Entrenador para el control de prótesis mioeléctricas

Víctor Rodríguez Doncel

Diciembre de 2001

Índice

Índice de cuadros

Índice de figuras

1. Resumen

La señal mioeléctrica se adquiere de los músculos con electrodos de superficie y guarda relación directa con su nivel de contracción. A partir de las señales mioeléctricas es posible extraer señales de control que gobiernen otros dispositivos tales como una prótesis mioeléctrica o el puntero de un ratón en la pantalla de un ordenador.

El problema de extraer las señales de control de manera certera es un problema de reconocimiento de patrones, y como tal ha sido planteado. Se puede separar en dos bloques la toma de decisiones: primero ha de extraerse un conjunto de características reducido pero significativo de la señal, y después un clasificador ha de decidir a partir de estos datos.

Frente a una solución más complicada que incluyera la transformada wavelet como origen de las características y una red neuronal como bloque clasificador, se ha optado por utilizar características sencillas y un decisor gaussiano, que, en todo caso, han rendido un porcentaje de aciertos satisfactorio.

En la realización práctica del clasificador se ha aprovechado el trabajo previo realizado en el Grupo de Bioingeniería[1]; se ha utilizado el hardware aquí desarrollado y se han integrado los componentes software que habían sido desarrollados de manera independiente. A partir de todo ello se han extendido los programas y se han desarrollado aplicaciones nuevas.

Dos nuevas librerías de enlace dinámico son el soporte para las aplicaciones presentes y futuras; una gestiona de manera transparente la adquisición de la señal EMG y proporciona un primer procesado vía software de la señal, y la otra contiene los algoritmos del clasificador de patrones. Sobre ellas se apoyan el resto. Una aplicación muestra en pantalla la señal EMG, otra muestra un brazo que responde a la señal mioeléctrica y otra permite mover un puntero por la pantalla guiado por la señal mioeléctrica.

Tanto el desarrollo teórico como el práctico que aquí se presentan, siendo completos, tienen carácter introductorio, y pueden ser tomados como un buen punto de partida para desarrollar futuros trabajos de interés en el campo del control mioeléctrico.

[1]El Grupo de Bioingeniería y Rehabilitación de la Universidad de Valladolid tiene su sede en la Escuela Técnica Superior de Ingenieros de Telecomunicación, camino del cementerio s/n, Campus Miguel Delibes, Valladolid.

2. Introducción

2.1. Marco

La señal mioeléctrica es la señal eléctrica que se produce en los músculos cuando éstos se contraen. Esta señal, si bien tenue, es susceptible de ser medida con un equipo adecuado. Uno de los usos que se han dado a esta señal de electromiografía (o EMG[2]) ha sido en el campo de las prótesis mioeléctricas.

Un amputado de la extremidad superior, a nivel del codo, puede poseer los músculos del brazo intactos, si bien incapaces de realizar función alguna. Las prótesis mioeléctricas recogen la señal EMG de estos músculos residuales, la procesan y la utilizan como señal de control para gobernar unos servomotores en la prótesis que pueden hacer las funciones del brazo original. Por tanto, el amputado puede recuperar parte de las funcionalidades de su antiguo miembro mediante la contracción o reposo de unos músculos que, en todo caso, habían de serle inútiles.

Sin embargo no todos los amputados son capaces de gobernar una prótesis, bien porque la amputación haya sido realizada a una altura demasiado alta, o bien porque el enfermo carezca de habilidad suficiente en unos músculos que quizás hayan degenerado por su falta de uso. Dado que una prótesis mioeléctrica es cara, sería de manifiesta utilidad un sistema capaz de distinguir entre pacientes aptos para vestir estas costosas prótesis y no aptos, antes de que éstos la adquieran y la adapten a su brazo.

El desarrollo de tal sistema, ha sido una meta del Grupo de Bioingeniería y Telemedicina de la Universidad de Valladolid en los últimos años. Este proyecto continúa la labor ya iniciada y espera ser continuada. Los hitos alcanzados hasta el momento son tres, disponibles en la forma de otros tantos proyectos.

- Antonio Hernández Bajo programó una aplicación gráfica, en la cual se muestra un brazo prostético virtual que ofrecía tres grados de libertad (DOF[3]) [69]. Dicha prótesis habría de ser gobernada por la señal

[2]En forma de apéndice figura una lista con los acrónimos que se emplean en esta memoria y su significado.

[3]En inglés *degrees of freedon*. A fin de evitar confusiones con el resto de la bibliografía, a lo largo de la memoria se han mantenido las siglas inglesas cuando las mismas en castellano no están generalizadas. También se comentan las palabras cuya traducción al castellano pueda empeorar la comprensión.

mioeléctrica en el futuro.

- Ramón de la Rosa desarrolló un sistema de adquisición de señales bioeléctricas, capaz de recoger la señal EMG y amplificarla hasta niveles razonables [93].

- Lorenzo Ferrero, finalmente, desarrolló un conversor analógico/digital, que toma la señal de la cabecera análogica y la digitaliza para que pueda ser procesada en el ordenador; programó además una aplicación que muestra en pantalla la señal EMG de manera continua.

Figura 1: Equipo disponible.

2.2. Objetivos

El objetivo de este proyecto es enlazar todo lo anterior y establecer los algoritmos necesarios que relacionen la señal de entrada en bruto con las decisiones tomadas sobre el brazo mioeléctrico virtual, o en un sentido más amplio, sobre cualquier otro sistema que haya de ser controlado. Es un trabajo de programación vincular todos los trabajos previos; es una trabajo de tratamiento y procesado digital de la señal tomar la decisión correcta.

La extracción de características y la clasificación de patrones a partir a ellas intentan maximizar la tasa de aciertos del decisor, es por tanto una tarea siempre inconclusa y susceptible de continuas mejoras; este trabajo debería ser continuado y mejorado por lo que ha de mantener un carácter abierto.

2.3. Plan

La primera sección fue ocupada por el resumen del proyecto. La segunda sección es la introducción presente, que se completa con un vistazo al panorama de las aplicaciones de la señal EMG y su uso en el campo de las prótesis. El éxito de una prótesis mioeléctrica no depende únicamente de la bondad en las estimaciones que haga, y en verdad no son menos importantes otros aspectos como el peso de la prótesis, su estética etc. Todos estos asuntos no son objeto de nuestro estudio[4], ciertamente, pero a fin de enmarcar este trabajo en su contexto, se da un repaso rápido a la tecnología de las prótesis en el siguiente apartado de la introducción.

En la tercera sección se mostrará la relación entre la fisiología y las matemáticas de la señal EMG. El enfoque desde el punto de vista del ingeniero es tratar al músculo como una caja negra, con la voluntad del sujeto como entrada y la respuesta eléctrica como salida.

Sin embargo, la señal que se mide en la superficie de la piel es el resultado de los procesos fisiológicos subyacentes, y si bien la relación es compleja y manchada por ruido, no se debería olvidar el origen de la señal adquirida, pues el análisis de dichos procesos dará pistas esenciales para mejorar el tratamiento de la señal.

En la cuarta sección se plantea el problema y se caracteriza de manera matemática, estableciendo las diferencias entre el control proporcional y el control discreto.

En la quinta se describe el clasificador de patrones. Un decisor ha clasificar la señal EMG a partir no de la señal mismo sino de ciertas características que se extraen de ella. Se presentan los distintos enfoques que ha recibido el clasificador, y se hace especial hincapié en la aproximación estadística, que es la que se ha aplicado en el proyecto. En la quinta sección se enumeran las diversas características de la señal que han sido empleadas como entradas del clasificador. Las características se han agrupado en cuatro bloques. Estos son la fuerza, como característica que gobernaba las prótesis proporcionales, las

[4]Y por eso figuran en la introducción

características que se extraen de la representación temporal de la señal , las que se extraen de la frecuencial y las que hacen uso de información temporal y frecuencial simultáneamente. Una sección séptima recapitula sobre las dos secciones anteriores antes de dar paso a la parte práctica.

La octava sección por fin describe el sistema propuesto, desde el nivel *hardware* hasta los algoritmos implementados. Se describen brevemente los hitos realizados previamente, se detalla el sistema realizado y se traza el trabajo futuro. Se ha elaborado un minucioso análisis de *software*, diseñando una arquitectura flexible y con vistas a ser mantenida. Es vocación explícita de este diseño integrar el trabajo pasado y proporcionar una plataforma sólida para el software que se desarrolle en el futuro.

Finalmente se extraen las conclusiones en un noveno y último apartado en el que además se esboza el trabajo futuro a realizar en este campo. En forma de apéndice se ofrece un lista de símbolos utilizados a lo largo de este proyecto y una lista con las referencias citadas y que han estado disponibles durante la elaboración del proyecto, y también se facilita la parte del código fuente considerada como más relevante.

2.4. Aplicaciones de la señal EMG

Aunque conocida desde principios del siglo XX, la señal EMG no ha servido para ningún fin útil hasta que estuvieron disponibles los avances de la electrónica y de la informática. Aun hoy, la continua evolución de estos sistemas propicia constantes avances en los sistemas que aprovechan la señal EMG y la aparición de sistemas nuevos.

Las posibles aplicaciones son tantas como pueda permitir la imaginación. En este apartado se ofrecerán algunos ejemplos de aplicaciones.

2.4.1. Aplicaciones médicas

Como señal originada por el cuerpo humano que es, tiene un interés evidente en la práctica médica. Por una parte se ha aplicado en la diagnosis de enfermedades y por otra, se ha utilizado para conocer mejor el cuerpo humano.

En el campo de la diagnosis se utiliza para detectar la esclerosis lateral amiotrófica (observable en la señal EMG porque cada unidad motora se dispara a una frecuencia más alta de lo normal), la miositis (se disparan demasiadas unidades motoras de baja amplitud y duración con niveles bajos de contracción muscular), la neuropatía motora hereditaria de tipo 1 o la distrofia muscular.

Dado que la señal EMG revela la actividad muscular, es muy útil también a la hora de estudiar el movimiento y la coordinación de los músculos del cuerpo humano, los mecanismos que tiene a la hora de realizar tareas muy complejas, tales como el acto de caminar [6][36] o los movimientos cotidianos de un brazo [98].

También la señal EMG se ha utilizado para recrear mejor algunas zonas del cuerpo humano, como las zonas de inervación en el bíceps [27], o la longitud de las fibras musculares como función de su estado de excitación [120]. Esto último que se había venido haciendo con técnicas de sonomicrometría y ultrasonografía, ahora es terreno de la señal EMG. Se ha podido simular un brazo virtual muy detallado (con huesos, músculos en su longitud exacta etc.) cuyos movimientos son determinados por la señal EMG de un hombre real [144]. Un último ejemplo en el terreno del entrenamiento deportivo de alto rendimiento es la predicción de la altura a la que saltará un atleta a partir de su señal EMG [140], y el posterior análisis la misma a fin de mejorar los

resultados.

2.4.2. Aplicaciones de control

Desde los años 60 se ha venido utilizando la señal EMG como señal de control para prótesis movidas por motores eléctricos (prótesis mioeléctricas), o, con la misma filosofía, como señal de control para la excitación de músculos que han sufrido parálisis.

La sencilla adquisición de la señal en la superficie de la piel es idónea para obtener una señal de control, bien en personas amputadas que desean gobernar una prótesis mioeléctrica, bien en tetrapléjicos que desean enviar estimulación eléctrica funcional (FES) a electrodos aplicados sobre los músculos inermes [75]. Del mismo modo que la contracción muscular produce una señal eléctrica, la estimulación de un músculo con una señal eléctrica externa provoca su contracción.

Pero como señal de control que es, puede dirigir otros muchos dispositivos. Por ejemplo, para mover y activar el ratón de un ordenador, se han usado las señales de EMG de los músculos de la cabeza [126] en conjunción de la señal EEG.

En otro ejemplo [66] se aplican electrodos a un operador sano que desea controlar un robot o una mano robótica para tareas delicadas o peligrosas con sus señales mioeléctricas de manera natural. También se ha utilizado la señal EMG para entrenar a robots, la colaboración de la señal de video con la señal mioeléctrica permite al robot aprender y refinar habilidades al aprender del hombre [86].

En casos concretos el control ha llegado a ser excelente, y así se ha logrado incluso distinguir entre algunas letras (tan confusas como por ejemplo la 'a', la 'α' y la 'o') que escribe una persona mediante el uso del análisis de la señal EMG de varios músculos de la mano [29].

2.5. Prótesis mioeléctricas

Una prótesis es el sustituto mecánico de un miembro amputado. Este proyecto toma como referencia las prótesis mioeléctricas que suplen el miembro superior de un amputado a la altura del codo.

Sólo en los Estados Unidos cada año hay unos 10.000 nuevos amputados

en la extremidad superior [118]. De éstos, sólo la mitad utiliza una prótesis, y de entre los que la empiezan a usar, sólo la mitad sigue usándola después de un año. Es decir, sólo el 25 % de los amputados norteamericanos acaban vistiendo una prótesis, y entre ellos son sólo una parte los que visten la prótesis mioeléctrica [95]. El porcentaje de uso en otros países no desarrollados es, evidentemente mucho menor, pues las prótesis además son caras, máxime si son gobernadas con la señal EMG. La conclusión es que el de las prótesis mioeléctricas es un mercado pequeño en el que no existen grandes empresas que puedan llevar a cabo investigaciones costosas[5].

Quede dicho desde un principio: el paciente suele tener expectativas superiores a lo que jamás se podrá desarrollar. La extremidades superiores humanas son extremadamente versátiles (no muchos animales alcanzan a rascarse todas las partes de su cuerpo con ellas, por ejemplo) y cualquier prótesis artificial que se desarrolle, por buena que sea, quedará ensombrecida por el órgano natural.

En todo caso, un amputado con una prótesis es capaz de realizar muchas más tareas que un amputado sin ella. Las prótesis actuales son complicados dispositivos mecánicos de alta tecnología, que permiten gran cantidad de movimientos y que restauran sensaciones como el frío, el calor o la presión mediante unos sensores en el brazo que transladan tales sensaciones al usuario, y ciertamente mejoran la calidad de vida de quienes las utilizan. Incluso quien utiliza una prótesis virtual reduce el dolor del miembro fantasma.

Una parte importante también de la población amputada viste una prótesis única y exclusivamente cosmética[6]. El miembro perdido es reemplazado por una prótesis que es similar en apariencia al original, aunque no sirve para casi nada. Estas prótesis están realizadas en PVC, en látex o en silicona y tienen la ventaja de que son ligeras y no requieren mantenimiento.

En vez de usar prótesis como adorno, también se pueden utilizar prótesis que puedan hacer alguna función, como por ejemlo abrir y cerrar la mano. El tipo de prótesis tradicional tradicional que permite esto es controlada con la fuerza directa del hombro y no hay ningún sistema eléctrico.

Con un arnés el movimiento del hombro se traduce en el cierre o apertura de la mano. El paciente debe tener el miembro residual suficientemente largo, y debe tener un rango de movimiento amplio para que la prótesis tradicional

[5]Algunos fabricantes de prótesis son “Otto bock”, “Motion Control”, “Hosmer” o “Liberty Mutual”, un panorama del mercado se encuentra en [103]

[6]De hecho, en el antiguo Egipto ya se usaban prótesis cosméticas

pueda funcionar. La ventaja de estas prótesis, sistemas totalmente mecánicos, es su sencillez, su robustez y su capacidad propioceptiva, es decir, al realizar una tarea el usuario es consciente de la fuerza que está desarrollando, lo cual no sucede con las prótesis mioeléctricas. A veces sin embargo, es algo incómodo el arnés.

Figura 2: Prótesis tradicional.

2.5.1. Prótesis mioeléctricas.

En las prótesis mioeléctricas, la señal EMG de los músculos residuales controla los motores eléctricos de la prótesis. Se comercializan prótesis de mano, de brazo por encima del codo y de brazo por debajo del codo, como se muestra en la figura 3. Cualquiera de ellas se puede sujetar al miembro residual de varias maneras, la más común es haciendo el vacío entre la prótesis y el miembro, con lo cual queda bien sujeta aunque también se puede enganchar de otras maneras.

Figura 3: Prótesis mioeléctricas.

Las prótesis mioeléctricas llevan vendiéndose desde los años 60, pero aún siguen sin resolverse ciertos problemas. La prótesis mioeléctrica es más pesada que la prótesis tradicional, debido a los motores y las baterías, y es además más frágil y no puede soportar demasiado peso (unos tres o cuatro kilogramos). Es difícil ofrecer varios grados de libertad y no todos los usuarios de estas prótesis muestran soltura en su manejo. Algunas ofrecen

control proporcional sobre una función concreta, y para cambiar de función hace falta activar un conmutador con la otra mano o bien realizar dos contracciones rápidas. Todas estas dificultades, sin embargo están siendo poco a poco superadas. El precio de una prótesis mioeléctrica es caro, y varía desde el millón y medio a los diez millones de pesetas [101].

Retroalimentación. Las prótesis modernas emplean sistemas de *feedback* o retroalimentación. Si no se emplea ningún sistema de retroalimentación, el usuario no sabrá la fuerza que está ejerciendo su prótesis, y le será imposible por ejemplo coger una cajetilla de tabaco sin aplastarla, a no ser que esté mirando. Es muy molesto estar mirando constantemente lo que se está haciendo, y eso fue una de las principales causas de insatisfacción de los usuarios de las primeras prótesis mioeléctricas [81].

Las prótesis han incorporado desde hace poco sensores que adquieren la señal en la extremidad perdida y se la hacen llegar al sujeto. La mano Otto Bock por ejemplo tiene sensores de contacto y de deslizamiento; la señal de deslizamiento es interpretada además por la propia prótesis para ejercer la fuerza justa para el agarre, como se detalla en [88] o en [125]; otros fabricantes incluyen sensores de temperatura [119].

En el fondo estas técnicas vienen a reestablecer la coordinación entre el sistema nervioso central (que sabe lo que es una endeble cajetilla de tabaco) y los sensores y actuadores (que ignoran cuánta fuerza han de aplicar y no hacen saber al cerebro el grado de sujección logrado).

Entrenamiento. Un requisito casi imprescindible para lograr el éxito con una prótesis mioeléctrica es un entrenamiento adecuado. Un paciente que haya estado largo tiempo sin miembro y sin prótesis, estará acostumbrado a no utilizar nada y será muy difícil adaptarle el nuevo miembro [101][99]. En cambio, en niños, la tarea es mucho más sencilla. En casos de falta congénita de un miembro, se adaptan prótesis pasivas desde los 3 meses de edad[7].

Se demuestra que quienes han recibido entrenamiento adecuado mejoran su habilidad con la prótesis. En tests realizados con tareas comunes, como abrir un bote de aspirinas o doblar una toalla, los sujetos entrenados mostraban una destreza mucho mayor que quienes no lo estaban.

[7]La primera prótesis mioeléctrica que se instaló a un niño fue en Suecia a una niña de 3 años [35].

El trabajo que aquí se propone puede servir como evaluación de las habilidades de un sujeto aun sin tener la prótesis, pero también podría servir como entrenador. No es la primera vez que se programa un entrenador con una animación gráfica en un ordenador [47]. Así, el sistema que se presenta en este proyecto puede servir simultáneamente como *entrenador* y como *evaluador*.

2.5.2. Tendencias y futuras prótesis

El diseño de una prótesis mioeléctrica es un trabajo multidisciplinar. En él intervienen ingenieros especializados en la señal eléctrica, ingenieros especializados en la mecánica del sistema, programadores con conocimientos de técnicas en inteligencia artificial; pero también médicos y especialistas ortopédicos, y sobre todo, los propios amputados con su experiencia. La coordinación y sinergia entre todos ellos no siempre es buena, y debería existir mayor fluidez. Sería bueno por ejemplo que en el futuro los cirujanos coordinaran algo con los expertos en prótesis para decidir entre ambos el punto en el cual se realizará la amputación, pero es imprescindible que el usuario realimente a todos indicando su grado de satisfacción y mostrando dónde se deben mejorar.

En 1998 se realizó una encuesta entre 80 usuarios de prótesis para definir cómo debería ser la siguiente generación de prótesis [82]. Sus conclusiones fueron:

- Tanto la estética como la funcionalidad de la prótesis son igualmente importantes, así por ejemplo una protesta trivial pero importante para el usuario es sobre la poca variedad en el color de las prótesis.
- El peso de la prótesis debería ser más bajo. El 'codo Boston' o el 'brazo Utah' pesan por ejemplo un kilo y medio.
- La prótesis deberían dar servicio durante todo el día; utilizando baterías de duración más larga.
- Se debería poder realizar incluso algún deporte con ellas y se debería poder conducir con ellas.

De aquí ha de extraerse la conclusión de que el diseño de una buena prótesis es un trabajo de equipo, y que un buen trabajo en el control mioeléctrico, por ejemplo, ha de ser respaldado por una buena calidad en el

resto de los elementos, por más que pueda parecer algo tan trivial como el color de la prótesis.

Una tecnología moderna que busca mejorar las cuestiones mecánico es el diseño de la prótesis mioeléctrica mediante el uso de músculos artificiales SMA[8] [94]. Este material tiene la característica de cambiar su longitud cuando se le aplica una tensión a sus extremos, y se viene usando en robótica desde 1983. Sin embargo aun tiene muchos problemas, como por ejemplo que es capaz de soportar poco peso.

En cuanto a las tendencias en el sistema de control, el futuro parece estar en dejar una parte del control de la mano (o brazo) al libre arbitrio de la prótesis. El primer atisbo en este sentido fue en 1993 [40], cuando se proponía corregir el error que solía cometer el usuario al maniobrar, punto de vista muy distinto a corregir los errores de la máquina.

La prótesis ha de contar con inteligencia artificial propia, subordinada a la voluntad del sujeto. Es el *control jerárquico*. El mismo cerebro humano de hecho utiliza un control jerárquico. Los subsistemas del individuo son lentos y las comunicaciones entre ellas no permitirían al sujeto reaccionar suficientemente rápido para responder al mundo exterior si todo el control fuera procesado en un solo punto. Por tanto, los elementos individuales deben autoregularse y dejar sólo al control central una parte del procesamiento.

Por ejemplo, si nuestro fumador usuario de antes desea agarrar su cajetilla, puede dejar la elección de la fuerza justa a los sistemas de inteligencia artificial de la prótesis, y un sensor de deslizamiento incrementará la fuerza hasta que la cajetilla no deslice; y la fuerza no será sin embargo suficiente para estrujarla. También se han empleado otros sensores, como sensores de presión o sensores ópticos.

Esta tendencia en las prótesis al control jerárquico invita también a utilizar el control discreto; basta con hacer entender a la prótesis que se desea agarrar algo y no es necesario graduar conscientemente la fuerza. Así pues, el control discreto que se presenta aquí queda más justificado [50][37][49][45]. También está en estudio el problema de la coordinación de dos agentes sobre un mismo sistema, siendo uno humano y el otro artificial.

Y finalmente cabe comentar un posible futuro sobre el origen de la señal de control de las prótesis. La señal EMG algunas veces se ha visto complementada con la señal de vibromiografía (VMG). Las contracciones musculares producen vibraciones que se pueden recoger mediante un acelerómetro

[8]En inglés, *shape memory alloy*.

acoplado a la superficie de la piel. En tanto que la existencia de estos “sonidos musculares” se han conocido desde hace mucho tiempo, nunca han sido explotadas para el control de prótesis y el campo está abierto.

3. Origen y adquisición de la señal EMG

La señal mioeléctrica o de electromiografía es la señal eléctrica que se mide en la superficie del músculo cuando se contrae. En esta sección se describe la señal EMG, con especial énfasis en los aspectos que puedan condicionar el desarrollo del sistema. La señal EMG es la base a partir de la cual se construye todo este trabajo, por tanto se desea caracterizarla desde un primer momento. No es de interés tanto el origen y la fisiología de la señal como una descripción matemática precisa. Se presta también más atención a los músculos del antebrazo por ser también los que centran el trabajo.

En primer lugar se establecerá el origen compuesto de la señal EMG y a continuación se repasará el origen fisico - químico del proceso. Después se describirá cómo se mide la señal EMG, y finalmente se esboza la compleja relación entre la señal EMG resultante y la fuerza del músculo. Ello es además el objeto de la subsección 6.1.

3.1. Origen de la señal EMG

Los músculos del cuerpo humano consisten en fibras musculares que son activadas por las motoneuronas. Los impulsos que llegan desde la médula espinal a través de una sola motoneurona activan varias fibras musculares, en un número que crece proporcionalmente con el tamaño del músculo. Estas fibras forman un grupo llamado *unidad motora* (MU). Cada motoneurona activa por tanto un número variable de fibras, número que va desde una decena hasta varias miles. Dicha cifra se llama *tasa de inervación.*

Figura 4: Estimulación del músculo

La respuesta eléctrica a la estimulación de una MU se llama *potencial*

de acción de unidad motora (MUAP), y para mantener la contracción de un músculo han de descargarse periódicamente MUAPs de refresco. Este es el origen de la señal EMG. La señal EMG que se mide contiene una suma de los trenes de MUAPs, ponderados por la posición y tamaño de las fibras. Su amplitud cuando se mide con electrodos de superficie es a lo sumo de 10 mV, y es función de multitud de variables, como el grado de humedad, las capas de grasa entre el electrodo y el músculo o la temperatura ambiente (a más frío mayor amplitud de la señal).

Figura 5: Composición de la señal EMG

Si se desea incrementar la fuerza de un músculo se reclutarán más motoneuronas, y si están ya todas en activo, se aumentará la frecuencia de refresco. Las MU de un músculo se reclutan en su totalidad cuando se ha realizado el 75 % del máximo esfuerzo voluntario (MVC) aproximadamente [12]. El umbral de activación de las motoneuronas que controlan pocas fibras parece ser que es más bajo que el resto, así que son las primeras en activarse. Esto tendrá alguna repercusión en el trabajo que aquí se presenta, como se verá más adelante. Es posible al menos estimar cuantas motoneuronas han sido reclutadas en todo momento, mediante un simple recuento de MUAPs detectados por un electrodo intramuscular [72][77]. Sin embargo, en nuestro contexto no tiene interés directo, dado que sólo se va a obtener la señal a partir de electrodos de superficie, y esa información que podría ser útil no estará disponible. Baste saber que para los músculos que son de nuestro interés, bíceps y tríceps, el número de MU implicado puede ser de varios cientos [9].

La amplitud de cada MUAP crece con el radio de la fibra elevado a la 1.7 aproximadamente [12], y decrece de manera inversamente proporcional a la distancia entre la fibra y el electrodo. Cada MUAP tiene una duración aproximada de 3 a 20 ms [3], (de 1 a 13 según DeLuca [12]).

El tren de MUAPs tiene una frecuencia que varía con la fuerza que se aplica, en el caso del bíceps, son de 7 a 25 descargas por segundo [3], si bien el tiempo que se sucede entre impulsos es aleatorio. En los mínimos niveles de fuerza, esta tasa se queda entre los 7 y 12 pulsos por segundo, en tanto que con un máximo esfuerzo se llegan a la veintena o más [12]. En condiciones de no fatiga del músculo y de actividad muscular baja o media, las descargas de las motoneuronas no muestran ninguna dependencia entre instantes sucesivos (señal incorrelada siguiendo un modelo de Poisson), en tanto que con una actividad muscular fuerte, la señal sí presenta alguna correlación [3].

Con estos datos se puede sacar una conclusión bastante útil para caracterizar la señal EMG. Si se considera que el tiempo de espera entre dos disparos de una MU es aproximadamente $t = 0{,}1$ (10 disparos por segundo) y que se cuenta con $N = 200$ MU, se puede aplicar la fórmula de Little $N = \lambda t$ y hallar que la tasa de llegadas es muy grande, $\lambda = 2000$, suficiente como para considerar que el proceso es gaussiano, o como para considerar que al menos se va a parecer bastante a un proceso gaussiano (por simple aplicación del teorema del límite central).

3.1.1. Descomposición de la señal EMG

Se han descrito muchos métodos para descomponer una señal EMG en sus MUAP que la componen [84][110][111][113][18][23][22][113], porque tiene especial interés médico para detectar enfermedades. Alguno muy reciente es especialmente interesante, pues considera la señal como una sucesión de símbolos que tienen interferencia entre sí, todo un enfoque según la teoría de la comunicación [137].

También se han desarrollado simuladores informáticos para representar la señal EMG [121], así como simuladores que comprueban la capacidad de los algoritmos descomponedores [146].

Sería de cierto interés para este trabajo descomponer la señal, pero en su gran mayoría los métodos propuestos la adquieren de un electrodo de aguja y eso queda descartado en nuestro sistema.

3.2. Proceso físico químico

El proceso fisicoquímico responsable de la activación de un solo MU se llama bomba de sodio potasio y produce la transmisión del impulso a lo largo de la fibra.

Aun cuando el tejido muscular está en reposo, se puede medir una diferencia de potencial de -90 mV: es el potencial de reposo de la membrana. Esta diferencia se da entre el interior de la fibra y el exterior, y es debido a la presencia de iones de sodio (Na+), potasio (K+) y cloro (Cl-).

En reposo, los iones de potasio están en mayor concentración dentro de la fibra muscular, en tanto que el sodio y cloro están en el exterior. Pero también hay otros iones que contribuyen al potencial de reposo de la membrana, siendo éstos tanto inorgánicos como orgánicos (proteínas, por ejemplo).

Para producir la fuerza muscular, las fibras reciben el impulso de la motoneurona, como ya se ha comentado. Esta sinapsis algo especial conduce a la generación en el músculo de un potencial de acción. La membrana modifica su permeabilidad a los iones, y entran en el interior suficientes iones de sodio como para invertir la polaridad y alcanzar los +30 mV. Pero mientras, la permeabilidad para el potasio también se ha modificado y los iones de potasio vuelven a salir de la célula hasta que ésta se repolariza y se restablece el potencial de reposo.

Este proceso es propagativo, pues las células contiguas también se excitan y entonces la fibra muscular entera se excita. La velocidad de conducción queda determinada por la velocidad de movimiento de los iones, y depende del estado del músculo y de su morfología. Así, en las fibras rápidas el potencial de acción es superior, y la silueta del MUAP es diferente. Las fibras de diámetro grande, también tienen un potencial de acción mayor, y los músculos largos propagan con mayor lentitud la señal. Dicha velocidad de propagación está entre el 1 y los 5 metros por segundo [4] (de 2 a 6 m/s según DeLuca [12]), que es muy inferior a la velocidad de conducción en los nervios (de 30 a 75 m/s).

3.3. Adquisición de la señal EMG

3.3.1. Tipos de electrodos

La medición de la señal EMG se puede realizar con un electrodo de aguja, si es para fines médicos, o bien con un electrodo de superficie si no se admiten técnicas invasivas. Los electrodos de aguja son incómodos para el paciente, y se descartan para su uso en las prótesis mioeléctricas [148].

Un electrodo de superficie recoge la señal de muchos cientos o miles de MUAPs, en tanto que la señal recogida con un electrodo de aguja recoge la influencia de apenas unas pocas fibras, a los sumo unas pocas decenas. La amplitud de la señal recogida con el electrodo de superficie es, evidentemente, mucho mayor que la señal del electrodo de aguja.

Figura 6: Tres tipos de electrodos.

Lo que sí podrían ser de utilidad son electrodos activos, en los cuales están integrados ya circuitos de amplificación a fin de reducir el ruido, aprovechando el principio básico de que cuanto antes se realice la amplificación en la cadena mayor será la relación señal a ruido. La implantación de microelectrodos en el interior del músculo es algo que por ahora sólo se ha experimentado en animales [127].

En la figura 6 se muestran tres tipos de electrodos: electrodo desechable de ECG, electrodo de superficie EMG y electrodos de aguja de EMG. El primero es que el fue usado en el laboratorio. El segundo parece más apropiado, puesto que los contactos tienen forma rectangular y la distancia de separación entre las placas es fija.

3.3.2. Posición de los electrodos

Para adquirir señales biomédicas se utiliza la amplificación diferencial, es decir, se recoge la diferencia de potencial entre dos puntos del cuerpo

humano respecto a una referencia y se elimina la parte común de la señal. Para adquirir la señal de un músculo hacen falta por tanto tres electrodos, dos de ellos se aplican sobre el músculo, y el tercero sobre un tejido que no interviene.

La elección de la posición óptima de los electrodos para tener una buena señal EMG no es sistemática, y los fabricantes de prótesis mismos recomiendan de hecho proceder 'por tanteo' hasta encontrar la mejor posición [112].

La distancia de separación de los electrodos influye en la amplitud y en el espectro de la señal recogida. En el espectro, cuanto mayor es la separación, más bajo es el rango de frecuencias que ocupa la señal.

La separación de los electrodos influye en la amplitud de la señal y en general, separándolos, se amplifica la señal. A cambio se obtiene más interferencia de los músculos cercanos y más ruido en general. Así por ejemplo, Hogan experimentó que pasar de 2 cm a 1 cm de separación entre los electrodos, aumentaba la relación señal a ruido en un 35 % [14]. De Luca también sugiere una distancia de 1 cm.

Sin embargo, llegado un momento la separación de los electrodos hace decaer de nuevo la amplitud. Así, si el par de electrodos se separa en 3 cm en el tríceps, la amplitud de la señal puede decaer hasta en un 25 % [10].

Hershler [10] llevó a cabo un estudio sobre la repetibilidad de las medidas teniendo en cuenta diferentes días y diferentes posiciones de los electrodos, pero su trabajo sólo se centró en estimar la amplitud tras una rectificación y un filtrado paso bajo, lo cual es un estudio demasiado limitado para el sistema aquí propuesto. Sí es válida por contra la idea de sistematizar las medidas y hallar el parámetro que menos varía cuando varían el resto de condiciones, o para conocer cuándo es necesaria una nueva recalibración. Ello merecería un análisis por nuestra parte, ya que poco o nada se ha publicado nada al respecto desde entonces, si bien su importancia es limitada. Es del todo imposible además que los electrodos tengan una posición fija puesto que el músculo se mueve, por lo que hay que contar con la inestabilidad de la señal.

Lo que sí parece obligatorio, en todo caso, es mantener fija la distancia; en nuestro sistema se tomó un estándar que garantizaba siempre la misma separación en 3 cm.

Hay un punto importante que no debe ser pasado por alto, analizado en [7]. Y es que la simple separación de los electrodos ya impone un filtro sobre la señal. Se ha detallado que la actividad mioeléctrica se propaga a una

velocidad de entre $v = 1$ y $v = 5$ metros por segundo. También se ha escrito que una separación razonable de los electrodos es de entre los $d = 0,01$ y los $d = 0,03$ metros, electrodos que recogen la señal para una amplificación diferencial que elimina la señal común. El tiempo τ que tarda un impulso en llegar de un electrodo a otro es:

$$\tau = \frac{d}{v} \tag{1}$$

Una señal senoidal periódica que tuviera la frecuencia $f = \frac{1}{\tau}$ tendría la propiedad de que un pico y el siguiente llegan simultáneamente al primer y al segundo electrodo respectivamente. Siendo el amplificador diferencial, la señal que se recoge en ambos puntos está en fase y es siempre idéntica, y por tanto rechazada. E igualmente son rechazados todos sus armónicos.

En el mejor de los casos para los datos dados, la frecuencia principal se maximiza si $v = 5$ m/s y $d = 0,01$ m: 500 Hz. Así, si el rechazo al modo común del amplificador cumple bien su tarea, la señal perderá irremediablemente las componentes de frecuencia en 500Hz y 1000Hz. Pero el peor de los casos[9] tiene la frecuencia de resonancia en 30Hz, y así se perderán los armónicos en 30Hz, 60Hz, 90Hz.... Así que se tiene un filtro de peine que 'peina' la señal y literalmente la hace trizas. Y aun con la separación de 1 cm, la frecuencia de resonancia puede también quedar en el indeseable punto de los 100Hz.

La respuesta del filtro es:

$$|H(f)| = \left| sin\left(\pi \frac{d}{v}\right) \right| \tag{2}$$

En la figura 7 se representa el caso $d/v = 1s$. En la práctica se pudo comprobar que el filtro no actuaba de una manera tan dramática, posiblemente porque los electrodos distaban mucho de ser puntuales.

3.3.3. Interferencias

La señal se mide aquí con electrodos de superficie desechables, lo cual conlleva que los electrodos en un músculo recogen en menor o en mayor grado la señal de músculos adyacentes[10].

[9] Cuando $v = 1$ m/s y $d = 0,03$ m.

[10] En la literatura inglesa esto se conoce como *crosstalk*

Figura 7: Respuesta del filtro que impone la amplificación diferencial.

Podría pensarse que una manera de estimar si hay mucha o poca interferencia entre las señales de dos músculos es calcular el coeficiente de correlación cruzada, y fijado un umbral, decir si es mucha o poca. Ello no es viable [74], ya que los tejidos entre (y dentro de) los varios músculos que intervienen son anisotrópos e inhomogéneos. Ello causa múltiples difracciones de los vectores del campo eléctrico así que cuando llega la señal al electrodo ya está de por sí incorrelada.

A fin de evitar estas interferencias se ha propuesto una amplificación doblemente diferencial, pero eso ya supondría cinco electrodos por cada señal recogida, un número demasiado alto para ser práctico si se han de recoger varias señales. También la utilización de varios pares de electrodos aplicados al mismo músculo mejoran los resultados. [55] Hudgins investigó distintas configuraciones de los electrodos sobre el bíceps y el tríceps, utilizando los métodos propuestos en su artículo clásico de 1993 [42], llegando a la conclusión de que dos canales muy próximos proporcionan una mejora muy importante sobre el comportamiente habitual. En el trabajo que se propone no se ha planteado ningún paliativo para estas interferencias.

3.3.4. Variación de la señal EMG en distintos brazos

La extracción de señales se realizará en nuestro laboratorio con señales adquiridas de personas sanas. Esto no supone una merma de realismo, puesto que las señales de las personas amputadas no son significativamente diferentes a la de un sujeto normal a la luz de algún estudio que se ha realizado [43]. En

la mayoría de los trabajos similares a éste también se han tomado las señales de personas sanas (por ejemplo [15],[17] o[42]), tomando la precaución de inmovilizar su brazo, de modo tal que al fin y al cabo sea lo mismo.

También se ha hecho algún estudio [139] para detectar alguna diferencia entre el brazo dominante (izquierdo para zurdos, derecho para diestros) y no dominante, llegando a la conclusión de que las diferencias se plasmaban sólo a partir del 80 % del máximo esfuerzo voluntario (MVE), sugiriendo diámetros de las fibras mayores. En todo caso sus resultados no nos afectan tampoco.

3.3.5. Relación fuerza - señal EMG

Se podría pensar que conocidos todos esos datos la relación entre fuerza y señal EMG debería ser conocida y determinística. Sin embargo ello no es cierto, el factor aleatorio es demasiado importante, la señal presenta demasiado ruido y atraviesa sistemas no lineales. Se da como aproximación que tras un filtrado, la relación entre fuerza y amplitud es cuadrática, pero ello es sólo aproximado y válido para señales de baja intensidad, menores que el 50 % de la máxima contracción voluntaria. Este punto será tratado con más detalle más adelante, en la subsección 6.1.

3.4. Digitalización de la señal EMG

Para ser procesada la señal ha de ser digitalizada, es decir, muestreada y cuantificada. Su procesado, a no ser que se diga lo contrario, se hará en bloques, en patrones de tamaño más o menos constante, y las decisiones que se tomen se harán sobre cada patrón por separado. El tamaño del patrón, por razones que más adelante se detallan, será de unos 220 ms.

La tasa de muestreo ha de ser suficiente como para abarcar la mayor parte de la energía de la señal; en nuestro caso se ha elegido una tasa de 1024 muestras por segundo. Por aplicación del teorema de Nyquist, la frecuencia más alta de la señal que es adquirida es de 512Hz. Debería existir un filtro paso bajo de 512Hz para evitar solapamiento de la señal.

Cada muestra de la señal es cuantificada con 12 bits, dando 4096 valores posibles para cada muestra. Ello significa un ruido de cuantificación muy bajo. El rango de entrada del conversor A/D es de -5 a +5 voltios, con lo que la resolución del conversor es $2,4$ mV; suponiendo una ganancia del amplificador de 30dB, significa que se conocerá a la señal EMG con una

precisión de $2,4\ \mu$V.

En la figura 8 se muestra un patrón de la señal tomada de un músculo en contracción. La muestra fue adquirida de la contracción en su fase estacionaria, no en su parte transitoria. El eje de abscisas está en milisegundos (se muestran los 200 primeros milisegundos) y comprende 205 muestras (pues la tasa de muestreo es de 1024). La cabecera analógica tenía tan sólo activado un filtro de red, y los filtros software[11] estaban deshabilitados. El eje de ordenadas presenta la muestra en el rango de -2048 a 2047; es decir, que el pico mayor de la señal presentaba unos 120 milivoltios de amplitud a la entrada del conversor A/D.

Figura 8: Patrón en el dominio del tiempo.

La densidad espectral de potencia de la señal se muestra en la figura 9. Fue hallada operando fuera de tiempo real con el programa Matlab. Se sometió al patrón de la figura 8 a una DFT de 205 puntos con el comando `fft`, tomándose luego su valor absoluto (`abs`) para después ser centrada (`fftshift`). El eje de abscisas presenta la frecuencia desde los 0 hasta los 512 Hz, pues la señal de 200ms constaba de 205 muestras al haber sido muestreada a 1024 muestras por segundo.

3.5. Estructura temporal de la señal EMG

Hasta 1993 se supuso que la señal EMG carecía de toda estructura temporal. Sus características habían de ser extraídas de los parámetros estadísticos que desde luego daba igual tomar en un instante o en otro de la contracción. Sin embargo Hudgins presentó ese año [42] la demostración de lo contrario, frente a todo lo que se podía pensar.

[11] Más adelante se comentará todo ello con más detalle.

Figura 9: Patrón en el dominio de la frecuencia.

La señal mioeléctrica en la fase inicial de la contracción de un músculo muestra una estructura determinística que puede ser aprovechada para el control mioeléctrico de la prótesis. Para cada persona y para cada movimiento que realice, durante los primeros, digamos, 200ms, se genera un patrón de señal mioeléctrica distinto. Tal patrón se demostró que no era debido al movimiento sistemáticamente igual de los electrodos y del brazo, sino a procesos internos.

Hudgins propuso utilizar esta información en la señal temporal para controlar una prótesis mioeléctrica. Para ello dividió el patrón de 240ms en seis subpatrones de 40ms de donde se extraían las características estadísticas. Así, para un movimiento dado, su caracterización venía dada por seis conjuntos de características; los 240ms del patrón comenzaban a contar a partir de que una muestra más alta de un cierto umbral. En su modelo, al rebasar la amplitud de las muestras un cierto umbral, se disparaba el proceso de reconocimiento del patrón, si lo deseáramos aplicar habríamos de tener más cuidado, pues en nuestro sistema los ruidos espúreos no son infrecuentes.

Las figuras 10 y 11 están tomadas de su artículo clásico. La figura 10(a) de la izquierda muestra cuatro fragmentos de señal EMG habiéndose estabilizado la contracción, en tanto que la figura 10(a) de la derecha muestra cuatro fragmentos adquiridos tan sólo en la fase inicial de cada movimiento, antes de que se estabilizara. El movimiento era el mismo en todos los casos, y era una flexión del codo. La imagen (b) de la izquierda y de la derecha representa el promedio de 60 patrones distintos tomados en distintos instantes.

Lo significativo es que la imagen (b) de la izquierda, correspondiente a las señales estables, se asemeja a un ruido blanco, lo cual era de esperar si se suponen todos los procesos independientes; por simple aplicación del

teorema de los grandes números cada muestra es gaussiana e independiente. En cambio, la imagen (b) de la derecha recoge una estructura que parece repetirse en cada contracción, dicha estructura es de amplitud mayor que en el caso anterior y la forma de la onda puede percibirse en los cuatro ejemplos del caso (a) de la derecha.

Figura 10: Experimento de Hudgins (1).

De hecho, la amplitud nos da una idea de cuan significativo es. Siendo suma de procesos gaussianos de media cero, era de esperar que la varianza decreciera con $\frac{1}{P}$ si hay P patrones sumados, es decir que la varianza debería decrecer en 1/60 si los procesos fueran totalmente aleatorios. En los datos que aportó Hudgins, comentaba que en la figura de la izquierda sí había caído a 1/56, en tanto que en la de la derecha sólo lo había hecho a 1/7.

En la figura 11 se muestra cómo diferentes movimientos traen aparejadas diferentes siluetas en la figura. Así, de tomado de izquierda a derecha y de arriba a abajo, las imágenes representan una supinación del antebrazo, una extensión del codo, una rotación del húmero hacia el centro, una contracción del grupo muscular flexor, una flexión de la muñeca , una pronación del antebrazo, una contracción del grupo muscular extensor y una contracción doble del bíceps y del tríceps.

En el laboratorio no se estudió con rigor la estructura temporal de la señal EMG. Sin embargo de manera cualitativa, sí fue perfectamente observable

Figura 11: Experimento de Hudgins(2).

este fenómeno, grabándose varias señales correspondientes al inicio de una flexión del codo, y por simple inspección de las gráficas de las señales se pudo observar claramente el fenómeno.

4. Clasificación de patrones mioeléctricos

El problema que nos ocupa es fundamentalmente un problema de clasificación de patrones. En esta sección se explica el sistema clasificador, que se divide en extracción de características (sección 6) y en clasificación a partir de las características (sección 5). Este es el enfoque clásico del libro de Fukunaga[30] aunque también ha sido explícitamente aplicado al problema de los patrones mioeléctricos en muchas ocasiones, siendo preciso destacar la aportación de K. Englehart [87].

En el primer apartado de esta sección se establece la diferencia entre los problemas de estimación y los problemas de clasificación, y cómo en nuestro trabajo los primeros quedan subordinados a los segundos. En el segundo apartado se plantea formalmente el problema, y en el tercero finalmente se ponen números reales a todo lo anterior.

4.1. Clasificación vs. estimación

Hay dos tipos de control mioeléctrico: *control discreto* y *control proporcional*.

- En el control proporcional, se estima la fuerza producida por el músculo a partir de la señal mioeléctrica para determinar la velocidad de movimiento (o fuerza) del brazo de una manera continua. Si la estimación de la fuerza es buena, y se conoce la mecánica del problema (longitud del brazo natural, peso etc.), por simple aplicación de las leyes de Newton se puede deducir la velocidad y fuerza de la prótesis en cada momento. Este fue el primer enfoque que recibieron las prótesis mioeléctricas, pero hoy esta aproximación ha caído en desuso.
- En el control discreto, la señal mioeléctrica es clasificada en una de las K clases posibles definidas para producir uno de los K estados de la prótesis, tal como 'abrir mano' o 'pronar muñeca'. El movimiento lo lleva a cabo la prótesis en este caso a una velocidad (o fuerza) constante.

El control proporcional plantea un problema de *estimación*, el control discreto plantea un problema de *clasificación*[12]. Cuando se estima la fuerza, se trata de que la diferencia entre el estimador y el estimado sea la mínima

[12]En realidad se suele llamar problema de *detección*.

posible, cuando se clasifica un patrón se trata de que la probabilidad de decidir una clase siendo la voluntad del usuario otra sea mínima. Estimar es maximizar una SNR, clasificar es minimizar una probabilidad de error.

En este trabajo se aborda principalmente el problema de la clasificación (o detección), y se propone un control discreto de la prótesis. Sin embargo, la teoría propuesta para una buena estimación de la fuerza no ha de ser desdeñada, y sí recibe la debida atención en este proyecto. Los problemas de estimación y de detección van ligados, y ya desde 1977 se propuso [8] una clasificación de los estados de la prótesis mioeléctrica pero tratada desde un punto de vista de la estimación. Y por supuesto, los resultados han sido aprovechados también como características para el control discreto de prótesis. En este trabajo, la estimación de la fuerza arrojará una característica más de la señal, y por tanto queda subordinado al problema de clasificación.

4.2. Representación formal de la señal

La clasificación de señales, su compresión y la eliminación de ruido en ellas son ejemplos de problemas clásicos en la teoría de la señal. El problema que nos atañe, que incluye todo ello, cae dentro del dominio de la teoría de la señal, por lo que se tratará de presentar el problema de una manera más formal. Ello será lo que ocupe este apartado.

Definamos *patrón* como un vector[13] de N variables $\vec{x} = [x(1), \ldots, x(N)]$. Estas variables en principio son las N muestras consecutivas de la señal. Cada patrón puede decirse que pertenece a una de las K *clases* posibles, y así $y = y_0$, $y = y_1$ .. ó $y = y_k$. Las K clases son los K estados de la prótesis, en principio se considerará $K = 5$ ó $K = 7$. y_0 será el estado de reposo, las otras dos o tres parejas corresponderán a los dos o tres grados de libertad que se permitan a la prótesis. Por ejemplo, según la tabla 1 de la subsección 4.3.2

Sea el *espacio de entrada* X el conjunto de valores que puede tomar el vector patrón $\vec{x}$ y sea el *espacio de la señal de salida* Y el conjunto de valores que pueda tomar $\vec{y}$. El espacio X tiene dimensión N, el espacio Y tiene dimensión K.

[13]Las variables vectoriales se designan en este trabajo con una flechita sobre ellas, así por ejemplo un vector es '$\vec{x}$' y no 'x'. Igualmente, se reservará el uso de las mayúsculas para las variables matriciales, y el acento circunflejo queda reservado para las variables estimadas.

En el caso del control proporcional Y es un espacio continuo y puede escribirse que $\vec{y} \in Y \subseteq \Re^K$. Sin embargo, en el control discreto, se impone a $\vec{y}$ una restricción, y es que, a diferencia de $\vec{x}$, no puede ser una combinación lineal de los vectores de su espacio, sino que ha de tomar *uno* de los valores discretos que se proponen, llamados *clases*. Formalmente, si $\vec{x} \in X \subseteq \Re^N$, $\vec{y} \in Y = \{\vec{y_1}, ..., \vec{y_K}\}$. De manera menos formal, se podrá escribir que las clases son y_0, y_1 etc. sin la notación vectorial.

Clasificar una señal $\vec{x}$ es establecer una aplicación entre X e Y, que asigne a cada patrón de entrada $\vec{x}$ una clase y. $d : X \longrightarrow Y$. La voluntad del usuario se traduce en unos patrones a partir de los cuales el clasificador propuesto ha de estimar su clase de salida. Entenderemos por *rendimiento* del clasificador al porcentaje de aciertos. Un porcentaje de aciertos inferior al 80 % se considera malo; más adelante se precisará cómo se mide el porcentaje de aciertos. Evidentemente, basta con reducir el tamaño K (es decir, el número de grados de libertad) para que el rendimiento aumente, pero esta solución trivial tampoco nos satisface.

Establecer una aplicación directa de los datos de entrada en el espacio de la señal de salida no es razonable. Se podría clasificar la salida y como función de las N componentes de $\vec{x}$, pero teniendo en cuenta que aproximadamente $N = 220$ en el sistema propuesto, se ha de descartar al instante.

El espacio de entrada es realmente grande para ser tratado directamente. *Grande* tiene sentido en términos de computación. Se considerará un número de datos como grande si no es posible procesarlo en tiempo real, pero también se considerará grande si han de proporcionársele demasiados patrones de entrenamiento de manera supervisada, tantos como para aburrir al ser humano que haya de introducírselos. En ambos sentidos el patrón tiene una dimensión demasiado grande, como se verá en el siguiente apartado.

Por todo ello se define un espacio de características, F[14] de dimensión M, entre el espacio de la señal de entrada y el espacio de la señal de salida, donde $M < N$. El esquema se presenta en la figura 12.

Un *extractor de características* se define como el sistema que establece la aplicación de X a F, y el *clasificador* como el que lo hace entre F e Y, aunque por extensión a todo el proceso se le conocerá como "clasificador".

En general se asumirá la presencia de una colección de datos de entrenamiento, que consiste en P pares de señales de entrada y de clases de salida $(\vec{x}, y)$: $(\vec{x}_1, y_1), (\vec{x}_2, y_1), (\vec{x}_3, y_2)$, ... A fin de evaluar la generalización de la

[14]Del inglés *feature*

Figura 12: Esquema de la clasificación de patrones

capacidad de un clasificador, se asume que esta colección de datos proviene de un modelo probabilístico, y que se trata de realizaciones de procesos estocásticos. En la práctica serán más interesantes los pares de entrenamiento $(\vec{f}, y)$, pues el clasificador habrá de operar sobre las características del espacio F.

Finalmente se define el *número de canales* como L si hay L amplificadores diferenciales. Así, en un momento dado se contará en realidad con los patrones $\vec{x}_1, \ldots, \vec{x}_L$, donde ahora los subíndices denotan el canal del que procede el patrón. Una representación adecuada en el momento de la decisión sería en forma de matriz X de tamaño $N \times L$ con todas las muestras con las que se cuenta, pero el análisis se torna excesivamente complicado y se seguirá manteniendo la notación en vectores.

4.3. Patrones mioeléctricos.

4.3.1. Espacio de la señal de entrada X

Como ya se ha comentado los patrones tienen una longitud aproximada de $N = 225$ muestras. En todos los sistemas que aparecen en la literatura el número de muestras en cada patrón es en principio fijo, sin embargo en el que se presenta se es más flexible y se toleran pequeñas variaciones sin que se altere el rendimiento del clasificador. En 1979, sin embargo, ya se proponía un modelo cuyo tamaño del patrón no era fijo sino función de la certidumbre que se tenía para tomar una decisión [11].

La tasa de muestreo en el sistema presentado es de 1024 muestras por segundo. Las muestras llegan periódicamente al ordenador, y periódicamente

debe el *software* recoger grupos de muestras que es lo que llamamos *patrones* o segmentos. En nuestro sistema se programa un temporizador, pero también existiría la posibilidad de que al llegar un determinado número de muestras se activara una interrupción; que siendo la llegada de datos síncrona no debería haber ninguna diferencia.

Si hubiera algún desfase, en un sistema operativo monotarea se podría garantizar la recogida de patrones de la longitud exacta programando las interrupciones a bajo nivel (lenguaje ensamblador). Sin embargo, siendo Windows un sistema operativo (S.O.) multitarea, se utilice la alternativa de las interrupciones o la del temporizador, nada nos garantiza la puntualidad, pues en ambos casos se trata de un evento más que va a la cola de mensajes de Windows. Si el S.O. en ese momento está especialmente ocupado, no se podrá evitar un pequeño retraso.

En principio está pensado que las muestras se recojan cada $T = 220ms$, que a $f_s = 1024$ muestras por segundo suponen unas $N = 225$ muestras por patrón; en la práctica estos valores pueden variar en 5 ó 6 muestras arriba o abajo. Si los parámetros característicos de la señal quedan normalizados, entonces se podrán tomar como equivalentes. Así por ejemplo en vez de computar la energía total de un patrón, se calcula la energía *por muestra* del patrón (o valor RMS). No se puede comparar la energía total de patrones de distinta longitud, sí se puede comparar el valor RMS aunque la longitud sea ligeramente distinta.

Cierto es que que si fuera necesario se podría establecer una longitud fija del patrón a base de truncar los segmentos demasiado largos y rellenar de alguna manera los demasiado cortos, pero eso es un retroceso frente a la flexibilidad de un modelo que se adapta a los ligeros cambios. En todo caso, desperdiciar unas pocas muestras no es crítico en el funcionamiento del sistema.

Ha habido alguna ocasión en que se ha propuesto que el tamaño N del patrón sea variable según convenga. Así Fleisher [11] propuso un algoritmo en el que iba procesando las muestras hasta que poseía suficiente certeza sobre la decisión del sujeto, y en ese momento ejecutaba la decisión. En caso de no llegar a un certeza tras un tamaño tope del patrón se forzaba la decisión. Con esto se mejoraba el tiempo de respuesta del dispositivo a igualdad de probabilidad de error.

Por otra parte puede considerarse la aplicación de una ventana al patrón a fin de suavizar la decisión, pero ello sólo es beneficioso dependiendo de la característica considerada. Farry [66] por ejemplo aplicaba una ventana de

Hamming a cada segmento de la señal mioeléctrica.

4.3.2. Espacio de la señal de salida Y

En este trabajo se considera $K = 5$ ó $K = 7$ estados, que aquí se definen en la tabla 1. En generar decidir cuántas clases definir depende de la fiabilidad que se desee, aquí se ha hecho de manera aproximada, pero hay métodos que calculan cuántos y qué grupos se pueden crear. Por ejemplo, en [32] se discute cúantos grupos se deben crear (*clustering*)para la discriminación de envolventes de la señal EMG.

Clase y_k	Función
0	Reposo
1	Abrir mano
2	Cerrar mano
3	Extensión
4	Flexión
5	Pronación
6	Supinación

Cuadro 1: Correspondencia entre clases y funciones

4.3.3. Espacio de características F

El espacio de la señal de entrada es muy grande porque hacen falta muchos patrones de entrenamiento y porque el ordenador ha de tener tiempo como para procesar los datos en tiempo real.

En primer lugar, Si $N = 225$ y cada muestra tiene un valor limitado a 4096 posibilidades, significa que el espacio de entrada tiene limitado su tamaño a $4096^{225} \approx 10^{800}$ vectores. Por muchos patrones de entrenamiento que se introduzcan, siempre serán despreciables frente a aquello a lo que quieren representar. Hay que reducir la dimensionalidad acordemente con el número de patrones de entrenamiento que se hayan de dar, de manera que sean representativos. En este sistema, hay unos 5 patrones por segundo, lo que significa que si un tiempo razonable de entrenamiento es de varios minutos, no se podrá contar más que con unos pocos miles de patrones de entrenamiento. La *maldición de la dimensionalidad* impone que el número de patrones de entrenamiento que han de conocerse en un espacio de tamaño N

ha de ser mucho mayor que N si se desea que la estimación no sea catastrófica. Si queremos relacionar las variables entre sí harán falta aproximadamente la mitad de $N \times N$, es decir, unas 25000 variables, un número demasiado grande.

Y en segundo lugar el clasificador habrá de procesar los datos en tiempo real y el procesado no tiene por qué realizar tan sólo operaciones sencillas. Posiblemente habrá, por ejemplo, inversiones de matrices; piénsese en lo costoso que es, en términos de cálculo, invertir una matriz de 100×100. Es obligatorio operar no con el patrón, sino con un conjunto de características que sea manejable.

Por tanto, el clasificador no puede decidir a partir del patrón de entrada directamente, sino que ha de hacerlo a partir de un conjunto de características representativas del patrón. Hay sin embargo algún sistema propuesto recientemente que opera con la señal en bruto [132], enviándose las 200 muestras que toma por cada patrón directamente a una red neuronal, pero no da cuenta del tiempo de procesamiento ni de los resultados.

Casi nada se puede decir del tamaño del vector de características $\vec{f}$, en principio lo supondremos limitado a la velocidad con la que puedan ser procesados los datos. La naturaleza de las características se abordará más adelante.

4.3.4. Número de canales L

El número de canales considerado es $L = 2$. La cabecera analógica P4 está siendo mejorada para permitir más canales, sin embargo su inclusión no se considera un avance decisivo puesto que con un par de amplificadores ya se puede atacar a los dos músculos más importantes, bíceps y tríceps.

Cabe considerar la toma de varios canales atacando a un mismo músculo, lo cual proporciona una señal más libre de ruido y de interferencia con otros músculos si es sabido aprovechar adecuadamente. Sin embargo, tampoco se ha considerado esta posibilidad.

Símbolo	Significado
$x(t)$	Señal de entrada.
$\vec{x}$	Vector de entrada o patrón. Por extensión, vector de características.
$\vec{f}$	Vector de características.
y_k	Clase o función de la prótesis.
N	Tamaño del patrón o vector de entrada.
M	Tamaño del vector de características.
K	Número de estados.
L	Número de canales.
F	Fuerza del músculo.

Cuadro 2: Símbolos empleados a lo largo del texto

5. Clasificador de características

5.1. Tipología de los métodos clasificadores.

La teoría de la clasificación distingue entre tres tipos de clasificadores. Se comentará en primer lugar la aproximación estadística (o teoría de la decisión), después la aproximación estructural (sintáctica) y finalmente la de aprendizaje (redes neuronales).

1. **Aproximación estadística.** Está basado en el análisis estadístico de los datos a ser clasificados. A partir de la colección de datos de entrenamiento se intenta inferir la función de densidad de probabilidad para cada clase. Todos los patrones que se sabe que pertenecen a una clase (porque son patrones de entrenamiento, que por eso mismo se conocen) no son sino realizaciones de un proceso estocástico.

   A partir de ello, según algún criterio predeterminado, se divide el espacio de la señal de entrada X. Y todo patrón $\vec{x}$ cuya clase y se ignore y se desee clasificar, será asignado a una clase y u otra según pertenezca a una región del espacio o a otra. Hay varios criterios para dividir el espacio de la señal de entrada, pero en general se distinguirá entre clasificadores *paramétricos* y clasificadores *no paramétricos*. Los clasificadores paramétricos asumen una función de densidad de probabilidad parametrizada para los datos, en tanto que los no paramétricos no presumen forma de la función alguna.

2. **Aproximación sintáctica.** La aproximación sintáctica, por otra parte, esta basada en utilizar la estructura de las patrones y la interrelación entre las componentes de un patrón. El reconocimiento de patrones sintáctico implica identificar componentes significativas o *primitivas* de los patrones, y desarrollar una sintaxis formal o *gramática* describiendo la síntesis de los patrones a partir de sus primitivas.

   El paralelismo con la teoría del lenguaje es clarificador: *primitivas* = palabras, *señales* = sentencias, *descripción estructural*=gramática etc. La aproximación estadística asume muy pocas cosas acerca de la estructura de los datos bajo consideración, lo métodos sintácticos tratan de hacerlo tan bien como pueden. En todo caso, lo que es natural para los humanos es muy complejo de procesar en las máquinas y es difícil programar analizadores sintácticos; frente a los métodos estadísticos que se prestan perfectamente a la computación (piénsese en lo complejo que es hacer que una máuina comprenda el lenguaje humano).

El problema realmente difícil es extraer las características de la señal que permitan este análisis, la clasificación luego debería ser más sencilla. Tampoco está claro que la señal esté sujeta a un tipo de estructura, por lo que estos métodos de reconocimiento de patrones no son en principio muy apropiados en nuestro problema y se comentan a fin de dar generalidad a las soluciones presentadas; para mayores detalles se puede consultar [28]. Esta aproximación tan sólo se utiliza en técnicas aisladas de procesado de la señal de vídeo y similares, donde hay formas bien delimitadas, bordes etc. que han de ser reconocidos.

Está mucho más allá del objetivo de este proyecto el investigar si la señal EMG se puede modelar o no de una forma jerárquica de primitivas y gramática, así que en lo que sigue se dejará de lado.

3. **Aproximación basada en aprendizaje.** Los algoritmos capaces de aprender toman la forma de redes neuronales artificiales o de lógica difusa[15]. Las redes neuronales artificiales pueden ser llamadas también determinísticas en oposición a las estadísticas, porque los algoritmos de aprendizaje no asumen nada acerca de las propiedades estadísticas de los patrones del espacio de la señal de entrada.

5.2. Clasificadores estadísticos

Sobre la notación. En esta sección se supone que ya se han escogido un grupo de características $\vec{f}$ que representen a la señal. La entrada del clasificador tal y como se ha definido hasta ahora venía representada por estas características $\vec{f}$, y el clasificador es el bloque que da una salida y a partir de una serie de características $\vec{f}$. Sin embargo, dado que la elección de la letra $\vec{f}$ para nombrar a un vector del espacio de entrada es muy desafortunada, en lo sucesivo se utilizará la letra $\vec{x}$ para designar a un vector de entrada del clasificador. Una vez aclarado que el clasificador no actúa sobre el espacio de entrada sino sólo sobre un extracto del mismo, se puede proseguir con la notación habitual en x. Del mismo modo, donde se lea *patrón*, según el contexto habrá que entender realmente *patrón* como se ha definido o bien *características* en un sentido más amplio.

[15]En inglés, *fuzzy logic*

5.2.1. Proceso de diseño del clasificador estadístico

Existe un proceso bien establecido para diseñar el clasificador (en el sentido amplio de la palabra)[30]. En el diseño del clasificador se deberían dar los siguientes pasos:

1. *Estimación no paramétrica del error sobre el espacio de entrada X.* En primer lugar, se recogen los datos, y las muestras son normalizadas y registradas. Después se debería medir la separabilidad de clases entre los datos recogidos mediante la estimación del error de Bayes en el espacio de la señal de entrada. En principio no se asume forma matemática alguna para la estructura de los datos, por lo que la estimación debe ser no paramétrica. Si el error de Bayes es mayor que el error del clasificador que se desea hacer, entonces es mejor no seguir; el extractor de características y el clasificador empeorarán aun la situación.

2. *Análisis de la estructura de datos y estimación del error sobre el espacio de características.* Si de momento el error es tolerable, se procede a extraer características, asignar grupos, hacer análisis estadísticos y de modelado etc. Cada vez que se extraiga un conjunto de características se estimará el error de Bayes sobre él y se comparará con el error de Bayes sobre el espacio de la señal de entrada. La diferencia entre ambos daría una medida de la bondad del extractor de características, procediéndose a su rediseño si tal diferencia no fuera tolerable.

3. *Diseño del clasificador y error final. Una vez que se ha entendido la estructura de los datos, se escoge un clasificador.* El clasificador normalmente es paramétrico, que siendo más sencillo es más apropiado para operar en tiempo real. Finalmente se compara el error final con el error de Bayes sobre el espacio de características; y si la diferencia es inaceptablemente alta ha de rediseñarse el clasificador.

En el sistema que aquí se propone no se parte de cero, sino que se aprovecha todo el conocimiento anterior sobre la clasificación de patrones mioeléctricos, pero en rigor, se debería proceder sistemáticamente no dando nada por supuesto.

5.2.2. Aproximación bayesiana

El problema en reconocimiento de patrones estadísticos es hallar las funciones de decisión óptimas, que se acerquen más al rendimiento del 100 %, y

que hagan coincidir la voluntad del sujeto con la respuesta del clasificador. *El clasificador bayesiano se define como aquel que maximiza el rendimiento.*

Las medidas de $\vec{x}$ e y son consideradas en un marco probabilístico en esta aproximación y son vistas como observaciones de las variables aleatorias X e Y. Lo que se desearía conocer es la probabilidad condicionada de que conociendo un patrón observado $\vec{x}$ pertenezca a la clase y_k. Obrando así con todas las clases, se escogerá evidentemente la que arroje una probabilidad de acierto mayor.

Las probabilidades condicionadas *a posteriori* para cada clase se denotan como

$$P(y_k|\vec{x}), k = 1, \ldots, K \tag{3}$$

No se conocen, pero a partir de un conjunto de pares de entrenamientos se habrán de inferir. Con una colección infinita de pares de entrenamiento se debería obtener un rendimiento máximo, como resultado de un conocimiento más preciso de la variable aleatoria.

La aproximación bayesiana exige que las funciones de densidad de probabilidad sean constantes en el tiempo. Esta hipótesis no es aceptada en cierto sentido por las aproximaciones sintácticas, donde la aplicación entre entrada y salida no es unívoca y un mismo patrón podría ser asignado a clases distintas en momentos diferentes, o por la aproximación de aprendizaje donde el algoritmo se podría adaptar a las nuevas situaciones. Piénsese que puede tomarse como información útil las últimas decisiones tomadas, y el saber que se ha tomado una misma decisión en los últimos patrones hace pensar que dicha decisión será más probable que vuelva a suceder [15], pues es de esperar que haya una cierta suavidad en las decisiones del sujeto a lo largo del tiempo, y a tal efecto se reduciría el umbral de decisión para esa clase. Si bien en este proyecto se utiliza la aproximación bayesiana, sí se ha incluido cierto suavizado a la hora de tomar decisiones.

En la realidad, las funciones de densidad de probabilidad tampoco son constantes y se requiere un calibrado periódico o idealmente, de manera continua [107][109][114][138] y automática [135][78][92]. La aproximación bayesiana por definición maximiza el rendimiento, pero las hipótesis de partida no se verifican y por tanto no se puede garantizar ese máximo en la realidad. En todo caso, aunque sí permanecieran constantes, no se dispone de la colección completa de pares de entrenamiento, con lo que el conocimiento de las funciones de densidad de probabilidad tampoco sería perfecto.

Pero en principio se supondrá que el dispositivo está bien calibrado y las

funciones de densidad de probabilidad son las mismas durante la operación del equipo que durante el entrenamiento. Las probabilidades *a posteriori* $P(y_k|\vec{x})$ no se pueden estimar directamente, si bien se hará a partir de las probablidades *a priori* $P(y_k)$ y la función de densidad $p(\vec{x}|y_k)$ usando el teorema de Bayes:

$$P(y_k|\vec{x}) = \frac{P(y_k)p(\vec{x}|y_k)}{p(\vec{x})} \tag{4}$$

donde $p(\vec{x})$ es la función de densidad de probabilidad de los vectores del espacio de entrada. Se calcula como:

$$p(\vec{x}) = \sum_{j=1}^{K} P(y_j)p(\vec{x}|y_j) \tag{5}$$

Dado que permanece constante para todas las $P(y_k|\vec{x})$, puede ser ignorado a efectos de discriminación.

- Clasificación entre dos clases. Si se desea saber si un patrón pertenece a una clase $k = 1$ o a una clase $k = 2$, se suele atender a la *tasa de similitud*[16], que es en este caso concreto:

  $$\ell(x) = \frac{p(\vec{x}|y_1)}{p(\vec{x}|y_2)} \tag{6}$$

  y se compara con el cociente $P(y_2)/P(y_1)$. Más convenientemente se puede definir la tasa *menos-logarítmica de similitud* como $h(x) = -\ln \ell(\vec{x})$. Esta $h(\vec{x})$ es la *función discriminante*.

- Clasificación entre más clases. El criterio habitual para dos clases ya no sirve. Si se extiende a más variables, se puede tomar como función discriminante a:

  $$h_k(\vec{x}) = p(\vec{x}|y_k)P(y_k), k = 1, \ldots, K. \tag{7}$$

  y el criterio para decidir es escoger la $h_k(\vec{x})$ mayor.

Para clarificar se pone un ejemplo sencillo. En la figura 13 se ha dibujado el problema típico en que se conocen las funciones de distribución de probabilidad y se ha observado una realización del patrón x. Se ignora a qué clase pertenece, y el objetivo es decidir bien a cuál de ellas corresponde.

[16]En inglés, *likelihood*

Se han dibujado con Matlab tres campanas gaussianas (aunque en este punto todavía no se ha supuesto que las distribuciones son gaussianas) de medias -1.5, 1 y 1.8 y varianzas 1, 1.5 y 0.8 respectivamente. Para un punto dado del eje x, se tienen tres valores de esta probabilidad h_k, que se han rodeado con un circulito. En este caso de 'patrón' unidimensional, se escogería el valor mayor de h_k. Considerando $P(y_k)$ iguales para todos los caso, se tiene que $h_0 = 0{,}0551$, $h_1 = 0{,}2510$ y $h_2 = 0{,}1305$. Es decir, que lo más probable es que la decisión correcta sea la campana central. Es más, también se sabe que la probabilidad de que acertemos es de un 57 %.

Figura 13: Típico problema de decisión.

El criterio bayesiano garantiza que se ha minimizado la probabilidad de error. Esa probabilidad de error se llama *error bayesiano*:

$$\varepsilon = \sum_{k=1}^{K} P(y_k)\varepsilon_k. \tag{8}$$

con

$$\varepsilon_k = \int_{L_k} p(\vec{x}|y_k)dx = \sum_{\vec{x}\subseteq L_k} p(\vec{x}|y_k) \tag{9}$$

Aquí L_k indica la región de x para la cual se debería haber escogido y_k y no se hizo.

Cuando se ignora la distribución de probabilidad de las clases, las probabilidades *a priori* se hacen iguales: $P(y_k) = 1/K$ con $k = 1, \ldots, K$, y entonces la función discriminante se reduce a conocer $p(\vec{x}|y_k)$.

No debería ser muy difícil estimar las $P(y_k)$, sin más que observar las tareas cotidianas del amputado y observar con qué frecuencia invoca a una u otra función de la prótesis mioeléctrica. Pero en general, y dado que no es posible suponer que el amputado abrirá la mano más veces de las que flexionará el codo, por ejemplo, se va a suponer que las clases son equiprobables.

Ahora bien, ¿qué forma tiene la función $p(\vec{x}|y_k)$?¿Cómo se puede modelar a partir de los pares de entrenamiento?

5.2.3. Clasificadores gaussianos bayesianos.

El clasificador gaussiano bayesiano propone que las funciones de densidad de probabilidad condicional $p(\vec{x}|y_k)$ son normales multivariadas, y que la observación de algunos pares de entrenamiento sirve para fijar los parámetros que las modelan, el vector de medias y la matriz de covarianza. Aunque para algunas colecciones de datos es dificil hacer estas suposiciones, la distribución normal es muy apropiada en la práctica como resultado del *teorema del límite central* [65]. Formalmente la función es:

$$p(\vec{x}|y_k) = \frac{1}{2\pi^{N/2}|C_k|^{1/2}} exp\left[-\frac{(\vec{x}-\vec{m}_k)^T C_k^{-1}(\vec{x}-\vec{m_k})}{2}\right], k = 1, \ldots, K \quad (10)$$

que queda completamente determinada con el vector de medias $\vec{m}_k$ y la matriz de covarianza C_k, definidos por:

$$\vec{m_k} = E_k[\vec{x}] \quad (11)$$

y

$$C_k = E[(\vec{x}-\vec{m}_k)(\vec{x}-\vec{m}_k)^T] \quad (12)$$

donde $E_k[.]$ denota el operador esperanza sobre los patrones de clase y_k y $|C_k|$ indica el determinante de la matriz C_k. La covarianza entre dos variables $x_k(i)$ y $x_k(j)$ expresa su tendencia a variar parejamente, y está acotado entre $-\sigma_k(i)\sigma_k(j)$ y $+\sigma_k(i)\sigma_k(j)$, tomando el valor 0 cuando las variables aleatorias son independientes.

Figura 14: Distintas covarianzas.

La matriz de covarianza es la matriz que contiene todas las relaciones entre las características del patrón mioeléctrico, e indicará su grado de independencia o su grado de correlación. Evidentemente, es una matriz simétrica, pero además será una matriz diagonal si cada característica del patrón $\vec{x}$ es independiente estadísticamente de las demás. En el caso de los patrones

mioeléctricos, no se puede contar con ello. En la figura 14 se muestran diveros valores de covarianzas. En los casos de los extremos, conocido un punto en el eje x se conoce al instante el punto del eje y, en el caso central, la covarianza es cero porque las variables aletorias son independientes, y la figura es (o mejor dicho, debería ser) una circunferencia.

Al multiplicar por la matriz de covarianza, tan sólo se está haciendo una transformación lineal [17] a fin de normalizar. Piénsese en la función de distribución gaussiana, que en su exponente normaliza la variable aleatoria para que tenga media 0 y varianza 1, $exp(\frac{(x-\eta_x)^2}{\sigma_x^2})$.

En definitiva, conceptualmente no es más que una función de densidad de probabilidad gaussiana para cada clase definida, sólo que en vez de ser función de una única variable, es función de un vector de variables, la media es sustituida por un vector de medias y la varianza es sustituida por la matriz de covarianza.

La media y la covarianza han de ser estimados a partir de los patrones de entrenamiento. Teniendo P patrones de entrenamiento en una clase, se podría tomar como estimador de la media de esa clase a:

$$\hat{\vec{m}}_k = \frac{1}{P}\sum_{p=1}^{P}\vec{x}_{kp} \tag{13}$$

a partir del cual se estimaría la covarianza. El estimador es insesgado y consistente: la media de la estimación es la media del estimado y la varianza de la estimación decrece con el número de patrones de ejemplo, cuantas más muestras mejor es la estimación.

También podría haberse utilizado otro estimador para la media, quizás más apropiado para el caso de los patrones mioeléctricos, un estimador que fuera variando constantemente la estimación según se fuera recogiendo patrones, pero no se ha considerado en los algoritmos propuestos en este proyecto.

[17] *Transformaciones lineales:* Una transformación lineal del patrón $\vec{x}$ en otro $\vec{z}$ se puede expresar como $\vec{z} = A^T\vec{x}$ donde A es una matriz de tamaño $N \times N$. Fácilmente se deduce que $\vec{m}_z = A^T\vec{m}_x$ y $C_z = A^T C_x A$.

El único requisito es que A sea inversible. La *transformación ortonormal* es la transformación lineal que diagonaliza a la matriz de covarianza. La matriz de transformación es la matriz de los autovectores de C. Ya que los autovalores son los que maximizan la distancia $(\vec{x} - \vec{m}_k)^T C^{-1}(\vec{x} - \vec{m}_k)$, se dice que se están seleccionando las componentes principales de la distribución como los nuevos ejes de coordenadas. Los autovalores son las varianzas de la variables transformadas. En la transformación, la distancia euclídea se conserva. En el campo de los patrones mioeléctricos se ha utilizado a menudo el análisis PCA para discriminar las funciones de la prótesis mioeléctrica [87][61][106].

La matriz de covarianza igualmente se puede calcular como:

$$\hat{C}_k = \frac{\sum_{p=1}^{P}(\vec{x}_{pk} - \vec{m}_k)(\vec{x}_{pk} - \vec{m}_k)^T}{P-1} \tag{14}$$

Si bien hay que tener mucho cuidado al decidir el número de patrones P de entrenamiento. $\vec{x}$ y $\vec{m}_k$ son vectores de N componentes, sí, pero C_k es una matriz de $\frac{N\times N}{2}$ componentes independientes. Para que la estimación de C_k sea buena son necesarios al menos $P = \frac{N\times N}{2}$ patrones de entrenamiento, y no N como podría parecer. Eso dispara el gasto computacional, y en cuanto crezca N se disparará el número de patrones de entrenamiento necesarios. Si no se extrajeran características del patrón N y se consideraran las 220 componentes, harían falta como mínimo unos 25000 patrones de entrenamiento por cada clase, es decir, que el entrenamiento supondría muchas horas[18].

La aproximación bayesiana toma como criterio el cálculo para cada clase K de $h_k(\vec{x}) = p(\vec{x}|y_k)P(y_k)$, y ahora se ha supuesto $p(\vec{x}|y_k)$ como gaussiano. Para hacerlo más manejable, se utiliza la versión menos-logarítmica de las funciones.

$$h_k(\vec{x}) = \ln p(\vec{x}|y_k) + \ln P(y_k) \tag{15}$$

Con lo que la función discriminante queda como:

$$h_k(\vec{x}) = \ln P(y_k) - \frac{N}{2}\ln 2\pi - \frac{1}{2}\ln|C_k| - \frac{1}{2}[(\vec{x} - \vec{m}_k)^T C_k^{-1}(\vec{x} - \vec{m}_k)] \tag{16}$$

El término $\frac{N}{2}\ln 2\pi$ es igual para cada clase, de modo que no ofrece discriminación y puede ser eliminado. Esta *función discriminante* h_k será calculada para cada clase, escogiendo como más verosímil la función que sea mayor (o escogiendo la función menor cuando se multiplique a la función por -1). En lo que sigue, se irán sucediendo simplificaciones en esta función, obteniendo expresiones cada vez más sencilas.

Función discriminante cuadrática. $P(y_k)$ se tomará constante para todas las clases (teniendo en mente que esto es inadecuado para nuestro problema). Multiplicando también a todas las clases por -2, se obtiene la función discriminante QDF[19].

$$h_k(\vec{x}) = \ln|C_k| + (\vec{x} - \vec{m}_k)^T C_k^{-1}(\vec{x} - \vec{m}_k) k = 1, \ldots, K \tag{17}$$

[18]El hecho de que el paciente usuario tenga agujetas en tan largo tiempo, suponiendo que aguantara tal esfuerzo, no es una frivolidad, pues el cansancio muscular modifica las características de la señal EMG, como se ha comentado numerosas veces. El entrenamiento quedaría invalidado.

[19]Del inglés, *quadratic discriminant function.*

Dado que la expresión 17 incluye términos cuadráticos en $\vec{x}$, se dice que es una *función discriminante cuadrática* (o hipercuadrática, si se desea recalcar que $\vec{x}$ es un vector). La superficie que divide el espacio de entrada es la que minimiza las probabilidades de error, y es una superfice cuadrática (paraboloide, elipsoide o hiperboloide).

Obsérvese que se ha multiplicado por un número negativo; eso significa que al evaluar todas las funciones discriminantes ha de escogerse la *menor* y no la mayor como antes. Por eso en vez de funciones discriminantes se puede hablar de 'distancias' como sinónimo.

En la figura 15 se representa un sencillo ejemplo visual. El vector $\vec{x}$ tiene sólo dos variables y hay sólo dos clases posibles; en el gráfico en 3D se representa la función de densidad de probabilidad en el eje z para cada valor de $\vec{x}$ y para cada una de las dos clases. Las clases se reconocen muy bien por los dos promontorios. Los datos no se apiñan en cada clase de igual manera y se observa que la covarianza en cada clase no es nula.

Figura 15: Función de densidad de probabilidad de dos variables

En la figura 16 se muestra cómo ha quedado dividido el espacio de la señal de entrada. A partir de la función de distribución de probabilidad dada en la figura 15, se calcula el umbral a partir del cual decidir una u otra clase, y se aprecia cómo la curva divisoria no es recta.

Función discriminante de Mahalanobis. El segundo término de la función 17 por sí solo se utiliza también como distancia entre un patrón y la distribución que caracteriza a un grupo, llamándose *distancia de Mahalanobis*:

$$h_k^2(\vec{x}) = (\vec{x} - \vec{m}_k)^T C_k^{-1} (\vec{x} - \vec{m}_k) k = 1, \ldots, K \tag{18}$$

Figura 16: El espacio no queda dividido con rectas.

En realidad lo único que se ha hecho es eliminar un término de energía de la QDF que al fin y al cabo no dependía del patrón. La distancia de Mahalanobis tiene la ventaja de que todos los parámetros quedan normalizados, no importando la escala en que estén (es posible que unos parámetros del patrón estén en microvoltios y otras en voltios, por ejemplo.) y no importando la correlación de las componentes. Ello es debido a que se conserva la transformación lineal normalizadora que supone multiplicar la covarianza, y a que aún se conserva de la regla 17. Esto es una ventaja muy importante, y se puede observar gráficamente más adelante al comentar las carencias del análisis discriminante lineal.

Función discriminante lineal. Si se desarrolla la regla de decisión 17:

$$h_k(\vec{x}) = \ln |C_k| + \vec{x}^T C^{-1} \vec{x} - 2\vec{x}^T C^{-1} \vec{m}_k + \vec{m}_k^T C^{-1} \vec{m}_k, k = 1, \ldots, K \quad (19)$$

Si además se asume que la matriz de covarianza es igual para todas las clases $C_k = C$ para $k = 1, \ldots, K$ entonces los dos primeros términos son iguales para todas las clases. Se tiene por tanto la función discriminante LDF[20]

$$h_k(\vec{x}) = \vec{m}_k^T C^{-1} \vec{m}_k - 2\vec{x}^T C^{-1} \vec{m}_k, k = 1, \ldots, K \quad (20)$$

La expresión 20 es una expresión sencilla. Se ha llegado a una colección de funciones discriminantes que son lineales en el término $\vec{x}$, Cuando se utiliza el clasificador normal[21] bayesiano de esta guisa se llama a menudo *análisis discriminante lineal* (LDA). Las superficies que dividen el espacio de entrada

[20] En inglés *linear discriminant function*

[21] *Normal* como sinónimo de gaussiano

ahora son planos (hiperplanos), y se puede demostrar que siguen siendo divisiones óptimas incluso para otras funciones de distribución no gaussianas. También puede interpretarse como una medida de la correlación entre el patrón $\vec{x}$ y el patrón representante de la clase $\vec{m}_k$, al que se le ha de sumar un término de energía.

El análisis discriminante lineal es muy sencillo de interpretar y de implementar y se entrena bien con un número reducido de patrones de entrenamiento, así que será la opción principal en el proyecto que se presenta.

Función discriminante de distancia euclídea. La función de clasificación de distancia es quizás el método más simple e intuitivo para solucionar el problema. El uso de funciones de distancia como herramientas de clasificación se entiende fácilmente a partir de la noción de que la similitud es una medida de la proximidad.

Los resultados que ofrece este clasificador son buenos sólo cuando las clases tienden a estar bien agrupadas, lo cual sucede de una manera muy aproximada en el problema que se trata. Este clasificador se llamará *clasificador de mínima distancia*. La distancia euclídea entre un vector $\vec{x}$ y uno $\vec{x}_k$ es

$$d_k(\vec{x})^2 = \|\vec{x} - \vec{x}_k\|^2 = (\vec{x} - \vec{x}_k)^T(\vec{x} - \vec{x}_k) \tag{21}$$

donde $\vec{x}_k$ es el patrón representante de la clase $\vec{y}_k$.

Un clasificador de mínima distancia calcula la distancia d_k de un patrón desconocido al prototipo de cada clase, y le asigna el patrón a la clase más próxima. La distancia no tiene por qué ser necesariamente euclídea, puede ser de otro orden distinto de 2. Para aplicar eficientemente la distancia euclídea, conviene que las componentes del patrón $\vec{x}$ estén normalizadas, que estén en el mismo rango. Para ello se puede pensar en escalar cada componente $x(n)$ de $\vec{x}$:

$$x(n) = \frac{x(n) - m_k(n)}{\sigma_k(n)} \tag{22}$$

El prototipo, el representante de cada clase, es único y en principio será el vector medio de los vectores de cada clase que sirvieron de entrenamiento, es decir, el estimador de la media de la clase. Si es así, y si la distancia es la euclídea, se puede demostrar que el clasificador de mínima distancia no es sino un caso particular del clasificador bayesiano. Retomemos la función discriminante QDF 17:

$$h_k(\vec{x}) = \ln|C_k| + (\vec{x} - \vec{m}_k)^T C_k^{-1}(\vec{x} - \vec{m}_k) k = 1, \ldots, K \tag{23}$$

Eso era la distribución de probabilidad normal que se desarrolló en el criterio bayesiano gaussiano, a la cual se le añade ahora otra hipótesis: que las características sean mutuamente incorreladas, es decir, que las componentes de $\vec{x}$ sean independientes entre sí. En ese caso la matriz de covarianza es una matriz diagonal. Y así:

$$d_k(\vec{x}) = \ln(\prod_{n=1}^{N} \sigma_k^2(n)) + \sum_{n=1}^{N} \left[\frac{(x(n) - m_k(n))^2}{\sigma_k^2(n)} \right] \tag{24}$$

donde $\sigma_k(n)$ y $m_k(n)$ es la desviación típica y la media de la componente n-ésima del patrón de la clase k.

Si además el primer término, el productorio, es el mismo para todas las clases, se puede eliminar del mismo modo que se eliminó al pasar de la función QDF a la de Mahalanobis. De modo que si la varianza es la misma en todas las componentes de cada clase, entonces no se tiene ni más ni menos que la distancia euclídea $d_k(\vec{x}) = \|\vec{x} - \vec{x}_k\|$

No se pierde eficacia respecto del clasificador bayesiano gaussiano con tal de que las varianzas de cada característica sean todas iguales. Si en el vector de características se incluyen términos con varianzas dispares, como es lo más común, el sistema funcionará muy mal a no ser que se normalicen todos los datos respecto a una varianza dada.

Este clasificador es lineal al igual que en análisis LDA del apartado anterior, y el espacio de la señal de entrada queda también fraccionado por hiperplanos. En la figura 17 se muestra una partición para un sencillo espacio de dos características (los hiperplanos son rectas). Cada punto del espacio se asocia a aquella clase cuyo representante quede más cercano en el sentido euclídeo. El diagrama así resultante, toma el nombre de *diagrama de Voronoi*.

Figura 17: Diagrama de Voronoi.

5.2.4. Función de coste.

La teoría bayesiana incluye además un factor de coste en las funciones discriminantes. Se suele denotar con c_{jk} y significa "el coste de decidir la clase j cuando en realidad se debió haber decidido k". Sirve para indicar que tomar una decisión errónea en unos casos es más grave que en otros.

En el caso que nos ocupa, seguramente nos duela igualmente el equivocarse al decidir "abrir mano" o "cerrar mano" cuando el usuario lo que deseaba era un "pronación". Sin embargo, sí puede ser más molesto que se decida un movimiento cuando el usuario lo que deseaba era el reposo; el reposo quizás debiera ser considerado como un caso especial que no debe ser abandonado a no ser que haya una convicción fuerte de lo contrario. Así, si el reposo es el estado $k = 0$ entonces la función $h_0(\vec{x})$ debería ser multiplicada por un coeficiente c_k, digamos 1,1 o 1,2, o, equivalentemente multiplicar al resto de funciones h_k por 0,8 o 0,9. La función del coste se puede aplicar a cualquier función discriminante de las ya vistas y es una contribución original del que escribe esta memoria.

5.2.5. Clasificador de los k vecinos más cercanos

A diferencia de todos los métodos anteriores el clasificador de los k vecinos más próximos (kNN[22]) es *no paramétrico*.

Es tan sólo una complicación del clasificador de distancia; la diferencia estriba en que en vez de tomar un patrón representante de cada clase se toman varios. Cada patrón de la clase y_i tenderá a agruparse alrededor de uno de los prototipos $z_1, \ldots, z_{N_i}$ donde N_i es el número de prototipos que representan a la clase i[23]

Después del entrenamiento se fijan los N_i patrones representativos de cada clase, en su versión más sencilla, cada patrón que sirvió de entrenamiento será un prototipo de su clase.

Se puede definir un clasificador de mínima distancia multi-prototipo, donde la función distancia entre un vector patrón $\vec{x}$ y la clase y_i es la distancia

[22]En inglés, *k Nearest Neighbours*

[23]Aunque hasta ahora se ha venido representando a una clase con la letra k, ahora se ha substituido por la letra i a fin de evitar confusiones con la k del clasificador de los k vecinos, que es el nombre con el que es conocido.

entre el patrón $\vec{x}$ y el prototipo más próximo de la clase i.

$$d_l = \text{mín}(x - z_l) l = 1, 2...N_i \tag{25}$$

Figura 18: Criterio del vecino más próximo

La clase que se elige es la que tiene la distancia mínima. En la figura 18 se muestra a los representantes de la clase A a la izquierda y a los representantes de la clase B a la derecha, y en el medio, la muestra que se desea clasificar. Su vecino más próximo es el cuadrado de la clase A, se muestra con una circunferencia que no hay otro vecino más próximo.

La regla de clasificación de los kNN, es el siguiente paso. Consiste en considerar los k vecinos más próximos, (por ejemplo 3 en el caso más sencillo), y contar cuál es la clase más representada en ellos, es decir, qué clase se repite más veces. En la figura 19 se muestra el ejemplo anterior pero con un criterio de 3-vecinos. En este caso, de los tres vecinos más próximos, dos pertenecen a la clase B y uno sólo a la clase A, por tanto se asigna a la clase B.

Figura 19: Criterio del vecino más próximo

La principal desventaja de este criterio es que la carga computacional es alta, hay que almacenar muchos prototipos y calcular muchas distancias, especialmente en el caso de que cada patrón (o vector de características) de entrenamiento sea considerado representante de su clase. Hay acotaciones precisas del error del clasificador kNN respecto del clasificador bayesiano [28]:

$$\varepsilon_B(x) < \varepsilon_{NN}(x) < 2\varepsilon_B(x) \tag{26}$$

donde ε_{eB} es la probabilidad de error del clasificador bayesiano y ε_{eNN} es la probabilidad de error del clasificador kNN. El clasificador de los k vecinos yerra poco, menos del doble del error mínimo.

5.2.6. Realización práctica del LDF

El análisis discriminante lineal es el elegido en el sistema que se propone por su sencillez. Sin embargo, para llegar a la función discriminante lineal se hizo alguna suposición que no es cierta para el caso de los patrones mioeléctricos y es necesaria una correción. La función discriminante lineal era:

$$h_k(\vec{x}) = \vec{m}_k^T C^{-1} \vec{m}_k - 2\vec{x}^T C^{-1} \vec{m}_k, k = 1, \ldots, K \tag{27}$$

De manera general se ha obtenido la regla de decisión 17, que se convertía en la regla de decisión lineal 20 con tan sólo solicitar que la matriz de covarianza fuera igual para todas las clases. En nuestro problema ya se ha comentado que las matrices de covarianza no son iguales para los distintos estados, y sin embargo sería deseable poder aplicar un clasificador lineal.

En la relización práctica se reescribe la función discriminante lineal como:

$$h_k(\vec{x}) = V^T \vec{x} + v_0 \tag{28}$$

Por simple identificación entre las ecuaciones 28 y 20:

$$V = -2C^{-1} \vec{m}_k \tag{29}$$

y

$$v_0 = \vec{m}_k^T C^{-1} \vec{m}_k \tag{30}$$

Esto ahora no vale porque las matrices de covarianza no son iguales, y el problema es que se ignora qué V y que v_0 tomar. El patrón N-dimensional $\vec{x}$ es proyectado en la dirección de un vector V para dar un escalar h y v_0 será el umbral para decidir una u otra clase. Se demuestra que si X tiene una función de distribución normal, entonces h también. La esperanza de $h_k(\vec{x})$ es:

$$\eta_{h_h} = E[h_k(\vec{x})] = V^T m_k + v_0 \tag{31}$$

y

$$\sigma_{h_k}^2 = Var[h_k(\vec{x})] = V^T C_k V \tag{32}$$

Cálculo de V y v_0 para el clasificador lineal entre dos clases. Se tratará de maximizar la separabilidad entre sólo dos clases $k = 1$ y $k = 2$, la extensión a más un clasificador de más de dos clases no será sencilla pero tomará como referencia este caso simple que no se puede por tanto eludir. Sea $f(\eta_0, \sigma_0, \eta_1, \sigma_1)$ la función que se desea maximizar. Derivando f respecto de V y v_0 e igualando a 0 se habrá obtenido el máximo, que se demuestra [30] que se da cuando:

$$V = [sC_1 + (1-s)C_2]^{-1} (\vec{m}_2 - \vec{m}_1), \tag{33}$$

donde

$$s = \frac{\partial f/\partial \sigma_1^2}{\partial f/\partial \sigma_1^2 + \partial f/\partial \sigma_2^2} \tag{34}$$

Una vez conocido f, v_0 se puede calcular a partir de la expresión:

$$\frac{\partial f}{\partial \eta_1} + \frac{\partial f}{\partial \eta_2} = 0 \tag{35}$$

Ahora bien, ¿qué f se debe tomar como medida de separabilidad? Veamos dos aproximaciones:

- *Criterio de Fisher.* El criterio de Fisher postula como función a maximizar la siguiente:

  $$f = \frac{(\eta_1 - \eta_2)^2}{\sigma_1^2 + \sigma_2^2} \tag{36}$$

  Operando como se ha descrito, se tiene:

  $$V = \left[\frac{C_1 + C_2}{2}\right]^{-1} (\vec{m}_2 - \vec{m}_1) \tag{37}$$

  Se observa que en el criterio no interviene v_0, dado que al restar $\eta_1 - \eta_2$ se elimina en la ecuación 31. Con tan sólo intruducir a V en la ecuación 28, se obtiene el *clasificador lineal de Fisher*.

- *Dispersión entre clases.* Se toma a f como la dispersión entre clases normalizada por la dispersión dentro de la clase. Es decir:

  $$f = \frac{P(y_1)\eta_1^2 + P(y_2)\eta_2^2}{P(y_1)\sigma_1^2 + P(y_1)\sigma_1^2} \tag{38}$$

  Aplicando las ecuaciones anteriormente descritas, se llega a que:

  $$V = [P(y_1)C_1 + P(y_2)C_2]^{-1} (\vec{m}_2 - \vec{m}_1) \tag{39}$$

  y

  $$v_0 = -V^T[P(y_1)\vec{m}_1 + P(y_2)\vec{m}_2] \tag{40}$$

Cálculo de V y v_0 para el clasificador lineal multiclase. En este caso, que es el del presente trabajo, es demasiado complicado llegar a una solución analítica, y se considerarán dos alternativas:

- *Promedio.* Se puede tomar la ecuación 20 y considerar la matriz de covarianza que debiera ser igual para todas las clases como la media de las matrices de covarianza. El método tan sólo es razonablemente válido si las matrices de covarianza no son muy distintas. Parece una solución sencilla apropiada al problema al que se enfrenta este proyecto. Es tan sólo calcular $h_k(\vec{x})$ para todas las clases y escoger la mayor:

$$h_k(\vec{x}) = 2\vec{m}_k \left(\frac{\sum_{i=1}^{K} C_i}{K} \right)^{-1} \vec{x} - \vec{m}_k^T \left(\frac{\sum_{i=1}^{K} C_i}{K} \right)^{-1} \vec{m}_k \qquad (41)$$

- *Ajuste de coeficientes.* Este método es realmente muy complicado de implementar y no se sugiere su uso, salvo cuando las matrices de covarianza sean muy distintas. En el caso de los patrones mioeléctricos no debería ser necesario implementarlo.

  El método consiste en hallar las funciones discriminantes lineales para cada par de clases. Es decir, calcular h_{ij} para todo i y para todo j, hallar cada vector V_{ij}^T mediante los métodos expuestos anteriormente para el caso particular de separación entre dos clases.

  Entonces, el espacio queda dividido por múltiples rectas divisoras, y un patrón dado s pertenece a la clase k si en todas y cada una de las combinaciones en las que está implicado sale vencedor. Una de las desventajas del método es que no todos los puntos del espacio de entrada caen en un sitio al que se pueda asignar con seguridad una clase (este espacio se llama *región de rechazo*). Un refinamiento sería retirar la condición de que el punto se asigna a una clase si es la mejor en *todas* las combinaciones, e imponer una condición menos restrictiva.

Procedimiento propuesto. Estos son los pasos que hay que dar:

- Tomar patrones de entrenamiento. Supuesto un espacio de unas 10 características, se deben tomar unos 100 patrones de entrenamiento. En nuestro caso se ha de pedir al usuario que realice una contracción determinada durante medio minuto. En el caso de aplicar la información temporal de Hudgins [42], hay que tener en cuenta que ésta sólo es válida en su fase transitoria, por lo que *no* es posible tomar el patrón

de entrenamiento a partir de medio minuto de esfuerzo, sino que han de tomarse unas 50 veces su ejercicio partiendo del reposo; y por supuesto, luego operar con los subpatrones de 40ms de manera individual.

- Calcular el vector de medias de cada clase. Se halla el vector estimador $\hat{\vec{m}}_k$ como simple media de los vectores de entrenamiento según la ec. 13 y se halla la matriz de covarianza estimada según la ec. 14 para cada clase.
- Se calcula la media de todas las matrices de covarianza C_k, bajo la hipótesis de que la diferencia no es muy grande. La función se escribe entonces como la ec. 41.
- En tiempo real se calculan las funciones h_k para cada patrón.
- Se pondera positivamente la clase del reposo, haciendo más probable su detección.
- Se suaviza la función discriminante; habiendo almacenado las funciones discriminantes de los dos o cuatro patrones anteriores, se puede aplicar un filtro de medianas, es como propone en [61].
- Se escoge la mejor clase h_k.

Cabe comentar que el uso del clasificador de los k-vecinos es también altamente recomendado.

Limitaciones del análisis discriminante lineal. Para que la regla de decisión del discriminante lineal funcionara bien, sin embargo, las hipótesis que se han aplicado (funciones de distribución gaussianas, clases equiprobables, matrices de covarianza iguales para cada clase etc.) deberían ser ciertas. Las limitaciones vienen si hay uno de estos casos:

- Desigualdad de las matrices de covarianza. La hipótesis de que las matrices de covarianza son iguales en todas las clases es bastante mala en el caso que nos ocupa. Un análisis a simple vista de los patrones mioeléctricos revela que la dispersión de los datos no es ni mucho menos la misma para cada clase. En general, niveles bajos de contracción muscular implican más agrupamiento, y esfuerzos mayores implican una dispersión mucho mayor. Esta hipótesis, que tan sólo se cumple de una manera burda, ha sido utilizada para llegar a la expresión más sencilla posible; en breve se tratará de cómo reponerse de este error.

- Alta correlación entre las componentes de $\vec{x}$. O bien una diferencia de escala muy grande entre las características de X. La solución pasa por emplear el discriminante de Mahalanobis o bien transformar los patrones. Con ese problema se ha de contar en las características que se extraigan de los patrones mioeléctricos. Gráficamente se puede comparar la situación buena con la situación en la que fracasa el clasificador:

Figura 20: Características fuertemente correladas.

- La función de distribución de X no es gaussiana. Si toma una disposición curva, el discriminante lineal será malo. Puede paliarse usando el discriminante de Mahalanobis que emplea superficies cuadráticas para dividir el espacio, pero si ello también fracasara habría que recurrir a los algoritmos de aprendizaje, a las redes neuronales.

Figura 21: Características poco aptas para un clasificador lineal.

- La distribución se agrupa en varios subgrupos. En ese caso también habría que recurrir al uso de redes neuronales, aunque es un caso que no debería presentarse con las características extraidas de los patrones mioeléctricos. El clasificador de los k vecinos también funcionaría bien.

Figura 22: Características con varios máximos.

- Distribución demasiado compleja. Si responde a una estructura compleja, como la de la figura, es mucho mejor recurrir a un método no estadístico sino sintáctico.

Figura 23: Características apropiada para un clasificador sintáctico.

5.2.7. Análisis discriminante de patrones mioeléctricos.

El LDA ha sido aplicado numerosas veces para clasificar los patrones mioeléctricos (por ejemplo [6]). El trabajo clásico de Saridis [15] proponía una pequeña mejora que consistía en hacer que el sistema lineal aprendiera en un periodo de entrenamiento, con el propósito de mejorar notablemente el rendimiento. Su análisis partía también de la ec. 28. Definía un vector W de modo que englobara a V y a v_0, lo cual es sencillo con sólo hacer $x_0 = 1$, $w_0 = v_0$ y $w_i = v_i$:

$$h_k(x) = W^T x \tag{42}$$

y en sucesivos patrones clasificados, se ajustaba el vector W en base al valor de W en el patrón anterior y al éxito pretérito. El método de corrección propuesto lo sugiere también Fukunaga[30], llamándolo *regla de correción del gradiente.*

Kang [54] comparó el rendimiento de tres clasificadores de patrones mioeléctricos para controlar prótesis; clasificadores que ya se han descrito aquí: el de máxima similitud (ec. 17), el de Mahalanobis (ec. 18) y el euclídeo (ec. 21). Los comparó tomando dos grupos de características distintos (eso no nos interesa de momento, pero eran los coeficientes de un modelo AR y los coeficientes cepstrales), obteniendo los resultados esperados de que cuanto más general es la regla mejor el rendimiento (a costa de mayor complejidad de los algoritmos). El método de los k vecinos ha sido usado también profusamente, y así por ejemplo se encuentra también en Zardoshti [58].

5.3. Clasificadores basados en aprendizaje.

5.3.1. Redes neuronales artificiales.

Las redes neuronales no son más que una forma de computación distribuida investigada hace 50 años. Su principal rasgo es que es un sistema capaz de aprender a partir de una colección de datos de entrenamiento. Para una

descripción más detallada se puede consultar [41].

En el problema de clasificación de patrones de señal EMG, lo que se ha utilizado es un conjunto de nodos o neuronas organizado en tres capas. La denominación de estas redes es 'red de perceptrones multicapa' (MLP).

Figura 24: Red neuronal de tres capas.

Cada nodo es un perceptrón, y es la unidad de cálculo elemental; más adelante se describe con más precisión. La primera capa es de entrada, la tercera de salida y la del medio es denominada oculta. También se ha ensayado más de una capa oculta para este problema, pero parece ser suficiente con una sola.

En la capa de salida debe haber K perceptrones, uno por cada salida posible. En la capa de entrada debe haber un número de perceptrones igual al conjunto de características que han de ser procesadas, y puesto que no se introducen las N componentes del patrón, se introducirán las M componentes del vector de características del patrón.

La fuerza de este algoritmo es que se adapta a cualquier colección de datos, aunque estos no respondan a un modelo predeterminado, y así se salva la dificultad de tener que aceptar la hipótesis sobre la distribución de probabilidad, como se hacía en la sección del clasificador gaussiano. No es necesaria esta información *a priori*.

El perceptrón es una unidad de cálculo elemental, que recibe N entradas y proporciona una sola salida. Lo definió Rosenblatt en 1958. Al aplicar un vector de entradas $[x(1), \ldots, x(N)]$ se le multiplica por uno de pesos $[w(1), \ldots, w(N)]$ y se le suma un sesgo $w(0)$ o θ.

$$neta_i = \sum_j x(j) w_i(j) + \theta \tag{43}$$

El resultado de todo ello se somete a una función de salida. La función de salida puede ser una función umbral:

$$f_m(s) = 1 \text{ si } s > 0 \text{ o } 0 \text{ si } s < 0 \tag{44}$$

o más comúnmente usada, la función llamada *sigmoide*:

$$f(s) = \frac{1}{1 + e^{(-\beta s)}}. \tag{45}$$

Esta función es continua y varía de 0 a 1 de manera monótona según s varía de $-\infty$ a $+\infty$. La ganancia β de la sigmoide, determina la curvatura. La ventaja de esta función es que es diferenciable.

El aprendizaje de la red será una actualización de los pesos $\vec{w}$ hasta que se ofrezca la salida deseada. Al introducir un conjunto de pares de entrenamiento, se compara la salida obtenida con la salida deseada, y la diferencia o *señal de error* sirve para recalibrar los pesos. Lo que se busca en cada instante es que la combinación de pesos $\vec{w}$ sea exactamente la que minimice el error ya que en general no existe un vector $\vec{w}$ que se ajuste para dar error cero con todos los patrones de entrenamiento.

El algoritmo de aprendizaje que más se ha utilizado en este problema es el de retropropagación del error [24]; en el que la señal de error va pasando desde la última capa a la primera y así va actualizando los pesos. Esto es un algoritmo de aprendizaje supervisado. Se dice *supervisado* si durante el entrenamiento ya está disponible la salida deseada, se dice *no supervisado* si los grupos se deciden finalmente tras el aprendizaje.

Los pesos reciben inicialmente unos pesos pequeños y aleatorios al igual que el término de tendencia o sesgo, digamos entre -0.5 y 0.5. El término de tendencia en realidad puede ser tratado como un peso más, con tal de considerar que está conectado a una unidad ficticia cuyo valor de salida es siempre 1. Estos pesos escalares pueden ser en ocasiones filtrados por un filtro FIR, en este caso la red neuronal es denominada FIRNN. En el caso de reconocimiento de patrones mioeléctricos para control de prótesis, Englehart la propuso ya desde muy temprano [59].

La velocidad de aprendizaje es importante en el rendimiento de la red que viene determinado por una constante η. El vector de pesos en el período de entrenamiento se va actualizando a una veocidad determinada por esa constante. El valor η suele tomar valores pequeños, por ejemplo 0.01, con lo cual necesita más entrenamiento pero se asegura llegar a la solución. Otras veces η varía dinámicamente, y según va siendo el error más pequeño va incrementando su valor. Otra forma de aumentar la velocidad de convergencia consiste en utilizar una técnica llamada *momento*. Consiste en que cuando se calcula el valor del cambio de peso, se añade una fracción del cambio anterior.

[24] En inglés, *backpropagation*

El proceso iterativo de calcular el gradiente y ajustar pesos se continúa hasta que se alcanza un mínimo en la superficie; en la práctica se finaliza cuando cuando la norma del gradiente es menor que un umbral dado. Hudgins [42] utilizó para su red una veintena de pares de entrenamiento para cada clase, pero en general se requieren muchos cientos de patrones o incluso muchos miles. El tiempo de entrenamiento suele ser importante, de al menos un par de horas.

El número de nodos a establecer en la capa oculta es difícil de establecer, y depende mucho de la naturaleza de los datos. Es difícil de saberlo a priori, así que se suele proceder por tanteo. Por ejemplo, Chaiyaratana con un número grande de características de la señal mioeléctrica hubo de enseñar con un número de nodos de 8 a 80 [60].

Lo que subyace para este problema, es que la probabilidad a posteriori $P(y_k|\vec{x})$ se estima directamente. Al usar una red neuronal se está entrenando a un estimador para que aproxime esas funciones de probabilidad $P(y_k|\vec{x})$ para cada clase $k = 1, \ldots, K$. El vector de pesos $\vec{w}$ representa los parámetros del estimador que se entrenan.

Cuando se emplea el algoritmo de retropropagación, la red neuronal puede aprender la mejor aproximación de las probabilidades a posteriori en el sentido de mínimos cuadrados. Se demuestra que para cualquier patrón dado, las salidas de la MLP que tienen como función una sigmoide, tienden a sumar 1, es decir, que funciona como una función de densidad de probabilidad. Así que además, al tomar una decisión, también se conoce el grado de fiabilidad con el que se da la respuesta El entrenamiento para ello debe de ser suficientemente amplio como para generalizar, y el número de nodos de la capa oculta suficientemente generoso. Sin embargo demasiados patrones de entrenamiento pueden hacer olvidar los primeros, y demasiados nodos pueden suponer demasiado tiempo de computación.

El primer sistema importante de redes neuronales aplicado al reconocimiento de patrones mioeléctricos para dar señales de control, fue también Hudgins [42], y tras él, su uso se ha generalizado. Como algunos ejemplos baste citar a Eriksson [89], Cunha [132] o Nishikawa [107].

Eriksson ofrece una mano virtual con un número de señales de entrada alto, y con muchos grados de libertad, $L = 8$ y $K = 9$. Nishikawa propuso controlar 5 DOF (L=11) de otra mano con tan sólo 2 canales. Usó 16 nodos en la primera capa y 18 en la segunda, tomando como características los

coeficientes más significativos de la transformada Gabor más el MAV[25].

Cunha usó una red neuronal algo especial: cada muestra de la señal atacaba a cada uno de los 200 primeros nodos de la red, es decir, sin extracción de características previa. En la segunda capa determina como óptimos tan sólo tres nodos. Con un canal, ofrecía sólo 1 DOF, centrándose en que la naturalidad fuera perfecta y el número de errores cero.

Todos ellos usaron el algoritmo de *backpropagation* para el aprendizaje. En diagnosis ha sido utilizado otro tipo de redes neuronales para procesar la señal mioeléctrica, las redes DAF (Distributed Approximating Functionals, DAFNN)[105].

5.3.2. Lógica difusa.

Los sistemas de lógica difusa son también aplicables para este problema. Se ha presentado como mejora de las redes neuronales para este problema, y se han presentado como una manera de procesar los datos más similar a lo que hace el ser humano. La señal EMG es a veces contradictoria; y pares de entrenamiento iguales podrían corresponder a clases distintas, como ya se ha comentado. Una de las propiedades más útiles de la lógica difusa es la capacidad para tolerar estas contradicciones.

Los sistemas difusos primero hacen más difusas las entradas en grados de pertenencia a conjuntos difusos. Después el motor de inferencia, a partir de reglas extraídas de la experiencia, infiere unas salidas que son de nuevo concretadas.

Park propone como clasificador un método de acumulación de evidencias muy bien descrito en [85]. Usó un electrodo y ofreció 3 DOF. Han [133] también ha propuesto la lógica difusa, una red neuronal difusa min-max, distinguiendo entre 4 DOF con 4 canales.

5.4. Validación del clasificador

Una vez diseñado el clasificador, se debe comprobar cual es su grado de eficacia. Probar la eficacia es ver qué tal clasifica una serie de patrones cuya clase sí que se conoce. Para ello se proponen los siguientes métodos:

[25]Más adelante se explican estas características de la señal.

- Prueba sobre los datos de entrenamiento. Si el clasificador no es capaz de clasificar bien ni siquiera los datos que sirvieron para entrenarle, evidentemente se debe empezar de nuevo.

- Método *holdout*, es práctico y funciona bien. De los patrones recogidos para el entrenamiento, sólo algunos se emplean para entrenar al clasificador, y se reservan los otros para comprobarlo. El inconveniente es que hay que tomar más patrones de entrenamiento de los normales, o bien desaprovechar los que ya se tienen.

- Método *leave-one-out*, es muy utilizado. El clasificador se entrena con todos los patrones disponibles excepto con uno, que sirve de comprobación. A continuación se vuelve a incluir al patrón separado, y se excluye a otro distinto para hacer otra comprobación; y así se procede sucesivamente con todos los patrones de entrenamiento. Este método además es una buena cota superior del error bayesiano. En el caso de clasificadores de patrones mioeléctricos lo utilizó por ejemplo Kang [54].

- Método *bootstrapping* Toma aleatoriamente alguno de los patrones de entrenamiento.

6. Extractor de características

Extraer características significa reducir un patrón de dimensión N a otro de dimensión M sin perder información significativa. El problema es que el patrón está enmarañado por el ruido y al elegir un conjunto de características de tamaño $M < N$ se perderá irremediablemente parte de la información útil para clasificar los patrones.

Escoger el conjunto óptimo de características es vital, y es un problema que todavía no está cerrado. Y aun dejando de lado los clasificadores estructurales, las características F que se deben utilizar dependen del clasificador que siga en la cadena; las que son óptimas para un clasificador, pueden no serlo para otro. El objetivo es extraer las características que mejor preserven la separabilidad para un clasificador dado.

A fin de simplificar, se asumirá que el clasificador de referencia es el bayesiano; de modo que maximizar la separabilidad es minimizar el error bayesiano. El vector de características $\vec{f}$ [26] idóneo en el caso bayesiano son como es bien sabido las $K-1$ probabilidades a posteriori $P(y_k|\vec{x})$[27]. Así que la mejor medida de la separabilidad de dos clases es el error bayesiano que arroja su clasificación; pero como no se conocen las probabilidades a posteriori esto sólo será una medida teórica, y el error se calculará experimentalmente o se darán medidas de la separabilidad de las clases.

En el caso de que las características se extrajeran como combinación lineal de las muestras del patrón, y dado un criterio de separabilidad de clases, sería muy sencillo extraer de manera automática la combinación óptima de muestras que mejor separabilidad presentara. Sin embargo los parámetros significativos que representan la señal no están linealmente relacionados con sus muestras. Y no existe un método para extraer características no lineales de un patrón de manera sistemática, de modo que la extracción de características es algo que depende de la naturaleza de cada problema, y es preciso tantear con las distintas posibilidades.

Evidentemente, si se conociera de manera determinística las ecuaciones que rigen la señal EMG, bastaría con conocer los parámetros, pero ello, claro está, no nos está dado a conocer. En esta sección se repasan cuantas características han sido empleadas para clasificar los patrones mioeléctricos. En el primer apartado se postula un método que explica la relación entre señal

[26] O $\vec{x}$ en un sentido generoso.

[27] Son $K-1$ componentes y no K porque la última se deduce al instante sabiendo que todas han de sumar la unidad.

EMG y fuerza, y cómo éste fue el primer parámetro escogido para gobernar prótesis mioeléctricas.

En el segundo apartado y en el tercero se abandona este enfoque y se evita dar una explicación al origen de la señal, tomando al músculo como a una caja negra. El segundo extrae características a partir de la representación temporal de la señal, el tercero a partir de la representación frecuencial. Finalmente un cuarto apartado explora los métodos en tiempo - frecuencia. Pero antes que todo ello se explican los conceptos de *selección* y de *proyección* de características.

6.0.1. Selección y proyección

El conjunto de características que se extrae inicialmente se puede dejar tal cual está o bien puede ser manipulado para mejorar la separabilidad de las clases. Por ejemplo, si se toma como características a la varianza y al número de cruces con cero de la señal, pueden dejarse en el vector $\vec{f}$ (o mejor dicho, $\vec{x}$) estas dos características tal cual están (entonces el procedimiento se llama *selección de características*) o bien se pueden combinar de manera que que se mejore la distinción de clases (se habla entonces de *proyección de características*).

La *proyección* de características se aplica sobre un conjunto de características ya elegido, de modo que la *selección* ha de realizarse en todo caso, no se puede eludir; y la *proyección* no es más que una mejora. La proyección puede ser, por ejemplo, ponderar de diversas maneras las componentes del vector $\vec{x}$, comprobar el error del clasificador para cada caso, y escoger la combinación óptima.

Así que el procedimiento es escoger un conjunto de características, tan amplio como sea posible, quedarse con un subconjunto (selección) y después combinar los más relevantes (proyección). En todo caso, la proyección es una mejora importante sobre la selección si no se tienen en cuenta las covarianzas, ya que puede hacer incorreladas a las características, además tiene la ventaja de que es un método no supervisado. Englehart comentó estas ventajas en [116]. Ahora se estudiarán las características candidatas a ser seleccionadas.

6.1. Estimación de la fuerza

Las primeras características que se presentaron como base para el clasificador fueron simples estimaciones de la fuerza. Al fin y al cabo, todos los movimientos se producen con la fuerza de los músculos y no con otro parámetro. Ya se ha comentado que las leyes de Newton deberían determinar una prótesis que fuera un sustituto realmente natural del miembro original: las prótesis de control proporcional.

Sin embargo la estimación de la fuerza como señal de control para prótesis, aun siendo buena, adolece de graves carencias. La más grave de todas es que no permite más de un grado de libertad por cada músculo del que se recoja señal. Las prótesis de control proporcional han sido prácticamente deshechadas por esto mismo. Sin embargo, la señal de control que regía las prótesis es una buena entrada para el clasificador, y así ha sido propuesto a menudo [58]. En este proyecto, la fuerza se considera como una característica más a ser considerada, si bien especialmente significativa.

La hipótesis habitual es que la potencia de la señal está relacionada con la fuerza realizada; a la hora de estimar se utilizaba una sencilla rectificación y un suavizado. La señal EMG demodulada como si fuera de AM, se tomaba como la estimación de la fuerza, y así se hizo por primera vez ya en 1952. En todo caso, para los primeros sistemas, era algo muy sencillo de realizar en hardware; y aunque desearan otros cálculos tampoco podían hacerlos. Así la estimación de la fuerza muscular puede servir para dirigir las señales de control de un actuador. Como la relación entre fuerza y amplitud es complicada pero cierta, los primeros sistemas usaron como señal de control la amplitud de la señal. En los años 80 se fijó la relación entre fuerza y amplitud, y se dedujo consiguientemente el procesado óptimo de la señal que maximizara la SNR según se definirá. Hoy la importancia de estos análisis ha decaído, pero aun se sigue investigando y presenta cierto interés.

6.1.1. La fuerza se manifiesta en la varianza

El modelo clásico supone a la señal $x(t)$[28] como un proceso aleatorio gaussiano de media cero y varianza $\sigma(F(t))$, donde $F(t)$ es una medida de la

[28]Obsérvese que ahora la señal es $x(t)$ y no $x(n)$, el cambio de notación que se sigue aquí deliberadamente denota con variables distintas al tiempo continuo t y a las muestras discretas n. Cuando se denota un patrón con un número fijo de muestras se sigue escribiendo $\vec{x}$

fuerza del sujeto.

Figura 25: Modelo del origen de la señal EMG

Ello es aceptable bajo la consideración de que la señal es gaussiana por la suma de otros muchos procesos aleatorios con la misma media y varianza (y no necesariamente la misma distribución, según asegura la ley de los grandes números). El sistema H representa el filtrado que modela al ruido blanco gaussiano de media cero. Ese filtro es dependiente de los tejidos, de la posición electrodos (ver ec. 2) etc. El sistema $\sigma(F)$ oculta la relación entre la fuerza y la amplitud de la señal EMG. Es decir, se representa el origen de la señal como un ruido blanco gaussiano pasado por un filtro desconocido, y modulado en amplitud por una desconocida función $\sigma(F)$ a la guisa de la figura 25.

Esta afirmación es en principio gratuita. Nada nos hace pensar que la fuerza haya de manifestarse en el parámetro *varianza* de la señal y no en cualquier otro, como por ejemplo la frecuencia de la señal. Así que ello hay que demostrarlo; y ese fue el trabajo clásico[29] de Hogan [13][14].

La actividad mioeléctrica es un proceso no estacionario, ya que la fuerza del músculo varía, pero considerando períodos suficientemente cortos no va a ser relevante la no estacionareidad. La función de densidad de probabilidad de una muestra es normal:

$$p(x(n)) = \frac{1}{\sqrt{2\pi}\sigma(F)} exp\left[-\frac{x^2(n)}{2\sigma^2(F)}\right] \tag{46}$$

La dependencia temporal entre muestras sucesivas queda determinada por la función de autocorrelación, o, equivalentemente, por la densidad espectral de potencia. Según se demuestra [13], la densidad espectral de potencia de la señal es:

$$S_x(f) = |H(f)|^2 \sigma(F)^2 \tag{47}$$

[29] Aunque algo confuso y con varios errores.

El espectro de $F(t)$ se limita a unos pocos hertzios, pues no es posible excitar los músculos mucho más deprisa, así que por fortuna la actividad mioeléctrica y la voluntad del sujeto centran su energía en bandas distantes en un orden de magnitud, y no se solapan demasiado. Considerando la frecuencia máxima de la voluntad en 10 Hz basta con tomar periodos de 100ms, es decir, que los patrones de tamaño 80 o 100 muestras en nuestro sistema pueden ser considerados estacionarios.

A partir de la función de autocorrelación (o si se prefiere, la densidad espectral de potencia) y la función de densidad de probabilidad de cada muestra dada en 46 se debe poder extraer la función de similitud para un patrón, $p(\vec{x}|F) = p(x(1)\dots x(N)|F)$. Esa es la probabilidad de que, dada una fuerza F, la señal sea $\vec{x}$. Se está en el problema inverso: dada una señal $\vec{x}$, que es la que se ha medido, cúal es la fuerza estimada $\hat{F}$ para la cual la probabilidad es mayor. Se trata pues, de maximizar $p(\vec{x}|\hat{F})$. Esta vez no se acude al teorema de Bayes, y para maximizar se deriva y se iguala a cero.

$$\frac{d}{dF}p(\vec{x}|F)\bigg|_{F=\hat{F};\vec{x}=\vec{x_{ob}}} = 0 \tag{48}$$

Tomando logaritmo nada se pierde:

$$\frac{d}{dF}[\ln p(\vec{x}|F)] = 0 \tag{49}$$

Aceptada la suposición de que todas las muestras son incorreladas, las probabilidades de cada muestra son independientes. La suposición es rotundamente falsa. El espectro de la señal EMG no es plano, y por tanto su función de autocorrelación no es una función delta. Así que para aceptar la hipótesis, se ha de aplicar un filtro de blanqueo. Un filtro de blanqueo es aquel cuya función de transferencia es inversa al espectro de la señal que ha de pasar por ella.

Así, tras el filtro de blanqueado, la señal presenta un espectro plano, y una correlación nula entre dos muestras distintas cualesquiera. Entonces, una vez blanqueada la señal, la probabilidad conjunta queda como un producto de probabilidades:

$$p(\vec{x}|F) = \prod_{n=1}^{N} p(x(n)|F) \tag{50}$$

Juntando 50 con 46, se tiene:

$$\ln p(\vec{x}|F) = \sum_{n=0}^{N} \ln p(x(n)|F) \tag{51}$$

es decir

$$\ln p(\vec{x}|F) = -N \ln \sqrt{2\pi} - N \ln \sigma(F) - \frac{1}{2}\sigma^{-2}(F) \sum_{n=1}^{N} x^2(n) \tag{52}$$

Como la derivada era cero...

$$\frac{d}{dF}[\ln p(\vec{x}|F)] = -N\sigma'(F)\sigma^{-1}(F) + \sigma'(F)\sigma^{-3}(F) \sum_{n=1}^{N} x^2(n) = 0 \tag{53}$$

Donde las primas representan derivación respecto a F. Sacando factor común:

$$\sigma'(F)\sigma^{-2}(F) \left(\sigma^{-1}(F) \sum_{n=1}^{N} x^2(n) - N\sigma(F) \right) = 0 \tag{54}$$

Por tanto:

$$\sigma^2(F) = \frac{\sum_{n=1}^{N} x^2(n)}{N} \tag{55}$$

Así que ya tenemos una relación teórica. Si se denota ahora ya la fuerza como estimación que es, y se representa la inversa de la desconocida función $g = \sigma(F)$ como $F = \sigma^{-1}(g)$...

$$\hat{F} = \sigma^{-1} \left[\sqrt{\frac{1}{N} \sum_{n=1}^{N} x^2(n)} \right] \tag{56}$$

Este es un gran resultado. Lo que está entre corchetes es ni más ni menos que la varianza de las muestras observadas. Es decir, que la fuerza se manifiesta *únicamente* a través de la varianza de la señal.

Intuitivamente este resultado era previsible. La señal iba a quedar determinada por dos agentes: el espectro de potencia y la función de densidad de probabilidad de amplitud de las muestras. Hogan *no* hace la reflexión, pero es fácil intuir que especificados ambos parámetros queda totalmente descrita la señal. La fuerza no se podía manifestar en parámetros frecuenciales porque la salida del filtro blanqueador es 'blanca', y como la función de distribución de probabilidad era gaussiana por su hipótesis de que hay muchos procesos aleatorios, entonces sólo esta última distribución iba a caracterizar la señal. Y la distribución normal queda completamente determinada por su media y su varianza. Siendo la media cero para toda señal EMG, el único parámetro en que podía manifestarse la fuerza era en la varianza.

Y la relación entre varianza y fuerza nos es desconocida, sí, pero una vez realizada esta demostración será tarea sencilla tomar datos experimentales

y ajustar la relación. Este resultado teórico es muy importante: *una vez blanqueada la señal, la fuerza F puede estimarse a partir de la desviación típica de la señal*. Se deben tomar las muestras, calcular su desviación típica y de ahí inferir la relación (por ejemplo, se ha tomado a menudo la relación $\sigma(F) = F^{1,75}$). Más adelante se profundiza en esta desconocida relación, pero ahora ya se llegado a un hito y es necesario recapitular el trabajo desarrollado.

Se ha deducido que el esquema de procesado óptimo de la señal debe incluir un blanqueador, una no linearidad de la que se saca la variable significativa (por ejemplo una relación cuadrática que calcule la varianza), y luego la no linearidad inversa para traer los niveles de amplitud a los originales (quedarlo en desviación típica). A este esquema se le puede añadir una etapa de filtro previo para eliminar el ruido en las bandas donde la señal no sea significativa, y se le puede añadir una etapa de suavizado para que las decisiones de la prótesis no sean bruscas. El esquema resultante fue usado más o menos así de manera habitual en los años 80, y se presenta en la figura 26.

Figura 26: Procesado para la estimación de la amplitud.

6.1.2. Filtrado para eliminación de ruido

La señal EMG centra la mayor parte de su energía entre los 40 Hz y los 1000 Hz. Un filtro paso banda que elimine la señal para el resto de las frecuencias reducirá la cantidad de ruido notablemente. Habitualmente se suele dejar pasar la señal entre los 10 Hz y los 1200 Hz, aunque se han llegado a recomendar en ciertas frecuencias filtrados paso alto a frecuencias de hasta 100Hz [134]. Frecuencias de corte tan altas son tolerables si se desea estimar la amplitud, pero no lo son si se desea obtener información espectral.

El filtrado paso alto es imprescindible para eliminar el ruido del movimiento de los cables o de los electrodos. Es una componente muy fuerte presente hasta los 5 o 10 Hz. También filtrando paso alto se elimina el ruido del electrocardiograma (EKG), con componentes importantes hasta los 30Hz. Y de paso, se suprimen las posibles insuficiencias del amplificador, así por ejemplo, se elimina la señal de *offset* que aunque debiera ser nula, siempre estaba presente.

El filtrado en este proyecto es software y hardware, si bien se deja como futura tarea la elaboración de unos filtros software más elaborados que los presentes y que optimicen la limpieza de la señal de una manera más eficaz. La cabecera P4 realiza un filtrado paso alto hacia los 3 Hz y uno paso bajo en los 3 kHz.

Para estimar la amplitud se han propuesto también un filtrado previo con un filtro cuyo tamaño de ventana se adapta según convenga. En todo caso, para una ventana rectangular, se demuestra que la señal mejora su relación SNR con la longitud de la ventana de manera cuadrática, según la fórmula deducida por [115], recordemos,

$$SNR = \sqrt{4B_eLT} \tag{57}$$

con B_e ancho de banda de la señal y L el numero de canales, siendo T el tamaño de la ventana en segundos. Cuanto mayor SNR, más fiable es la estimación de amplitud y menor el error del estimador debido a la varianza. Pero tampoco se puede tomar una ventana demasiado larga, o se corrererá el riesgo de que la señal sea no estacionaria (por ejemplo ha empezado a mover el brazo) y crezca demasiado el error de sesgo del estimador. Hay que llegar a un compromiso.

Por otra parte, para estimar la amplitud de la señal EMG en un intervalo de tiempo, se pueden tomar todas las muestras de un patrón equiponderadas, lo cual equivale a un filtro rectangular, o se puede aplicar otro filtro que atenúe las muestras del inicio y del final, previamente habiendo tomado intervalos solapados. Ello es más bien un filtro de sueavizado. Se podría considerar fijo el tamaño de esta ventana pero también se podría modificar dinámicamente en función de la señal, que parece bastante más razonable. Para hacerlo se deberían tener en cuenta no sólo la propia amplitud de la señal sino también sus dos primeras derivadas [104], aunque la mejora sobre una ventana de tamaño fijo no es tan significativa como para ser considerada en nuestro sistema.

6.1.3. Filtrado de blanqueo.

Un filtro de blanqueo es aquel que cuando se le introduce una señal, arroja una salida cuyo espectro es plano, como ya se ha comentado. Si se modela la señal EMG como un proceso gaussiano que es muestreado, entonces el blanqueado hace las muestras ortogonales entre sí pudiendo ser tratada entonces cada muestra de manera independiente. Esta afirmación se ve con

claridad si se piensa que una densidad espectral plana corresponde implica una función de autocorrelación que es una función δ, de manera tal que dos muestras cualesquiera son independientes.

La respuesta en frecuencia del filtro, debería variar en función del nivel de contracción, pues la señal también lo hace. Dicho filtro puede construirse a partir de un modelo autorregresivo AR, y se ha probado para la señal mioeléctrica obteniendo que los resultados duplican la relación señal a ruido. Mejorar la SNR no sólo mejora la estimación de la amplitud, sino que mejora también a cualquier otra característica que se emplee con cualquier otro clasificador. Un estudio muy completo se encuentra en [21].

Clancy y Hogan [44] usaron un filtro digital MA[30] para contracciones a fuerza constante y probaron que con filtros blanqueadores de cuarto orden, calibrados con una pequeña muestra de datos (5 segundos), se mejoraba la SNR en un 63 % para distintos esfuerzos, si bien, en el caso de una contracción a un 10 % del MVC los resultados introducían ruido. Posteriormente, el mismo Clancy [131] introdujo una mejora, en la cual el filtro se adaptaba al nivel de la señal para superar ese error y no introducir ruido en las señales tenues. Era la aplicación del filtrado adaptativo a los patrones mioeléctricos [104][108]. De todos los algoritmos de blanqueado de la señal, los más adecuados son los filtrados adaptativos [53], ya que la forma del espectro de la señal varía con el tiempo. Dada la gran variabilidad de las señales EMG, la constante de tiempos no puede ser fijada y debe ser adapatada a la secuencia que llegue [25].

6.1.4. Combinación de canales.

Sirve para eliminar las interferencias. Hogan fue el primero en sentar las bases para la estimación de la amplitud a partir de múltiples canales [13], y más tarde otros han actualizado su trabajo [51][63]. La razón de utilizar varios canales es que cuantos más puntos espaciales se muestreen, más preciso es el conocimiento de los procesos eléctricos subyacentes. En el caso de múltiples variables, la función de densidad de probabilidad es normal multivariada y como parámetro característico no está la varianza sino la matriz de covarianza como ya se vió anteriormente en la deducción de los clasificadores estadísticos. El procedimiento es ortogonalizar tal matriz y describir los autovectores.

El uso de cuatro canales por Hogan atacando al bíceps en vez de uno sólo mejoraba la SNR hasta en un 90 %. Sus cálculos de blanqueado etc. no se

[30]En inglés, *moving average*

realizaron en tiempo real, pero más tarde sí se implementaron para el control de prótesis. La mejora disminuía en todo caso para niveles de contracción excesivamente bajos o altos.

Clancy mejoró aún mas el sistema hasta aumentar la SNR en aproximadamente 200 %, usando también cuatro canales o un 300 % usando ocho. Pero tantos electrodos son incómodos en una prótesis real, o a veces el amputado sencillamente no tiene superficie muscular como para acogerlos a todos. En nuestro sistema no se utilizarán varios canales para estimar la amplitud, sino uno sólo, pero queda como trabajo futuro esta ampliación.

6.1.5. Demodulación de la señal.

Este bloque del esquema de procesado sirve para establecer la relación entre la varianza de las muestras de la señal y la fuerza del músculo. Dicha relación entre varianza y fuerza se había tomado habitualmente como de tipo potencial:

$$\sigma(F) = kF^a \tag{58}$$

Shwedyk [5] dedujo teóricamente el valor óptimo de esta a cifrándola en $a = 0,5$. Su demostración teórica partía de un modelo en que representaba a la señal EMG como superposición de trenes MUAPs y de la asunción de que la fuerza se traducía en el número de MU reclutados; asunción válida para contracciones no demasiado fuertes (menores del 70 % MVC). El modelo se ve en la figura 27, donde la entrada es el tren de pulsos ya caracterizado, los a_i son coeficientes de peso, y los filtros h_i son la respuesta al impulso que caracteriza la silueta de cada MUAP.

Figura 27: Modelo de Shwedyk para el origen de la señal EMG

La demostración teórica arroja $a = 0,5$ en la ecuación 58. Si $w(t)$ es un proceso AGWN de varianza 1, y el número de MU activadas es proporcional a la fuerza, la fórmula 58 puede quedar como:

$$x(t) = w(t)\sqrt{F} \tag{59}$$

En la práctica otros valores han funcionado mejor. Hogan [13] realizó también su análisis teórico por otro camino distinto (hacer nulo el sesgo del estimador de la fuerza antes descrito) llegando a la misma conclusión, que $a = 0,5$, pero que desviaciones en su valor no iban a ser dramáticas (para el valor tomado de $a = 1,75$ el sesgo del estimador es sólo de un 4 por 1000); y que de hecho [14] los mejores valores experimentales de a estaban entre 1 y $1,75$.

En la opinión del autor de esta memoria, el que la relación sea de tipo $\sigma(F) = kF^a$ es arbitraria y poco general. Otros autores [21][20] han propuesto relaciones de tipo logarítmica, que no son de tipo potencial, y han obtenido buenos resultados. El autor de esta memoria cree que la relación entre la teoría de la generación de la señal EMG y los resultados experimentales es demasiado vaga, y propone generalizar y recrear el modelo de manera experimental. Propone hacer un desarrollo en serie en el que tampoco harían falta muchos términos:

$$\sigma(F) = \sum_{i=0}^{s} a_i F^i \tag{60}$$

Luego, por ajuste de mínimos cuadrados, se pueden hallar los coeficientes a partir de mediciones simultáneas de la fuerza y de la señal EMG realizadas por un sujeto sano.

Kreifeldt [1] por ejemplo es otro de los que también realizó un estudio de qué no linearidad era más conveniente para representar la relación 58, probando distintos filtros no lineales. Los filtros que probó o bien incorporaban una transformación de tipo potencial(a^b) o bien de tipo radical ($a^{\frac{1}{b}}$). En su estudio teórico y experimental probó elevar las muestras de la señal a las siguientes potencias: $(4, 2, \frac{1}{2}, \frac{1}{4})$. En su estudio teórico no podía determinar qué potencia era óptima por ser dependiente del grado de 'blancura' de la señal.

El resto de los bloques de la figura 26 también los respetaba; en la siguiente etapa situaba un filtro paso bajo lineal para suaviazar la señal, y en la

siguiente aplicaba la transformación inversa. La relinearización fue cuestionada y probó experimentos con ella y sin ella.

Para decidir cuál es lo más apropiado es necesario un criterio; el criterio establecido fue definir una relación SNR donde la señal es la media de la señal a la salida y el ruido es el valor RMS también en la salida del estimador. Su conclusión es que una raíz cuarta sin aplicar relinearizado es lo más apropiado. Su análisis hoy apenas nos es válido. Sus filtros y las funciones fueron 'hardware'[31] y con los medios de hoy podría haberse hecho más sistemático y podrían haberse probado más no linearidades.

Hogan [13] a partir del modelo matemático con que describió a la señal predijo que el procesador óptimo era el cuadrático pero no encontró experimentalmente mejora sobre el rectificador de onda completa. Parker [8] también experimentó con el modelo gaussiano y lo aceptó.

El modelo gaussiano no ajusta perfectamente con los datos, parece ser que la campana gaussiana se escarpa en torno al cero. De hecho Clancy [108] situaba la distribución a medio camino entre gaussiana y laplaciana (MAV ligerísimamente mejor que RMS).

De todo este gazpacho de teorías, propuestas y procesados sólo cabe extraer la conlusión de que la teoría no es suficiente para probar el procesado óptimo y que el mejor procesado se obtiene tras experimentar mucho y contrastar resultados.

6.1.6. Suavizado de la señal.

El suavizado de la señal es importante para evitar cambios bruscos en la prótesis. El suavizado más trivial es el filtro paso bajo RC que se usó en 1952. El filtro paso bajo debe tener una constante de tiempo no demasiado grande, pues de lo contrario aparecería un retardo importante, y no demasiado pequeño, en cuyo caso la SNR empeoraría.

St-Amant [76] hizo un estudio experimental sistemático para comprobar los efectos de la longitud de la ventana de suavizado en la SNR. También Clancy estudió [90] de forma experimental la SNR como función de la longitud de la ventana, el blanqueo, el número de canales y el tipo de detector para la amplitud (MAV/RMS). Los resultados obtenidos fueron comparados con los predichos por [115], y siempre resultaron tener menor bondad

[31] Sólo contaba con un PDP-8

que lo dicho, si bien sí seguían vagamente la relación. Se probó que tanto el blanqueado de la señal como la adquisición multicanal mejora la SNR de los estimadores, y que el tamaño de la ventana debe ser alargado en condiciones estáticas y reducido en dinámicas, como dicta el sentido común, infiriéndose a partir de aquí un tamaño de ventana ajustable.

La señal adquirida cuando el músculo se fatiga difiere de la de un músculo no fatigado. El problema es que las señales se transmiten con mayor lentitud, con lo cual el espectro se desplaza a frecuencias más bajas. Se ha propuesto también algún método de compensarlo cuando se trate de controlar las prótesis. También se han tratado de compensar directamente los errores del usuario al emitir los comandos [39].

6.1.7. Relinearización.

Para devolver la amplitud de la señal EMG a su rango de valores original ha de aplicarse la no linearidad introducida anteriormente. Dicho de manera más precisa, el relinearizador posibilita la estimación insesgada de la señal muscular, trae la media de la estimación a la media del estimado. Sin embargo para extraer las características para la clasificación no es obligatorio relinearizar; de hecho Kreifeldt puso en duda la necesidad de hacerlo [1].

6.2. Parámetros sencillos en el dominio del tiempo

- Valor absoluto medio(MAV)[32]. También es llamado IAV[33]. Se calcula como:

$$MAV = \frac{1}{N}\sum_{n=1}^{N}|x(n)| \tag{61}$$

- Varianza

$$VAR = \sigma^2 = \frac{1}{N}\sum_{n=1}^{N}x^2(n) \tag{62}$$

  Si la función de densidad de probabilidad de las amplitudes de las muestras fuera gaussiana, debería cumplirse que [13]:

$$VAR = \sqrt{\frac{\pi}{2}}MAV \tag{63}$$

  Sin embargo se verificó en el laboratorio que dicha relación no se cumplía.

[32]Del inglés *median absolute value*
[33]Del inglés *integral absolute value*

- Momentos de orden superior (hasta orden 5)

$$\sigma^3 = \frac{1}{N}\sum_{n=1}^{N}|x(n)|^3 \tag{64}$$

  e igualmente para ordenes superiores.

- Cruces con cero (ZC[34]).

$$ZC = \frac{1}{N}\sum_{n=1}^{N} sgn(-x(n)x(n+1)) \tag{65}$$

  En presencia de ruido éste y otros parámetros que representen la frecuencia introducen un cierto sesgo. El número de cruces con cero, la frecuencia mediana y la frecuencia media fueran comparados por [33], quien halló en ésta última mayor solidez ante el ruido.

- Frecuencia mediana. El valor mediano de la frecuencia se ha usado como característica, se puede encontrar en [26] y en [31], y puede ser calculado analógicamente o mediante FFT.

- Momentos frecuenciales de orden superior. Si la frecuencia media es el momento de frecuencia de orden 1, también los momentos superiores pueden aportar información, y también se han usado para el análisis de la señal mioeléctrica [57]. Puede expresarse como:

$$M_k = \frac{\int_0^F f^k P(f)df}{\int_0^F P(f)df} \tag{66}$$

- Longitud de la señal

$$WL = \frac{1}{N}\sum_{n=1}^{N}(x(n) - x(n-1)) \tag{67}$$

- Amplitud Willison

$$WAMP = \sum_{n=1}^{N} f(|x(n) - (x(n+1)|) \tag{68}$$

  con

$$f(x) = \begin{cases} 1 & \text{if } x > \text{umbral} \\ 0 & \text{en otro caso.} \end{cases} \tag{69}$$

[34]Del inglés *zero crossings*

6.3. Información frecuencial

La forma del espectro de frecuencia responde a procesos subyacentes que pueden ser investigados [7]. En este apartado se revisan los modelos autorregresivos, los coeficientes cepstrales y las transformadas de Fourier como orígenes de características susceptibles de ser seleccionadas.

6.3.1. Modelos autorregresivos

Graupe [2] fue el primero en proponer el uso de los parámetros de un modelo autorregresivo como características de los patrones mioeléctricos. El ajuste de las muestras de la señal a un modelo autorregresivo presupone que la señal es estacionaria. Esto es falso para la señal EMG, como ya hemos dicho hasta la saciedad, pero considerando fragmentos de la señal suficiemente cortos, patrones pequeños, se suaviza la no estacionariedad. El hecho de que el inicio de un movimiento del brazo suponga una no estacionariedad ha sido aprovechado en sí mismo, es decir, ¿por qué no detectar el abandono del reposo a partir de una detección de la no estacionariedad asociada? D'Alessio [38] lo propuso monitorizando las frecuencias media y mediana de la señal.

Si los parámetros para distintas funciones son suficientemente distintos, se podrá establecer la separabilidad de las funciones.

En realidad en el método de Graupe la característica sobre la que opera el clasificador, no son los parámetros del modelo AR, sino la *señal de error* que se obtiene como la diferencia de la muestra que se obtiene de la muestra que se esperaba obtener para una clase determinada de aplicarse correctamente el modelo autorregresivo.

Graupe propuso un modelo ARMA[35] porque era un sistema lineal que proporcionaba la discriminación de funciones con un número mínimo de parámetros, lo cual en 1975, no disponiendo de potentes microcontroladores, era realmente decisivo[36]. Recogió la señal en el bíceps, tríceps y definió en un principio tan sólo cuatro estados. En 1978 presentó una notable mejora del sistema [9], donde propuso en concreto 5 estados de la prótesis (2 DOF más reposo) con un sólo par de electrodos.

[35] De *autoregressive moving-average*

[36] Se proponía el uso del hoy vetusto microprocesador Intel 8008.

En esta aproximación se asume que el patrón se ajusta al modelo:

$$x(n) = \sum_{i=1}^{s} \hat{a}_i x_{n-i} + \hat{e}_n \tag{70}$$

Es decir, que cada muestra se expresa como una combinación lineal de las anteriores muestras más un término de ruido e_n. La hipótesis asumida es que los valores de los coeficientes a_i son fijos para cada individuo y para cada movimiento definido, lo cual sirve para caracterizar al patrón. El periodo de entrenamiento servirá para hallar los coeficientes a_i que determinan la clase, la manera de hallarlos es la minimización de la señal de ruido e.

El orden s del modelo debe ser suficiente como para que el término de ruido sea pequeño, en realidad, se escoge una s para la cual el término de ruido e_n pueda considerarse blanco. Graupe la propuso de 3 a 10, aunque finalmente se queda con $s = 3$, aunque también se han propuesto órdenes tan altos como 12 [32].

Con los patrones de entrenamiento se calculan los coeficientes del modelo AR. Los acentos circunflejos indican que los coeficientes son *estimados*, la estimación se puede realizar con distintos algoritmos. El más usado (y el que propone Graupe) es el algoritmo RLS [37]. Así, si

$$\hat{a}_j = [\hat{a}_1, \ldots, \hat{a}_s]_j^T, \tag{71}$$

y

$$X_n = [x(n-1), \ldots, x(n-s)]^T \tag{72}$$

entonces los coeficientes del modelo AR se calculan como:

$$\hat{a}_n = \hat{a}_{n-1} + P_n X_n (x(n) - \hat{a}_{n-1}^T X_n) \tag{73}$$

Hallándose P_n en:

$$P_n = P_{n-1} - Pn - 1X_n(1 + X_n^T P_{n-1} X_n)^{-1} X_n^T Pn - 1 \tag{74}$$

Y una vez con los coeficientes del modelo AR se pueden hallar los del coeficientes ARMA y a partir de ellos diseñar un filtro Kalman que luego el mismo Graupe señala que no tiene importancia. De hecho en su trabajo posterior[38] [9] descarta el uso de ARMA para hacer más rápido el algoritmo y utiliza tan solo el modelo AR, sin perder mucho en el cambio.

[37] Del inglés *recursive least square*

[38] En 1978 ya contaba con un microprocesador 8080

Una vez hallados los coeficientes en el periodo de entrenamiento, tan sólo se trata de hacer pasar a la señal por un filtro Kalman, y obervar la salida del error. Se compara la $x(n)$ que se observa con la $\hat{x}_k(n)$ que se esperaba para cada una de las clases, y entonces se asigna a la mejor clase con tal de que el error sea menor que un determinado umbral.

Para calcular el error se toma una longitud de la señal que es variable y que depende de unos parámetros ρ_m. Así, para la función k-ésima el error se estima como:

$$E_k(n) = \frac{1}{N-p} \sum_{j=n-N+p+1}^{n} (x(n) - \hat{x}_k(n))^2 \tag{75}$$

Para que una función se activara Graupe solicitaba tres condiciones. La primera es que la función k elegida debía ser la que tuviera menor error $E_k(n)$. La segunda es que además el error debía ser pequeño, debía estar limitado: $E_m(n) < \rho_k E_{mk}$ (donde E_{mk} es el error medio durante el entrenamiento). La tercera condición es que la potencia de la señal EMG superara un cierto umbral, a fin de que no se dispararan con el ruido por casualidad.

El modelo AR es completamente óptimo si la señal es gaussiana, pero aunque no lo sea, sigue siéndo linealmente óptimo, es decir, el mejor modelo lineal. Un defecto muy importante del sistema, sin embargo, es que es sensible a la amplitud de la señal y si se opera sin normalizar la señal fácilmente se obtienen errores.

El trabajo de Graupe presenta serias limitaciones. Utiliza un sólo canal, y no aprovecha la correlación espacial entre los canales. No permite más de una función simultáneamente. Así que fue extendido por Doerschuk [16] intentado superar esas limitaciones. Amplió el número de electrodos a cuatro y fijó tres grados de libertad. Para ello Doerschuk propuso aprovechar la señal de autocorrelación en cada electrodo, como hiciera Grape, pero también la correlación cruzada entre los distintos electrodos, lo cual era novedad. Como clasificador utilizó asimismo un modelo probabilístico bayesiano gaussiano.

Así, el modelo ARMA de una sóla dimensión de Graupe queda ahora en forma vectorial y matricial.

$$\vec{x}(n) = \sum_{j=1}^{s} A_{kj}\vec{x}(n-j) + \vec{e}_k(n) k = 1, \ldots, K \tag{76}$$

Donde K es como antes el número de clases y k es una clase en concreto y s es el orden del modelo AR. Ahora $\vec{e}_k(n)$ no es un error, sino el vector de errores de todas los electrodos, vector de L componenes porque hay L

electrodos. El error no queda caracterizado sólo por una varianza, sino por una matriz de covarianza S_k. Del mismo modo A_{kj} es una matriz de $L \times L$ con los coeficientes para la clase k-ésima.

La estimación de la muestra que ha de llegar se calcula entonces como

$$\hat{\vec{x}}_k(n) = \sum_{j=1}^{s} A_{kj}\vec{x}(n-j) \tag{77}$$

Se halla entonces el error $e_k(n) = x(n) - \hat{x}_k(n)$. Éste debería corresponder a un ruido blanco gaussiano para una clase. En la elección en cada instante, no se escoge una clase y_k en tanto que el ruido $E(n) < \rho_m E_{km}(n)$ y es en todo igual como hizo Graupe. Esa constante ρ es arbitraria y se halla probando diversos valores.

Posteriormente, en 1985 Triolo [24] demostró que el sistema de Graupe no era sino un caso particular del de Doerschuk bajo ciertas circunstancias, y además formalizó el trabajo previo. Probó que si en el modelo de Doerschuk [16] la varianza del error para cada clase es la misma y que las probabilidades de elegir una función son iguales, entonces el sistema de Graupe era sólo un caso especial del de Doerschuk.

El clasificador que ha de utilizarse para estos modelos es el bayesiano gaussiano. Ahora $P(x(n)|y_k)$ es un vector aleatorio gaussiano (cuyas componentes son las muestras de la señal en los distintos electrodos) de media $\hat{\vec{x}}_k(n)$ y matriz de covarianza S_k.

Así por ejemplo, la probabilidad de que llegue una muestra de amplitud $x(n)$ conociendo las anteriores hasta $x(n-1)$ es:

$$P(x_k(n)) = (2\pi\sigma_x^2)^{-1/2} \exp\left(-\frac{1}{2}(x(n) - \hat{x}(n))^2/\sigma_x^2\right) \tag{78}$$

donde σ_x^2 es la varianza de la señal. Así que esta función de probabilidad de cada muestra es sólo función del error residual $(x(n) - \hat{x}(n))$. Extendido a que las muestras en instantes anteriores hayan de satisfacer el modelo queda:

$$P((x(n)\dots x(n-N+1))|(x(n-1)\dots x(n-N+1))) = (2\pi\sigma_x^2)^{-N/2} \cdot \\ \cdot \exp\left[-\frac{1}{2\sigma_e^2}\sum_{j=0}^{N-1} e^2(n-j)\right]$$

así que se en el LDA las funciones a tomar como distancias son:

$$\sum_{j=0}^{N-1} e_k^2(n-j) \tag{79}$$

y se toma la mayor.

Específicamente se han desarrrollado algoritmos para estimar los parámetros cambiantes con el tiempo de un modelo autorregresivo; parámetros correspondientes a los coeficientes menos sesgados [34]. Cabe decir además que este modelo está basado directamente en el contenido espectral de la señal, y que en el plano Z la función del sistema se expresa directamente como:

$$H(z) = \left[1 + \sum_{i=0}^{s} a_i z^{-i}\right]^{-1} \tag{80}$$

La validez de estos sistemas basados en la represenación de la señal con un modelo autorregresivo estaba condicionada, según se comentó, a la estacionariedad de la señal EMG. Esta se ha asumido para periodos cortos de tiempo, digamos 80ms. Sin embargo hay algo más que objetar. Y es que en cada contracción, digamos, que dure 500ms, según suponen Graupe y Doerschuk, cada uno de los fragmentos responden al mismo modelo AR. Ello no es cierto según Sherif objetó poco después [19].

La idea de partir el patrón en varios subsegmentos y suponer que en cada uno de ellos existe estacionaridad sí es válida. Sin embargo, la señal muscular tiene como mínimo dos fases, decía Sherif, en cada una de las cuales hay un espectro de potencia diferente, y así en la primera fase la energía se concentra por ejemplo en bandas más bajas. La energía total de cada subsegmento además es distinta, y convendría una normalización respecto a una varianza dada.

6.3.2. Coeficientes cepstrales

Kang [54] propuso el uso de los coeficientes cepstrales para la clasificación de los patrones mioeléctricos, comparando la separabilidad con la obtenida con los coeficientes AR y estableciendo la superioridad de estos últimos.

La señal cepstral se define como la transformada inversa de Fourier del logaritmo del espectro de potencia de la señal. Trabajando en el plano z, la función $\ln H(z)$ puede ser expresada por la serie de Laurent:

$$\ln H(z) = C(z) = \sum_{i=1}^{\infty} c_i z^{-1} \tag{81}$$

Los coeficientes cepstrales c_i pueden ser hallados juntando 80 y 81 como:

$$c_i = -a_i - \sum_{n=1}^{i-1} \left(1 - \frac{n}{i} a_n c_{i-n}\right) \tag{82}$$

con $c_1 = -a_1$. Una vez hallados los parámetros, el clasificador propuesto por Kang fue el de máxima similitud, el euclídeo o el de Mahalanobis.

Los parámetros adaptativos cepstrales (ACV) también han sido propuestos para clasificar a los patrones EMG. Lee [62] propone un algoritmo a medias entre el procesado adaptativo y procesado fijo. Los coeficientes cepstrales son hallados con el algoritmo RLS, pero utilizando un algoritmo adaptativo. Adquirió la señal del bíceps y del tríceps y propuso tres grados de libertad; comparó entonces los resultados con los obtenidos con el modelo AR, y evidenció la superioridad de los suyos. La bondad de los coeficientes ACV como características de los patrones mioeléctricos también es probada en [85].

6.3.3. Transformadas de Fourier

La transformada de Fourier ofrece algo de información sobre el patrón mioeléctrico, pero esta información no es fiable por no ser estacionaria la señal. La transformada de Fourier continua se define como:

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi tf} dt. \tag{83}$$

No es de nuestro interés trabajar con funciones continuas, sino discretas, de modo se podría pensar en hallar la DFT (transformada de Fourier discreta) de cada patrón:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn} dt. \tag{84}$$

Los coeficientes de la DFT no sirven tal cual están como características del patrón pues son tan numerosas como las muestras de la señal. Sin embargo, ya puede ir pensándose en aplicar algún tipo de reducción más eficaz a partir de la DFT. El número de operaciones para realizar una DFT es del orden de N^2, es decir, unas 51000.

Si la DFT se realiza sobre segmentos de un número potencia de 2, entonces es aplicable un algoritmo que reduce notablemente las operaciones a realizar,

la FFT. Con la FFT sólo hacen falta una cantidad aproximada de $Nlog_2N$ operaciones, es decir, de 1800. 50000 operaciones está rozando el límite de lo razonable para que nuestros equipos pudieran trabajar en tiempo real.

Estas transformaciones tiene sentido aplicarlas si las señales son estacionarias, si no evolucionan en el tiempo. En caso de que varíen en el tiempo, carecerían de sentido. Si se aplica una FFT sobre un patrón en el que al principio se estaba en reposo y al final ya hay una contracción, el resultado tendrá características extrañas. Tomando patrones suficientemente cortos, habrá un gran número de ellos que sean válidos y sólo algunos serán no válidos. Pero hay mejores maneras de solucionar este problema.

6.4. Representaciones de la señal en tiempo y en frecuencia.

Las representaciones de la señal en tiempo - frecuencia combinan el análisis de la señal en ambos dominios temporal y frecuencial. Ha sido utilizada en casi todos los campos del procesado de la señal, como el compresión de la señal, la codificación, el filtrado, la eliminación de ruido, o los problemas de detección, clasificación y visualización. Aquí nos es de interés porque es una estupenda caracterización de la señal que facilitará la clasificación de los patrones mioeléctricos. La intención de este apartado es dar una idea comprensiva al lector sin entrar en detalles.

Las representaciones en tiempo - frecuencia se dividen en *métodos lineales* (como la STFT y la transformada wavelet) y los *cuadráticos* (distribución Wigner-Ville). Es una teoría complicada y extensa que desborda los propósitos de este proyecto, pero que no puede dejar de ser mencionada como tendencia más importante en la extracción de características para la clasificación de patrones mioeléctricos.

6.4.1. Transformada STFT

Hacer una STFT (Short Time Fourier Transform) de un patrón equivale a hacer DFTs en pequeños trocitos del patrón tras ser enventanados. Aquí se verá cómo y por qué utilizar esta transformada.

Los coeficientes de la transformada discreta de Fourier, los $X(k)$, dan la distribución de la señal en el dominio de la frecuencia en toda la señal, sin dar resolución temporal, y las muestras $x(n)$ de la señal dan información

sólo en el dominio temporal sin dar información frecuencial. La ventaja de la STFT es que da información de la señal tanto el dominio temporal como en el frecuencial.

La STFT considera a la señal como estacionaria sólo durante intervalos limitados de tiempo, en esto supera a la DFT, y se define como:

$$STFT(t,f) = \int x(\tau) g^*(\tau - t) e^{-2j\pi f\tau} d\tau \tag{85}$$

Direcciona la señal en un plano bidimensional t y f, aplicando una transformada de Fourier a los pedacitos tras un enventanado.

Hay dos pararámetros en los que se debe centrar la antención, que son la *resolución temporal* Δt y la *resolución frecuencial* Δf. La resolución indica con qué precisión se conocerá la información en un dominio o en otro; la mejora de una resolución implica el empeoramiento del otro. Si se toman fragmentos grandes de la señal en el tiempo, la DFT tendrá más puntos y arrojará más precisión, pero a costa de hacer la partición en el tiempo más gruesa; y viceversa, un grano del tiempo fino dará sólo una tosca DFT.

Las resoluciones dependen de la función ventana $g(t)$. Dada una función ventana $g(t)$ y su transformada de Fourier $G(f)$ se puede tomar como medida de la resolución frecuencial al ancho de banda medio cuadrado:

$$\Delta f^2 = \frac{\int f^2 |G(f)|^2 df}{\int |G(f)|^2 df} \tag{86}$$

y la resolución temporal es el ancho cuadrático medio de la señal en el dominio del tiempo:

$$\Delta t^2 = \frac{\int t^2 |g(t)|^2 dt}{\int |g(t)|^2 dt} \tag{87}$$

Una u otra resolución pueden ser arbitrariamente pequeñas, pero el producto de ambas resoluciones está limitada por el principio de incertidumbre (desigualdad de Heisemberg):

$$\Delta f \Delta t >= 1/(4\pi) \tag{88}$$

La ventana gaussiana es la que consigue llegar a la igualdad, siendo el

mejor caso. En este caso particular, la transformada STFT se denomina *transformada Gabor*. La ventanta que se elija, en todo caso, debe de poder ser calculada con rapidez. Ahora se va a escribir la STFT de una manera más conveniente. La DFT de la ecuación 84 puede ser reescrita como:

$$X(m) = \sum_{i=1}^{L-1} x[i] e^{-2j\pi(mF)(iT_s)} \tag{89}$$

donde T_s, f_s son el periodo y frecuencia de muestreo. $F = \frac{1}{LT_s}$ es el tamaño del paso de la frecuencia de muestreo. Así la STFT queda como una serie de DFTs:

$$STFT(k,m) = STFT(kTs, mF) = \sum_{i=1}^{L-1} x[i] g[i-k] e^{-2j\pi mi/L)} \tag{90}$$

$T = KT_s$ es el tamaño del paso del muestreo temporal, el equivalente en el tiempo a F. Ahora sí que se puede escribir la STFT en función de estos dos parámetros fundamentales, los pasos de muestreo temporal - frecuenciales. Se escribe como:

$$STFT(k,m) = STFT(nT, mF) = \sum_{i=1}^{L-1} x[iT_s] g[iT_s - nT] e^{-2j\pi mi/L)} \tag{91}$$

Si $K = L$ no se pisan las sucesivas ventantas de análisis.

La STFT tiene muchas propiedades útiles incluyendo una teoría desarrollada bien conocida, una interpretación sencilla y una buena eficiencia computacional. Por contra tiene el fallo de que la rejilla espacio temporal es fija.

Se define *espectrograma* como la magnitud cuadrada de la STFT. Da información de la energía, mucho mejor para ser representada ya que no incluye partes real e imaginaria. Se dice que el periodograma es una representación temporal - frecuencial *cuadrática*. Farry [66] utiliza una modificación del periodograma, llamado método de múltiple ventana de Thomson, ofreciendo 2 ó 3 DOF a partir de un solo canal.

6.4.2. Transformada wavelet

La transformada Wavelet (WT) no tiene fija las resoluciones Δt y Δf en el plano tiempo - frecuencia, como la STFT. Para analizar la señal EMG, en un momento dado quizás podamos suponer que la contracción es constante o el reposo es prolongado, de modo que la resolución temporal interese poco y la frecuencial pueda ser afinada más. Y en otros momentos, quizá sepamos que se está realizando un movimiento, se está iniciando una contracción y la señal varía su estructura con rapidez. Entonces lo adecuado sería reducir al mínimo el tamaño de las ventanas sobre las que analizar la frecuencia. Esta flexibilidad sólo la puede procporcionar la transformada wavelet y sus familiares.

Desde que los ordenadores ahora tienen capacidad para calcular en tiempo real la transformada wavelet de patrones mioeléctricos, y siendo ésta una herramienta más potente, ha sido utilizada para diagnosis médica [68] (Por ejemplo se ha usado para analizar la fatiga muscular [142]) y para control.

La transformada Wavelet continua (CWT) cubre el espacio tiempo - frecuencia de una manera variable y se dfine en la ecuación 92.

$$CWT(\tau, a) = \frac{1}{\sqrt{a}} \int x(t)\psi^* \left(\frac{t-\tau}{a}\right) \tag{92}$$

Donde ψ es una función ventana prototipo que se llama función wavelet madre. El análisis determina la correlación de la señal con versiones de la señal escaladas por a y desplazadas en τ.

La implementación discreta de la CWT se puede calcular directamente convolucionando la señal con una version escalada y dilatada de la wavelet madre, si bien es más eficiente hallarlo vía FFT. En la versión discreta, $a = a_o^j$ y $\tau = na_0^{-j}$ con j y n enteros; y entonces ya no se llama CWT sino DWT (*transformada wavelet discreta*). Escoger $a = 1$ y $\tau = n2^{-j}$equivale al caso continuo; lo habitual es elegir $a = 2^j$ y $\tau = n2^{-j}$ (base diádica). Cuando se habla de transformada wavelet, se sobreentiende que se habla de la CWT con base diádica.

La *transformada wavelet packet* es una versión generalizada de la CWT y de la DWT. La transformada es redundate y permite elegir muchas bases ortonormales simultáneamente, de manera que la división en tiempo - frecuencia es configurable, la partición se ajusta a las necesidades de la señal.

Desde hace tiempo se han propuesto los wavelet packets como método para analizar la señal EMG [102][64][83]. Distintas transformaciones fueron comparadas por Karlsson [122] para la señal EMG ofreciendo un interesante informe que incluían las consideraciones conretas para la señal EMG de lo aquí comentado.

El cómo rinden estas transformaciones temporales - frecuenciales para la clasificación de patrones mioeléctirocs es el asunto principal de las investigaciones de K. Englehart de estos últimos años y son una referencia ineludible; entre otras cosas porque detalla cómo debería ser la selección y proyección de este conjunto tan vasto de características en uno manejable sin perder información.

6.5. Evaluación de las características.

Para evaluar la bondad en nuestra elección de características un buen método sería comparar las distintas tasas de errores que originan en la salida del clasificador según alguno de los métodos ya comentados. Sin embargo, la opción más común es aprovechar alguna medida de la separabilidad de las clases que ya esté bien establecida, como es la *distancia de Bhattacharyya*.

6.5.1. Distancia de Bhattacharyya

La *distancia de Bhattacharyya* es un caso particular de la distancia de Chernoff y se encuentra bien descrita en Fukunaga [30]. La distancia entre dos clases k y j se define aquí como:

$$\mu(1/2) = \frac{1}{8}(m_k - m_j)^T \left(\frac{C_k + C_j}{2}\right)^{-1} (m_k - m_j) + \frac{1}{2}\ln \frac{|\frac{C_k+C_j}{2}|}{\sqrt{|C_k||C_j|}} \qquad (93)$$

La distancia de Bhattacharyya ha sido ampliamente utilizada para la medida de la separación de las distribuciones de los patrones mioeléctricos. Por ejemplo, Park [85] justificaba su elección de características mostrando esta distancia. También lo ha usado Han [133].

7. Recapitulación

Aquí se recapitula sobre las características propuestas para definir la señal EMG y los clasificadores posibles para reconocer los patrones.

7.1. Resumen de clasificadores propuestos.

Clasificador	Tipo	Referencia
QDF	Estadístico paramétrico	ec. 17
MDF	Estadístico paramétrico	ec. 18
LDF	Estadístico paramétrico	ec. 20
EDF	Estadístico paramétrico	ec. 21
k Vecinos	Estadístico no paramétrico	ch. 5.2.5
Red neuronal	De aprendizaje	ch. 5.3.1
Lógica difusa	De aprendizaje	ch. 5.3.2

Cuadro 3: Resumen de clasificadores

7.2. Resumen de características propuestas.

Característica	Referencias	Separabilidad	Cálculo
MAV	[143][42][85][58][133]	***	*
σ^3,σ^4 etc.	[15][1]	**	*
LOG	[1]	***	*
VAR	[15]	**	*
ZC	[15][17][42][143][133][61][87]	***	*
HISTO	[58]	***	*
MAVSLP	[15] [147]	*	*
SSC	[42]	*	*
Amplitud Willison	[58]	*	**
WL	[42]	*	*
Parámetros AR	[2][9][16][54][61]	**	*
Parámetros CV	[54][53]	***	**
Parámetros ACV	[54]	***	***
STFT	[106]	***	***
WT	[106]	***	***
WPT	[106]	***	***
Periodograma Thomson	[66]	***	***

Cuadro 4: Características usadas para clasificar patrones mioeléctricos.

8. Sistema realizado

Esta sección describe minuciosamente el trabajo realizado en el laboratorio del Grupo de Bioingeniería. Se descompone en cuatro apartados. El primer apartado resume el trabajo previo al comienzo de este proyecto. El segundo apartado describe el diseño y la arquitectura propuestos en este trabajo como solución integradora, justificándose las decisiones tomadas. El tercer apartado, el 'manual de usuario' es una lectura obligada para quien desee tan sólo manejar los programas sin interesarse por el cómo han sido hechos; ofrece las funcionalidades de los programas sin atender a su construcción. El cuarto apartado, el "manual de referencia", profundiza en los aspectos técnicos del desarrollo de los programas, es una referencia para el programador que desee dar continuidad a estos programas o para el usuario que quiera conocer más detalles de las aplicaciones.

8.1. Trabajo previo

Al comenzar el trabajo que se presenta en esta memoria existía un material previo importante que había de ser reutilizado; este trabajo es sólo continuación del anterior y tiene vocación de ser mejorado continuamente en sucesivos proyectos.

Varios proyectos habían sido realizado como partes de un sistema global pero faltaba realizar el engranaje, el trabajo que aquí se presenta tiene como objetivo explícito integrar todo el trabajo anterior y dejar las bases para que el trabajo pueda ser continuado.

El trabajo realizado hasta ahora se resume en:

1. Brazo. El "Brazo Mioeléctrico Virtual"se presentó en noviembre de 1997 por Antonio Hernández Bajo [69]. Es una aplicación gráfica que muestra un brazo. El brazo dispone de tres grados de libertad, los movimientos se llevaban a cabo mediante el teclado. Los movimientos que permite son pronación y supinación de la muñeca, extensión y flexión del codo y apertura y cierre de la mano. Además permitía recorrer un circuito de entrenamiento con diversos ejercicios objetivo.

   Fue programada en Visual C++ y se utilizó el estándar gráfico Open GL. La aplicación es un proyecto apoyado en las biblioteca de clases de las Microsoft Foundation Classes (MFC).

2. Cabecera analógica P4. El cuarto y actual prototipo de cabecera analógica fue presentado en 1999 por Ramón de la Rosa [93]. Es un circuito electrónico que amplifica la señal bioeléctrica recogida hasta niveles aceptables para el conversor analógico / digital. Diseñado para cuatro canales, sólo están en servicio dos actualmente si bien el dispositivo está en continua mejora. Por señal bioeléctrica entendemos a señal de electromiograma, de electroencefalograma o de electrocardiograma (EMG, EKG o EEG). En la imagen 28 se muestra por ejemplo una señal de EKG adquirida con la cabecera P4.

Figura 28: Señal de electrocardiograma.

Es un amplificador diferencial que está adecuado a las características de las señales bioeléctricas, como su banda de trabajo, su ganancia, su alta impedancia de entrada y su rechazo al modo común[39].

La cabecera permite varias opciones. Permite añadir una ganancia de 20dB, un filtrado paso bajo con frecuencia de corte en los 40 Hz, y un filtro elimina banda en los 50 Hz (a fin de eliminar el ruido de las líneas eléctricas). Tiene un filtro paso alto con frecuencia de corte en 3 Hz que no puede ser desactivado. Eventualmente se puede utilizar un filtro conmutado a cualquier frecuencia pero ha de suministrársele una señal cuadrada de la frecuencia elegida; dicha opción no ha sido utilizada en este proyecto.

Además de los pines de salida con la señal amplificada y de los pines de entrada de la señal, tiene también varios bits más a la entrada para configurar sus opciones. El propio autor se valió de un conversor A/D comercial (CIODAS 1602[40]) y una aplicación desarrollada en Visual Basic para comprobar su funcionamiento con la que no se ha contado en este trabajo.

[39]El modo común de un amplificador diferencial es la señal común a los dos electrodos
[40]De la casa Computer Boards Inc.

Figura 29: En primer plano se ve la cabecera P4.

La ganancia la cifró experimentalmente en 70.2 dB, es decir una ganancia de 3236 (antes de la ganancia extra de 20dB), el amplificador principal es un amplificador de instrumentación INA114. Tiene un prefiltro paso bajo de 4 kHz.

3. Convesor A/D y representación gráfica de la señal. Presentados por Lorenzo Ferrero en 1999 como proyecto fin de carrera de la EUP.

   El conversor analógico / digital tiene un rango de entrada de -5 a +5 voltios y ofrece su salida codificada con 12 bits por un cable que se debe conectar al puerto paralelo de un PC. Su frecuencia de muestreo puede ser elegida entre 512 Hz y 1024 Hz, y puede elegirse que digitalice un canal o cuatro canales simultáneamente. Dado que no hay disponibles tantos pines en el puerto paralelo para transmitir, los 12 bits se mandan de dos veces, enviándose primero 8 y luego 4 bits. Su configuración es determinada por otros pines del conector al puerto paralelo.

   `biosad` es una aplicación que recogía las muestras del puerto paralelo siguiendo los protocolos esperados de su cabecera analógia y mostraba la señal a la manera de un registro continuo en pantalla. Se permitía almacenar secuencias temporales y se daba a escoger entre representar uno o los cuatro canales. Al inicio de la aplicación se ofrecía un

Figura 30: La placa conversora A/D.

menú para configurar la tarjeta conversora y también la cabecera P4 a través suyo. Su programación se llevó a cabo también con el compilador Visual C++ aprovechando la biblioteca de clases MFC.

4. Trabajos teóricos previos. También en la ETSIT se han presentado varios proyectos que abordaban directa o indirectamente el problema aquí planteado pero de una manera teórica o trabajando con datos no obtenidos de este laboratorio. Sus algoritmos fueron escritos en Matlab y sin intención de ser ejecutados en tiempo real, por lo que la influencia que han tenido en la elaboración del presente proyecto ha sido prácticamente nula. Sin embargo, sentadas aquí ya las bases, se mencionan dado que un interesante futuro trabajo sería su inclusión en el sistema y su adaptación a la realidad. Ello aprovecharía al máximo los conocimientos y la experiencia de los miembros del Grupo de Bioingeniería.

   Tres de estos trabajos proponían la extracción de los coeficientes wavelet de la señal EMG, pero con fines de diagnóstico [79][80][91], en tanto que otro se centraba en el problema del control de prótesis sugiriendo wavelets y redes neuronales [96]. Los algoritmos que están programados lo están en Matlab, de modo que su transformación a lenguaje C no

iba a ser inmediata[41].

Aun siendo buenos trabajos por separado, al comenzar este proyecto faltaba integrarlos en uno sólo, pues el autor de cada parte había seguido su propio criterio. El siguiente esquema puede ser aclaratorio de la situación inicial.

Figura 31: Elementos previos a la elaboración de este proyecto

8.2. Arquitectura del sistema

No falta una caracterización en UML del sistema que nos ocupa [141][124][129][136]. La arquitectura del sistema diseñado que propongo se estructura en tres capas, como se ve en la figura 32.

Figura 32: Capas en que se estructura el sistema.

1. La capa inferior engloba a todos los dispositivos hardware, y a los componentes software proporcionados por los fabricantes de las tarjetas. En

[41]Si bien el propio fabricante ofrece la posibilidad de una conversión automática.

el presente momento, esta primera capa engloba a la cabecera analógica P4, al conversor A/D diseñado por Lorenzo Ferrero, al conversor A/D comercial CIODAS1602 y al software proporcionado por esta última.

2. La segunda capa se materializa en el módulo llamado `biomedida`. Los servicios que ofrece la capa 2 a las aplicaciones de capa 3 son los siguientes:

   - Configurar e iniciar la adquisición de datos
   - Adquirir los datos y almacenarlos en un *buffer*, para entregarlos a las aplicaciones en el momento en que éstas los soliciten. El dispositivo de capa 1 es posible que requiera unas peticiones de datos síncronas, sin embargo es deseable que las aplicaciones de capa 3 puedan recoger los datos de manera asíncrona; la capa 2 es el intermediario.
   - Finalizar el flujo de datos.
   - Proporcionar un procesado de la señal básico (por ejemplo, limpiar la componente de continua de la señal)

   `biomedida` es una librería de enlace dinámico (DLL) que proporciona a las aplicaciones de capa superior un flujo formateado estándar de los datos, independientemente del origen de los mismos.

   Permite la multiplexación, de modo que una única entrada de datos pueda servir a más de una aplicación ejecutándose simultánemante. La primera aplicación que invoque los servicios de `biomedida` será capaz de configurar a los elementos de la capa 1 a su designio (eligiendo qué filtros analógicos actúan por ejemplo), en tanto que las sucesivas aplicaciones que invoquen los servicios de `biomedida` estando la primera aplicación en servicio, habrán de aceptar la configuración impuesta por la primera, pues no se puede configurar de dos maneras distintas y a la vez un mismo dispositivo.

   Cuando una aplicación de capa 3 se inicie, podrá elegir el flujo de datos de varias fuentes. Actualmente, podrá solicitarlos del conversor de Lorenzo, del conversor comercial CIO/DAS, de un archivo, o finalmente como esclavo de otra aplicación maestra. Estas fuentes son fácilmente extensibles, pues si llegara un nuevo conversor, bastaría con implementar un par de funciones con apenas una veintena de líneas de código, el trabajo *sucio* ya está hecho.

3. La tercera capa la componen las aplicaciones. Entre ellas están tanto las que tienen una utilidad final, como mostrar en pantalla una secuencia

continua de la señal, como las que no, destinadas a ofrecer servicios a las otras aplicaciones. `vroddon` es el módulo de capa tres que ofrece servicios a las otras aplicaciones. No tiene, por tanto, ningún ejecutable, y adopta la forma de librería de enlace dinámico en Windows.

Estos servicios son:

- Cálculo de parámetro básicos de la señal, tal como cruces con cero, varianza de la señal etc.
- Establecimiento de los vectores a ser utilizados a la hora de decidir, a partir de los datos del entrenamiento.
- Función que decide a partir de un patrón.

Figura 33: Arquitectura del sistema propuesto.

8.3. Manual del usuario

Junto con esta memoria se presenta un CDROM. Para instalar el CDROM, basta con ejecutar el programa 'setup' que se incluye en el mismo y seguir las instrucciones. Automáticamente se copiarán al disco duro las aplicaciones,

Prioridad	Aplicación
1	Brazo
2	Espacio
3	Entrena
4	Biosad
5	Captura

Cuadro 5: Nivel de prioridad de las distintas aplicaciones.

el código fuente, documentación adicional y la memoria misma del proyecto. Para comenzar cualquier programa, basta con realizar doble clic sobre el icono que lo representa, como una aplicación cualquiera.

Tan sólo hay que tomar alguna precaución si se desean ejecutar dos programas simultáneamente. En este caso, hay un orden de prioridades que debe ser cumplido, y el orden en que se lancen los programas ha de atender a un orden predeterminado o de lo contrario no funcionará del todo satisfactoriamente, pudiendo llegar a convertirse el sistema en inestable. Se ha mantenido la posibilidad de ejecutarlos fuera de orden porque ello presenta información útil si se sabe exactamente lo que está sucediendo.

Pero en principio, si se desean ejecutar dos aplicaciones simultáneamente, ha de lanzarse primero la de mayor prioridad, según la tabla 5.

8.3.1. Menú de opciones

Según se describió en la arquitectura del sistema, una capa común alimentaba con la señal adquirida a cualquier aplicación. Dicha capa común presenta una serie de opciones al usuario antes de iniciar cada aplicación (`biomedida`, esta capa inferior toma la forma de una DLL, pero eso no es de nuestro interés en el manual de usuario). El menú que se presenta muestra una presentación en la forma de pestañas, cada pestaña agrupa las opciones de cada sección del diagrama de bloques; dicha forma de menús no debería ser extraña al usuario familiarizado con Windows. En general, en todos los programas se ha buscado deliberadamente que el Interfaz Gráfico de Usuario (GUI) estuviera perfectamente integrado en el estilo Windows.

La primera pestaña muestra tan sólo la presentación y el nombre del sujeto del cual se tomarán los datos. El interés de introducir su nombre está tan sólo a la hora de almacenar los datos; en todos ellos figurará el nombre o el comentario introducido en este menú.

Figura 34: Menú de opciones. Primera pestaña.

La segunda pestaña muestra las opciones sobre el primer bloque, la cabecera analógica P4. Estas opciones son 4.

- Ganancia extra 30dB. Introduce una ganancia extra de 30dB.
- Filtro paso bajo de 40 Hz. Para señales con su energía centrada en frecuencias muy bajas.
- Filtro de red. Elimina el ruido de red en los 50Hz.
- Filtro conmutado. Filtro paso bajo conmutado en el que la frecuencia de corte se decide suministrando una señal a una determinada frecuencia. La tarjeta conversora A/D de CIODAS es capaz de suministrar tal señal. Sin embargo esta opción ha sido deshabilitada hasta posteriores reformas.

La tercera pestaña muestra las opciones de la fuente de datos, el conversor analógico digital. La fuente de datos de la aplicación puede ser el conversor A/D de Lorenzo, el comercial CIODAS, datos almacenados en disco y datos proporcionados por otra aplicación maestra. Además, en este apartado se puede especificar la frecuencia de muestreo. Por defecto es de 1024Hz, el máximo valor permitido para este campo es de 4096Hz; la introducción de datos superiores arrojaría resultados impredecibles.

Figura 35: Menú de cada aplicación. Segunda pestaña.

En el caso de que se seleccione el conversor de Lorenzo puede especificarse además el filtro paso bajo que se aplica inmediatamente antes del muestreo para evitar el solapamiento de la señal; si se marca la casilla la frecuencia de corte será de 1024Hz, si no se marca será tan sólo de 512Hz. Muestreando a 1024Hz, la frecuencia máxima que se permite pasar es de 512Hz, y la casilla debería ser desactivada si usa este sistema de adquisición.

Figura 36: Menú de opciones. Tercera pestaña.

La cuarta pestaña muestra las opciones de procesado digital básico. Estas opciones son un filtrado que elimine la componente DC de la señal, un filtro de medianas con el objeto de eliminar los picos espúreos, y un filtro digital FIR paso alto de orden 32. También se puede especificar una ganan-

cia digital, es decir, una vez digitalizada la señal realizar un escalado. Con ello evidentemente se pierde resolución pero es útil para ajustar las amplitudes manualmente según requieran las aplicaciones que tomen los datos. Si está activada la casilla 'auto' entonces el escalado será automático. En ese caso debería haber sido ejecutada la aplicación de calibración, `entrena` que valora el nivel de ruido y la amplitud de la señal.

Figura 37: Menú de opciones. Cuarta pestaña.

8.3.2. Biosad

`biosad` Es un programa que muestra una representación gráfica de la señal de manera continua. En la versión actual se muestran los cuatro canales indefectiblemente. Una imagen del programa se muestra en la figura 40, mostrando una contracción del bíceps por el canal 1.

Para comenzar la adquisición han de activarse los dos botones que se muestran en las figuras 39, en el orden en que figuran.

Para ajustar la base de tiempos y la amplitud basta con utilizar las siguientes flechas. El resto de botones han perdido su función.

8.3.3. Brazo

`brazo` es la representación gráfica de la prótesis, la aplicación se denomina *Brazo Mioeléctrico Virtual*. Para iniciar la toma de datos, hay que activar en el menú de archivo, la orden 'opciones'. Desde ese momento, el control

Figura 38: Imagen de Biosad

Figura 39: Cómo comenzar la adquisición.

del brazo es realizado por la señal mioeléctrica. Por defecto se ofrecen dos grados de libertad, asociados a los movimientos "cierre de mano", "apertura de mano", "extensión del codo", "flexión del codo".

Si se oprime el botón marcado con una 'F' (figura 41) se conmuta de 5 a 7 estados, y entonces se pueden realizar dos movimiento más: "pronación de la muñeca" y "supinación de la muñeca". Oprimiendo nuevamente el mismo botón se conmuta de 7 a 5 estados.

El programa ofrece numerosas posibilidades de presentación de la imagen (desde el color del fondo a la renderización en modo puntos, líneas o superficies). Esto no ha sido modificado desde que fuera programado por A. Hernández [69], por lo que se remite al lector a la revisión de su proyecto para más detalles; en todo caso su uso es trivial. En la figura 42 se muestra una representación del brazo.

8.3.4. Espacio

`espacio` Tiene dos usos. En primer lugar sirve para relacionar las posiciones del usuario que le son cómodas con los estados de la prótesis. Siendo una herramienta visual, su funcionamiento mejora notablemente. En segundo lugar sirve como demostración sencilla de que es posible diseñar multitud de herramientas de control y sin demasiado esfuerzo. De este modo se presenta

Figura 40: Cómo ajustar la base de tiempos y la amplitud.

Figura 41: Botón que conmuta entre 5 y 7 estados.

un teclado virtual; útil por ejemplo para un amputado de ambos brazos.

La aplicación principal, que es la determinación de los estados, tiene el aspecto que se muestra en la figura 43. Se muestra un cuadrado de fondo blanco, siete círculos de colores y un punto grueso negro.

El punto grueso negro o 'mosca' se mueve con la señal mioeléctrica adquirida de los dos canales bíceps y tríceps. El origen de coordenadas se sitúa en la esquina inferior derecha, de modo que en un estado de reposo, la mosca se colocará en dicha posición. Un esfuerzo del bíceps implicará un desplazamiento de la mosca hacia la izquierda, un esfuerzo del tríceps implicará el desplazamiento de la mosca hacia arriba. Ambas coordenadas no son totalmente independientes en la voluntad del usuario, pero sí se goza de cierta libertad.

El usuario ha de hallar seis posiciones, además del reposo, en las que se encuentre particularmente cómodo y no haya de realizar un esfuerzo excesivo. A cada una de estas posiciones se asignará la posición de un círculo, de modo que los estados quedan así definidos. Cada estado viene representado por un color distinto.

Los círculos son meros elementos de información al usuario sobre dónde está el estado definido, pero su radio no implica ninguna información. En cada momento, la posición de la mosca llevará asociada un estado determinado, aquel al que se encuentre más cerca. El criterio de proximidad es el de distancia euclídea, y en todo momento se muestra la decisión tomada en la esquina inferior derecha en la forma de un cuadrado que toma el color del círculo elegido.

Existe una casilla de verificación que permite conmutar entre un sistema de cinco y uno de siete estados; menos estados implican menos libertad de movimientos para la prótesis pero también más fiabilidad en las decisiones.

La opción del suavizado, con su correspondiente casilla de verificación

Figura 42: Brazo Mioeléctrico Virtual.

sirve para suavizar el brusco movimiento de la mosca, que siempre es susceptible a picos espúreos etc. La mosca tiene así un movimiento mucho más suave y sin saltos bruscos, pero a cambio se ha perdido velocidad de respuesta, y en llevar la mosca de uno a otro extremo del cuadrado requerirá más tiempo.

Inicialmente se muestran los estados que se definieron la última vez que se utilizó el programa. Para modificar uno de los estados, hay que oprimir el botón de 'Cambiar'. A continuación se preguntará qué estado se desea cambiar, se debe introducir un número de 0 a 6; representando cada número un estado, ya se indica a qué número corresponde cada color. Los siete estados se almacenan en otros siete archivos distintos que toman el nombre de `vector0`, `vector1` etc., lo cual se ha hecho deliberadamente así. Esto permite almacenar una determinada configuración en otro directorio distinto, y cuando se desee recuperar la vieja configuración, basta con ubicar los vectores almacenados en el directorio del ejecutable.

Un ejemplo práctico de cómo aprovechar la señal de control se muestra oprimiendo el botón 'Relativo', lo cual conduce a mostrar la pantalla de la figura 44. Se muestra un teclado de 21 letras, en el que del alfabeto español se han omitido las siguientes letras: 'K', la 'Ñ', la 'V', la 'W', la 'X' y la 'Y'. Se han añadido dos botones de tamaño mayor que figuran con un 'NO' y con un 'SI'.

La mosca se desplaza ahora por el teclado no como antes, de una manera absoluta, sino de una manera relativa. El reposo no significa que la mosca cae al origen inferior derecho como antes, sino que el reposo significa tan sólo mantener el punto en la posición en la que estaba. Los proximidad de la señal a los cuatro estados definidos antes, ahora son el criterio para decidir una

Figura 43: Espacio. Utilidad para fijar estados.

de las cuatro operaciones posibles: 'mover mosca a la derecha', hacerlo a la 'izquierda', 'arriba' o 'abajo'. Así, la mosca se mueve con total libertad por el tablero.

Un doble sobreesfuerzo del bíceps debería ser interpretado como el doble click de un ratón, y de este modo, se seleccionan las letras. Las letras que se van seleccionando conforman un texto en el cuadro de texto que figura en el lateral derecho de la ventana. En caso de error, una selección del botón 'No' limpiará la última tecla escrita, en el caso de seleccionar 'Si', se elimina todo el texto.

En todo caso, el objetivo de esta aplicación no era tanto hacer un programa útil como demostrar que era sencillo. Este programa tal cual, que no tiene demasiada utilidad, podría haberse utilizado para un amputado de ambos brazos mandara un mensaje SMS a un móvil (lo cual ya es útil en sí) o para que Windows considerara como teclado lógico a este teclado mioeléctrico.

8.3.5. Entrena

El objeto de este programa es estimar el nivel de ruido. Se solicita al usuario que permanezca en reposo, con la intención de recoger la muestra y hacer una estimación del ruido.

Figura 44: Espacio. Ejemplo de aplicación: teclado.

Figura 45: El entrenador realiza un calibrado.

También se le pide que realice el máximo esfuerzo a fin de poder establecer el rango en el cual se moverá el usuario. Este rango ciertamente no será real, puesto que se procurará hacer trabajar al usuario por debajo del 50 % MVE. Por otra parte, conocidos los nieveles de ruido, se desestimará la utilización de puntos que definan estados en la prótesis si no superan claramente el umbral de ruido. Adicionalmente se podrían realizar más cálculos sobre estos límites, como por ejemplo la correlación o las bandas de frecuencia del ruido (con miras a diseñar un filtro blanqueador, por ejemplo).

8.4. Elección de los parámetros óptimos.

Cómo se ha ido viendo, hay muchas posibilidades de configuración en el sistema, como la frecuencia de muestreo, los filtros que deben actuar y no, etc. Para elegir la configuración óptima lo mejor sería considerar la combinación

de los parámetros que proporcionara mayor separabilidad entre las clases. Ello es sin embargo un parámetro difícil de medir de manera sistemática, dado que son muchos los parámetros y no es posible tomar datos exhaustivamente. En general, no hay más remedio que guiarse por la intuición o por medidas indirectas, bajo la hipótesis de que la influencia de los distintos parámetros es individual y no conjunta en la separabilidad de la señal.

En este apartado se comentará cada posibilidad de manera estructurada. Se mostrarán imágenes con los resultados de los filtros u opciones en la pantalla, comparándolas con imágenes de la señal sin procesar. En la comparación, no es la misma señal la que se compara, pues en general no se puede obtener la señal simultáneamente procesada y sin procesar, sino que se compara la señal sin procesar recogida en un instante, con la señal procesada recogida unos instantes después; no se muestran los resultados del procesado sobre una misma señal, sino sobre realizaciones de un mismo proceso estocástico. Supuesta cierta estacionariedad en la señal, no debería ser un impedimento serio al menos para obtener un análisis cualitativo. Para ver los efectos en realidad el procedimiento adecuado sería tomar muchas muestras de la señal sin procesar y muchas muestras de la señal procesada, y hacer un promedio. En la práctica se ha comprobado que muestras recogidas en instantes de tiempos muy diferentes sí mantienen las propiedades básicas y por tanto basta con considerar una única muestra. Los resultados serán válidos de manera cualitativa pero no serán estrictamente válidos considerados de manera cuantitativa (y aun así, la idea que dan es bastante aproximada).

Para justificar este procedimiento, se muestra un ejemplo experimental que avala este razonamiento, donde se muestra que la señal no varía dramáticamente sus características de un momento a otro. Se tomó la señal de un músculo en reposo con todas las opciones de la cabecera deshabilitadas y se extrajo un segmento aleatorio. Otro día se repitió la operación, dejando totalmente al azar la elección del segmento a ser analizado. La densidad espectral de potencia de uno y de otro día se muestran en la figura 6.

La similitud entre los espectros es asombrosa, e indica que parece razonable asumir cierta continuidad ante el cambio de condiciones.

8.4.1. Opciones de la cabecera analógica P4.

Filtro de red 50Hz. Las señales ruidosas cuyos espectros se mostraban en la figura 6 son sencillas de interpretar. Como era de esperar, se observa una componente muy fuerte de ruido en los 50Hz, el zumbido de red. Lo

Cuadro 6: El espectro de ruido se mantiene constante en días distintos.

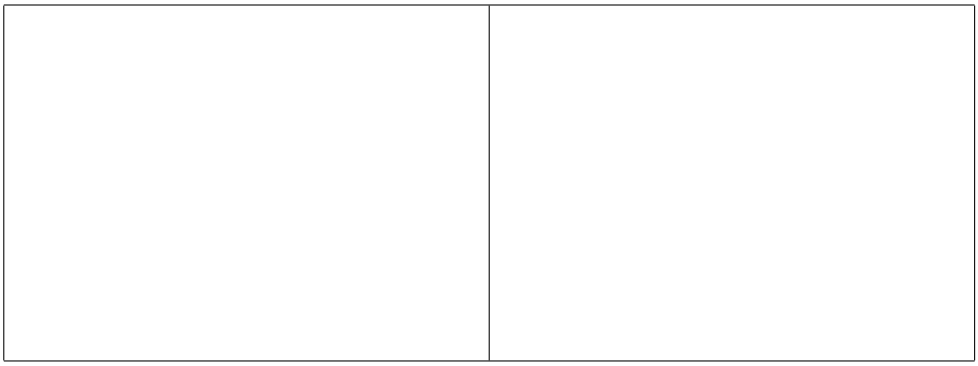
que ya es más nuevo es que aparecen asimismo frecuencias de ruido en los 100Hz, y en menor medida en los 75Hz. En uno de los patrones incluso se aprecia una componente fuerte de ruido en los 150Hz. Es evidente que las señales de 75, 100 y 150 Hz están relacionadas de algún modo con la de 50Hz (son para empezar armónicos sencillos de la señal de red, corresondientes a las frecuencias de 3/2, 2, 3. etc.), todo ello apunta a que en algún sitio se mezclan las señales y atraviesan una no linealidad.

Podría pensarse que los picos de ruido sólo eran significativos en ausencia de señal EMG, y que la señal EMG sería de potencia suficiente como para ahogar por completo al ruido, pero en la figura 46 se muestra el espectro de un patrón de señal EMG de un músculo contraído, y sin embargo dichos espúreos se siguen apreciando. La adquisición se llevó a cabo deshabilitados todos los filtros y opciones, y se muestra, por vez única en toda la memoria, al espectro completo, sin omitir la parte simétrica, desde los -512 hasta los 512Hz.

Figura 46: Espectro de un patrón.

Ramón de la Rosa diseñó en la cabecera analógica P4 un filtro elimina

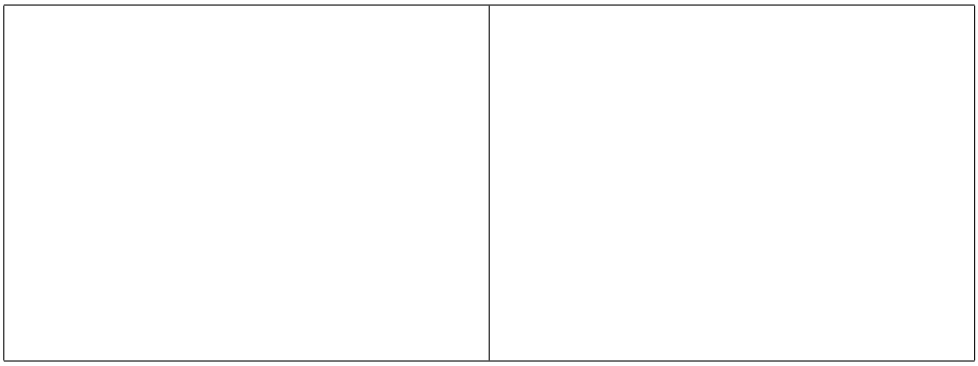
banda que mitigara el ruido de los 50Hz. Se comprobó experimentalmente el efecto de aplicar o no el filtro a la señal, observándose los resultados en la figura 7.



Cuadro 7: Efecto de aplicar el filtro de red de 50 Hz sobre el espectro de la señal.

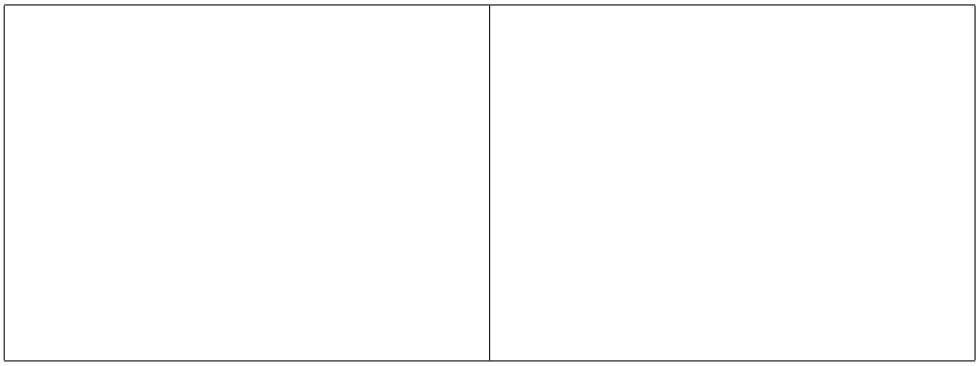
Se deduce al instante que el uso del filtro de red de 50 Hz es muy recomendable.

Filtro paso bajo de 40Hz. El segundo de los filtros de la cabecera P4 es un filtro paso bajo de 40Hz. Nuevamente se procedió a tomar muestras de la señal estando deshabilitada la opción y estando habilitada a fin de comparar los resultados. El resultado de esta opción se muestra en la figura 8.



Cuadro 8: Efecto de aplicar el filtro paso bajo de 40 Hz sobre el espectro de la señal.

Ganancia +30dB En este caso ambas señales tenían aplicado el filtro de 50Hz para evitar el molesto pico que enturbiaba en demasía la gráfica. El resultado de esta opción se muestra en la figura 9.



Cuadro 9: Efecto de aplicar la ganancia extra sobre el espectro de la señal.

8.4.2. Conversión A/D

Aquí el parámetro reseñable es la frecuencia de muestreo. La frecuencia de muestreo ha de ser suficiente como para adquirir la señal EMG sin perder parte significativa de la misma. Habitualmente se ha tomado 1kHz como frecuencia de muestreo y eso es lo que se ha hecho en este laboratorio (en concreto 1024Hz). Así, según impone el teorema de Nyquist, alcanceremos a adquirir las frecuencias de la señal de hasta 512 Hz.

Como justificación de nuestra decisión se muestreó la señal EMG de un músculo en tensión a unos 3kHz (3072 muestras por segundos, en concreto) y se visualizó la densidad espectral de potencia de la señal. El resultado se muestra en la figura 47. La señal había recibido también los filtros digitales y el filtro de red, así como la ganancia extra. Se realizaron cálculos sobre estos datos, y se llegó a la conclusión de que el 95.2 % de la energía está entre los 0 y los 512Hz, quedando sólo un 4.8 % en el resto, que va de los 512 a los 1536 Hz. Otros resultados publicados [38] afirman que el 99 % de la energía queda bajo los 350 Hz, aunque los resultados varían evidentemente con la separación de los electrodos y otros factores.

En todo caso, otros electromiógrafos comerciales también proponen el muestreo a 1kHz. Todos los electromiógrafos vienen acompañados de un software de análisis similar al desarrollado aquí, y puede apreciarse cómo el espectro de una señal EMG mostrado por ellos[42] (figura 48) no difiere realmente mucho, con lo cual parece que lo desarrollado en este laboratorio es en principio fiable.

[42]Durante la elaboración del proyecto se dispuso del software de análisis de electromiogramas EMGWorks de la empresa Delsys en su versión de prueba.

Figura 47: No hay componentes frecuenciales importantes más allá de los 500Hz.

Figura 48: Espectro analizado por otro electromiógrafo.

8.4.3. Filtros digitales.

La programación de filtros digitales mediante software es sencilla; su tiempo de desarrollo es menor que la de un filtro hardware y su modificación menos costosa. Se han implementado tres filtros, un filtro que elimina la componente DC de la señal, un filtro que elimina el ruido impulsivo y un filtro FIR paso alto.

Filtro DC. En la figura 49 se muestra una imagen de `biosad`. En ella se muestra en el canal 4 a la señal adquirida del canal 1, tal cual está, sin recibir ningún procesado digital. En el canal 1 se muestra la señal adquirida por el canal 1 tras haberse sometido a un filtro digital de medianas y a un filtro de DC.

Se han marcado tres puntos en los que el efecto del filtro es evidente. En el de la izquierda y en el de la derecha, se aprecia cómo el filtro de DC impide a la señal fluctuaciones duraderas en la media. De no estar el filtro, habría varios segmentos (se aprecia muy bien en el tramo de la izquierda) con la media mucho más alta, lo cual es sólo debido al movimiento del cable o algo similar, pero que en todo caso no corresponde a la señal EMG. El filtro ha actuado correctamente.

Filtro que elimina el ruido impulsivo. Es básicamente un filtro de medianas ligeramente modificado. Hay que decir que este filtro no es lineal. El filtro de medianas toma muestras de 7 en 7, y a la muestra central la asigna el valor mediano de entre los siete valores; tal filtro es aplicado a cada muestra. Tiene la propiedad de que elimina los ruidos impulsivos de manera eficaz, es decir, elimina aquellas muestras que son anormalmente altas. El filtro es rápido actuando y hace lo que haría un filtro paso bajo pero sin dañar tan seriamente las componentes espectrales de alta frecuencia. La modificación que se ha comentado consiste en impedir que una muestra sea mucho más alta que la que la precede, de manera que el valor abosoluta de una muestra y la siguiente no pueden guardar una relación mayor de 1 a 10; esta añagaza sirve para reforzar la lucha contra las muestras extrañas.

En el punto señalado en el medio, en la figura 49 se aprecia cómo ha actuado el filtro de medianas. El filtro de medianas elimina picos como el señalado, si bien no siempre es efectivo. Cuando hay varias muestras de ruido impulsivo consecutivas no es capaz de eliminarlo; el filtro de medias es sólo de orden 7.

Figura 49: Efecto de los filtros de DC y de medianas.

Coef.	Valor	Coef.	Valor
b_0=b_{32}	-0.0013	b_8=b_{24}	-0.0104
b_1=b_{31}	-0.0015	b_9=b_{23}	-0.0122
b_2=b_{30}	-0.0020	b_{10}=b_{22}	-0.0140
b_3=b_{29}	-0.0028	b_{11}=b_{21}	-0.0157
b_4=b_{28}	-0.0039	b_{12}=b_{20}	-0.0171
b_5=b_{27}	-0.0053	b_{13}=b_{19}	-0.0184
b_6=b_{26}	-0.0068	b_{14}=b_{18}	-0.0193
b_7=b_{25}	-0.0085	b_{15}=b_{17}	-0.0198
b_{16}	0.9813		

Cuadro 10: Coeficientes del filtro FIR.

Cuadro 11: Efecto de aplicar el filtro FIR.

Filtro paso alto La cabecera analógica P4 ofrece un filtro paso alto de frecuencia de corte 3Hz que en ocasiones puede ser escaso. Si se desea un filtro paso alto con frecuencia de corte algo superior, ha de utilizarse este filtro digital. Es un sencillo filtro FIR. La respuesta en frecuencia del filtro se ve en la figura 50, los coeficientes se ofrecen en la tabla 10 y se comprueba el funcionamiento del filtro en una situación real en la figura 11.

El filtro FIR tiene una frecuencia de corte en 0.02 respecto a 1. Es decir, que tomando un muestreo a 1024 muestras por segundo, como se ha propuesto, significa que la frecuencia de corte del filtro es el 2

Los coeficientes que se muestran en la tabla 10 fueron calculados con Matlab. El filtro FIR sólo llevó al programa 444 operaciones de punto flotante para calcular los coeficientes[43]. Para calcular los coeficientes de un filtro But-

[43]La variable de Matlab flops indica cuantas operaciones de punto flotante se realizaron; es algo impreciso porque cuenta igual una suma que una raíz cuadrada, pero da una idea

Figura 50: Respuesta en frecuencia del filtro FIR.

terworth del mismo orden, necesitó 1297498 operaciones de punto flotante requiriendo de varios segundos de proceso de la máquina. Esto es importante. Si en una futura mejora del programa se flexibilizaran los filtros y estos pudieran ser cambiados en tiempo real, habría que recalcular rápidamente los coeficientes, y en este punto a la vista de estos resultados hay que comentar que un filtro sencillo es viable, pero uno más complejo como Butterworth o elíptico ha de ser descartado.

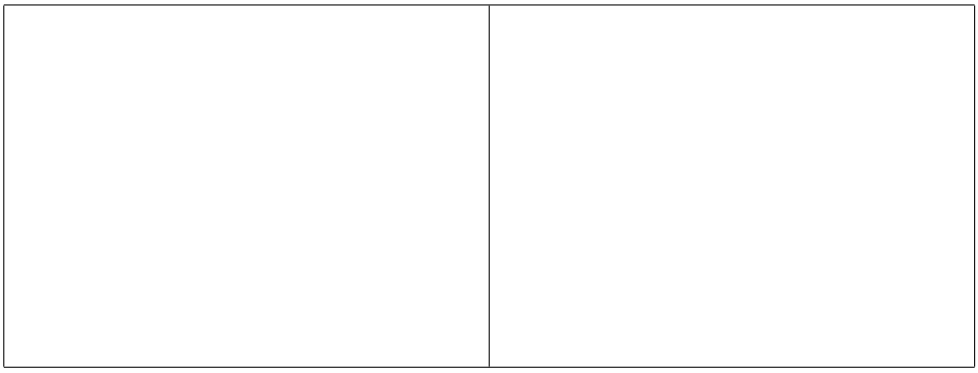
8.4.4. Peso del ruido de red.

En esta subsección se muestra una característica importante que podría ser utilizado por el software como discriminatorio de si la señal de entrada corresponde a una señal EMG o corresponde tan sólo a ruido.

En ausencia de señal EMG, el ruido de red es la única fuente importante de señal, y su importancia relativa en la señal medida es enorme. En cambio, habiendo señal EMG, el ruido de red sigue estando presente, sí, pero su importancia relativa es menor. En la figura 12 se muestra una señal adquirida en reposo (sin ningún filtro activo en la cabecera) y una señal adquirida en esfuerzo medio (idéntica configuración). Se puede apreciar que el peso de la componente de 50 Hz queda rebajado, así como el peso de la de 100Hz.

Este hecho puede ser aprovechado para determinar fácilmente si una señal corresponde a señal EMG o a ruido, basta con tener en cuenta la importancia relativa del ruido. Se puede definir esa SNR (considerando como 'señal' al ruido de red de 50 Hz) como la densidad espectral de potencia a 50Hz partido por la densidad espectral de potencia media excepto en dicho punto. En la

del orden de magnitud de la complejidad del problema.



Cuadro 12: Importancia relativa del ruido de red sin filtro de red.

DFT de 205 puntos de una señal muestreada a 1024Hz, no está disponible la señal a 50Hz, sino a 47.5 Hz.

Se realizó en Matlab tomando el peso de la muestra número 93 de 205 del espectro de potencia (o la 113), partido por la media de la señal no considerando dicho punto. Los resultados son que, en la figura de la izquierda, en ausencia de señal EMG, esta SNR entre el ruido de red y el ruido 'blanco' toma el valor de 55 (17 dB), en tanto que en presencia de señal, la SNR toma el valor de 8 (9dB). También se podía haber tomado como criterio el pico de los 100Hz, que también es apreciablemente hegemónico en la señal ruidosa.

Si se activa el filtro de red este criterio para diferenciar entre lo que es señal y lo que es ruido, el criterio ya no es tan válido. Sin embargo, el filtro de red no elimina el armónico de 100Hz. En la figura 13 se aprecia cómo el ruido de 50Hz ha sido filtrado por la cabecera P4, pero no el armónico de 100Hz. Puede utilizarse el armónico de 100Hz para distinguir si la señal adquirida corresponde a la señal EMG o no, aunque el grado de confianza es menor.

Para ellos se procedió igual que antes (considerando la densidad espectral de potencia a la frecuencia de 97.45 Hz de la muestra 83 o 123. Los resultados en esta ocasión son de SNR=20 para el caso de ruido, y de SNR=5.8 para el caso de señal EMG. Un resumen de todas estas SNR se muestra en la tabla 14

8.5. Manual de referencia.

Como comentar exhaustivamente los programas sería algo excesivamente pesado para el lector y de poca aportación, se ha seguido la estrategia en

Cuadro 13: Importancia relativa del ruido de red con filtro de red.

Filtro de red	NO	NO	SI	SI
Señal EMG	NO	SI	NO	SI
SNR (50Hz)	55	8	1	1
SNR (100Hz)	9	8	20	6

Cuadro 14: SNR entre ruido de red y resto de la señal.

este apartado de resaltar sólo aquellos aspectos que son particularmente significativos.

En esta sección se detallan algunos aspectos particulares de cada programa. Se destaca el método por el cual se comunican los programas, y también se hace referencia a los aspectos concretos de la adquisición de los datos según qué origen.

8.5.1. Sobre la compilación y ejecución

Todo el software ha sido desarrollado en el entorno Visual C++. Se retomaron dos proyectos ya existentes (`biosad` y `brazo`) y se crearon cuatro proyectos más (`biomedida`, `vroddon`, `espacio`, `entrena`). Adicionalmente se creó algún otro programa, especialmente para procesar y analizar datos particulares, pero eso no ha sido incluido en la memoria ni en el CDROM. Los seis proyectos (*projects*) fueron reunidos en un solo espacio de trabajo (*workspace*) y esto es lo que se ofrece en el CDROM.

Para compilar cada proyecto basta con abrir en el 'Developer' el espacio de trabajo mencionado y buscar la opción compilar. La estructura de directorios que haya creado el programa de instalación es la correcta, pero en todo caso hay que tener en cuenta que hacen falta ciertas bibliotecas para que la

	Bibliotecas necesarias	Ejecutable
biomedida4	cbw32.lib	biomedida4.dll
vroddon		vroddon4.dll
espacio	biomedida4.lib vroddon.lib	espacio.exe
entrena	biomedida4.lib vroddon.lib	entrena.exe

Cuadro 15: Bibliotecas necesarias y ejecutable resultante.

compilación sea exitosa y hacen falta todos los archivos de código fuente que se enumeran en la subsección 8.5.10.

En la tabla 15 se muestran las bibliotecas necesarias para cada ejecutable. Las bibliotecas estáticas `.lib` han de estar presentes en el momento de compilación, en tanto que las bibliotecas dinámicas DLL sólo es necesario que estén presentes en el tiempo de ejecución. Al compilar, no sólo hará falta el archivo `.lib`, sino también el archivo de cabecera `.h` correspondiente en el cual estén declaradas las funciones que se vayan a invocar.

La biblioteca estática `cbw32.lib` es la ofrecida por el fabricante de la tarjeta CIODAS para que se puedan compilar las funciones en C que ofrece. Las bibliotecas estáticas han de estar en el directorio del código fuente, pero las bibliotecas de enlace dinámico, las que han de estar presentes en tiempo de ejecución, deben estar en el mismo directorio que el ejecutable, o si no, en el directorio `C:\WINDOWS\SYSTEM`.

Si se ejecuta el programa de instalación del CDROM se instalará todo lo necesario para que los programas puedan compilar y no haya que preocuparse; en todo caso ya se ofrece una versión compilada de las DLL y de las aplicaciones.

En el caso de que sólo deseen instalarse las aplicaciones, efectivamente se copiarán los archivos `.exe` y los `.dll` que se muestran en la tabla 15, pero eso *no* garantiza que vayan a funcionar correctamente. En casi todos los proyectos se ha utilizado la biblioteca de funciones MFC, y al realizarse la compilación se ha elegido la opción que dice que las clases MFC permanezcan implementadas en su DLL independiente, por lo que esta DLL ha de estar disponible. El archivo `MFC42D.DLL` ha de estar en el ordenador en el que se copien los ejecutables, y si no está los programas que aquí se presentan *no* funcionarán.

El archivo MFC42D.DLL figura en todos los ordenadores que tengan instalado el Visual C++, pero también en todos los ordenadores que tengan instalado algún programa que haga uso de él, que son bastantes. Pero un ordenador que cumpla uno u otro requisito no tendrá la biblioteca y los programas no funcionarán. Para solucionarlo, se puede copiar este archivo directamente desde un ordenador que sí lo tenga o se puede recompilar todo usando la opción que no requiere la presencia de esta biblioteca. Al compilar se ofrecen dos posibilidades para incluir las clases de esta biblioteca: se pueden dejar en su DLL tal y cómo están y se pueden incluir en el ejecutable que sea compilado; esta segunda opción, que no se ha elegido, genera ejecutables mucho más voluminosos si bien no requiere preocuparse por la presencia del MFC42D.DLL. Es una política contraria a la filosofía de Windows no reutilizar código, y así se justifica la decisión de no incluir las clases en los ejecutables que se ofrecen.

Además de la MFC42D.DLL deberían estar presentes las bibliotecas MSVCRTD.DLL, KERNEL32.DLL y USER32.DLL, pero esto no es ningún problema porque son parte del núcleo del sistema operativo y están en todo ordenador con Windows. Para conocer las DLL que son necesarias para ejecutar cada programa puede usarse la herramienta 'Dependency Walker'.

8.5.2. Biomedida

Esta aplicación es una DLL que ofrece los servicios básicos de adquisición, como muchas veces se ha comentado en esta memoria. Las funciones que exporta se declaran de la siguiente manera:

```
extern "C" __declspec(dllexport) int Recoger (struct Datos *lote);
extern "C" __declspec(dllexport) void Acabar ();
extern "C" __declspec(dllexport) void Configurar(struct Datos *lote);
extern "C" __declspec(dllexport) void Grabar(struct Datos *lote);
```

y cuando otra aplicación desee hacer uso de una de estas funciones, ha de incluir las lineas:

```
extern "C" __declspec(dllimport) int Recoger (struct Datos *lote);
extern "C" __declspec(dllimport) void Acabar ();
extern "C" __declspec(dllimport) void Configurar(struct Datos *lote);
extern "C" __declspec(dllimport) void Grabar(struct Datos *lote);
```

Estas cuatro funciones son precisamente el núcleo de todo. Únicamente estas cuatro servicios se ofrecen a las aplicaciones de capa superior de modo que el programador pueda olvidarse de todo. `Configurar` ha de ejecutarse para preparar la adquisición y `Acabar` para finalizarla. `Recoger` es invocada en cualquier momento para recoger los datos y `Grabar` si se desean grabar datos. Los datos se representa como un objeto lógico encapsulado en una estructura que es la más importante de todo el proyecto, `struct Datos`.

```
struct Datos
{
unsigned short data[BUFTAM];  //datos(d0,d1,d2,d3,d0,d1,d2,d3,d0...)
signed short data0[BUFTAM];   //datos del canal 0.
signed short data1[BUFTAM];   //datos del canal 1.
signed short data2[BUFTAM];   //datos del canal 2.
signed short data3[BUFTAM];   //datos del canal 3.
long validos;                 //numero de muestras (contando todos los canales)
long inicio;                  //inicio a partir del cual tienen sentido las muestras.
char nombre[BUFSTR];          //nombre del sujeto
char nomfile[BUFSTR];         //nombre del archivo(+nº de secuencia)
BOOL huboerror;               //indica si hubo algun error
BOOL ganancia;                //ganancia hardware de 30dB.
BOOL fred;                    //filtro notch de 50Hz de la cabecera
BOOL fpalto;                  //prefiltro paso bajo de 40hz,
BOOL ciodas;                  //si FALSE, es que es lorenzo's.
BOOL autonomo;                //adquisicion autonoma
BOOL lorenzo;
BOOL fcorte;                  //true si frecuencia de corte alta para lorenzo's
BOOL fmedianas;               //filtro de medianas
BOOL fsoftalto;               //filtro paso alto software
BOOL fDC;                     //filtro DC
BOOL gananciassoft;
int canales;                  //numero de canales adquiridos.
                              //canales=0:ch0, canales=1:ch1 ... canales=4: ch0,ch1,ch2,ch3
long milisegundos;            //milisegundos que abarca el intervalo
float ganancias[4];           //ganancias software
int secuencia;                //es el numero de secuencia.
long fmuestreo;               //frecuencia de muestreo (por defecto 1024).
CTime hora;                   //indica la hora actual.
float flotante;               //propositos varios...
};
```

Esta estructura sirve para almacenar los datos de un patrón y todos lo parámetros que lo configuren, tal como su duración en milisegundos, número de muestras etc.

`biomedida` además tiene una docena de funciones que no se exportan pero que utiliza para realizar sus cálculos, como son los filtros digitales de medianas, FIR y DC. Hay más funciones y clases, pero son las propias de mostrar el interfaz gráfico; no olvidemos que `biomedida` ofrece un menú al invocarse la función de configuración. Estas funciones ya no tienen tanto interés.

Baste conocer los detalles técnicos básicos. El menú se presenta mediante un cuadro de diálogo que muestra un control 'de etiqueta' (*control tab*) con varias pestañas; y cada pestaña es representada por una clase; el control genera eventos cuando se cambia de pestaña y al cambiar de pestaña se invoca a la función de dibujo de la clase correspondiente para que plasme su ventana justo encima del control. Cada pestaña es pues una ventana distinta de tipo *child* que depende de la ventana padre del control, y no tiene ni borde ni barra de títulos. Este es el estilo de programación más habitual para programar los controles 'de etiqueta'.

Un diseño altenativo de `biomedida` podría haber sido programarla como un objeto COM. También tomaría la forma de una DLL pero de una forma más general; el objeto se tendría que registrar en el registro de Windows y ofrecería en teoría más flexibilidad si se deseara acceder a sus funciones desde otras aplicaciones programadas en otros lenguajes distintos del C. Sin embargo, y dado que no aporta mucho más que un estilo de programación algo más elegante, la complejidad del modelo no aconseja su uso para una aplicación más sencilla como es ésta.

8.5.3. Vroddon

Es la DLL que ofrece servicios del análisis de los datos. Para exportar las funciones es preciso incluir las líneas de manera análoga a como se ha mostrado antes. No tiene nada especial que sea digno de ser comentado, excepto un montón de líneas de código realizando cálculos sobre la señal.

Ofrece 37 funciones de las cuales 11 son exportadas y pueden ser invocadas por otros programas. Algunas de estas funciones están sobrecargadas, y así presentan una implementación en la que operan sobre datos enteros y otra sobre datos de punto flotante.

La función `Decisor` es la que toma la decisión, es el clasificador. Inicialmente operaba sobre la base de un vector de características que incluía los 9 grupos de un histograma sobre la amplitud de las muestras, la varianza, la integral del valor absoluto y los cruces con cero. En el periodo de entrenamiento se fijaban los 5 o los 7 estados, y cada estado venía definido por un vector con estos elementos. A tal efecto se programó una estructura llamada `struct Caracteristicas`. Los vectores representantes de cada clase se creaban como sencilla media aritmética de todas las características extraídas de cada uno de los patrones de entrenamiento correspondiente a esa clase. El rendimiento con esta configuración no era bueno. Todas las características

eran normalizadas entre 0 y 1, adecuando su valor máximo al valor de la característica ante un MVE y su valor mínimo ante el valor de reposo.

La siguiente versión redujo el conjunto de características a sólo 2, el número de cruces con cero y la varianza, tal y como hiciera Saridis [15][17]. El rendimiento era apreciablemente mejor que el anterior. La razón por la que el modelo anterior fracasó estrepitosamente es que, según se ha expuesto en la parte teórica de esta memoria, ignoraba las matrices de covarianza y por tanto hacía suposiciones radicalmente erróneas. A pesar de introducir la normalización, que era fundamental, no se tenía en cuenta que las variables estaban íntimamente relacionadas y no eran en ningún caso independientes.

De este modelo se pasó al modelo final, en el que el único parámetro tomado en consideración es el valor central del histograma de 9 grupos, siendo cada grupo de igual tamaño y cubriendo entre todos todo el rango dinámico de entrada de la señal. Se probó con otras características (cruces con cero e integral del valor absoluto) para ver su desempeño funcionando como único criterio, pero el resultado era algo inferior al del valor central del histograma de amplitudes. Los cruces con cero se siguieron utilizando como información mucho más fiable que la información de amplitud para determinar si había reposo o no. Un ruido podía elevar la amplitud de la señal y podrían detectarse estados cuando el usuario lo que estaba era en reposo; pero un ruido no modificaba de manera dramática el número de cruces con cero.

Utilizar los cruces con cero para determinar si una señal de amplitud fuerte es ruido o es señal es una buena idea, pero se ha propuesto como mejora siguiente lo expuesto en el apartado 8.4.4. Y es que el pico de 50Hz se diluye ante una señal EMG más que ante un ruido, por lo que esta opción propuesta podría suponer una mejora significativa. Sin embargo presenta la clara desventaja de que una mejora en la cabecera P4 que eliminara el ruido de red a la perfección anularía este sistema.

8.5.4. Biosad

Fue programado originalmente por Lorenzo Ferrero para probar su conversor analógico digital. Ha sido adaptado para que encaje en la arquitectrua propuesta, de modo que ahora funciona sobre los servicios de `biomedida`. Así por ejemplo como se ha comentado antes, podrá tomar la señal del conversor comercial, o podrá compartir datos con otras aplicaciones, para llevar eso a cabo hubieron de modificarse varias funciones. El programa fue simplificado, por ejemplo ya no se permite la grabación de datos desde `biosad` ya

que en la nueva arquitectura estas atribuciones pertenecen a la capa inferior `biomedida`. Para cambiar las opciones de adquisición, no es posible invocar de nuevo a la pantalla de opciones de `biomedida` y el único remedio, en este y en el resto de las aplicaciones es salir del programa y volver a ejecutarlo.

Además, otras opciones han sido eliminadas por no estar comprobado su funcionamiento. El programa sigue presentando problemas de estabilidad, como ya notificara el propio Lorenzo. Si el programa se ejecuta largo tiempo seguido, es posible que se detenga su ejecución. El resto de las aplicaciones es perfectamente sólido.

8.5.5. Brazo

El original, de Antonio Hernández Bajo, ofrecía tan sólo la interfaz gráfica, accionándose el movimiento con el teclado. Ha sido modificado para utilizar los servicios de adquisición de `biomedida` y los servicios de `vroddon` como decisor. También han sido mutiladas muchas de sus funcionalidades. Las más importantes, son la que sugería una serie de entrenamientos, y la que permitía en todo momento cambiar los parámetros de la adquisición. Puesto que tales asuntos ahora carecían de importancia y presentaban algunos problemas, han sido descartados.

8.5.6. Espacio.

No tiene ninguna dificultad especial entender el código de este programa.

Para calcular dónde ubicar la mosca se siguen los datos de amplitud de la señal. La amplitud de la señal no es una magnitud acotada, y las magnitudes no acotadas podrían salirse del marco establecido. En realidad, se sabe que la señal está limitada en amplitud por el el límite del conversor A/D, pero entonces la sensibilidad sería extraordinariamente pequeña y la mosca no se movería mucho.

Por tanto, lo que se genera un histograma de amplitudes y se toman los valores que se quedan dentro de unos límites como dato origen para la representación. Este procesado sencillo es para la representación, pero para la decisión se pueden tomar los criterios más complejos que se han expuesto antes. En la pantalla de 2 dimensiones no se puede reflejar mucha más información de manera simultánea. El calibrador de ruido 'entrena' fija el umbral para el histograma.

Adicionalmente, se había pensado en incluir el detector de ruido, que detectara si la señal es sólo ruido o hay actividad mioeléctrica; sin embargo se ha dejado finalmente como trabajo futuro y no demasiado complicado.

Cuando en `espacio` se ejecuta la opción de fijar los patrones característicos se graban datos duranto unos segundos, aproximadamente 10, y se van almacenando las características por cada patrón, que finalmente son promediadas arrojando el nuevo vector característico de cada patrón. Se calcula el vector íntegramente, si bien el decisor actualmente implementado sólo hace uso de una de esas características.

Cuando se activa la opción de suavizado se realiza un sencillo filtrado paso bajo. Como manera de suavizar se propuso inicialmente hacer que los patrones se fueran pisando unos a otros y que los bordes de cada patrón, las primeras y las últimas muestras fueran atenuadas, bajo algún tipo de ventana. Sin embargo eso no aportaba un suavizado especialmente efectivo, y finalmente se fijó el suavizado como el resultado del más sencillo filtrado paso bajo: cada característica en un momento dado es promediada con la característica en el momento previo.

8.5.7. Compartición de datos entre aplicaciones

La obtención de la señal ha de realizarla una única aplicación pues de lo contrario habría colisiones y sólo la primera que solicitara los datos a la tarjeta se llevaría el gato al agua, en tanto que los otros programas se quedarían sin datos. Es evidente, por tanto, que no puede haber dos programas simultáneamente accediendo a los puertos.

Sin embargo es deseable que más programas tengan acceso a los datos a la vez. Se planteó como interesante por ejemplo tener una ventanta abierta con el `brazo` y otra con la representación gráfica de la señal. Una de las razones por las que se escogió la arquitectura del programa tal y como está es salvar este problema. Siendo un único módulo el que recoge las señales, `biomedida`, el resto de las aplicaciones deberían acceder a esta bibliotecas de enlace dinámico (DLL) y tener acceso a los mismos datos.

Pero el funcionamiento de las DLL en principio no nos satisface. Cuando dos aplicaciones invocan a una función de una DLL una instancia de ésta es cargada en la memoria, en mapeados distintos para cada aplicación, de modo que el valor de las variables de la DLL es distinto en cada espacio de direcciones. Ello es debido a que se sigue el principio lógico de Windows de

que un proceso no puede tener acceso al espacio de otro proceso bajo ningún concepto, por motivos básicos de estabilidad y de seguridad; es propósito de Windows 95/98 que un puntero equivocado de una aplicación no haga fallar a otra aplicación (cosa que sucedía con relativa frecuencia bajo Windows 3.1)

Puesto que lo que se desea es precisamente lo contrario, que dos aplicaciones distintas tengan acceso a unos mismos datos, se hubo de indicar explícitamente que era eso lo que se quería. Para ello es necesario almacenar las variables a ser compartidas en una *sección distinta*, mediante la directiva al preprocesador:

```
#pragma data_seg("seccion")
int compartido=0;
#pragma data_seg
```

e indicar que los atributos de esta sección de datos son RWS, lectura, escritura y compartición.

```
#pragma comment(linker,"/SECTION:compartido,RWS")
```

Los datos que se incluyan en esa sección han de estar inicializados, pues de lo contrario el compilador los incluiría en otra sección especial para los datos sin inicializar y ya no estarían siendo compartidos.

A la hora de utilizar punteros, por tanto, se tiene un problema serio, pues el valor devuelto por el `malloc` de uno de los procesos no será accesible para los demás, o apuntará a regiones de memoria prohibidas para los otros procesos. Por tanto, habrá que utilizar arreglos inicializados. Es por esto por lo que se ha hecho así, y no se ha compartido la estructura `struct Datos` entera, como sería lógico. Lo que se ha comentado para un arreglo, es también aplicable a una estructura, a una función o a una clase.

8.5.8. Adquisición con la tarjeta CIODAS

La cabecera analógica P4 es conectada a la tarjeta conversora A/D CIO-DAS1602 que está instalada dentro del ordenador. El cable de conexión es de 37 pines, de los cuales son relevantes los que se muestran en la tabla 16.

Opcion	E/S	Canal	#canal
CANAL1	E	CH0	37
CANAL2	E	CH1	36
CANAL3	E	CH1	35
CANAL4	E	CH3	34
TIERRA	-	GND	29
!GANANCIA	S	DO0	23
FILTRO-RANURA50	S	DO1	4
!FILTRO-CONMUTADO	S	DO2	22
!FILTRO-PREVIO40	S	DO3	3
CLOCK	E	CTRO	2

Cuadro 16: Líneas de conexión con la tarjeta CIODAS

8.5.9. Adquisición con la tarjeta Lorenzo

Si se adquiere la señal con la tarjeta Lorenzo, un cable paralelo sirve para llevar los datos de la placa al ordenador. El puerto paralelo ha de ser configurado en modo EPP (Extended Paralel Port) o de lo contrario no será posible la adquisición. Esto es muy importante y no es la configuración por defecto en los ordenadores. No se utilizan interrupciones, de modo que la llegada de datos ha de ser atendida síncronamente.

El módulo `biomedida` abstrae el problema, y en él se incluye el código para recoger datos del puerto paralelo. Los datos se recogen del puerto `LPT1`. Éste queda mapeado en los ordenadores que se han utilizado (y en la mayoría) en el puerto lógico `0x0378`, `0x0379` y `0x037A`, correspondientes al puerto de datos, de estatus y de control respectivamente.

Figura 51: Correspondencia de pines en el puerto paralelo.

8.5.10. Distribución del código.

El código fuente ha sido incluido parcialmente en esta memoria en un apéndice. En la tabla 17, se muestran todos los archivos involucrados en los proyectos del Visual C++, de entre los cuales se han marcado en negrita aquellos archivos cuyo código ha sido parcial o totalmente incluido en el apéndice.

Para la distribución del código se ha grabado un CDROM que incluye los programas ejecutables, el código fuente, la memoria y algunos artículos encontrados en la internet. El CDROM incluye un programa de instalación. El programa de instalación fue creado con 'Install Shield', al igual que la gran mayoría de los programas de instalación con los que se distribuye el software. Así, con una interfaz gráfica amigable común se permite instalar en un ordenador distinto unos u otros componentes.

	Fuente	Cabecera	Recursos
biomedida4	**biomedida4.cpp** tabgeneral.cpp tabd.cpp tab1.cpp abecera.cpp procesado.cpp conversor.cpp stdafx.cpp	**biomedida4.h** tabgeneral.h tabd.h tab1.h abecera.h procesado.h conversor.h stdafx.h cbw.h	biomedida4.rc2 bitmap1.bmp
vroddon	**vroddon.cpp** stdafx.cpp vroddon.def	vroddon.h stdafx.h biomedida4.h resrov.h	vroddon.rc
espacio	**espacio.cpp** espaciodlg.cpp fijarpuntos.cpp stdafx.cpp	espacio.h espaciodlg.h fijarpuntos.h stdafx.h resespa.h vroddon.h biomedida.h	espacio.rc espacio.ico letrasbuenas.bmp espacio.rc2
entrena	**entrena.cpp** **entrenadlg.cpp** **entrena2.cpp** stdafx.cpp	entrena.h entrenadlg.h entrena2.h stdafx.h resentr.h vroddon.h biomedida.h	entrena.rc entrena.ico entrena.rc2

Cuadro 17: Archivos fuente de cada proyecto.

9. Conclusiones y trabajo futuro.

El reconocimiento de patrones mioeléctricos permite reconocer el estado de un músculo y las aplicaciones que de ello se pueden derivar son innúmeras. En este proyecto se ha propuesto un sistema sencillo de adquisición, procesado y reconocimiento de patrones pero se ha esbozado la teoría general en la cual se ha de enmarcar todo desarrollo posterior.

La validez del sistema que se ha propuesto es globalmente satisfactoria pero asimismo susceptible de importantes mejoras que ofrecen un prometedor futuro si se prosigue esta labor. Se apuntan como futuras líneas generales tanto la mejora del sistema extractor de características como la mejora del sistema clasificador, siendo razonablemente sólido el software del sistema. Como líneas particulares, muchos temas concretos que apenas han sido explorados aquí ofrecen buenas perspectivas.

La tendencia actual en el diseño del extractor de características es el uso de transformadas wavelets, en la opinión del autor de este proyecto, las características no deberían reducirse a los coeficientes de una transformación sino que inicialmente deberían comprender el mayor número posible de campos. Ante la ausencia de un conocimiento claro de los parámetros que mejor caracterizan la señal, y ante la imposibilidad de un método extracción sistemático de características no lineales, es propuesta personal de este autor el uso de *todas* las características que se hayan propuesto, dejando a un algoritmo de proyección de carácter general la optimización de estas características respecto a su separabilidad. En este proyecto apenas se comprobaron unas cuantas características de manera práctica, pero se emplaza a su futuro uso.

En cuanto al sistema clasificador, parece tendencia generalizada el uso de una red neuronal y así se propone como línea futura de trabajo aquí. El clasificador gaussiano bayesiano que se ensayó en el laboratorio presenta un rendimiento razonablemente bueno, si bien finalmente se decidió por implementar el método más sencillo. Otros clasificadores como el de los k-vecinos quedan pendientes de exploración, pues por su sencillez y su posibles mejoras todo trabajo parece que será compensado. El desarrollo de una red neuronal, por contra, con un rendimiento en principio superior puede suponer un trabajo importante antes de obtener resultados, con lo cual se sugiere la compra de un módulo software o la asignación de un proyecto en exclusiva para trabajar con redes neuronales como clasificador de patrones mioeléctricos[44].

[44]Si bien [91] ya realizó un proyecto sobre el tema en esta misma escuela, sería interesante el desarrollo en C++ sobre los datos reales.

El sistema software básico funciona bien y no merece la pena que sea mejorado. En todo caso, debería evolucionar hacia una orientación a objetos total. En la actualidad, la orientación a objetos está esbozada y sólo se está a un paso de conseguir un software totalmente flexible y modular.

Se ha definido la *cocontracción* como la rápida contracción doble de los músculos del brazo. Ello se utiliza en las prótesis comerciales para conmutar de tarea u otras funciones, y es algo de utilidad práctica manifiesta. En el sistema aquí propuesto, servía para hacer 'clic' cuando el usuario deseaba marcar una tecla en el programa 'espacio'. Sin embargo su rendimiento no era bueno. Fijando un umbral tal que el disparo accidental fuera despreciable, el número de cocontracciones detectadas era apenas el 20 % de los intentos, de modo que por cada letra que se quería marcar había que de media unos 5 intentos y se hacía muy pesado el manejo. La detección de la cocontracción no es un asunto particularmente difícil, y la mejora inmediata del sistema software

9.1. Recursos empleados

Para la elaboración de este proyecto fue necesaria una extensa bibliografía, diversos instrumentos, herramientas software y el ingenio de su autor así como la colaboración de su equipo.

Los recursos bibliográficos estuvieron disponibles en las bibliotecas y hemerotecas de la Escuela Técnica Superior de Ingenieros de Telecomunicación, de la Facultad de Ciencias y de la Facultad de Ciencias Económicas y Empresariales de la Universidad de Valladolid, así como de la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universidad Politécnica de Madrid. Las ponencias de congresos y la información comercial de las prótesis fue obtenida a través de la internet, y cinco documentos fueron remitidos personalmente por la cortesía de sus autores.

Estos fueron las principales aplicaciones software que fueron usadas: Visual C++ v6.0, Adobe Photoshop v5.0, Matlab v4.0, paquete de edición de texto LaTeX Miktex v2.1 (TeX, dvipdfm, yap, texify, dvi2ps), Microsoft Word 97, Microsoft Power Point 97, GSView 2.1, e Install Shield for Visual C++ 6.0, todo ello bajo un sistema operativo Windows 98. Para programar las aplicaciones se utilizó el compilador de C++ de Microsoft, Visual C++ versión 6.0. Como referencias de programación se tomaron [52][117][67][71][73]. El sistema operativo fue en todo momento Windows 95/98.

El ordenador en que se implementó todo tenía un procesador AMD K6-2 funcionando a 450MHz con 64Mb de RAM, aunque inicialente se había dispuesto sólo de un Pentium a 133MHz con 48Mb de RAM.

La capacidad de cálculo en ambos casos era sobrada para realizar la mayoría de las operaciones. Como ejemplo, en el ordenador más viejo se midió el tiempo que se tardaba en calcular el valor cuadrático medio de un patrón, y se halló que este intervalo era de 0.165 ms, es decir que el ordenador iba muy sobrado al respecto. La complejidad de este cáculo dada en operaciones de punto flotante es del orden de $O(N)$, en concreto $2N$, unas 500 operaciones de punto flotante. Si imponemos que los cálculos que haga el ordenador deben realizarse en 100ms o menos, lo cual es muy razonable para no ralentizar la decisión demasiado, tendremos incluso como complejidades aceptables algoritmos de orden $O(N^2)$. Una FFT requiere $Nlog_2N$ multiplicaciones, la FFT de un patrón son entonces unas 2000 operaciones, y por tanto es asequible.

Como material de instrumentación, se contó con una fuente de alimentación Promax FAC662B, un osciloscopio Hewlett Packard 54520A además de multímetros, generadores de funciones y un completo laboratorio electrónico. Fue indispensable la cabecera analógica P4, y el concurso del conversor A/D de Lorenzo Ferrero o la tarjeta CIODAS 1602.

El proyecto que aquí se ha presentado fue iniciado el 26 de marzo de 2001 y fue presentado en diciembre del mismo año.

9.2. Agradecimientos

Agradezco a mi tutor Ramón de la Rosa su colaboración y su pronta respuesta a la hora de solucionar los inevitables problemas que fueron surgiendo. A Alonso Alonso deseo agradecer sus interesantes sugerencias desde el pragmático punto de vista del usuario que nunca debiera ser olvidado, y sobre todo el interés y la ilusión que ha puesto en un proyecto que en principio no prometía buenos resultados. Pero por encima de todo he de agradecerles la absoluta libertad que me han concedido a la hora de trabajar; gracias a eso he podido plasmar mi creatividad y mi entusiasmo en este proyecto.

Referencias

[1] J.G. Kreifeldt[45], S. Yao. "A Signal-to-Noise Investigation of Nonlinear Electromyographic Processors", IEEE Transactions on Biomedical Engineering, vol BME-21, No 4, July 1974.

[2] D. Graupe, W. K. Cline. "Functional Separation of EMG Signals via ARMA Identification Methods for Prosthesis Control Purposes", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-5, No. 2, March 1975

[3] G.C. Agarwal, G.L. Gottlieb. "An Analysis of the Electromyogram by Fourier, Simulation and Experimental Techniques", IEEE Transactions on Biomedical Engineering, Vol. BME 22, No. 3, May 1975

[4] T.W. Calvert, A.E. Chapman. "The Relationship Between the Surface EMG and Force Transients in Muscle: Simulation and Experimental Studies". Proceedings of the IEEE, Vol. 65, No. 5, May 1977

[5] E. Shwedyk, R. Balasubramanian, R. N. Scott. "A Nonstationary Model for the Electromyogram". IEEE Transactions on Biomedical Engineering, Vol. BME 24, No. 5, September 1977.

[6] G. A. Bekey, C. Chang, J. Perry, M. Hoffer. "Pattern Recognition of Multiple EMG Signals Applied to the Description of Human Gait", Proceedings of the IEEE, Vol. 65, No. 5, May 1977.

[7] L.H. Lindström, R.I. Magnusson. "Interpretation of Myoelectric Power Spectra: A Model and Its Applications", Proceedings of the IEEE, Vol. 65, No. 5, May 1977.

[8] P.A. Parker, J.A. Stuller, R.N. Scott. "Signal Processing for the Multistate Myoelectric Channel". Proceedings of the IEEE, Vol. 65, No. 5, May 1977.

[9] D. Graupe, J. Magnussen, A.A. Beex. "A Microprocessor System for Multifuncional Control of Upper-Limb Prostheses via Myoelectric Signal Identification". IEEE Transactions on Automatic Control, vol. AC 23, No 4, Aug. 1978

[10] C. Hershler, M. Milner. "An Optimality Criterion for Processing Electromyographic (EMG) Signals Relating to Human Locomotion", IEEE Transactions on Biomedical Engineering, Vol. BME 25, No. 5, Sep. 1978

[45] La bibliografía ha sido ordenada por años, sin entrar en otro criterio.

[11] S.M. Fleisher, E. Shwedyk. "Sequential Multistate EMG Signal Processor", IEEE Transactions on Biomedical Engineering, Vol. BME-26, No. 10, Oct. 1979.

[12] C.J. de Luca. "Physiology and Mathematics of Myoelectric Signals", IEEE Transactions on Biomedical Engineering, Vol. BME-26, No. 6, June 1979

[13] N. Hogan, R. W. Mann. "Myoelectric Signal Processing: Optimal Estimation Applied to Electromyography - Part I: Derivation of the Optimal Myoprocessor", IEEE Transactions on Biomedical Engineering, Vol. BME-27, No. 7, July 1980

[14] N. Hogan, R. W. Mann. "Myoelectric Signal Processing: Optimal Estimation Applied to Electromyography - Part II: Experimental Demostration of Optimal Myoprocessor Performance", IEEE Transactions on Biomedical Engineering, Vol. BME-27, No. 7, July 1980

[15] G. N. Saridis, T. P. Gootee. "EMG Pattern Analysis and Classification for a Prosthetic Arm", IEEE Transactions on Biomedical Engineering, Vol. BME-29, No. 6, June 1982

[16] P. C. Doerschuk, D. E. Gustafson, A. S. Willsky. "Upper Extremity Limb Function Discrimination Using EMG Signal Analysis", IEEE Transactions on Biomedical Engineering, Vol. BME-30, No. 1, January 1983

[17] S. Lee, G. N. Saridis. "The Control of a Prosthetic Arm by EMG Pattern Recognition", IEEE Transactions on Automatic Control, Vol. AC-29, No. 4, April 1984

[18] J. Coatrieux, "On-Line Electroymyographic Signal Processing System", IEEE Transactions on Biomedical Engineering, Vol. BME-31, No.2, February 1984

[19] M. H. Sherif, R. J. Gregor, J. Lyman. "Comments on 'Upper Extremity Limb Function Discrimination Using EMG Signal Analysis'.", IEEE Transactions on Biomedical Engineering, Vol. BME-31, No. 5, May 1984

[20] H. B. Evans, Z. Pan, P. A. Parker, R. N. Scott. "Signal Processing for Proportional Myoelectric Control". IEEE Transactions on Biomedical Engineering, Vol.BME31, No. 2, Feb. 1984

[21] G. C. Filligoi, P. Mandarini. "Some Theoretic Results on a Digital EMG Signal Processor", IEEE Transactions on Biomedical Engineering, Vol. BME31, No. 4, Apr. 1984

[22] M. H. Hassoun, R. Spitzer. "NNERVE: Neural Network Extraction of Repetitive Vector for Electromyography -Part I: Algorithm", IEEE Transactions on Biomedical Engineering, Vol. 41, No. 11, November 1984.

[23] R. M. Studer, R. P. de Figueiredo, G. S. Moschytz. "An Algorithm for Sequential Signal Estimation and System Idetification for EMG Signals", IEEE Transactions on Biomedical Engineering, Vol. BME-31, March 1984.

[24] R. J. Triolo, G. D. Moskowitz, "Comments on 'Upper Extremity Limb Function Discrimination Using EMG Signal Analysis' and the Relationship Between Parallel-Filtering and Hypothesis-Testing Limb Function Classifiers", IEEE Transactions on Biomedical Engineering, Vol. BMB-32, No. 3, March 1985

[25] T. D'Alessio. "Analysis of a Digital EMG Signal Processor in Dynamic Conditions", IEEE Transactions on Biomedical Engineerign, Vol. BME-32, No. 1, January 1985

[26] R. Merletti, D. Biey, M. Biey, G. Prato, A. Orusa. "On-Line Monitoring of the Median Frequency of the Surface EMG Power Spectrum", IEEE Transactions on Biomedical Engineering, Vol. BME-32, No. 1, January 1985.

[27] T. Masuda, H. Myano, T. Sadoyama. "The Position of Innervation Zones in the Biceps Brachii Investigated by Surface Electromyography", IEEE Transactions on Biomedical Engineering, Vol. BME-32, No. 1, January 1985.

[28] P.R. Krishnaiah, L. N. Kanal. "Classification, Pattern Recognition and Reduction of Dimensionality", Elsevier Science, 1987

[29] L.M. Chicote, J.V. Fernández, P. Gómez, V. Rodellar. "Estimation and Classification of the Electromyographic Activity in the Human Upper Limbs through Linear Predictive Filtering". Actas del III Simposium Nacional de Reconocimiento de Formas y Análisis de Imágenes, Oviedo. Sep. 1988, pp. 309-316.

[30] K. Fukunaga. "Introduction to Statistical Pattern Recognition", Academic Press 1990

[31] A. J. Pratt, R. E. Gander, B. R. Brandell. "Real-Time Digital Median Frequency Estimator for Surface Myoelectric Analysis", IEEE Transactions on Biomedical Engineering, Vol. 38, No. 3, March 1991.

[32] L. Zhang, R. Shiavi, M. A. Hung, J. J. Chen. "Clustering Analysis and Pattern Discrimination of EMG Linear Envelopes", IEEE Transactions on Biomedical Engineering, Vol. 38, No. 8, August 1991.

[33] L. Hof, "Error in Frequency Parameters of EMG Power Spectra", IEEE Transactions on Biomedical Engineering, Vol. 38, No. 11, November 1991.

[34] T. Kiryu, Y. Saitoh, K. Ishioka. "Investigation on Parametric Analysis of Dynamic EMG Signals by a Muscle-Structured Simulation Model", IEEE Transactions on Biomedical Engineering, Vol. 39, No. 3, March 1992

[35] C. D. Brenner. "Electric Limbs for Infants and Pre-School Children", Journal of Prosthetics and Orthotics, Vol. 4, Num. 4, 1992.

[36] J. J. Chen, R. G. Shiavi, L. Zhang. "A Quantitative and Qualitative Description of Electroyogrphic Linear Envelopes for Synergy Analysis", IEEE Transactions on Biomedical Engineering, vol. 39, No. 1, January 1992.

[37] T. Iberall, D.J. Beattie, G. Sukhatme, G.A. Bekey. "Control Philosophy for a Simulated Prosthetic Hand". Proc. RESNA, Las Vegas, Nevada, June 1993.

[38] T. D'Alessio, M. Knaflitz, G. Balestra, S. Paggi. "On-Line Estimation of Myoelectric Signal Spectral Parameters and Nonstationarities Detection", IEEE Transactions on Biomedical Engineering, Vol. 40, No. 9, September 1993

[39] E. Park, S. G. Meek. "Fatigue Compensation of the Electromyographic Signal for Prosthetic Control and Force Estimation", IEEE Transactions on Biomedical Engineering, Vol. 40, No. 10, October 1993.

[40] E. Morin, P. A. Parker, R. N. Scott. "Operator Error in a Level Coded Myoelectric Control Channel", IEEE Transactions on Biomedical Engineering, Vol. 40, No. 6, June 1993.

[41] J. A. Freeman, D. M. Skapura. "Redes neuronales. Algoritmos, aplicaciones y técnicas de programación", Addison-Wesley, 1993.

[42] B. Hudgins, P. Parker, R. N. Scott. "A New Strategy for Multifunction Myoelectric Control", IEEE Transactions on Biomedical Engineering, Vol. 40, No. 1, January 1993

[43] P. A. O'Neill, E. L. Morin, R. N. Scott. "Myoelectyric Signal Characteristics from Muscles in Residual Upper Limbs", IEEE Transactions on Rehabilitation Engineering, Vol. 2, No. 4, December 1994.

[44] E.A. Clancy, N. Hogan. "Single Site Electromyograph Amplitude Estimation", IEEE Transactions on Biomedical Engineering, Vol. 44, No. 2, Feb. 1994

[45] P. Michelman, P. Allen. "Shared autonomy in a robot hand teleoperation system". Proceedings of the 1994 Conference on Intelligent Robotics Systems, 1994.

[46] Englehart, K., Hudgins, B., Stevenson, M. and P.A. Parker. "Myoelectric signal classification using a finite impulse response neural network." 16th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Baltimore, Maryland, November, 1994.

[47] A. Dupont, E. L. Morin. "A Myoelectric Control Evaluation System", IEEE Transactions on Rehabilitation Engineering, Vol. 2, No. 2, June 1994

[48] T. Iberall, D. J. Beattie, G. Sukhatme, G. A. Bekey. "On the Development of EMG Control for a Prosthetic Hand". Proceedings of the 1994 IEEE International Conference on Robotics and Automation. San Diego, May 1994.

[49] T. Iberall, D. J. Beattie, G. Sukhatme, G. A. Bekey. "EMG Control for a Robot Hand Used as a Prosthesis". Proc. International Conference on Rehabilitation Robotics, Wilmington, June 1994.

[50] P. J. Kyberd, O. E. Holland, P. H. Chappell, S. Smith, R. Tregidgo, P. J. Bagwell, M. Snaith. "MARCUS: A Two Degree o f Freedom Hand Prosthesis with Hierarchical Grip Control", IEEE Transactions on Rehabilitation Engineering, Vol. 3, No. 1, March 1995.

[51] E. A. Clancy, N. Hogan. "Multiple Site Electromyograph Amplitude Estimation", IEEE Transactions on Biomedical Engineering, Vol. 42, No. 2, February 1995.

[52] D. J. Kruglinski. "Programación avanzada con Visual C++", McGraw Hill, 1995.

[53] E. Park, S. G. Meek. "Adaptive Filtering of the Electromyographic Signal for Prosthetic Control and Force Estimation", IEEE Transactions on Biomedical Engineering, Vol. 42, No. 10, 1995

[54] W. Kang, J. Shiu, C. Cheng, J. Lai, H. Tsao, T. Kuo. "The Application of Cepstral Coefficients and Maximum Likelihood Method in EMG Pattern Recognition", IEEE Transactions on Biomedical Engineering, Vol. 42, No. 8, August 1995

[55] U. Kuruganti, B. Hudgins, R. N. Scott. "Two-Channel Enhancement of a Multifunction Control System", IEEE Transactions on Biomedical Engineering, Vol. 42, No. 1, January 1995.

[56] Englehart, K., Hudgins, B., Stevenson, M. and P.A. Parker. "Classification of myoelectric signal burst patterns using a dynamic neural network." IEEE EMBS Northeast Bioengineering Conference, Bar Harbor, Maine, May, 1995.

[57] R. Merletti, A. Gulisashvili, L.R. Lo Conte. "Estimation of Shape Characteristics of Surface Muscle Signal Spectra from Time Domain Data", IEEE Transactions on Biomedical Engineering, Vol. 42, No. 8, Aug. 1995

[58] M. Zardoshti-Kermani, B. C. Wheeler, K. Badie, R. M. Hashemi. "EMG Feature Evaluation for Movement Control of Upper Extremity Prostheses", IEEE Transactions on Rehabilitation Engineering, Vol. 3, no. 4, December 1995.

[59] Englehart, K., Hudgins, B., Stevenson, M. and P.A. Parker, "A dynamic feedforward neural network for subset classification of myoelectric signal patterns." 19th Annual International Conference of the IEEE Engineering in Medicine and Biology Society / CMBEC 21, Montreal, P.Q., September, 1995.

[60] N. Chaiyaratana, A. M. S. Zalzala, D. Datta. "Myoelectric signals pattern recognition for intelligent functional operation of upper-limb prosthesis", 1st European Conference on Disability, Virtual Reality and Associated Technologies, Maidenhead, UK, Jul. 1996.

[61] C. La Rota, F. Castaño, A. García, J. Bohorquez, "Sistema de Generación de Comandos para el Control de Prótesis Electromiográficas",

Memorias X Jornadas de Electrónica y Telecomunicaciones, ACIEM, Bogotá, Colombia, 1996.

[62] S. Lee, J. Kim, S. Park. "An Enhanced Feature Extraction Algorithm for EMG Pattern Classification", IEEE Transations on Rehabilitation Engineering, Vol. 4, No. 4, December 1996.

[63] D. Zazula, D. Korze, A. Sostaric, D. Korosec. "Study for Decomposition of Superimposed Signals with Application to Electromyograms", Neuroprosthetics, Springer Berlin 1996.

[64] A. Šoštarič, D. Korošec, D. Zazula. "Wavelet Transform in EMG Analysis". Proceedings of the 18th International Conference on Information Technology Interfaces, ITI'96. Pula, Croatia, June 1996.

[65] D. C. Montgomery, G. C. Runger. "Probabilidad y estadística aplicadas a la ingeniería", McGraw Hill, 1996.

[66] K.A. Farry, I.D. Walker, R.G. Baraniuk. "Myoelectric Teleoperation of a Complex Robotic Hand", IEEE Transactions on Robotics and Automation, Vol 12, No 5, October 1996.

[67] V. Toth, "Visual C++ 5 Unleashed", SAMS Publishing, 1997.

[68] W. M. Sloboda, V. M. Zatsiorsky. "Wavelet Analysis of EMG Signals", 21st Annual Meeting of the American Society of Biomechanics, South Carolina, Sep. 1997.

[69] A. Hernández Bajo. "Estudio sobre prótesis mioeléctricas aplicadas a extremidades superiores: desarrollo de una prótesis virtual", Proyecto Fin de Carrera ETSIT, Universidad de Valladolid, Noviembre 1997.

[70] Hudgins, B., K. Englehart, P.A. Parker, and R.N. Scott, "A microprocessor-based multifunction myoelectric control system." 23rd Canadian Medical and Biological Engineering Society Conference, Toronto, May, 1997.

[71] A. Denning. "Controles ActiveX", McGraw Hill, 1997.

[72] K.C. McGill, Z.C. Lateva, M. Zardoshti. "Estimation of Recruitment and Firing Rate in EMG Signals", Proc. 19th International Conference IEEE/EMBS Oct-Nov 1997, Chicago, USA.

[73] J. Richter. "Programación avanzada en Windows", McGraw Hill, 1997

[74] C. J. De Luca. "The Use of Surface Electromyography in Biomechanics", 1997 DelSys Inc.

[75] T. Keller, A. curt, M. R. Popovi, V. Dietz, A. Signer. "Grasping in High Lesioned Tetraplegic Subjects Using the EMG Controlled Neuroprosthesis", Journal of NeuroRehabilitation, Vol. 10, pp. 251-255, 1998.

[76] Y. St-Amant, D. Rancourt, E. A. Clancy. "Influence of Smoothing Window Length on Electromyogram Amplitude Estimates", IEEE Transactions on Biomedical Engineering, Vol. 45, No. 6, June 1998.

[77] K.C. McGill, Z.C. Lateva, M.Zardoshti. "Estimation of Recruitment by Counting the Number of Active Motor Units Seen by an Intramuscular Electrode", Proc. 12th Congress International Society of Electrophysiology and Kinesiology 1998.

[78] C. Bonivento, C. Fantuzzi. "Supervisory system of myoelectric prostheses", Proceedings of the 1998 IEEE International Conference on Control Applications. Trieste, Italy, Sep. 1998

[79] R. Hornero. "Análisis de señales biomédicas con fines diagnósticos mediante wavelets y métodos derivados de la teoría del caos", Tesis Doctoral ETSIT, Universidad de Valladolid, Mayo 1998.

[80] P. Saludes. "Extracción de características de las señales EMG mediante wavelets", Proyecto Fin de Carrera ETSIT, Universidad de Valladolid, Marzo 1998

[81] R. V. Baratta, B. Zhou, M. Solomonow, R. D. D'Ambrosia. "Force feedback control of motor unit recruitment in isometric muscle", Journal of Biomechanics, December 1998.

[82] P.J. Hyberd, J.J. Davey, J.D. Morrison. "A Survey of Upper-Limb Prosthesis Users in Oxfordshire". Journal of Prosthetics and Orthotics, Vol. 10, Num. 4, 1998.

[83] K. Englehart, B. Hudgins, P.A. Parker, M. Stevenson. "Time-Frequency Representation for Classification of the Transient Myoelectric Signal". 20th Annual International Conference of the IEEE Engineering on Medicine and Biology Society, Hong Kong, October 1998.

[84] A. Šoštarič, D. Korošec, D. Zazula. "Time-scale phase decomposition analysis", Proceeding of the IAESTED International Conference Signal and Image Proecessing, Las Vegas, USA, Oct. 1998.

[85] S. Park, S. Lee. "EMG Pattern Recognition Based on Artificial Intelligence Techniques", IEEE Transactions on Rehabilitarion Engineering, Vol. 6, No. 4, December 1998

[86] O. A. Alsayegh, D. P. Brzakovic. "Guidance of Video Data Adquisition by Myoelectric Signals for Smart Robot-Human Interface", Proceedings of the 1998 IEEE International Conference on Robotics & Automation, Leuven, Belgium, May 1998

[87] K. Englehart "Signal Representation for Classification of the Transient Myoelectric Signal", Ph. D. Thesis, University of New Brunswick, Fredericton, Canada 1998.

[88] A. Tura, C. Lamberti, A. Davalli, R. Sacchetti. "Experimental development of a sensory control system for an upper limb myoelectric prosthesis with cosmetic covering". Journal of Rehabilitation Research and Development, Vol. 35, Num. 1, January 1998.

[89] L. Eriksson, F. Sebelius, C. Balkenius. "Neural Control of a Virtual Prosthesis", ICANN 98, Perspectives in Neural Computing, Springer Verlag.

[90] E. A. Clancy, N. Hogan. "Influence of Joint Angle on the Calibration and Performance of EMG Amplitude Estimators", IEEE Transactions on Biomedical Engineering, Vol. 45, No. 5, May 1998

[91] S. Peña Calvo. "Cuantificación de los coeficientes de las señales de EMG mediante wavelets", Proyecto Fin de Carrera ETSIT, Universidad de Valladolid, Noviembre 1998

[92] C. Bonivento, A. Davalli, C. Fantuzzi, R. Sacchetti. "Automatic Tuning of Myoelectric Prostheses", Journal of Rehabilitation Research & Development, Vol. 35, No. 3, 1998.

[93] R. de la Rosa. "Diseño de un sistema de adquisición de señales bioeléctricas", Proyecto Fin de Carrera ETSIT, Universidad de Valladolid, ???? 1999

[94] C. Pfeiffer, K. DeLaurentis, C. Mavroidis. "Shape Memory Alloy Actuated Robot Prostheses: Intial Experiments". Proceedings of the 1999 IEEE International Conference on Robotics and Automotion. Detroit, Michigan, May 1999.

[95] H. Sears, J. Rendi. "A Look at Myoelectric Prosthetic Technology", Orthetics and Prosthetics Business World, December 1999.

[96] A. Tejedo. "Análisis de señales EMG mediante wavelets y redes neuronales para el control de una prótesis mioeléctrica", Proyecto Fin de Carrera ETSIT, Universidad de Valladolid, Diciembre 1999.

[97] J. Fang, G. C. Agarwal, B. T. Shahani. "Decomposition of Multiunit Electromyographic Signals", IEEE Transactions on Biomedical Engineering, Vol. 46, No. 6, June 1999.

[98] X. Shen, J. Cheng, K. Manal, T. S. Buchanan. "Design of a Real Time EMG-driven Virtual Arm", 23rd Annual Meeting of the American Society of Biomechanics, University of Pittsburgh, October 1999

[99] C. Lake. "Effects of Prosthetic Training on Upper-Extremity Prosthesis Use". Journal of Prosthetics and Orthotics. Vol. 9, Num. 1, 1999.

[100] J. P. Paul. "Strength Requirements for internal and external prostheses", Journal of Biomechanics, March 1999

[101] G. Wiley. "Upper Limb Replacement". Orthopedic Technical Review, September 1999.

[102] S. Karlsson, J. Yu, M. Akay. "Enhancemet of Spectral Analysis of Myoelectric Signals During Static Contractions Using Wavelet Methods", IEEE Transactions on Biomedical Engineering. Vol. 46, No. 6, June 1999

[103] J. Billock, C. C. Brenner, J. Miguelez, H. H. Sears. "Upper-Limb Electronic Technology Moves Forward", Orthotics & Prosthetics Business, December 1999

[104] E. A. Clancy. "Electromyogram Amplitude Estimation with Adaptive Smoothing Window Length", Vol. 46, No. 6, June 1999.

[105] Z. Shi, D. S. Zhang, D. J. Kouri, D. K. Hoffman. "Biomedical Signal Processing Using DAF Networks", IEEE 33rd Asilomar Conference on Signals, Systems and Computers. Pacific Grove, CA, USA, Oct. 1999

[106] K. Englehart, B. Hudgins, P.A. Parker, M. Stevenson. "Improving Myoelectric Signal Classification Using Wavelet Packets and Principal Component Analysis". 21st Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Atlanta, October 1999.

[107] D. Nishikawa, W. Yu, H. Yokoi, Y. Kakazu. "Analyzing and Discriminating EMG Signals using Wavelet Transform and Real-Time Learning Method", Intelligent Engineering Systems Through Artificial Nerual Networks, vol 9, pp281-286, 1999.

[108] E. A. Clancy, N. Hogan. "Probability Density of The Surface Electromyogram and Its Relation to Amplitude Dectectors", IEEE Transactions on Biomedical Engineering, Vol. 46, No. 6, June 1999

[109] D. Nishikawa, W. Yu, H. Yokoi, Y. Kakazu. "EMG Prosthetic Hand Controller Using Real-Time Learning Method", The 1999 IEEE Systems, Man and Cybernetics Conference (SMC'99), Tokyo, Japan, Oct. 1999, pp1153-1158

[110] R. Merletti, L. L. Conte, E. Avignone, P. Guglielminotti. "Modeling of Surface Myoelectric Signals. Part I: Model Implementation". IEEE Transactions on Biomedical Engineering, Vol. 46, No. 7, July 1999.

[111] R. Merletti, L.L. Conte, E. Avignone, P. Guglielminotti. "Modeling of Surface Myoelectric Signals. Part II: Model-Based Signal Interpretation". IEEE Transactions on Biomedical Engineering, Vol. 46, No. 7, July 1999.

[112] "Otto Bock System Electric Hand with DMC Plus", Otto Bock Orthopäedische Industrie, November 1999.

[113] J. Fang, G. C. Agarwall, B.T. Shahani. "Decomposition of Multiunit Electromyographic Signals". IEEE Transactions on Biomedical Engineering, Vol. 46, No. 6, June 1999.

[114] D. Nishikawa, W. Yu, H. Yokoi, Y. Kakazu. "EMG Prosthetic Hand Controller Discriminating Ten Motions using Real-Time Learning Method", IEEE International Conference on Intelligent Robots and Systems, (IROS99), Kyongiyu, Korea, Oct. 1999. pp 1592-1597.

[115] R. Merletti. "Standards for Reporting EMG Data", Journal of Electromyography and Kinesiology, Juanuary 1999.

[116] K. Englehart, B. Hudgins, P.A. Parker, M. Stevenson. "Classification of the Myoelectric Signal using Time-Frequency Based Representations". Medical Engineering and Physics, July 1999.

[117] H. Schildt. "Programación con MFC 6.0", McGraw Hill, 1999.

[118] J.M. Miguelez. "Critical Factors in Electrically Powered Upper Extermity Prosthetics", Journal of Proceedings of the American Academy of Orthotists & Prosthetists, 1999.

[119] É. L. Machado, M. A. C. Corrêa, F. L. da Cunha, T. F. Bastos Filho. "Implementação do Sentido de Temperatura em Próteses Mioelétricas para membros Superiores". Anales del Congreso Iberoamericano Iberdiscap 2000, Madrid Octubre 2000.

[120] W. Herzog, T. R. Leonard. "In Vivo Fibre Length Changes in the Cat Soleus", XXVe Congres de la Sociètè de Biomècanique, August 2000

[121] J. Duchêne, J. Hogrel. "A Model of EMG Generation", IEEE Transactions on Biomedical Engineering. Vol. 47, No. 2, February 2000.

[122] S. Karlsson, J. Yu, M. Akay. "Time-Frequency Analysis of Myoelectric Signals During Dynamic Contractions: A Comparative Study", IEEE Transactions on Biomedical Engineering. Vol. 47, No. 2, February 2000

[123] K. Englehart, B. Hudgins, P.A. Parker. "Time-Frequency Based Classification of the Myoelectric Signal: Static vs. Dynamic Contractions", 22nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Chicago, July, 2000.

[124] C. V. de Vincenzo, F. L. da Cunha, H. A. Schneebeli, V. I. Dynnikov, T. F. B. Filho. "Simulador de Arquitetura de Controla Basada em Agentes para Uso em Próteses Mioelétricas da Membro Superior", XIII Congresso Brasileiro de Automática, Florianópolis, Santa Catarina, 11 a 14 de setembro de 2000

[125] J. M. Franca, R. L. Ortolan, F. L. Cunha, V. I. Dannikov, A. Cliquet. "A Specific System for an Anthropomorphic Myoelectric Hand Prosthesis", Nonlinear Dynamics, Chaos, Control and Their Applications to Engineering Sciences. Brasil, I.L. Caldas, F.B. Rizatto, Editors, 2000

[126] A. B. Barreto, S.D. Scargle, M. Adjouadi. "A practical EMG-based human-computer interface for users with motor disabilities". Journal of Rehabilitation Research and Development, Vol. 37, Num. 1, Jan/Feb 2000.

[127] P. Dario, M.C. Carrozza, S. Micera, B. Massa, M. Zecca. "Design and Experiments on a Novel Biomechatronic Hand", Proceedings of the International Symposium on Experimental Robotics, December 2000

[128] D. Moshou, I. Hostens, G. Papaioannou, H. Ramon. "Wavelts and Self-Organising Maps in Electromyogram (EMG) Analysis", European Symposium on Intelligen Techniques 2000, Aachen, Germany

[129] C. V. De Vincenzo, F. L. da Cunha, H. A. Schneebeli, V. I. Dynnikov, T. F. Bastos Filho. "Agent-Based Control Of A Multifunction Mioelectric Prosthesis". World Congress on Medical Physics and Biomedical Engineering, Chicago, julho de 2000

[130] A. M. S. Zalzala, N. Chaiyaratana. "Myoelectric Signal Classification Using Evolutionary Hybrid RBF-MLP Networks", Proceedings of the 2000 Congress on Evolutionary Computation, July California, USA, 2000.

[131] E.A. Clancy, K.A. Farry. "Adaptive Whitening of the Electromyogram to Improve Amplitude Estimation", IEEE Transactions on Biomedical Engineering, Vol. 47, No. 6, June 2000

[132] F. L. Cunha, J.E.M. Franca, R.L. Ortolan, A. Cliquet. "O Uso de Redes Neurais Artificiais Para o Reconhecimento de Padroes em uma Prótese Mioeléctrica de Mao", Anales del congreso iberoamericano IBERDISCAP00, p 339-342. Madrid, Oct. 2000.

[133] J. Han et al., "New EMG Pattern Recognition based on Soft Computing Techniques and Its Application to Control of a Rehabilitation Robotic Arm", 6th International Conference on Soft Computing, IIZUZKA, Japan, Oct. 2000.

[134] J. Potvin, S. Brown, J. Dowling, S. Tolmie. "High Pass Filtering Beyond 100Hz Improves Surface EMG-Based Froce Predictions for the Biceps Brachii", XIth Congress of the Society for Biomechanics, August 2000.

[135] C. Bonivento, A. Davalli, C. Fantuzzi. "Tuning of Myoelectric Prostheses Using Fuzzy Logic", Artificial Intelligent in Medicine, 2000.

[136] R. L. Ortolan, V. O. Del Cura, F. A. Ferreira, M. L. Aguiar, F. L. Cunha, A. Cliquet. "Proposta de um Sistema de Controle de una Prótese Mioelétrica Multifunç ao para Membros Superiores", Anales del congreso iberoamericano IBERDISCAP00, p 339-342. Madrid, Oct. 2000.

[137] R. Gut, G.S. Moschytz. "High-Precision EMG Signal Decomposition Using Communication Techniques", IEEE Transactions on Signal Processing, Vol 48, No 9, September 2000.

[138] D. Nishikawa, Y. Ishikawa, W. Yu, M. Maruishi, I. Watanabe, H. Yokoi, Y. Mano, Y. Kakazu. "On-Line Learning Based EMG Prosthetic Hand", Electrophysiology and Kinesiology, pp 575-580, Monduzzi Editore, 2000.

[139] A. S. Oliveira, D. Rodrigues, F. Bèrzin. "EMG/Force relation in different levels of static fatiguing contractions of the biceps braquialis muscle in normal volunteers", 12th Conference of the European Society of Biomechanics, Dublin 2000

[140] B. K. Verma, C. Lane. "Vertical Jump Height Prediction using EMG Characteristics and Neural Network", Cognitive System Research, Elsevier 2000.

[141] C. V. de Vincenzo, F. L. da Cunha, H. A. Schneebeli, V. I. Dynnikov, T. F. B. Filho. "Simulador de Comportamentos para Próteses Mioelétricas da Membro Superior", Anales del Congreso Iberoamericano Iberdiscap 2000, Madrid, Outubre 2000.

[142] P. J. Sparto, M. Parnianpour, E. A. Barria, J. M. Jagadeesh. "Wavelet and Short-Time Fourier Transform Analysis of Electromyography for Detection of Back Muscle Fatigue", IEEE Transactions on Rehabilitation Engineering, Vol. 8, No. 3, September 2000.

[143] F. H. Y. Chan, Y. Yang, F.K. Lam, Y. Zhang, P. A. Parker. "Fuzzy EMG Classification for Prosthesis Control", IEEE Transactions on Rehabilitation Engineering, Vol. 8, No. 3, September 2000

[144] T. S. Buchanan, K. Manal, X. Shen, D. G. Lloyd, R. V. Gonzalez. "The Virtual Arm: estimating joint moments using an EMG-driven model", 12th Conference of the European Society of Biomechanics, Dublin, 2000

[145] Chan, A., Englehart, K., Hudgins, B., and D.F. Lovely, "Myoelectric Signals to Augment Speech Recognition." Medical and Biological Engineering & Computing, 39(4), pp. 500-506, May 2001

[146] D. Farina, R. Merletti. "A Model for the Generation of Synthetic Intramuscular EMG Signals to Test Decomposition Algorithms", IEEE Transactions on Biomedical Engineering, Vol. 48, No. 1, January 2001.

[147] Z. Bien, W. Song, D. Kim, J. Han, J. Choi, H. Lee, J. Kim, "Vision-based Control with Emergency Stop through EMG of the Wheelchair-based Rehabilitation Robotic Arm, KARES II", 7th International Conference on Rehabilitation Robotics, Insitut National des Telecommunications, Evry, France, April 2001.

[148] P. Kampas. “Myoelektroden - optimal eingesetzt”. Medizin Orth. Tech., Gentner Verlag, Januar 2001.

[149] K. Englehart, B. Hudgins, P. A. Parker. “A Wavelet Based Continuous Classification Scheme for Multifunction Myoelectric Control”. IEEE Transactions on Biomedical Engineering, Vol. 48, No. 3, March 2001

A. Glosario

- ACV. Adaptive cepstrum vector
- ARMA. Autoregresive moving average
- AR. Autoregresive
- AWGN. Aditive white gaussian noise
- CV. Cepstral vector
- DFT. Discrete Fourier transform
- DLL. Dynamic link library
- DOF. Degrees of freedom
- EDF. Euclidean discriminant function
- EEG. Electroencefalograma
- EKG. Electrocardiograma
- EMG. Electromiograma
- FES. Electroestimulación funcional
- FFT. Fast Fourier Transform
- FIR. Finite impulse response
- GUI. Graphical user interface
- HISTO. Histograma
- IAV. Integral absolute value
- LDA. Linear discriminant analysis
- LDF. Linear discriminant function
- MAV. Median absolute value
- MDF. Mahalanobis discriminant function
- MUAP. Motor unit action potential

- MU. Motor unit
- MVC. Máximo esfuerzo voluntario
- QDF. Quadratic discriminant function
- RLS. Recursive least squares
- RMS. Root mean square
- SNR. Signal to noise ratio
- STFT. Short time Fourier transform
- VAR. Variance
- WAMP. Willingson amplitude
- WL. Wave length
- WPT. Wavelet packet transform
- WT. Wavelet transform
- ZC. Zero crossings

A. Código

A.1. Biomedida4.cpp

```
/*************
TITULO: Biomedida v4.0
AUTOR: Víctor Rodríguez Doncel (vroddon@hotmail.com)
FECHAS: Julio de 2001-Octubre 2001
DESCRIPCION:
biomedida4 ofrece un interfaz a otras aplicaciones que quieran acceder a los datos de P4
se pretende aislar las capas superiores de las capas inferiores.
ofrece lo siguiente:
-Los datos en una estructura bien organizada
-Configurar la adquisicion
-Recoger datos
-Grabar los mismos a ficheros.
-Acabar la adquisicion
-Procesado mínimo de los datos (eliminación de componente DC etc.)
NOTAS:
1. Ojo, no es posible invocar desde cualquier sitio a estas funciones,
pues crean ventanas etc.
2. Se recomienda canal 1 al biceps, canal 2 al triceps por seguir un estandar
WARNINGS:
En las conversiones de datos.
BUGS:
1. El funcionamiento de los canales 3 y 4 no ha sido probado.
**************/

#include "stdafx.h"
#include "biomedida4.h"
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <malloc.h>
#include <string.h>
#include "tab1.h"
#include "abecera.h"
#include "procesado.h"
#include "conversor.h"
#include "cbw.h"
#include "Tabgeneral.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

extern "C" __declspec(dllexport) int Recoger (struct Datos *lote);
extern "C" __declspec(dllexport) void Acabar ();
extern "C" __declspec(dllexport) void Configurar(struct Datos *lote);
extern "C" __declspec(dllexport) void Grabar(struct Datos *lote);

void ConfigurarCioDas(struct Datos *lote);
void ConfigurarLorenzo(struct Datos *lote);
int RecogerCioDas(struct Datos *lote);
int RecogerLorenzo(struct Datos *lote);
void QuitarMedia(short *, int);
void QuitarPicos(short *, int);
```

```
float Mediana(float a, float b, float c, float d, float e, float g, float h);
int Leerdato();
int Complemento(int valor);
void Amplificar(short * datos,int longitud,float factor);
void FIR(short * datos,int longitud);


struct Datos loteprevio;
void *asamemoria2; //el tipo de memoria que  solicita la tarjeta ciodas
//Dialogo dialogo; //es el cuadro de dialogo...
long lastread,lastmili; //necesarios para llevar la cuenta diferencial
//de datos leidos y de milisegundos de la vez anterior
int BoardNum = 1; //número de placa, la cero es prueba. Ver CB.CFG.
tab1 paguno;
Cabecera pagdos;
conversor pagtres;
procesado pagcuatro;

//Lo que sigue a continuacion es un segmento independiente del resto para el compilador
//llamado 'compartido'. Posee no solo los atributos de lectura y escrritura (RW) sino
//tambien el atributo 'S' de compartido, es decir, que son datos compartidos
//Todos los datos de este segmento han de ser inicializado so iran a parar al segmento bss
//Las diferentes aplicaciones que utilicen los servicios de biomedida no tendran un espacio
//distinto para estas variables, sino que lo compartiran
//Lo que haga una aplicacion en estos datos sera visible para el resto

#pragma comment(linker,"/SECTION:compartido,RWS")
#pragma data_seg("compartido")
int instancias=0;
int secuenc=0;
unsigned short cdata0[BUFTAM]={0}; //datos del canal 0 para ser compartidos
unsigned short cdata1[BUFTAM]={0}; //datos del canal 1 para ser compartidos
unsigned short cdata2[BUFTAM]={0}; //datos del canal 2 para ser compartidos
unsigned short cdata3[BUFTAM]={0}; //datos del canal 3 para ser compartidos
long cvalidos=0; //numero de muestras (en total, contando todos los canales)
#pragma data_seg

/******************
Grabar
Almacena en archivo un lote de datos.
No se debe invocar a grabar si previamente no se ha configurado, por supuesto.
*******************/
void Grabar(struct Datos *lote)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState()); //Es necesaria una invocacion asi
//antes de invocar funciones MFC
//posiblemente innecesario
//Ver MFC Technical Notes 33 and 58 para mas detalles

char nombre[BUFSTR];
char fechas[BUFSTR];
char opciones[BUFSTR];
FILE *archivo;
int i;
short datito;
for (i=0;i<BUFSTR;i++)
{
nombre[i]=0;fechas[i]=0;opciones[i]=0;
}

sprintf(opciones,"Opciones: ");
if (lote->canales)
```

```
strcat(opciones,"Multicanal, ");
if (lote->ganancia)
strcat(opciones,"+20dB, ");
if (lote->fred)
strcat(opciones,"Filtro de Red, ");
if (lote->fpalto)
strcat(opciones,"Filtro bajo 40Hz, ");
if (lote->ciodas)
strcat(opciones,"Conversor: CIODAS ");
else
strcat(opciones,"Conversor: Lorenzo's ");

sprintf(nombre,"%s%d",lote->nomfile,lote->secuencia);
archivo=fopen(nombre,"wb");
fprintf(archivo,"Numero de secuencia: %d\r\n",lote->secuencia);
fprintf(archivo,"Nombre: %s\r\n",lote->nombre);
fprintf(archivo,"Fecha: %02d/%02d%/%d \r\n",lote->hora.GetDay(),lote->hora.GetMonth(),lote->hora.GetYear());
fprintf(archivo,"Hora: %02d:%02d\r\n",lote->hora.GetHour(),lote->hora.GetMinute());
fprintf(archivo,"-----------\r\n");
fprintf(archivo,"Duracion: %d ms\r\n",lote->milisegundos);
fprintf(archivo,"Datos validos: %d\r\n", lote->validos);
fprintf(archivo,"%s\r\n",opciones);
fprintf(archivo,"Frecuencia de muestreo teorica: %d Hz\r\n",lote->fmuestreo);
fprintf(archivo,"Dato flotante: %.0f\r\n",lote->flotante);
fprintf(archivo,"-----------\r\n");
if ((lote->validos)<20000)
{
fprintf(archivo,"! ");
for (i=0;i<lote->validos;i++)
{
if (lote->canales==4)
{
if ((i%4)==0)
datito=lote->data0[(i/4)];
if ((i%4)==1)
datito=lote->data1[(i/4)];
if ((i%4)==2)
datito=lote->data2[(i/4)];
if ((i%4)==3)
datito=lote->data3[(i/4)];
}
else
datito=lote->data0[i];

fprintf(archivo,"%d ",datito);
}
}
else //se pone un limite para sacar en archivo
fprintf(archivo,"Hay más de 20000 datos\n");
fclose(archivo);

}

/******************
Configurar
Establece la configuración
*******************/
void Configurar(struct Datos *lote)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState()); //ver comentarios similares

Tabgeneral tg;
```

```
if (tg.DoModal()==IDCANCEL)
{
lote->huboerror=TRUE; //si el usuario hizo 'cancelar'
return;
}
else
lote->huboerror=FALSE;

lote->ganancia=loteprevio.ganancia;
lote->fred=loteprevio.fred;
lote->fpalto=loteprevio.fpalto;
lote->autonomo=loteprevio.autonomo;
lote->ciodas=loteprevio.ciodas;
lote->lorenzo=loteprevio.lorenzo;
lote->fcorte=loteprevio.fcorte;
lote->fmuestreo=loteprevio.fmuestreo;
lote->canales=loteprevio.canales; //significa todos activos;
lote->ganancias[0]=loteprevio.ganancias[0];
lote->ganancias[1]=loteprevio.ganancias[1];
lote->ganancias[2]=loteprevio.ganancias[2];
lote->ganancias[3]=loteprevio.ganancias[3];
lote->gananciassoft=loteprevio.gananciassoft;
lote->fmedianas=loteprevio.fmedianas;
lote->fDC=loteprevio.fDC;
lote->fsoftalto=loteprevio.fsoftalto;
strcpy(lote->nombre,loteprevio.nombre);

lote->hora=CTime::GetCurrentTime();
lote->secuencia=0;
lote->validos=0;
lote->inicio=0;
lote->flotante=0;
strcpy(lote->nomfile,"dat");
lastmili=GetTickCount();
instancias++;

if (lote->ciodas)
ConfigurarCioDas(lote);
else if (lote->lorenzo) //si es adquisicion autonoma no hay configurar.
ConfigurarLorenzo(lote);
else
return;
return;
}

/***************************
ConfigurarCioDas
Establece la configuración para la placa ciodas.
***************************/
void ConfigurarCioDas(struct Datos *lote)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState()); //ver notas anteriores.
    int ULStat = 0;
    float    RevLevel = (float)CURRENTREVNUM;
    int PortNum, Direction, PortType;
int opcionestarjeta;
int configu=0;
int BitNum, BitValue;
int canalini, canalfin;

//se fija el numero de canales
if (lote->canales<CANALES)
```

```
{
canalini=lote->canales;
canalfin=lote->canales;
}
else  //es decir, todos los canales.
{
canalini=0;
canalfin=CANALES-1;
}

//Comienza la configuración basica del puerto.
ULStat = cbDeclareRevision(&RevLevel);
cbErrHandling (PRINTALL, STOPALL);
PortNum = AUXPORT;
Direction = DIGITALOUT;
ULStat = cbDConfigPort (BoardNum, PortNum, Direction);

//Envia los bits convenientes al p4.
PortType=AUXPORT;
BitNum=0;
BitValue=!(lote->ganancia);
ULStat = cbDBitOut (BoardNum, PortType, BitNum, BitValue);
BitNum=1;
BitValue=lote->fred;
ULStat = cbDBitOut (BoardNum, PortType, BitNum, BitValue);
BitNum=2;
BitValue=1; //el filtro conmutado
ULStat = cbDBitOut (BoardNum, PortType, BitNum, BitValue);
BitNum=3;
BitValue=!(lote->fpalto);
ULStat = cbDBitOut (BoardNum, PortType, BitNum, BitValue);

// Perpara a la tarjeta para adquirir datos.
opcionestarjeta=BACKGROUND | CONTINUOUS;
asamemoria2=cbWinBufAlloc(long(BUFTAM));  //funciona incluso siendo dinamico !!!
if (!asamemoria2)
AfxMessageBox("no se puedo asignar memoria al CIODAS");

//BUFTAM-4 podria ser 1024; en realidad es canales*muestrasatomar.
if (cbAInScan(BoardNum,canalini,canalfin,BUFTAM-4,&(lote->fmuestreo),BIP5VOLTS,asamemoria2,opcionestarjeta))
AfxMessageBox("Hubo un error al adquirir las muestras");

}


/*************
ConfigurarLorenzo
Establece la configuración para la placa de Lorenzo
***************/
void ConfigurarLorenzo(struct Datos *lote)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState()); //la he puesto yo
int configu=0;
int freq,muestras;
//Mando datos de configuración al p4.
configu+=(1)*0x80; //m_fcorte
configu+=(lote->fred)*0x40;
configu+=(lote->ganancia)*0x20;
if (lote->canales==4)
configu+= 0x04; //en el caso de ser multicanal
_outp(DATOS,0x02);
_outp(CONTROL,0x00);
```

```
Sleep (100);
_outp(DATOS,configu);
_outp(CONTROL,0x04);
//Mando frecuencia de muestreo a la tarjeta A/D
if ((lote->fmuestreo!= 512) && (lote->fmuestreo!=1024))
lote->fmuestreo=1024;
if (lote->canales==4) //los cuatro canales...
{
freq = 24000/(lote->fmuestreo * 4);
muestras=(24000/freq)/40;
lote->validos=(24000/freq)/40;
}
else
{
freq = 24000/lote->fmuestreo;
muestras = (24000 / freq)/10 ;
lote->validos=(24000 / freq)/10 ;
}
_outp(DATOS,freq);
_outp(CONTROL,0xC);
}

/***************
Recoger
Toma los datos y los mete en la estructura.
*****************/
int Recoger(struct Datos *lote)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState());
long temp,aux;
int i,j=0;
short datito;
int destino;

if (lote->autonomo)
{
if (instancias==1) //si es la unica instancia...
{
lote->validos=0; //entonces no hay datos disponibles
return 0;
}

if ((cvalidos<BUFTAM*4)&&(secuenc>0))
{
lote->validos=cvalidos;
secuenc--;
}
else
lote->validos=0;
for (i=0;i<lote->validos/4-4;i++)
{
lote->data0[i]=cdata0[i];
lote->data1[i]=cdata1[i];
lote->data2[i]=cdata2[i];
lote->data3[i]=cdata3[i];
}
return 0;
}


lote->secuencia++; // Se incrementama el numero de secuencia del lote.
temp=(long) GetTickCount();
```

```
lote->milisegundos=temp-lastmili;
lastmili=temp;

if (lote->ciodas)
RecogerCioDas(lote);
else if (lote->lorenzo)
RecogerLorenzo(lote);

secuenc=1; // y no es secuenc++, porque se quiere secuenc=1,
// si hay datos listos para ser cogidos por otra aplicacion.
for (i=0;i<lote->validos;i++) //relleno data0, data1 etc. no es obligatorio...
{
if (lote->ciodas)
{
aux=(lote->inicio)+i-(lote->validos);
if (aux<0)
aux=BUFTAM + aux;
datito=lote->data[aux];
datito=(WORD)(datito>>4);
datito=datito&4095;
datito-=2048; //para centrar las muestras en el cero
destino=(lote->data[aux])&(3);
}
else if (lote->lorenzo)
{
datito=lote->data[(lote->inicio+i)%(1024)];
destino=i%4;
}
if (lote->canales)
{
if (destino==0)
lote->data0[j/4]=datito;
if (destino==1)
lote->data1[j/4]=datito;
if (destino==2)
lote->data2[j/4]=datito;
if (destino==3)
lote->data3[j/4]=datito;
j++;
}
else
lote->data0[i]=datito;
}
cvalidos=lote->validos;
if (cvalidos>10000)
cvalidos=1;
for (i=0;i<cvalidos/4-4;i++)
{
cdata0[i]=lote->data0[i];
cdata1[i]=lote->data1[i];
}

if (strcmp(lote->nombre, "Comparar")==0)
{
for (i=0;i<lote->validos/4;i++)
lote->data3[i]=lote->data0[i];
}

if (lote->fmedianas)
{
QuitarPicos((short *)lote->data0,lote->validos/4);
QuitarPicos((short *)lote->data1,lote->validos/4);
```

```
}
if (lote->fDC)
{
QuitarMedia((short *)lote->data0,lote->validos/4);
QuitarMedia((short *)lote->data1,lote->validos/4);
}
if (lote->gananciassoft)
{
Amplificar((short *)lote->data0,lote->validos/4,lote->ganancias[0]);
Amplificar((short *)lote->data1,lote->validos/4,lote->ganancias[1]);
}
if (lote->fsoftalto)
{
FIR((short *)lote->data0,lote->validos/4);
FIR((short *)lote->data1,lote->validos/4);
}


return 0;
}



/***************
RecogerCioDas
RecogerCioDas es especifico para la tarjeta CioDas
*****************/
int RecogerCioDas(struct Datos *lote)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState());
long leidos;
long inicio;
int resultado;
short estatus;

if(cbGetStatus(BoardNum,&estatus,&leidos,&inicio))
{
AfxMessageBox("Hubo un error gordo en getstatus");
exit(1);
}
cbWinBufToArray (asamemoria2, lote->data, 0, BUFTAM-1);
resultado=cbGetStatus(BoardNum,&estatus,&leidos,&inicio);
if (estatus!=RUNNING) //lo contrario de running es iddle.
{
AfxMessageBox("Se ha detenido la captura de datos!");
exit(1);
}
lote->inicio=inicio;

if (lote->secuencia<2)
{
lote->validos=0;
lote->flotante=0;
}
else
lote->validos=leidos-lastread;
lastread=leidos;
return 0;
}

/*****
RecogerLorenzo
```

```
Recoge los datos de la tarjeta de Lorenzo.
********/
int RecogerLorenzo(struct Datos *lote)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState());
int deberia,i,dato;
lote->inicio=0; //los datos van siempre al principio.
//en lorenzo hay que calcular directamente las muestras que recoger.
//se fijara un maximo en 1024 MAXMUESTRASLORENZO

deberia=(lote->milisegundos)*(lote->fmuestreo)/1000;
if (lote->canales)
deberia*=4;

lote->validos=deberia-4; //se concede un margen de 4 muestras...
if (lote->validos>4095)
lote->validos=4095;
for (i=0;i<lote->validos;i++)
{
dato=Leerdato();
lote->data[i]=dato;
}
return 0;
}



/************
Acabar
Acabar libera la adquisicion de la ciodas
BUG: Deberia saber distinguir si hay o no aplicacion ciodas en ejecucion.
pues todas las aplicacions, ciodas o no, llamaran a esta funcion
***************/
void Acabar()
{
AFX_MANAGE_STATE(AfxGetStaticModuleState());
if (loteprevio.ciodas)
{
cbStopBackground(BoardNum);
cbWinBufFree (asamemoria2);
}
instancias--;
}

/*************
Complemento
devuelve el complemento a 2 de un numero.
**************/
int Complemento(int valor)
{
if ((valor & 2048) == 2048)
{
valor ^= 0x0FFF;

valor += 1;
valor *= -1;
}
return valor;
}

/***************
Leerdato
```

```
lee un dato de la placa de lorenzo.
*****************/
int Leerdato()
{
int dato;
dato = _inp(DATOS) <<4;  // Lee y desplaza 4 bits a la izquierda;
dato = dato + (_inp(DATOS)>>4);//desplaza 4 bits a la derecha y  lo
// suma a lo que ya tenia.
dato = Complemento(dato);// Halla el complemento a 2.
return dato;
}


/////////////////////////////////////////////////////////////////////////////
// CBiomedida4App

BEGIN_MESSAGE_MAP(CBiomedida4App, CWinApp)
//{{AFX_MSG_MAP(CBiomedida4App)
// NOTE - the ClassWizard will add and remove mapping macros here.
//    DO NOT EDIT what you see in these blocks of generated code!
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CBiomedida4App construction

CBiomedida4App::CBiomedida4App()
{
// TODO: add construction code here,
// Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CBiomedida4App object

CBiomedida4App theApp;

/************************
Amplificar
Multiplica la señal por un factor fijo.
************************/
void Amplificar(short * datos,int longitud,float factor)
{
int i;
for (i=0;i<longitud;i++)
datos[i]=(short)(datos[i]*factor);
}


/************************
QuitarMedia
Quita la componente dc de la señal.
************************/
void QuitarMedia(short * datos,int longitud)
{
int i;
float media=0;
for (i=0;i<longitud;i++)
media+=datos[i];
media/=longitud;
for (i=0;i<longitud;i++)
datos[i]=datos[i]-(short)media;
```

```
}

/************************
Filtropasoalto
Hace un filtro paso alto de la señal.
La frecuencia de corte, entre 0 y 1, es de 0.02. Es decir, unos 10Hz.
Es un filtro FIR generado con Matlab como FIR1.
************************/
void FIR(short * datos,int longitud)
{
int i,j;
short *datoss;
float coeficientes[]={-13, -15, -20, -28, -39, -53, -53, -68, -85, -104, -122, -140, -157, -171,
-184, -193, -198, 9813, -198, -193, -184, -171, -157, -140, -122, -104, -85, -68, -53, -39, -28,
-20, -15, -13};
for (i=0;i<33;i++)
coeficientes[i]/=10000;
datoss=(short *)malloc((longitud+33)*sizeof(short));
for(i=0;i<longitud;i++)
datoss[i+32]=datos[i];
for (i=0;i<32;i++)
datoss[i]=0; //de momento paddeo con ceros.
for (i=0;i<longitud;i++)
{
datos[i]=0;
for (j=1;j<33;j++)
datos[i]=datos[i]+(short)(coeficientes[j-1]*datoss[i-j+33]);
}
}

/**********
QuitarPicos
Quita los picos espúreos de la señal
**********/
void QuitarPicos(short * datos,int longitud)
{
long i;
float media=0;
for (i=3;i<longitud-3;i++)
{
if ((abs(datos[i])>abs(3*datos[i-1]))&&datos[i-1])
datos[i]=Mediana(datos[i-3],datos[i-2],datos[i-1],datos[i],datos[i+1],datos[i+2],datos[i+3]);
}
//quita picos de la manera mas simple por si han quedado algunos
for (i=1;i<longitud;i++)
if ((abs(datos[i])>abs(3*datos[i-1]))&&datos[i-1])
{
if (datos[i]*datos[i-1]>0) //del mismo signo
datos[i]=datos[i-1]*3; //lo limito al triple
else //de signo contrario
datos[i]=-1*datos[i-1]*3; //lo limito al triple
}
}

/*************
Mediana
devuelve la mediana de 7 numeros
***************/
float Mediana(float a, float b, float c, float d, float e,float g, float h)
{
int i,j; //algoritmo propio!!
float f[7];
```

```
int puntos;
f[0]=a+0.01;f[1]=b+0.02;f[2]=c+0.03;f[3]=d+0.04;f[4]=e+0.05;f[5]=g+0.06;f[6]=h+0.07;
for (i=0;i<7;i++)
{
puntos=0;
for (j=0;j<7;j++)
{
if (f[j]>f[i]) puntos++;
if (f[j]<f[i]) puntos--;
}
if (puntos==0) //es el del medio!!
return f[i];
}
return 0;
}
```

A.2. Biomedida4.h (incompleto)

```
// Dirección del registro de datos del puerto paralelo.
// Dirección del registro de control del puerto paralelo.
// Dirección del registro de estado del puerto paralelo.
#define DATOS 0x37C
#define CONTROL 0x37A
#define ESTADO 0x379
#define TRUE    1
#define FALSE   0
// Ojo con el tamaño de los datos, que no es solo 1 byte....
//BUFTAM es el tamaño del buffer de datos
#define BUFTAM 2048*4*4+4 //el ultimo por4 lo he puesto el 25 de julio
//BUFSTR es el tamaño del buffer de las cadenas de caracteres
#define BUFSTR 100
//POSICIONES es el número de posiciones definidas (reposo, mve, 50%mve, etc.)
#define POSICIONES 3
//NIVELESH es el numero de niveles del histograma
//NIVELESC es el nivel medio
#define NIVELESH 9
#define NIVELESC (NIVELESH-1)/2
//MAXMUESTRASLORENZO es el numero maximo que recogere de Lorenzo's placa.
#define MAXMUESTRASLORENZO 1024
//CANALES es el numero maximo de canales del p4
#define CANALES 4

/***********
Datos
Es la estructura de datos que representa un lote de datos, un paquetito...
************/
struct Datos
{

unsigned short data[BUFTAM];//datos(dato0,dato1,dato0,dato1...) si hay dos canales p.ej.
signed short data0[BUFTAM];//datos del canal 0.
signed short data1[BUFTAM];//datos del canal 1.
signed short data2[BUFTAM];//datos del canal 2.
signed short data3[BUFTAM];//datos del canal 3.
```

```
long validos; //numero de muestras (en total, contando todos los canales)
long inicio; //inicio a partir del cual tienen sentido las muestras.
char nombre[BUFSTR]; //nombre del sujeto
char nomfile[BUFSTR]; //nombre del archivo(+nº de secuencia)
BOOL huboerror; //indica si hubo algun error
BOOL ganancia; //ganancia hardware de 30dB.
BOOL fred; //filtro notch de 50Hz de la cabecera
BOOL fpalto; //prefiltro paso bajo de 40hz,
BOOL ciodas; //si FALSE, es que es lorenzo's.
BOOL autonomo; //adquisicion autonoma
BOOL lorenzo; //
BOOL fcorte; //true si frecuencia de corte alta para lorenzo's
BOOL fmedianas; //filtro de medianas
BOOL fsoftalto; //filtro paso alto software
BOOL fDC; //filtro DC
BOOL gananciassoft;
int canales; //numero de canales adquiridos.
//canales=0, canal0, canales=1, canal1, canales=4, todos
long milisegundos; //milisegundos que abarca el intervalo
float ganancias[4]; //ganancias software
int secuencia; //es el numero de secuencia.
long fmuestreo; //frecuencia de muestreo (por defecto 1024).
CTime hora; //indica la hora actual.
float flotante; //propositos varios...

};

/*****************
Caracteristica
Es la estructura de datos que representa las caracteristicas de un sujeto,
una posicion en concretom, no el global...
*****************/
struct Caracteristica
{
char nombre[BUFSTR]; //nombre del sujeto
char comentario[BUFSTR]; //un cometario
int fragmentos; //numero de fragmentos que compusieron el lote.
float medmed[CANALES]; //media de las medias
float medvar[CANALES]; //desv de las medias
float varmed[CANALES]; //media de las desv
float varvar[CANALES]; //desv de las desv
float zcmed[CANALES]; //media de los zero crossings
float zcvar[CANALES]; //desv de los zero crossings
float iavmed[CANALES]; //media de los valores iav
float iavvar[CANALES]; //desv de los valores iav
float damvmed[CANALES]; //media de los valores damv.
float damvvar[CANALES]; //desv de los valores damv
float hismed[CANALES][NIVELESH]; //media de los histogramas
float hisvar[CANALES][NIVELESH]; //desv de los histogramas
float umbralruido; //el 95% de las muestras estan bajo este nivel.
struct Datos *datos;
};
```

A.3. vroddon.cpp

```
/*************
TITULO: Espacio
AUTOR: Víctor Rodríguez Doncel (vroddon@hotmail.com)
FECHAS: Abril - Octubre 2001
DESCRIPCION:
 vroddon.dll es el codigo para decidir que hacer con unos datos.
 Será invocado por brazo, para hacer las cuentas.
 Implementa todos los algoritmos matematicos necesarios.
NOTAS:
BUGS:
Ninguno conocido
**************/
#include "stdafx.h"
#include "vroddon.h"

#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "biomedida4.h"
#include <mmsystem.h>
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


extern "C" __declspec(dllexport) int Decisor (struct Datos *lote, int criterio);
extern "C" __declspec(dllexport) int Analiza(char *nombre, int fragmentos, struct Datos *lote);
extern "C" __declspec(dllexport) void Limpiar(struct Datos *lote);
extern "C" __declspec(dllexport) void LeeVectores(char *nombre, struct Caracteristica *carac2);
extern "C" __declspec(dllexport) void Histograma(short *datos, long longitud,float niveles[NIVELESH]);
extern "C" __declspec(dllexport) float Std(short *datos,long longitud);

extern "C" __declspec(dllexport) void Compactar(char *nombre, int fragmentos);
extern "C" __declspec(dllexport) void Guardar(char *nombre);
extern "C" __declspec(dllexport) void GuardarUmbral(char *nombre,int umbral);

extern "C" __declspec(dllexport) int Clasificador(struct Datos *lote,
struct Caracteristica carac3[7], int estados);
extern "C" __declspec(dllexport) BOOL DobleClic(int datoslargos[2047]);



//FUNCIONES DE CALCULO
float Mediana(float a, float b, float c, float d, float e);
float Rms(short *datos,long longitud);
float Std(short *datos,long longitud);
float Std(float *datos,long longitud);
float Media(short *datos,long longitud);
float Media(float *datos,long longitud);
float ZC(float *datos,long longitud);
float ZC(short *datos,long longitud);
float IAV(short *datos,long longitud);
float DAMV(short *datos,long longitud);
void Histograma(short *datos, long longitud,float niveles[NIVELESH]);
```

```
void Limpiar(struct Datos *lote); //fuera de uso, ahora se hace en biomedida y bien
int Secuencia();
void IniciaVectores(struct Caracteristica *carac2);
void IniciaTodosVectores(void);
float Desbordan(short *datos, long longitud, int limitex);
int Umbral(short *datos, long longitud, int enesima);

//FUNCIONES DE PRESENTACION DE DATOS EN ARCHIVOS
void Compactar(char *nombre, int fragmentos); //compacto los datos.
void Guardar(char *nombre);
void GuardarUmbral(char *nombre);
int LeerUmbral(char *nombre, int *umbral);

//FUNCIONES DEL TRATAMIENTO DE LOS VECTORES
void EscribeVectores(char *nombre, struct Caracteristica *carac2);
void RellenaVectores(void);
void LeeVectores(char *nombre, struct Caracteristica *carac2);
float DistanciaPonderada (float vector1[CANALES][14],float vector2[CANALES][14],float peso[CANALES][14]);
float DistanciaEuclidea (float vector1[CANALES][14],float vector2[CANALES][14]);

int sec; //es el numero de secuencia de datos almacenados
struct Caracteristica carac;
struct Caracteristica carac3[7];
BOOL primeravez=TRUE;
float vector2[CANALES][14];
float vector[CANALES][14];
float peso[CANALES][14];
int umbralruido=0; //muestra numero 22 en amplitud

float oldfeature0=0, oldfeature1=0;


/////////////////////////////////////////////////////////////////////////////
// CVroddonApp

BEGIN_MESSAGE_MAP(CVroddonApp, CWinApp)
//{{AFX_MSG_MAP(CVroddonApp)
// NOTE - the ClassWizard will add and remove mapping macros here.
//    DO NOT EDIT what you see in these blocks of generated code!
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CVroddonApp construction

CVroddonApp::CVroddonApp()
{
// TODO: add construction code here,
// Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CVroddonApp object
CVroddonApp theApp;


/***********************
Decisor
Decide en base a los datos sobre cual es la mejor decision
************************/
int Decisor (struct Datos *lote, int criterio)
{
```

```
int decision=0;
int i,imin; //imin es el indice del minimo d elas distancias.
float min;
float distancias[7];
char nombre[BUFSTR];
float histograma[CANALES][NIVELESH];
float feature0, feature1;
struct Caracteristica carac2;
if (primeravez)
{
primeravez=FALSE;
IniciaTodosVectores();
}

if (!lote->validos) //si no hay datos es mejor regresar.
return 0;


Histograma((short *)lote->data0,lote->validos/4,histograma[0]);
Histograma((short *)lote->data1,lote->validos/4,histograma[1]);
feature0=histograma[0][NIVELESC];
feature1=histograma[1][NIVELESC];

feature0=0.3*feature0+0.7*oldfeature0;
feature1=0.3*feature0+0.7*oldfeature1;
oldfeature0=feature0;
oldfeature1=feature1;

min=20000;
imin=0;
for(i=0;i<5;i++)
{
distancias[i]=sqrt((feature0-carac3[i].hismed[0][NIVELESC])
*(feature0-carac3[i].hismed[0][NIVELESC])+(feature1-carac3[i].hismed[1][NIVELESC])
*(feature1-carac3[i].hismed[1][NIVELESC]));

if (distancias[i]<min)
{
min=distancias[i];
imin=i;
}
}
decision=imin;
lote->flotante=Std((short *)lote->data0,lote->validos/4);
return decision;
}


/**********************
Analiza
Analiza los datos ya recogidos, estaticamente
abre los 'fragmentos' archivos que sean, los lee y calcula
datos, con sus medias y varianzas.
el resultado de reposo1, reposo2, reposo3... lo gurada en reposo0
**********************/
int Analiza(char *nombre, int fragmentos, struct Datos *lote)
{
int i,j,k,l,w,lim;
// lim es el numero de canales, o 1 o 4
FILE *archivo;
short datos[CANALES][BUFTAM];
char cadena[100];
```

```
float media[CANALES][50]; //50 es el numero maximo de cacharros.
float zc[CANALES][50];
float desviacion[CANALES][50];
float iav[CANALES][50];
float damv[CANALES][50];
float histograma[CANALES][NIVELESH];
float histogramas[CANALES][NIVELESH][50];
float umbrales[50]; //umbral de ruido para el canal cero.
int umbral;

Secuencia(); //Halla el número de secuencia para los archivos

for (i=0;i<CANALES;i++)
for (j=0;j<50;j++)
{
media[i][j]=0;zc[i][j]=0;umbrales[j]=0;iav[i][j]=0;damv[i][j]=0;
}


//CALCULO LOS DATOS EN CADA FRAGMENTO
if (lote->canales==4) lim=4; else lim=1;

for(i=2;i<fragmentos;i++) //para cada uno de los cincuenta archivos...
{
sprintf(cadena,"%s%d",nombre,i);
archivo=fopen(cadena,"r"); //abro el archivo
j=0;
while( getc(archivo)!=('!')) //pongo el puntero en la posicion de los datos...
if ((++j)>10000) //hemos patinado por algun sitio.
exit(1);
k=0;
do
{
fscanf(archivo,"%d ",&datos[k%lim][k/lim]);
k++;
}while(!feof(archivo));

//ya he leido los datos y los he metido en su canal correspondiente.

for (w=0;w<lim;w++)
{
media[w][i-2]=Media(datos[w],k/lim);
desviacion[w][i-2]=Std(datos[w],k/lim);
zc[w][i-2]=ZC(datos[w],k/lim);
iav[w][i-2]=IAV(datos[w],k/lim);
damv[w][i-2]=DAMV(datos[w],k/lim);

Histograma(datos[w],k/lim,histograma[w]);
for (l=0;l<NIVELESH;l++)
histogramas[w][l][i-2]=histograma[w][l];
}

umbrales[i]=Umbral(datos[0],k/4,20);
fclose(archivo);
}

umbral=(int)Media(umbrales,fragmentos-2);
carac.umbralruido=umbral;
GuardarUmbral("umbral.cfg",umbral);

//HAGO LA MEDIA Y LA VARIANZA DE LOS FRAGMENTOS.
```

```
for (w=0;w<lim;w++)
{
carac.medmed[w]=Media(media[w],fragmentos-2);
carac.medvar[w]=Std(media[w],fragmentos-2);

carac.zcmed[w]=Media(zc[w],fragmentos-2);
carac.zcvar[w]=Std(zc[w],fragmentos-2);

carac.varmed[w]=Media(desviacion[w],fragmentos-2);
carac.varvar[w]=Std(desviacion[w],fragmentos-2);

carac.iavmed[w]=Media(iav[w],fragmentos-2);
carac.iavvar[w]=Std(iav[w],fragmentos-2);

carac.damvmed[w]=Media(damv[w],fragmentos-2);
carac.damvvar[w]=Std(damv[w],fragmentos-2);

for (l=0;l<NIVELESH;l++)
{
carac.hismed[w][l]=Media(histogramas[w][l],fragmentos-2);
carac.hisvar[w][l]=Std(histogramas[w][l],fragmentos-2);
}
}
carac.datos=lote; //validez limitada!! al acabarse la funcion, se joroba.

// GUARDO LOS DATOS EN UN ARCHIVO
// Compactar(nombre,fragmentos); //Temporalmente quitado. No deberia venir aqui!!!

// Guardar(nombre); //Temporalmente quitado. No debria venir aqui.
// OOOOOOOOOOJJJJJJJJJOOOOOOOOOOOO
//RellenaVectores();
//empieza la chapuza
// carac.umbralruido=-227;
EscribeVectores(nombre,&carac);
return 0;
}


/*********
Media. Esta funcion devuelve el valor medio de las señales.
Se le pasan las señales. Sobrecargada
Tardaba exactamente 0.165 milisegundos en este ordenador (P133)
Victor, Abril 2001
*********/
float Media(float *datos,long longitud)
{
long j;
float media=0;
if (!longitud)
return -1;

for (j=0;j<longitud;j++)
media+=datos[j];
media/=longitud;
return((float)media);

}

float Media(short *datos,long longitud)
{
long j;
float media=0;
```

```
if (!longitud)
return -1;
for (j=0;j<longitud;j++)
media+=datos[j];
media/=longitud;
return((float)media);

}


/*********
Rms. Esta funcion devuelve el valor RMS de las señales.
Se le pasan las señales. Esta sobrecargada
Tardaba exactamente 0.165 milisegundos en este ordenador (P133)
Le quito la media!!!!!!!
Victor, Abril 2001
*********/
float Rms(short *datos,long longitud)
{
long j;
float rms;
if (!longitud)
return -1;

rms=0;
for (j=0;j<longitud;j++)
rms+=datos[j]*datos[j];
rms/=longitud;
return((float)sqrt(rms));

}

/*********
Std. Esta funcion devuelve la desviación estandar de las señales.
Se le pasan las señales. Esta sobrecargada
Victor, Julio 2001
*********/
float Std(short *datos,long longitud)
{
long j;
float med;
float std=0;
if (!longitud)
return -1;

med =Media(datos,longitud);
for (j=0;j<longitud;j++)
std+=(float)fabs(datos[j]-med)*(float)fabs(datos[j]-med);
std=sqrt(std);
std/=(float)longitud;


return(std);
}
float Std(float *datos,long longitud)
{
long j;
float med;
float std=0;
if (!longitud)
return -1;
```

```
med =Media(datos,longitud);
for (j=0;j<longitud;j++)
std+=(float)fabs(datos[j]-med)*(float)fabs(datos[j]-med);
std=sqrt(std);
std/=(float)longitud;


return(std);
}


/*********
ZC Esta funcion devuelve el numero de cruces con cero de las señales.
Esta sobrecargada. Deberia ser mejorada, pues no contar cruces con cero
sino cruces con cero con algo de margen
Version 2:
damos los ZC despues de restar la media!!
damos los valores por cada muestra.
Tarda exactamente 0.11 milisegundos en este ordenador (P133)
Victor, Abril 2001
*********/
float ZC(short *datos,long longitud)
{
int j;
float zc,media;
if (!longitud)
return -1;

BOOL signo=TRUE; //1=positivo, 0=negativo
zc=0;
media=Media(datos,longitud);
for (j=0;j<longitud;j++)
{
if (((datos[j]>media)&&(signo==FALSE))||((datos[j]<media)&&(signo==TRUE)))
{
zc++;
signo==TRUE ? signo=FALSE : signo=TRUE;
}
}
zc=zc/longitud;
return zc;

}

float ZC(float *datos,long longitud)
{
int j;
float zc,media;
if (!longitud)
return -1;

BOOL signo=TRUE; //1=positivo, 0=negativo
zc=0;
media=Media(datos,longitud);
for (j=0;j<longitud;j++)
{
if (((datos[j]>media)&&(signo==FALSE))||((datos[j]<media)&&(signo==TRUE)))
{
zc++;
signo==TRUE ? signo=FALSE : signo=TRUE;
}
```

```
}
zc=zc/longitud;
return zc;
}

/*********
IAV Esta funcion devuelve la Integral del Valor Absoluto.
Se le pasan las señales y el sitio donde se da el resultado
Tarda exactamente ??? milisegundos en este ordenador (P133)
Victor, Abril 2001
*********/
float IAV(short *datos,long longitud)
{
int j;
float iav=0, media;
if (!longitud)
return -1;
media=Media(datos,longitud);

for (j=0;j<longitud;j++)
iav+=abs(datos[j]-media);
iav/=longitud;
return iav;
}

/*********
DAMV Esta funcion devuelve la diferenca media en valor absoluto..
Se le pasan las señales y el sitio donde se da el resultado
Tarda exactamente ??? milisegundos en este ordenador (P133)
Victor, Abril 2001
*********/
float DAMV(short *datos,long longitud)
{
int j;
float damv=0;
if (!longitud)
return -1;
for (j=0;j<longitud-1;j++)
damv+=abs(datos[j+1]-datos[j]);
damv/=longitud;
return damv;
}

/***************
HISTOGRAMA
Devuelve la distribucion en histograma, normalizada a 1,
en el numero de niveles que se especifique en biomedida4.h
***************/
void Histograma(short *datos, long longitud,float niveles[NIVELESH])
{
int i,j;
float media;
int umbral;
short limite[NIVELESH];
if (!longitud)
return;

media=Media(datos,longitud);
for (j=0;j<NIVELESH;j++)
niveles[j]=0;
for (j=0;j<(NIVELESH);j++)
limite[j]=((4096/NIVELESH)*j)-2048;
```

```
for (j=0;j<longitud;j++)
{
for (i=(NIVELESH-1);i>=0;i--)
{
if ((datos[j]-media)>=limite[i])
{
niveles[i]++;
goto infame;
}
}
infame:
;
}

//trafullo vil
//este umbar de sensibilidad es para el caso de que haya ruido.
// en oondiciones normales, 225, esta bastante bien.


niveles[NIVELESC]=Desbordan((short *)datos,longitud,-227)-Desbordan((short *)datos,longitud,227);
niveles[NIVELESC]=Desbordan((short *)datos,longitud,-227)-Desbordan((short *)datos,longitud,227);

umbral=LeerUmbral("umbral.cfg",&umbral);
/*
niveles[NIVELESC]=Desbordan((short *)datos,longitud,-1*umbral)-Desbordan((short *)datos,longitud,umbral);
niveles[NIVELESC]=Desbordan((short *)datos,longitud,-1*umbral)-Desbordan((short *)datos,longitud,umbral);
*/

for (i=(NIVELESH-1);i>=0;i--)
niveles[i]/=longitud;

}

/*************************
Umbral
Halla el umbral de ruido. Da la muestra enésima tras ordenar por amplitud.
Asi me aseguro que, digamos, el 90% de las muestras no rebasan ese umbral.
Algoritmo mio del quienes le superan?
*************************/
int Umbral(short *datos, long longitud, int enesima)
{
int i,j,lesuperan,iguales;
for (i=0;i<longitud;i++)
{
iguales=-1;
lesuperan=1;
for (j=0;j<longitud;j++)
{
if (datos[i]==datos[j])
iguales++;
if (datos[j]>datos[i])
lesuperan++;
}
if ((lesuperan==enesima)||((lesuperan<enesima)&&((lesuperan+iguales)>=enesima)))
return datos[i];
}
return -228; //nunca deberia ser alcanzado este punto
}



/************
```

```
Compactar
Almacena todos los fragmenos en un .zip
Borra todos los fragmentos.
*************/
void Compactar(char *nombre, int fragmentos)
{
char cadena[100];
sprintf(cadena,"pkzip %s%d.zip %s*",nombre,sec,nombre);
system(cadena);
sprintf(cadena,"del %s?",nombre);
system(cadena);
sprintf(cadena,"del %s??",nombre);
system(cadena);
}

/*********
Guardar
Almacena en un archivo el analisis estadistico del asunto.
**********/
void Guardar(char *nombre)
{
char cadena[BUFSTR];
char opciones[BUFSTR];
FILE *destino;
int l,w,lim;

if ((carac.datos)->canales==4) lim=4; else lim=1;


sprintf(cadena,"0%s%d",nombre,sec);
destino=fopen(cadena,"w");
sprintf(opciones,"Opciones: ");

if ((carac.datos)->canales)
strcat(opciones,"Multicanal, ");
if ((carac.datos)->ganancia)
strcat(opciones,"+20dB, ");
if ((carac.datos)->fred)
strcat(opciones,"Filtro de Red, ");
if ((carac.datos)->fpalto)
strcat(opciones,"Filtro bajo 40Hz, ");
if ((carac.datos)->ciodas)
strcat(opciones,"Conversor: CIODAS ");
else
strcat(opciones,"Conversor: Lorenzo's ");

fprintf(destino,"Nombre: %s\r\n",(carac.datos)->nombre);
fprintf(destino,"Fecha: %02d/%02d%/%d \r\n",(carac.datos)->hora.GetDay(),
(carac.datos)->hora.GetMonth(),(carac.datos)->hora.GetYear());
fprintf(destino,"Hora: %02d:%02d\r\n",(carac.datos)->hora.GetHour(),
(carac.datos)->hora.GetMinute());
fprintf(archivo,"Duracion: %d ms\r\n",(carac.datos)->milisegundos);
fprintf(archivo,"Datos validos: %d\r\n", (carac.datos)->validos);
fprintf(destino,"%s\r\n",opciones);
fprintf(destino,"Frecuencia de muestreo teorica: %d Hz\r\n",(carac.datos)->fmuestreo);
fprintf(destino,"Comentarios: \r\n");
fprintf(destino,"Archivos: %s.xx. Las desviaciones son tipicas relativas\n\n",nombre);
fprintf(destino,"-----------\r\n");

for (w=0;w<lim;w++)
{
fprintf(destino,"\nCANAL %d\n",w);
```

```
fprintf(destino,"Medias: media %.0f desviacion %.3f\n",carac.medmed[w],
carac.medvar[w]/carac.medmed[w]);
fprintf(destino,"Desviaciones tipicas: media %.0f desviacion %.3f\n",
carac.varmed[w],carac.varvar[w]/carac.varmed[w]);
fprintf(destino,"Cruces con cero: media %.3f desviacion %.3f\n",
carac.zcmed[w],carac.zcvar[w]/carac.zcmed[w]);
fprintf(destino,"IAV: media %.0f desviacion %.3f\n",carac.iavmed[w],
carac.iavvar[w]/carac.iavmed[w]);
fprintf(destino,"DAMV: media %.0f desviacion %.3f\n",carac.damvmed[w],
carac.damvvar[w]/carac.damvmed[w]);
for (l=0;l<NIVELESH;l++)
fprintf(destino,"HIST%d: media %.3f desviacion %.3f\n",l,carac.hismed[w][l],
carac.hisvar[w][l]);
}
fclose(destino);
return;
}

/***********
RellenaVectores rellena los vectores con los puntos y los pesos.
//lo que pasa es que previamente hay que haber ejecutado el que rellena carac.
************/
void RellenaVectores()
{
int i,j;
for (i=0;i<4;i++)
{
vector[i][0]=carac.medmed[i];
peso[i][0]=carac.medvar[i];
vector[i][1]=carac.varmed[i];
peso[i][1]=carac.varvar[i];
vector[i][2]=carac.zcmed[i];
peso[i][2]=carac.zcvar[i];
vector[i][3]=carac.iavmed[i];
peso[i][3]=carac.iavvar[i];
vector[i][4]=carac.damvmed[i];
peso[i][4]=carac.damvvar[i];
for (j=0;j<NIVELESH;j++)
{
vector[i][5+j]=carac.hismed[i][j];
peso[i][5+j]=carac.hisvar[i][j];
}
}
}


/******
Escribe los vectores en un archivo (1reposo);
*******/
void EscribeVectores(char *nombre, struct Caracteristica *carac2)
{
FILE *archivo;
char cadena[BUFSTR];
int i;

sprintf(cadena,"v%s%d",nombre,sec);
archivo=fopen(cadena,"wb");
i=fwrite(carac2,sizeof(struct Caracteristica),1,archivo);
if (i!=1) {AfxMessageBox("ERROR en escritura",0,NULL);exit(1);}
fclose(archivo);
}
```

```
/******
Lee los vectores en un archivo (1reposo);
por narices que lee el tio los cuatro canales...
...ya los despreciaremos luego...
*******/
void LeeVectores(char *nombre, struct Caracteristica *carac2)
{
FILE *archivo;
char cadena[BUFSTR];
int i;

sprintf(cadena,"%s",nombre);
archivo=fopen(cadena,"rb");
if (archivo!=NULL)
i=fread(carac2,sizeof(struct Caracteristica),1,archivo);
else
i=0;

if (i!=1) {
IniciaVectores(carac2);
AfxMessageBox("AVISO: vector inicializado",1,NULL);
}
else
fclose(archivo);
}


/**********
DistanciaEuclidea da la distancia eclidea de dos vectores.
**********/
float DistanciaEuclidea (float vector1[CANALES][14],float vector2[CANALES][14])
{
//de momento descarto los histogramas.
int i,j;
float parcial;
float distancia[CANALES];
float distanciatotal=0;
for (j=0;j<CANALES;j++)
{
distancia[j]=0;
for (i=0;i<5;i++)
{
parcial=(vector1[j][i]-vector2[j][i])*(vector1[j][i]-vector2[j][i]);
if (parcial<0)
parcial*=-1;
distancia[j]+=parcial;
}
distanciatotal+=distancia[j];
}
return distanciatotal;
}



/**********
DistanciaPonderada da la distancia euclidea ponderada de dos vectores.
**********/
float DistanciaPonderada (float vector1[CANALES][14],float vector2[CANALES][14],float peso[CANALES][14])
{
//de momento descarto los histogramas.
```

```
int i,j;
float parcial;
float distancia[CANALES];
float distanciatotal=0;
for (j=0;j<CANALES;j++)
{
distancia[j]=0;
for (i=0;i<5;i++)
{
if ((peso[j][i])<0.01)
peso[j][i]=(float)0.01; //para evitar los infinitos
parcial=(vector1[j][i]-vector2[j][i])*(vector1[j][i]-vector2[j][i]);
if (parcial<0)
parcial*=-1;
parcial/=peso[j][i];
distancia[j]+=parcial;
}
distanciatotal+=distancia[j];
}
return distanciatotal;
}


/******
Esta funcion quita la media a la señal en los cuatro canales.
********/
void Limpiar(struct Datos *lote)
{
int i;
return; //no funciona bien lo de la media... ya lo arreglare
//elimino primero la media...
int actuo=0;
for (i=4;i<(lote->validos/4)-4;i++)
{
if (fabs(lote->data0[i])>fabs(2*lote->data0[i-1]))
{
lote->data0[i]=Mediana(lote->data0[i-2],lote->data0[i-1],
lote->data0[i],lote->data0[i+1],lote->data0[i+2]);
actuo++;
}
}
}

/*************
Mediana
devuelve la mediana de 5 numeros
***************/
float Mediana(float a, float b, float c, float d, float e)
{
int i,j;//burbuja!!!!
float f[5],g;
f[0]=a;f[1]=b;f[2]=c;f[3]=d;f[4]=e;

for (j=0;j<4;j++)
for (i=1;i<5;i++)
if (f[i]>f[i-1])
{
g=f[i];
f[i]=f[i-1];
f[i-1]=g;
}
return f[2];
```

```
}

/************
Secuencia
Calculo el numero de secuencia y lo autoincremento, en el archivo mio.cfg
***************/
int Secuencia()
{
FILE *co;
co=fopen("mio.cfg","r");
if (co==NULL) //si no existe...
sec=0;
else
{
fscanf(co,"%d",&sec);
sec++;
fclose(co);
}
co=fopen("mio.cfg","w");
fprintf(co,"%d",sec);
fclose(co);
return sec;
}

/*******************
IniciaVectores
*******************/
void IniciaVectores(struct Caracteristica *carac2)
{
carac2=NULL;
}

/*****************
Desbordan
Indica el numero de datos que desbordan al limite dado en limitex.
La funcion parece logico que sea invocada solo tras aplicar filtrado dc
******************/
float Desbordan(short *datos, long longitud, int limitex)
{
int i;
float desbordan=0;
for (i=0;i<longitud;i++)
if (datos[i]>limitex)
desbordan++;
return desbordan;
}

/*****************
GuardarUmbral
Lee el umbral grabado en el archivo de configuracion
******************/
void GuardarUmbral(char *nombre,int umbral)
{
FILE *archivo;
archivo=fopen(nombre,"w");
if (archivo!=NULL)
{
fprintf(archivo,"%d",umbral);
fclose(archivo);
}
}
```

```
/*****************
LeerUmbral
Lee el umbral grabado en el archivo de configuracion
*****************/
int LeerUmbral(char *nombre, int *umbral)
{
FILE *archivo;
archivo=fopen(nombre,"r");
if (archivo!=NULL)
{
fscanf(archivo,"%d",umbral);
fclose(archivo);
if (umbral>0)
return *umbral;
}
return 227;
}


void IniciaTodosVectores()
{
int i;
char nombre[BUFSTR];
for (i=0;i<7;i++)
{
sprintf(nombre,"vector%d",i);
LeeVectores(nombre,&carac3[i]);
}
}

int Clasificador(struct Datos *lote, struct Caracteristica carac4[7], int estados)
{
float feature0,feature1,min;
int i, imin,decision=0;
float distancias[7];
float histograma[CANALES][NIVELESH];

if (!lote->validos)
return 0;


Histograma((short *)lote->data0,lote->validos/4,histograma[0]);
Histograma((short *)lote->data1,lote->validos/4,histograma[1]);
feature0=histograma[0][NIVELESC]*250;
feature1=histograma[1][NIVELESC]*250;
if (feature0<20)
feature0=20;
if (feature1<20)
feature1=20;
if (feature0>250)
feature0=250;
if (feature1>250)
feature1=250;

feature0=0.3*feature0+0.7*oldfeature0;
feature1=0.3*feature1+0.7*oldfeature1;
oldfeature0=feature0;
oldfeature1=feature1;

min=2000000;
imin=0;
for(i=0;i<estados;i++)
```

```
{
distancias[i]=(feature0-carac4[i].hismed[0][NIVELESC]*250)*
(feature0-carac4[i].hismed[0][NIVELESC]*250)+(feature1-carac4[i].hismed[1][NIVELESC]*250)
*(feature1-carac4[i].hismed[1][NIVELESC]*250);

if (distancias[i]<min)
{
min=distancias[i];
imin=i;
}
}
decision=imin;
lote->flotante=Std((short *)lote->data0,lote->validos/4);
return decision;

}

BOOL DobleClic(int datoslargos[2047])
{
int i,j;
int c1,c2,c3;
int lapso=500;
for (i=lapso+20;i<1555;i++)
{
c1=0;c2=0;c3=0;
if (abs(datoslargos[i])>1000)
{
for(j=0;j<20;j++)
{
if (abs(datoslargos[i+j])>800)
c1++;
}
if (c1<4) goto fin; //condicion primera: debe de haber muchos picos seguidos.

 for(j=-lapso;j<0;j++)
{
if (abs(datoslargos[i+j])>300)
c2++;
}
if (c2>50) goto fin; //condicion segunda: debe de haber 120 ms de reposo.
PlaySound("MouseClick",NULL,SND_ASYNC);
for(j=-(lapso+20);j<-lapso;j++)
{
if (abs(datoslargos[i+j])>500)
c3++;
}
if (c3<4) goto fin; //condicion tercera: debe de haber muchos picos seguidos.

return TRUE;
}
fin:
;
}
return FALSE;
}
```

A.4. espaciodlg.cpp

```
// espacioDlg.cpp : implementation file
//

#include "stdafx.h"
#include "espacio.h"
#include "espacioDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#include "vroddon.h"
#include "biomedida4.h"
#include "FijarPuntos.h"
#include <mmsystem.h>


int ClasificadorLocal();
void IniVectores();

extern "C" __declspec(dllimport) void LeeVectores(char *nombre, struct Caracteristica *carac2);
extern "C" __declspec(dllimport) void Histograma(short *datos, long longitud,float niveles[NIVELESH]);
extern "C" __declspec(dllimport) float Std(short *datos, long longitud);
extern "C" __declspec(dllimport) int Decisor (struct Datos *lote, int criterio);
extern "C" __declspec(dllimport) int Clasificador(struct Datos *lote,
struct Caracteristica carac3[7], int estados);
extern "C" __declspec(dllimport) BOOL DobleClic(int datoslargos[2047]);



extern "C" __declspec(dllimport) int Recoger (struct Datos *lote);
extern "C" __declspec(dllimport) void Acabar ();
extern "C" __declspec(dllimport) void Configurar(struct Datos *lote);
extern "C" __declspec(dllimport) void Grabar(struct Datos *lote);



// fuente es la fuente de los datos para la representación.
// fuente=0 Bolas
// fuente=1 Teclado
int fuente=0;
int fijando=0;
int estados=5; //numero de estados
int MinDistancia(int aa,int bb,float *f1,float *f2);
int Minimo(int *a,int elementos);
char textos[20]; /// ojo que invade la region de memoria de otros!!!!!
char textos2[3];
int textoindex=0;
int pintarfondo=0;
int x=30;
int y=30;
int xold=30,yold=30;
int contador=0;
int xviejos[6];
int yviejos[6];
int sancion=0; //cada vea que se elige un vlor critico, hay 5 turnos de sancion.
```

```
const char caracteres[]='A','B','C','D','E','F','G','H','I','J','L','M',
'N','O','P','Q','R','S','T','U','Z','X','X','Y','Y'};


CDC *asa, *asa2, *asa3;
//CPen lapiz,lapiz2;
CPen lapizrojo, lapizverde, lapizazul, lapizamarillo, lapizmorado, lapizcielo, lapiznegro, lapizgris;
CBrush cepillorojo, cepilloverde, cepilloazul, cepilloamarillo,cepillomorado, cepillocielo, cepillonegro, cepillogris;
BOOL suavizado=FALSE;
//amarillo=rojo+verde
//morado=rojo+azul
//cielo=verde+azul



struct Caracteristica carac2;
struct Caracteristica carac3[7];
struct Datos lote;
float f1[7],f2[7];
float radio1[7],radio2[7];
char nombre[100];
void Suavizado();
int datoslargos[2047];


/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
CAboutDlg();

// Dialog Data
//{{AFX_DATA(CAboutDlg)
enum { IDD = IDD_ABOUTBOX };
//}}AFX_DATA

// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CAboutDlg)
protected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
//}}AFX_VIRTUAL

// Implementation
protected:
//{{AFX_MSG(CAboutDlg)
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
//{{AFX_DATA_INIT(CAboutDlg)
//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
CDialog::DoDataExchange(pDX);
//{{AFX_DATA_MAP(CAboutDlg)
//}}AFX_DATA_MAP
```

```
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
//{{AFX_MSG_MAP(CAboutDlg)
// No message handlers
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CEspacioDlg dialog

CEspacioDlg::CEspacioDlg(CWnd* pParent /*=NULL*/)
: CDialog(CEspacioDlg::IDD, pParent)
{
//{{AFX_DATA_INIT(CEspacioDlg)
m_colores = FALSE;
m_rotulo1 = _T("");
m_texto = _T("");
//}}AFX_DATA_INIT
// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
int i;
for (i=0;i<2047;i++)
datoslargos[i]=0;
}

void CEspacioDlg::DoDataExchange(CDataExchange* pDX)
{
CDialog::DoDataExchange(pDX);
//{{AFX_DATA_MAP(CEspacioDlg)
DDX_Control(pDX, IDC_SUAVIZADO, m_csuavizado);
DDX_Control(pDX, IDC_COLORES, m_ccolores);
DDX_Control(pDX, IDSTD, m_cstd);
DDX_Control(pDX, IDHISTO, m_chisto);
DDX_Control(pDX, IDC_TEXTO, m_ctexto);
DDX_Control(pDX, IDC_LETRAS, m_letras);
DDX_Control(pDX, IDC_ROTULO1, m_rotulo2);
DDX_Control(pDX, IDC_CUADRO, m_cuadro);
DDX_Control(pDX, IDC_CAMBIAR, m_cambiarc);
DDX_Control(pDX, IDC_MARCO, m_marco);
DDX_Check(pDX, IDC_COLORES, m_colores);
DDX_Text(pDX, IDC_ROTULO1, m_rotulo1);
DDX_Text(pDX, IDC_TEXTO, m_texto);
DDV_MaxChars(pDX, m_texto, 20);
//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CEspacioDlg, CDialog)
//{{AFX_MSG_MAP(CEspacioDlg)
ON_WM_SYSCOMMAND()
ON_WM_PAINT()
ON_WM_QUERYDRAGICON()
ON_WM_TIMER()
ON_BN_CLICKED(IDHISTO, OnHisto)
ON_BN_CLICKED(IDSTD, OnStd)
ON_BN_CLICKED(IDC_SUAVIZADO, OnSuavizado)
ON_BN_CLICKED(IDC_CAMBIAR, OnCambiar)
ON_BN_CLICKED(IDC_COLORES, OnColores)
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
```

```
// CEspacioDlg message handlers

BOOL CEspacioDlg::OnInitDialog()
{
CDialog::OnInitDialog();
int i;
Configurar(&lote); //Configuramos...
if (lote.huboerror)
exit(0);
if (lote.autonomo)
m_cambiarc.EnableWindow(FALSE);
for(i=0;i<20;i++)
textos[i]=0;
textos2[0]=0;textos2[1]=0;
m_chisto.EnableWindow(FALSE);


// Add "About..." menu item to system menu.

// IDM_ABOUTBOX must be in the system command range.
ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
ASSERT(IDM_ABOUTBOX < 0xF000);

CMenu* pSysMenu = GetSystemMenu(FALSE);
if (pSysMenu != NULL)
{
CString strAboutMenu;
strAboutMenu.LoadString(IDS_ABOUTBOX);
if (!strAboutMenu.IsEmpty())
{
pSysMenu->AppendMenu(MF_SEPARATOR);
pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
}
}

// Set the icon for this dialog.  The framework does this automatically
//  when the application's main window is not a dialog
//SetIcon(m_hIcon, TRUE); // Set big icon
SetIcon(m_hIcon, FALSE); // Set small icon

// TODO: Add extra initialization here
asa2=m_cuadro.GetDC();
asa=m_marco.GetDC();
asa3=m_letras.GetDC();

lapiznegro.CreatePen(PS_SOLID,3,RGB(0,0,0));
lapizrojo.CreatePen(PS_SOLID,1,RGB(200,0,0));
lapizazul.CreatePen(PS_SOLID,1,RGB(0,0,200));
lapizverde.CreatePen(PS_SOLID,1,RGB(0,200,0));
lapizamarillo.CreatePen(PS_SOLID,1,RGB(200,200,0));
lapizmorado.CreatePen(PS_SOLID,1,RGB(200,0,200));
lapizcielo.CreatePen(PS_SOLID,1,RGB(0,200,200));
lapizgris.CreatePen(PS_SOLID,1,RGB(50,50,50));

cepillonegro.CreateSolidBrush(RGB(0,0,0));
cepillorojo.CreateSolidBrush(RGB(200,0,0));
cepilloazul.CreateSolidBrush(RGB(0,0,200));
cepilloverde.CreateSolidBrush(RGB(0,200,0));
cepilloamarillo.CreateSolidBrush(RGB(200,200,0));
cepillomorado.CreateSolidBrush(RGB(200,0,200));
cepillocielo.CreateSolidBrush(RGB(0,200,200));
cepillogris.CreateSolidBrush(RGB(50,50,50));
```

```
asa->SelectObject(&lapizrojo);
asa2->SelectObject(&lapiznegro);

SetTimer(1,180,NULL);
IniVectores();
return TRUE;  // return TRUE  unless you set the focus to a control
}

void CEspacioDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
if ((nID & 0xFFF0) == IDM_ABOUTBOX)
{
CAboutDlg dlgAbout;
dlgAbout.DoModal();
}
else
{
CDialog::OnSysCommand(nID, lParam);
}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CEspacioDlg::OnPaint()
{
int i,aa,bb,c;
if (IsIconic())
{
CPaintDC dc(this); // device context for painting

SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

// Center icon in client rectangle
int cxIcon = GetSystemMetrics(SM_CXICON);
int cyIcon = GetSystemMetrics(SM_CYICON);
CRect rect;
GetClientRect(&rect);
int x = (rect.Width() - cxIcon + 1) / 2;
int y = (rect.Height() - cyIcon + 1) / 2;

// Draw the icon
dc.DrawIcon(x, y, m_hIcon);
}
else
{
if (fuente==0)
{
asa->Rectangle(10,10,260,260);
for (i=0;i<estados;i++)
{
if (i==0) asa->SelectObject(&lapizrojo);
if (i==1) asa->SelectObject(&lapizverde);
if (i==2) asa->SelectObject(&lapizazul);
if (i==3) asa->SelectObject(&lapizmorado);
if (i==4) asa->SelectObject(&lapizcielo);
if (i==5) asa->SelectObject(&lapizamarillo);
if (i==6) asa->SelectObject(&lapizgris);
asa->Ellipse((int)f1[i]-25,(int)f2[i]-25,(int)f1[i]+25,(int)f2[i]+25);
}
```

```
asa->SelectObject(&lapiznegro);
asa->Ellipse(x-2,y-2,x+3,y+3);
asa->SelectObject(&lapizamarillo);
m_letras.ShowWindow(FALSE);
}
if (fuente==1)
{
m_letras.ShowWindow(TRUE);
asa3->SelectObject(&lapiznegro);
asa3->Ellipse(x-2,y-2,x+3,y+3);
asa3->SelectObject(&lapiznegro);
}
asa2->Rectangle(0,0,60,60);
CDialog::OnPaint();
}
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CEspacioDlg::OnQueryDragIcon()
{
return (HCURSOR) m_hIcon;
}

void CEspacioDlg::OnTimer(UINT nIDEvent)
{
// TODO: Add your message handler code here and/or call default
int i,decision;
float histo[NIVELESH];
int auxi,a,b;
RECT recta;
BOOL tope=FALSE, dc=FALSE;
CString cadena;

for(i=0;i<225;i++)
{
datoslargos[i]=datoslargos[i+225];
datoslargos[i+225]=datoslargos[i+450];
datoslargos[i+450]=datoslargos[i+675];
datoslargos[i+675]=datoslargos[i+900];
datoslargos[i+900]=datoslargos[i+1125];
datoslargos[i+1125]=datoslargos[i+1350];
}


Recoger(&lote);
if (fijando)
Grabar(&lote);

if (!fuente) //modo absoluto
{
Histograma((short *)lote.data0,lote.validos/4,histo);
x=histo[4]*1000/4;
Histograma((short *)lote.data1,lote.validos/4,histo);
y=histo[4]*1000/4;
decision=Clasificador(&lote, carac3,estados);
if (x<20) {x=20;}
if (y<20) {y=20;}
if (x>250) {x=250;}
if (y>250) {y=250;}
if (suavizado==TRUE)
```

```
Suavizado();
}
else //modo relativo
{
for(i=0;i<225;i++)
{
if ((lote.validos/4-1)>i)
datoslargos[i+1350]=lote.data0[i];
}
dc=DobleClic(datoslargos);
if (dc)
{
for(i=0;i<1750;i++)
datoslargos[i]=0; //para evitar que haya más doblesclics

Beep(440,200);
a=xviejos[0]-3;b=yviejos[0]-2; //tomamos el valor de x,y hace un segundo
a=a/50;b=b/50;
a=b*5+a;
if ((a<24)&&(caracteres[a]!='X')&&(caracteres[a]!='Y'))
{
sprintf(textos2,"%c\0",caracteres[a]);
textoindex++;
if (textoindex<20)
strcat(textos,textos2);
}
if (caracteres[a]=='X') //borrar
{
textoindex--;
textos[textoindex]=0;
}
if (caracteres[a]=='Y') //aceptar
{
textos[0]=0;
}

m_ctexto.SetWindowText(textos);
}

decision=Clasificador(&lote, carac3,5);
recta.top=y-4;
recta.bottom=y+4;
recta.left=x-4;
recta.right=x+4;

tope=FALSE; //en principio hay que repintar
switch(decision)
{
case 1: //verde
if (sancion==0) //si no hay sancion
x+=PASO;
else
tope=TRUE; //NO se repinta
break;
case 2:
x-=PASO;
sancion=6;
break;
case 3:
if (sancion==0)
y+=PASO;
else
```

```
tope=TRUE;
break;
case 4:
y-=PASO;
sancion=3;
break;
}
if (sancion>0)
sancion--;

if (x<20) {x=20;tope=TRUE;}
if (y<25) {y=25;tope=TRUE;}
if (x>240) {x=240;tope=TRUE;}
if (y>245) {y=245;tope=TRUE;}

}


if ((decision)&&(fuente)&&(!tope))
m_letras.RedrawWindow(&recta,NULL,RDW_INVALIDATE);



switch(decision)
{
case 0:
asa2->SelectObject(&cepillorojo);
break;
case 1:
asa2->SelectObject(&cepilloverde);
break;
case 2:
asa2->SelectObject(&cepilloazul);
break;
case 3:
asa2->SelectObject(&cepillomorado);
break;
case 4:
asa2->SelectObject(&cepillocielo);
break;
case 5:
asa2->SelectObject(&cepilloamarillo);
break;
case 6:
asa2->SelectObject(&cepillogris);
break;

}

yold=y;xold=x;
xviejos[0]=xviejos[1];xviejos[1]=xviejos[2];xviejos[2]=xviejos[3];
xviejos[3]=xviejos[4];xviejos[4]=xviejos[5];xviejos[5]=x;
yviejos[0]=yviejos[1];yviejos[1]=yviejos[2];yviejos[2]=yviejos[3];
yviejos[3]=yviejos[4];yviejos[4]=yviejos[5];yviejos[5]=y;

OnPaint(); //esto es una barrabasada pero funciona
CDialog::OnTimer(nIDEvent);
}

//Pequeño filtro FIR
void Suavizado()
{
```

```
x=(x*0.3+xold*0.7);
y=(y*0.3+yold*0.7);
}


void CEspacioDlg::OnHisto()
{
// TODO: Add your control notification handler code here
estados=5;
m_colores=FALSE;
x=30;y=30;
fuente=0;
m_cambiarc.EnableWindow(TRUE);
m_cstd.EnableWindow(TRUE);
m_chisto.EnableWindow(FALSE);
m_ccolores.EnableWindow(TRUE);
m_csuavizado.EnableWindow(TRUE);

// PlaySound("SystemStart", NULL, SND_SYNC);

}

void CEspacioDlg::OnStd()
{
// PlaySound("SystemHand", NULL, SND_SYNC);

// TODO: Add your control notification handler code here
estados=0;
fuente=1;
m_cambiarc.EnableWindow(FALSE);
m_cstd.EnableWindow(FALSE);
m_chisto.EnableWindow(TRUE);
m_ccolores.EnableWindow(FALSE);
m_csuavizado.EnableWindow(FALSE);
x=100;
y=105;
Invalidate();
}

void CEspacioDlg::OnSuavizado()
{
// TODO: Add your control notification handler code here
if (suavizado)
suavizado=FALSE;
else
suavizado=TRUE;


}

void CEspacioDlg::OnCambiar()
{
// TODO: Add your control notification handler code here
FijarPuntos fija;
fija.DoModal();
// m_aceptar.ShowWindow(SW_SHOW);

}



void IniVectores()
```

```
{
int i;
for(i=0;i<estados;i++)
{
sprintf(nombre,"c:\\victor\\bin\\vector%d",i);
LeeVectores(nombre,&carac3[i]);
f1[i]=carac3[i].hismed[0][NIVELESC]*1000/4;
f2[i]=carac3[i].hismed[1][NIVELESC]*1000/4;
}
}


/********
MinDistancia
*********/
int MinDistancia(int aa,int bb,float *f1,float *f2)
{
int i;
int d[7];
for (i=0;i<estados;i++)
d[i]=(aa-f1[i])*(aa-f1[i])+(bb-f2[i])*(bb-f2[i]);

i=Minimo(d,estados);
return i;
}

/**********
Minimo
***********/
int Minimo(int *a,int elementos)
{
int i, min=10000, imin=0;
for (i=0;i<elementos;i++)
if (a[i]<min)
{
min=a[i];
imin=i;
}

return imin;
}

void CEspacioDlg::OnColores()
{
// TODO: Add your control notification handler code here
if (estados==5)
estados=7;
else
estados=5;
IniVectores();
}

int CEspacioDlg::ClasificadorLocal()
{
int i,imin=0;
float distancia,min;
min=200000;
for (i=0;i<estados;i++)
{
distancia=(x-f1[i])*(x-f1[i])+(y-f2[i])*(y-f2[i]);
if (distancia<min)
{
```

```
min=distancia;
imin=i;
}
}
return imin;
}
```

A.5. entrena.cpp

```
/*************
TITULO: Entrena
AUTOR: Víctor Rodríguez Doncel (vroddon@hotmail.com)
FECHAS: Julio - Octubre 2001
DESCRIPCION:
Este entrenador pide un esfuerzo máximo al usuario y un reposo total a fin de calibrar.
Su salida da un umbral de ruido.
NOTAS:
BUGS:
Ninguno conocido
**************/

// entrena.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "entrena.h"
#include "entrenaDlg.h"
#include "biomedida4.h" //si que es necesario!
#include "vroddon.h" //si que es necesario!
#include "Entrena2.h"
#include <stdio.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//importadas del modulo biomedida4
// con el declspec me evito utilizar el archivo .def
extern "C" __declspec(dllimport) int Recoger (struct Datos *lote);
extern "C" __declspec(dllimport) void Configurar (struct Datos *lote);
extern "C" __declspec(dllimport) void Grabar (struct Datos *lote);
extern "C" __declspec(dllimport) void Acabar ();

//importadas del modulo vroddon
extern "C" __declspec(dllimport) int Analiza(char *nombre, int fragmentos, struct Datos *lote);
extern "C" __declspec(dllimport) void Compactar(char *nombre, int fragmentos);
extern "C" __declspec(dllimport) void Guardar(char *nombre);
extern "C" __declspec(dllimport) void GuardarUmbral(char *nombre);

void Inacabado();
int LOTES;
struct Datos lote;
```

```
/////////////////////////////////////////////////////////////////////////////
// CEntrenaApp

BEGIN_MESSAGE_MAP(CEntrenaApp, CWinApp)
//{{AFX_MSG_MAP(CEntrenaApp)
// NOTE - the ClassWizard will add and remove mapping macros here.
//    DO NOT EDIT what you see in these blocks of generated code!
//}}AFX_MSG
ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CEntrenaApp construction

CEntrenaApp::CEntrenaApp()
{
// TODO: add construction code here,
// Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CEntrenaApp object

CEntrenaApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CEntrenaApp initialization

BOOL CEntrenaApp::InitInstance()
{
// Standard initialization
// If you are not using these features and wish to reduce the size
//  of your final executable, you should remove from the following
//  the specific initialization routines you do not need.

int nResponse;

#ifdef _AFXDLL
Enable3dControls(); // Call this when using MFC in a shared DLL
#else
Enable3dControlsStatic(); // Call this when linking to MFC statically
#endif

Configurar(&lote); // Se invoca al menu de configuracion.
if (lote.huboerror==TRUE)
exit(0);

Entrena2 dlg3;
CEntrenaDlg dlg,dlg2;
//m_pMainWnd = &dlg;  //asigna al hilo una ventanta, cuando esta se cierre, se cierra el hilo
//estaba incluido por el AppWizard, pero yo lo he quitado.


//PASO 1
dlg.m_mensaje="PASO 1: MANTENGA RELAJADO EL BICEPS";
AfxMessageBox("Mantenga relajado el biceps. >Listo?");
strcpy(lote.nomfile, "repos");
lote.secuencia=0;
LOTES=50;
nResponse = dlg.DoModal();
if (nResponse == IDCANCEL)
Inacabado();
```

```
Analiza("repos",50,&lote); //50 lotes de 220ms son 11seg.
Compactar("repos",50);
Guardar("repos");

//GuardarUmbral("umbral.cfg");


//PASO 2
dlg2.m_mensaje="PASO 2: MAXIMO ESFUERZO DEL BICEPS";
AfxMessageBox("Realice el maximo esfuerzo con el biceps. >Listo?");
strcpy(lote.nomfile, "mve");
lote.secuencia=0;
LOTES=20;
nResponse = dlg2.DoModal();
if (nResponse == IDCANCEL)
Inacabado();
Analiza("mve",20,&lote); //es decir, 4.4 segundos...
Compactar("mve",20);
Guardar("mve");

Acabar();
return FALSE;

/*
//PASO 3
AfxMessageBox("Ahora ha de realizar un esfuerzo intermedio");
dlg3.m_mensaje="PASO 3: INTENTE MANTENER LA BARRA EN EL PUNTO";
LOTES=30;
lote.secuencia=0;
strcpy(lote.nomfile, "medio");
dlg3.DoModal();
// if (nResponse == IDCANCEL)
Inacabado();
// Analiza("medio",30,&lote); //
// Acabar();
// exit(0);
// Since the dialog has been closed, return FALSE so that we exit the
//  application, rather than start the application's message pump.
return FALSE;

  */
}

void Inacabado()
{
AfxMessageBox("El entrenamiento no se finalizó");
Acabar();
exit(1);
}
```

A.6. entrena2.cpp

```
// Entrena2.cpp : implementation file
//

#include "stdafx.h"
```

```
#include "entrena.h"
#include "Entrena2.h"
#include "vroddon.h"
#include "biomedida4.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

int contador2=0;


//importadas del modulo biomedida4
extern "C" __declspec(dllimport) int Recoger (struct Datos *lote);
extern "C" __declspec(dllimport) void Configurar (struct Datos *lote);
extern "C" __declspec(dllimport) void Grabar (struct Datos *lote);
extern "C" __declspec(dllimport) void Acabar ();

//importadas del modulo vroddon
extern "C" __declspec(dllimport) int Analiza(char *nombre, int fragmentos, struct Datos *lote);
extern "C" __declspec(dllimport) int Decisor (struct Datos *lote);
extern "C" __declspec(dllimport) void Compactar(char *nombre, int fragmentos);
extern "C" __declspec(dllimport) void Guardar(char *nombre);

extern int LOTES;
extern struct Datos lote;

/////////////////////////////////////////////////////////////////////////////
// Entrena2 dialog


Entrena2::Entrena2(CWnd* pParent /*=NULL*/)
: CDialog(Entrena2::IDD, pParent)
{
//{{AFX_DATA_INIT(Entrena2)
m_mensaje = _T("");
//}}AFX_DATA_INIT
}


void Entrena2::DoDataExchange(CDataExchange* pDX)
{
CDialog::DoDataExchange(pDX);
//{{AFX_DATA_MAP(Entrena2)
DDX_Control(pDX, IDACEPTAR, m_aceptar);
DDX_Control(pDX, IDCANCEL, m_cancelar);
DDX_Control(pDX, IDC_MUESCA2, m_muesca2);
DDX_Control(pDX, IDC_MENSAJE, m_cmensaje);
DDX_Control(pDX, IDC_MUESCA, m_muesca);
DDX_Control(pDX, IDC_PROGRESS1, m_progreso);
DDX_Control(pDX, IDC_ESFUERZO, m_esfuerzo);
DDX_Text(pDX, IDC_MENSAJE, m_mensaje);
//}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(Entrena2, CDialog)
//{{AFX_MSG_MAP(Entrena2)
ON_WM_TIMER()
ON_BN_CLICKED(IDACEPTAR, OnAceptar)
```

```
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// Entrena2 message handlers

void Entrena2::OnTimer(UINT nIDEvent)
{
// TODO: Add your message handler code here and/or call default
CDC *dibujo, *dibujo2;
COLORREF colorito;
CBrush rojo,verde,azul;
float f;

if (nIDEvent==13)
{
rojo.CreateSolidBrush(RGB(80,0,0));
verde.CreateSolidBrush(RGB(160,0,0));
azul.CreateSolidBrush(RGB(240,0,0));
//dibujo->SelectObject(&azul);



dibujo=m_muesca.GetDC();
dibujo2=m_muesca2.GetDC();
//m_muesca.Invalidate();
//m_muesca2.Invalidate();


contador2++;
Recoger(&lote);
Grabar(&lote);
Decisor(&lote);

if(m_progreso.GetPos()==LOTES) // Se finaliza
{
KillTimer(13);
//EndDialog(0);//Esto mata el dialogo de golpe y porrazo. No me vale
m_cancelar.ShowWindow(SW_HIDE);
m_aceptar.ShowWindow(SW_SHOW);
return;
}


f=(lote.flotante/0.8);
if (f>100)
f=100;
if (f<0)
f=0;

if (f<33)
{
dibujo->FillSolidRect(2,8,26,4,RGB(200,0,0));
dibujo2->FillSolidRect(2,8,26,4,RGB(200,0,0));
}
else if (f>66)
{
dibujo->FillSolidRect(2,8,26,4,RGB(0,200,0));
dibujo2->FillSolidRect(2,8,26,4,RGB(0,200,0));

}
else
```

```
{
dibujo->FillSolidRect(2,8,26,4,RGB(0,0,200));
dibujo2->FillSolidRect(2,8,26,4,RGB(0,0,200));
}
//m_esfuerzo.Invalidate(TRUE);

dibujo->DeleteDC();
dibujo2->DeleteDC();
m_progreso.StepIt();
m_esfuerzo.SetPos((int)f);


}
else
CDialog::OnTimer(nIDEvent);
}

//puesta por mi!!!!!!
BOOL Entrena2::OnInitDialog()
{
CDialog::OnInitDialog();
SetTimer(13,(UINT)(195),NULL);  //13 es el identificador del temporizador
m_progreso.SetRange(0,LOTES); //digamos que de 0 a 100
m_progreso.SetStep(1);

m_esfuerzo.SetRange(0,120); //digamos que de 0 a 100
m_esfuerzo.SetStep(1);
m_esfuerzo.SetPos(0);
// TODO: Add extra initialization here

return TRUE;  // return TRUE unless you set the focus to a control
              // EXCEPTION: OCX Property Pages should return FALSE
}

/***
Esto no deberia estar hecho asi, pero bueno....
****/
void Entrena2::OnAceptar()
{
// TODO: Add your control notification handler code here
Analiza("medio",30,&lote); //
Compactar("medio",30);
Guardar("medio");

Acabar();

exit(0);
}
```

A.7. entrenadlg.cpp

```
// entrenaDlg.cpp : implementation file
//
#include "stdafx.h"
#include "entrena.h"
#include "entrenaDlg.h"
```

```
#include "biomedida4.h"

//#define LOTES 50 //el numero de lotes que se han de recoger.

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#define TEMPORIZADOR 13

extern "C" __declspec(dllimport) int Recoger (struct Datos *lote);
extern "C" __declspec(dllimport) void Configurar (struct Datos *lote);
extern "C" __declspec(dllimport) void Grabar (struct Datos *lote);
extern "C" __declspec(dllimport) void Limpiar (struct Datos *lote);
extern "C" __declspec(dllimport) void Acabar ();


extern struct Datos lote;
int contador=0;
extern int LOTES;

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
CAboutDlg();

// Dialog Data
//{{AFX_DATA(CAboutDlg)
enum { IDD = IDD_ABOUTBOX };
//}}AFX_DATA

// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CAboutDlg)
protected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
//}}AFX_VIRTUAL

// Implementation
protected:
//{{AFX_MSG(CAboutDlg)
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
//{{AFX_DATA_INIT(CAboutDlg)
//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
CDialog::DoDataExchange(pDX);
//{{AFX_DATA_MAP(CAboutDlg)
//}}AFX_DATA_MAP
}
```

```
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
//{{AFX_MSG_MAP(CAboutDlg)
// No message handlers
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CEntrenaDlg dialog

CEntrenaDlg::CEntrenaDlg(CWnd* pParent /*=NULL*/)
: CDialog(CEntrenaDlg::IDD, pParent)
{
//{{AFX_DATA_INIT(CEntrenaDlg)
m_mensaje = _T("");
//}}AFX_DATA_INIT
// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CEntrenaDlg::Inicializa(struct Datos *lote3)
{
}


void CEntrenaDlg::DoDataExchange(CDataExchange* pDX)
{
CDialog::DoDataExchange(pDX);
//{{AFX_DATA_MAP(CEntrenaDlg)
DDX_Control(pDX, IDOK, m_aceptar);
DDX_Control(pDX, IDCANCEL, m_cancelar);
DDX_Control(pDX, IDC_PROGRESS1, m_progreso);
DDX_Text(pDX, IDC_MENSAJE, m_mensaje);
//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CEntrenaDlg, CDialog)
//{{AFX_MSG_MAP(CEntrenaDlg)
ON_WM_SYSCOMMAND()
ON_WM_PAINT()
ON_WM_QUERYDRAGICON()
ON_WM_TIMER()
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CEntrenaDlg message handlers

BOOL CEntrenaDlg::OnInitDialog()
{
CDialog::OnInitDialog();

// Add "About..." menu item to system menu.

// IDM_ABOUTBOX must be in the system command range.
ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
ASSERT(IDM_ABOUTBOX < 0xF000);

CMenu* pSysMenu = GetSystemMenu(FALSE);
if (pSysMenu != NULL)
{
CString strAboutMenu;
strAboutMenu.LoadString(IDS_ABOUTBOX);
```

```
if (!strAboutMenu.IsEmpty())
{
pSysMenu->AppendMenu(MF_SEPARATOR);
pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
}
}

// Set the icon for this dialog.  The framework does this automatically
//  when the application's main window is not a dialog
SetIcon(m_hIcon, TRUE); // Set big icon
SetIcon(m_hIcon, FALSE); // Set small icon

// TODO: Add extra initialization here
m_progreso.SetRange(0,LOTES);
m_progreso.SetStep(1);
SetTimer(TEMPORIZADOR,(UINT)(195),NULL);

return TRUE;  // return TRUE  unless you set the focus to a control
}

void CEntrenaDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
if ((nID & 0xFFF0) == IDM_ABOUTBOX)
{
CAboutDlg dlgAbout;
dlgAbout.DoModal();
}
else
{
CDialog::OnSysCommand(nID, lParam);
}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CEntrenaDlg::OnPaint()
{
if (IsIconic())
{
CPaintDC dc(this); // device context for painting

SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

// Center icon in client rectangle
int cxIcon = GetSystemMetrics(SM_CXICON);
int cyIcon = GetSystemMetrics(SM_CYICON);
CRect rect;
GetClientRect(&rect);
int x = (rect.Width() - cxIcon + 1) / 2;
int y = (rect.Height() - cyIcon + 1) / 2;

// Draw the icon
dc.DrawIcon(x, y, m_hIcon);
}
else
{
CDialog::OnPaint();
}
}
```

```
// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CEntrenaDlg::OnQueryDragIcon()
{
return (HCURSOR) m_hIcon;
}



void CEntrenaDlg::OnTimer(UINT nIDEvent)
{
// TODO: Add your message handler code here and/or call default
if (nIDEvent==TEMPORIZADOR)
{
m_progreso.StepIt(); // Avanzo la barra de progreso
contador++;
Recoger(&lote); // Se recoge un lote de muestras
//Limpiar(&lote);

Grabar(&lote); // Se archiva
if(m_progreso.GetPos()==LOTES) // Se finaliza
{
KillTimer(TEMPORIZADOR);
//EndDialog(0);//Esto mata el dialogo de golpe y porrazo. No me vale
m_cancelar.ShowWindow(SW_HIDE);
m_aceptar.ShowWindow(SW_SHOW);
}
}
else{
CDialog::OnTimer(nIDEvent);
}
}
```

A.8. brazoview.cpp (incompleto)

```
unsigned long tiempo=0;
int indice1=-1;
int indice2=-1;

void CBrazoView::OnTimer(UINT nIDEvent)
{
int decision;

if (nIDEvent==ID_TIEMPO) { // es el que nos interesa

Recoger(&lote);
// decision=Decisor(&lote,0);

if (estados==5)
{
decision=Clasificador(&lote,carac4,5);
if (decision==0)
sancion=2;

if (decision==1)
{
```

```
if (sancion==0)
ExtCodo(incr_flex_codo);
}
if (decision==2)
{
CierMano(incr_flex_dedos);
sancion=5;
}
if (decision==3)
{
if (sancion==0)
AbreMano(incr_flex_dedos);
}
if (decision==4)
{
FlexCodo(incr_flex_codo);
sancion=5;
}
if (sancion>0)
sancion--;
}
else if (estados==7)
{
decision=Clasificador(&lote,carac4,7);
switch(decision)
{
case 0:
sancion=4;
break;
case 1:
if (lastdecision!=1)
sancion=3;
if (sancion==0)
ExtCodo(incr_flex_codo);
break;
case 2:
if (lastdecision!=2)
sancion=3;
if (sancion==0)
CierMano(incr_flex_dedos);
break;
case 3:
if (lastdecision!=3)
sancion=3;
if (sancion==0)
AbreMano(incr_flex_dedos);
break;
case 4:
if (lastdecision!=4)
sancion=3;
if (sancion==0)
FlexCodo(incr_flex_codo);
break;
case 5:
if (lastdecision!=5)
sancion=3;
if (sancion==0)
Supinacion(incr_pronacion);
break;
case 6:
if (lastdecision!=6)
sancion=3;
```

```
if (sancion==0)
Pronacion(incr_pronacion);
break;
}
if (sancion>0)
sancion--;
lastdecision=decision;
}
if ((lote.secuencia)<100) //ponemos un limite para no tener mil archivos...
Grabar(&lote);
}

else CView::OnTimer(nIDEvent);
}
```

A.9. biosadview.cpp (incompleto)

```
void CBioSADView::multicanal()
{

int n,i;
int dato;
CDC* pDC = GetDC();
pDC->SetMapMode (MM_HIMETRIC);
struct Datos lote2;
memcpy (&lote2,&lote,sizeof(struct Datos));
Recoger(&lote2);
memcpy (&lote,&lote2,sizeof(struct Datos));
// Limpiar(&lote); ///añadido por mi, quitar si no funciona
// memcpy(&lote,&lote2,sizeof(struct Datos));
m_muestrTemp=lote2.validos/4-4;

// Almacenamos en los cuatro canales la posición de la
// muestra, en la representación grafica.
for ( n = 1; n <= m_muestrTemp; n++)
{
dato=lote2.data0[n];
m_canal1[n] =(int)( -2400 + dato * m_escala);

dato=lote2.data1[n];
if (dato>30)
dato=dato*1;
m_canal2[n] =(int)( -5400 + dato * m_escala);

dato=lote2.data2[n];
m_canal3[n] =(int)( -8000 + dato * m_escala);

dato=lote2.data3[n];
m_canal4[n] =(int)( -11000 + dato * m_escala);
}

int error;

// Borra el trozo que vamos a redibujar.
m_pMemDC->PatBlt (m_paso+27,0,(m_muestrTemp * (int)m_basTiempo)+27,
ALTO,WHITENESS) ;
```

```
m_guardaPaso = m_paso;
BOOL final = TRUE;

// Representamos graficamente los cuatro canales
for (int c= 1; c<=4; c++)
{
// Guardamos el inicio (x) de la representación.
m_paso = m_guardaPaso;
for ( n= 1; n <= m_muestrTemp; n++)
{
// Calcula la posición (x) de la proxima muestra-
m_paso += (int)m_basTiempo;
error = m_paso - ANCHO;

// Si se sale de la superficie de dibujo
// Calcula cuanto se sale, y dibuja este trozo en el comienzo
// de sa zuperficie (a la izquierda)
if (error > 0 )
{
// Borra el principio de la superficie de dibujo en una
// igual a la que no cabia en el final de la superficie.
if (final == TRUE) m_pMemDC->PatBlt (0,0, 27 + error +
((m_muestrTemp - n)* (int)m_basTiempo),ALTO,WHITENESS);
final = FALSE;
switch(c)
{
case 1:
m_pMemDC->SelectStockObject(BLACK_PEN);
m_iniDib[c] = dibujaerror(m_canal1[n],m_iniDib[c],error);
break;
case 2:
m_pMemDC->SelectObject(m_penRojo);
m_iniDib[c] = dibujaerror(m_canal2[n],m_iniDib[c],error);
break;
case 3:
m_pMemDC->SelectObject(m_penAzul);
m_iniDib[c] = dibujaerror(m_canal3[n],m_iniDib[c],error);
break;
case 4:
m_pMemDC->SelectObject(m_penVerde);
m_iniDib[c] = dibujaerror(m_canal4[n],m_iniDib[c],error);
break;
}


}

// Dibuja una linea entre la muestra actual y la anterior.
switch(c)
{
case 1:
m_pMemDC->SelectStockObject(BLACK_PEN);
m_iniDib[c] = dibujar(m_canal1[n],m_iniDib[c]);
break;
case 2:
m_pMemDC->SelectObject(m_penRojo);
m_iniDib[c] = dibujar(m_canal2[n],m_iniDib[c]);
break;
case 3:
m_pMemDC->SelectObject(m_penAzul);
m_iniDib[c] = dibujar(m_canal3[n],m_iniDib[c]);
```

```
break;
case 4:
m_pMemDC->SelectObject(m_penVerde);
m_iniDib[c] = dibujar(m_canal4[n],m_iniDib[c]);
break;
}
}
}
// Copia la superficie de dibujo en la sperficie final borrando lo que
// hubiera antes. Lo hace en dos tramos para mantener a la derecha
// de la pantalla lo ultimo dibujado en la superficie de dibujo.
m_rejillaDC->BitBlt(0,0,ANCHO-m_paso,ALTO,m_pMemDC,m_paso,0,SRCCOPY);
m_rejillaDC->BitBlt((ANCHO-m_paso),0,m_paso,ALTO,m_pMemDC,0,0,SRCCOPY);
//dibuja la cuadricula
dibujaCuadricula();
// Copia El resultado final con cuadricula en la pantalla.
pDC->BitBlt(0,0,ANCHO,ALTO,m_rejillaDC,0,0,SRCCOPY);
}
```