Universidad de Valladolid

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO DE FIN DE GRADO
GRADO EN INGENIERÍA EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN

Diseño de una aplicación de audio controlada con señales mioeléctricas

Septiembre 2017

Autora: Andrea Marín Brezmes
Tutor: Ramón de la Rosa Steinz

Resumen

El presente trabajo de fin de grado se engloba dentro de las tecnologías de rehabilitación. El objetivo del mismo es añadir una nueva funcionalidad a la plataforma UVa-NTS de adquisición y entrenamiento en tiempo real de señales neuromusculares. Se añade esta funcionalidad con el propósito final de ser utilizada en el entrenamiento en el manejo de prótesis mioeléctricas.
Se introducen unos conceptos básicos de fisiología que ayudan al lector a comprender el proceso de generación de señales mioeléctricas. A continuación, se explica la parte software de la plataforma citada anteriormente, el programa *Miocon*, al cual se le añade una herramienta nueva, *MioSong*. Esta herramienta produce tonos audibles de una frecuencia directamente proporcional al esfuerzo muscular realizado.
Tras explicar el proceso de creación de la herramienta citada, se recogen los datos y conclusiones de una pequeña prueba realizada para comprobar el funcionamiento y utilidad de la misma.
Finalmente, antes de concluir y comentar las posibles líneas futuras, se añade un capítulo dedicado a otros trabajos realizados en el ámbito de la bioingeniería.

Palabras clave: rehabilitación, sistema en tiempo real, señal mioeléctrica, tono audible.

Abstract

The present work is included in the field of rehabilitation technologies. The target is to add a new functionality to the UVa-NTS platform for real-time acquisition and training of neuromuscular signals. This functionality is added with the final purpose of being used in training in the management of myoelectric prostheses.
Basic concepts of physiology are introduced for help the reader to understand the process of myoelectric signals generation. The software part of the platform mentioned above, the Miocon program, is explained below, to which a new tool, MioSong, is added. This tool produces audible tones of a frequency directly proportional to the muscular effort made.
After explaining the process of creation of the cited tool, it is collected the data and conclusions of a small test performed to verify the operation and utility of the tool. Finally, before concluding and discuss on possible future lines, it is added a chapter dedicated to other works carried out in the field of bioengineering.

Keywords: rehabilitation, real-time system, myoelectric signal, audible tone.

Índice general

Capítulo 1

Introducción

1.1. Objetivo

El trabajo presentado a continuación se engloba dentro del ámbito de las tecnologías de rehabilitación. Con el mismo, se busca añadir una nueva funcionalidad a la plataforma UVa-NTS de adquisición y entrenamiento en tiempo real de señales neuromusculares. Esta plataforma ha sido desarrollada en diferentes etapas, por diversos colaboradores y proyectandos que han pasado por el laboratorio de Electrónica y Bioingeniería de la E.T.S de Ingenieros de Telecomunicación de la Universidad de Valladolid. La novedad que se pretende introducir, consiste en la creación de una nueva herramienta que sea capaz de generar una serie de sonidos cuya frecuencia esté relacionada con los valores de las muestras que llegan de la cabecera de adquisición de señales mioeléctricas(SME). Tanto la cabecera de adquisición como el programa a modificar, forman parte de UVa-NTS, una plataforma de adquisición en tiempo real de SME desarrollada por colaboradores del laboratorio citado anteriormente.
El objetivo final de esta propuesta consiste en abrir el abanico de opciones que presenta la herramienta de la que se parte, para facilitar y motivar, en la medida de lo posible, a los pacientes que precisan un entrenamiento específico para el aprendizaje del uso de una prótesis.
Pese a que lo comentado anteriormente se mantiene como objetivo final y fundamental del presente trabajo, se han llevado a cabo actividades complementarias que, englobadas también en el ámbito de la bioingeniería, se recogen a modo de complemento en el presente documento. Una de estas actividades ha consistido en el diseño de un protocolo de pruebas, así como la realización de las mismas, para el sistema TactileCom Belt, desarrollado también por colaboradores del laboratorio de Electrónica y Bioingeniería de la E.T.S.I.T.

1.2. Organización del documento

Esta memoria ha quedado organizada en ocho capítulos y un apéndice.
En primer lugar, en el primer capítulo, en el que se incluye esta sección, se realiza

una introducción general del trabajo desarrollado, dejando constancia de los objetivos buscados, así como una breve explicación de los elementos que se han utilizado en el proceso de alcanzar dichos objetivos.
En el siguiente capítulo, se explican brevemente unos conceptos básicos de fisiología, que permitirán al lector una mayor comprensión de las señales capturadas por el programa *Miocon*.
A continuación, en el capítulo tercero, se hace una explicación de la manera en que llegan las muestras al programa, así como una aclaración a nivel funcional de cana una de las herramientas del mismo, incluyendo en esta explicación la nueva herramienta introducida.
En el capítulo cuarto, se hace una exposición, paso a paso, del proceso de desarrollo de la herramienta *MioSong*. Se trata del capítulo central de esta memoria y recoge, en el apartado dedicado al desarrollo del código, un diagrama de flujo que detalla el funcionamiento de la nueva herramienta implementada. Además, al final de este capítulo, se comenta uno de los problemas que surgieron en el desarrollo de la herramienta.
El capítulo quinto contiene los datos y conclusiones de una serie de pruebas que se realizaron para probar la herramienta *MioSong*.
A continuación, en el sexto capítulo, se realiza un repaso del trabajo realizado de manera paralela al presente proyecto. Incluye el proceso de diseño y aplicación de un protocolo de actuación para la realización de pruebas con un prototipo, el cinturón TactileCom, para lo que previamente se explica el funcionamiento básico del prototipo. Este trabajo se incluyó en un artículo [17] cuyas conclusiones, tomadas a partir de los datos conseguidos, se recogen también al final de este capítulo.
En los dos últimos capítulos, séptimo y octavo, se exponen las líneas futuras en las que se pretende continuar el desarrollo del trabajo y las conclusiones globales que se han sacado tras la realización del proyecto.
Por último, en el apéndice, se recogen las instrucciones detalladas del sistema utilizado para realizar las pruebas cuyo protocolo se ha diseñado y el cual se recoge en el capitulo sexto.

1.3. Elementos utilizados

Para el desarrollo del presente trabajo de fin de grado se ha utilizado un sistema que consta de diversos elementos explicados a continuación.

1.3.1. Electrodos

Para hacer llegar la señal al sistema es necesario utilizar 5 electrodos de plata/cloruro de plata que son capaces de captar la señal eléctrica generada por la actividad muscular (señal mioeléctrica). Dos de estos electrodos se colocan en un músculo, generalmente bíceps o tríceps y otros dos en su antagonista, tríceps o bíceps, respectivamente. El quinto electrodo se coloca en una zona de referencia, que se ha de encontrar lo más alejada posible de alguno de los músculos, generalmente en torno al codo.

Figura 1.1: Electrodos utilizados para capturar la señal mioeléctrica.

Los electrodos usados se muestran en la Figura 1.1. y se conectan al sistema de adquisición mediante el cable conector que aparece en la Figura 1.2. Este cable posee, en un extremo, los 5 conectores necesarios para conectar los electrodos y, en el extremo opuesto, un conector din macho de 5 pines, que se conecta con la cabecera de adquisición.

1.3.2. Cabecera de adquisición de señales mioeléctricas

La cabecera de adquisición de señales mioeléctricas es el elemento clave de la plataforma UVa-NTS [1]; se encarga del procesado de la señal analógica y es el traductor digital de la misma. Su apariencia se muestra en la Figura 1.3.

1.3.3. Ordenador y software

Para el desarrollo de este trabajo, se ha empleado un ordenador portátil ASUS F556U con un procesador Intel Core i7-6500U y 12GB de RAM. Con un sistema operativo Windows10, ha sido necesario instalar un Windows XP en una máquina virtual(VMware Workstation 12 Player) para poder desarrollar el programa en el entorno de desarrollo Microsoft Visual C++ 6.0.
El programa está escrito en lenguaje C++, a excepción de alguna de las bibliotecas. Estas bibliotecas han sido programadas en C, y se trata de traducciones de funciones de MATLAB, que son aplicadas en el procesado de la señal.

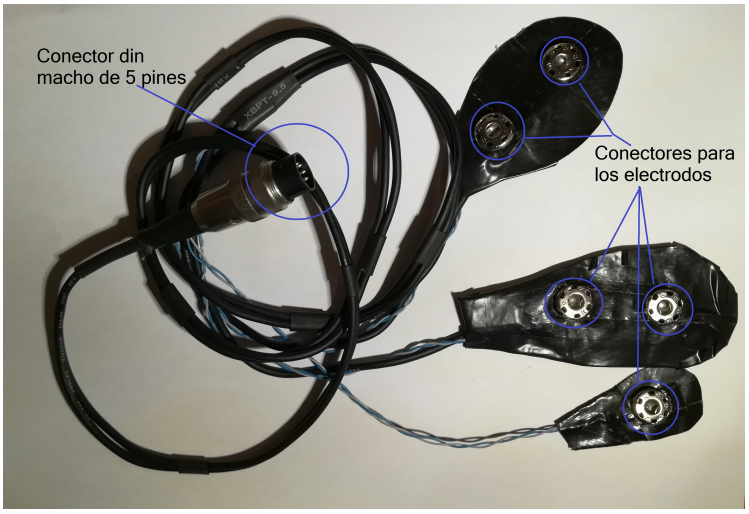


Figura 1.2: Cableado y conectores que portan la señal de los electrodos al sistema de adquisición.

Figura 1.3: Cabecera de adquisición de señales mioeléctricas.

Capítulo 2

Fundamentos fisiológicos

Con el objetivo de alcanzar una mayor comprensión de los procesos subyacentes a la generación de señales mioeléctricas, objeto de estudio en este trabajo de fin de grado, se introducen a continuación unos conceptos básicos de fisiología que, baasados en las referencias [3] y [2], permitirán mantener una visión más global del mismo.

2.1. Sistema nervioso

Se define el sistema nervioso como el conjunto de neuronas, ganglios y centros nerviosos que constituyen los órganos responsables de integrar y coordinar la actividad orgánica del organismo. El sistema nervioso y el endocrino son los encargados de conseguir que los distintos elementos de un organismo trabajen al unísono y varíen su ritmo en función de las circunstancias externas o internas del individuo.
Desde el punto de vista anatómico, el sistema nervioso puede dividirse en sistema cerebroespinal o central y sistema nervioso vegetativo o autónomo. El sistema nervioso central, situado en el cráneo y la cavidad raquídea, se encuentra representado en la Figura 2.1, y está formado por el encéfalo(cerebro, bulbo raquídeo y cerebelo), la médula espinal y los nervios que surgen de la médula. El sistema nervioso vegetativo está formado por todo el entramado de células nerviosas y nervios que regulan las actividades del aparato digestivo, el circulatorio, la musculatura lisa, las glándulas de secreción interna y externa, el metabolismo, y el aparato urogenital, es decir, los procesos orgánicos denominados vegetativos.

2.1.1. Tejido nervioso

El tejido nervioso es un tejido de origen ectodérmico(que proviene de una de las tres capas germinales del embrión [4]), altamente especializado, que constituye el principal componente del sistema nervioso. Las neuronas son sus unidades funcionales, pero junto a ellas también se encuentran las células de la glía. Se puede considerar que el tejido nervioso es el más especializado de todos los que forman los organismos, aunque ha perdido su capacidad de reproducción.

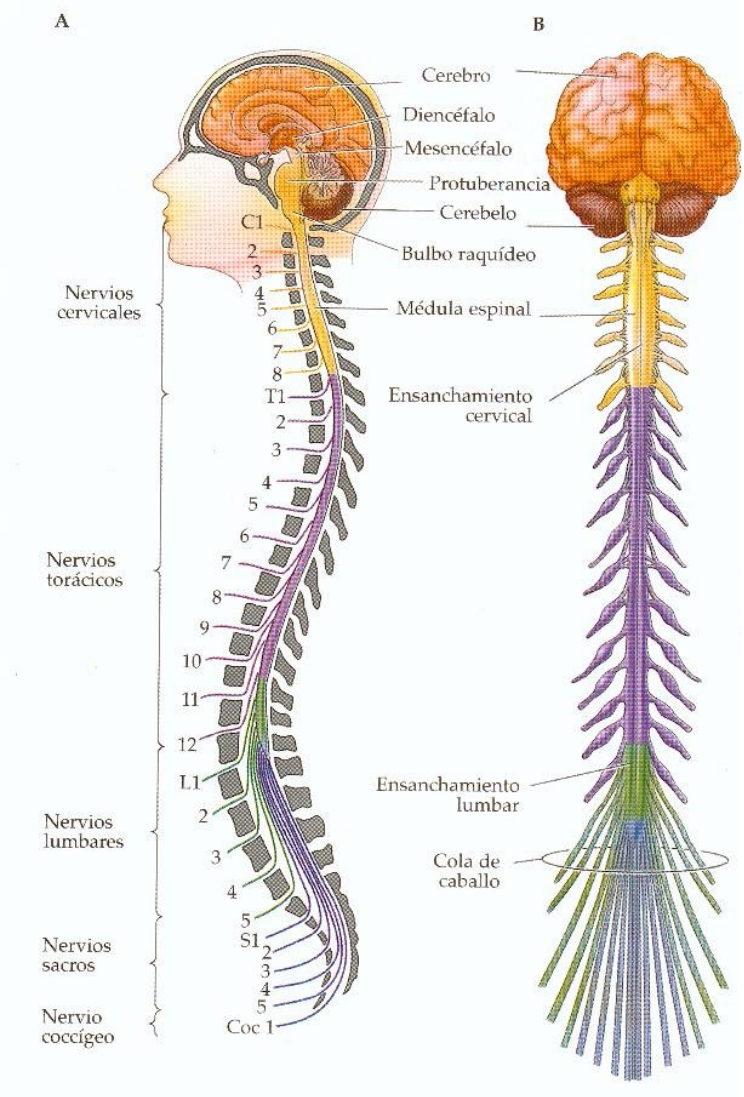


Figura 2.1: Esquema del SNC humano(Imagen tomada de [5])

.

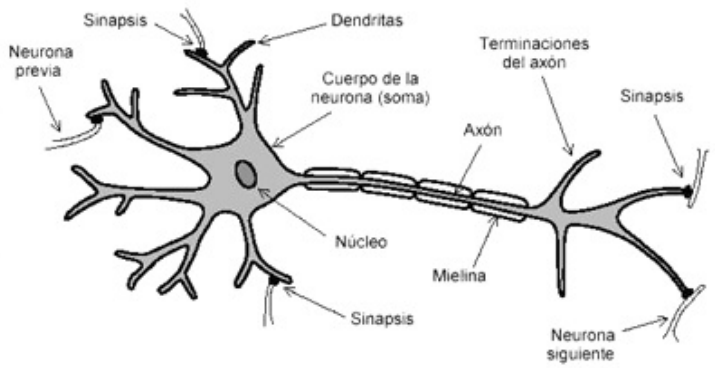


Figura 2.2: Esquema de una neurona.

2.1.2. Células nerviosas

Dentro del conjunto de células consideradas como células nerviosas, podemos distinguir dos tipos: neuronas y células de la glía.

Neuronas

En una neurona típica, como la representada en la Figura 2.1, pueden identificarse morfológicamente cuatro regiones:

- El cuerpo celular o soma.
- Las dendritas.
- El axón.
- Los terminales axónicos o sinápticos.

La función de las neuronas es la generación de señales eléctricas, actividad en la cual cada una de las partes citadas realiza una función determinada. El *cuerpo celular* constituye el centro metabólico de la neurona, las *dendritas* son arborizaciones del cuerpo celular que desempeñan el papel de principal zona receptora para la neurona, el *axón* actúa como la unidad conductiva de la neurona y los *terminales axónicos o sinápticos* constituyen los elementos de transmisión de la neurona. A través de los *terminales axónicos*, una neurona contacta y transmite información a la zona receptiva de otra neurona, o de una célula efectora (célula que actúa inmediatamente en la respuesta inmune [4]). La zona de contacto recibe el nombre de *sinapsis*.

Células de la glía

El tipo de células más abundante en el sistema nervioso central(SNC) está constituido por células de la glía. Carecen de la capacidad de generar activamente señales

eléctricas, sin embargo, ejercen las siguientes funciones:

- *Función de soporte*, semejante al papel del tejido conectivo en otros órganos.
- *Función de eliminación* de productos de deshecho del metabolismo neuronal, o de restos celulares tras la muerte celular.
- *Función de tampón* espacial de k^+ y de captación de neurotransmisores.
- *Función de guía* para la migración neuronal durante el desarrollo.
- *Función de nutrición* neuronal.

2.2. Potenciales de reposo y de acción

El potencial de reposo resulta, como en toda célula del organismo, de la separación de cargas eléctricas a través de la membrana celular semipermeable.
Un potencial de acción es un cambio rápido en el potencial de membrana, que se propaga a lo largo de toda la longitud de la célula. Los potenciales de acción constituyen la base para la mayor parte de las comunicaciones entre las neuronas, provocan la contracción de las células del músculo esquelético y permiten su aparición, casi sincrónica, a lo largo de todo el tejido celular. Al valor de potencial de membrana en el cual se genera el potencial de acción se le llama ümbral".
El potencial de acción es un cambio rápido en el potencial de membrana seguido de un retorno al potencial de reposo. Tal y como se muestra en la figura 2.3, la amplitud y la forma de los potenciales de acción cambian considerablemente de un tejido excitable a otro, pero se propagan con la misma forma y tamaño a lo largo de toda la célula muscular o nerviosa. Las proteínas de los canales iónicos dependientes de voltaje en la membrana plasmática, son las responsables de la morfología del potencial de acción. En la figura 2.3 aparecen diferentes potenciales de acción en los distintos tipos celulares; esto se debe a que dichas células tienen poblaciones diferentes de canales iónicos dependientes del voltaje.
Si a través de la membrana plasmática de una célula fluye un pulso de corriente, el potencial de membrana cambia. Los pulsos de corriente son despolarizantes o hiperpolarizantes, dependiendo de la dirección del flujo de la corriente. Los términos despolarizante e hiperpolarizante pueden ser confusos. Un cambio en el potencial de membrana de -90mV a -70mV es una despolarización, ya que supone un descenso de la diferencia de potencial, o de la polarización, a través de la membrana. Si el potencial de membrana cambia de -90mV a -100mV, la polarización de la membrana ha aumentado; esto es una hiperpolarización. Cuanto mayor sea la corriente aplicada, mayor será la modificación del potencial de membrana.
Cuando se aplican pulsos de corriente subumbral, la dimensión del cambio de potencial observado depende de la distancia del electrodo de registro al punto desde el que pasa la corriente. Cuanto más próximo esté el electrodo al lugar donde se aplica la corriente, mayor será el cambio observado en el potencial. Las dimensiones del cambio de potencial descienden exponencialmente con la distancia desde el punto de aplicación de la corriente. Se dice que la respuesta se conduce en declive.

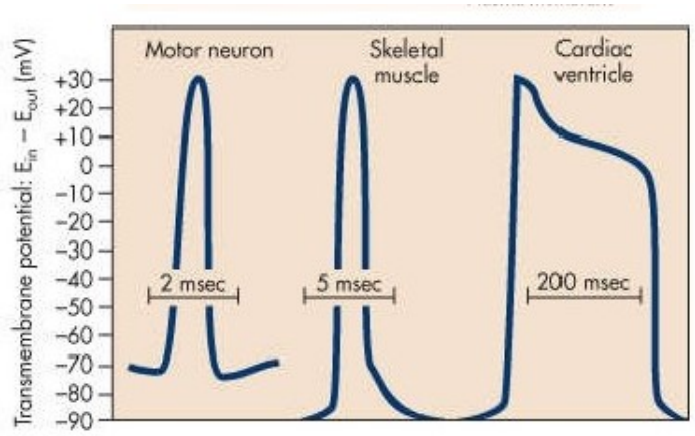


Figura 2.3: Potenciales de acción de tres tipos celulares de de los vertebrados(Imagen tomada de [6]).

Los electrodos registran la señal de voltaje compuesta por multitud de potenciales de acción y llegan al sistema de adquisición. Son estas pequeñas señales las que, tras un amplificado en en la cabecera de adquisición, aportan la información necesaria para el funcionamiento de las herramientas desarrolladas, que interactuarán en tiempo real con los datos capturados.

Capítulo 3

Explicación del proyecto Miocon

La aplicación Miocon surge con la intención de ser utilizada en el ámbito de la rehabilitación, como instrumento para el entrenamiento en el uso de prótesis. Con esta idea, busca ser una aplicación escalable y que presente robustez en tiempo real. El programa basa su estructura en tres módulos diferenciados, englobados en el entorno de desarrollo Visual C++ 6.0 como tres proyectos aislados, que ofrecen servicio mediante la interactuación entre ellos. Esta interactuación entre los bloques se ve reflejada en la Figura 3.1.

Miocon.exe constituye el bloque principal y en él se lleva a cabo el procesado de alto nivel de los datos, siendo el encargado de generar el entorno de ventanas y de englobar los diferentes instrumentos aplicables al entrenamiento neuromuscular. Como nodo intermedio con la cabecera, el módulo principal utiliza la biblioteca de funciones *Conversor.dll*, que es el resultado de compilar el proyecto *Conversor* y se encarga de realizar el procesado a bajo nivel, quedando el bloque principal aislado de estos detalles, lo que permite a la aplicación trabajar con diferentes cabeceras de adquisición. La aplicación principal se apoya a su vez en el módulo *ProcesMat*,que da lugar a la biblioteca *ProcesMat.dll*, una biblioteca desarrollada en C que supone la traducción a este lenguaje de las funciones generadas en lenguaje Matlab, a diferencia de *Conversor.dll* que se desarrolla en C++.

El módulo *ProcesMat.dll* es el que hace posible un procesado de la señal en tiempo real, ya que aprovecha la potencia de procesado de las funciones implementadas en *Matlab*. Esta estructura de bloques, además de aportar robustez y escalabilidad, supone una gran ventaja a la hora de aislar los posibles fallos que se pueden dar en

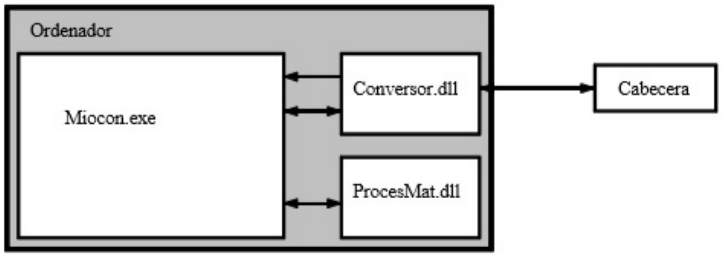


Figura 3.1: Diagrama de bloques de la aplicación Miocon(Imagen tomada de [1]).

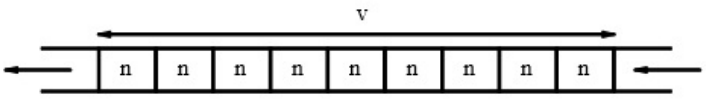


Figura 3.2: Organización de la pila de tamaño v muestras dividida en bloques de n (Imagen tomada de [1]).

el desarrollo de la misma.

3.1. Recepción de las muestras

El sistema de adquisición, incluido en el módulo hardware de la cabecera, amplifica y acondiciona la señal analógica. Ésta pasa por un conversor analógico-digital y, a través de un bus serie, la señal digitalizada es entregada al ordenador y procesada por el módulo *Conversor.dll*. Es en el conversor, donde estos valores se cuantifican y agrupan en bloques de un tamaño fijo de n muestras, que son pasadas a la aplicación principal. Este tamaño n se fija teniendo en cuenta que el sistema debe proporcionar una respuesta en tiempo real y basándonos en las investigaciones de K. Englehart , B. Hudgins y P.A. Parker, se considera aceptable en el desarrollo de prótesis mioeléctricas un tiempo no superior a 300ms[8].
Se ha de tener en cuenta en la recepción tanto el periodo de adquisición de cada bloque de n muestras, que tendrá un valor t_n=100ms, como el tamaño de la ventana de suavizado, ya que $t_v \geq t_n$.
El sistema lee un bloque cada 100ms, por lo que el número de muestras por canal será n=$t_n \cdot f_s$, donde f_s es la frecuencia de muestreo por canal, que según viene recogido en[1], tiene un valor de 1010Hz, por lo que el sistema recoge n=101 muestras/lectura.
Posteriormente, en el conversor, los bloques de n muestras se agrupan en un pila FIFO (*First In – First Out*, representada en la Figura 3.2 , que lee en orden de llegada y elimina los bloques una vez utilizados, dejando espacio para los nuevos bloques). La pila tiene un tamaño v y dado que se trabaja con ventanas de 1s, v=1010 muestras por canal , ya que f_s=1010Hz y v=$t_v \cdot f_s$. Es sobre esta ventana sobre la que se aplica el extractor de características.
En caso de tener varios canales, las muestras se organizan en una multitrama, como la que aparece en la Figura 3.3 , de tamaño N=n $\cdot$ c, donde n es el número de muestras por canal y c el número de canales.
Cuando se trabaja con dos canales activos, con un tamaño de ventana de v=1010 muestras, la aplicación toma 2020 muestras, 1010 para cada canal, que se organizan como se muestra en la Figura 3.3.
Esta forma de tratar las muestras presenta la ventaja extra de que al almacenarlas en el disco todos los canales se concentran en un mismo archivo.

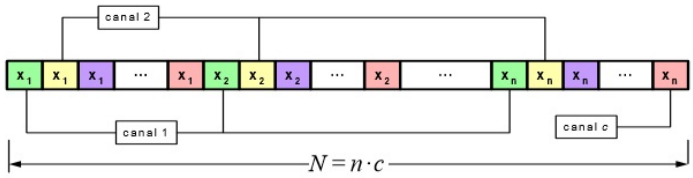


Figura 3.3: Organización de las muestras correspondientes a cada uno de los canales dentro de una multitrama (Imagen tomada de [7]).

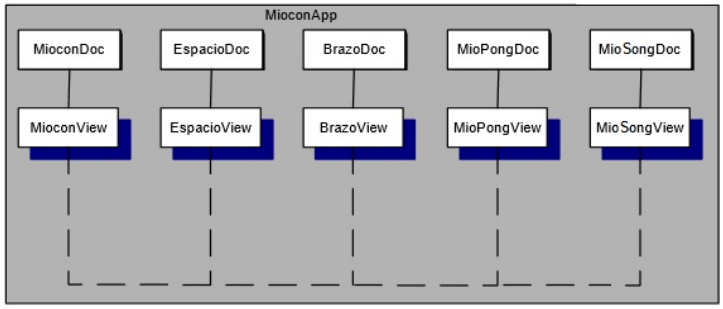


Figura 3.4: Esquema de documentos y vistas asociadas a la aplicación *Miocon.exe*.

3.2. Aplicación principal: Miocon.exe

Dado que en el desarrollo de este trabajo de fin de grado únicamente se han realizado cambios en este bloque, se hará de forma más exhaustiva la explicación del mismo. La aplicación principal es la encargada de realizar el procesado de los datos que le llegan a través del bloque *Conversor*, haciendo uso de la biblioteca que le ofrece el bloque *ProcesMat* para, finalmente, presentarlo en un entorno de ventanas manejable por el usuario.
Toda la aplicación se desarrolla bajo la filosofía vista/documento utilizada en las aplicaciones Microsoft; además, se trata de una aplicación de tipo MDI(*Multiple Document Interface*), por lo que para cada herramienta introducida se crean dos clases, la clase vista y la clase documento, relacionadas entre sí y heredadas de la clase *CView* y *CDocument* respectivamente. Este modelo de desarrollo, vista/documento, se basa en la concepción de que el programa que vamos a desarrollar hace uso de una serie de datos con los que va a trabajar, modificándolos y gestionándolos de diversas formas. Estos datos y las funciones necesarias para hacer uso de ellos se recogen el la clase documento. Por otra parte, la clase vista es la encargada de contener las funciones necesarias para la representación gráfica de las operaciones sobre los datos [13]. La estructura de la aplicación (con las modificaciones incluidas en este trabajo de fin de grado), en cuanto a las clases vista/documento, queda como se muestra en la Figura 3.4.

La estructura de clases de la aplicación en esta versión se muestra en la Figura 3.5. En este esquema se observa que los módulos Miocon y Espacio hacen uso directo de la clase *CBloqMuestras*; el resto de herramientas, incluida la añadida en esta última

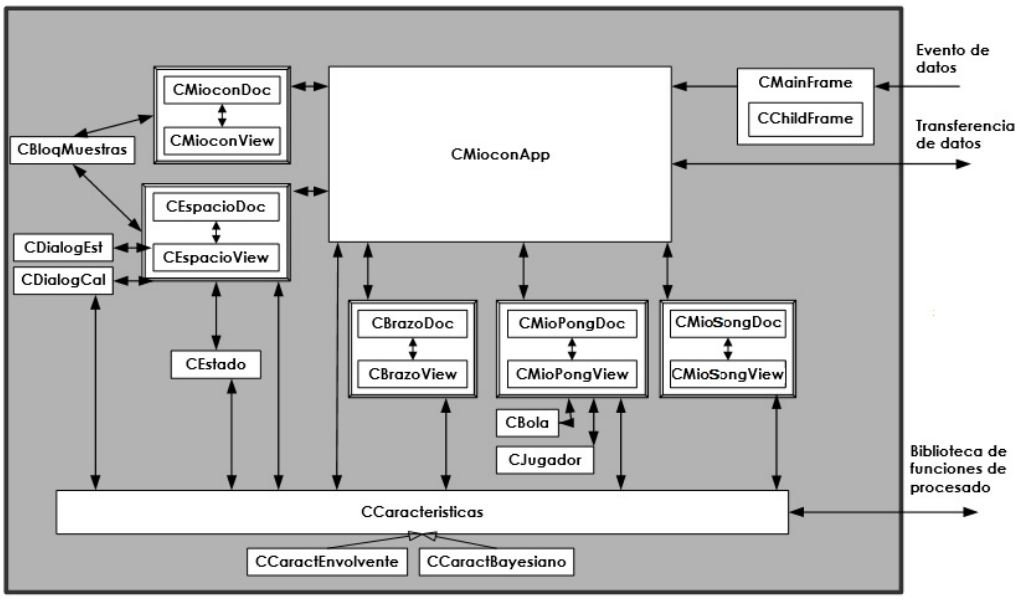


Figura 3.5: Estructura de clases de la aplicación *Miocon.exe*.

versión, trabajan con los datos de calibración generados a partir de la herramienta espacio y almacenados en la clase *CCaracterísticas*. Una vez realizado el calibrado, todas las herramientas pueden trabajar al mismo tiempo, haciendo uso común de los datos, bien sea el flujo original, procedente del conversor(en el caso de Miocon), o los datos calculados tras el procesado en el calibrado. La explicación más detallada de la estructura del código se lleva a cabo en el capítulo cuarto de este documento.

Los cinco tipos de documentos soportados por la aplicación se relacionan con las cinco herramientas disponibles y se desarrollan a continuación. [9].

3.2.1. Miocon

La herramienta *Miocon* hace uso directo de las muestras que le llegan del sistema de adquisición (previo procesado por el módulo *Conversor.dll*).
Su clase documento asociada es *CMioconDoc*, que contiene los datos que la herramienta necesita para poder representar la señal en el dominio temporal. Ésta, a su vez, tiene tantos objetos *CMioconView* asociados como canales estén configurados en el conversor, ya que cada uno de estos canales se muestra como una ventana individual. Esta representación, en caso de tratarse de una señal mioeléctrica(SME), como es la idea primigenia del uso del programa, supondría el electromiograma(EMG) del sujeto.
La ventaja que supone este modelo de trabajo, con múltiples vistas múltiples documentos, es de gran importancia en este módulo, ya que puede presentar varias vistas para el mismo documento, representando cada una de ellas un canal de los configurados en el conversor. Para permitir la visualización de cada canal, cada vista se encarga de extraer las muestras correspondientes al canal que representa, ya que

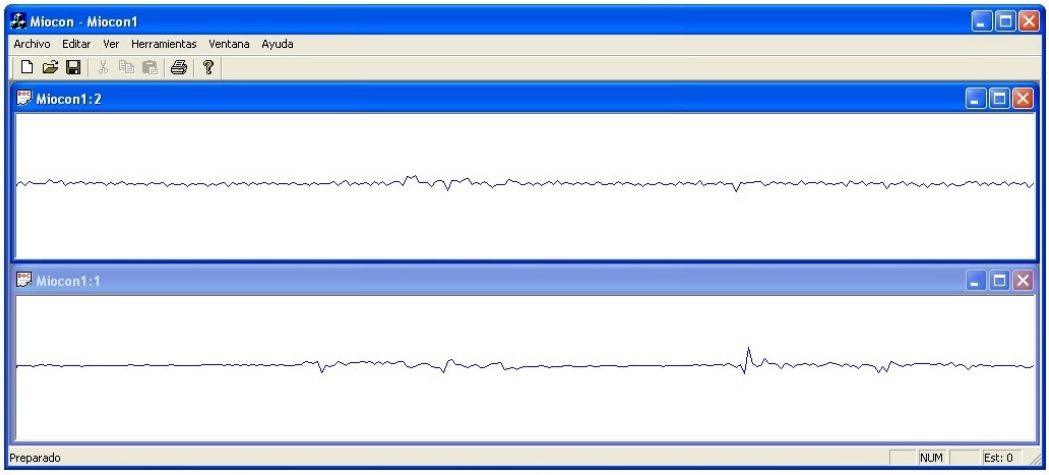


Figura 3.6: Señal mioeléctrica de bíceps y tríceps de un sujeto en reposo.

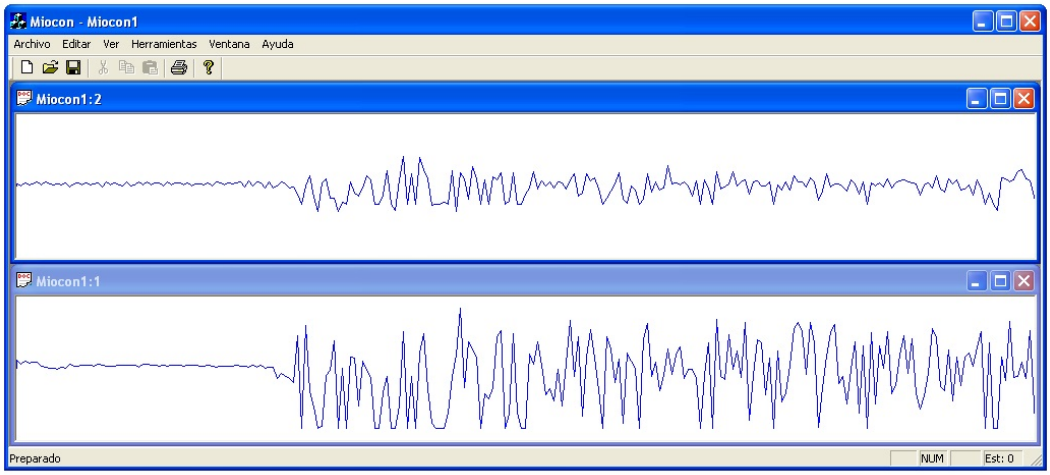


Figura 3.7: Señal mioeléctrica de bíceps y tríceps de un sujeto que pasa de reposo a realizar un esfuerzo.

todas la muestras se encuentran incluidas como un único flujo serie de muestras en la clase documento asociada, *CMioconDoc*.
Esta herramienta recibe de la aplicación principal, *MioconApp*, el bloque de muestras a representar, organizado en una multitrama tal y como se mostraba en la Figura 3.3 , y de ésta extrae las v muestras de los respectivos canales (dos por defecto) y los representa, tal y como se observa en la Figura 3.6 y 3.7.

En el caso de que la señal que se muestra por pantalla tuviera una amplitud muy pequeña o si se observara mucho ruido introducido por la red, se podría aplicar a esta señal una ganancia de +35dB o un filtro de 50Hz respectivamente. Estas configuraciones se realizan, cerrando todas las vistas de las herramientas activas, en la ruta *Herramientas » Opciones*, que abre un menú como el mostrado en la Figura 3.8, en el que también se puede configurar el puerto de entrada y seleccionar los canales.

Figura 3.8: Menú de configuración de la aplicación.

Figura 3.9: Menú de configuración del conversor.

Para configurar la salida del conversor, la ruta que ha de seguirse es la misma que en el punto anterior sin llegar a entrar en la pestaña de *Opciones*, presentándose un menú como el mostrado en la Figura 3.9.

Los archivos generados por esta herramienta tienen la extensión *.mio*.

3.2.2. Espacio

La herramienta *Espacio* representa en pantalla una serie de posiciones o estados entre los que se podrá navegar con un puntero. Inicialmente, previo calibrado, el sujeto fija una serie de estados (que podrán ser hasta un máximo de 5 o 7 dependiendo de si los niveles de intensidad de esfuerzo con los que trabajamos son 3 o 4), llevando el cursor hasta el punto deseado mediante esfuerzos isométricos del propio brazo, que genera cambios en la señal mioeléctrica capturada y con ello distintos giros y movimientos del puntero por la pantalla. Una vez fijados los estados, el sujeto deberá llegar con el cursor a cada uno de éstos, buscando controlar el estado destino (el número del estado se fija con un menú como el mostrado en la Figura 3.10), lo que le fuerza a intentar ser consciente de los cambios mioeléctricos que suponen cada uno de sus movimientos. El origen de coordenadas se sitúa en el la esquina superior izquierda y el movimiento del cursor se va a realizar en el eje horizontal, correspondiente con el canal 1 y generalmente con el bíceps y en el eje vertical, que representa el canal 2 y va a registrar los esfuerzos del tríceps. La estética de la herramienta se muestra en la Figura 3.11. Como se ve en el ejemplo, la circunferencia de estado número 4

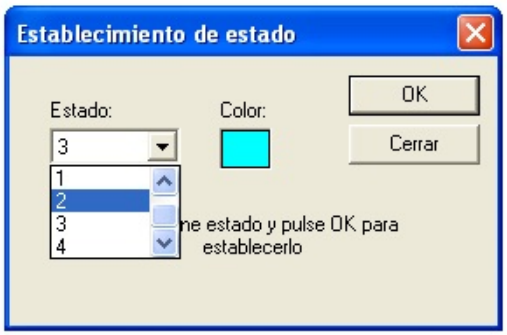


Figura 3.10: Menú de fijación de los estados para la herramienta *Espacio*.

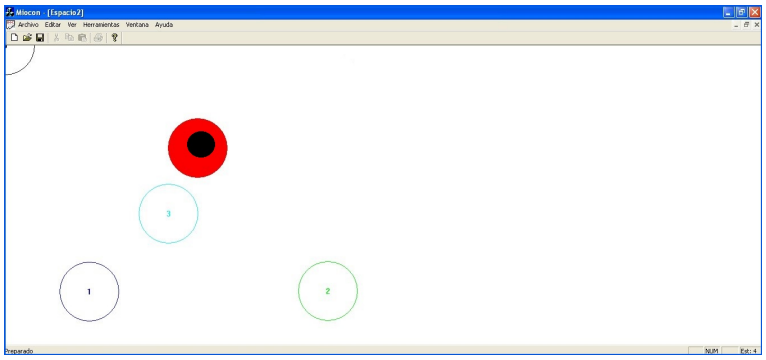


Figura 3.11: Pantalla ejemplo de la herramienta *Espacio*.

aparece sombreada de color rojo: se detecta proximidad al estado resaltado y el rojo es el color asignado a ese estado.

Esta herramienta está pensada para que el sujeto aprenda a dirigir el cursor hacia cada uno de los estados y a mantenerlo centrado en el mismo, siendo la primera herramienta del programa que busca que el sujeto vaya teniendo consciencia de las variaciones de la propia señal. Está pensada como paso previo y complementario a la herramienta *Brazo*.
Los archivos generados por esta herramienta tienen la extensión *.bsp*

Calibrado

Mediante la calibración, se calculan los esfuerzos promedios de cada músculo, así como un valor de referencia, lo que permitirá, posteriormente, comparar el esfuerzo de cada sujeto con su propia referencia, dando una muestra más ajustada a la realidad de cada individuo.
La herramienta ofrece la posibilidad de cargar una configuración anterior, cargando también el calibrado que se realizó. Pero en caso contrario, al abrir la herramienta, se nos muestra una ventana que nos avisa de que se va a proceder al calibrado y ,a continuación, le sigue la secuencia de ventanas recogida en la Figura 3.12.

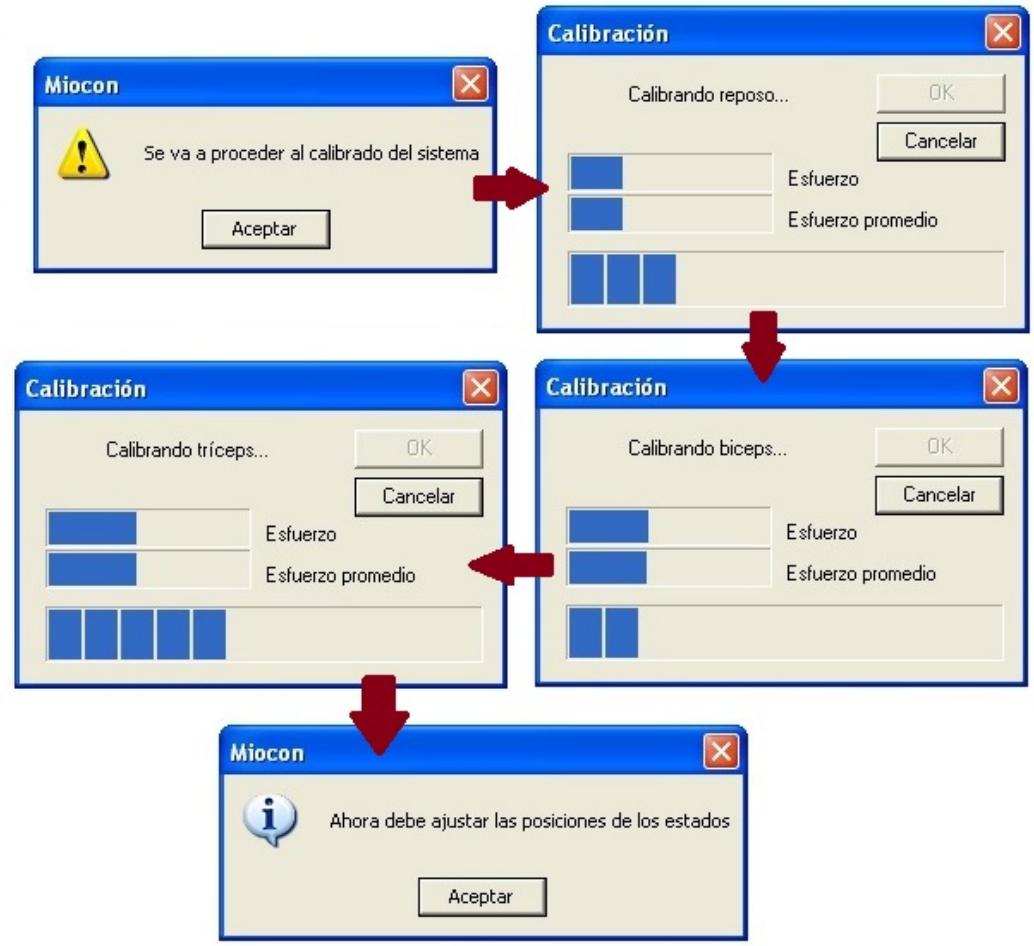


Figura 3.12: Secuencia del proceso de calibrado.

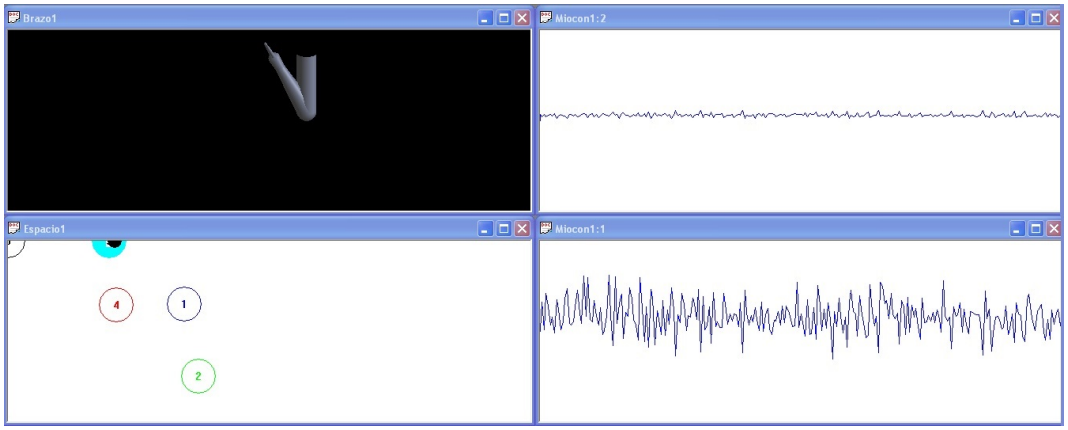


Figura 3.13: Ejemplo de relación entre el estado de la herramienta ,*espacio*, el movimiento del *Brazo virtual* y la señal capturada por la herramienta *Miocon*.

3.2.3. Brazo

La herramienta *Brazo* consiste en una representación tridimensional de un brazo que realiza diferentes movimientos. Estos movimientos dependen de los estados fijados anteriormente en la herramienta *Espacio*, por lo que es necesario trabajar con ambas herramientas abiertas para poder usar la configuración de *Espacio*. Los posibles movimientos que el brazo puede realizar dependiendo de los estados fijados son los siguientes:

- Estado 0: mantiene el gesto anterior.
- Estado 1: flexiona el codo.
- Estado 2: extiende el codo.
- Estado 3: abre la mano (ejemplo mostrado en la Figura 3.13).
- Estado 4: cierra la mano.

En este caso se ha trabajado sobre la versión que acepta 5 estados, pero como se comentó anteriormente el programa se puede adaptar para trabajar con 7 estados, en cuyo caso, las otras dos correspondencias serían las siguientes:

- estado 5: pronación de la muñeca (presentando el dorso de la mano).
- estado 6: supinación de la muñeca (presentando la palma de la mano).

3.2.4. MioPong

La herramienta *MioPong* se basa en el clásico juego Pong, que consiste en el enfrentamiento de dos jugadores, cada uno representado por una barra que se desplaza en ambos sentidos en la dirección vertical, tal y como se muestra en la Figura 3.14.
Aunque la apariencia es la misma que en el juego clásico, en nuestro caso, el movimiento de las barras se controla mediante los diferentes esfuerzos realizados con cada uno de los dos músculos antagonistas. Para modelar este movimiento es necesario

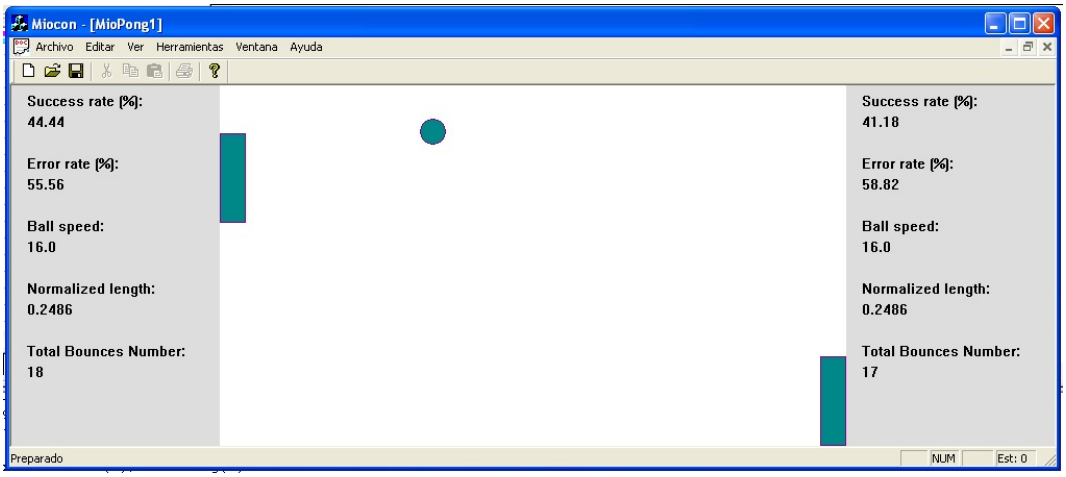


Figura 3.14: Apariencia de la herramienta *MioPong*.

Figura 3.15: Cuadro de diálogo que alerta de la necesidad de un calibrado previo.

haber realizado previamente el calibrado mediante la herramienta *Espacio*, ya que en caso de intentar abrir este módulo sin la calibración previa aparecerá un cuadro de diálogo como el mostrado en la Figura 3.15.

El origen de coordenadas para ambos canales se sitúa en la parte superior de la pantalla, de forma que con ambos músculos en reposo la aplicación se muestra con las dos barras en la parte superior, tal y como se observa en la Figura 3.16. La realización de un esfuerzo con el bíceps provoca el movimiento descendente de la barra situada a la izquierda, mientras que la barra de la derecha se mueve en función de los esfuerzos del tríceps.

Esta herramienta incluye, además, el cálculo de la tasa de acierto y error de cada uno de los músculos implicados, la velocidad de la bola y la longitud normalizada de la barra que representa a cada jugador. Estos dos últimos parámetros pueden ser ajustados para dificultar o facilitar el entrenamiento. El acierto en la colisión con la barra se muestra mediante el cambio de color de la bola, que pasa a ser de color rojo durante un breve periodo de tiempo, tal y como se muestra en la Figura 3.17

Con *MioPong* el sujeto tiene la motivación extra de estar jugando al mismo tiempo que entrena para el uso de una prótesis. Con la herramienta se busca una mejora en la precisión y tiempo de reacción del sujeto. Además, el hecho de registrar los datos de la tasa de acierto y error y de poder modificar la dificultad mediante el cambio de ciertos parámetros, supone un añadido a la herramienta, que permite un mayor control de la mejora del individuo de cara al uso futuro de una prótesis.

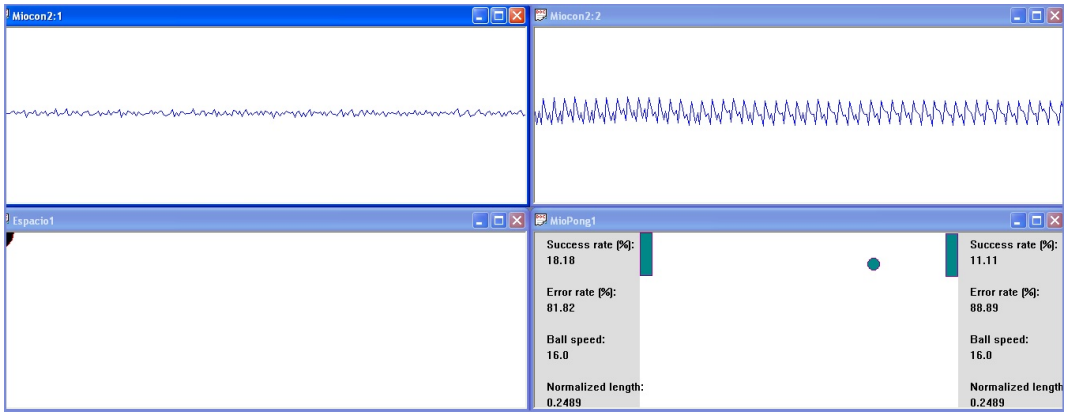


Figura 3.16: Representación de las herramienta *MioPong*, *Miocon* y *Espacio* con ambos músculos en reposo.

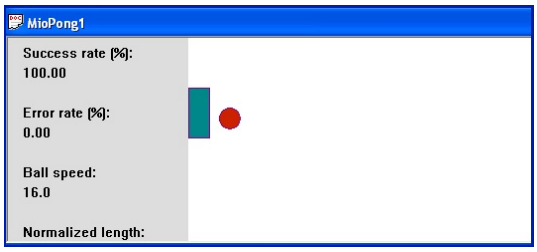


Figura 3.17: Acierto en la colisión con la herramienta *MioPong*.

3.2.5. MioSong

La herramienta *MioSong* es el objeto central de este trabajo. Se plantea como un módulo que busca dar una funcionalidad extra a la aplicación y sumarle un punto de interés. Esta herramienta introduce como novedad la interactuación con la tarjeta de sonido, ya que lo que sucede cuando se usa, es que pasamos a escuchar un tono de frecuencia directamente proporcional al esfuerzo realizado por el músculo que se reproduce.
MioSong puede ser utilizada en cuatro modos, tal y como se muestra en la Figura 3.18 :

- **Reproducir**: reproduciendo ambos músculos.
- **Reproducir bíceps**: reproduciendo únicamente los esfuerzos del bíceps.

Figura 3.18: Menú de herramientas de *MioSong*.

- **Reproducir tríceps**: reproduciendo únicamente los esfuerzos del tríceps.
- **Entrenamiento**: reproduciéndose muestras alternadas en caso de acierto.

El sonido que se produce está generado por una función *Beep*, que se trata de una función síncrona que da lugar a un tono de la frecuencia y duración que se determine. Esta función es capaz de reproducir tonos cuya frecuencia se encuentre comprendida entre los valores [37,32767](valor tomado de [11]).
Por otro lado, los datos con los que trabaja el programa y que dejan constancia del esfuerzo realizado por los dos músculos bajo registro, van a tomar valores comprendidos entre 0 y 1. Esto se debe a que se trata de valores normalizados de la diferencia entre el esfuerzo puntual (en el tiempo de ventana, t_v, definido en la sección 3.1 de este documento) del músculo que se quiere reproducir y su referencia.
Dado que se busca una relación lineal entre la frecuencia del tono generado y el esfuerzo capturado, se le pasará a la función *Beep*, como parámetro de frecuencia, el valor del esfuerzo del músculo de interés multiplicado por un factor 32767. De esta forma, las muestras podrán llegar a generar la frecuencia máxima posible sin llegar a salirse del rango permitido por la función *Beep*.
En cuanto al parámetro que fija el tiempo de reproducción, se le pasará a la función un 1, que establece que el tono se ha de reproducir durante 1ms. Se fija este valor porque se busca un compromiso entre una duración suficiente para que sean distinguibles las características de los diferentes tonos (más agudos o más graves) y un tiempo lo suficientemente reducido como para que el retardo que añade al programa no suponga un problema para la respuesta en tiempo real. Al tratarse de una función síncrona, el programa se detiene en el *Beep* hasta el final de su reproducción. Los archivos generados por esta herramienta tienen la extensión *.bsg*

Reproducir

En el modo *Reproducir* se pasan de manera sucesiva muestras alternativas procedentes de ambos canales, de manera que primero se intenta reproducir la muestra procedente del bíceps y posteriormente la del tríceps.
Cuando se encuentra funcionando este modo, en la pantalla aparece un mensaje como el que se observa en la Figura 3.19.

Reproducir bíceps

En este modo se pasan a la función *Beep* únicamente datos relacionados con las muestras del primer canal, correspondiente con el bíceps. Busca que el sujeto se centre únicamente en los cambios mioeléctricos que van a producir sus movimientos en este músculo.
Cuando se encuentra activo se muestra en pantalla un mensaje como el mostrado en la Figura 3.20.

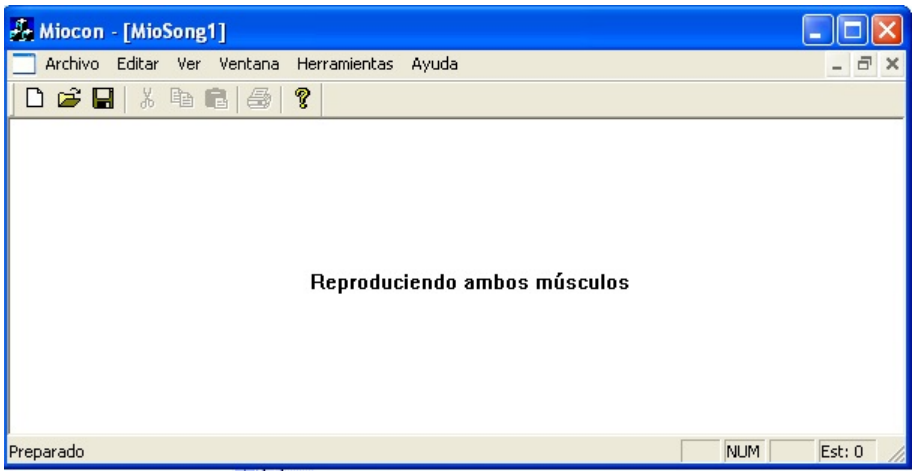


Figura 3.19: Modo *Reproducir* de *MioSong*.

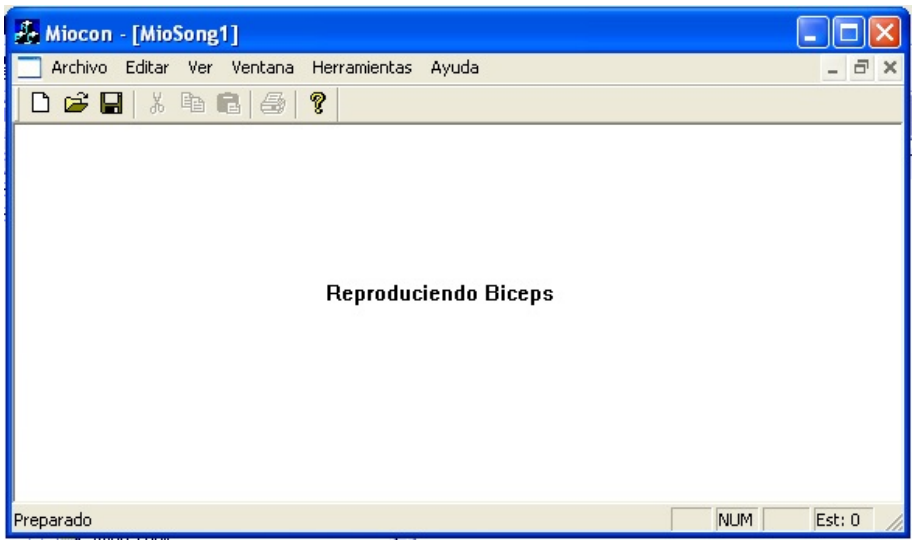


Figura 3.20: Modo *Reproducir bíceps* de *MioSong*.

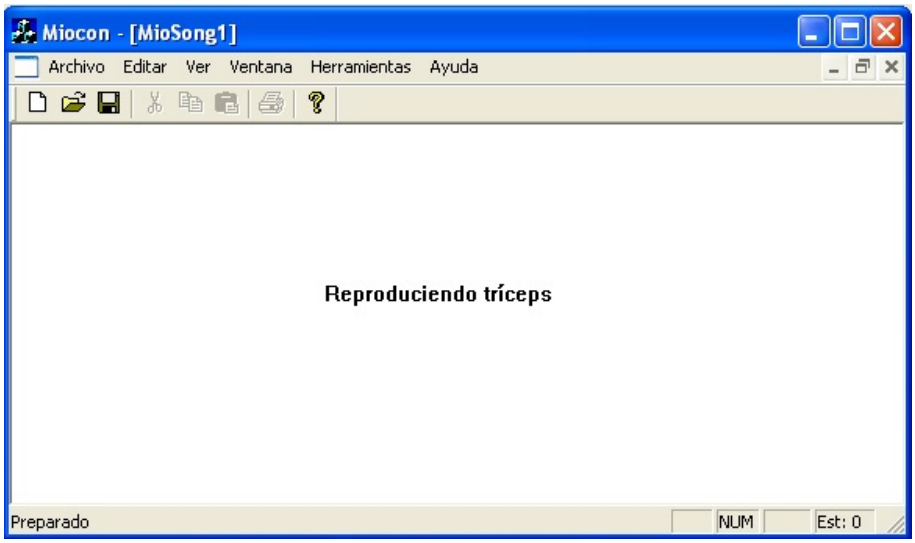


Figura 3.21: Modo *Reproducir tríceps* de *MioSong*.

Reproducir tríceps

Cuando se trabaja seleccionando este modo, las muestras que se reproducen proceden del canal 2, que toma las muestras del tríceps. El objetivo es el mismo que para el modo anterior, pero en este caso, busca que el sujeto sea capaz de centrarse únicamente en los esfuerzos del tríceps.
Cuando se activa la herramienta *Reproducir tríceps* aparece la pantalla mostrada en la Figura 3.21.

Entrenamiento

El modo *Entrenamiento* es el que ofrece opciones de mayor interés, ya que el programa emite un mensaje que hace que el sujeto trate de ir cambiando el músculo que ejerce mayor esfuerzo y contabiliza el porcentaje de intentos en los que el sujeto efectivamente superpone el músculo que le indica el programa.
En este modo, aparece en la pantalla el músculo que deberá predominar en ese momento y si, efectivamente, es el músculo que está realizando un mayor esfuerzo, se muestra por pantalla que es correcto y se emite el sonido de frecuencia relacionada al esfuerzo del músculo. Sin embargo, si el músculo predominante no es el que el programa indica, se muestra la palabra 'error' en pantalla. Los porcentajes de acierto y error se actualizan constantemente mientras se encuentra seleccionado este modo.
Se puede ver un ejemplo de la estética en la Figura 3.22, en la que aparece, en la parte superior, un caso de acierto en el que se pide predominancia del bíceps y, en la parte inferior, un caso de error, ya que aunque se pide la predominancia del bíceps de nuevo, se puede observar con ayuda de la herramienta *Espacio* (representada en la parte inferior de ambos casos), que el programa está detectando mayor esfuerzo por parte del tríceps.

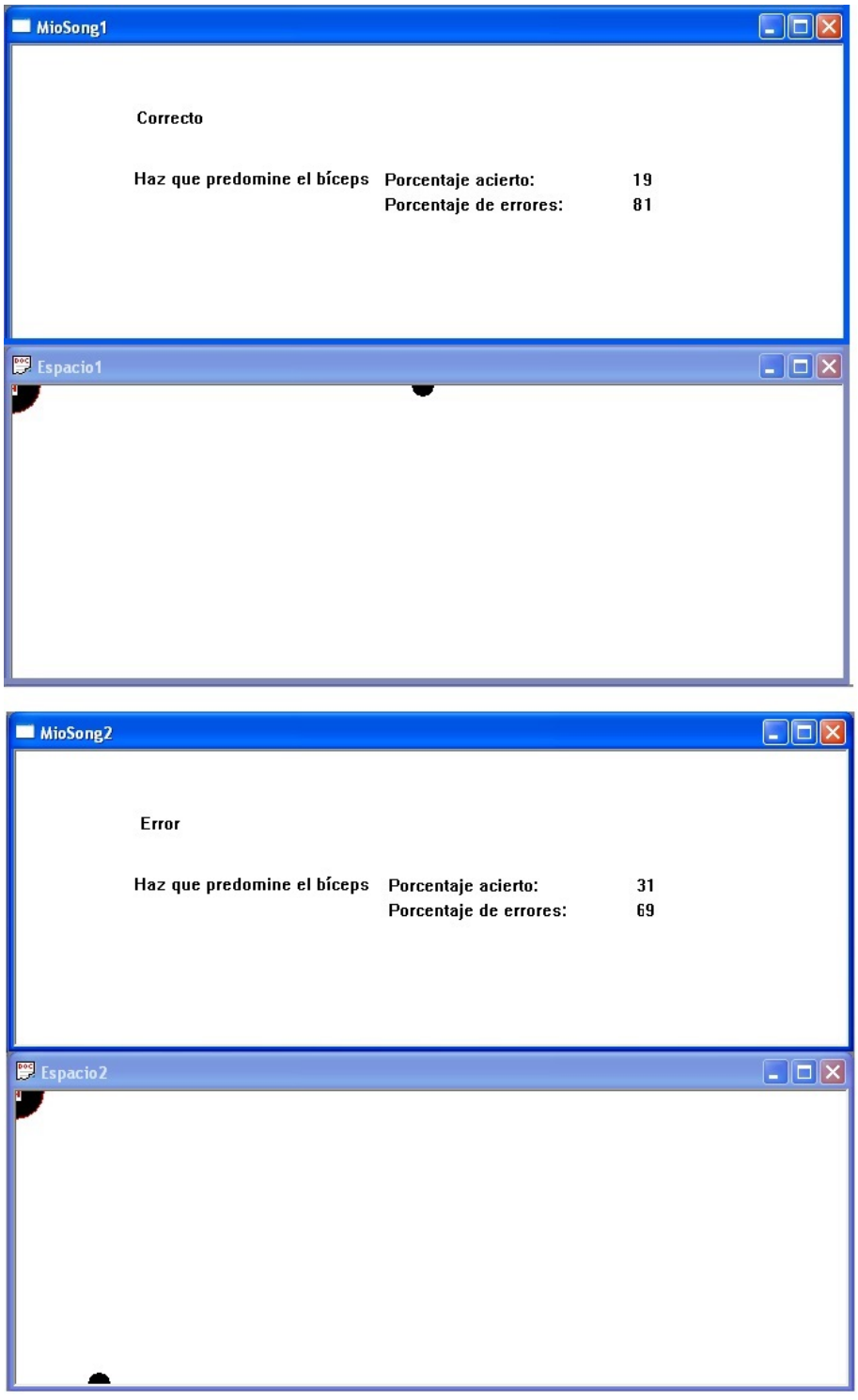


Figura 3.22: Ejemplo de acierto y error en la herramienta *MioSong*.

Capítulo 4

Desarrollo de la herramienta *MioSong*

El proceso de desarrollo de este trabajo de fin de grado se ha llevado a cabo en varias etapas. Inicialmente, fue necesario un periodo de toma de contacto y entendimiento de la herramienta de desarrollo del código, Microsoft Visual C++ 6.0; en segundo lugar, los esfuerzos se centraron en comprender la estructura del código de partida de *Miocon*; el siguiente paso, consistió en la implementación de un nuevo módulo,*MioSong*; y por último, se desarrolló el código propio de las funcionalidades de la herramienta *MioSong*.

4.1. Familiarización con el entorno de desarrollo

En primer lugar, fue necesario un periodo de adaptación al entorno de desarrollo *Microsoft Visual C++ 6.0*. Para ello se siguieron los contenidos y tutoriales propuestos en [13], ojeando la mayor parte del contenido del libro y revisando en profundidad los capítulos dedicados a documentos y vistas y a la impresión por pantalla, así como el Anexo A, dedicado al repaso de C++ y a conceptos de orientación a objetos, que junto con el manual [14], sirvieron de apoyo para el aprendizaje del lenguaje C++. Para la práctica con el lenguaje y el entorno, se desarrollaron pequeños programas enfocados a trabajar con la tarjeta de sonido, creando pequeñas aplicaciones que mostraran, además, un mensaje por pantalla, tomando así contacto con la clase *view*.

4.2. Estructura del código de partida

Para el entendimiento en profundidad de la estructura del código del programa, se recurrió al apéndice A de [9].
Tal y como se muestra en la Figura 3.1, la aplicación está compuesta por tres proyectos que interactúan entre sí, pero dado que el propósito de este trabajo consiste en la modificación del proyecto que contiene la aplicación principal, *Miocon*, se centró

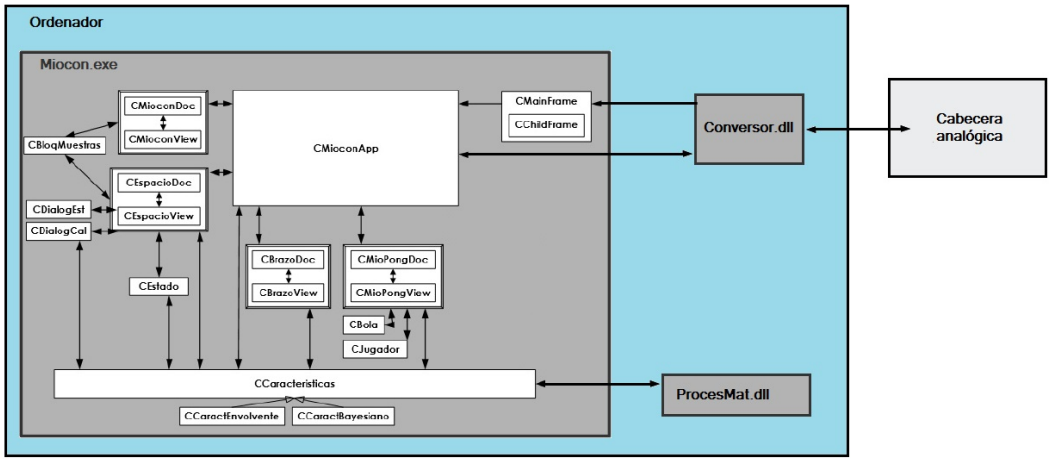


Figura 4.1: Relación de los componentes y clases de la plataforma UVa-NTS.

la atención en la comprensión del funcionamiento y estructura de éste.
En la Figura 4.1 se representa un esquema de relación entre todos los elementos de la plataforma UVa-NTS. La cabecera analógica muestrea y cuantifica la señal, dejando una serie de datos, expresados en voltios, disponibles para ser utilizados. El *Conversor.dll* es el encargado de generar un mensaje alertando de la disponibilidad de los datos y la localización de los mismos (*evento de datos* en la Figura 3.5). Este mensaje es recibido por *CMainFrame*, que alerta a *CMioconApp*, la cual recoge las muestras y las distribuye entre las *clase documento* que las utilicen directamente (*CMioconDoc*y *CEspacioDoc*), ya que el resto trabajan a partir de las características sacadas de los datos. Cada una de las *clases documento* procesa los datos y actualiza sus *clases vista*.

El programa Microsoft Visual C++ 6.0 ofrece, entre muchas otras funcionalidades, un *WorkSpace*, que incluye, entre sus pestañas, el *ClassView*, una ventana permite ver todas la clases definidas en el proyecto y agruparlas en carpetas por bloques funcionales. Por lo que el primer paso consistió en entender el funcionamiento y relación de las distintas clases que se observaban en el *ClassView*tal y como se muestran en la Figura 4.2. En esta figura, aparecen algunas clases agrupadas por carpetas, a excepción de la carpeta de *Procesado*, que contiene clases comunes a todos los módulos. El resto de carpetas contienen las clases relacionadas con cada herramienta. Las clases que se encuentran en la carpeta raíz (*Miocon Classes*) también son comunes a todas las herramientas.

Para el entendimiento en profundidad de las diferentes clases involucradas en el funcionamiento del programa, se recomienda el Apéndice A de [9], documento que se siguió en este punto del desarrollo.

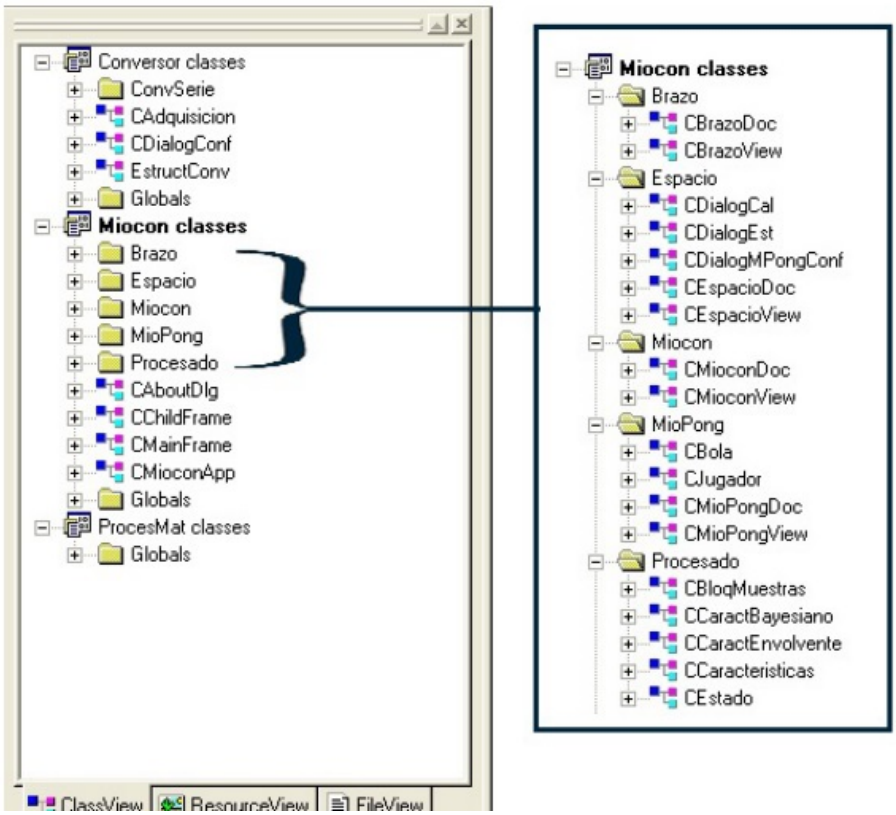


Figura 4.2: *ClassView* del programa del que se partió.

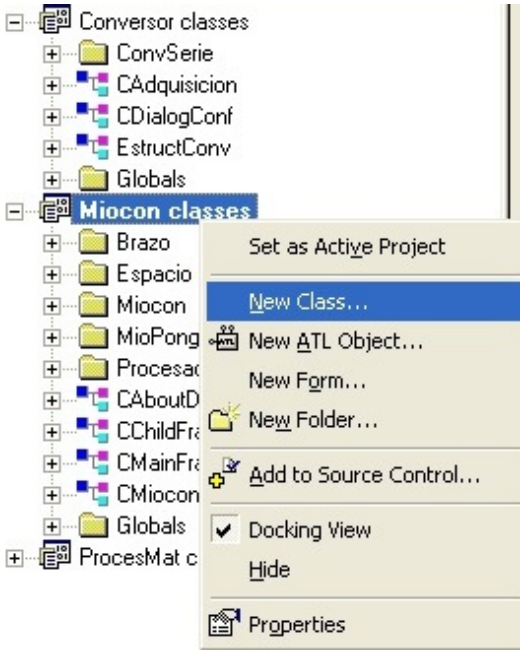


Figura 4.3: Pasos para crear una nueva clase.

4.3. Implementación del módulo

Para incluir una nueva herramienta en el programa y dada la estructura modular del mismo, el primer paso que se ha de seguir es añadir una nueva clase documento (-doc) y una clase vista(-view) relacionada con ésta. Resulta conveniente además, organizarlo en una carpeta independiente dentro del programa, ya que va a ayudar a mantener un orden y va a facilitar el entendimiento de la relación entre clases.
Dada la funcionalidad que se esperaba que tuviese el programa, se decidió llamar a la herramienta *MioSong*. En primer lugar, se crearon la clase documento y la clase vista que serían utilizadas para esta herramienta, por lo que se las llamó CMioSongDoc y CMioSongView respectivamente. Para la creación de éstas se siguieron los siguientes pasos: botón derecho en *Miocon classes* » *ClassView* » *New Class...*, tal y como se muestra en la Figura 4.3.

Una vez hecho esto, en el menú *New Class*, se seleccionó una clase de tipo *MFC* (Microsoft Foundation Classes, aplicación que inclye la biblioteca de clases estándar para aplicaciones Windows [13]) , con clase base *CDocument* y se le asignó el nombre CMioSongDoc (Figura 4.4).

A continuación, se realizó el proceso equivalente para la creación de la clase CMioSongView, cuya clase base es *CView* (Figura 4.5)

Con las clases ya establecidas, se creó la carpeta que las contendría,a la que se llamó *MioSong*. Para ello se seleccionó, en el *ClassView*, con el botón derecho *Miocon classes* » *New Folder*. Para incluir las clases en la carpeta, se arrastraron hasta ésta desde el visor de *ClassView*, que quedó como se muestra en la Figura 4.6.

Como se mencionó anteriormente, la clase vista muestra los datos almacenados en el

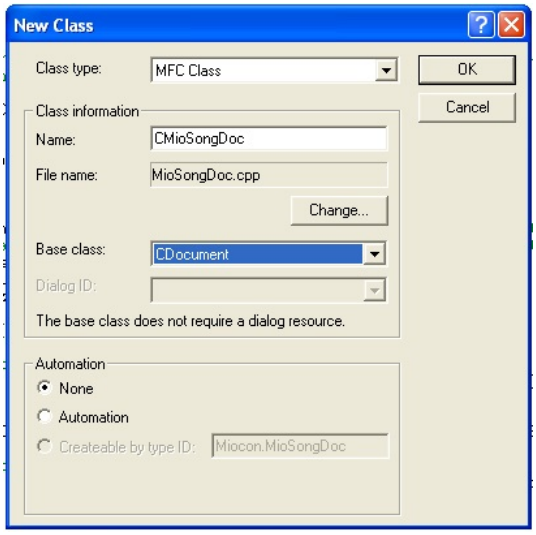


Figura 4.4: Creación de la clase CMioSongDoc.

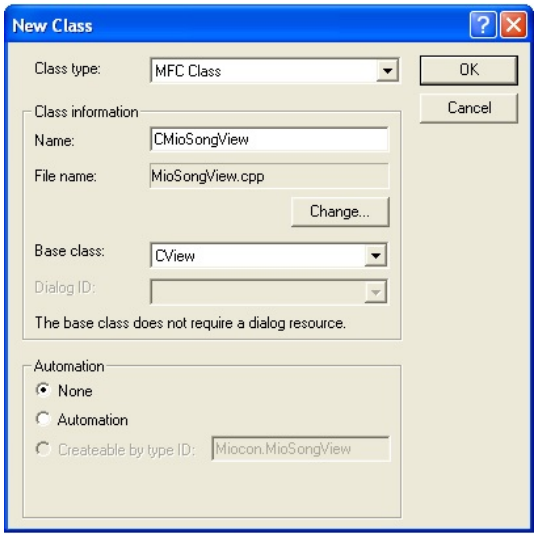


Figura 4.5: Creación de la clase CMioSongView.

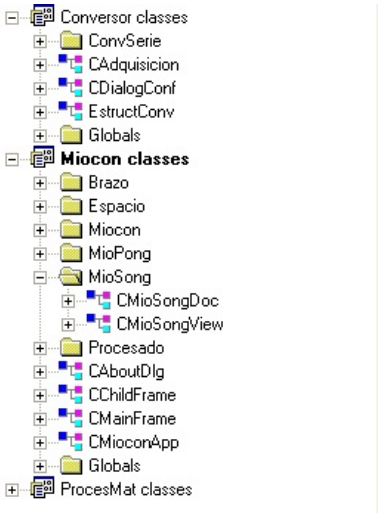


Figura 4.6: *ClassView* una vez añadidas las clases del módulo *MioSong*.

objeto documento y permite al usuario modificarlos. El objeto vista deberá mantener un puntero al objeto documento, que utiliza para acceder a las variables miembro del documento y así poder visualizarlas o modificarlas [13]. Para mantener este puntero al documento, fue necesario seguir una serie de pasos:

1. Añadir tanto en el archivo de cabecera (*MioSongView.h*), como en *MioSongView.cpp*, un include del documento de *MioSong*:

   ```
   #include "MioSongView.h"
   ```

2. Añadir, en el archivo de cabecera de la clase CMioSongView, una nueva línea como atributo público de la clase:

   ```
   // Operations
   public:
   CMioSongDoc* GetDocument();
   ```

3. Añadir la implementación de la función *GetDocument* en *MioSongView.cpp*:

   ```
   CMioSongDoc* CMioSongView::GetDocument()
   {
   ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CMioSongDoc)));
   return (CMioSongDoc*)m_pDocument;
   }
   ```

4. Añadir al final del archivo de cabecera de la clase CMioSongView la declaración de *GetDocument* como inline (función tal que el compilador puede sustituir la llamada de ésta por la definición de la misma en cualquier lugar del código [15]):

   ```
   inline CMioCongDoc* CMioCongView::GetDocument()
      { return (CMioCongDoc*)m_pDocument; }
   ```

A continuación, se configuró el código de forma que se mostrara la nueva herramienta

entre las opciones disponibles de *Miocon.exe*, siendo posible su selección desde el menú principal. Los pasos seguidos fueron los siguientes:

1. En el *Workspace*, en la pestaña *ResourceView* » *Menu*, botón derecho *Insert Menu*. Se crea así un recurso de menú, que aparece en el *WorkSpace* con un nombre parecido a *IDR_MENU1*. Para cambiar el identificador, se pulsó el botón derecho sobre el nombre y se seleccionó *Properties*. En *ID* se remplazó el nombre dado por *IDR_MIOSONGTYPE*.

2. En el *Workspace*, *ResourceView* » *String Table* » *String Table*, sobre alguno de los identificadores tipo *IDR_ ...*, click en botón derecho » *New String*, en *ID* se establece el mismo que en el recurso de menú, en nuestro caso se reemplazó el dado por *IDR _ MIOSONGTYPE*; en *caption*, en base a lo establecido en el resto de entradas para las herramientas ya configuradas, se estableció *\nMioSong\nMioSong\nArchivos MioSong (*.bsg)\n.bsg\nMioSong.Document \nMioSong Document*.

3. Se añadió en el archivo *Miocon.cpp* de CMioconApp los includes de la clase vista y documento de *MioSong*:

   ```
   #include "MioSongDoc.h"
   #include "MioSongView.h"
   ```

4. En la función InitInstance de CMioconApp se establece el registro de las plantillas de los documentos de la aplicación (*CMultiDocTemplate*). Fue necesario añadir, al final de las cuatro declaraciones de las plantillas, una quinta para *MioSong* añadiendo las siguientes líneas de código:

   ```
   CMultiDocTemplate* pDocTemplate5;
   pDocTemplate5 = new CMultiDocTemplate(
   IDR_MIOSONGTYPE,
   RUNTIME_CLASS(CMioSongDoc),
   RUNTIME_CLASS(CChildFrame),
   RUNTIME_CLASS(CMioSongView));
   AddDocTemplate(pDocTemplate5);
   ```

Al finalizar estos pasos, aunque la herramienta no realizaba ninguna función ni interactuaba con las muestras, podía ser seleccionada como opción en el menú principal que aparece al abrir la aplicación, abriéndose una ventana que mostraba un menú básico de Windows.

4.4. Desarrollo del código

Una vez incluidas las clases y configurado el menú de inicio, se empezó a desarrollar el código propio de la herramienta *MioSong*.
El funcionamiento que la herramienta debía presentar y que, por tanto, se debía alcanzar con el desarrollo del código, se muestra en el diagrama de flujo de la Figura 4.7.

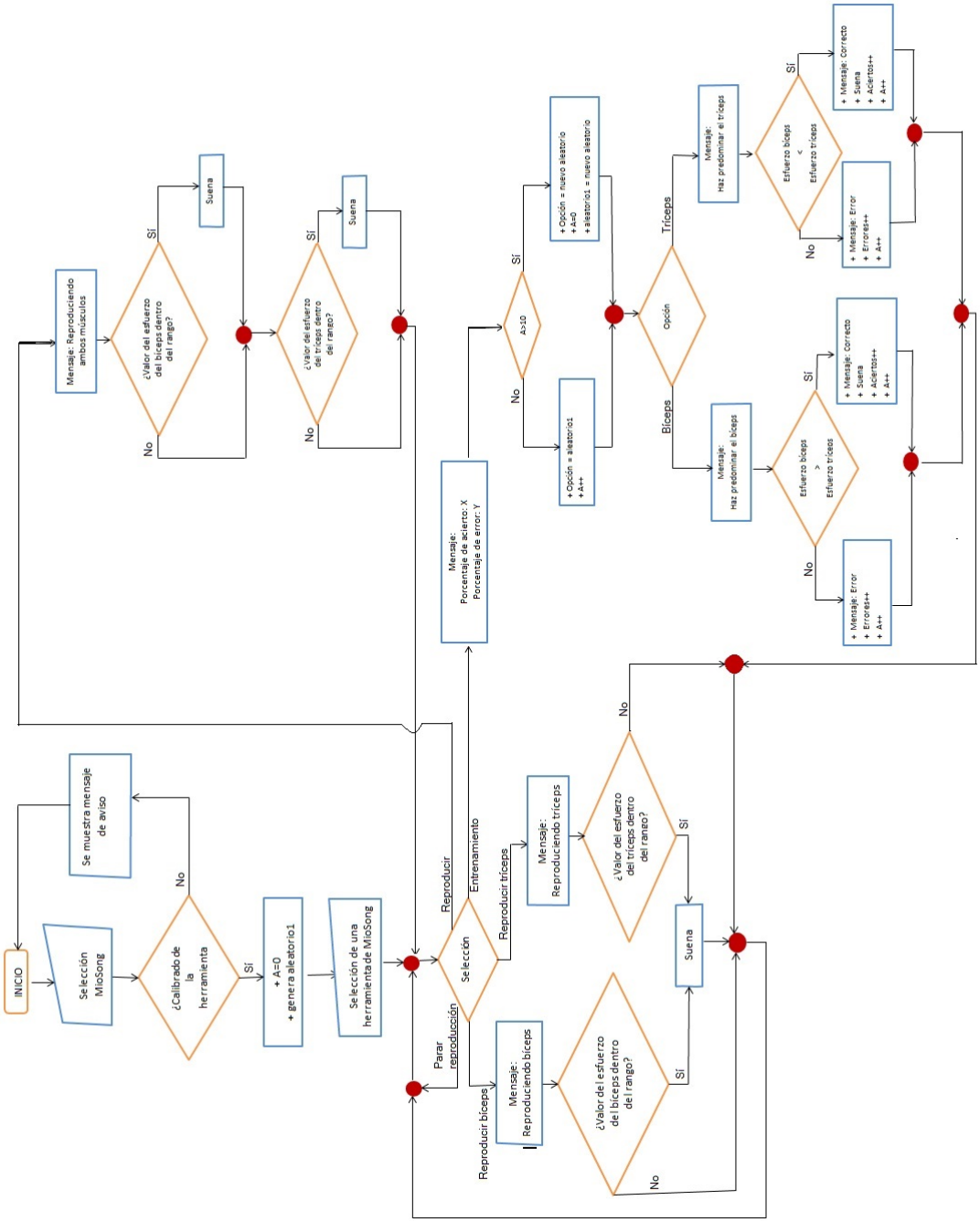


Figura 4.7: Diagrama de flujo de la herramienta *MioSong*

Al igual que en los puntos anteriores, se explican a continuación, por etapas, los pasos seguidos para el desarrollo del código.
En primer lugar, se modificó el menú principal de *MioSong*, de forma que aparezcan las opciones que ofrece en la pestaña *Heramientas*:

1. Se abrió el menú a editar (*ResourceView » Menu » IDR_MIOSONGTYPE*).
2. Dado que la mayoría de funcionalidades son comunes a todas las herramientas, se copiaron de las otras herramientas las pestañas de : *Archivo, Editar, Ver, Ventana y Ayuda*. Para hacer esto, se abrió otro de los menús y pestaña a pestaña se copiaron. A continuación, se pegaron en el menú de *MioSong*, sobre la barra superior (sombreada). De esta forma se mantiene una estética y funcionalidad uniforme entre todas las herramientas.
3. Se añadió una pestaña *Herramientas*. Para llevar esto a cabo, se hizo doble click sobre la barra sombreada, que provocó la apertura de una ventana *Menu Item Properties*. En el espacio de *Caption* se introdujo: &Herramientas. Con '&' se selecciona la 'H' como tecla de acceso rápido (CTRL+H -» Herramientas).
4. Una vez creado, se seleccionó *Herramientas* y se hizo doble click sobre la pestaña emergente de la parte inferior de ésta. Se abrió de nuevo un menú *Menu Item Properties*. En este caso se completaron las pestañas de *ID* y *Caption*. Este proceso se repitió para cada una de las 5 funcionalidades:
   - **Reproducir:** *ID:* ID_HERRAMIENTAS_REPROUCIR,*Caption:* &Reproducir.
   - **Reproducir bíceps** *ID:* ID_HERRAMIENTAS_REPROUCIR_ BICEPS, *Caption:* Reproducir &bíceps.
   - **Reproducir tríceps** *ID:*ID_HERRAMIENTAS_REPROUCIR_ TRICEPS, *Caption:* Reproducir &tríceps.
   - **Entrenamiento** *ID:* ID_HERRAMIENTAS_REPROUCIR_ ENTRENAMIENTO, *Caption:* &Entrenamiento.
   - **Parar reproducción** *ID:* ID_HERRAMIENTAS_PARAR_ REPROUCCION, *Caption:* &Parar reproducción.

A continuación, se añadieron los esqueletos de las funciones que se ejecutan al seleccionar cada recurso. Para ello, se hizo click derecho sobre el recurso en el menú y se seleccionó *Class Wizard...*, observándose algo similar a lo mostrado en la Figura 4.8. Todo ID de recurso tiene asociados dos mensajes: COMMAND y UPDATE_COMMAND_UI. El primero le permite añadir una función que maneja el usuario seleccionando una opción de menú . El segundo le permite añadir una función para establecer el estado del componente del menú [13].

Se seleccionó uno de los identificadores de recuro de la pesataña *Object ID* y la opción de mensaje COMMAND. Al seleccionar COMMAND, se abre una pantalla como la mostrada en la Figura 4.9, que nos permite cambiar los nombres de las funciones por defecto, lo cual no es apropiado casi nunca [13].

Tras aceptar el nombre por defecto (*OK*), se repitió el proceso para cada uno de los

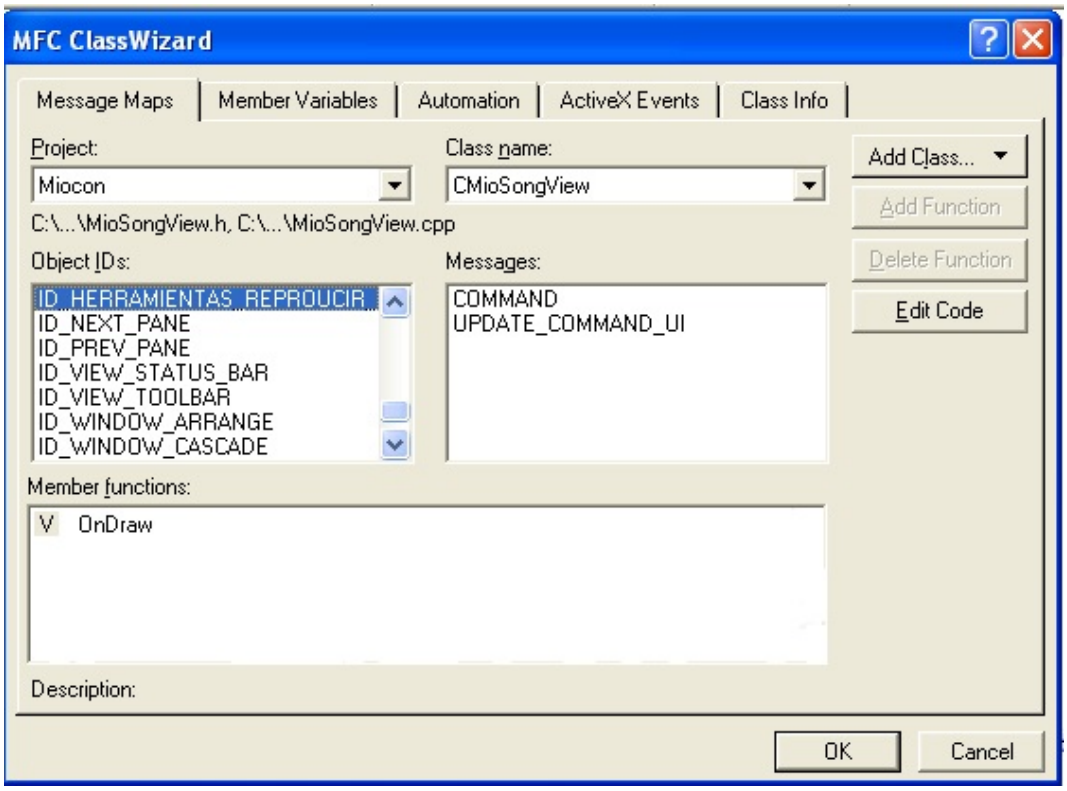


Figura 4.8: Ejemplo de pantalla ClassWizard

Figura 4.9: Ventana Add Member Function

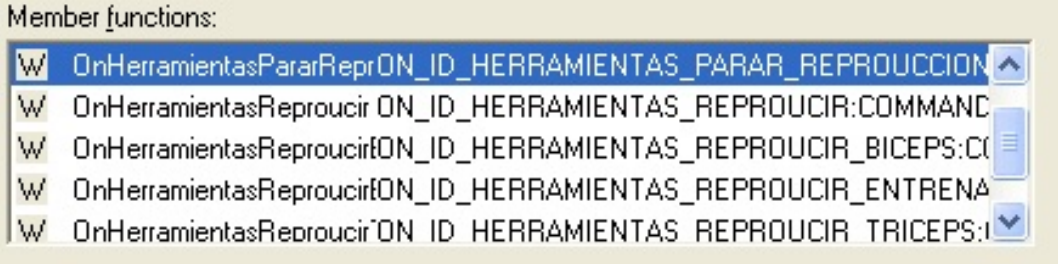


Figura 4.10: Cuadro Member Function del *ClassWizard*

recursos, quedando finalmente un cuadro *Member functions* como el mostrado en la Figura 4.10.

Una vez hecho esto, era momento de empezar a escribir el código funcional de la herramienta *MioSong*. Esta herramienta trabaja con las características sacadas de las muestras, por lo que es necesario que se encuentre activo el calibrado de la herramienta *Espacio* antes de abrirla. Por tanto, se estableció un control que dejara abrir la herramienta únicamente en caso de calibrado activo. Para ello, se añadió una *sentencia if* al código de la función OnNewDocument de CMioSongDoc, ya que esta función es llamada cada vez que se crea un objeto documento de *MioSong* (Es decir, cada vez que se selecciona la herramienta), por lo que el programa comprueba si, efectivamente, se cumple la condición dada antes de crear el documento.
El contenido de dicho bucle es el siguiente:

```
if (!((CMioconApp*) AfxGetApp())->m_ClasificadorActivo)
{
AfxMessageBox("Debe activar la herramienta Espacio para operar con
 MioSong", MB_ICONEXCLAMATION);
return FALSE;
}
```

Si el bucle no se cumple, la función devuelve TRUE y se crea el nuevo documento.
A continuación, dado que se trata de un programa que va a trabajar de manera continua, tomando datos en tiempo real, se configuró el refresco de los datos.
La clase CMioSongDoc tiene que poder actualizar sus datos y sus vistas, para ello, se añadió una nueva función UpdateDoc destinada a esta tarea:

1. Se declaró en el archivo de cabecera esta nueva función (pública). No devuelve ningún valor y recibe como parámetro un objeto de la clase características.

   ```
   // Implementation
   public:
   void UpdateDoc(const CCaracteristicas* caract);
   ```

2. Se creó un objeto(m_Caracteristicas) de tipo const CCaracteristicas*(No cambia hasta que se vuelve a cargar el documento), que es utilizado por el documento y la clase view asociada:

   ```
   // Implementation
   public:
   ```

```
const CCaracteristicas* m_Caracteristicas;
```

3. Se añadió el cuerpo de la función al archivo *MioSongDoc.cpp*. En esta función se almacena el contenido que le llega como parámetro en el objeto m_Características. Además, se refrescan todas las vistas asociadas.

```
void CMioSongDoc::UpdateDoc(const CCaracteristicas* caract)
{
m_Caracteristicas=caract;
UpdateAllViews(NULL); // Las vistas actualizan sus datos
}
```

4. Se añadió al constructor de CMioSongDoc la inicialización de m_Caracteristicas:

```
CMioSongDoc::CMioSongDoc()
{
m_Caracteristicas=CCaracteristicas::Crear(ID_PROCESADO);
}
```

Para que el programa pase datos de manera continua a *MioSong*, se editó la función UpdateDocumentos deCMioconApp:

1. Se declaró un puntero de tipo CMioSongDoc (pSongDoc).

```
CMioSongDoc *pSongDoc;
```

2. Dentro del While, que comprueba el refresco de todos los documentos, a continuación de las sentencias if y else if, asociados al resto de herramientas, se añadió un else if que establece que, en caso de tratarse de un documento de tipo CMioSongDoc, pase a la función UpdateDoc de CMioSongDoc el objeto de la clase CCaracterísticas que se actualiza con las nuevas muestras que van llegando del sistema de adquisición. La función UpdateDoc guarda estas características en la variable local que utiliza para el tratamiento de las mismas y actualiza los datos de las vistas.
   El código añadido dentro del while de la función UpdateDocumentos de CMioconApp es el siguiente:

```
else if (tipoDoc=="MioSong")
{
if(m_ClasificadorActivo)
{
pSongDoc=(CMioSongDoc *)pDocument;
pSongDoc->UpdateDoc(m_Caracteristicas);
}
}
```

Llegados a este punto, se podía empezar a programar las diferentes funcionalidades teniendo en cuenta que, en cada refresco del documento, la función de la *clase vista* que se ejecuta es OnDraw.
En el archivo de cabecera de *MioSongDoc*, se declaró una variable de tipo int, que se usa para controlar la función que debe ejecutar la herramienta. Esta variable, deno-

minada m_Reproactiva, toma valores del 0 al 4. Este valor se actualiza al seleccionar una opción en el menú y se inicializa a 0 en el constructor:

- Declaración de la variable en *CMioSongDoc.h*:

```
// Implementation
public:
int m_Reproactiva;
```

- Código del constructor:

```
CMioSongDoc::CMioSongDoc()
{
m_Caracteristicas=CCaracteristicas::Crear(ID_PROCESADO);
int m_Reproactiva= 0;

}
```

- De acuerdo con la selección por parte del usuario, cada una de las funciones asociadas actualiza la variable con un valor. Para ello, fue necesario añadir primero un puntero de clase documento de *MioSong*. Por ejemplo, para la opción de *Reproducir*, la función queda de la siguiente manera:

```
void CMioSongView::OnHerramientasReproucir()
{

CMioSongDoc* pDoc = GetDocument();

pDoc->m_Reproactiva=1;
}
```

Para el resto de opciones, la función queda de manera equivalente, modificando m_Reproactiva con diferentes valores:

- Reproducir : m_Reproactiva=1.
- Reproducir bíceps: m_Reproactiva=2.
- Reproducir tríceps: m_Reproactiva=3.
- Entrenamiento: m_Reproactiva=4.
- Parar reproducción: m_Reproactiva=0.

Es en la función OnDraw donde, en función del valor de m_Reproactiva, el programa llama a diferentes funciones que realizan las tareas concretas. Para poder acceder a esta variable, se crea un puntero de tipo CMioSongDoc.

4.4.1. Función Reproducir

Se creó una función encargada de las tareas propias de *Reproducir*. Esta función se llama SonarTodo y recibe como parámetro un objeto de la clase CDC, que propor-

ciona funciones miembro para trabajar con un dispositivo de contexto, como una pantalla [10].
Para ello se siguieron los siguientes pasos:

1. Se declaró la función en el archivo de cabecera *MioSongView.h*

   ```
   void SonarTodo(CDC *pDC);
   ```

2. Se incluyó el esqueleto de la función en el archivo *MioSonView.cpp*

   ```
   void CMioSongView::SonarTodo(CDC *pDC)
   {
   }
   ```

3. Se añadió una llamada a GetDocument, que devuelve un puntero a la clase CMioSongDoc y permite acceder a los datos del documento.

   ```
   CMioSongDoc* pDoc = GetDocument();
   ```

4. Se declaró una variable de tipo *CString* y un vector de *float* de tamaño 2, para poder mostrar un mensaje por pantalla y almacenar dos valores numéricos, respectivamente.

   ```
   CString txt;
   float posicion[2];
   ```

5. Se almacenó en cada una de las dos posiciones del vector un valor comprendido entre 0 y 1. Este valor equivale a la diferencia entre la señal mioeléctrica de los músculos conectados a cada uno de los dos canales y la señal que se capturó de éstos en el calibrado. Para tomar estos valores, el puntero tipo *CMioSongDoc*, (pDoc), apunta al puntero m_Características y este a su vez a la variable m_distancia0 en primer lugar, y m_distancia1 en segundo, tomando así los valores del primer y segundo canal. Además, se añadió un truncaminto del tipo de datos para forzarlos a que sean de tipo *float*.

   ```
   posicion[0]=(float)(pDoc->m_Caracteristicas->m_distancia0);
   posicion[1]=(float)(pDoc->m_Caracteristicas->m_distancia1);
   ```

6. Se añadieron dos llamadas consecutivas a la función *Beep*, una función síncrona, que genera tonos simples por el altavoz de frecuencia y duración manipulables (se pasan como parámetros a la misma). Se pasa como parámetro de frecuencia el valor de posicion[0] y posicion[1], respectivamente, pero ambos multiplicados por 32767 (razonado en la sección 3.2.5 de este documento). La duración deseada es de 1 ms para ambos, por lo que el segundo parámetro que se pasa es un 1 en los dos casos.

   ```
   Beep(32767*posicion[0],1);
   Beep(32767*posicion[1],1);
   ```

7. Se añadió el código necesario para mostrar un mensaje por pantalla, almacenando, en primer lugar, el mensaje a mostrar en la variable txt (En este caso: Reproduciendo bíceps"). A continuación, para imprimir por pantalla, se añadió la función TextOut, a la que se le pasa como parámetros dos valores

numéricos(que corresponden a las coordenadas x e y donde se quiere mostrar el mensaje en la pantalla) y un tercer valor con el nombre de la variable en la que se almacena el *string* que se desea mostrar.

```
txt = "Reproduciendo ambos músculos";
pDC->TextOut(200, 100, txt);
```

El código completo se muestra a continuación:

```
void CMioSongView::SonarTodo(CDC *pDC)
{
CMioSongDoc* pDoc = GetDocument();
CString txt;
float posicion[2];
posicion[0]=(float)(pDoc->m_Caracteristicas->m_distancia0);
posicion[1]=(float)(pDoc->m_Caracteristicas->m_distancia1);

Beep(32767*posicion[0],1);
Beep(32767*posicion[1],1);

txt = "Reproduciendo ambos músculos";
pDC->TextOut(200, 100, txt);

}
```

El programa entra en esta función si la variable m_Reproactiva tiene el valor 1. Para comprobar esto, se añadió una sentencia *if* en la función OnDraw, tal y como se muestra a continuación:

```
if(pDoc->m_Reproactiva== 1){
SonarTodo(pDC);
}
```

4.4.2. Función Reproducir bíceps

La función que realiza las tareas propias de *Reproducir bíceps* es SonarBiceps(CDC* pDC). Para la declaración y el desarrollo de la misma, se siguieron pasos equivalentes a los de la funcionalidad anterior. Las diferencias, además del propio nombre de la función, son las siguientes:

1. En el paso 4, la variable de tipo *float* es un valor único y no un vector.
2. En el paso 5, únicamente se recoge el valor de m_distancia0.
3. En el paso 6, tan sólo hay una llamada a la función Beep, igual a la primera llamada del caso anterior.
4. Por último, la frase a mostrar y que, por tanto, se almacena en txt, dice: Reproduciendo bíceps"

El código completo te muestra a continuación:

```
void CMioSongView::SonarBiceps(CDC *pDC)
{
CMioSongDoc* pDoc = GetDocument();
CString txt;
float posicion;

posicion=(float)(pDoc->m_Caracteristicas->m_distancia0);
Beep(32767*posicion,1);

txt = "Reproduciendo bíceps";
pDC->TextOut(200, 100, txt);

}
```

El programa entra en esta función si la variable m_Reproactiva tiene el valor 2. Para comprobar esto, se añadió una sentencia *if* en la función OnDraw, tal y como se muestra a continuación:

```
if(pDoc->m_Reproactiva== 2){
SonarBiceps(pDC);
}
```

4.4.3. Función Reproducir tríceps

En el desarrollo de este punto, se siguieron pasos equivalentes a los del punto anterior. Las diferencias respecto al punto anterior son las siguientes:

1. La función, en este caso, recibe el nombre de SonarTriceps.
2. El valor que se recoge en la variable *float* es la de m_distancia1 (del segundo canal).
3. El texto que se almacena en txt, y se mostrará, por tanto, por pantalla, dice: Reproduciendo tríceps".

El código completo se muestra a continuación:

```
void CMioSongView::SonarTriceps(CDC *pDC)
{
CMioSongDoc* pDoc = GetDocument();
CString txt;
float posicion;

posicion=(float)(pDoc->m_Caracteristicas->m_distancia1);
Beep(32767*posicion,1);

txt = "Reproduciendo tríceps";
pDC->TextOut(200, 100, txt);
```

```
}
```

El programa entra en esta función si la variable m_Reproactiva tiene el valor 3. Para comprobar esto, se añadió una sentencia *if* en la función OnDraw, tal y como se muestra a continuación:

```
if(pDoc->m_Reproactiva== 3){
SonarTriceps(pDC);
}
```

4.4.4. Función Entrenamiento

La función encargada de llevar a cabo las tareas de *Entrenamiento* es SonarEntrenamiento(CDC* pDC). A la hora de declarar y completar esta función, se siguiron los pasos descritos a continuación:

1. Se declaró la función en el archivo de cabecera *MioSongView.h*

   ```
   void SonarEntrenamiento(CDC *pDC);
   ```

2. Se incluyó el esqueleto de la función en el archivo *MioSonView.cpp*

   ```
   void CMioSongView::SonarEntrenamiento(CDC *pDC)
   {
   }
   ```

3. Se añadió una llamada a GetDocument.

   ```
   CMioSongDoc* pDoc = GetDocument();
   ```

4. Se declararon dos variables de tipo *CString*(txt y txt1), para poder mostrar dos posibles mensajes por pantalla; un vector de *float* de tamaño 2(float posicion[2]) y una variable de tipo *int* .

   ```
   CString txt;
   CString txt1;
   float posicion[2];
   int num;
   ```

5. Se incluyeron al inicio del archivo *CMioSong.cpp* las cabeceras de las librerías *stdlib* y *time*, que permiten generar un número aleatorio.

   ```
   #include<stdlib.h>
   #include<time.h>
   ```

6. Se añadió una llamada a la función srand(time(NULL)), que fija una semilla partir de la cual se genera el número aleatorio.

   ```
   srand(time(NULL));
   ```

7. Se añadieron las líneas que cargan, como variable *float*, los valores de m_distancia0 y m_distancia1 en los los elementos del vector posición.[Explicación desarrollada en el punto 5 de 4.4.1]

```
posicion[0]=(float)(pDoc->m_Caracteristicas->m_distancia0);
posicion[1]=(float)(pDoc->m_Caracteristicas->m_distancia1);
```

8. Se incluyeron en el archivo de cabecera, *MioSongDoc.h*, dos nuevas variables (m_RefrescoEntrenamiento y m_NumAnterior), que sirven para controlar cuándo se vuelve a generar un número aleatorio y para, en caso de no generarse, mantener el anterior, respectivamente.

```
int m_RefrescoEntrenamiento;
int m_NumAnterior;
```

9. Se añadió en el constructor de *CMioSongDoc* una inicialización de las dos variables anteriores a 0, de forma que este sea el valor que toma al iniciarse el documento.

```
int m_RefrescoEntrenamiento=0;
int m_NumAnterior=0;
```

10. Se incluyeron las mismas librerías que en el paso 5 en el archivo *MioSongDoc.cpp*

11. Se añadió una llamada a la función srand(time(NULL)) (de igual forma que en el paso 6) dentro del constructor de CMioSongDoc, para poder inicializar m_NumAnterior con un número realmente aleatorio.

12. Se incorporó la generación de un número aleatorio que puede tomar valores 0 y 1 y cuyo valor se almacena en m_NumAnterior.

```
srand(time(NULL));
m_NumAnterior=rand()%2;
```

13. Se incluyeron en el archivo de cabecera, *MioSongDoc.h*, dos nuevas variables (m_Aciertos y m_Fallos), que sirven para controlar el número total de aciertos y errores que se acumulan en la ejecución del programa.

```
int m_Aciertos;
int m_Fallos;
```

14. Se añadió en el constructor de *CMioSongDoc* una inicialización de las dos variables anteriores a 0, de forma que este sea el valor que toma al iniciarse el documento.

```
int m_Aciertos=0;
int m_Fallos=0;
```

15. Se implementó una sentencia *if* que comprueba si la variable m_RefrescoEntrenamiento tiene un valor mayor que 10, lo que significaría que *Entrenamiento* lleva 11 actualizaciones de datos actuando en el mismo modo y es hora de volver a generar el número aleatorio. Por este motivo, en el interior de la sentencia condicional,

se genera un número aleatorio (haciendo uso de la función rand) que, almacenado en la variable num, tomará un valor 0 o 1. Además, el contenido de la variable num, actualiza la variable de *CMioSongDoc*, m_NumAnterior.

```
if(pDoc->m_RefrescoEntrenamiento>10){//Deja 11 actualizaciones hasta cambiar

num=rand()%2;

pDoc->m_NumAnterior=num;
pDoc->m_RefrescoEntrenamiento=0;
}
```

16. Se añadió una sentencia por defecto (else), para que en caso de no llevar 11 actualizaciones, la variable num tome el valor almacenado en m_NumAnterior y se actualice el valor de la variable de control m_RefrescoEntrenamiento

17. Se introdujo una primera sentencia *if* destinada a controlar si el número generado es un 1 o un 0. En este caso, el programa entra si en la variable num hay un 0.

    a) Se añadió el código necesario para que se muestre por pantalla la opción de la que se trata. Consistió en almacenar en la variable txt la frase 'Haz que predomine el bíceps' y ,a continuación, incluir una llamada a TextOut, que pasa esta variable y las coordenadas donde se quiere mostrar la frase, como parámetros.

    ```
      if(num==0){

    txt = "Haz que predomine el bíceps";
    pDC->TextOut(100, 100, txt);
    }
    ```

    b) Se incorporó una sentencia *if* que comprueba si se cumple la condición del programa (en este caso, dominancia del bíceps). Entra en la sentencia si el valor del primer canal es superior al del segundo.

    ```
    if(posicion[0]>posicion[1]){
    }
    ```

    c) Se añadió el código que gestiona los eventos en caso de entrar en el bucle anterior. En la variable txt1 se almacena el mensaje 'Correcto', se muestra este mensaje por pantalla con TextOut, se pasa el valor del primer canal a la función Beep (multiplicado por 32767, como se explica en los casos anteriores) y se actualiza el valor de m_Aciertos, que aumenta su valor en 1.

    ```
    if(posicion[0]>posicion[1]){

    txt1="Correcto";
    pDC->TextOut(100, 50, txt1);
    Beep(32767*posicion[0],1);
    ```

```
(pDoc->m_Aciertos)=1+(pDoc->m_Aciertos);
}
```

d) Se configuró una sentencia por defecto, else, en la que entra en caso de no cumplir la condición anterior. Al entrar en esta sentencia, en la variable txt1 se almacena el mensaje 'Error', se muestra este mensaje por pantalla con TextOut y se actualiza m_Fallos, añadiéndole 1.

```
else{
txt1="Error";
pDC->TextOut(100, 50, txt1);
(pDoc->m_Fallos)=1+(pDoc->m_Fallos);
}
```

18. Se introdujo una segunda sentencia *if* destinada a controlar si el número generado es un 1 o un 0. En este caso, el programa entra si en la variable num hay un 1. Los pasos seguidos son equivalentes a los relatados para el ocaso de num =0, con las diferencias que se indican a continuación:

    a) En el paso a), la frase contenida en txt es 'Haz que predomine el tríceps'.

    b) En el paso b), la condición de la sentencia if será la opuesta. Se cumple si hay dominancia del tríceps (posicion[0] » posicion[1]).

    c) En el paso c), el parámetro que se pasa a la función Beep es 32767 veces el valor del segundo canal (posicion[1]).

El código completo se muestra a continuación:

```
void CMioSongView::SonarEntrenamiento(CDC *pDC)
{
CMioSongDoc* pDoc = GetDocument();
CString txt;
CString txt1;
float posicion[2];
int num;
srand(time(NULL));

posicion[0]=(float)(pDoc->m_Caracteristicas->m_distancia0);
posicion[1]=(float)(pDoc->m_Caracteristicas->m_distancia1);

if(pDoc->m_RefrescoEntrenamiento>10){//Deja 11 actualizaciones hasta cambiar

num=rand()%2;

pDoc->m_NumAnterior=num;
}

else{
```

```
num=pDoc->m_NumAnterior;
++(pDoc->m_RefrescoEntrenamiento);
}

if(num==0){

txt = "Haz que predomine el bíceps";
pDC->TextOut(100, 100, txt);

if(posicion[0]>posicion[1]){

txt1="Correcto";
pDC->TextOut(100, 50, txt1);
Beep(32767*posicion[0],1);
(pDoc->m_Aciertos)=1+(pDoc->m_Aciertos);
}

else{
txt1="Error";
pDC->TextOut(100, 50, txt1);
(pDoc->m_Fallos)=1+(pDoc->m_Fallos);
}

}



if(num==1){

txt = "Haz que predomine el tríceps";
pDC->TextOut(100, 100, txt);

if(posicion[0]<posicion[1]){

txt1="Correcto";
pDC->TextOut(100, 50, txt1);
Beep(32767*posicion[0],1);
(pDoc->m_Aciertos)=1+(pDoc->m_Aciertos);
}

else{
txt1="Error";
pDC->TextOut(100, 50, txt1);
(pDoc->m_Fallos)=1+(pDoc->m_Fallos);
}
```

```
}
}
```

Por último, se añadieron algunas modificaciones en la función OnDraw para gestionar este modo:

1. Se declararon dos variables CString (textA y textB) y dos de tipo int (acierto y error).
2. El programa entra en esta función si la variable m_Reproactiva tiene el valor 4. Para comprobar esto, se añadió un *if*.

   ```
   if(pDoc->m_Reproactiva==4){

   SonarEntrenamiento(pDC);
   }
   ```

3. Se incluyó, dentro de la sentencia, el código necesario para que se muestre por pantalla el porcentaje de acierto que se acumula mientras se ejecuta el modo *Entrenamiento*.
   a) Se introdujo el código necesario para mostrar por pantalla el mensaje 'Porcentaje de acierto: ', con el mismo método que se ha relatado en los puntos anteriores. Con esto el programa mostraba el mensaje, pero aún no había incluido un código que calculara este valor.

      ```
      txtA = "Porcentaje acierto:";
      pDC->TextOut(300, 100, txtA);
      ```

   b) Se añadió el código del cálculo del porcentaje de aciertos. Se divide m_Aciertos entre la sum de m_Aciertos y m_Fallos, se multiplica por 100 y se almacena en la variable acierto.

      ```
      acierto=100*(pDoc->m_Aciertos)/((pDoc->m_Aciertos)+(pDoc->m_Fallos));
      ```

   c) Se implementó una sentencia *if* para que en caso de que ambas variables (m_Aciertos y m_Fallos) valgan 0, acierto tome directamente el valor 0.

      ```
      if((pDoc->m_Aciertos == 0)&&( pDoc->m_Fallos ==0)){
      acierto=0;
      }
      ```

   d) Se añadió el código que permite configurar el formato de la variable acierto para poder ser pasada a la función TextOut, la cual se añadió a continuación.

      ```
      txtA.Format("%i", acierto);
      pDC->TextOut(500, 100, txtA);
      ```

4. Se introdujo dentro de la sentencia el código necesario para que se muestre por pantalla el porcentaje de error que se acumula mientras se ejecuta el modo *Entrenamiento*. El procedimiento llevado a cabo es equivalente al anterior, con las diferencias que se indican a continuación:
   a) textA se sustituye por textE.

b) El mensaje del paso a) es 'Porcentaje de error:'.

c) Siempre que aparece la variable m_Aciertos se sustituye por m_Fallos y viceversa.

d) La variable acierto se sustituye por error.

El código completo del bucle dentro de OnDraw se muestra a continuación:

```
if(pDoc->m_Reproactiva==4){

SonarEntrenamiento(pDC);

txtA = "Porcentaje acierto:";
pDC->TextOut(300, 100, txtA);

acierto=100*(pDoc->m_Aciertos)/((pDoc->m_Aciertos)+(pDoc->m_Fallos));

if((pDoc->m_Aciertos == 0)&&( pDoc->m_Fallos ==0)){
acierto=0;

}
txtA.Format("%i", acierto);
pDC->TextOut(500, 100, txtA);

txtE = "Porcentaje de errores:";
pDC->TextOut(300, 120, txtE);
error=1+(100*(pDoc->m_Fallos)/((pDoc->m_Aciertos)+(pDoc->m_Fallos)));

if((pDoc->m_Aciertos == 0)&&( pDoc->m_Fallos ==0)){

error=0;
}
txtE.Format("%i", error);
pDC->TextOut(500, 120, txtE);

}
```

4.4.5. Función Parar reproducción

La función de *Parar reproducción* no tiene ninguna función asociada, ya que al actualizar el valor de m_Reproactiva y ponerlo a 0, hace que el programa no pueda entrar en los otros modos sin necesidad de añadir código extra.

4.5. Problemas surgidos en el desarrollo

Pese a que en el presente documento los pasos llevados a cabo se relatan de forma directa y sin cambios de 'dirección', a la hora de desarrollar este trabajo se probaron diversas opciones hasta llegar al método final. Uno de los mayores problemas que se encontraron fue concretar cómo hacer que las muestras llegasen a ser reproducidas. En un primer momento, se pretendía pasar los datos directamente a la tarjeta de sonido. Para ello, parecía necesario crear primero un pequeño archivo *Wav* que contuviese los datos y que sería el elemento que realmente se pasara a la tarjeta de sonido.
Se desarrolló una clase auxiliar a la clase *CMioSongDoc*, a la que se llamó CWav. Ésta contenía la estructura necesaria para crear una cabecera de un archivo *.Wav*. A continuación, haciendo uso de esta clase, se creaba un pequeño archivo *Wav* en la función OnDraw de *CMioSongView*. En este archivo se almacenaba una muestra de datos a reproducir y, posteriormente, se pasaba a la tarjeta de sonido mediante la función PlaySound. Esto no resultaba efectivo ni eficiente. En primer lugar, las lecturas y escrituras constantes en el disco suponen un desgaste elevado para el mismo, lo que a la larga reduce su tiempo de vida útil. En segundo lugar, no resulta eficiente crear todo un archivo para pasar una única muestra (aunque ésta estuviera repetida un número x de veces para que llegara a ser audible), ya que la cabecera tiene un tamaño mucho mayor en comparación. Por otro lado, se contempló la opción de pasar un bloque de muestras, pero esto haría que se perdiese la característica de trabajo 'en tiempo real'.
Por los motivos expuestos anteriormente, finalmente se desechó la idea de trabajar con la función PlaySound y, tras consultar varias fuentes en relación a este tema, se llegó a la conclusión de que la opción más viable, como una primera aproximación a la idea final, era hacer uso de la función *Beep*, opción finalmente utilizada.

Capítulo 5

Resultados de la aplicación

Una vez desarrollado el programa, se realizó una prueba a varios sujetos para comprobar la utilidad de la herramienta *MioSong*. Para ello, se utilizó el modo *Entrenamiento* que ofrece esta herramienta.

5.1. Protocolo

Se realizaron 4 pruebas de 1 minuto a cada sujeto. En esta prueba, el participante debía intentar realizar un mayor esfuerzo con el músculo que le indicaba el programa en cada momento (aleatorio). Al finalizar el minuto, se anotaba el porcentaje de acierto y error que acumulaba.
Pasados unos minutos (siempre más de 5), se repetía la prueba de la misma manera, así hasta realizar las 4 pruebas necesarias.
Durante la realización de la prueba, el sujeto debía permanecer sentado y en una posición cómoda, con los electrodos colocados en el brazo dominante.

5.2. Pruebas

Se realizaron las pruebas a un total de cinco sujetos sanos. Ninguno de ellos había tenido un contacto previo con la herramienta.
En la Figura 5.1 se muestra una gráfica con los porcentajes de acierto que se registraron al finalizar las cuatro pruebas realizadas a cada uno de ellos. Por otro lado, la gráfica de la Figura 5.2 recoge los porcentajes de error anotados en las mismas pruebas.

Se observa que hay una tendencia de mejora con las pruebas, tal y como se confirma en la Figura 5.3, que representa el promediado de aciertos y errores de todos los sujetos en cada una de las pruebas. En estas gráficas se ve como el promediado de errores desciende, aumentando por tanto el de acierto. Esta tendencia descendente del error se debe a un pequeño aprendizaje del sujeto, no sólo del uso de la herramienta, si no también del control de sus propias señales. No obstante, durante la

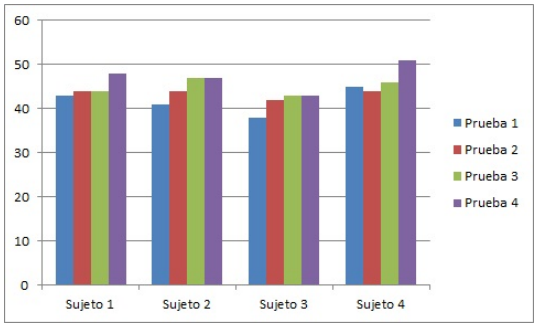


Figura 5.1: Porcentajes de acierto registrados en las pruebas.

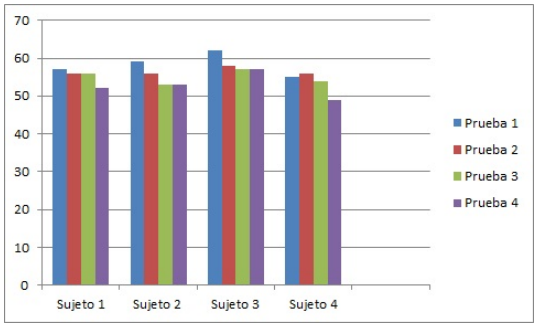


Figura 5.2: Porcentajes de error registrados en las pruebas.

realización de las pruebas, se observó que los sujetos tenían mayores dificultades cuando la herramienta esperaba la dominancia del tríceps, ya que en la mayoría de ocasiones se producía un aumento simultáneo de los esfuerzos realizados por ambos músculos (haciendo uso de la herramienta *Espacio*).

Pese a observarse una leve mejora en los resultados a medida que se realizaban las pruebas, el porcentaje medio de error de todas las pruebas y sujetos es del 55,62 %, frente al 44,38 % que se acumula de acierto. Esto puede deberse a que esta herramienta requiere de un mayor control de la propia señal, ya que no sólo exige incrementar el esfuerzo puntual con uno de los músculos, si no que además este incremento debe realizarse de manera aislada, es decir, intentando que el esfuerzo

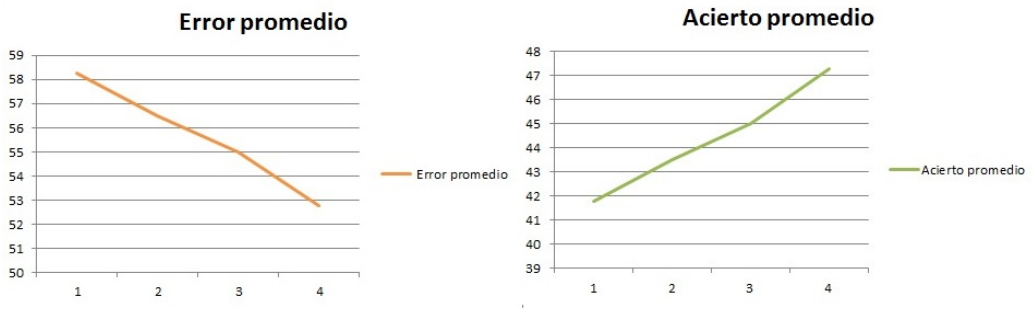


Figura 5.3: promediado de los porcentajes de acierto y error registrados en las pruebas a los sujetos sanos.

en el antagonista se mantenga o incluso disminuya.

5.3. conclusiones

Al finalizar las pruebas se realizaron una serie de preguntas a los participantes: todos ellos coincidieron en que el esfuerzo del tríceps resultaba más difícil de controlar, la mitad de ellos dijeron haber encontrado la prueba de dificultad elevada; sin embargo, todos volvieron a coincidir en que resultaba interesante y entretenido.
A partir de estas cuestiones y de los resultados de las pruebas, se han podido sacar pequeñas conclusiones en relación a diferentes aspectos relacionados con la herramienta.
En primer lugar, parece presentarse mayor dificultad en el control del esfuerzo del tríceps. Esto puede deberse a una cuestión de entrenamiento y práctica, ya que la mayoría de nosotros es capaz de concentrarse en realizar una contracción aislada del bíceps y se hace de manera más o menos habitual. Sin embargo, para contraer el tríceps, la mayoría necesitamos ayudarnos de un elemento externo, una superficie sobre la que hacer fuerza.
Por otro lado, en cuanto a la herramienta, parece interesante incluir diferentes niveles de dificultad. Estos niveles podrían depender del tiempo que se mantiene la herramienta en cada uno de los modos aleatorios que exigen la dominancia de uno de los músculos. De esta forma, aunque los resultados de todos los sujetos han sido similares, se conseguiría evitar una sensación de frustración que puede llegar a sentir alguno de los sujetos que afirman que la herramienta tiene una elevada dificultad.

Capítulo 6

Otros trabajos realizados

De manera paralela al desarrollo de la aplicación, he participado en algunas tareas que se han llevado a cabo en el laboratorio de Electrónica y Bioingeniería de la E.S.T de Ingenieros de Telecomunicación de la UVa.Una de estas actividades ha consistido en la creación de un protocolo de pruebas y la realización de las mismas con el dispositivo TactileCom Belt, desarrollado por miembros del laboratorio citado y cuyos datos buscaban ser incluidos como parte del artículo presentado por el Doctor Alonso Alonso Alonso al congreso DRT4ALL de tecnología y turismo para todas las personas, 2017.

6.1. TactileCom Belt

Este sistema trabaja con un cinturón vibratorio, un panel numérico de 4 filas y 3 columnas, que se sitúa bajo 5 leds que darán información de 5 actuadores (colocados en el cinturón), y un dispositivo Android,donde se instalará la aplicación, y el cual servirá como medio de comunicación entre personas con discapacidad auditiva y visual y sujetos sanos.
Se trata de un sistema que permite una comunicación bidireccional entre la persona que lleva el cinturón y el panel numérico y la aplicación Android. Por un lado, la pulsación de los diferentes botones del panel numérico permite el envío de comunicaciones hacia el dispositivo Android. Por otro lado, el camino de comunicación inverso se realizará mediante vibraciones en el cinturón que estarán asociadas a los comandos que aparecen en la pantalla principal de la aplicación Android. Las vibraciones enviadas serán de frecuencias 1Hz, 10Hz o continua(0Hz), pudiendo estas ser modificadas a nivel hardware.
El funcionamiento detallado con todas las posibilidades ofrecidas se recoge en el manual extendido incluido en el anexo al final del presente documento.

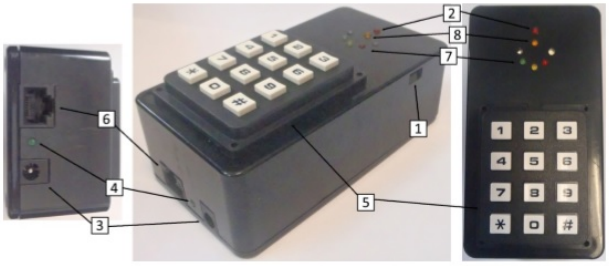


Figura 6.1: Apariencia del teclado asociado al cinturón estimulador

6.1.1. Descripción hardware

Teclado y leds de información

1. Interruptor de encendido.
2. Led indicador de encendido.
3. Conector jack de alimentación 12V.
4. Led indicador de carga.
5. Teclado de 4 filas y 3 columnas (Figura 6.1).
6. Conector RJ-45 para conexión de los actuadores.
7. Leds indicadores de orden enviada.
8. Led indicador de segunda pulsación habilitada.

Una vez que se enciende el dispositivo, automáticamente se indicará con el led correspondiente (2). En estado de reposo, el resto de leds permanecen apagados.
El teclado se utiliza como interfaz aferente por el usuario del sistema TactileCom. Se entiende como interfaz aferente un dispositivo encargado de captar señales de una determinada naturaleza y llevarlas hacia un núcleo central de un sistema electrónico[12]. En este caso, el teclado envía una orden determinada mediante Bluetooth al dispositivo Android vinculado.

Cinturón estimulador

La apariencia del cinturón se puede observar en la Figura 6.2, donde se aprecian también los 5 actuadores, responsables directos de la comunicación con el sujeto.

El cinturón actúa como interfaz eferente y se conecta al controlador mediante un cable RJ45. Se entiende como interfaz eferente a aquella que sintetiza una serie de señales en el núcleo de un dispositivo y las trasmite, de alguna forma, hacia su

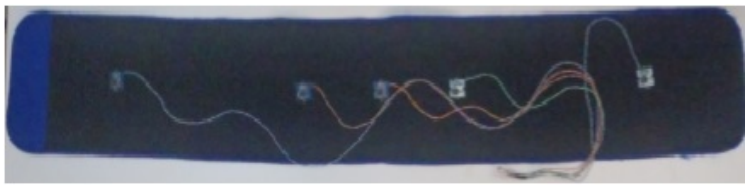


Figura 6.2: Apariencia de la parte interior del cinturón estimulador.

Figura 6.3: Apariencia de la pantalla principal de la aplicación

exterior [12]. Será el controlador el encargado de codificar la estimulación que corresponda en función del dato recibido por Bluetooth.

6.1.2. Descripción software

La aplicación para Android de TactileCom está diseñada para permitir la comunicación bidireccional entre el dispositivo móvil y el cinturón. Ésta muestra una interfaz gráfica en la que los comandos aparecen como iconos que se pueden pulsar. Cada uno de estos iconos lleva asociado un comando que será enviado al cinturón (codificado mediante los caracteres 0-9 y A-Z). Las estimulaciones correspondientes a cada comando se explican en el anexo con las instrucciones detalladas.
Tal y como se puede observar en la parte inferior de la Figura 6.3, la interfaz dispone también de una entrada de comandos por teclado o por voz. En este caso se hace el reconocimiento del texto insertado y se determina si el comando es válido o no. Si lo es, se envía a la interfaz eferente.

6.2. Descripción del protocolo de pruebas

En primer lugar se estableció cuál era el objetivo final del dispositivo, que buscaba enviar diferentes patrones de vibración discriminables por el sujeto portador del cinto. Por tanto, las pruebas consistirían en enviar una serie de patrones seleccionados previamente y comprobar si el portador del cinturón era capaz de distinguir qué

actuadores se encontraban activos con cada patrón y la frecuencia de vibración de los mismos.
El siguiente paso consistió en buscar la posición que debía mantener el sujeto sometido a las pruebas, de forma que interfiriera lo menos posible en la respuesta del mismo. El cinturón se situó tal y como se sugiere en las instrucciones detalladas del dispositivo. En un primer momento se propuso un cambio de orientación, colocando tres actuadores en la espalda y dos en el abdomen, posicionados ligeramente por encima de la altura del ombligo. Sin embargo, tras realizar una serie de pruebas, se concluyó que, no sólo no suponía una mejora, si no que además en ciertas ocasiones dificultaba la distinción de algunas frecuencias. Por tanto, la localización final del cinturón que se aplicó en las pruebas, corresponde a la representada en la Figura 6.4, quedando éste con los actuadores unos dos centímetros por encima de la altura del ombligo del sujeto y orientado de forma que tres de éstos se encuentren en la parte delantera y dos en la zona de la espalda.
El sujeto, durante la realización de la prueba, debía mantenerse de pie en una posición neutra, intentando equilibrar lo máximo posible el peso que recaía sobre ambos pies, con la espalda erguida y los brazos en una posición que, aunque relajada, debía evitar el contacto total con el tronco. De esta forma se podía evaluar el nivel real de capacidad de discriminación estando el cinturón únicamente en contacto con el abdomen y la espalda. El teclado debía ser sostenido por el propio sujeto, pero orientado hacia abajo. Con esta orientación, los leds situados en la parte delantera del teclado no darían pistas al sujeto.
Finalmente, a la hora de realizar las pruebas,se eligieron participantes sin entrenamiento previo con el sistema, se informó a los mismos de la finalidad de las pruebas y se les colocó la interfaz eferente con los 5 estimuladores. Como se ha explicado, cada estimulador puede reproducir 3 tipos de excitación: 1Hz, 10Hz, operación continua (0Hz), o bien puede permanecer inactivo. Se diseñó una tabla de mensajes que recoge 20 combinaciones de estimulación de diversa dificultad y que se incluye a continuación, en la Figura 6.5. En esta tabla los colores se relacionan con el color indicativo de cada actuador del cinturón, que se encuentran representados por los leds sobre el teclado.[TI:Trasero izquierza ,DI:Delantero izquierda, C:Centro, DD: Delantero derecha, TD: Trasero derecha].
Las pruebas se realizaron en dos fases. En una primera fase se enviaban al sujeto los patrones en el orden en que se encuentran en la tabla, que busca ir de menor a mayor dificultad; en la segunda fase, se enviaban exactamente los mismos patrones pero en diferente orden (para todos el mismo) y esperando entre fases un mínimo de 15 minutos. En caso de fallo, se anotaba la respuesta errónea percibida y el patrón era enviado una segunda vez, si el fallo persistía, se anotaba de nuevo y volvía a repetirse la acción de enviar ese patrón por tercera y última vez. En caso de darse un tercer fallo, se anotaba de nuevo la respuesta y se informaba del error al sujeto. A continuación, se procedía a enviar el siguiente patrón de la tabla.

6.2.1. Resultados y conclusiones

(Los siguientes resultados y conclusiones se recogen en [17].) Se sometieron a la prueba un total de 15 sujetos de control, todos ellos individuos sanos, que representaban

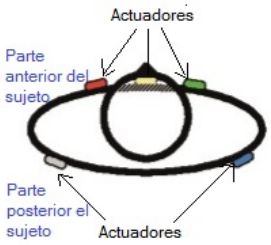


Figura 6.4: Vista esquematizada desde arriba de la localización final del cinturón en el sujeto portador.

PATRÓN	TI	DI	C	DD	TD
1				1Hz	
2	1Hz				
3					10Hz
4			10Hz		
5				0Hz	
6			0Hz		
7	1Hz				1Hz
8		1Hz		1Hz	
9			1Hz	1Hz	
10			10Hz		10Hz
11		10Hz		10Hz	
12			10Hz	10Hz	
13			0Hz	0Hz	
14		0Hz		0Hz	
15	0Hz				0Hz
16	10Hz		1Hz		
17			1Hz	10Hz	
18	1Hz	1Hz	1Hz	1Hz	1Hz
19	10Hz	10Hz	10Hz	10Hz	10Hz
20	0Hz	0Hz	0Hz	0Hz	0Hz

Figura 6.5: Tabla con los patrones seleccionados para la realización de las pruebas.

Figura 6.6: Aciertos totales y fallos comparando las dos pruebas[Gráfico sacado de [17]]

Figura 6.7: Aciertos desglosados para cada patrón en cualquiera de los tres posibles intentos[Gráfico sacado de [17]]

a ambos sexos y a muy diversas complexiones. A partir de las dos tablas de datos recogidos de cada sujeto se han sacado una serie de conclusiones que forman parte del artículo citado anteriormente. Para sacar estas conclusiones se ha hecho uso de una serie de gráficos presentados a continuación.
En primer lugar se realizó un gráfico que representa el número de aciertos y fallos totales en ambas pruebas,y el cual se puede observar en la Figura 6.6. Se aprecia que hay un total de 491 aciertos en el primer intento, un 82 % sobre el total de los intentos. Por otra parte, se registraron 52 fallos tras 3 oportunidades, lo que representa un 8,6 %. Si tenemos en cuenta que, según la Figura 6.7, los fallos se concentran en 3 patrones 18, 19 y 7, que en la práctica podrían ser sustituidos por otros más sencillos, los resultados mejorarían sensiblemente. Además, la imagen 8 muestra que 11 de los 52 fallos se concentran en el participante 1, probablemente debido a su constitución endomorfa (con elevado porcentaje de grasa), que provocaría una peor percepción del efecto de los estimuladores a través de la capa grasa.

Por otro lado, a la hora de analizar la capacidad de aprendizaje con el sistema, podemos apoyarnos en la Figura 6.6, en la que se observa que, pese a que el porcentaje total de aciertos en el primer intento es prácticamente el mismo para ambos

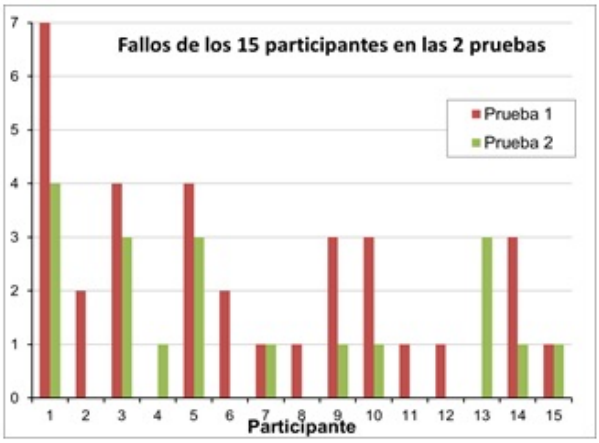


Figura 6.8: Fallos por cada participante en las dos pruebas [Gráfico sacado de [17]]

experimentos, el porcentaje total, que se puede observar en las dos columnas de la izquierda de esta figura. De la misma forma, se puede apoyar esta teoría a partir de la Figura 6.8, en la que se representa el número de fallos y aciertos de cada participante en cada experimento, y se puede observar que en ningún caso de produce un mayor número de errores en el segundo experimento, éste mejora o se mantiene constante.

La dificultad relativa de identificación de cada patrón puede estudiarse a partir de la Figura 6.7, con la que podemos concluir que hay patrones mas difíciles de distinguir que otros, como pueden ser el 18 y el 19, que acumulan un mayor número de fallos o el patrón 7, que acumula un total de 5 fallos. En los dos primeros casos puede deberse a que al actuar los 5 estimuladores a la vez se dificulta la discriminación espacial por separación física, haciendo que algunos actuadores queden camuflados. En el caso del patrón 7, puede deberse a una mayor dificultad que parece observarse para diferenciar las vibraciones a 1Hz.
Por último, también entra en juego la diferente capacidad de percepción de cada uno de los 15 sujetos, que es una pequeña muestra de la población, y nos permite hacernos una idea de que cada sujeto tiene una habilidad preexistente diferente en relación al sistema, ya que como se mencionó anteriormente ninguno había recibido un entrenamiento previo y, como podemos observar en la Figura 6.8, es muy diferente por ejemplo entre el primer sujeto y sujetos como el 2 o el 4.

Capítulo 7

Conclusiones

El objetivo principal de este trabajo de fin de grado consistía en añadir una nueva funcionalidad a la plataforma UVa-NTS de adquisición y entrenamiento en tiempo real de señales neuromusculares. A lo largo de este documento se ha incluido el proceso de desarrollo e implementación de esta nueva funcionalidad, la herramienta *MioSong*.

En cuanto al funcionamiento de la herramienta, una vez realizada una pequeña prueba y una serie de preguntas a los sujetos sometidos al experimento, se pudieron sacar algunas conclusiones. En primer lugar, se concluyó que el funcionamiento de la herramienta era satisfactorio. Para comprobar el correcto funcionamiento, se trabajó con la ventana de la herramienta *Espacio* abierta en paralelo a la de *MioSong*. De esta forma se comprobó que la herramienta desarrollada actúa con coherencia en base a los resultados esperados. Además, la herramienta resultó de interés y entretenida a todos lo usuarios de la misma.

Por otro lado, según las experiencias de los sujetos sometidos a las pruebas, parece necesario añadir diferentes niveles de dificultad. Esto se debe a que la habilidad preexistente por parte de cada sujeto es distinta y, fundamentalmente al principio, se debería adecuar la dificultad a cada sujeto para evitar una sensación de frustración que pueda hacerle perder el interés por el uso de la herramienta.

En cuanto al control de las señales mioeléctricas, se concluyó que, en general, resulta de mayor dificultad dificultad el control de los esfuerzos realizados por el tríceps. Debido a esto, parece razonable pensar que una disminución en el porcentaje de tiempo en que la aplicación solicita la predominancia del tríceps frente al bíceps, produciría una mejora en los porcentajes de acierto recogidos en las pruebas.

Por último, en cuanto al trabajo de desarrollo de un protocolo de pruebas para un prototipo (TactileCom Belt), se pudo concluir que el procedimiento desarrollado era correcto. Por tanto, a partir de los datos recogidos, se pudieron sacar una serie de conclusiones. Estas conclusiones se recogen en el punto 6.2.1 y están incluidas en [17].

Capítulo 8

Lineas futuras

El objetivo final de todo el sistema, en el ámbito del entrenamiento, consiste en alcanzar una herramienta que permita controlar los avances del sujeto, así como mantenerle motivado en la práctica, lo que podría derivar en una mejora más rápida. Por esto, aunque el sistema completo se compone de la plataforma UVA-NTS y el programa *Miocon*, este trabajo se ha centrado en complementar la parte software, el programa *Miocon* .

Como se comenta al principio de este documento, la implementación de la herramienta *MioSong* pretende ser un primer paso hacia un juego mucho más elaborado y con una estética más llamativa, lo que le aporta un valor añadido y motiva su utilización en el ámbito del entrenamiento, enfocado al uso de prótesis.

En cuando a investigaciones futuras, un primer punto podría consistir en añadir una parte gráfica que resulte más visual, representando, por ejemplo, las muestras en un pentagrama y haciendo coincidir, mediante cuantificación, cada movimiento con una nota que podría ser superpuesta sobre una canción de entre un conjunto. Este juego tendría unas características similares al ya conocido $SingStar^{TM}$[16] y podría suponer un gran cambio en el interés hacia el entrenamiento enfocado al uso de prótesis mioeléctricas.

Por otro lado, sería interesante conseguir volcar las muestras directamente a la tarjeta de sonido, sin necesidad de hacer uso de la función *Beep*, realizando, eso sí, una cuantificación previa que permitiese tener cierto control de los sonidos que se van a producir.

Por último, resultaría interesante realizar un estudio del rendimiento del programa. De esta forma se podría concretar el número óptimo de actualizaciones de las características que debe esperar el programa antes de calcular de nuevo el número aleatorio que controla la función *Entrenamiento*. Actualmente, el establecimiento de un refresco del número aleatorio cada 11 actualizaciones, se ha establecido mediante análisis experimental, de forma que el nivel de dificultad resultara moderado. Sin embargo, se podría tratar de establecer una relación numérica que permitiese el establecimiento de este valor en base a unos datos concretos.

Anexo A

Instrucciones tactileCom-Belt

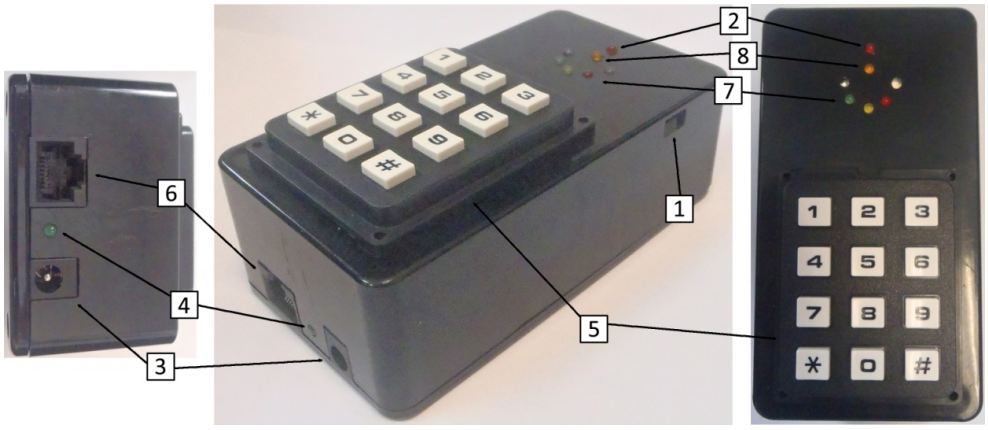
A continuación, a lo largo de este anexo, se incluyen las instrucciones completas del dispositivo TactileCom Belt utilizado en el desarrollo de las pruebas explicadas a lo largo del capítulo 6.

¿Qué hace este sistema?

Este sistema trabaja con un cinturón vibratorio, un panel numérico de 4 filas y 3 columnas y un dispositivo Android, donde se instalará la aplicación, y servirá como medio de comunicación para personas con discapacidad auditiva y visual. Con este sistema, se podrá realizar una comunicación bidireccional entre la persona que lleva el cinturón y el panel numérico y la aplicación Android. Por un lado, la pulsación de los diferentes botones del panel numérico permite el envío de comunicaciones hacia el dispositivo Android. Por otro lado, el camino de comunicación inverso se realizará mediante vibraciones en el cinturón que estarán asociadas a los comandos que aparecen en la pantalla principal de la aplicación Android. Además, este sistema incluye otras funciones adicionales como la notificación de llamadas perdidas de Skype™ y la posibilidad de configurar 9 direcciones para ir tanto a pie como en coche. Estas direcciones serán accesibles desde el panel numérico y tras su elección se abrirá la aplicación de Google Maps para realizar la ruta desde el punto donde se encuentre el usuario, hasta el destino elegido.

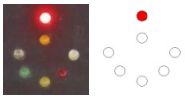
HARDWARE

Aspecto externo



1. Interruptor de encendido
2. Led indicador de encendido
3. Conector jack de alimentación 12 V
4. Led indicador de carga
5. Teclado de 4 filas y 3 columnas
6. Conector RJ-45 para conexión de los actuadores
7. Leds indicadores de orden enviada
8. Led indicador de segunda pulsación habilitada

Una vez que se enciende el dispositivo automáticamente se indicará con el led correspondiente (2). En estado de reposo, el resto de leds permanecen apagados.



Teclado

Utilizado como **interfaz aferente** por el usuario del sistema TactileCom, ésta envía una orden determinada mediante Bluetooth al dispositivo Android vinculado, por simplicidad las órdenes enviadas se corresponden con un único carácter. La orden puede ser de una pulsación o de dos como combinación de una pulsación previa de las teclas *, 0 ó # y cualquier otra.

Una vez que se haya pulsado cualquiera de las teclas *, 0 ó # y, mientras está habilitada la segunda pulsación, un led indicador permanecerá encendido (8).

Sin pulsación previa	Pulsación previa *	Pulsación previa 0	Pulsación previa #
1 2 3 4 5 6 7 8 9	A B C D E F G H I	J K L M N O P Q R	S T U V W X Y Z (

Por lo tanto, tal y como puede verse en la tabla de las órdenes posibles se permiten 36 (9 utilizando pulsación simple y 27 utilizando pulsación doble (9 órdenes distintas para cada una de las 3 teclas habilitadas a tal fin: *, 0 y #).

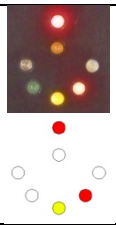
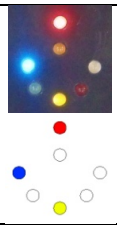
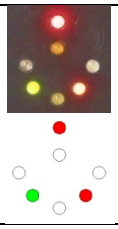
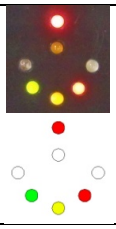
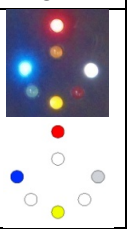
Asimismo, el teclado permite obtener de manera codificada la configuración de diferentes parámetros del dispositivo: activación de ecualizado, frecuencia de estimulación variable, intensidad de las estimulaciones continua y de 1 Hz e intensidad de estimulación de la frecuencia variable. Se mostrará de manera codificada utilizando los 5 leds de indicación de órdenes (7).

Pulsación previa *			Pulsación previa #		
	Ecualizado	Frecuencia variable	Intensidad continua y 1 Hz	Intensidad frecuencia variable	

Ecualizado (pulsación consecutiva *, seguido de 0): muestra si está activo o no. El ecualizado permite cambiar las intensidades para la estimulación continua (0 Hz) y de 1 Hz (en conjunto) y para la estimulación de frecuencia variable. En caso contrario se utiliza la misma intensidad para ambas.

Activado	No activado
	

Frecuencia de estimulación variable (pulsación consecutiva *, seguido de #): muestra un código para cada una de las posibles opciones de frecuencia variable (2 Hz, 4 Hz, 6 Hz, 8 Hz, 10 Hz, 14 Hz y 18 Hz).

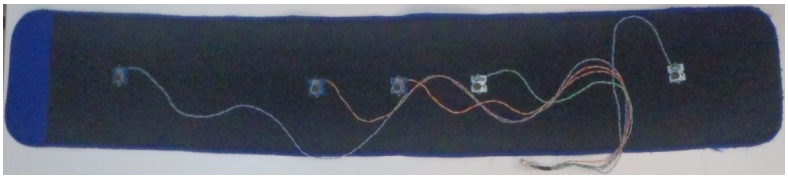
2 Hz	4 Hz	6 Hz	8 Hz	10 Hz	14 Hz	18 Hz
						

Intensidad de las estimulaciones continua y de 1 Hz (pulsación consecutiva #, seguido de *): Se muestra la intensidad utilizando los leds a modo de barra de intensidad.

Intensidad mínima	Intensidad máxima
	

Intensidad de la estimulación de frecuencia variable (pulsación consecutiva #, seguido de 0): Se muestra la intensidad utilizando los leds a modo de barra de intensidad, de forma análoga al anterior.

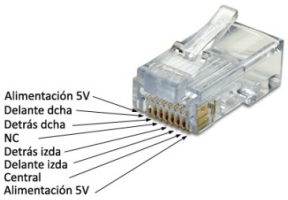
Cinturón estimulador

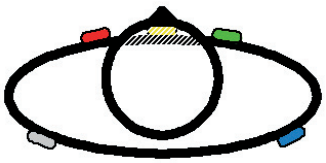


La **interfaz eferente** se conecta al controlador mediante un cable RJ45 y éste será el encargado de codificar la estimulación que corresponda en función del dato recibido por Bluetooth, nuevamente por simplicidad las órdenes enviadas se corresponden con un único carácter.

Carácter recibido	Actuadores en funcionamiento y frecuencia				
	Detrás izda	Delante izda	Central	Delante dcha	Detrás dcha
A			1 Hz		
B				1 Hz	
C		1 Hz			
D					1 Hz
E	1 Hz				
F			Freq var		
G				Freq var	
H		Freq var			
I					Freq var
J	Freq var				
K	1 Hz	1 Hz	1 Hz	1 Hz	1 Hz
L	Freq var	Freq var	Freq var	Freq var	Freq var
M			1 Hz	1 Hz	
N		1 Hz	1 Hz		
O			1 Hz		1 Hz
P	1 Hz		1 Hz		
Q		1 Hz		1 Hz	
R	1 Hz				1 Hz
S			Freq var	Freq var	
T		Freq var	Freq var		
U			Freq var		Freq var
V	Freq var		Freq var		
W		Freq var		Freq var	
X	Freq var				Freq var
Y			1 Hz	Freq var	
Z	Freq var		1 Hz		
1			Continua		
2				Continua	
3		Continua			
4					Continua
5	Continua				
6			Continua	Continua	
7		Continua	Continua		
8			Continua		Continua
9	Continua		Continua		
0		Continua		Continua	
(	Continua				Continua
)	Continua	Continua	Continua	Continua	Continua

Con el sistema de control puede utilizarse cualquier estimulador con 5 actuadores, siempre y cuando, se respete la conexión del dispositivo, conector (6). El actual diseño emplea dos pines para la alimentación, uno para cada uno de los actuadores (un total de cinco) y uno sin conexión.

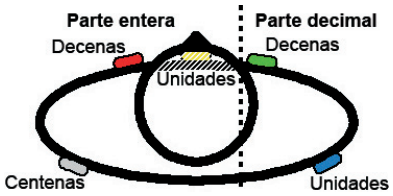




El cinturón mostrado debe colocarse de manera que los actuadores queden colocados ligeramente por encima del ombligo del usuario, siguiendo el patrón representado en la figura. Se considera una vista desde arriba del sujeto en la que se ha representado de manera esquemática la cabeza y el tronco.

Para el caso concreto de la codificación de cantidades, se ha utilizado una representación numérica mediante los cinco actuadores incluidos en el cinturón.

De esta forma, se ha elegido poder representar números enteros con dos decimales, de manera que cada uno de los actuadores se utiliza para codificar cada una de las cifras que conforman el número, tal y como se muestra en el esquema presentado a la derecha.



La forma elegida de codificar cada una de las cifras ha sido mediante vibraciones largas o cortas de cada uno de los actuadores. Así cada una de ellas del 0 al 9 se corresponde con una serie de vibraciones largas y cortas (dónde - representa una vibración larga y · una vibración corta):

1	2	3	4	5	6	7	8	9	0
·	··	···	····	-	-·	-··	-···	-····	--

En todo caso, aunque se trate de un número menor que 100 o que 10, la estimulación comenzará en el actuador correspondiente a las centenas codificando las cifras más significativas con 0. La parte decimal sólo será codificada si ésta existe, es decir, para números enteros, se enviará el número sin estimulación en las cifras correspondiente a la parte decimal. En la siguiente tabla se muestran tres ejemplos de codificación de números, utilizando la codificación de colores de las anteriores figuras y de vibraciones largas y cortas de la anterior tabla:

7		
--	--	-··
0	0	7

9,35				
--	--	-····	···	-
0	0	9	3	5

853,2			
-····	-	···	··
8	5	3	2

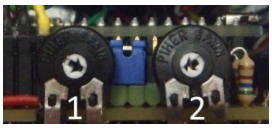
Configuración de los parámetros del sistema

Como se ha indicado anteriormente, el sistema permite configurar diferentes parámetros:

- Ecualizado. Se permite que la intensidad de estimulación sea diferente para la continua y de 1 Hz y para la frecuencia variable.

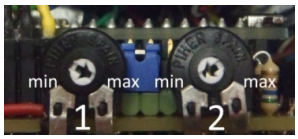
- Frecuencia de estimulación variable. Se puede configurar la tercera frecuencia de estimulación entre los valores: 2, 4, 6, 8, 10, 14 y 18 Hz.
- Intensidad de estimulación continua y de 1 Hz. Se permite configurar la intensidad de estimulación desde un mínimo hasta un máximo.
- Intensidad de estimulación de frecuencia variable. Se permite configurar la intensidad de estimulación desde un mínimo hasta un máximo, sólo si está activado el ecualizado si no es idéntica a la anterior.

Esta configuración se realiza por medio de dos potenciómetros y un *jumper* visibles si se retira la tapa inferior del sistema de procesado. En la figura adjunta se observan estos dispositivos para la configuración



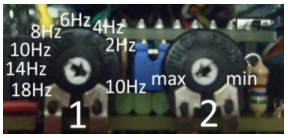
Jumper izquierda:

- Ecualizado: activado.
- Frecuencia de estimulación variable: leída de la EEPROM o del estado anterior.
- Intensidad de estimulación continua y de 1 Hz: potenciómetro 2.
- Intensidad de estimulación de frecuencia variable: potenciómetro 1.



Jumper derecha:

- Ecualizado: inactivado.
- Frecuencia de estimulación variable: potenciómetro 1.
- Intensidad de estimulación continua y de 1 Hz: potenciómetro 2.
- Intensidad de estimulación de frecuencia variable: potenciómetro 2.



El sistema permite guardar datos en la EEPROM que serán cargados al inicio del programa. Para grabar los datos es necesario enviar por puerto USB, conectando el cable con el ordenador, el carácter '>' y serán almacenados los datos relativos a la frecuencia de estimulación variable. En el caso que se quieran conocer los parámetros configurados vía USB, en el ordenador, basta con enviar el carácter '<'.

SOFTWARE

La aplicación para Android de *TactileCom* está diseñada para permitir la comunicación bidireccional entre el dispositivo móvil y el cinturón. Pero, ¿cómo funciona este sistema?

La aplicación muestra una interfaz gráfica en la que los comandos aparecen como iconos que se pueden pulsar. Cada uno de estos iconos lleva asociado un comando que será enviado al cinturón codificado mediante los caracteres 0-9, A-Z, (ó). Las estimulaciones correspondientes a cada comando se han explicado en el anterior apartado.

En la siguiente imagen se muestra la pantalla principal de la aplicación:

De forma adicional, tal y como se puede observar, la interfaz dispone también de una entrada de comandos por teclado o por voz, en este caso, se hace el reconocimiento del texto insertado y se determina si el comando es válido o no. Si lo es, se envía a la interfaz eferente.

En la siguiente tabla se muestra la correspondencia entre los comandos que se pueden enviar y la correspondiente codificación que corresponde a dicho comando en forma de carácter alfanumérico.

Carácter recibido	Comando
A	Amarillo
B	Verde
C	Rojo
D	Azul
E	Blanco
F	Hola
G	Adiós
H	Comida
I	Colegio
J	Medicina
K	Cuidador

L	Emergencia
M	Visita
N	Habitación
O	A dormir
P	Muy bien
Q	Mal
R	Padre
S	Madre
T	Hermano
V	Estudiar
W	Salir
X	Ayuda
Y	No olvides

Z	Solo
1	NO DEFINIDO
2	NO DEFINIDO
3	NO DEFINIDO
4	NO DEFINIDO
5	NO DEFINIDO
6	NO DEFINIDO
7	NO DEFINIDO
8	NO DEFINIDO
9	NO DEFINIDO
0	NO DEFINIDO
(	Necesitas algo
)	NO DEFINIDO

Como se puede ver en la tabla aún se pueden añadir algunos comandos más hasta completar todas las posibles combinaciones de los sensores vibratorios de los que dispone el cinturón.

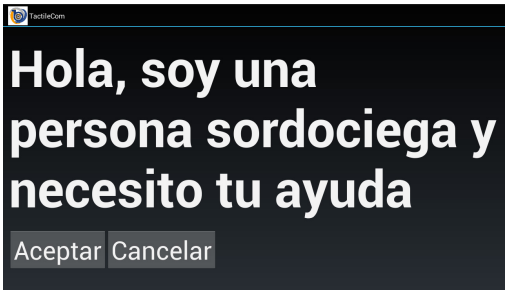
Por otra parte tenemos la parte de comunicación entre el teclado numérico y la aplicación móvil. En este caso lo que se tiene es una serie de comandos pre-programados en un archivo que están asociados a cada una de las posibles combinaciones que puede dar el teclado y que se han mostrado anteriormente en este documento.

Debido a las características del teclado se definen varios entornos en los que la persona usuaria de este sistema puede desenvolverse. Estos son cuatro: hogar, paseo/compras, hospital/médico y transporte público. En este punto, se procede a mostrar cada uno de los comandos y su combinación de teclas necesaria para proceder a su envío.

Combinación de teclas	Comando
HOGAR	
1	Estoy bien
2	Estoy mal
3	¿Qué ocurre?
4	¿Puedo ayudar?
5	NO DEFINIDO
6	NO DEFINIDO
7	NO DEFINIDO
8	Ir al aseo
9	NO DEFINIDO
PASEO/COMPRAS	
*+1	Cruzar la calle
*+2+(0-9)	Buscar dirección
*+3	¿Cantidad?
*+4	¿Talla?
*+5	¿Color?
*+6	¿Precio?
*+7	Estoy perdido
*+8	Ir al aseo
*+9	Estoy mal

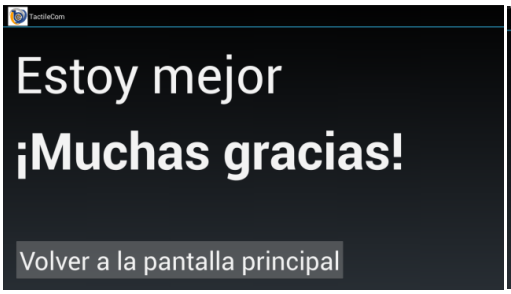
Combinación de teclas	Comando
HOSPITAL/MÉDICO	
0+1	Estoy mejor
0+2	Estoy peor
0+3	Avisar a
0+4	Lugar de la cita
0+5	¿Tratamiento?
0+6	¿Cuidados?
0+7	NO DEFINIDO
0+8	Ir al aseo
0+9	Dolor intenso
TRANSPORTE PÚBLICO	
#+1	Nº de bus y ayuda para subir
#+2	Indicación de parada de bus
#+3+(0-9)	Dirección para el taxista
#+4	NO DEFINIDO
#+5	NO DEFINIDO
#+6	¿Precio?
#+7	Estoy mal
#+8	NO DEFINIDO
#+9	Estoy pedido

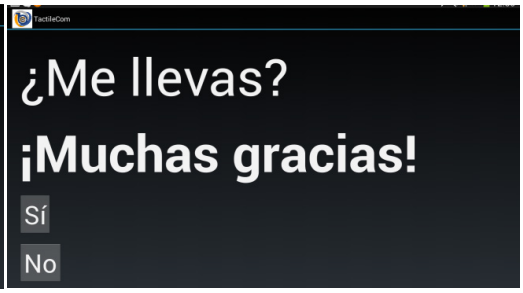
Cuando el sordociego quiera comunicarse con el cuidador, generalmente en el entorno etiquetado como hogar, la aplicación se comportará de manera diferente que en el resto de entornos, ya que en éstos la persona que reciba, lea, el mensaje será informada de la situación personal de la persona discapacitada y se le invitará a que le preste su ayuda mediante la pantalla de la derecha:



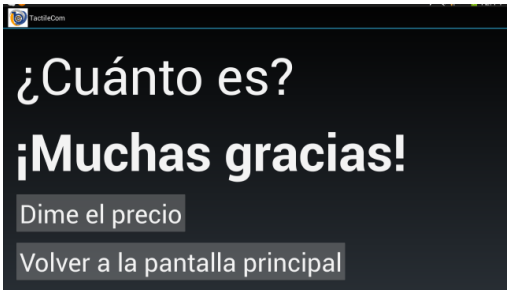
En el caso de que el receptor acepte la petición de ayuda se le mostrará el comando seleccionado. En caso contrario, se informará mediante el cinturón (mediante la vibración asociada al comando “todo mal”) de que la persona receptora no ha podido ofrecerle su ayuda.

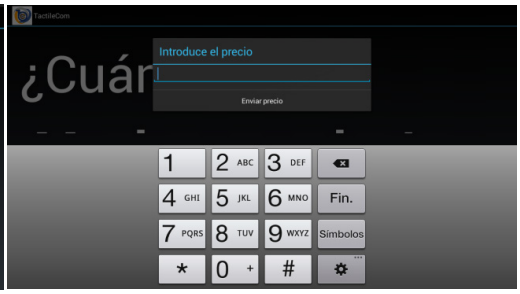
Una vez se ha aceptado la petición o se ha mostrado directamente en el caso del entorno hogar. La pantalla mostrará, en este caso, el comando seleccionado y un botón para volver a la pantalla principal. También se dará las gracias por la ayuda prestada como se muestra en las siguientes capturas de pantalla.





Este es el funcionamiento básico de la aplicación, pero hay comandos con los que se efectúan operaciones especiales; éste es el caso de los comandos en los que se pregunta por el precio o cantidad, en los que además de mostrar el comando se invita a la persona que ofrece la ayuda a introducir esta mediante un cuadro de texto. Esta cantidad será enviada a la interfaz de estimulación y será recibida por el sordociego según la codificación presentada anteriormente. Concretamente se enviarán las cifras del número delimitadas antes y después por # y colocando los 0 necesarios a la izquierda del número si éste es menor que 100.





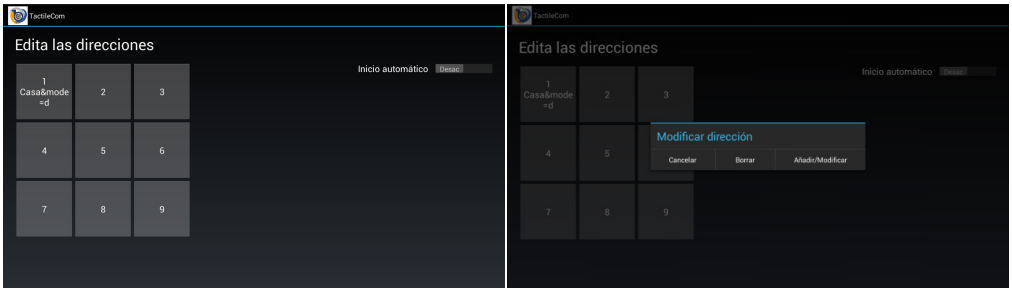
Otra de las posibilidades que ofrece la aplicación es la de buscar una dirección o preguntar para que te lleven hasta un determinado punto, en este caso lo que se hace es mediante la aplicación de Google Navigator iniciar una navegación asistida por GPS hasta la dirección seleccionada por el teclado.

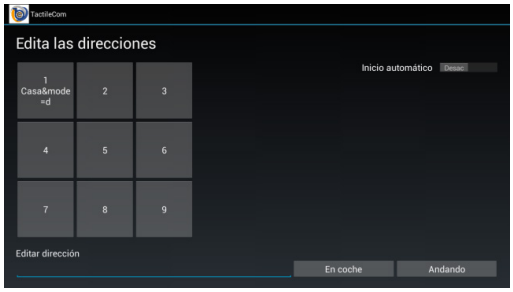
La forma de seleccionar las direcciones es también mediante el teclado numérico. Es decir, tras seleccionar el comando que indica que vas a preguntar por una dirección se puede elegir una dirección configurada en la aplicación y programada en uno de los 9 números del teclado. Anteriormente, se mostró en la tabla de comandos disponibles mediante teclado esta tercera pulsación necesaria en los comandos que requieren una dirección.

De manera adicional, se ha determinado que cuando se pulse una tecla en el teclado físico del dispositivo se cargue la pantalla con el comando correspondiente a la tecla, o en el caso que no tenga ninguna función asignada dicha tecla, la última pantalla visitada. Si pulsamos la tecla 5, se mostrará el menú principal. También se hace que el dispositivo móvil vibre para indicar que se ha recibido un comando.

Configuración

En las siguientes capturas se puede ver el funcionamiento del menú de configuración. En la primera se muestra un panel con los 9 botones que se corresponden con el teclado real. Al pulsar en uno de ellos se puede añadir, modificar o borrar la dirección de ese botón.

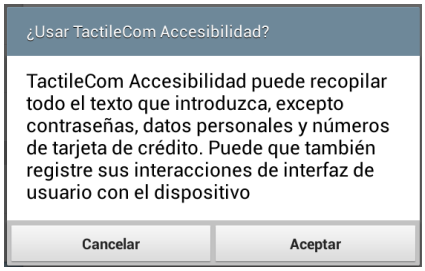




Además como se ve en la parte derecha de las capturas, hay un botón que se puede activar y desactivar y que vale para hacer que la aplicación se inicie de manera automática al iniciar el teléfono.

Como funcionalidad adicional se añade como servicio de accesibilidad, que hay que activar en Ajustes->Accesibilidad->TactileCom Accesibilidad. Y que nos sirve para reconocer las notificaciones de Skype™, en el caso en que esté instalada y notificar mediante el cinturón de este hecho a la persona que lo lleve enviándole un comando “Hola”.

Al activarlo se pedirá confirmación de este hecho mediante un mensaje que informa sobre si se quiere realmente aceptar el uso del servicio de accesibilidad; si se acepta, la aplicación será capaz de recoger las notificaciones del sistema, comprobar de que tipo son y notificar al cinturón de las correspondientes a Skype™.



Por último, es recomendable, iniciar el dispositivo controlador del cinturón antes que la aplicación para que la conexión Bluetooth sea efectiva, si no se hace así puede que sea necesario reiniciar la aplicación para que el dispositivo sea reconocido de forma correcta.

Índice de figuras

Bibliografía

[1] De la Rosa Steinz, R "Acondicionamiento y procesado en tiempo real de señales biológicas", Tesis Doctoral, Universidad de Valladolid, 2005.

[2] Mompín Poblet, J. Ïntroducción a la bioingeniería",Ed. Marcombo, S.A.,Barcelona, 1988.

[3] Cardinalli, D.P "Manual de neurofisiología", Ed. Díaz de Santos, S.A., España, 1992.

[4] "Diccionario de Biología", Larousse, Spes editorial, S.L, 2003.

[5] Chaves,A *Apuntes de la Cátedra de Neurofisiología*, Facultad de Psicología, Universidad de Buenos Aires, 2005.

[6] FlicKinger, CJ y cols."Medical cell biology",WB Saunders, Filadelfia,1979.

[7] De la Rosa Steinz, R "Diseño de un sistema de adquisición de señales bioeléctricas", Proyecto Fin de Carrera Ingeniero de Telecomunicación, Universidad de Valladolid, 1999.

[8] K. Englehart, B. Hudgins, P. A. Parker, "A Wavelet-Based Continuous Classification Scheme for Multifunction Myoelectric Control", IEEE Trans. Trans. Biomed. Eng., vol. 48, pp. 302-311. Marzo 2001

[9] De la Rosa Steinz, S "Desarrollo de nuevas técnicas de procesado y entrenamiento en tiempo real con señales biológicas sobre la plataforma UVa-NTS", Proyecto Fin de Carrera Ingeniero de Telecomunicación, Universidad de Valladolid, 2008.

[10] MSDN, Microsoft.CDC Class.
`https://msdn.microsoft.com` Visitado por última vez: 25/08/2017.

[11] MSDN, Microsoft.Beep Function.
`https://msdn.microsoft.com` Visitado por última vez: 24/08/2017.

[12] Alonso Alonso, A. "Tecnologías de rehabilitación". Laboratorio de Electrónica y Bioingeniería de la Universidad de Valladolid.`http://albergueweb.uva.es/leb/index.html`. Visitado por última vez: 25/08/2017

[13] Gregory,K .Edición Especial. Microsoft Visual C++ 6", capítulos del 1 al 5 y 8,Ed. PRENTICE HALL, Madrid, 1999.

[14] Ceballos, F.J,"Programación orientada a objetos con C++", Madrid,2007.

[15] Mann, G "What is C++ inline functions",Página de C++, India, Mayo 2014. `www.cplusplus.com`. Visitado por última vez: 25/08/2017.

[16] *Singstar*^{TM}, PlayStation. Desarrollador:London Studio. Publicado por Sony Computer. Primera versión: Mayo 2004.

[17] Alonso Alonso,A "Sistema Versátil de Comunicación para Sordociegos: TactileCom",Valladolid, 2017