double *y1, *y2, *y3, *y4;
        int n = fc5Channel->Count;

        s = sqrt(2.0);
        y1 = new double[n];
        y2 = new double[n];
        y3 = new double[n];
        y4 = new double[n];

        //  Initialize
        for (i = 0; i < n; i++)
        {
                y1[i] = 0.0L;
                y2[i] = 0.0L;
                y3[i] = 0.0L;
                y4[i] = 0.0L;
```

```
}

//Determina la potencia mayor de K en 2, donde K<=N
k = 1;
while (k * 2 <= n)
{
        k = k * 2;
}

while (1 < k)
{
        k = k / 2;
        for (i = 0; i < k; i++)
        {
                //F3
                y1[i] = (f3Channel[2 * i] + f3Channel[2 *
                    i + 1]) / s;
                y1[i + k] = (f3Channel[2 * i] - f3Channel
                   [2 * i + 1]) / s;

                //FC5
                y2[i] = (fc5Channel[2 * i] + fc5Channel[2
                    * i + 1]) / s;
                y2[i + k] = (fc5Channel[2 * i] -
                   fc5Channel[2 * i + 1]) / s;

                //FC6
                y3[i] = (fc6Channel[2 * i] + fc6Channel[2
                    * i + 1]) / s;
                y3[i + k] = (fc6Channel[2 * i] -
                   fc6Channel[2 * i + 1]) / s;

                //F4
                y4[i] = (f4Channel[2 * i] + f4Channel[2 *
                    i + 1]) / s;
                y4[i + k] = (f4Channel[2 * i] - f4Channel
                   [2 * i + 1]) / s;
        }

        for (i = 0; i < k * 2; i++){
                f3Channel[i] = y1[i];
                fc5Channel[i] = y2[i];
                fc6Channel[i] = y3[i];
                f4Channel[i] = y4[i];
        }
}

for (i = 0; i < n; i++)
{
        // 8 - 13.99 - MU
        // 14 - 30 - BETA

        array<String^>^ waveType = gcnew array<String
           ^>(4);
        waveType[0] = setWaveType(f3Channel[i]);
        waveType[1] = setWaveType(fc5Channel[i]);
        waveType[2] = setWaveType(fc6Channel[i]);
        waveType[3] = setWaveType(f4Channel[i]);
```

```
                        boolean noData = (waveType[0]->CompareTo("na") ==
                            0 &&
                                waveType[1]->CompareTo("na") == 0 &&
                                waveType[2]->CompareTo("na") == 0 &&
                                waveType[3]->CompareTo("na") == 0);

                        if(!noData)
                        {
                                String^ result = gcnew String((to_string
                                    ((numSecs * 1000) + 7.8125 * i) +
                                        "," +
                                        (waveType[0]->CompareTo("na") !=
                                            0 ? to_string(f3Channel[i]) :
                                            "na") +
                                        "," +
                                        (waveType[1]->CompareTo("na") !=
                                            0 ? to_string(fc5Channel[i]) :
                                             "na") +
                                        "," +
                                        (waveType[2]->CompareTo("na") !=
                                            0 ? to_string(fc6Channel[i]) :
                                             "na") +
                                        "," +
                                        (waveType[3]->CompareTo("na") !=
                                            0 ? to_string(f4Channel[i]) :
                                            "na")).c_str()) +
                                        "," +
                                        waveType[0] +
                                        "," +
                                        waveType[1] +
                                        "," +
                                        waveType[2] +
                                        "," +
                                        waveType[3] +
                                        "\n";
                                outputFile->Write(result);
                        }
                }
                delete[] y1;
                delete[] y2;
                delete[] y3;
                delete[] y4;
                f3Channel->Clear();
                fc5Channel->Clear();
                fc6Channel->Clear();
                f4Channel->Clear();


        }

        void Extraction::inverseHaar(List<double>^ frequencyChannel)
        {
                int i, k;
                double s;
                double *y;
                int n = frequencyChannel->Count;

                s = sqrt(2.0);
                y = new double[n];
```

```
            //  Initialize
            for (i = 0; i < n; i++)
            {
                    y[i] = 0.0L;
            }

            k = 1;
            while (k * 2 <= n)
            {
                    for (i = 0; i < k; i++)
                    {
                            y[2 * i] = (frequencyChannel[i] +
                               frequencyChannel[i + k]) / s;
                            y[2 * i + 1] = (frequencyChannel[i] -
                               frequencyChannel[i + k]) / s;


                    }
                    for (i = 0; i < k * 2; i++)
                    {
                            frequencyChannel[i] = y[i];
                    }
                    k = k * 2;
            }

            delete[] y;
    }

    void Extraction::inverseHaar(array<List<double>^>^ frequencies,
       array<List<String^>^>^ waves, array<List<String^>^>^
       sampleTimes)
    {
            Hashtable^ extractionResults = gcnew Hashtable();
            for (int k = 0; k < frequencies->Length; k++)
            {
                    inverseHaar(frequencies[k]);
                    if (k == 0)
                    {
                            array<String^>^ er = gcnew array<String
                               ^>(frequencies[k]->Count);
                            for (int j = 0; j < frequencies[k]->Count
                               ; j++)
                            {
                                    String^ result = gcnew String(
                                       to_string(frequencies[k][j]).
                                       c_str()) +
                                            "," +
                                            waves[k][j] +
                                            "," +
                                            sampleTimes[k][j];
                                    er[j] = result;

                                    extractionResults[sampleTimes[k][
                                       j]] = er[j];
                            }

                    }
                    else
```

```
                        {
                                for (int j = 0; j < frequencies[k]->Count
                                    ; j++)
                                {
                                        updateMap(extractionResults,
                                            frequencies[k][j], waves[k][j
                                            ], sampleTimes[k][j], k);
                                }
                        }
                }

                map<double, string> orderMap;
                for each(DictionaryEntry de in extractionResults)
                {
                        String^ line = (String^) de.Value;
                        array<String^>^ columns
                                = line->Split(SEPARATOR, System::
                                    StringSplitOptions::None);
                        if (columns->Length == 3) line += ",,," + columns
                            [2] + "," + ",," + columns[2] + "," + ",," +
                            columns[2];
                        else if (columns->Length == 6) line += ",,," +
                            columns[5] + "," + ",," + columns[5];
                        else if (columns->Length == 9) line += ",,," +
                            columns[8];
                        double key = atof(CString(columns[2]));
                        orderMap[key] = CString(line);
                }
                for (map <double, string>::iterator it = orderMap.begin()
                    ; it != orderMap.end(); ++it)
                {
                        String^ line = gcnew String(it->second.c_str());
                        outputFile->WriteLine(line);
                }

                for (int k = 0; k < frequencies->Length; k++)
                {
                        frequencies[k]->Clear();
                        waves[k]->Clear();
                        sampleTimes[k]->Clear();
                }

                extractionResults->Clear();

        }

        void Extraction::graph(String^ fileName)
        {
                f3Channel = gcnew List<double>();
                fc5Channel = gcnew List<double>();
                fc6Channel = gcnew List<double>();
                f4Channel = gcnew List<double>();
                f3Wave = gcnew List<String^>();
                fc5Wave = gcnew List<String^>();
                fc6Wave = gcnew List<String^>();
                f4Wave = gcnew List<String^>();
                f3Time = gcnew List<String^>();
                fc5Time = gcnew List<String^>();
                fc6Time = gcnew List<String^>();
```

```
f4Time = gcnew List<String^>();
array<List<double>^>^ amplitudes = gcnew array<List<
    double>^>{f3Channel, fc5Channel, fc6Channel, f4Channel
    };
array<List<String^>^>^ waves = gcnew array<List<String
    ^>^> {f3Wave, fc5Wave, fc6Wave, f4Wave};
array<List<String^>^>^ sampleTimes = gcnew array<List<
    String^>^>{f3Time, fc5Time, fc6Time, f4Time};
List<String^>^ lines = gcnew List<String^>();

if (inputFile == nullptr)
{
        inputFile = File::OpenText("Extraction\\" +
            fileName);
        String^ header = inputFile->ReadLine();
        SEPARATOR = gcnew array<String^>(1);
        SEPARATOR[0] = ",";
        stopped = false;
        done = false;
}

delete[] dataPoints;
dataPoints = gcnew List<array<DataPoint^>^>();

int k = 0;

while (!inputFile->EndOfStream && k++ < sampleRate)
{
        String^ line = inputFile->ReadLine();
        lines->Add(line);
}

int i = 0;
for each(String^ line in lines)
{
        // Decompose record in 4 Channels Info
        array<String^>^ columns
                = line->Split(SEPARATOR, System::
                    StringSplitOptions::None);
        for (int n = 0; n < amplitudes->Length; n++)
        {
                String^ amplitudeStr = columns[3 * n];
                String^ wave = columns[3 * n + 1];
                String^ sampleTimeStr = columns[3 * n +
                    2];

                if (amplitudeStr != nullptr &&
                    amplitudeStr->CompareTo("") != 0)
                {
                        amplitudes[n]->Add(atof(CString(
                            amplitudeStr)));
                        waves[n]->Add(wave);
                        sampleTimes[n]->Add(sampleTimeStr
                            );
                }
        }
        i++;
        if (i >= datasetSize)
        {
```

```
                                array<DataPoint^>^ dp = gcnew array<
                                    DataPoint^>(8);
                                array<List<double>^>^ yPoints = gcnew
                                    array<List<double>^>(8);
                                // Create a new group of data points
                                dataPoints->Add(dp);
                                setDataBlock(dp, yPoints, amplitudes,
                                    waves, sampleTimes);
                                i = 0;
                                delete[] dp;
                                delete[] yPoints;
                                for (int n = 0; n < amplitudes->Length; n
                                    ++)
                                {
                                        amplitudes[n]->Clear();
                                        waves[n]->Clear();
                                        sampleTimes[n]->Clear();
                                }

                        }
                }
                // Add remaining points if not module of 32
                if (i > 0)
                {
                        array<DataPoint^>^ dp = gcnew array<DataPoint
                            ^>(8);
                        array<List<double>^>^ yPoints = gcnew array<List<
                            double>^>(8);
                        // Create a new group of data points
                        dataPoints->Add(dp);
                        setDataBlock(dp, yPoints, amplitudes, waves,
                            sampleTimes);

                        i = 0;
                        delete[] dp;
                        delete[] yPoints;
                }
                if (inputFile->EndOfStream)
                {
                        done = true;
                        stopped = true;
                }

        }

        void Extraction::setDataBlock(array<DataPoint^>^ dp, array<List<
            double>^>^yPoints, array<List<double>^>^ amplitudes, array<
            List<String^>^>^ waves, array<List<String^>^>^ sampleTimes)
        {
                int n = 0;
                for (int k = 0; k < amplitudes->Length; k++)
                {
                        dp[n] = gcnew  DataPoint();
                        dp[n + 1] = gcnew  DataPoint();
                        yPoints[n] = gcnew List<double>();
                        yPoints[n + 1] = gcnew List<double>();

                        for (int m = 0; m < amplitudes[k]->Count; m++)
                        {
```

```
                                if (waves[k][m]->CompareTo("MU") == 0)
                                {
                                        dp[n]->XValue = atof(CString(
                                            sampleTimes[k][m]))/1000;  //
                                            Convert to seconds
                                        //dp[n]->XValue = atof(CString(
                                            sampleTimes[k][m]));

                                        yPoints[n]->Add(amplitudes[k][m])
                                            ;
                                }
                                else
                                {
                                        dp[n + 1]->XValue = atof(CString(
                                            sampleTimes[k][m]))/1000; //
                                            Convert to seconds
                                        //dp[n + 1]->XValue = atof(
                                            CString(sampleTimes[k][m]));
                                        yPoints[n + 1]->Add(amplitudes[k
                                            ][m]);
                                }

                        }

                        dp[n]->YValues = yPoints[n]->ToArray();
                        dp[n + 1]->YValues = yPoints[n + 1]->ToArray();
                        n = n + 2;
                }
        }

        String^ Extraction::setWaveType(double amplitude)
        {
                String^ waveType = "na";
                if (amplitude >= +8.0 && amplitude < +14.0)
                {
                        waveType = "MU";
                }
                else if (amplitude >= +14.0 && amplitude <= +30.0)
                {
                        waveType = "BETA";
                }
                return waveType;
        }

        void Extraction::updateMap(Hashtable^ results, double amplitude,
            String^ wave, String^ sampleTime, int nChannel)
        {
                String^ mapValue;
                String^ result = gcnew String(to_string(amplitude).c_str
                    ()) +
                        "," +
                        wave +
                        "," +
                        sampleTime;
                if (results->ContainsKey(sampleTime))
                {
                        mapValue = (String^)results[sampleTime];
                        array<String^>^ columns
                                = mapValue->Split(SEPARATOR, System::
```

```
                                StringSplitOptions::None);
                int numCols = columns->Length;
                if (nChannel == 1) mapValue += "," + result;
                else if (nChannel == 2)
                {
                        if (numCols < 6)
                        {
                                mapValue += ",,," + sampleTime;
                                   //Fil up missing channel
                        }
                        mapValue += "," + result;
                }
                else if (nChannel == 3)
                {
                        if (numCols < 9)
                        {
                                do
                                {
                                        mapValue += ",,," +
                                           sampleTime; //Fil up
                                           missing channels
                                        numCols += 3;
                                } while (numCols < 9);

                        }
                        mapValue += "," + result;
                }

                results[sampleTime] = mapValue;
        }
        else
        {   //Fill up spaces with time to order those when
            validating the file
                if (nChannel == 1) result = ",," + sampleTime + "
                    ," + result;
                else if (nChannel == 2) result = ",," +
                   sampleTime + "," + ",," + sampleTime + ","  +
                   result;
                else if (nChannel == 3) result = ",," +
                   sampleTime + "," + ",," + sampleTime + "," + "
                   ,," + sampleTime + "," + result;
                results[sampleTime] = result;
        }
}

void Extraction::stop()
{
        stopped = true;
        if (inputFile != nullptr)
        {
                inputFile->Close();

        }
        if (outputFile != nullptr)
        {
                outputFile->Close();
        }
        done = true;
```

```
                delete[] f3Channel;
                delete[] fc5Channel;
                delete[] fc6Channel;
                delete[] f4Channel;
                delete[] f3Wave;
                delete[] fc5Wave;
                delete[] fc6Wave;
                delete[] f4Wave;
                delete[] f3Time;
                delete[] fc5Time;
                delete[] fc6Time;
                delete[] f4Time;
        }

        Extraction::~Extraction()
        {}
}
```

Código 7.14: Pantala para extracción de características

```
#pragma once
#include "Extraction.h"
namespace EEGRECORD {

        using namespace System;
        using namespace System::ComponentModel;
        using namespace System::Collections;
        using namespace System::Windows::Forms;
        using namespace System::IO;
        using namespace System::Data;
        using namespace System::Drawing;
        using namespace System::Windows::Forms::DataVisualization::
           Charting;
        using namespace EEGExtraction;

        /// <summary>
        /// Summary for ExtractionBrowser
        /// </summary>
        public ref class ExtractionBrowser : public System::Windows::
           Forms::Form
        {
        private: bool acqFileOpened;
                        bool extractionFileOpened;
                        Extraction^ extraction = gcnew Extraction();
                        bool starting = true;
        private: System::Windows::Forms::ColumnHeader^
           acqFileNameColumnHeader;
        private: System::Windows::Forms::ColumnHeader^
           extractionFileNameColumnHeader;
        private: System::ComponentModel::BackgroundWorker^
           extractionBackgroundWorker;
        private: System::Windows::Forms::Timer^  extractionTimer;

                        array<Series^>^series = gcnew array<Series^>(8);
        private: System::ComponentModel::BackgroundWorker^
           extractionGraphBackgroundWorker;
        private: System::Windows::Forms::Timer^  extractionGraphTimer;
                        String^ acqOpenFileName;
                        String^ extractionOpenFileName;
```

```
        private: System::Windows::Forms::Panel^  extractionMethodPanel;


        private: System::Windows::Forms::Label^  extractionMethodLabel;
                 Color^ eventColor;
                 String^ extractionMethod = "Haar Wavelet";
        private: System::Windows::Forms::ComboBox^  eegChComboBox;
        private: System::Windows::Forms::Label^  eegChLabel;
        private: System::Windows::Forms::ComboBox^
            extractionMethodComboBox;
                 String^ extractFolderName;
        public:
                ExtractionBrowser(String^ folderName)
                {
                        InitializeComponent();
                        //
                        //TODO: Add the constructor code here
                        //
                        acqFileOpened = false;
                        extractionFileOpened = false;
                        String^ acqFolderName = folderName->Concat(
                            folderName, Path::DirectorySeparatorChar);
                        acqFolderName = acqFolderName->Concat(
                            acqFolderName, "Records");
                        acqFolderName = acqFolderName->Concat(
                            acqFolderName, Path::DirectorySeparatorChar);
                        for each (String^ fileName in Directory::
                            EnumerateFiles(acqFolderName, "*.csv", IO::
                            SearchOption::TopDirectoryOnly))
                        {
                                String^ shortFileName = fileName->
                                    Substring(fileName->LastIndexOf(Path::
                                    DirectorySeparatorChar) + 1);
                                acqFileListView->Items->Add(shortFileName
                                    )->ImageIndex = 0;
                        }
                        extractFolderName = folderName->Concat(folderName
                            , Path::DirectorySeparatorChar);
                        extractFolderName = extractFolderName->Concat(
                            extractFolderName,"Extraction");
                        extractFolderName = extractFolderName->Concat(
                            extractFolderName,Path::DirectorySeparatorChar
                            );
                        for each (String^ fileName in Directory::
                            EnumerateFiles(extractFolderName, "*
                            _extraction.csv", IO::SearchOption::
                            TopDirectoryOnly))
                        {
                                String^ shortFileName = fileName->
                                    Substring(fileName->LastIndexOf(Path::
                                    DirectorySeparatorChar) + 1);
                                extractionListView->Items->Add(
                                    shortFileName)->ImageIndex = 0;
                        }

                        int i = 0;
                        for (Generic::IEnumerator<Series^>^ it = this->
                            extractionChart->Series->GetEnumerator(); it->
                            MoveNext();)
```

```
                {
                        series[i] = it->Current;
                        series[i++]->IsVisibleInLegend = false;
                }
                extractionMethodComboBox->SelectedIndex = 0;
                eegChComboBox->SelectedIndex = 0;
        }

protected:
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        ~ExtractionBrowser()
        {
                if (components)
                {
                        delete components;
                }
        }

protected:

private: System::Windows::Forms::ImageList^  csvImageList;
private: System::Windows::Forms::ToolStripContainer^
    extractionToolStripContainer;
private: System::Windows::Forms::StatusStrip^
    extractionStatusStrip;
private: System::Windows::Forms::SplitContainer^
    extractionSplitContainer;
private: System::Windows::Forms::TableLayoutPanel^
    extractionFileTableLayoutPanel;
private: System::Windows::Forms::TextBox^  acqFileSearchTextBox;
private: System::Windows::Forms::TextBox^
    extractionFileSearchTextBox;



private: System::Windows::Forms::ListView^  acqFileListView;
private: System::Windows::Forms::ListView^  extractionListView;
private: System::Windows::Forms::ToolStripStatusLabel^
    extractionToolStripStatusLabel;

private: System::Windows::Forms::ToolStripProgressBar^
    extractionToolStripProgressBar;
private: System::Windows::Forms::DataVisualization::Charting::
    Chart^  extractionChart;


private: System::ComponentModel::IContainer^  components;


protected:


private:
        /// <summary>
        /// Required designer variable.
        /// </summary>
```

```
#pragma region Windows Form Designer generated code
                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                void InitializeComponent(void)
                {
                        this->components = (gcnew System::ComponentModel
                           ::Container());
                        System::ComponentModel::ComponentResourceManager^
                             resources = (gcnew System::ComponentModel::
                           ComponentResourceManager(ExtractionBrowser::
                           typeid));
                        System::Windows::Forms::DataVisualization::
                           Charting::ChartArea^  chartArea1 = (gcnew
                           System::Windows::Forms::DataVisualization::
                           Charting::ChartArea());
                        System::Windows::Forms::DataVisualization::
                           Charting::ChartArea^  chartArea2 = (gcnew
                           System::Windows::Forms::DataVisualization::
                           Charting::ChartArea());
                        System::Windows::Forms::DataVisualization::
                           Charting::Legend^  legend1 = (gcnew System::
                           Windows::Forms::DataVisualization::Charting::
                           Legend());
                        System::Windows::Forms::DataVisualization::
                           Charting::Series^  series1 = (gcnew System::
                           Windows::Forms::DataVisualization::Charting::
                           Series());
                        System::Windows::Forms::DataVisualization::
                           Charting::Series^  series2 = (gcnew System::
                           Windows::Forms::DataVisualization::Charting::
                           Series());
                        System::Windows::Forms::DataVisualization::
                           Charting::Series^  series3 = (gcnew System::
                           Windows::Forms::DataVisualization::Charting::
                           Series());
                        System::Windows::Forms::DataVisualization::
                           Charting::Series^  series4 = (gcnew System::
                           Windows::Forms::DataVisualization::Charting::
                           Series());
                        System::Windows::Forms::DataVisualization::
                           Charting::Series^  series5 = (gcnew System::
                           Windows::Forms::DataVisualization::Charting::
                           Series());
                        System::Windows::Forms::DataVisualization::
                           Charting::Series^  series6 = (gcnew System::
                           Windows::Forms::DataVisualization::Charting::
                           Series());
                        System::Windows::Forms::DataVisualization::
                           Charting::Series^  series7 = (gcnew System::
                           Windows::Forms::DataVisualization::Charting::
                           Series());
                        System::Windows::Forms::DataVisualization::
                           Charting::Series^  series8 = (gcnew System::
                           Windows::Forms::DataVisualization::Charting::
                           Series());
                        this->csvImageList = (gcnew System::Windows::
```

```
    Forms::ImageList(this->components));
this->extractionToolStripContainer = (gcnew
    System::Windows::Forms::ToolStripContainer());
this->extractionStatusStrip = (gcnew System::
    Windows::Forms::StatusStrip());
this->extractionToolStripStatusLabel = (gcnew
    System::Windows::Forms::ToolStripStatusLabel()
    );
this->extractionToolStripProgressBar = (gcnew
    System::Windows::Forms::ToolStripProgressBar()
    );
this->extractionSplitContainer = (gcnew System::
    Windows::Forms::SplitContainer());
this->extractionFileTableLayoutPanel = (gcnew
    System::Windows::Forms::TableLayoutPanel());
this->acqFileSearchTextBox = (gcnew System::
    Windows::Forms::TextBox());
this->extractionFileSearchTextBox = (gcnew System
    ::Windows::Forms::TextBox());
this->acqFileListView = (gcnew System::Windows::
    Forms::ListView());
this->acqFileNameColumnHeader = (gcnew System::
    Windows::Forms::ColumnHeader());
this->extractionListView = (gcnew System::Windows
    ::Forms::ListView());
this->extractionFileNameColumnHeader = (gcnew
    System::Windows::Forms::ColumnHeader());
this->extractionMethodPanel = (gcnew System::
    Windows::Forms::Panel());
this->extractionMethodComboBox = (gcnew System::
    Windows::Forms::ComboBox());
this->eegChComboBox = (gcnew System::Windows::
    Forms::ComboBox());
this->eegChLabel = (gcnew System::Windows::Forms
    ::Label());
this->extractionMethodLabel = (gcnew System::
    Windows::Forms::Label());
this->extractionChart = (gcnew System::Windows::
    Forms::DataVisualization::Charting::Chart());
this->extractionBackgroundWorker = (gcnew System
    ::ComponentModel::BackgroundWorker());
this->extractionTimer = (gcnew System::Windows::
    Forms::Timer(this->components));
this->extractionGraphBackgroundWorker = (gcnew
    System::ComponentModel::BackgroundWorker());
this->extractionGraphTimer = (gcnew System::
    Windows::Forms::Timer(this->components));
this->extractionToolStripContainer->
    BottomToolStripPanel->SuspendLayout();
this->extractionToolStripContainer->ContentPanel
    ->SuspendLayout();
this->extractionToolStripContainer->SuspendLayout
    ();
this->extractionStatusStrip->SuspendLayout();
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->
    extractionSplitContainer))->BeginInit();
this->extractionSplitContainer->Panel1->
    SuspendLayout();
```

```
            this->extractionSplitContainer->Panel2->
                SuspendLayout();
            this->extractionSplitContainer->SuspendLayout();
            this->extractionFileTableLayoutPanel->
                SuspendLayout();
            this->extractionMethodPanel->SuspendLayout();
            (cli::safe_cast<System::ComponentModel::
                ISupportInitialize^>(this->extractionChart))->
                BeginInit();
            this->SuspendLayout();
            //
            // csvImageList
            //
            this->csvImageList->ImageStream = (cli::safe_cast
                <System::Windows::Forms::ImageListStreamer^>(
                resources->GetObject(L"csvImageList.
                ImageStream")));
            this->csvImageList->TransparentColor = System::
                Drawing::Color::Transparent;
            this->csvImageList->Images->SetKeyName(0, L"csv.
                ico");
            //
            // extractionToolStripContainer
            //
            //
            // extractionToolStripContainer.
                BottomToolStripPanel
            //
            this->extractionToolStripContainer->
                BottomToolStripPanel->Controls->Add(this->
                extractionStatusStrip);
            //
            // extractionToolStripContainer.ContentPanel
            //
            this->extractionToolStripContainer->ContentPanel
                ->Controls->Add(this->extractionSplitContainer
                );
            this->extractionToolStripContainer->ContentPanel
                ->Size = System::Drawing::Size(784, 540);
            this->extractionToolStripContainer->Dock = System
                ::Windows::Forms::DockStyle::Fill;
            this->extractionToolStripContainer->Location =
                System::Drawing::Point(0, 0);
            this->extractionToolStripContainer->Name = L"
                extractionToolStripContainer";
            this->extractionToolStripContainer->Size = System
                ::Drawing::Size(784, 562);
            this->extractionToolStripContainer->TabIndex = 2;
            this->extractionToolStripContainer->Text = L"
                toolStripContainer1";
            this->extractionToolStripContainer->
                TopToolStripPanelVisible = false;
            //
            // extractionStatusStrip
            //
            this->extractionStatusStrip->Dock = System::
                Windows::Forms::DockStyle::None;
            this->extractionStatusStrip->Items->AddRange(
                gcnew cli::array< System::Windows::Forms::
```

```
    ToolStripItem^  >(2) {
        this->extractionToolStripStatusLabel,
                this->
                    extractionToolStripProgressBar
});
this->extractionStatusStrip->Location = System::
    Drawing::Point(0, 0);
this->extractionStatusStrip->Name = L"
    extractionStatusStrip";
this->extractionStatusStrip->RenderMode = System
    ::Windows::Forms::ToolStripRenderMode::
    Professional;
this->extractionStatusStrip->Size = System::
    Drawing::Size(784, 22);
this->extractionStatusStrip->TabIndex = 2;
this->extractionStatusStrip->Text = L"
    statusStrip1";
//
// extractionToolStripStatusLabel
//
this->extractionToolStripStatusLabel->Name = L"
    extractionToolStripStatusLabel";
this->extractionToolStripStatusLabel->Size =
    System::Drawing::Size(59, 17);
this->extractionToolStripStatusLabel->Text = L"
    Loading...";
//
// extractionToolStripProgressBar
//
this->extractionToolStripProgressBar->Alignment =
     System::Windows::Forms::
    ToolStripItemAlignment::Right;
this->extractionToolStripProgressBar->Maximum =
    1000;
this->extractionToolStripProgressBar->Name = L"
    extractionToolStripProgressBar";
this->extractionToolStripProgressBar->Size =
    System::Drawing::Size(1300, 16);
this->extractionToolStripProgressBar->Step = 1;
//
// extractionSplitContainer
//
this->extractionSplitContainer->Dock = System::
    Windows::Forms::DockStyle::Fill;
this->extractionSplitContainer->FixedPanel =
    System::Windows::Forms::FixedPanel::Panel1;
this->extractionSplitContainer->IsSplitterFixed =
     true;
this->extractionSplitContainer->Location = System
    ::Drawing::Point(0, 0);
this->extractionSplitContainer->Name = L"
    extractionSplitContainer";
//
// extractionSplitContainer.Panel1
//
this->extractionSplitContainer->Panel1->Controls
    ->Add(this->extractionFileTableLayoutPanel);
this->extractionSplitContainer->Panel1MinSize =
    10;
```

```
            //
            // extractionSplitContainer.Panel2
            //
            this->extractionSplitContainer->Panel2->Controls
                ->Add(this->extractionChart);
            this->extractionSplitContainer->Size = System::
                Drawing::Size(784, 540);
            this->extractionSplitContainer->SplitterDistance
                = 252;
            this->extractionSplitContainer->TabIndex = 1;
            //
            // extractionFileTableLayoutPanel
            //
            this->extractionFileTableLayoutPanel->ColumnCount
                 = 1;
            this->extractionFileTableLayoutPanel->
                ColumnStyles->Add((gcnew System::Windows::
                Forms::ColumnStyle(System::Windows::Forms::
                SizeType::Absolute,
                    252)));
            this->extractionFileTableLayoutPanel->Controls->
                Add(this->acqFileSearchTextBox, 0, 1);
            this->extractionFileTableLayoutPanel->Controls->
                Add(this->extractionFileSearchTextBox, 0, 3);
            this->extractionFileTableLayoutPanel->Controls->
                Add(this->acqFileListView, 0, 2);
            this->extractionFileTableLayoutPanel->Controls->
                Add(this->extractionListView, 0, 4);
            this->extractionFileTableLayoutPanel->Controls->
                Add(this->extractionMethodPanel, 0, 0);
            this->extractionFileTableLayoutPanel->Dock =
                System::Windows::Forms::DockStyle::Fill;
            this->extractionFileTableLayoutPanel->Location =
                System::Drawing::Point(0, 0);
            this->extractionFileTableLayoutPanel->Name = L"
                extractionFileTableLayoutPanel";
            this->extractionFileTableLayoutPanel->RowCount =
                5;
            this->extractionFileTableLayoutPanel->RowStyles->
                Add((gcnew System::Windows::Forms::RowStyle(
                System::Windows::Forms::SizeType::Absolute,
                    75)));
            this->extractionFileTableLayoutPanel->RowStyles->
                Add((gcnew System::Windows::Forms::RowStyle())
                );
            this->extractionFileTableLayoutPanel->RowStyles->
                Add((gcnew System::Windows::Forms::RowStyle(
                System::Windows::Forms::SizeType::Percent,
                    50)));
            this->extractionFileTableLayoutPanel->RowStyles->
                Add((gcnew System::Windows::Forms::RowStyle())
                );
            this->extractionFileTableLayoutPanel->RowStyles->
                Add((gcnew System::Windows::Forms::RowStyle(
                System::Windows::Forms::SizeType::Percent,
                    50)));
            this->extractionFileTableLayoutPanel->Size =
                System::Drawing::Size(252, 540);
            this->extractionFileTableLayoutPanel->TabIndex =
```

```
    0;
//
// acqFileSearchTextBox
//
this->acqFileSearchTextBox->Dock = System::
    Windows::Forms::DockStyle::Fill;
this->acqFileSearchTextBox->Location = System::
    Drawing::Point(3, 78);
this->acqFileSearchTextBox->Name = L"
    acqFileSearchTextBox";
this->acqFileSearchTextBox->Size = System::
    Drawing::Size(246, 20);
this->acqFileSearchTextBox->TabIndex = 0;
this->acqFileSearchTextBox->KeyDown += gcnew
    System::Windows::Forms::KeyEventHandler(this,
    &ExtractionBrowser::
    acqFileSearchTextBox_KeyDown);
//
// extractionFileSearchTextBox
//
this->extractionFileSearchTextBox->Dock = System
    ::Windows::Forms::DockStyle::Fill;
this->extractionFileSearchTextBox->Location =
    System::Drawing::Point(3, 310);
this->extractionFileSearchTextBox->Name = L"
    extractionFileSearchTextBox";
this->extractionFileSearchTextBox->Size = System
    ::Drawing::Size(246, 20);
this->extractionFileSearchTextBox->TabIndex = 1;
this->extractionFileSearchTextBox->KeyDown +=
    gcnew System::Windows::Forms::KeyEventHandler(
    this, &ExtractionBrowser::
    extractionFileSearchTextBox_KeyDown);
//
// acqFileListView
//
this->acqFileListView->Columns->AddRange(gcnew
    cli::array< System::Windows::Forms::
    ColumnHeader^ >(1) { this->
    acqFileNameColumnHeader });
this->acqFileListView->Dock = System::Windows::
    Forms::DockStyle::Fill;
this->acqFileListView->FullRowSelect = true;
this->acqFileListView->LargeImageList = this->
    csvImageList;
this->acqFileListView->Location = System::Drawing
    ::Point(3, 104);
this->acqFileListView->MultiSelect = false;
this->acqFileListView->Name = L"acqFileListView";
this->acqFileListView->Size = System::Drawing::
    Size(246, 200);
this->acqFileListView->SmallImageList = this->
    csvImageList;
this->acqFileListView->TabIndex = 3;
this->acqFileListView->
    UseCompatibleStateImageBehavior = false;
this->acqFileListView->View = System::Windows::
    Forms::View::Details;
this->acqFileListView->SelectedIndexChanged +=
```

```
        gcnew System::EventHandler(this, &
        ExtractionBrowser::
        acqFileListView_SelectedIndexChanged);
    this->acqFileListView->MouseDoubleClick += gcnew
        System::Windows::Forms::MouseEventHandler(this
        , &ExtractionBrowser::
        acqFileListView_MouseDoubleClick);
    //
    // acqFileNameColumnHeader
    //
    this->acqFileNameColumnHeader->Text = L"Recording
         File Name";
    this->acqFileNameColumnHeader->Width = 200;
    //
    // extractionListView
    //
    this->extractionListView->Columns->AddRange(gcnew
         cli::array< System::Windows::Forms::
        ColumnHeader^ >(1) { this->
        extractionFileNameColumnHeader });
    this->extractionListView->Dock = System::Windows
        ::Forms::DockStyle::Fill;
    this->extractionListView->FullRowSelect = true;
    this->extractionListView->HideSelection = false;
    this->extractionListView->LargeImageList = this->
        csvImageList;
    this->extractionListView->Location = System::
        Drawing::Point(3, 336);
    this->extractionListView->MultiSelect = false;
    this->extractionListView->Name = L"
        extractionListView";
    this->extractionListView->Size = System::Drawing
        ::Size(246, 201);
    this->extractionListView->SmallImageList = this->
        csvImageList;
    this->extractionListView->TabIndex = 4;
    this->extractionListView->
        UseCompatibleStateImageBehavior = false;
    this->extractionListView->View = System::Windows
        ::Forms::View::Details;
    this->extractionListView->SelectedIndexChanged +=
         gcnew System::EventHandler(this, &
        ExtractionBrowser::
        extractionListView_SelectedIndexChanged);
    this->extractionListView->MouseDoubleClick +=
        gcnew System::Windows::Forms::
        MouseEventHandler(this, &ExtractionBrowser::
        extractionListView_MouseDoubleClick);
    //
    // extractionFileNameColumnHeader
    //
    this->extractionFileNameColumnHeader->Text = L"
        Extraction File Name";
    this->extractionFileNameColumnHeader->Width =
        200;
    //
    // extractionMethodPanel
    //
    this->extractionMethodPanel->AutoSize = true;
```

```
this->extractionMethodPanel->Controls->Add(this->
    extractionMethodComboBox);
this->extractionMethodPanel->Controls->Add(this->
    eegChComboBox);
this->extractionMethodPanel->Controls->Add(this->
    eegChLabel);
this->extractionMethodPanel->Controls->Add(this->
    extractionMethodLabel);
this->extractionMethodPanel->Dock = System::
    Windows::Forms::DockStyle::Fill;
this->extractionMethodPanel->Location = System::
    Drawing::Point(3, 3);
this->extractionMethodPanel->Name = L"
    extractionMethodPanel";
this->extractionMethodPanel->Size = System::
    Drawing::Size(246, 69);
this->extractionMethodPanel->TabIndex = 5;
//
// extractionMethodComboBox
//
this->extractionMethodComboBox->DropDownStyle =
    System::Windows::Forms::ComboBoxStyle::
    DropDownList;
this->extractionMethodComboBox->FlatStyle =
    System::Windows::Forms::FlatStyle::Popup;
this->extractionMethodComboBox->FormattingEnabled
     = true;
this->extractionMethodComboBox->Items->AddRange(
    gcnew cli::array< System::Object^  >(2) { L"
    Haar Wavelet", L"ICA" });
this->extractionMethodComboBox->Location = System
    ::Drawing::Point(104, 9);
this->extractionMethodComboBox->Name = L"
    extractionMethodComboBox";
this->extractionMethodComboBox->Size = System::
    Drawing::Size(140, 21);
this->extractionMethodComboBox->TabIndex = 4;
this->extractionMethodComboBox->
    SelectedIndexChanged += gcnew System::
    EventHandler(this, &ExtractionBrowser::
    extractionMethodComboBox_SelectedIndexChanged)
    ;
//
// eegChComboBox
//
this->eegChComboBox->DropDownStyle = System::
    Windows::Forms::ComboBoxStyle::DropDownList;
this->eegChComboBox->FlatStyle = System::Windows
    ::Forms::FlatStyle::Popup;
this->eegChComboBox->FormattingEnabled = true;
this->eegChComboBox->Items->AddRange(gcnew cli::
    array< System::Object^  >(4) { L"F3", L"FC5",
    L"FC6", L"F4" });
this->eegChComboBox->Location = System::Drawing::
    Point(104, 41);
this->eegChComboBox->Name = L"eegChComboBox";
this->eegChComboBox->Size = System::Drawing::Size
    (140, 21);
this->eegChComboBox->TabIndex = 3;
```

```
this->eegChComboBox->SelectedIndexChanged +=
    gcnew System::EventHandler(this, &
    ExtractionBrowser::
    eegChComboBox_SelectedIndexChanged);
//
// eegChLabel
//
this->eegChLabel->AutoSize = true;
this->eegChLabel->Location = System::Drawing::
    Point(5, 42);
this->eegChLabel->Name = L"eegChLabel";
this->eegChLabel->Size = System::Drawing::Size
    (74, 13);
this->eegChLabel->TabIndex = 2;
this->eegChLabel->Text = L"EEG Channel:";
//
// extractionMethodLabel
//
this->extractionMethodLabel->Anchor = static_cast
    <System::Windows::Forms::AnchorStyles>(((
    System::Windows::Forms::AnchorStyles::Top |
    System::Windows::Forms::AnchorStyles::Bottom)
         | System::Windows::Forms::AnchorStyles::
            Left));
this->extractionMethodLabel->AutoSize = true;
this->extractionMethodLabel->Location = System::
    Drawing::Point(5, 9);
this->extractionMethodLabel->Name = L"
    extractionMethodLabel";
this->extractionMethodLabel->Size = System::
    Drawing::Size(96, 13);
this->extractionMethodLabel->TabIndex = 0;
this->extractionMethodLabel->Text = L"Extraction
    Method:";
//
// extractionChart
//
this->extractionChart->BorderSkin->SkinStyle =
    System::Windows::Forms::DataVisualization::
    Charting::BorderSkinStyle::Sunken;
chartArea1->AxisX->LabelStyle->Format = L"N0";
chartArea1->AxisX->MaximumAutoSize = 90;
chartArea1->AxisX->ScaleBreakStyle->Enabled =
    true;
chartArea1->AxisX->ScaleBreakStyle->
    MaxNumberOfBreaks = 5;
chartArea1->CursorX->AutoScroll = false;
chartArea1->CursorX->IsUserEnabled = true;
chartArea1->CursorX->IsUserSelectionEnabled =
    true;
chartArea1->Name = L"muChartArea";
chartArea2->AxisX->LabelStyle->Format = L"N0";
chartArea2->AxisX->MaximumAutoSize = 90;
chartArea2->AxisX->ScaleBreakStyle->Enabled =
    true;
chartArea2->AxisX->ScaleBreakStyle->
    MaxNumberOfBreaks = 5;
chartArea2->CursorX->AutoScroll = false;
chartArea2->CursorX->IsUserEnabled = true;
```

```
chartArea2->CursorX->IsUserSelectionEnabled =
    true;
chartArea2->Name = L"betaChartArea";
this->extractionChart->ChartAreas->Add(chartArea1
    );
this->extractionChart->ChartAreas->Add(chartArea2
    );
this->extractionChart->Dock = System::Windows::
    Forms::DockStyle::Fill;
legend1->Name = L"Legend1";
this->extractionChart->Legends->Add(legend1);
this->extractionChart->Location = System::Drawing
    ::Point(0, 0);
this->extractionChart->Name = L"extractionChart";
this->extractionChart->Palette = System::Windows
    ::Forms::DataVisualization::Charting::
    ChartColorPalette::EarthTones;
series1->ChartArea = L"muChartArea";
series1->ChartType = System::Windows::Forms::
    DataVisualization::Charting::SeriesChartType::
    Spline;
series1->CustomProperties = L"IsXAxisQuantitative
    =True";
series1->Legend = L"Legend1";
series1->Name = L"F3    ";
series2->ChartArea = L"betaChartArea";
series2->ChartType = System::Windows::Forms::
    DataVisualization::Charting::SeriesChartType::
    Spline;
series2->CustomProperties = L"IsXAxisQuantitative
    =True";
series2->Legend = L"Legend1";
series2->Name = L"F3    ";
series3->ChartArea = L"muChartArea";
series3->ChartType = System::Windows::Forms::
    DataVisualization::Charting::SeriesChartType::
    Spline;
series3->CustomProperties = L"IsXAxisQuantitative
    =True";
series3->Legend = L"Legend1";
series3->Name = L"FC5    ";
series4->ChartArea = L"betaChartArea";
series4->ChartType = System::Windows::Forms::
    DataVisualization::Charting::SeriesChartType::
    Spline;
series4->CustomProperties = L"IsXAxisQuantitative
    =True";
series4->Legend = L"Legend1";
series4->Name = L"FC5    ";
series5->ChartArea = L"muChartArea";
series5->ChartType = System::Windows::Forms::
    DataVisualization::Charting::SeriesChartType::
    Spline;
series5->CustomProperties = L"IsXAxisQuantitative
    =True";
series5->Legend = L"Legend1";
series5->Name = L"FC6    ";
series6->ChartArea = L"betaChartArea";
series6->ChartType = System::Windows::Forms::
```

```
    DataVisualization::Charting::SeriesChartType::
    Spline;
series6->CustomProperties = L"IsXAxisQuantitative
    =True";
series6->Legend = L"Legend1";
series6->Name = L"FC6    ";
series7->ChartArea = L"muChartArea";
series7->ChartType = System::Windows::Forms::
    DataVisualization::Charting::SeriesChartType::
    Spline;
series7->CustomProperties = L"IsXAxisQuantitative
    =True";
series7->Legend = L"Legend1";
series7->Name = L"F4    ";
series8->ChartArea = L"betaChartArea";
series8->ChartType = System::Windows::Forms::
    DataVisualization::Charting::SeriesChartType::
    Spline;
series8->CustomProperties = L"IsXAxisQuantitative
    =True";
series8->Legend = L"Legend1";
series8->Name = L"F4    ";
this->extractionChart->Series->Add(series1);
this->extractionChart->Series->Add(series2);
this->extractionChart->Series->Add(series3);
this->extractionChart->Series->Add(series4);
this->extractionChart->Series->Add(series5);
this->extractionChart->Series->Add(series6);
this->extractionChart->Series->Add(series7);
this->extractionChart->Series->Add(series8);
this->extractionChart->Size = System::Drawing::
    Size(528, 540);
this->extractionChart->TabIndex = 0;
this->extractionChart->Text = L"Extraction";
this->extractionChart->Click += gcnew System::
    EventHandler(this, &ExtractionBrowser::
    extractionChart_Click);
//
// extractionBackgroundWorker
//
this->extractionBackgroundWorker->
    WorkerReportsProgress = true;
this->extractionBackgroundWorker->
    WorkerSupportsCancellation = true;
this->extractionBackgroundWorker->DoWork += gcnew
     System::ComponentModel::DoWorkEventHandler(
    this, &ExtractionBrowser::
    extractionBackgroundWorker_DoWork);
this->extractionBackgroundWorker->
    RunWorkerCompleted += gcnew System::
    ComponentModel::RunWorkerCompletedEventHandler
    (this, &ExtractionBrowser::
    extractionBackgroundWorker_RunWorkerCompleted)
    ;
//
// extractionTimer
//
this->extractionTimer->Interval = 50;
this->extractionTimer->Tick += gcnew System::
```

```
            EventHandler(this, &ExtractionBrowser::
            extractionTimer_Tick);
        //
        // extractionGraphBackgroundWorker
        //
        this->extractionGraphBackgroundWorker->
            WorkerReportsProgress = true;
        this->extractionGraphBackgroundWorker->
            WorkerSupportsCancellation = true;
        this->extractionGraphBackgroundWorker->DoWork +=
            gcnew System::ComponentModel::
            DoWorkEventHandler(this, &ExtractionBrowser::
            extractionGraphBackgroundWorker_DoWork);
        this->extractionGraphBackgroundWorker->
            ProgressChanged += gcnew System::
            ComponentModel::ProgressChangedEventHandler(
            this, &ExtractionBrowser::
            extractionGraphBackgroundWorker_ProgressChanged
            );
        this->extractionGraphBackgroundWorker->
            RunWorkerCompleted += gcnew System::
            ComponentModel::RunWorkerCompletedEventHandler
            (this, &ExtractionBrowser::
            extractionGraphBackgroundWorker_RunWorkerCompleted
            );
        //
        // extractionGraphTimer
        //
        this->extractionGraphTimer->Interval = 50;
        this->extractionGraphTimer->Tick += gcnew System
            ::EventHandler(this, &ExtractionBrowser::
            extractionGraphTimer_Tick);
        //
        // ExtractionBrowser
        //
        this->AutoScaleDimensions = System::Drawing::
            SizeF(6, 13);
        this->AutoScaleMode = System::Windows::Forms::
            AutoScaleMode::Font;
        this->ClientSize = System::Drawing::Size(784,
            562);
        this->Controls->Add(this->
            extractionToolStripContainer);
        this->Icon = (cli::safe_cast<System::Drawing::
            Icon^>(resources->GetObject(L"$this.Icon")));
        this->MaximizeBox = false;
        this->Name = L"ExtractionBrowser";
        this->ShowIcon = false;
        this->ShowInTaskbar = false;
        this->Text = L"Extraction Browser";
        this->WindowState = System::Windows::Forms::
            FormWindowState::Maximized;
        this->extractionToolStripContainer->
            BottomToolStripPanel->ResumeLayout(false);
        this->extractionToolStripContainer->
            BottomToolStripPanel->PerformLayout();
        this->extractionToolStripContainer->ContentPanel
            ->ResumeLayout(false);
        this->extractionToolStripContainer->ResumeLayout(
```

```
                        false);
                    this->extractionToolStripContainer->PerformLayout
                        ();
                    this->extractionStatusStrip->ResumeLayout(false);
                    this->extractionStatusStrip->PerformLayout();
                    this->extractionSplitContainer->Panel1->
                        ResumeLayout(false);
                    this->extractionSplitContainer->Panel2->
                        ResumeLayout(false);
                    (cli::safe_cast<System::ComponentModel::
                        ISupportInitialize^>(this->
                        extractionSplitContainer))->EndInit();
                    this->extractionSplitContainer->ResumeLayout(
                        false);
                    this->extractionFileTableLayoutPanel->
                        ResumeLayout(false);
                    this->extractionFileTableLayoutPanel->
                        PerformLayout();
                    this->extractionMethodPanel->ResumeLayout(false);
                    this->extractionMethodPanel->PerformLayout();
                    (cli::safe_cast<System::ComponentModel::
                        ISupportInitialize^>(this->extractionChart))->
                        EndInit();
                    this->ResumeLayout(false);

                }
#pragma endregion
                Void clearExtraction()
                {
                        extractionToolStripProgressBar->Value = 0;
                        extractionToolStripProgressBar->Visible = false;
                        extractionToolStripStatusLabel->Text = "";
                        extractionChart->SuspendLayout();
                        for (int i = 0; i < 8; i++)
                        {
                                series[i]->Points->Clear();
                                series[i]->IsVisibleInLegend = false;
                        }

                        extractionChart->ResumeLayout(true);
                        extractionChart->Visible = false;

                        if (extractionBackgroundWorker->IsBusy)
                            extractionBackgroundWorker->CancelAsync();
                        if (extractionGraphBackgroundWorker->IsBusy)
                            extractionBackgroundWorker->CancelAsync();
                        extractionTimer->Stop();
                        extractionGraphTimer->Stop();
                        if (extraction != nullptr) {
                                extraction->stop();
                                delete extraction;
                        }
                        extraction = gcnew Extraction();
                }

                Void extractionFileFilter(String^ method)
                {
                        extractionListView->Items->Clear();
                        String^ fileFilter = "";
```

```
                        if (method->CompareTo("Haar Wavelet") == 0)
                        {
                                fileFilter = "*_wav_extraction.csv";
                        }
                        else
                        {
                                fileFilter = "*_ica_extraction.csv";
                        }
                        for each (String^ fileName in Directory::
                            EnumerateFiles(extractFolderName, fileFilter,
                            IO::SearchOption::TopDirectoryOnly))
                        {
                                String^ shortFileName = fileName->
                                    Substring(fileName->LastIndexOf(Path::
                                    DirectorySeparatorChar) + 1);
                                extractionListView->Items->Add(
                                    shortFileName)->ImageIndex = 0;
                        }

                        clearExtraction();

                }
                 Void search(TextBox^ searchTextBox, ListView^
                     csvFileListView)
                 {
                         if (!String::IsNullOrEmpty(searchTextBox->Text)
                             &&
                                 searchTextBox->Text->CompareTo("Search
                                     File") != 0 &&
                                 csvFileListView->Items->Count > 0)
                         { // Call FindItemWithText with the contents of
                             the textbox.
                                 ListViewItem^ foundItem =
                                         csvFileListView->
                                             FindItemWithText(
                                             searchTextBox->Text, false,
                                             0, true);

                                 if (foundItem != nullptr)
                                 {
                                         foundItem->Focused = true;
                                         foundItem->Selected = true;
                                         csvFileListView->TopItem =
                                             foundItem;
                                         csvFileListView->Focus();
                                         acqOpenFileName = foundItem->
                                             Text;
                                         extraction->start(
                                             acqOpenFileName,
                                             extractionMethod);

                                 }
                                 searchTextBox->ForeColor = System::
                                     Drawing::SystemColors::ControlLight;
                                 searchTextBox->Text = "Search File";
                         }
                 }
private: System::Void acqFileSearchTextBox_KeyDown(System::Object^
    sender, System::Windows::Forms::KeyEventArgs^  e) {
```

```
                        if ((e->KeyCode.CompareTo(System::Windows::Forms
                            ::Keys::Enter)) == 0)
                        {
                                search(acqFileSearchTextBox,
                                    acqFileListView);
                        }
}
private: System::Void extractionFileSearchTextBox_KeyDown(System::Object^
     sender, System::Windows::Forms::KeyEventArgs^  e) {
                        if ((e->KeyCode.CompareTo(System::Windows::Forms
                            ::Keys::Enter)) == 0)
                        {
                                search(extractionFileSearchTextBox,
                                    extractionListView);
                        }
}
private: System::Void acqFileListView_MouseDoubleClick(System::Object^
    sender, System::Windows::Forms::MouseEventArgs^  e) {
                        clearExtraction();
                        Collections::IEnumerator^ it = acqFileListView->
                            SelectedItems->GetEnumerator();
                        it->MoveNext();
                        ListViewItem^ currFile = (ListViewItem^)it->
                            Current;
                        acqOpenFileName = currFile->Text;
                        extraction->start(acqOpenFileName,
                            extractionMethod);
                        extractionTimer->Start();
                        extractionToolStripProgressBar->Visible = true;
                        extractionToolStripStatusLabel->Text = "Loading
                            ...";
}
private: System::Void extractionTimer_Tick(System::Object^  sender,
    System::EventArgs^  e) {
                        if (!extractionBackgroundWorker->IsBusy && !
                            extraction->isStopped())
                        {
                                extractionBackgroundWorker->
                                    RunWorkerAsync();
                        }
                        else
                        {
                                if (!extraction->isDone())
                                {
                                        extractionToolStripStatusLabel->
                                            Text = "Loading...";
                                        if (
                                            extractionToolStripProgressBar
                                            ->Value >= 1000)
                                        {
                                                extractionToolStripProgressBar
                                                    ->Value = 500;
                                        }
                                        extractionToolStripProgressBar->
                                            PerformStep();
                                        extractionTimer->Start();
                                }
                        }
```

```
}
private: System::Void acqFileListView_SelectedIndexChanged(System::Object
    ^  sender, System::EventArgs^  e) {
                            extractionTimer->Stop();
                            extractionGraphTimer->Stop();

                            if (extractionBackgroundWorker->IsBusy)
                            {
                                    extractionBackgroundWorker->CancelAsync
                                        ();
                            }
                            if (extractionGraphBackgroundWorker->IsBusy)
                            {
                                    extractionGraphBackgroundWorker->
                                        CancelAsync();
                            }
                            clearExtraction();
                            extractionFileFilter(extractionMethod);

}
private: System::Void extractionBackgroundWorker_DoWork(System::Object^
    sender, System::ComponentModel::DoWorkEventArgs^  e) {
                            if (extraction != nullptr)
                            {
                                    extractionTimer->Stop();
                                    extraction->extract(extractionMethod);
                                    if (extraction->isStopped())
                                    {
                                            extraction->stop();
                                    }
                            }
}
private: System::Void extractionBackgroundWorker_RunWorkerCompleted(
    System::Object^  sender, System::ComponentModel::
    RunWorkerCompletedEventArgs^  e) {
                            extractionToolStripProgressBar->Value = 1000;
                            extractionToolStripStatusLabel->Text = "Done";
                            extractionFileFilter(extractionMethod);
}
                    private: System::Void startGraph()
                    {
                               clearExtraction();
                                   Collections::IEnumerator^ it =
                                      extractionListView->SelectedItems->
                                      GetEnumerator();
                                   if (it->MoveNext())
                                   {
                                           ListViewItem^ currFile = (
                                              ListViewItem^)it->Current;
                                           extractionOpenFileName = currFile
                                              ->Text;
                                   }
                                   if (extractionOpenFileName->CompareTo("")
                                      !=0)
                                   {
                                           extractionGraphTimer->Start();
                                           extractionToolStripProgressBar->
                                              Visible = true;
                                           extractionToolStripStatusLabel->
```

```
                                        Text = "Loading...";
                                }
                }
private: System::Void extractionListView_MouseDoubleClick(System::Object^
     sender, System::Windows::Forms::MouseEventArgs^  e) {
                        startGraph();
                        starting = false;
}
private: System::Void extractionGraphBackgroundWorker_RunWorkerCompleted(
   System::Object^  sender, System::ComponentModel::
   RunWorkerCompletedEventArgs^  e) {

        List<array<DataPoint^>^>^ dataPoints = extraction->getDataPoints
            ();
        int datasetSize = extraction->getDatasetSize();
        String^ selectedChannel = (String^) eegChComboBox->SelectedItem;
        int start = 0;
        int end = 0;
        if (selectedChannel->CompareTo("F3") == 0)
        {
                start = 0;
                end = 2;
        }
                else if (selectedChannel->CompareTo("FC5") == 0)
            {
                        start = 2;
                        end = 4;
                }
                else if (selectedChannel->CompareTo("FC6") == 0)
                {
                        start = 4;
                        end = 6;
                }
                else if (selectedChannel->CompareTo("F4") == 0)
                {
                        start = 6;
                        end = 8;
                }
                series[start]->IsVisibleInLegend = true;
                series[end -1]->IsVisibleInLegend = true;
                for each (array<DataPoint^>^ samples in dataPoints)
                {
                        for (int i = start; i < end; i++)
                        {

                        if (samples[i]->YValues->Length == 0)
                                {
                                        continue;
                                }
                                samples[i]->XValue = samples[i]->XValue;
                                System::Collections::Generic::
                                   IEnumerator<Series^>^ ie =
                                   extractionChart->Series->
                                   GetEnumerator();

                                if (samples[i]->YValues->Length <
                                   datasetSize  && samples[i]->YValues->
                                   Length != 0)
                                {
```

```
                                        while (ie->MoveNext())
                                        {
                                                ie->Current->
                                                    YValuesPerPoint =
                                                    samples[i]->YValues->
                                                    Length;
                                        }
                                }
                                series[i]->Points->Add(samples[i]);
                        }
                }

                if (!extraction->isDone())
                {
                        extractionToolStripStatusLabel->Text = "Loading
                            ...";
                        if (extractionToolStripProgressBar->Value >=
                            1000)
                        {
                                extractionToolStripProgressBar->Value =
                                    500;
                        }
                        extractionToolStripProgressBar->PerformStep();
                        extractionGraphTimer->Start();
                }
                else
                {
                        extractionToolStripProgressBar->Value = 1000;
                        extractionToolStripStatusLabel->Text = "Done";
                        extractionGraphTimer->Stop();
                        extractionChart->UseWaitCursor = false;
                        extractionChart->Visible = true;
                }
}

private: System::Void extractionGraphTimer_Tick(System::Object^  sender,
    System::EventArgs^  e) {
                        if (!extractionGraphBackgroundWorker->IsBusy &&
                            !extraction->isStopped())
                        {
                                extractionGraphBackgroundWorker->
                                    RunWorkerAsync();
                        }
                        else
                        {
                                extractionGraphTimer->Stop();
                                extractionGraphBackgroundWorker->
                                    CancelAsync();
                        }
}
private: System::Void extractionListView_SelectedIndexChanged(System::
    Object^  sender, System::EventArgs^  e) {
                        clearExtraction();
}
private: System::Void extractionMethodComboBox_SelectedIndexChanged(
    System::Object^  sender, System::EventArgs^  e) {
                        extractionMethod = (String^)
                            extractionMethodComboBox->SelectedItem;
                        extractionFileFilter(extractionMethod);
```

```
                        extractionOpenFileName = "";
}
private: System::Void extractionGraphBackgroundWorker_DoWork(System::
    Object^  sender, System::ComponentModel::DoWorkEventArgs^  e) {
                        if (extraction != nullptr)
                        {
                                extractionGraphTimer->Stop();
                                extraction->graph(extractionOpenFileName
                                    );
                                if (extraction->isStopped())
                                {
                                        extraction->stop();
                                }
                        }

}
private: System::Void extractionGraphBackgroundWorker_ProgressChanged(
    System::Object^  sender, System::ComponentModel::
    ProgressChangedEventArgs^  e) {
                        if (!extraction->isDone())
                        {
                                extractionToolStripStatusLabel->Text = "
                                    Loading...";
                                if (extractionToolStripProgressBar->
                                    Value >= 1000)
                                {
                                        extractionToolStripProgressBar->
                                            Value = 500;
                                }
                                extractionToolStripProgressBar->
                                    PerformStep();
                                extractionGraphTimer->Start();
                        }
                        else
                        {
                                extractionToolStripProgressBar->Value =
                                    1000;
                                extractionToolStripStatusLabel->Text = "
                                    Done";
                                extractionGraphTimer->Stop();
                                extractionChart->UseWaitCursor = false;
                                extractionChart->Visible = true;
                        }
}
private: System::Void eegChComboBox_SelectedIndexChanged(System::Object^
     sender, System::EventArgs^  e) {
                        if (!starting){
                                startGraph();
                                starting = false;
                        }

}
private: System::Void extractionChart_Click(System::Object^  sender,
    System::EventArgs^  e) {
}
};
}
```

Código 7.15: Cabecera con declaración de funciones para entrenamiento

```
#pragma once
#include <map>
#include <string>
namespace EEGTraining
{
        using namespace System;
        using namespace System::IO;
        using namespace std;
        using namespace System::Collections;
        using namespace System::Collections::Generic;

        ref class Training
        {
        public:
                Training(String^ extractionDirectory, List<String^>^
                   fileList);
                Training(String^ trainingDirectory, String^
                   trainingFileName);
                array<double>^ getMeans() { return means; }
                array<double>^ getVariance() { return variance; }
                array<double, 2>^ getCovarianceMatrix(){ return
                   covariance; }
                void train();
                void view();

        private:
                array<String^>^ SEPARATOR;
                String^ extractionDirectory;
                String^ trainingDirectory;
                List<String^>^ fileList;
                String^ trainingFileName;
                array<double>^ means = gcnew array<double>(8);
                array<double>^ variance = gcnew array<double>(8);
                array<int>^   numSamples = gcnew array<int>(8);
                array<double, 2>^ covariance = gcnew array<double, 2>(8,
                   8);
                void save();
                void calculateMean();
                void calculateVariance();
                void calculateMatrix();
        };
}
```

Código 7.16: Definición de funciones para entrenamiento

```
#include "Training.h"
#include <atlstr.h>
#include <math.h>

namespace EEGTraining
{
        using namespace System;

        Training::Training(String^ extractionDirectory, List<String^>^
           fileList)
        {
```

```
            this->extractionDirectory = extractionDirectory;
            this->fileList = fileList;
            this->trainingDirectory = extractionDirectory->Substring
                (0, extractionDirectory->LastIndexOf("\\Extraction"))
                + "\\Training\\";

            SEPARATOR = gcnew array<String^>(1);
            SEPARATOR[0] = ",";
    }

    Training::Training(String^ trainingDirectory, String^
        trainingFileName)
    {
            this->trainingDirectory = trainingDirectory;
            this->trainingFileName = trainingFileName;
            SEPARATOR = gcnew array<String^>(1);
            SEPARATOR[0] = ",";
    }

    void Training::train()
    {
            calculateMean();
            //calculateVariance();
            calculateMatrix();
            save();
    }

    void Training::calculateMean()
    {
            for each(String^ file in fileList)
            {
                    StreamReader^ inputFile = File::OpenText(
                        extractionDirectory->Concat(
                        extractionDirectory, file));
                    String^ header = inputFile->ReadLine();

                    while (!inputFile->EndOfStream)
                    {
                            String^ line = inputFile->ReadLine();
                            array<String^>^ columns
                                    = line->Split(SEPARATOR, System::
                                        StringSplitOptions::None);
                            int j = 0;
                            for (int i = 1; i < 5; i++)
                            {
                                    if (columns[i]->CompareTo("na")
                                        != 0)
                                    {
                                            if (columns[i+4]->
                                                CompareTo("MU") == 0)
                                            {
                                                    means[j] += atof(
                                                        CString(
                                                        columns[i]));
                                                    numSamples[j] +=
                                                        1;
                                            }
                                            else
                                            {
```

```
                                                means[j+1] +=
                                                    atof(CString(
                                                    columns[i]));
                                                numSamples[j+1]
                                                    += 1;
                                        }
                                }
                                j+=2;
                        }
                }
                inputFile->Close();
        }

        for (int i = 0; i < means->Length; i++)
        {
                if (numSamples[i] > 0)
                {
                        means[i] = means[i] / numSamples[i];
                }
        }
}

void Training::calculateVariance()
{
        for (int i = 0; i < 8; i++)
        {
                numSamples[i] = 0;
        }
        for each(String^ file in fileList)
        {
                StreamReader^ inputFile = File::OpenText(
                    extractionDirectory->Concat(
                    extractionDirectory, file));
                String^ header = inputFile->ReadLine();

                while (!inputFile->EndOfStream)
                {
                        String^ line = inputFile->ReadLine();
                        array<String^>^ columns
                                = line->Split(SEPARATOR, System::
                                    StringSplitOptions::None);
                        int j = 0;
                        double value = means[j];

                        for (int i = 1; i < 5; i++)
                        {
                                if (columns[i]->CompareTo("na")
                                    != 0)
                                {
                                        value = atof(CString(
                                            columns[i]));
                                }
                                if (columns[i + 4]->CompareTo("MU
                                    ") == 0)
                                {
                                        variance[j] += pow((value
                                            - means[j]), 2);
                                        numSamples[j]++;
                                }
```

```
                                        else
                                        {
                                                variance[j + 1] += pow((
                                                    value - means[j + 1]),
                                                     2);
                                                numSamples[j+1]++;
                                        }

                                        j += 2;
                                }

                        }
                        inputFile->Close();
                }

                for (int i = 0; i < variance ->Length; i++)
                {
                        if (numSamples[i] > 0)
                        {
                                variance[i] = variance[i] / (numSamples[i
                                    ] -1);
                                //covariance[i, i] = variance[i];
                        }
                }
        }
        void Training::calculateMatrix()
        {
                FileStream^ output = File::Open(trainingDirectory + "
                    trainingValues.csv", FileMode::Create);

                StreamWriter^ outputFile = gcnew StreamWriter(output);
                outputFile->WriteLine("F3MU,F3BETA,FC5MU,FC5BETA,FC6MU,
                    FC6BETA,F4MU,F4BETA");
                int totalSamples = 0;
                for each(String^ file in fileList)
                {
                        StreamReader^ inputFile = File::OpenText(
                            extractionDirectory->Concat(
                            extractionDirectory, file));
                        String^ header = inputFile->ReadLine();

                        while (!inputFile->EndOfStream)
                        {
                                String^ line = inputFile->ReadLine();
                                array<String^>^ columns
                                        = line->Split(SEPARATOR, System::
                                            StringSplitOptions::None);
                                int j = 0;

                                array<double>^ values = gcnew array<
                                    double>(8);

                                //Initialize values with the
                                    corresponding mean
                                for (int i = 0; i < values->Length; i++)
                                {
                                        values[i] = means[i];
                                }
                                //Transform data in one row and fill
```

```
                        wholes with mean
                for (int i = 1; i < 5; i++)
                {
                        if (columns[i]->CompareTo("na")
                            != 0)
                        {
                                if (columns[i + 4]->
                                    CompareTo("MU") == 0)
                                {
                                        values[j] = atof(
                                            CString(
                                            columns[i]));
                                }
                                else
                                {
                                        values[j + 1] =
                                            atof(CString(
                                            columns[i]));
                                }
                        }

                        j += 2;
                }
                String^ allValues = "";
                for (int k = 0; k < 8; k++)
                {
                        allValues += gcnew String(
                            to_string(values[k]).c_str());
                        if(k < 7) allValues +=",";
                }
                outputFile->WriteLine(allValues);
                //Multiply values for the variable
                    combination
                for (int m = 0; m < 8; m++)
                {
                        for (int n = 0; n < 8; n++)
                        {
                                covariance[m, n] += (
                                    values[m]-means[m]) *
                                    (values[n] - means[n])
                                    ;
                        }
                }
                totalSamples++;
        }
        inputFile->Close();
}
// Calulate the means of the products and substract
    product of the means
totalSamples--;
for (int m = 0; m < 8; m++)
{
        for (int n = 0; n < 8; n++)
        {

                covariance[m, n] = (covariance[m, n] /
                    totalSamples);
        }
}
```

```
            outputFile->Close();

        }

        void Training::save()
        {
            trainingFileName = fileList[0];
            array<String^>^ splitChar = gcnew array<String^>(1);
            splitChar[0] = "_";
            array<String^>^ parts = trainingFileName->Split(splitChar
                , System::StringSplitOptions::None);
            trainingFileName = "";
            trainingFileName = parts[0] + splitChar[0] + parts[1] +
                        "_" + parts[parts->Length - 1]->Substring
                            (0, parts[parts->Length - 1]->IndexOf(".
                            ")) +
                             "_training.csv";
            trainingDirectory = extractionDirectory->Substring(0,
                extractionDirectory->LastIndexOf("Extraction\\")) + "
                Training\\";
            FileStream^ output = File::Open(trainingDirectory +
                trainingFileName, FileMode::Create);
            StreamWriter^ outputFile = gcnew StreamWriter(output);

            outputFile->WriteLine("WAVE,MEAN,F3MU,F3BETA,FC5MU,
                FC5BETA,FC6MU,FC6BETA,F4MU,F4BETA");
            array<String^>^ waves = gcnew array<String^> {"F3MU", "
                F3BETA", "FC5MU", "FC5BETA", "FC6MU", "FC6BETA", "F4MU
                ", "F4BETA"};
            for (int i = 0; i < 8; i++)
            {
                String^ result = waves[i] + "," + gcnew String(
                    to_string(means[i]).c_str());
                for (int j = 0; j < 8; j++)
                {
                    result += ",";
                    result += gcnew String(to_string(
                        covariance[i, j]).c_str());
                }
                outputFile->WriteLine(result);
            }
            outputFile->Close();
        }

        void Training::view()
        {
            StreamReader^ inputFile = File::OpenText(
                trainingDirectory + trainingFileName);
            String^ header = inputFile->ReadLine();
            int i = 0;
            while (!inputFile->EndOfStream)
            {
                String^ line = inputFile->ReadLine();
                array<String^>^ columns
                        = line->Split(SEPARATOR, System::
                            StringSplitOptions::None);
                //Do not start in 0 because it is info not needed
                for (int j = 1; j < columns->Length; j++)
```

```
                        {
                                Double value = atof(CString(columns[j]));
                                if (j == 1)
                                {
                                        means[i] = value;
                                }
                                else
                                {
                                        covariance[i, j - 2] = value;
                                        if (i == (j - 2))
                                        {
                                                variance[i] = value;
                                        }
                                }
                        }
                        i++;
                }
                inputFile->Close();
        }
}
```

Código 7.17: Pantalla de entrenamiento

```
#pragma once
#include "training.h"

namespace EEGRECORD {

        using namespace System;
        using namespace System::ComponentModel;
        using namespace System::Collections;
        using namespace System::Collections::Generic;
        using namespace System::Windows::Forms;
        using namespace System::Data;
        using namespace System::Drawing;
        using namespace System::IO;
        using namespace System::Reflection;
        using namespace EEGTraining;


        /// <summary>
        /// Resumen de TrainingScreen
        /// </summary>
        public ref class TrainingScreen : public System::Windows::Forms::
           Form
        {
        private:
                String^ extractFolderName;
                String^ trainingFolderName;
                String^ trainingFileName;
                bool starting = true;
                Training^ training;

        private: System::Windows::Forms::Button^  deselectAllButton;
        private: System::Windows::Forms::Button^  selectAllButton;
        private: System::Windows::Forms::Button^  trainButton;
        private: System::ComponentModel::BackgroundWorker^
           trainingBackgroundWorker;
        private: System::Windows::Forms::Timer^  trainingTimer;
```

```
        private: System::Windows::Forms::Panel^  trainingMediaPanel;
        private: System::Windows::Forms::GroupBox^  meanGroupBox;
        private: System::Windows::Forms::GroupBox^
            covarianceMatrixGroupBox;
        private: System::Windows::Forms::DataGridView^
            covarianceDataGridView;
        private: System::Windows::Forms::DataGridView^  meanDataGridView;
        private: System::Windows::Forms::TableLayoutPanel^
            traningResultsTableLayoutPanel;
        private: System::Windows::Forms::ListView^  trainingFileListView;
        private: System::Windows::Forms::ColumnHeader^
            trainingFilesColumnHeader;
                        System::Windows::Forms::DataGridViewCellStyle^
                            headerDataGridViewCellStyle = (gcnew System::
                            Windows::Forms::DataGridViewCellStyle());


        public:
                TrainingScreen(String^ folderName)
                {
                        InitializeComponent();
                        trainingTimer->Stop();

                        extractionMethodComboBox->SelectedIndex = 0;
                        movementTypeComboBox->SelectedIndex = 0;
                        extractFolderName = folderName->Concat(folderName
                            , Path::DirectorySeparatorChar);
                        extractFolderName = extractFolderName->Concat(
                            extractFolderName, "Extraction");
                        extractFolderName = extractFolderName->Concat(
                            extractFolderName, Path::
                            DirectorySeparatorChar);

                        trainingFolderName = folderName->Concat(
                            folderName, Path::DirectorySeparatorChar);
                        trainingFolderName += "Training" + Path::
                            DirectorySeparatorChar;
                        subjectIdFilter();
                        subjectIdComboBox->SelectedIndex = 0;


                        DoubleBuffered(meanDataGridView, true);
                        DoubleBuffered(covarianceDataGridView, true);
                        trainingToolStripProgressBar->Visible = false;
                        trainingToolStripStatusLabel->Text = "";
                        headerDataGridViewCellStyle->Alignment = System::
                            Windows::Forms::DataGridViewContentAlignment::
                            MiddleCenter;
                        headerDataGridViewCellStyle->BackColor = System::
                            Drawing::SystemColors::Control;
                        headerDataGridViewCellStyle->Font = (gcnew System
                            ::Drawing::Font(L"Microsoft Sans Serif", 8.25F
                            , System::Drawing::FontStyle::Regular,
                                System::Drawing::GraphicsUnit::Point,
                                    static_cast<System::Byte>(0)));
                        headerDataGridViewCellStyle->ForeColor = System::
                            Drawing::SystemColors::WindowText;
                        headerDataGridViewCellStyle->SelectionBackColor =
                            System::Drawing::SystemColors::Control;
```

```
headerDataGridViewCellStyle->SelectionForeColor =
    System::Drawing::SystemColors::WindowText;
headerDataGridViewCellStyle->WrapMode = System::
   Windows::Forms::DataGridViewTriState::True;

meanDataGridView->ColumnCount = 9;
meanDataGridView->RowCount = 1;
meanDataGridView->Columns[0]->HeaderText = L"
   ";
meanDataGridView->Columns[1]->HeaderText = L"F3
      ";
meanDataGridView->Columns[2]->HeaderText = L"F3
      ";
meanDataGridView->Columns[3]->HeaderText = L"FC5
      ";
meanDataGridView->Columns[4]->HeaderText = L"FC5
      ";
meanDataGridView->Columns[5]->HeaderText = L"FC6
      ";
meanDataGridView->Columns[6]->HeaderText = L"FC6
      ";
meanDataGridView->Columns[7]->HeaderText = L"F4
      ";
meanDataGridView->Columns[8]->HeaderText = L"F4
      ";
int columnWidth = 56;

meanDataGridView->Columns[0]->Width = 69;
meanDataGridView->Columns[1]->Width = columnWidth
   ;
meanDataGridView->Columns[2]->Width = columnWidth
   ;
meanDataGridView->Columns[3]->Width = columnWidth
   ;
meanDataGridView->Columns[4]->Width = columnWidth
   ;
meanDataGridView->Columns[5]->Width = columnWidth
   ;
meanDataGridView->Columns[6]->Width = columnWidth
   ;
meanDataGridView->Columns[7]->Width = columnWidth
   ;
meanDataGridView->Columns[8]->Width = columnWidth
   ;
meanDataGridView[0, 0]->Value = "  ";
meanDataGridView[0, 0]->Style =
   headerDataGridViewCellStyle;
DataGridViewAdvancedBorderStyle^ borderStyle =
   gcnew DataGridViewAdvancedBorderStyle();
borderStyle->All =
   DataGridViewAdvancedCellBorderStyle::
   InsetDouble;
meanDataGridView[0, 0]->AdjustCellBorderStyle(
   borderStyle, borderStyle,true,true, true,true)
   ;
covarianceDataGridView->ColumnCount = 9;
covarianceDataGridView->RowCount = 8;

covarianceDataGridView->Columns[0]->HeaderText =
```

```
                    L"    ";
                covarianceDataGridView->Columns[1]->HeaderText =
                    L"F3   ";
                covarianceDataGridView->Columns[2]->HeaderText =
                    L"F3   ";
                covarianceDataGridView->Columns[3]->HeaderText =
                    L"FC5   ";
                covarianceDataGridView->Columns[4]->HeaderText =
                    L"FC5   ";
                covarianceDataGridView->Columns[5]->HeaderText =
                    L"FC6   ";
                covarianceDataGridView->Columns[6]->HeaderText =
                    L"FC6   ";
                covarianceDataGridView->Columns[7]->HeaderText =
                    L"F4   ";
                covarianceDataGridView->Columns[8]->HeaderText =
                    L"F4   ";
                covarianceDataGridView->Columns[0]->Width = 69;
                covarianceDataGridView->Columns[1]->Width =
                    columnWidth;
                covarianceDataGridView->Columns[2]->Width =
                    columnWidth;
                covarianceDataGridView->Columns[3]->Width =
                    columnWidth;
                covarianceDataGridView->Columns[4]->Width =
                    columnWidth;
                covarianceDataGridView->Columns[5]->Width =
                    columnWidth;
                covarianceDataGridView->Columns[6]->Width =
                    columnWidth;
                covarianceDataGridView->Columns[7]->Width =
                    columnWidth;
                covarianceDataGridView->Columns[8]->Width =
                    columnWidth;
                array<String^>^ rowHeaders = gcnew array<String^>
                     {L"F3   ", L"F3   ", L"FC5   ", L"FC5   ", L"
                    FC6   ", L"FC6   ", L"F4   ", L"F4   "};
                for (int i = 0; i < 8; i++)
                {
                        covarianceDataGridView[0, i]->Value =
                            rowHeaders[i];
                        covarianceDataGridView[0, i]->Style =
                            headerDataGridViewCellStyle;

                }



        }

protected:
        /// <summary>
        /// Limpiar los recursos que se est n utilizando.
        /// </summary>
        ~TrainingScreen()
        {
                if (components)
                {
                        delete components;
```

```
                    }
                }
        private: System::Windows::Forms::ToolStripContainer^
            trainingToolStripContainer;
        private: System::Windows::Forms::StatusStrip^
            trainingStatusStrip;
        private: System::Windows::Forms::ToolStripStatusLabel^
            trainingToolStripStatusLabel;
        private: System::Windows::Forms::ToolStripProgressBar^
            trainingToolStripProgressBar;
        private: System::Windows::Forms::SplitContainer^
            trainingSplitContainer;
        private: System::Windows::Forms::TableLayoutPanel^
            trainingTableLayoutPanel;
        private: System::Windows::Forms::Panel^  trainingSetupPanel;

        private: System::Windows::Forms::ComboBox^
            extractionMethodComboBox;
        private: System::Windows::Forms::Label^  methodLabel;
        private: System::Windows::Forms::ComboBox^  movementTypeComboBox;
        private: System::Windows::Forms::Label^  movementTypeLabel;
        private: System::Windows::Forms::ComboBox^  subjectIdComboBox;
        private: System::Windows::Forms::Label^  subjectIdLabel;
        private: System::Windows::Forms::ImageList^  csvImageList;
private: System::Windows::Forms::ListView^  extractionFileListView;


        private: System::ComponentModel::IContainer^  components;

        protected:

        protected:

        private:
                /// <summary>
                /// Variable del dise ador requerida.
                /// </summary>


#pragma region Windows Form Designer generated code
                /// <summary>
                /// M todo necesario para admitir el Dise ador. No se
                    puede modificar
                /// el contenido del m todo con el editor de c digo.
                /// </summary>
                void InitializeComponent(void)
                {
                        this->components = (gcnew System::ComponentModel
                            ::Container());
                        System::Windows::Forms::ColumnHeader^
                            extractionColumnHeader;
                        System::ComponentModel::ComponentResourceManager^
                             resources = (gcnew System::ComponentModel::
                            ComponentResourceManager(TrainingScreen::
                            typeid));
                        System::Windows::Forms::DataGridViewCellStyle^
                            dataGridViewCellStyle1 = (gcnew System::
                            Windows::Forms::DataGridViewCellStyle());
                        System::Windows::Forms::DataGridViewCellStyle^
```

```
    dataGridViewCellStyle2 = (gcnew System::
    Windows::Forms::DataGridViewCellStyle());
System::Windows::Forms::DataGridViewCellStyle^
    dataGridViewCellStyle3 = (gcnew System::
    Windows::Forms::DataGridViewCellStyle());
System::Windows::Forms::DataGridViewCellStyle^
    dataGridViewCellStyle4 = (gcnew System::
    Windows::Forms::DataGridViewCellStyle());
System::Windows::Forms::DataGridViewCellStyle^
    dataGridViewCellStyle5 = (gcnew System::
    Windows::Forms::DataGridViewCellStyle());
this->trainingToolStripContainer = (gcnew System
    ::Windows::Forms::ToolStripContainer());
this->trainingStatusStrip = (gcnew System::
    Windows::Forms::StatusStrip());
this->trainingToolStripStatusLabel = (gcnew
    System::Windows::Forms::ToolStripStatusLabel()
    );
this->trainingToolStripProgressBar = (gcnew
    System::Windows::Forms::ToolStripProgressBar()
    );
this->trainingSplitContainer = (gcnew System::
    Windows::Forms::SplitContainer());
this->trainingTableLayoutPanel = (gcnew System::
    Windows::Forms::TableLayoutPanel());
this->trainingMediaPanel = (gcnew System::Windows
    ::Forms::Panel());
this->extractionFileListView = (gcnew System::
    Windows::Forms::ListView());
this->csvImageList = (gcnew System::Windows::
    Forms::ImageList(this->components));
this->trainingSetupPanel = (gcnew System::Windows
    ::Forms::Panel());
this->deselectAllButton = (gcnew System::Windows
    ::Forms::Button());
this->selectAllButton = (gcnew System::Windows::
    Forms::Button());
this->trainButton = (gcnew System::Windows::Forms
    ::Button());
this->extractionMethodComboBox = (gcnew System::
    Windows::Forms::ComboBox());
this->methodLabel = (gcnew System::Windows::Forms
    ::Label());
this->movementTypeComboBox = (gcnew System::
    Windows::Forms::ComboBox());
this->movementTypeLabel = (gcnew System::Windows
    ::Forms::Label());
this->subjectIdComboBox = (gcnew System::Windows
    ::Forms::ComboBox());
this->subjectIdLabel = (gcnew System::Windows::
    Forms::Label());
this->traningResultsTableLayoutPanel = (gcnew
    System::Windows::Forms::TableLayoutPanel());
this->meanGroupBox = (gcnew System::Windows::
    Forms::GroupBox());
this->meanDataGridView = (gcnew System::Windows::
    Forms::DataGridView());
this->covarianceMatrixGroupBox = (gcnew System::
    Windows::Forms::GroupBox());
```

```
this->covarianceDataGridView = (gcnew System::
    Windows::Forms::DataGridView());
this->trainingFileListView = (gcnew System::
    Windows::Forms::ListView());
this->trainingFilesColumnHeader = (gcnew System::
    Windows::Forms::ColumnHeader());
this->trainingBackgroundWorker = (gcnew System::
    ComponentModel::BackgroundWorker());
this->trainingTimer = (gcnew System::Windows::
    Forms::Timer(this->components));
extractionColumnHeader = (gcnew System::Windows::
    Forms::ColumnHeader());
this->trainingToolStripContainer->
    BottomToolStripPanel->SuspendLayout();
this->trainingToolStripContainer->ContentPanel->
    SuspendLayout();
this->trainingToolStripContainer->SuspendLayout()
    ;
this->trainingStatusStrip->SuspendLayout();
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->
    trainingSplitContainer))->BeginInit();
this->trainingSplitContainer->Panel1->
    SuspendLayout();
this->trainingSplitContainer->Panel2->
    SuspendLayout();
this->trainingSplitContainer->SuspendLayout();
this->trainingTableLayoutPanel->SuspendLayout();
this->trainingMediaPanel->SuspendLayout();
this->trainingSetupPanel->SuspendLayout();
this->traningResultsTableLayoutPanel->
    SuspendLayout();
this->meanGroupBox->SuspendLayout();
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->meanDataGridView))
    ->BeginInit();
this->covarianceMatrixGroupBox->SuspendLayout();
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->
    covarianceDataGridView))->BeginInit();
this->SuspendLayout();
//
// extractionColumnHeader
//
extractionColumnHeader->Text = L"Extraction File
    Name";
extractionColumnHeader->Width = 340;
//
// trainingToolStripContainer
//
//
// trainingToolStripContainer.
    BottomToolStripPanel
//
this->trainingToolStripContainer->
    BottomToolStripPanel->Controls->Add(this->
    trainingStatusStrip);
//
// trainingToolStripContainer.ContentPanel
```

```
            //
            this->trainingToolStripContainer->ContentPanel->
                Controls->Add(this->trainingSplitContainer);
            this->trainingToolStripContainer->ContentPanel->
                Size = System::Drawing::Size(913, 514);
            this->trainingToolStripContainer->Dock = System::
                Windows::Forms::DockStyle::Fill;
            this->trainingToolStripContainer->
                LeftToolStripPanelVisible = false;
            this->trainingToolStripContainer->Location =
                System::Drawing::Point(0, 0);
            this->trainingToolStripContainer->Name = L"
                trainingToolStripContainer";
            this->trainingToolStripContainer->Size = System::
                Drawing::Size(913, 536);
            this->trainingToolStripContainer->TabIndex = 0;
            this->trainingToolStripContainer->Text = L"
                toolStripContainer1";
            this->trainingToolStripContainer->
                TopToolStripPanelVisible = false;
            //
            // trainingStatusStrip
            //
            this->trainingStatusStrip->Dock = System::Windows
                ::Forms::DockStyle::None;
            this->trainingStatusStrip->Items->AddRange(gcnew
                cli::array< System::Windows::Forms::
                ToolStripItem^ >(2) {
                    this->trainingToolStripStatusLabel,
                            this->
                                trainingToolStripProgressBar
            });
            this->trainingStatusStrip->Location = System::
                Drawing::Point(0, 0);
            this->trainingStatusStrip->Name = L"
                trainingStatusStrip";
            this->trainingStatusStrip->Size = System::Drawing
                ::Size(913, 22);
            this->trainingStatusStrip->TabIndex = 0;
            //
            // trainingToolStripStatusLabel
            //
            this->trainingToolStripStatusLabel->Name = L"
                trainingToolStripStatusLabel";
            this->trainingToolStripStatusLabel->Size = System
                ::Drawing::Size(0, 17);
            //
            // trainingToolStripProgressBar
            //
            this->trainingToolStripProgressBar->Alignment =
                System::Windows::Forms::ToolStripItemAlignment
                ::Right;
            this->trainingToolStripProgressBar->Name = L"
                trainingToolStripProgressBar";
            this->trainingToolStripProgressBar->Size = System
                ::Drawing::Size(720, 16);
            //
            // trainingSplitContainer
            //
```

```
this->trainingSplitContainer->Dock = System::
    Windows::Forms::DockStyle::Fill;
this->trainingSplitContainer->FixedPanel = System
    ::Windows::Forms::FixedPanel::Panel1;
this->trainingSplitContainer->IsSplitterFixed =
    true;
this->trainingSplitContainer->Location = System::
    Drawing::Point(0, 0);
this->trainingSplitContainer->Name = L"
    trainingSplitContainer";
//
// trainingSplitContainer.Panel1
//
this->trainingSplitContainer->Panel1->Controls->
    Add(this->trainingTableLayoutPanel);
//
// trainingSplitContainer.Panel2
//
this->trainingSplitContainer->Panel2->Controls->
    Add(this->traningResultsTableLayoutPanel);
this->trainingSplitContainer->Size = System::
    Drawing::Size(913, 514);
this->trainingSplitContainer->SplitterDistance =
    374;
this->trainingSplitContainer->TabIndex = 0;
//
// trainingTableLayoutPanel
//
this->trainingTableLayoutPanel->ColumnCount = 1;
this->trainingTableLayoutPanel->ColumnStyles->Add
    ((gcnew System::Windows::Forms::ColumnStyle())
    );
this->trainingTableLayoutPanel->Controls->Add(
    this->trainingMediaPanel, 0, 1);
this->trainingTableLayoutPanel->Controls->Add(
    this->trainingSetupPanel, 0, 0);
this->trainingTableLayoutPanel->Dock = System::
    Windows::Forms::DockStyle::Fill;
this->trainingTableLayoutPanel->Location = System
    ::Drawing::Point(0, 0);
this->trainingTableLayoutPanel->Name = L"
    trainingTableLayoutPanel";
this->trainingTableLayoutPanel->RowCount = 2;
this->trainingTableLayoutPanel->RowStyles->Add((
    gcnew System::Windows::Forms::RowStyle()));
this->trainingTableLayoutPanel->RowStyles->Add((
    gcnew System::Windows::Forms::RowStyle()));
this->trainingTableLayoutPanel->Size = System::
    Drawing::Size(374, 514);
this->trainingTableLayoutPanel->TabIndex = 0;
//
// trainingMediaPanel
//
this->trainingMediaPanel->Controls->Add(this->
    extractionFileListView);
this->trainingMediaPanel->Dock = System::Windows
    ::Forms::DockStyle::Fill;
this->trainingMediaPanel->Location = System::
    Drawing::Point(3, 109);
```

```
            this->trainingMediaPanel->Name = L"
                trainingMediaPanel";
            this->trainingMediaPanel->Size = System::Drawing
                ::Size(374, 402);
            this->trainingMediaPanel->TabIndex = 1;
            //
            // extractionFileListView
            //
            this->extractionFileListView->CheckBoxes = true;
            this->extractionFileListView->Columns->AddRange(
                gcnew cli::array< System::Windows::Forms::
                ColumnHeader^ >(1) { extractionColumnHeader
                });
            this->extractionFileListView->Dock = System::
                Windows::Forms::DockStyle::Fill;
            this->extractionFileListView->LargeImageList =
                this->csvImageList;
            this->extractionFileListView->Location = System::
                Drawing::Point(0, 0);
            this->extractionFileListView->Name = L"
                extractionFileListView";
            this->extractionFileListView->Size = System::
                Drawing::Size(374, 402);
            this->extractionFileListView->SmallImageList =
                this->csvImageList;
            this->extractionFileListView->TabIndex = 2;
            this->extractionFileListView->
                UseCompatibleStateImageBehavior = false;
            this->extractionFileListView->View = System::
                Windows::Forms::View::Details;
            this->extractionFileListView->
                SelectedIndexChanged += gcnew System::
                EventHandler(this, &TrainingScreen::
                trainingListView_SelectedIndexChanged);
            //
            // csvImageList
            //
            this->csvImageList->ImageStream = (cli::safe_cast
                <System::Windows::Forms::ImageListStreamer^>(
                resources->GetObject(L"csvImageList.
                ImageStream")));
            this->csvImageList->TransparentColor = System::
                Drawing::Color::Transparent;
            this->csvImageList->Images->SetKeyName(0, L"csv.
                ico");
            //
            // trainingSetupPanel
            //
            this->trainingSetupPanel->Controls->Add(this->
                deselectAllButton);
            this->trainingSetupPanel->Controls->Add(this->
                selectAllButton);
            this->trainingSetupPanel->Controls->Add(this->
                trainButton);
            this->trainingSetupPanel->Controls->Add(this->
                extractionMethodComboBox);
            this->trainingSetupPanel->Controls->Add(this->
                methodLabel);
            this->trainingSetupPanel->Controls->Add(this->
```

```
            movementTypeComboBox);
        this->trainingSetupPanel->Controls->Add(this->
            movementTypeLabel);
        this->trainingSetupPanel->Controls->Add(this->
            subjectIdComboBox);
        this->trainingSetupPanel->Controls->Add(this->
            subjectIdLabel);
        this->trainingSetupPanel->Location = System::
            Drawing::Point(3, 3);
        this->trainingSetupPanel->Name = L"
            trainingSetupPanel";
        this->trainingSetupPanel->Size = System::Drawing
            ::Size(371, 100);
        this->trainingSetupPanel->TabIndex = 1;
        //
        // deselectAllButton
        //
        this->deselectAllButton->Location = System::
            Drawing::Point(241, 70);
        this->deselectAllButton->Name = L"
            deselectAllButton";
        this->deselectAllButton->Size = System::Drawing::
            Size(75, 23);
        this->deselectAllButton->TabIndex = 8;
        this->deselectAllButton->Text = L"Deselect All";
        this->deselectAllButton->UseVisualStyleBackColor
            = true;
        this->deselectAllButton->Click += gcnew System::
            EventHandler(this, &TrainingScreen::
            deselectAllButton_Click);
        //
        // selectAllButton
        //
        this->selectAllButton->Location = System::Drawing
            ::Point(241, 40);
        this->selectAllButton->Name = L"selectAllButton";
        this->selectAllButton->Size = System::Drawing::
            Size(75, 23);
        this->selectAllButton->TabIndex = 7;
        this->selectAllButton->Text = L"Select All";
        this->selectAllButton->UseVisualStyleBackColor =
            true;
        this->selectAllButton->Click += gcnew System::
            EventHandler(this, &TrainingScreen::
            selectAllButton_Click);
        //
        // trainButton
        //
        this->trainButton->Location = System::Drawing::
            Point(241, 10);
        this->trainButton->Name = L"trainButton";
        this->trainButton->Size = System::Drawing::Size
            (75, 23);
        this->trainButton->TabIndex = 6;
        this->trainButton->Text = L"Train";
        this->trainButton->UseVisualStyleBackColor = true
            ;
        this->trainButton->Click += gcnew System::
            EventHandler(this, &TrainingScreen::
```

```
    trainButton_Click);
//
// extractionMethodComboBox
//
this->extractionMethodComboBox->FormattingEnabled
     = true;
this->extractionMethodComboBox->Items->AddRange(
    gcnew cli::array< System::Object^  >(2) { L"
    Haar Wavelet", L"ICA" });
this->extractionMethodComboBox->Location = System
    ::Drawing::Point(102, 74);
this->extractionMethodComboBox->Name = L"
    extractionMethodComboBox";
this->extractionMethodComboBox->Size = System::
    Drawing::Size(121, 21);
this->extractionMethodComboBox->TabIndex = 5;
this->extractionMethodComboBox->
    SelectedIndexChanged += gcnew System::
    EventHandler(this, &TrainingScreen::
    extractionMethodComboBox_SelectedIndexChanged)
    ;
//
// methodLabel
//
this->methodLabel->AutoSize = true;
this->methodLabel->Location = System::Drawing::
    Point(6, 77);
this->methodLabel->Name = L"methodLabel";
this->methodLabel->Size = System::Drawing::Size
    (96, 13);
this->methodLabel->TabIndex = 4;
this->methodLabel->Text = L"Extraction Method:";
//
// movementTypeComboBox
//
this->movementTypeComboBox->FormattingEnabled =
    true;
this->movementTypeComboBox->Items->AddRange(gcnew
     cli::array< System::Object^  >(2) { L"Left
    Arm", L"Right Leg" });
this->movementTypeComboBox->Location = System::
    Drawing::Point(102, 42);
this->movementTypeComboBox->Name = L"
    movementTypeComboBox";
this->movementTypeComboBox->Size = System::
    Drawing::Size(121, 21);
this->movementTypeComboBox->TabIndex = 3;
this->movementTypeComboBox->SelectedIndexChanged
    += gcnew System::EventHandler(this, &
    TrainingScreen::
    movementTypeComboBox_SelectedIndexChanged);
//
// movementTypeLabel
//
this->movementTypeLabel->AutoSize = true;
this->movementTypeLabel->Location = System::
    Drawing::Point(6, 45);
this->movementTypeLabel->Name = L"
    movementTypeLabel";
```

```
this->movementTypeLabel->Size = System::Drawing::
    Size(87, 13);
this->movementTypeLabel->TabIndex = 2;
this->movementTypeLabel->Text = L"Movement Type:"
    ;
//
// subjectIdComboBox
//
this->subjectIdComboBox->FormattingEnabled = true
    ;
this->subjectIdComboBox->Location = System::
    Drawing::Point(102, 8);
this->subjectIdComboBox->Name = L"
    subjectIdComboBox";
this->subjectIdComboBox->Size = System::Drawing::
    Size(121, 21);
this->subjectIdComboBox->TabIndex = 1;
this->subjectIdComboBox->SelectedIndexChanged +=
    gcnew System::EventHandler(this, &
    TrainingScreen::
    subjectIdComboBox_SelectedIndexChanged);
//
// subjectIdLabel
//
this->subjectIdLabel->AutoSize = true;
this->subjectIdLabel->Location = System::Drawing
    ::Point(6, 13);
this->subjectIdLabel->Name = L"subjectIdLabel";
this->subjectIdLabel->Size = System::Drawing::
    Size(58, 13);
this->subjectIdLabel->TabIndex = 0;
this->subjectIdLabel->Text = L"Subject Id:";
//
// traningResultsTableLayoutPanel
//
this->traningResultsTableLayoutPanel->AutoSize =
    true;
this->traningResultsTableLayoutPanel->ColumnCount
     = 1;
this->traningResultsTableLayoutPanel->
    ColumnStyles->Add((gcnew System::Windows::
    Forms::ColumnStyle(System::Windows::Forms::
    SizeType::Percent,
         100)));
this->traningResultsTableLayoutPanel->Controls->
    Add(this->meanGroupBox, 0, 0);
this->traningResultsTableLayoutPanel->Controls->
    Add(this->covarianceMatrixGroupBox, 0, 1);
this->traningResultsTableLayoutPanel->Controls->
    Add(this->trainingFileListView, 0, 2);
this->traningResultsTableLayoutPanel->Dock =
    System::Windows::Forms::DockStyle::Fill;
this->traningResultsTableLayoutPanel->Location =
    System::Drawing::Point(0, 0);
this->traningResultsTableLayoutPanel->Name = L"
    traningResultsTableLayoutPanel";
this->traningResultsTableLayoutPanel->RowCount =
    3;
this->traningResultsTableLayoutPanel->RowStyles->
```

```
        Add((gcnew System::Windows::Forms::RowStyle())
        );
    this->traningResultsTableLayoutPanel->RowStyles->
        Add((gcnew System::Windows::Forms::RowStyle())
        );
    this->traningResultsTableLayoutPanel->RowStyles->
        Add((gcnew System::Windows::Forms::RowStyle())
        );
    this->traningResultsTableLayoutPanel->Size =
        System::Drawing::Size(535, 514);
    this->traningResultsTableLayoutPanel->TabIndex =
        17;
    //
    // meanGroupBox
    //
    this->meanGroupBox->Controls->Add(this->
        meanDataGridView);
    this->meanGroupBox->FlatStyle = System::Windows::
        Forms::FlatStyle::System;
    this->meanGroupBox->Location = System::Drawing::
        Point(3, 3);
    this->meanGroupBox->Name = L"meanGroupBox";
    this->meanGroupBox->Size = System::Drawing::Size
        (528, 68);
    this->meanGroupBox->TabIndex = 15;
    this->meanGroupBox->TabStop = false;
    this->meanGroupBox->Text = L"Mean";
    //
    // meanDataGridView
    //
    this->meanDataGridView->AllowUserToAddRows =
        false;
    this->meanDataGridView->AllowUserToDeleteRows =
        false;
    this->meanDataGridView->AllowUserToResizeColumns
        = false;
    this->meanDataGridView->AllowUserToResizeRows =
        false;
    this->meanDataGridView->BackgroundColor = System
        ::Drawing::SystemColors::Window;
    this->meanDataGridView->BorderStyle = System::
        Windows::Forms::BorderStyle::Fixed3D;
    this->meanDataGridView->ClipboardCopyMode =
        System::Windows::Forms::
        DataGridViewClipboardCopyMode::
        EnableAlwaysIncludeHeaderText;
    dataGridViewCellStyle1->Alignment = System::
        Windows::Forms::DataGridViewContentAlignment::
        MiddleCenter;
    dataGridViewCellStyle1->BackColor = System::
        Drawing::SystemColors::Control;
    dataGridViewCellStyle1->Font = (gcnew System::
        Drawing::Font(L"Microsoft Sans Serif", 8.25F,
        System::Drawing::FontStyle::Regular,
             System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(0)));
    dataGridViewCellStyle1->ForeColor = System::
        Drawing::SystemColors::WindowText;
    dataGridViewCellStyle1->SelectionBackColor =
```

```
    System::Drawing::SystemColors::Control;
dataGridViewCellStyle1->SelectionForeColor =
    System::Drawing::SystemColors::WindowText;
dataGridViewCellStyle1->WrapMode = System::
    Windows::Forms::DataGridViewTriState::True;
this->meanDataGridView->
    ColumnHeadersDefaultCellStyle =
    dataGridViewCellStyle1;
this->meanDataGridView->
    ColumnHeadersHeightSizeMode = System::Windows
    ::Forms::
    DataGridViewColumnHeadersHeightSizeMode::
    DisableResizing;
this->meanDataGridView->Dock = System::Windows::
    Forms::DockStyle::Fill;
this->meanDataGridView->EditMode = System::
    Windows::Forms::DataGridViewEditMode::
    EditProgrammatically;
this->meanDataGridView->Location = System::
    Drawing::Point(3, 16);
this->meanDataGridView->MultiSelect = false;
this->meanDataGridView->Name = L"meanDataGridView
    ";
dataGridViewCellStyle2->Alignment = System::
    Windows::Forms::DataGridViewContentAlignment::
    MiddleCenter;
dataGridViewCellStyle2->BackColor = System::
    Drawing::SystemColors::Control;
dataGridViewCellStyle2->Font = (gcnew System::
    Drawing::Font(L"Microsoft Sans Serif", 8.25F,
    System::Drawing::FontStyle::Regular,
         System::Drawing::GraphicsUnit::Point,
            static_cast<System::Byte>(0)));
dataGridViewCellStyle2->ForeColor = System::
    Drawing::SystemColors::WindowText;
dataGridViewCellStyle2->SelectionBackColor =
    System::Drawing::SystemColors::Control;
dataGridViewCellStyle2->SelectionForeColor =
    System::Drawing::SystemColors::WindowText;
dataGridViewCellStyle2->WrapMode = System::
    Windows::Forms::DataGridViewTriState::True;
this->meanDataGridView->
    RowHeadersDefaultCellStyle =
    dataGridViewCellStyle2;
this->meanDataGridView->RowHeadersVisible = false
    ;
this->meanDataGridView->RowHeadersWidth = 69;
this->meanDataGridView->RowHeadersWidthSizeMode =
     System::Windows::Forms::
    DataGridViewRowHeadersWidthSizeMode::
    DisableResizing;
this->meanDataGridView->RowTemplate->
    DefaultCellStyle->Alignment = System::Windows
    ::Forms::DataGridViewContentAlignment::
    MiddleRight;
this->meanDataGridView->RowTemplate->
    DefaultCellStyle->Format = L"N6";
this->meanDataGridView->RowTemplate->
    DefaultCellStyle->NullValue = L"0.000000";
```

```
this->meanDataGridView->RowTemplate->
    DefaultCellStyle->SelectionBackColor = System
    ::Drawing::Color::White;
this->meanDataGridView->RowTemplate->
    DefaultCellStyle->SelectionForeColor = System
    ::Drawing::Color::Black;
this->meanDataGridView->RowTemplate->Height = 20;
this->meanDataGridView->RowTemplate->ReadOnly =
    true;
this->meanDataGridView->RowTemplate->Resizable =
    System::Windows::Forms::DataGridViewTriState::
    False;
this->meanDataGridView->SelectionMode = System::
    Windows::Forms::DataGridViewSelectionMode::
    FullRowSelect;
this->meanDataGridView->ShowCellToolTips = false;
this->meanDataGridView->Size = System::Drawing::
    Size(522, 49);
this->meanDataGridView->TabIndex = 0;
this->meanDataGridView->RowUnshared += gcnew
    System::Windows::Forms::
    DataGridViewRowEventHandler(this, &
    TrainingScreen::meanDataGridView_RowUnshared);
//
// covarianceMatrixGroupBox
//
this->covarianceMatrixGroupBox->Controls->Add(
    this->covarianceDataGridView);
this->covarianceMatrixGroupBox->Location = System
    ::Drawing::Point(3, 77);
this->covarianceMatrixGroupBox->Name = L"
    covarianceMatrixGroupBox";
this->covarianceMatrixGroupBox->Size = System::
    Drawing::Size(528, 210);
this->covarianceMatrixGroupBox->TabIndex = 16;
this->covarianceMatrixGroupBox->TabStop = false;
this->covarianceMatrixGroupBox->Text = L"
    Covariance Matrix";
//
// covarianceDataGridView
//
this->covarianceDataGridView->AllowUserToAddRows
    = false;
this->covarianceDataGridView->
    AllowUserToDeleteRows = false;
this->covarianceDataGridView->
    AllowUserToResizeColumns = false;
this->covarianceDataGridView->
    AllowUserToResizeRows = false;
dataGridViewCellStyle3->Alignment = System::
    Windows::Forms::DataGridViewContentAlignment::
    MiddleRight;
dataGridViewCellStyle3->BackColor = System::
    Drawing::Color::FromArgb(static_cast<System::
    Int32>(static_cast<System::Byte>(224)),
         static_cast<System::Int32>(static_cast<
             System::Byte>(224)), static_cast<
             System::Int32>(static_cast<System::
             Byte>(224)));
```

```
dataGridViewCellStyle3->SelectionBackColor =
    System::Drawing::Color::FromArgb(static_cast<
    System::Int32>(static_cast<System::Byte>(224))
    ,
          static_cast<System::Int32>(static_cast<
             System::Byte>(224)), static_cast<
             System::Int32>(static_cast<System::
             Byte>(224)));
dataGridViewCellStyle3->SelectionForeColor =
    System::Drawing::Color::Black;
this->covarianceDataGridView->
    AlternatingRowsDefaultCellStyle =
    dataGridViewCellStyle3;
this->covarianceDataGridView->BackgroundColor =
    System::Drawing::SystemColors::Window;
this->covarianceDataGridView->BorderStyle =
    System::Windows::Forms::BorderStyle::Fixed3D;
this->covarianceDataGridView->ClipboardCopyMode =
     System::Windows::Forms::
    DataGridViewClipboardCopyMode::
    EnableAlwaysIncludeHeaderText;
dataGridViewCellStyle4->Alignment = System::
    Windows::Forms::DataGridViewContentAlignment::
    MiddleCenter;
dataGridViewCellStyle4->BackColor = System::
    Drawing::SystemColors::Control;
dataGridViewCellStyle4->Font = (gcnew System::
    Drawing::Font(L"Microsoft Sans Serif", 8.25F,
    System::Drawing::FontStyle::Regular,
          System::Drawing::GraphicsUnit::Point,
             static_cast<System::Byte>(0)));
dataGridViewCellStyle4->ForeColor = System::
    Drawing::SystemColors::WindowText;
dataGridViewCellStyle4->SelectionBackColor =
    System::Drawing::SystemColors::Control;
dataGridViewCellStyle4->SelectionForeColor =
    System::Drawing::SystemColors::WindowText;
dataGridViewCellStyle4->WrapMode = System::
    Windows::Forms::DataGridViewTriState::True;
this->covarianceDataGridView->
    ColumnHeadersDefaultCellStyle =
    dataGridViewCellStyle4;
this->covarianceDataGridView->
    ColumnHeadersHeightSizeMode = System::Windows
    ::Forms::
    DataGridViewColumnHeadersHeightSizeMode::
    DisableResizing;
this->covarianceDataGridView->Dock = System::
    Windows::Forms::DockStyle::Fill;
this->covarianceDataGridView->EditMode = System::
    Windows::Forms::DataGridViewEditMode::
    EditProgrammatically;
this->covarianceDataGridView->Location = System::
    Drawing::Point(3, 16);
this->covarianceDataGridView->MultiSelect = false
    ;
this->covarianceDataGridView->Name = L"
    covarianceDataGridView";
this->covarianceDataGridView->ReadOnly = true;
```

```
dataGridViewCellStyle5->Alignment = System::
    Windows::Forms::DataGridViewContentAlignment::
    MiddleCenter;
dataGridViewCellStyle5->BackColor = System::
    Drawing::SystemColors::Control;
dataGridViewCellStyle5->Font = (gcnew System::
    Drawing::Font(L"Microsoft Sans Serif", 8.25F,
    System::Drawing::FontStyle::Regular,
         System::Drawing::GraphicsUnit::Point,
            static_cast<System::Byte>(0)));
dataGridViewCellStyle5->ForeColor = System::
    Drawing::SystemColors::WindowText;
dataGridViewCellStyle5->SelectionBackColor =
    System::Drawing::SystemColors::Control;
dataGridViewCellStyle5->SelectionForeColor =
    System::Drawing::SystemColors::WindowText;
dataGridViewCellStyle5->WrapMode = System::
    Windows::Forms::DataGridViewTriState::False;
this->covarianceDataGridView->
    RowHeadersDefaultCellStyle =
    dataGridViewCellStyle5;
this->covarianceDataGridView->RowHeadersVisible =
     false;
this->covarianceDataGridView->RowHeadersWidth =
    69;
this->covarianceDataGridView->
    RowHeadersWidthSizeMode = System::Windows::
    Forms::DataGridViewRowHeadersWidthSizeMode::
    DisableResizing;
this->covarianceDataGridView->RowTemplate->
    DefaultCellStyle->Alignment = System::Windows
    ::Forms::DataGridViewContentAlignment::
    MiddleRight;
this->covarianceDataGridView->RowTemplate->
    DefaultCellStyle->Format = L"N6";
this->covarianceDataGridView->RowTemplate->
    DefaultCellStyle->NullValue = L"0.000000";
this->covarianceDataGridView->RowTemplate->
    DefaultCellStyle->SelectionBackColor = System
    ::Drawing::Color::White;
this->covarianceDataGridView->RowTemplate->
    DefaultCellStyle->SelectionForeColor = System
    ::Drawing::Color::Black;
this->covarianceDataGridView->RowTemplate->Height
     = 20;
this->covarianceDataGridView->RowTemplate->
    ReadOnly = true;
this->covarianceDataGridView->RowTemplate->
    Resizable = System::Windows::Forms::
    DataGridViewTriState::False;
this->covarianceDataGridView->SelectionMode =
    System::Windows::Forms::
    DataGridViewSelectionMode::CellSelect;
this->covarianceDataGridView->ShowCellToolTips =
    false;
this->covarianceDataGridView->Size = System::
    Drawing::Size(522, 191);
this->covarianceDataGridView->TabIndex = 44;
this->covarianceDataGridView->RowUnshared +=
```

```
            gcnew System::Windows::Forms::
            DataGridViewRowEventHandler(this, &
            TrainingScreen::
            covarianceDataGridView_RowUnshared);
        //
        // trainingFileListView
        //
        this->trainingFileListView->Columns->AddRange(
            gcnew cli::array< System::Windows::Forms::
            ColumnHeader^  >(1) { this->
            trainingFilesColumnHeader });
        this->trainingFileListView->Dock = System::
            Windows::Forms::DockStyle::Fill;
        this->trainingFileListView->LargeImageList = this
            ->csvImageList;
        this->trainingFileListView->Location = System::
            Drawing::Point(3, 293);
        this->trainingFileListView->Name = L"
            trainingFileListView";
        this->trainingFileListView->Size = System::
            Drawing::Size(529, 218);
        this->trainingFileListView->SmallImageList = this
            ->csvImageList;
        this->trainingFileListView->TabIndex = 17;
        this->trainingFileListView->
            UseCompatibleStateImageBehavior = false;
        this->trainingFileListView->View = System::
            Windows::Forms::View::Details;
        this->trainingFileListView->MouseDoubleClick +=
            gcnew System::Windows::Forms::
            MouseEventHandler(this, &TrainingScreen::
            trainingFileListView_MouseDoubleClick);
        //
        // trainingFilesColumnHeader
        //
        this->trainingFilesColumnHeader->Text = L"
            Training File Name";
        this->trainingFilesColumnHeader->Width = 460;
        //
        // trainingBackgroundWorker
        //
        this->trainingBackgroundWorker->
            WorkerReportsProgress = true;
        this->trainingBackgroundWorker->
            WorkerSupportsCancellation = true;
        this->trainingBackgroundWorker->DoWork += gcnew
            System::ComponentModel::DoWorkEventHandler(
            this, &TrainingScreen::
            trainingBackgroundWorker_DoWork);
        this->trainingBackgroundWorker->
            RunWorkerCompleted += gcnew System::
            ComponentModel::RunWorkerCompletedEventHandler
            (this, &TrainingScreen::
            trainingBackgroundWorker_RunWorkerCompleted);
        //
        // trainingTimer
        //
        this->trainingTimer->Tick += gcnew System::
            EventHandler(this, &TrainingScreen::
```

```
    trainingTimer_Tick);
//
// TrainingScreen
//
this->AutoScaleDimensions = System::Drawing::
    SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::
    AutoScaleMode::Font;
this->ClientSize = System::Drawing::Size(913,
    536);
this->Controls->Add(this->
    trainingToolStripContainer);
this->FormBorderStyle = System::Windows::Forms::
    FormBorderStyle::FixedToolWindow;
this->Icon = (cli::safe_cast<System::Drawing::
    Icon^>(resources->GetObject(L"$this.Icon")));
this->MaximizeBox = false;
this->Name = L"TrainingScreen";
this->StartPosition = System::Windows::Forms::
    FormStartPosition::CenterScreen;
this->Text = L"Training";
this->trainingToolStripContainer->
    BottomToolStripPanel->ResumeLayout(false);
this->trainingToolStripContainer->
    BottomToolStripPanel->PerformLayout();
this->trainingToolStripContainer->ContentPanel->
    ResumeLayout(false);
this->trainingToolStripContainer->ResumeLayout(
    false);
this->trainingToolStripContainer->PerformLayout()
    ;
this->trainingStatusStrip->ResumeLayout(false);
this->trainingStatusStrip->PerformLayout();
this->trainingSplitContainer->Panel1->
    ResumeLayout(false);
this->trainingSplitContainer->Panel2->
    ResumeLayout(false);
this->trainingSplitContainer->Panel2->
    PerformLayout();
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->
    trainingSplitContainer))->EndInit();
this->trainingSplitContainer->ResumeLayout(false)
    ;
this->trainingTableLayoutPanel->ResumeLayout(
    false);
this->trainingMediaPanel->ResumeLayout(false);
this->trainingSetupPanel->ResumeLayout(false);
this->trainingSetupPanel->PerformLayout();
this->traningResultsTableLayoutPanel->
    ResumeLayout(false);
this->meanGroupBox->ResumeLayout(false);
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->meanDataGridView))
    ->EndInit();
this->covarianceMatrixGroupBox->ResumeLayout(
    false);
(cli::safe_cast<System::ComponentModel::
    ISupportInitialize^>(this->
```

```
                        covarianceDataGridView))->EndInit();
                    this->ResumeLayout(false);

                }
#pragma endregion


private: System::Void trainingListView_SelectedIndexChanged(System::
    Object^  sender, System::EventArgs^  e) {
}

                Void subjectIdFilter()
                {
                        List<String^>^ subjects = gcnew List<String^>();
                        extractionFileListView->Items->Clear();
                        String^ fileFilter = "";
                        fileFilter = fileFilter->Concat(fileFilter, "*
                            _extraction.csv");

                        for each (String^ fileName in Directory::
                            EnumerateFiles(extractFolderName, fileFilter,
                             IO::SearchOption::TopDirectoryOnly))
                        {
                                String^ shortFileName = fileName->
                                    Substring(fileName->LastIndexOf(Path
                                    ::DirectorySeparatorChar) + 1);
                                String^ subjectId = shortFileName->
                                    Substring(0, shortFileName->IndexOf("
                                    _"));
                                if (!subjects->Contains(subjectId))
                                {
                                        subjects->Add(subjectId);
                                        subjectIdComboBox->Items->Add(
                                            subjectId);
                                }
                        }

                }
                Void fileFilter(String^ subjectId, String^ method,
                    String^ movement)
                {
                        if (starting) return;
                        extractionFileListView->Items->Clear();
                        trainingFileListView->Items->Clear();
                        subjectId = subjectId->Concat(subjectId, "_");
                        String^ fileFilter = subjectId->Concat(subjectId
                            , movement->Replace(" ", "")->ToLower());
                        String^ extractionFileFilter;
                        String^ trainingFileFilter = "*_training.csv";
                        if (method->CompareTo("Haar Wavelet") == 0)
                        {
                                extractionFileFilter = fileFilter->
                                    Concat(fileFilter, "_*_wavelet.csv");
                                //trainingFileFilter = fileFilter->
                                    Concat(fileFilter, "_wavelet_training
                                    .csv");
                        }
                        else
                        {
```

```
                                extractionFileFilter = fileFilter->
                                    Concat(fileFilter, "_*_ica.csv");
                                //trainingFileFilter = fileFilter->
                                    Concat(fileFilter, "_ica_training.csv
                                    ");
                            }
                            for each (String^ fileName in Directory::
                                EnumerateFiles(extractFolderName,
                                extractionFileFilter, IO::SearchOption::
                                TopDirectoryOnly))
                            {
                                String^ shortFileName = fileName->
                                    Substring(fileName->LastIndexOf(Path
                                    ::DirectorySeparatorChar) + 1);
                                extractionFileListView->Items->Add(
                                    shortFileName)->ImageIndex = 0;
                            }
                            for each (String^ fileName in Directory::
                                EnumerateFiles(trainingFolderName,
                                trainingFileFilter, IO::SearchOption::
                                TopDirectoryOnly))
                            {
                                String^ shortFileName = fileName->
                                    Substring(fileName->LastIndexOf(Path
                                    ::DirectorySeparatorChar) + 1);
                                trainingFileListView->Items->Add(
                                    shortFileName)->ImageIndex = 0;
                            }

                        }

                        Void trainingFileFilter(String^ subjectId, String^
                            method, String^ movement)
                        {
                                if (starting) return;
                                trainingFileListView->Items->Clear();

                                String^ trainingFileFilter = "*_training.csv";

                                for each (String^ fileName in Directory::
                                    EnumerateFiles(trainingFolderName,
                                    trainingFileFilter, IO::SearchOption::
                                    TopDirectoryOnly))
                                {
                                    String^ shortFileName = fileName->
                                        Substring(fileName->LastIndexOf(Path
                                        ::DirectorySeparatorChar) + 1);
                                    trainingFileListView->Items->Add(
                                        shortFileName)->ImageIndex = 0;
                                }

                        }
private: System::Void extractionMethodComboBox_SelectedIndexChanged(
    System::Object^  sender, System::EventArgs^  e) {
                                fileFilter((String^)subjectIdComboBox->
                                    SelectedItem,
                                            (String^)extractionMethodComboBox
                                                ->SelectedItem,
                                                    (String^)
```

```
                                                    movementTypeComboBox->
                                                    SelectedItem);
}
private: System::Void subjectIdComboBox_SelectedIndexChanged(System::
    Object^  sender, System::EventArgs^  e) {
                         if (starting) starting = false;
                         fileFilter((String^)subjectIdComboBox->
                             SelectedItem,
                                  (String^)extractionMethodComboBox->
                                      SelectedItem,
                                  (String^)movementTypeComboBox->
                                      SelectedItem);
}
private: System::Void movementTypeComboBox_SelectedIndexChanged(System::
    Object^  sender, System::EventArgs^  e) {
                         fileFilter((String^)subjectIdComboBox->
                             SelectedItem,
                                  (String^)extractionMethodComboBox->
                                      SelectedItem,
                                  (String^)movementTypeComboBox->
                                      SelectedItem);
}
private: System::Void selectAllButton_Click(System::Object^  sender,
    System::EventArgs^  e) {

                         for (System::Collections::IEnumerator^ it =
                             extractionFileListView->Items->GetEnumerator
                             (); it->MoveNext();){
                                  ListViewItem^ lvi = (ListViewItem^) it->
                                      Current;
                                  lvi->Checked = true;
                         }
}
private: System::Void deselectAllButton_Click(System::Object^  sender,
    System::EventArgs^  e) {
                         for (System::Collections::IEnumerator^ it =
                             extractionFileListView->Items->GetEnumerator
                             (); it->MoveNext();){
                                  ListViewItem^ lvi = (ListViewItem^)it->
                                      Current;
                                  lvi->Checked = false;
                         }
}
private: System::Void trainButton_Click(System::Object^  sender, System::
    EventArgs^  e) {
                         List<String^>^ fileList = gcnew List<String^>();
                         for (System::Collections::IEnumerator^ it =
                             extractionFileListView->Items->GetEnumerator
                             (); it->MoveNext();){
                                  ListViewItem^ lvi = (ListViewItem^)it->
                                      Current;
                                  if (lvi->Checked)
                                  {
                                          fileList->Add(lvi->Text);
                                  }
                         }
                         if (training != nullptr) delete training;
                         training = gcnew Training(extractFolderName,
                             fileList);
```

```
                        trainingTimer->Start();

}
private: System::Void trainingBackgroundWorker_DoWork(System::Object^
    sender, System::ComponentModel::DoWorkEventArgs^  e) {
                        if (training != nullptr)
                        {
                                training->train();
                        }
}
                System::Void setTrainingResults()
                {

                        array<double>^ means = training->getMeans();
                        for (int i = 0; i < means->Length; i++)
                        {

                                meanDataGridView[i+1, 0]->Value = means[
                                    i];
                        }
                        array<double,2>^ covariance = training->
                            getCovarianceMatrix();
                        for (int i = 0; i < 8; i++)
                        {
                                for (int j = 0; j < 8; j++)
                                {

                                        covarianceDataGridView[j+1, i]->
                                            Value = covariance[i, j];
                                }

                        }

                        trainingToolStripStatusLabel->Text = "Done";
                        trainingToolStripProgressBar->Step = 100;
                        trainingToolStripProgressBar->PerformStep();

                }
private: System::Void trainingBackgroundWorker_RunWorkerCompleted(System
    ::Object^  sender, System::ComponentModel::RunWorkerCompletedEventArgs
    ^  e) {
                        trainingTimer->Stop();
                        setTrainingResults();
                        trainingFileFilter((String^)subjectIdComboBox->
                            SelectedItem,
                                (String^)extractionMethodComboBox->
                                    SelectedItem,
                                (String^)movementTypeComboBox->
                                    SelectedItem);

}
private: System::Void trainingTimer_Tick(System::Object^  sender, System
    ::EventArgs^  e) {
                        if (!trainingBackgroundWorker->IsBusy){
                                trainingTimer->Stop();
                                trainingToolStripStatusLabel->Text = "
                                    Calculating...";
                                trainingToolStripProgressBar->Step = 1;
                                trainingToolStripProgressBar->Visible =
```

```
                                        true;
                                trainingBackgroundWorker->RunWorkerAsync
                                    ();
                                trainingTimer->Start();
                        }
                        else{

                                if (trainingToolStripProgressBar->Step <
                                     100)
                                {
                                        trainingToolStripProgressBar->
                                            Step += 5;
                                }
                                else
                                {
                                        trainingToolStripProgressBar->
                                            Step = 50;
                                }
                                trainingToolStripProgressBar->
                                    PerformStep();
                        }
}

                public: static void DoubleBuffered(DataGridView^ dgv,
                    bool setting)
                {
                        Type^ dgvType = dgv->GetType();
                        PropertyInfo^ pi = dgvType->GetProperty("
                            DoubleBuffered",
                                BindingFlags::Instance | BindingFlags::
                                    NonPublic);
                        pi->SetValue(dgv, setting, nullptr);
                }

private: System::Void trainingFileListView_MouseDoubleClick(System::
    Object^  sender, System::Windows::Forms::MouseEventArgs^  e) {
                        if (training != nullptr){
                                delete[] training;
                        }
                        Collections::IEnumerator^ it =
                            trainingFileListView->SelectedItems->
                            GetEnumerator();
                        if (it->MoveNext())
                        {
                                ListViewItem^ currFile = (ListViewItem^)
                                    it->Current;
                                trainingFileName = currFile->Text;
                        }
                        if (trainingFileName->CompareTo("") != 0)
                        {

                                trainingToolStripProgressBar->Visible =
                                    true;
                                trainingToolStripStatusLabel->Text = "
                                    Loading...";
                                trainingToolStripProgressBar->Step = 50;
                                trainingToolStripProgressBar->
                                    PerformStep();
                                training = gcnew Training(
```

```
                                        trainingFolderName, trainingFileName)
                                        ;
                                    training->view();
                                    setTrainingResults();
                                }

}
private: System::Void covarianceDataGridView_RowUnshared(System::Object^
    sender, System::Windows::Forms::DataGridViewRowEventArgs^  e) {
                                bool unshared = true;
}
private: System::Void meanDataGridView_RowUnshared(System::Object^
   sender, System::Windows::Forms::DataGridViewRowEventArgs^  e) {
                                bool unshared = true;
}


};
}
```