IMPLEMENTACIÓN DE UN ALGORITMO DE RECONOCIMIENTO DE PATRONES ELECTROMIOGRÁFICOS EN UNA TARJETA DE DESARROLLO PARA LA IDENTIFICACIÓN DE LOS MOVIMIENTOS BÁSICOS DE LA MANO

Por:
FABÍAN PEÑA SÁNCHEZ
JULIO CESAR PERALTA DÍAZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERIAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2011

IMPLEMENTACIÓN DE UN ALGORITMO DE RECONOCIMIENTO DE PATRONES ELECTROMIOGRÁFICOS EN UNA TARJETA DE DESARROLLO PARA LA IDENTIFICACIÓN DE LOS MOVIMIENTOS BÁSICOS DE LA MANO

Por:
FABÍAN PEÑA SÁNCHEZ
JULIO CESAR PERALTA DÍAZ

Trabajo de grado presentado como requisito para optar al título de ingeniero electrónico.

Director:
MPE. JAIME GUILLERMO BARRERO PÉREZ

Codirector:
MIE. JHONATAN CAMACHO NAVARRO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERIAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2011

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

Pág.

RESUMEN

TITULO: Implementación de un algoritmo de reconocimiento de patrones electromiográficos en una tarjeta de desarrollo para la identificación de los movimientos básicos de la mano[*].

AUTORES: Fabián Peña Sánchez, Julio Cesar Peralta Días[**].

PALABRAS CLAVES: Electromiográficos (*EMG's*), identificación de movimiento, algoritmo, recursos computacionales y velocidad de ejecución.

En este artículo, se encuentran los resultados de la implementación numérica de un algoritmo de reconocimiento de patrones electromiográficos en la tarjeta de desarrollo MC56F8357EVM de la familia 56F800 de Freescale Semiconductor, para la identificación de los movimientos básicos de la mano (Apertura/Cierre, Extensión/Flexión, Pronación/Supinación y Reposo); la metodología utilizada para dar cumplimiento al objetivo de la presente investigación implementó técnicas de procesamiento digital de señales tales como: Transformada Wavelet Discreta, Análisis de Componentes Principales y Maquinas de Soporte Vectorial.

En igual sentido la inventiva aplicada en la presente investigación, empleo un procedimiento comprendido en tres etapas, las cuales son: envío de señales EMG y modelos propios del estudio de cada paciente, adquisición y procesamiento de las señales EMG por parte del sistema de desarrollo y por último la visualización del movimiento identificado en una LCD 2x16 caracteres; con el procedimiento se logró, primero la visualización del movimiento identificado en pantalla LCD 2x16. Y como aspecto más relevante se dio origen a un algoritmo eficiente en consumo de recursos computacionales con 7.2Mb y 8.7Mb de memoria de programa (flash) y memoria de datos (RAM) respectivamente y velocidad de ejecución de 226 milisegundos, los cuales contribuirán en futuras investigaciones locales sobre el desarrollo de controladores para prótesis de manos, en aras de fortalecer y mejorar la calidad de vida del ser humano.

[*] Proyecto de Grado
[**] Facultad de ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica Electrónica y Telecomunicaciones. Director MPE. Jaime Guillermo Barrero Pérez. Codirector MIE. Jhonatan Camacho Navarro.

ABSTRACT

TITLE: Implementation of a recognition algorithm for electromyographic patterns on a development board for basic hands' movement identification[*].

AUTHORS: Fabián Peña Sánchez, Julio Cesar Peralta Días[**].

KEYWORDS: Electromyographics (*EMG's*), movement identification, algorithm, computational resources, execution speed.

In this article, the results of a numerical implementation of a recognition algorithm of electromyographic patterns on a development board MC56F8357EVM, from the 56F800 Freescale semiconductor family, for basic hands movements (Open/Closure, Extension/Flexion, Pronation/Supination and rest) are found; the methodology used to achieve this investigation's objective implemented digital signal processing techniques such as: Discrete Wavelet Transform, Principal Component Analysis and Support Vector Machines.

On the same idea, the inventiveness applied on the present investigation, applied a three staged procedure which are: EMG signal and particular model for each patient sending, EMG signal acquisition and processing by the development board and, at last, visualization of the identified movement on a 2x16 characters LCD screen. With this procedure the following things were achieved, first identified movement visualization on the 2x16 LCD screen and as the most relevant aspect, an efficient algorithm was created, minimizing computational resources consumption with 7.2 MB 8.7 Mb program memory (flash) and data memory (RAM) respectively and an execution speed of 226 milliseconds, that will contribute on future local research about controller development for hand prosthesis, in order to strengthen and improve life quality on human beings.

[*] Degree Project

[**] Faculty of Physics-Mechanical Engineering. School of Electrical Engineering, Electronics and Telecommunications. Director MPE. Jaime Guillermo Barrero Pérez. Co-Director MIE. Jhonatan Camacho Navarro.

Universidad Industrial de Santander

UIS Ingenierías

REVISTA DE LA FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS

Implementación de un algoritmo de reconocimiento de patrones electromiográficos en una tarjeta de desarrollo para la identificación de los movimientos básicos de la mano.

JULIO CESAR PERALTA DÍAZ
Ingeniero electrónico
Grupo de investigación CEMOS
Escuela de ingenierías Eléctrica, Electrónica y de Telecomunicaciones
Universidad Industrial de Santander
rasecyyyjc@hotmail.com, raseczzzjc@gmail.com

FABIÁN PEÑA SÁNCHEZ
Ingeniero electrónico
Grupo de investigación CEMOS
Escuela de ingenierías Eléctrica, Electrónica y de Telecomunicaciones
Universidad Industrial de Santander
Fabio_tu20@yahoo.es,fabiotu20@gmail.com

JAIME GUILLERMO BARRERO PÉREZ
Magíster en Potencia Eléctrica
Grupo de investigación CEMOS
Escuela de ingenierías Eléctrica, Electrónica y de Telecomunicaciones
Universidad Industrial de Santander
jbarrero@uis.edu.co

RESUMEN

En este artículo, se encuentran los resultados de la implementación numérica de un algoritmo de reconocimiento de patrones electromiográficos en la tarjeta de desarrollo MC56F8357EVM de la familia 56F800 de Freescale Semiconductor, para la identificación de los movimientos básicos de la mano (Apertura/Cierre, Extensión/Flexión, Pronación/Supinación y Reposo); la metodología utilizada para dar cumplimiento al objetivo de la presente investigación implementó técnicas de procesamiento digital de señales tales como: Transformada Wavelet Discreta, Análisis de Componentes Principales y Maquinas de Soporte Vectorial.

PALABRAS CLAVE: Electromiográficos (EMG*'s*), identificación de movimiento, algoritmo, recursos computacionales y velocidad de ejecución.

ABSTRACT

In this article, the results of a numerical implementation of a recognition algorithm of electromyographic patterns on a development board MC56F8357EVM, from the 56F800 Freescale semiconductor family, for basic hands movements (Open/Closure, Extension/Flexion, Pronation/Supination and rest) are found; the methodology used to achieve this investigation's objective implemented digital signal processing techniques such as: Discrete Wavelet Transform, Principal Component Analysis and Support Vector Machines.

KEYWORDS: Electromyographics (EMG's), movement identification, algorithm, computational resources, execution speed.

1. INTRODUCCIÓN

El reconocimiento de patrones de movimiento obtenidos a partir de EMG*'s* (Señales electromiográficas superficiales) comprende un desarrollo de algoritmos empleando diferentes teorías y recursos computacionales. Por esta razón la presente investigación constituye un esfuerzo para la creación e implementación de un algoritmo de reconocimiento de patrones electromiográficos en un sistema de desarrollo qué identifique los movimientos básicos de la mano (Apertura/Cierre, Extensión/Flexión, Pronación/Supinación y Reposo)ilustrados en la Figura 1, permitiendo en un futuro el entrenamiento de extremidades superiores; así como también promover la investigación y desarrollo de controladores de prótesis de fabricación nacional.

Este sistema embebido permitirá que los pacientes con alguna discapacidad de miembro superior no pierdan la habilidad de control sobre sus extremidades. Estudios llevados acabo en empresas como *OttoBock*[1] [1] han revelado grandes dificultades cuando no han pertenecido previamente a programas de entrenamiento para el uso de extremidades mecánicas o uso de prótesis de miembro superior.

El algoritmo mencionado buscará la mejora de controladores de sistemas mecánicos o prótesis, que utilizan señales EMG´s en aspectos como: Velocidad de respuesta, reducción de recursos computacionales y aumento de movimientos identificados.

Movimientos básicos de la mano

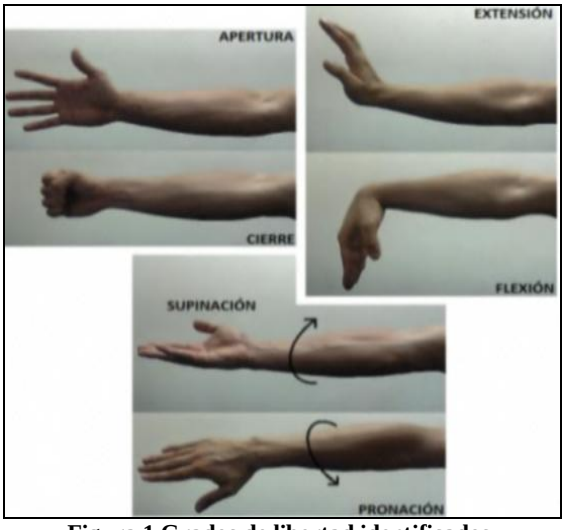

Figura 1.Grados de libertad identificados.
Fuente: Autores

[1]Empresa de fabricación de prótesis

2. ANTECEDENTES

En la actualidad el procesamiento de señales EMG´s es el método más común usado para el control de prótesis de mano activas. Sin embargo, las prótesis comerciales generalmente implementan dos *DoF*[2] (Grados de libertad).

No obstante, en las últimas décadas y especialmente entre los últimos años, muchos esfuerzos se han llevado a cabo con el fin de aplicar un control efectivo con base en técnicas para el procesamiento de señales EMG*'s*.

A partir de los primeros intentos a finales de la década de los 40, varios algoritmos basados en señales EMG*'s* se han desarrollado y utilizado para mejorar la funcionalidad y facilidad de uso de prótesis de mano [2].

En Colombia se presentan diversos casos de discapacidad, según el último reporte del DANE, el 14.6% de 593.546 personas discapacitadas tienen limitaciones de brazos o manos [3]:, y un bajo porcentaje de estos sujetos tienen acceso a ayudas médicas, como prótesis u otros sistema robóticos, esto debido a los altos costos ya que la mayoría de los dispositivos son fabricados por empresas extranjeras como (Otto bock y Delsys[3]), y no por entidades colombianas que favorezcan el precio de adquisición de los mismos.

Por otro lado los pacientes en condiciones de discapacidad de miembro superior desarrollan un fenómeno llamado efecto fantasma. Este fenómeno hace que el paciente sienta su extremidad a pesar que se encuentra ausente. Sin embargo, con el paso del tiempo esta sensación se pierde, debido a que el cerebro asimila la pérdida de dicho miembro.

Así mismo, a pesar del alto porcentaje de discapacitados de miembros superiores y del incremento en investigación y desarrollo en éste campo a nivel regional, en universidades como la Universidad Tecnológica de Pereira [4]: los sistemas de control de prótesis aún no cumplen con los altos requerimientos de desempeño para emular el funcionamiento real del antebrazo como: facilidad de uso, control de velocidad, control de fuerza y alta funcionalidad (cantidad de movimientos identificados).

[2]Por sus siglas en ingles (Degree of freedom)
[3]Empresa de investigación en el área de Bioingeniería.

En la UIS, a través del proyecto "Diseño de una interfaz electrónica para el reconocimiento de patrones EMG para prótesis de mano" [2] se estableció una metodología para identificar movimientos. Así mediante la TWD (Transformada Wavelet Discreta) se realiza el proceso de extracción de características; luego, aplicando ACP (Análisis de Componentes Principales) [6] se reduce el conjunto de características anteriormente obtenido; y finalmente, a través de MSV (Máquinas de Soporte Vectorial) [7] se determina el movimiento realizado.

Es notable decir que con dicha metodología se logran aciertos de identificación superiores al 90%.

DISEÑO DE LA IMPLEMENTACIÓN.

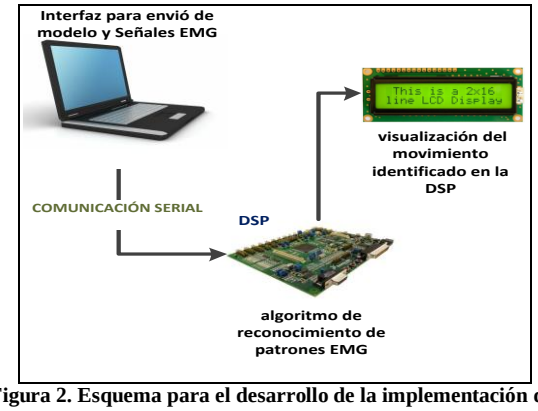

Figura 2. Esquema para el desarrollo de la implementación del algoritmo de reconocimiento de patrones.
Fuente: Autores

3. METODOLOGIA PARA ELDESARROLLO DEL PROYECTO

Luego de una revisión bibliográfica se planteó una metodología descrita en la Figura 2La cual consta de tres partes fundamentales que son:

I. Diseñar una interfaz en Matlab® para suministrar al sistema de desarrollo la señal a analizar, así como los parámetros de procesamiento necesarios para ejecutar técnicas de análisis de señales como la Transformada Wavelet Discreta, Análisis de Componentes Principales y Máquinas de Soporte Vectorial necesarios para la investigación, todo esto en un sistema de desarrollo con *DSC* (Controlador Digital de Señales Motorola Freescale).

II. Adaptar un algoritmo numérico para ser implementado en forma digital sobre el *DSC*MC56F8357EVM, un código de reconocimiento de patrones electromiográficos basado en las técnicas matemáticas de procesamiento de señales mencionadas anteriormente para reconocimiento de los movimientos básicos de la mano.

III. Mediante la utilización de los puertos de propósito general GPIO de la tarjeta de desarrollo visualizar en una *LCD* de 2x16 caracteres el movimiento identificado por la *DSC* mediante el algoritmo implementado.

3.1. Diseño de la interfaz

Para cumplir con el propósito del envío de señales EMG*'s* y modelos pertenecientes a cada paciente se diseñó y creo una interfaz gráfica en *Matlab* (Laboratorio de Matrices) mediante la herramienta GUIDE (Entorno de Desarrollo de Interfaz Gráfica de Usuario) siguiendo los siguientes 4 parámetros de funcionamiento:

a. Selección de señales EMG pertenecientes a un paciente determinado.

b. Así mismo selección de modelos ACP y MSV que contienen los parámetros para una correcta identificación de los movimientos enviados mediante señales EMG*'s*.

c. Seleccionar dos opciones de envío que son, envío de los modelos y envío de las señales EMG*'s*.

d. Permitir seleccionar cual muestra de las señales EMG se enviará mediante comunicación serial.

Teniendo en cuenta que el tipo de datos de las señales EMG y de los modelos es de tipo *double*[4] y están en archivos .m de Matlab, se propone una metodología que codifica estos datos a tipo *CHAR*, para poder ser enviados mediante comunicación serial.

Para él envío de datos tipo *CHAR* mediante el puerto serial se realizó un algoritmo que convirtiera los datos tipo *double* en una cadena de caracteres que contiene el signo, los enteros, el punto decimal y los respectivos decimales descrito en la Figura 3.

Esta cadena de caracteres tiene un tamaño de 15 datos tipo *CHAR* por cada dato *double*, así uniendo todos los datos *CHAR* en un string que contiene la totalidad de la señal EMG. En este punto la señal ya está lista para enviarla por el puerto serial.

[4]Tipo de datos utilizado en programación numérica al igual que enteros y en coma flotante *(int, float, etc.*).

Algoritmo para codificación de datos tipo double a carácter.

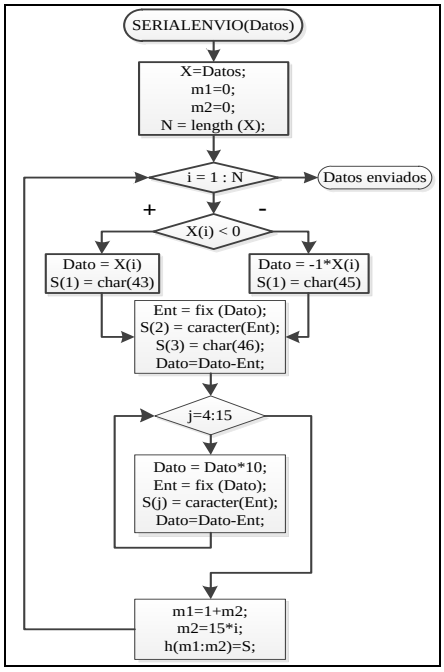

Figura 3. Diagrama de flujo para codificación de modelos y señales EMG's.

Fuente Autores.

3.2. Implementación del algoritmo de recepción y procesamiento del movimiento sobre la tarjeta de desarrollo MC56F8357EVM.

La parte fundamental de la presente investigación fue implementada sobre la tarjeta de desarrollo MC56F8357EVM *freescale* que cuenta con un controlador de señales digitales *DSC* de la misma familia, la cual presenta diversos periféricos como lo son *ADC, PWM, TIMERS, SCI* [5] entre otros muy importantes. También del software CodeWarrior para *DSP* V8.3 de distribución libre.

Tabla 1. Recursos computacionales incorporados en la DSC.

Velocidad	*60MHz/60MIPS*
Bus	*16 bit*
Memoria Interna	*256 KB Program Flash*
	4 KB Program Ram
	8 KB Data Flash
	16 KB Data Ram
	16 KB Boot Flash

Fuente: Autores

Tabla 2. Periféricos incorporados en la DSC.

Periféricos	
PWM	*2 módulos de 6 canales*
ADC	*4 canales de 12 bits*
Sensores	*Temperatura*
Decodificadores	*2 módulos de 4 canales*
Comunicación	*FlexCAN*
	2 interfaz de comunicación serial SCI
	2 interfaz periférico serial SPI
Timer	*4 módulos*
programación	*Un Modulo JTAG*
Pines	*76 líneas de propósito general*

Fuente: Autores.

El software de programación del algoritmo cuenta con un gran número de módulos llamados *BEAN'S* que tienen algoritmos de procesamiento numérico ya sea análisis de señales como FFT o filtro FIR, IIR, operaciones vectoriales, matriciales, módulos de control de motores, comunicaciones entre otros muy usados en ingeniería, que permiten una rápida programación y aprovechamiento del sistema de desarrollo.

Recepción de los datos mediante el puerto serial y módulo SCI.

Para la recepción de datos mediante el puerto serial se configuro uno de los *Bean's*[5] SCI Serial Asíncrono (AsynchoSerial), utilizando la herramienta de recepción de datos tipo *CHAR*.

Mediante un algoritmo sencillo mostrado en la Figura 4y la función (atof[6]), se convierte una cadena de caracteres *ASCII* a datos en coma flotante que permiten el procesamiento de datos planteado para la investigación.

[5]Módulo preconfigurado generado por *Processor Expert* de CodeWarrior.

[6]Función que transforma una cadena de caracteres en coma flotante de 16 bit.

Conversión de CHAR a coma flotante

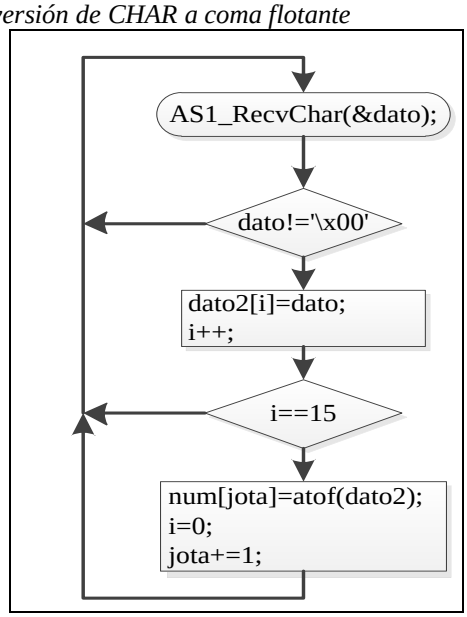

Figura 4. Diagrama de flujo que permite la conversión de caracteres a coma flotante.

Fuente: Autores.

3.2.2 *Implementación del algoritmo de reconocimiento de patrones* EMG*'s.*

Para lograr identificar movimientos mediante un algoritmo de reconocimiento de patrones EMG*'s* la señal es procesada después de su digitalización hasta su identificación siguiendo los lineamientos descritos a continuación.

a. Señal enviada desde el *PC*.

b. Extracción de características principales mediante TWD.

c. Estadísticos aplicados a los coeficientes extraídos.

d. Reducción de la dimensionalidad.

e. Identificación del movimiento.

En la Figura 5 se esbozan los pasos anteriormente mencionados haciendo diferencia entre lo comprendido por el PC y el sistema de desarrollo.

a) Señal enviada desde el PC.

La señal enviada desde el computador se encuentra en la base de datos del proyecto "Diseño de una interfaz electrónica para el reconocimiento de patrones EMG para prótesis de mano" [2] de forma digitalizada mediante los siguientes parámetros.

- Dos canales por señal (Anterior Figura 6 y Posterior Figura 7).
- Frecuencia de muestreo de 1*kHz*.
- Duración por movimiento de 5s.
- Orden de los movimientos adquiridos:
  - Apertura.
  - Cierre.
  - Extensión.
  - Flexión.
  - Pronación.
  - Reposo.
  - Supinación.

Secuencia para el procesamiento e identificación del movimiento

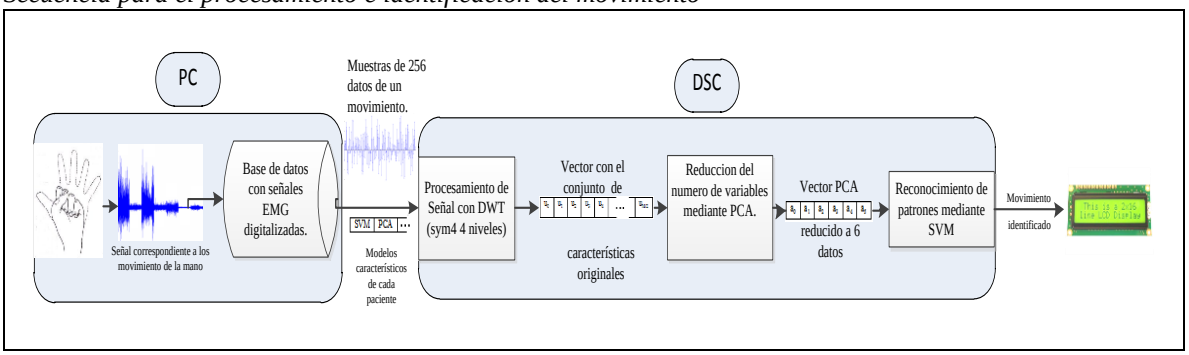

Figura 5. Diagrama general del algoritmo implementado en la tarjeta de desarrollo.

Fuente: Autores.

Canal Anterior

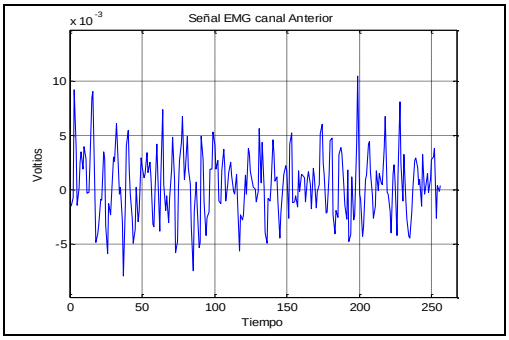

Figura 6. Señal EMG de 256 ms de un movimiento pertenecientes al canal anterior enviado por el PC.
Fuente Autores

Canal Posterior

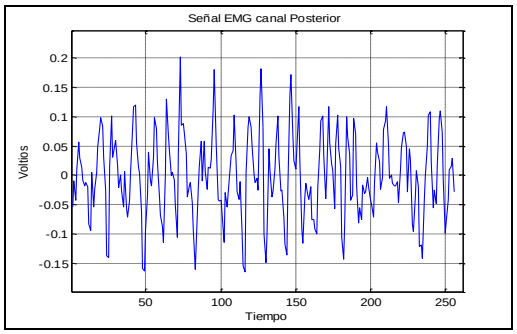

Figura 7. Señal EMG de256 ms de un movimiento pertenecientes al canal posterior enviado por el PC.
Fuente: Autores.

b) *Extracción de características principales mediante* TWD.

En la extracción de características la señal EMG´s es procesada con el fin de tomar las estructuras pertinentes de los datos, mientras se rechaza el ruido y los datos irrelevantes, produciendo el llamado "conjunto de características originales".

Esto se realizó haciendo uso del algoritmo de descomposición piramidal de Mallat [8] mostrado en la Figura 8, el cual aplica la Wavelet madre Symlet 4 (familia Symlet tipo 4) con 4 niveles de resolución.

La descomposición se divide en un filtro pasa bajas y uno pasa altas para obtener los coeficientes de aproximación y detalle respectivamente.

El filtro pasa bajas obtiene los coeficientes de aproximación $A[n]$ mediante la ecuación 1 y la ecuación 2. Donde x[n] tiene un tamaño de 256 datos los cuales según [9] son idóneos para la aplicación de TWD.

$$a[n] = x[n] * g[n] \qquad (1)$$

Luego

$$A[n] = a[\downarrow 2n] \qquad (2)$$

Los coeficientes de detalle $D[n]$ se obtienen de la misma forma cambiando $g[n]$ por $h[n]$, donde $g[n]$ y $h[n]$ son la respuesta al impulso de la función de escala y la función wavelet discreta respectivamente para la wavelet madre symlet4 [10].

Diagrama de descomposición piramidal

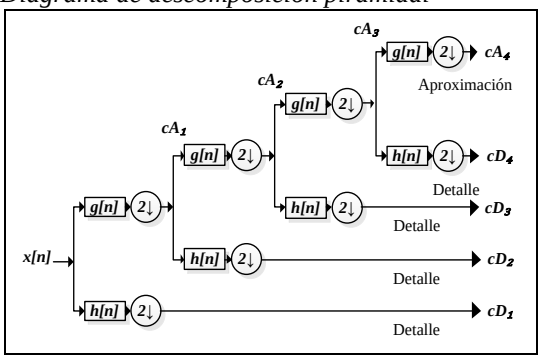

Figura 8. Algoritmo de descomposición piramidal con 4 niveles de resolución.
Fuente: Autores.

Luego de procesar $x[n]$ de 256 datos cada canal con el banco de filtros, los coeficientes de aproximación y detalle quedan organizados como indica la Tabla 2 y mostrados en la Figura 9 y Figura 10, en el mismo orden una respecto de la otra.

Tabla 3. Organización y tamaño de los coeficientes por canal.

Coeficientes	cA_4	cD_4	cD_3	cD_2	cD_1
Tamaño	*22*	*22*	*38*	*96*	*131*

Fuente: Autores.

Coeficientes extraídos canal anterior

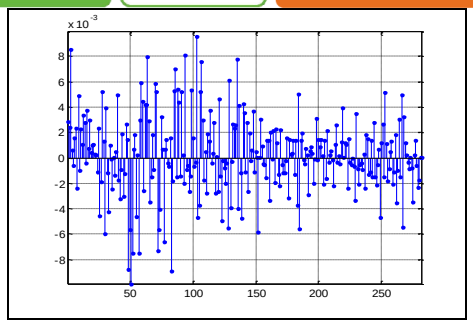

Figura 9. Coeficientes extraídos mediante el algoritmo de descomposición de Mallat del canal anterior.

Fuente: Autores.

Coeficientes extraídos canal posterior

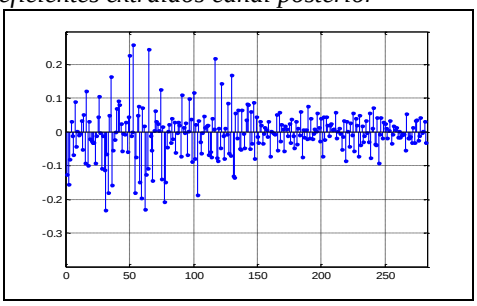

Figura 10. Coeficientes extraídos mediante el algoritmo de descomposición de Mallat del canal posterior.

Fuente: Autores.

Consideraciones de la implementación.

Para la implementación del algoritmo se tomó como modelo de ejecución de la investigación el modelo presentado por [2] ilustrado en la Tabla 3, el cual presento un alto desempeño frente a otros modelos evaluados por los autores para identificación de los movimientos básicos de la mano.

Tabla 4. Modelo Implementado.

Modelo implementado	
parámetro	*descripción*
Wavelet	*Symlet4*
Nivel de Resolución	*4*
Estadísticos calculados	*RMS*
Análisis ACP	*CORRELACION*
Criterios de selección	*ESTANDAR*

Fuente: Autores.

c) Estadísticos aplicados a los coeficientes extraídos.

Los estadísticos aplicados a los coeficientes de aproximación y detalle obtenidos es el valor *RMS* de cada nivel de resolución descrito en la ecuación 3.

$$RMS(coef) = \sqrt{\frac{{c_1}^2 + {c_2}^2 + \cdots {a_{n_i}}^2}{n_i}} \qquad (3)$$

Donde $n_i$es el tamaño del vector de coeficientes de cada nivel de resolución mostrado en la Figura 8, tomando como valor los cinco datos de la Tabla 3.

Como resultado se tiene un vector de 10 datos *RMS* (5 por canal mostrados en la Figura 11 y Figura 12) de los coeficientes extraídos en el paso anterior.

RMS canal anterior

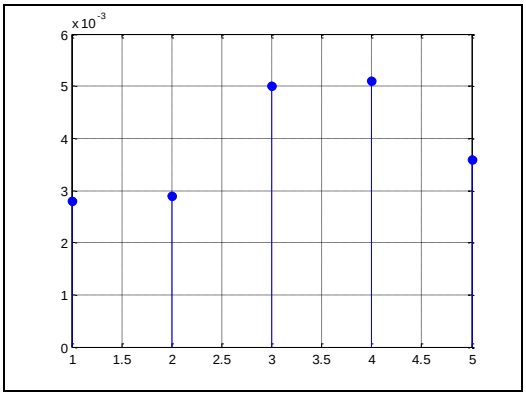

Figura 11. RMS canal anterior.

Fuente Autores.

RMS canal posterior

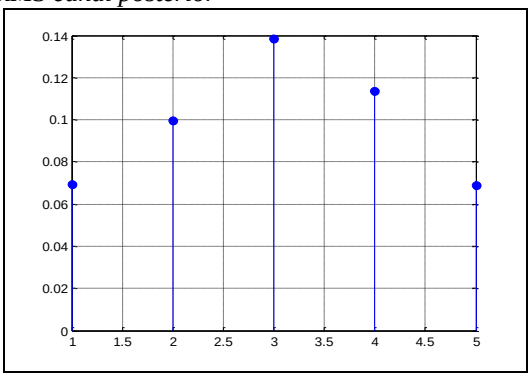

Figura 12. RMS canal posterior.

Fuente Autores.

d) Reducción de la dimensionalidad.

La reducción de dimensionalidad se hace mediante ACP (análisis de componentes principales) criterio correlación que para el caso en estudio es la misma varianza, a razón de la estandarización de los datos originales [6] [2], esta técnica estadística y algebraica ayuda a reducir dimensiones de datos e identificar patrones sobre muestras de estos datos sin perder mucha información.

$$acp' = \frac{x[i]-\overline{x[i]}}{s[i]} \quad (4)$$

$$ACP = acp' \times CP \quad (5)$$

Donde $x[i]$ son los datos provenientes del proceso de descomposición wavelet y aplicación de estadísticos RMS con un tamaño de 10 datos, $\overline{x[i]}$ y $S[i]$ es la media y la varianza de los datos originales proporcionadas en el modelo obtenido en [2].

Como resultado se tiene el vector de datos ajustados (centrados) acp', el cual se multiplica por la matriz $CP(10x6)$ de características principales para obtener los datos finales ACP, dicha matriz es proporcionado en el modelo *ACP* enviado por la interfaz gráfica descrita al inicio de metodología propuesta.

Los datos finales, con reducción de dimensionalidad son los que se muestran el la Figura 13 y que se usaron en el proceso de clasificación con MSV.

ACP *ambos canales*

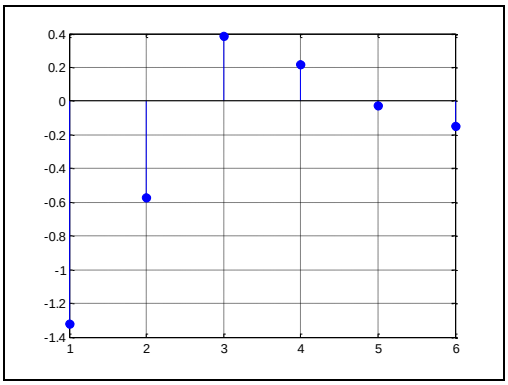

Figura 13. Datos obtenidos del análisis ACP.
Fuente: Autores.

e) Identificación del movimiento.

Intuitivamente, dado un grupo de datos distribuidos en dos clases, una MSV lineal busca un hiperplano de tal manera que la mayor cantidad de puntos de la misma clase queden al mismo lado, mientras se maximiza la distancia de dichas clases al hiperplano.

De acuerdo a Beatnik [11], este hiperplano minimiza el riesgo de clasificaciones erróneas en el grupo tomado para realizar el proceso de validación mencionado en [2].

Para un grupo de entrenamiento compuesto de n pares *atributo-etiqueta* $(\overline{x}_i, y_i)$, mostrado en el siguiente enunciado.

$$\overline{x}_i \in \mathbb{R}^N \text{ Y } y_i \in \{+1, -1\} \text{ con } 1 < i < n$$

Se desea obtener una función f tal que para una entrada en $\mathbb{R}^n$ produzca una salida en $\{\pm 1\}$,

$$f : \mathbb{R}^n \rightarrow \{+1, -1\}$$

Para que así se pueda clasificar un nuevo dato $(\bar{x}, y)$. Téngase en cuenta que $y = f(\bar{x})$para este nuevo dato y es generado con la misma probabilidad de los datos de entrenamiento.

Diagrama de clasificación MSV

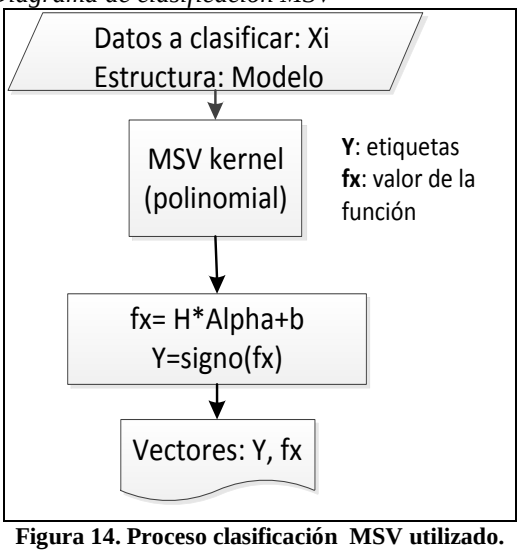

Figura 14. Proceso clasificación MSV utilizado.
Fuente: Autores.

En el proceso de la Figura 14 se clasifican los datos. Este tiene como variables de entrada los datos a clasificar y los respectivos modelos (*b, Arpa, etc.*), las variables de salida Y corresponden a la etiqueta hallada y fx corresponde a la

evaluación numérica de la función de decisión, y así como resultado se toma el valor máximo y la posición que corresponde al movimiento asociado.

Por ejemplo en la Figura 15 se toma el dato más grande que corresponde a 6,228 y se encuentra en la posición (1), donde (1)está asociado al movimiento de *Apertura*.

Clasificación

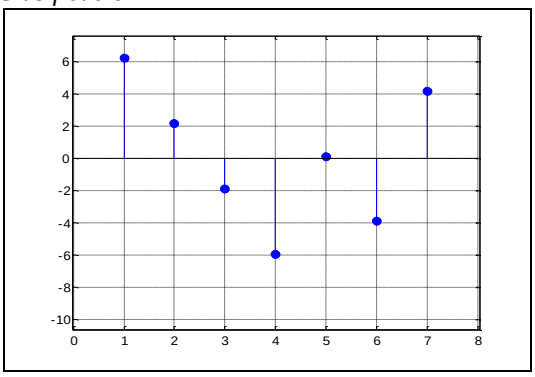

Figura 15. Clasificación del movimiento de apertura.
Fuente: Autores.

4. VISUALIZACIÓN DEL MOVIMIENTO.

Finalmente se visualiza el movimiento identificado en una pantalla LCD 2x16 configurando como salida de datáoslos puertos GPIOB4 al GPIOB7, como control GPIOD0 y GPIOD1.

Para la correcta utilización de la LCD 2x16 en la DSC 56f8357 se utilizó la librería o código de manejo propuesta por [12], la cual es sencilla, rápida y eficiente para controlar dicha pantalla.

La LCD presenta mensajes de funcionamiento que son indicaciones al usuario para la correcta identificación del movimiento y por último se visualiza dicha identificación.

5. RESULTADOS

5.1. Pruebas preliminares

Para las pruebas preliminares se tomó una estructura de validación del procesamiento del algoritmo por partes mostrado en la Figura 16 mediante la *interfazEJECUCION* propuesta por [2] y la adaptación preliminar del algoritmo desarrollada en DevC++ v4.9 de distribución gratuita. De esta forma cada vez se depuraba el algoritmo a medida que avanzaba y a si mismo se hacía más eficientes en aspectos como reducción de variables utilizadas.

En el desarrollo del algoritmo y validación de los datos mencionado anteriormente se estableció el método de reducción de efecto borde por medio de la técnica simetrización [13] la cual realiza una expansión y reflejo de los primeros y últimos 7 datos que se procesan después de cada filtrado en la extracción de características principales con TWD. Ya que si solo se filtraba la señal EMG sin el método de expansión el error por movimiento y a su vez el error total se incrementaba de manera considerable como se muestra en la Figura17 y Figura 18, que presenta la identificación de los movimientos básicos de la mano del paciente número 2 con el método de expansión y sin él.

Diagrama de validación

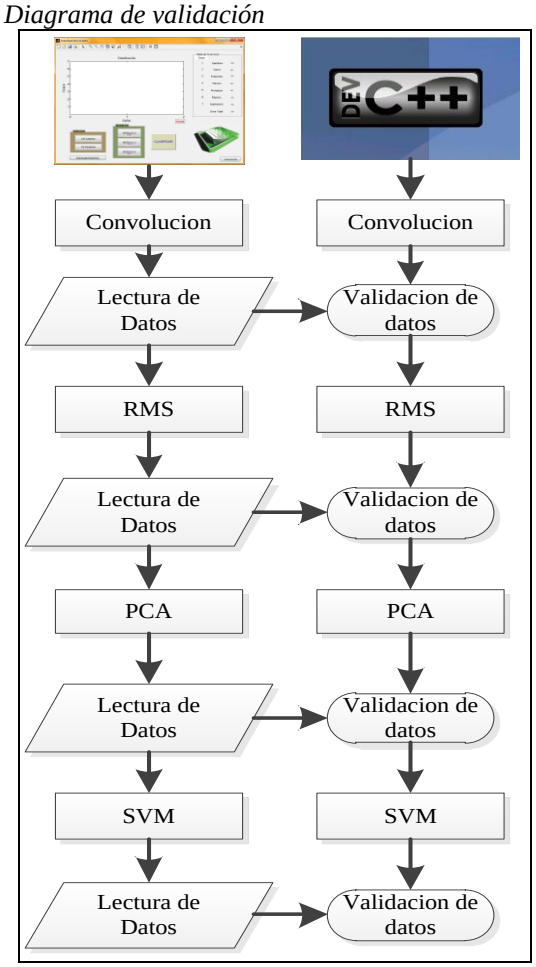

Figura 16. Estructura preliminar de validación del algoritmo.

Fuente: Autores

Al mirar detalladamente los datos arrojados por esta interfaz se observa un incremento del error total de 0.75% a 8.27% para el paciente 2. Este tipo de incremento en el error también está presente en otros pacientes.

InterfazEJECUCION Paciente 2

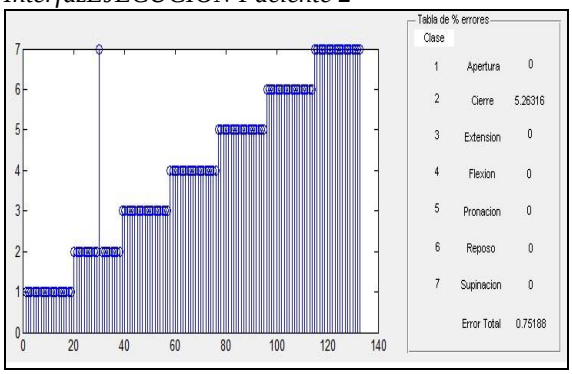

Figura 17. Identificación de los movimientos básicos de la mano con el método de expansión.

Fuente: InterfazEJECUCION [2]

A razón de estas fluctuaciones en el procesamiento de los datos se optó por dejar definido el parámetro citado en la estructura del algoritmo implementado sobre la DSC.

InterfazEJECUCION Paciente 2

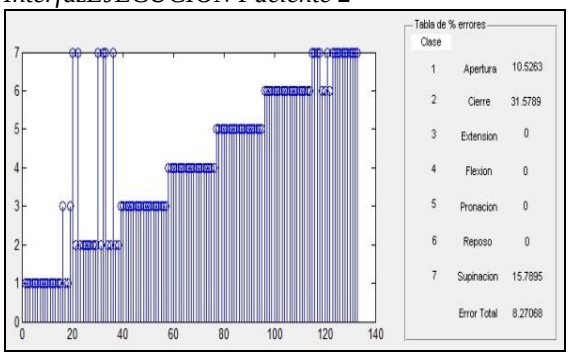

Figura 18. Identificación de los movimientos básicos de la mano sin el método de expansión.

Fuente: InterfazEJECUCION [2]

Cabe aclarar que a pesar de este método presenta mejoras substanciales en determinados pacientes, en otros pacientes arroja minúsculas desventajas respecto al error total de la identificación de los movimiento, que no es relevante en el análisis de todos los casos.

5.2. Envío de modelos y señales EMG*'s mediante puerto serial.*

Durante el envío de los modelos y las señales EMG's mediante puerto serial no se presentaron perdidas substanciales de información por conexión u otros factores, ya que los datos se enviaron con una exactitud de 13 cifras significativas y la mayoría de estos datos tiene una exactitud de 10 cifras, así entregando un porcentaje de perdida de datos transmitidos de 0%.

En la Tabla 5 se presentan algunos de los datos tomados antes del envío en *Matlab* y posteriormente en la tarjeta de desarrollo mediante visualización de variables en *CodeWarrior*.

Tabla 5. Datos enviados Vs Recibidos

Matlab	*DSC*	*Error*
0,00183105	*0,00183105*	*0,00%*
-0,00213623	*-0,0021362*	*0,00%*
0,000839233	*0,00083923*	*0,00%*
0,00129700	*0,00129700*	*0,00%*
0,00137329	*0,00137329*	*0,00%*
0,00190735	*0,00190735*	*0,00%*
0,00190735	*0,00190735*	*0,00%*

Fuente: Autores.

5.3. Recursos computacionales utilizados en la DSC.

Para la medición de los recursos utilizados por la *DSC* en la identificación de los movimientos así como la recepción de los datos a analizar y visualización en pantalla del movimiento identificado, se divide en dos pasos que son.

5.3.1 Asignación de direcciones en memoria *Program Flash* y *Data Flash.*

5.3.2 Mediante de la hoja de datos del *DSC* [5] se tiene la cantidad de memoria en bytes de cada posición de memoria y las diferentes distribuciones de esta.

Para la medición de los recursos se presentan tres aspectos que son:

a. Únicamente la memoria ocupada por el algoritmo sin agregar herramientas del software Codewarrior.

b. El modelo, los datos de un movimiento y un *Bean* de pulsador son incluidos en la programación del algoritmo sobre la *DSC*.

c. El modelo para cada paciente y dos *Bean's* de pulsador se incluye en la programación del algoritmo y se envía mediante comunicación serial los datos de la señal EMG de cada movimiento a la *DSC*.

d. Se envía modelo y señal EMG por medio de comunicación serial y solo se reservan las variables y los *Bean's* de pulsador en la *DSC* que almacenaran las señales y modelos para la identificación.

Los resultados de consumo de recursos de los aspectos mencionados anteriormente se presentan a continuación.

Tabla 6. Memoria utilizada

Tipo de memoria	*Dir. Asignadas*	*Mem Kbytes*
Aspecto a)		
Program Flash	*3701*	*7,3*
Data RAM	*4496*	*8,8*
Aspecto b)		
Program Flash	*3822*	*7,5*
Data RAM	*4500*	*8,8*
Aspecto c)		
Program Flash	*6532*	*12,8*
Data RAM	*5772*	*11,3*
Aspecto d)		
Program Flash	*7110*	*13,9*
Data RAM	*7551*	*14,8*

Fuente: Autores.

5.4. Tiempo de ejecución del algoritmo.

Para la medición del tiempo de ejecución del algoritmo se establece como salida el puerto del sistema de desarrollo GPIOD7 que maneja uno de los led de la tarjeta. Así midiendo con un osciloscopio el tiempo de cambio mostrado en la Figura 19.

Teniendo como resultado el tiempo de ejecución del algoritmo medido en la Tabla 7.

Tabla 7. Tiempo de ejecución del algoritmo.

# de datos de la muestra	*Tiempo de ejecución del algoritmo (ms)*
256	*226*

Fuente: Autores.

Tiempo de ejecución

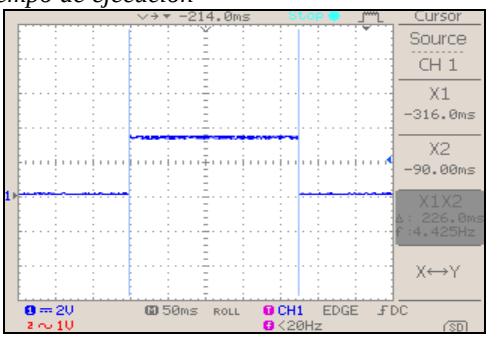

Figura 19. Medición de tiempo de ejecución del algoritmo mediante osciloscopio.

Fuente: Autores.

5.5. Visualizaciones en pantalla.

En la pantalla de 2x16 caracteres se muestran una serie de mensajes correspondientes a indicaciones para el usuario, así como también el movimiento identificado mostrado en la Figura 20.

Movimiento identificado.

Figura 20. Movimiento apertura visualizado en pantalla LCD 2x16.

Fuente: Autores.

6. CONCLUSIONES

Es importante resaltar que el algoritmo implementado tiene la capacidad de recibir los modelos característicos de cada paciente, necesarios para poder procesar cualquier muestra que corresponda a uno de los 7 movimientos de la mano, mencionados anteriormente.

Los tiempos de ejecución del algoritmo para muestras de 256 datos, muestran que se obtuvo un tiempo promedio de identificación de movimientos de $[226\ ms]$, dicho tiempo está dentro de valores aceptables para identificación de movimientos según [9].

Los resultados obtenidos mediante el algoritmo desarrollado fueron satisfactorios. La desviación de los datos fue nula, respecto a los resultados obtenidos en el trabajo de grado Diseño de una interfaz electrónica para el reconocimiento de patrones *EMG* para prótesis de mano, tomado como base y punto de partida en la realización este proyecto.

Se logro obtener datos fiables y aceptables haciendo uso más eficiente del cálculo numérico de la *DWT*, respecto a otros planteamientos que demandaban más recursos, especialmente en la parte de la convolución, que se redujo a las operaciones necesarias que permitieron un menor tamaño del algoritmo, y así hacer factible su implementación en cuanto a tamaño disponible para su programación en el DSC.

Al introducir en el código el método de reducción del efecto borde en la aplicación de la DWT a los datos de la señal electromiográfica, también aporto una mejora de los resultados y reducción del tamaño del código.

Los resultados permiten concluir que el algoritmo implementado para el reconocimiento de patrones electromiográficos en el *DSC*, se puede considerar viable desde el punto de vista de desempeño con un el tiempo de ejecución de 226ms comparado con lo mencionado en [9] y aprovechamiento de recursos del procesador con un tamaño del código de *13,88671875 Kbytes* en memoria de programa y *14,74804688 Kbytes* en memoria de datos, dimensiones muy reducidas a partir de la mejora en la convolución realizada en la DWT.

A través de dispositivos como los *DSC* se demostró en este trabajo que se pueden elaborar aplicaciones en donde sea necesaria una considerable carga computacional para ejecutar algoritmos de procesado de señal como la *DWT* sin la utilización de una máquina robusta como un computador, y a partir de una adecuada programación, construir equipos con visualización de eventos en pantalla *LCD*.

Las aplicaciones presentadas por Codewarrior a través de su herramienta *Processor Expert* y en específico los "*Bean's*", son de ayuda para aprender a utilizar el *DSC* como tal, es decir, con los debidos pasos para realizar una correcta configuración de un módulo y las funciones que emplea, además el entorno gráfico para la configuración de cada *Bean*, hacen cómoda la programación del *DSC*.

Con referencia a trabajos realizados en Colombia en arquitecturas embebidas es importante que la ubicación de aplicaciones en tiempo real esté de la mano del diseño de hardware, lo cual será una mejora a muchos trabajos de investigación realizados, en aras de fortalecer el desarrollo y fabricación de dispositivos a nivel nacional.

7. REFERENCIAS

[1] Otto Bock, enlace: http://www.ottobock.com/cps/rde/xchg/ob_com_es/hs.xsl/5064.html

[2] CAMACHO, Jhonatan y LEON, Fabián; Diseño de una interfaz electrónica para el reconocimiento de patrones electromiográficos para prótesis de mano, Pregrado, Escuela de Ingeniería Eléctrica, Electrónica y Telecomunicaciones, Universidad Industrial de Santander, Bucaramanga 2008.

[3] DANE, "Último censo", enlace: http://www.dane.gov.co/censo/files/boletines/discapacidad.pdf

[4] PINZÓN, Rubén y MORALES, Diego y GRISALES, Víctor, "Caracterización de señales electromiográficas para la discriminación de seis movimientos de la mano", Universidad Tecnológica de Pereira, 2009. Enlace:

[5] Hoja de datos del *DSC* MC56F8357: http://cache.freescale.com/files/*DSC*/doc/data_sheet/MC56F8357.pdf

[6] LINDSAY I. SMITH. A tutorial on Principal Components Analysis. 2002.

[7] CAMPOS E.; SUÁREZ A. Clasificación automática de perturbaciones de señales de tensión o corriente utilizando máquinas de soporte vectorial (*MSV*). Universidad Industrial de Santander, 2007.

[8] Y. Sheng, "The Transforms and Applications Handbook". CRC Press, 1996.

[9] ENGLENHART K.; PARKER P. A Wavelet-based Continuos Classification Scheme for multifunction Myoelectric Control. IEEE transactions on bidirectional engineering, vol. 48, no. 3, March 2001.

[10] WAVELET-BROWSER/Wavelet Symlets 4(sym4)/Properties http://wavelets.pybytes.com/wavelet/sym4/

[11] VLADIMIR VAPNIK. The Nature of Statistical Learning Theory. Springer, NY, 1995.

[12] Martinez, Adriana y Arenas, Edgar; Generador de señales para mediciones de impedancia electroquímica basado en DSP. Electrónica y Telecomunicaciones, Universidad Industrial de Santander, Bucaramanga 2006.

[13] Strang, G.; T. Nguyen (1996), Wavelets and filter banks, Wellesley- Cambridge Press.

ANEXOS

ANEXO A Manual de usuario.

1. Descripción general.

La presente investigación permite mediante un equipo de cómputo y un sistema de procesamiento de señales la identificación de los movimientos básicos de la mano. A partir de señales EGM digitalizadas y datos de modelos con características propias de pacientes sanos, también digitalizadas y procesadas mediante un sistema de desarrollo (Tarjeta de desarrollo *Motorola Freescale MC56F8357EVMUM*) que identifica el movimiento haciendo uso de técnicas matemáticas mencionadas en el artículo.

1.1. *Aspectos importantes*.

La investigación se desarrolló para ser ejecutada desde dos dispositivos electrónicos a saber:

a) Equipo de cómputo que cuente con hardware de conexión serial y paralelo (COM y LPT), además de tener software *Matlab 7* o superior para la ejecución de la interfaz de envío de los datos para ser procesados, también el software CodeWarrior (CW_DSC56800E_8.3_Evaluation), en el cual se implementa el algoritmo y posteriormente se programa el sistema de desarrollo.
b) Tarjeta de desarrollo *Motorola Freescale MC56F8357EVMUM* que cuenta con puerto JTAG, RS232 entre otros muy importantes, la tarjeta mencionada cuenta con un DSC (Controlador Digital de Señales) como

dispositivo de procesamiento central a una frecuencia de 60 MIPS o 60MHz.

2. Conexiones.

Para la puesta en funcionamiento de la investigación se conecta la tarjeta de desarrollo al equipo de cómputo mediante un cable de conexión paralela (Cable paralelo, para programación y visualización de variables y datos mediante software CodeWarrior), otro de conexión seriada (Cable serial para envío de señales EMG y modelos propios de cada paciente) y el cable de alimentación de energía del sistema de desarrollo *Motorola Freescale*.

3. Funcionamiento

Para la puesta en funcionamiento del proyecto se siguen los pasos mostrados a continuación con las respectivas acotaciones misionadas en la sección 1.1 del presente manual.

3.1. *Instalación en Matlab.*

a) Ejecutar MATLAB 7 o superior.
b) Copiar la carpeta TWD_EMG_DSC a un directorio especificado por el usuario.
c) Adicionar al path de MATLAB la carpeta TWD_EMG_DSC y subcarpetas.

3.2. *Ejecución en Matlab.*

a) Ahora se procede a correr el archivo llamado InterfazEnvioVALIDACION.m.
b) Se selecciona el archivo de la señal EMG del canal anterior picando en la pestaña *Canal Anterior* que se encuentra en la interfaz anteriormente

mencionada y el archivo se encuentra en una subcarpeta de un paciente estudiado.

c) Al igual que el ítem anterior se seleccionan el *canal posterior*, el *Modelo PCA* Y *Modelo MSV.*

3.3. Programación

a) Abrimos CodeWarrior para DSC.
b) Abrimos el proyecto llamado TWD_EMG_DSC.
c) Compilamos el proyecto y luego procedemos a programarlo.
d) Una vez programado procedemos a dar *PLAY* mediante la interfaz de depuración de CodeWarrior.

3.4. Ejecución completa del sistema de identificación de los movimientos básicos de la mano.

Una vez corrido el algoritmo en el sistema de desarrollo se presentaran unas instrucciones en pantalla, las cuales indicaran los pasos a seguir.

a) Los primeros textos mostrados en la pantalla hacen referencia al nombre del proyecto, institución y autores del proyecto, ilustradas en la Figura A1.

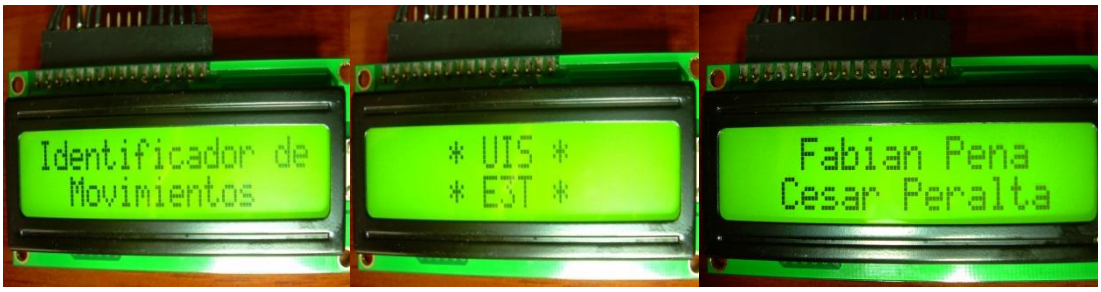

Figura A 1. Presentación del proyecto.

Fuente: Autores.

b) El primer mensaje informativo indica que se debe enviar el modelo desde la interfaz en *Matlab,* lo cual procedemos a hacer.

c) Luego de esperar a la transferencia de los datos se procede a cargar el modelo en las variables asignadas dentro del código por los autores presionando el pulsador *IRQB* del sistema de desarrollo.

Los pasos mencionados en los ítems anteriores se muestran en la Figura A2.

Figura A 2. Envío de modelo.

Fuente: Autores.

d) Inmediatamente la indicación informa que se debe enviar alguna señal mediante la interfaz en *Matlab*.

NOTA: La señal enviada desde *Matlab* para ser procesada se debe pertenecer al mismo paciente del cual se enviaron los modelos ACP y *MSV.* Estas señales se deben seleccionar de un grupo de 133 muestras que contienes lo 7 movimientos a identificar, donde las primeras 19 muestras contienen el movimiento apertura, las segundas 19 muestras contienen el movimiento cierre y así hasta terminar (Se puede enviar cualquiera de estas muestras).

e) Después de enviada alguna señal EMG se procede a la parte más importante de la investigación que es la identificación, esta se hace presionando el pulsador *IRQA* del sistema de desarrollo. Ver Figura A3.

f) Por último se visualiza en pantalla el movimiento identificado por el sistema de desarrollo. Ver Figura A3.

Figura A 3. Envío de señales e identificación.

Fuente: Autores.

De la misma forma que se envío la señal EMG de algún movimiento se puede enviar la de otro sin interrumpir el mecanismo o las conexiones de la prueba. Así mismo se puede enviar de nuevo otro modelo y muestras de un nuevo paciente que se quiera analizar.

ANEXO B. Configuración de los módulos del DSC.

1. Descripción general.

Para la implementación del algoritmo de reconocimiento de patrones electromiográficos se utilizó la herramienta *Processor Expert* el cual entrega módulos preconfigurados de distintos dispositivos con los que cuenta el sistema de desarrollo, como lo son: SCI, SPI, TIMERS, BUSS de memoria ETC.

2. Módulos utilizados.

   El orden es de acuerdo como parece en el proyecto implementado en CodeWarrior.

   2.1. 53F8357EVM_Button_IRQA[Button], ver Figura B1.

   a) Nombre: Identifica

   b) Función: Se encarga de ejecutar el código implementado mediante una sentencia if que detecta la pulsación y ejecuta el mismo.

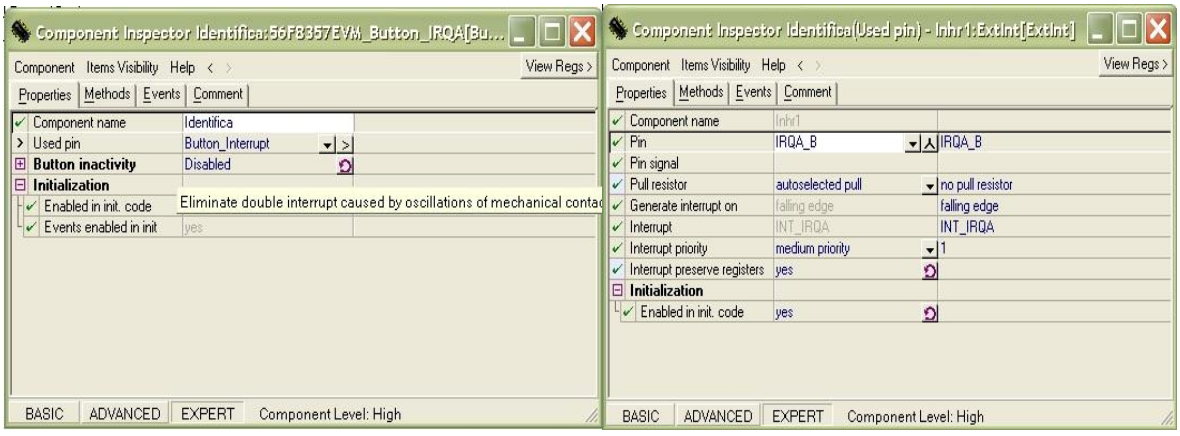

Figura B 1. Configuración del *Bean* 53F8357EVM_Button_IRQA[Button],

Fuente: Autores.

2.2. 53F8357EVM_Button_IRQB[Button], ver Figura B2.

a) Nombre: CargaMODELO

b) Función: Se encarga de actualizar las variables reservadas de los parámetros de procesamiento después de la transferencia de estos mediante una sentencia *if* que detecta la pulsación y actualiza la misma.

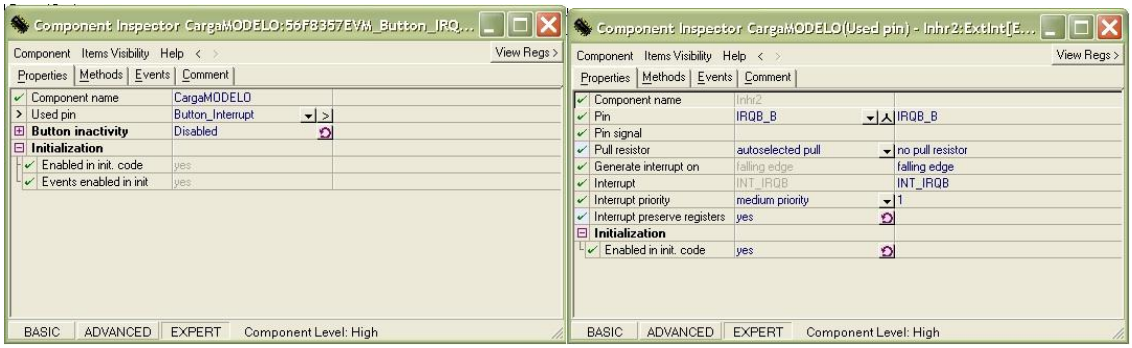

Figura B 2. Configuración del*Bean*53F8357EVM_Button_IRQB[Button]

Fuente: Autores.

2.3. AsynchroSerial, ver Figura B3

a) Nombre: RxDATO

b) Función: Se encarga de la recepción de los caracteres ASCII mediante comunicación serial.

2.4. BitsIO, ver Figura B4.

a) Nombre: DATA

b) Función: Bus de datos que comunica el sistema de desarrollo con la LCD 2x16, la cual mostrara diferentes textos instructivos.

2.5. BitIO, ver Figura B4.

a) Nombre: RS

b) Función: Bit de configuración Registro/Instrucción que comunica el sistema de desarrollo con la LCD.

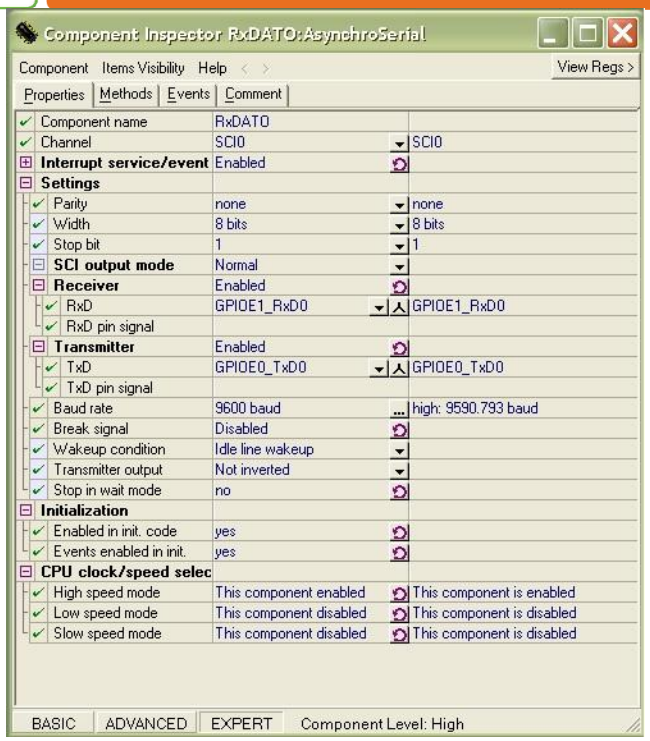

Figura B 3. Configuración del *Bean* AsynchroSerial

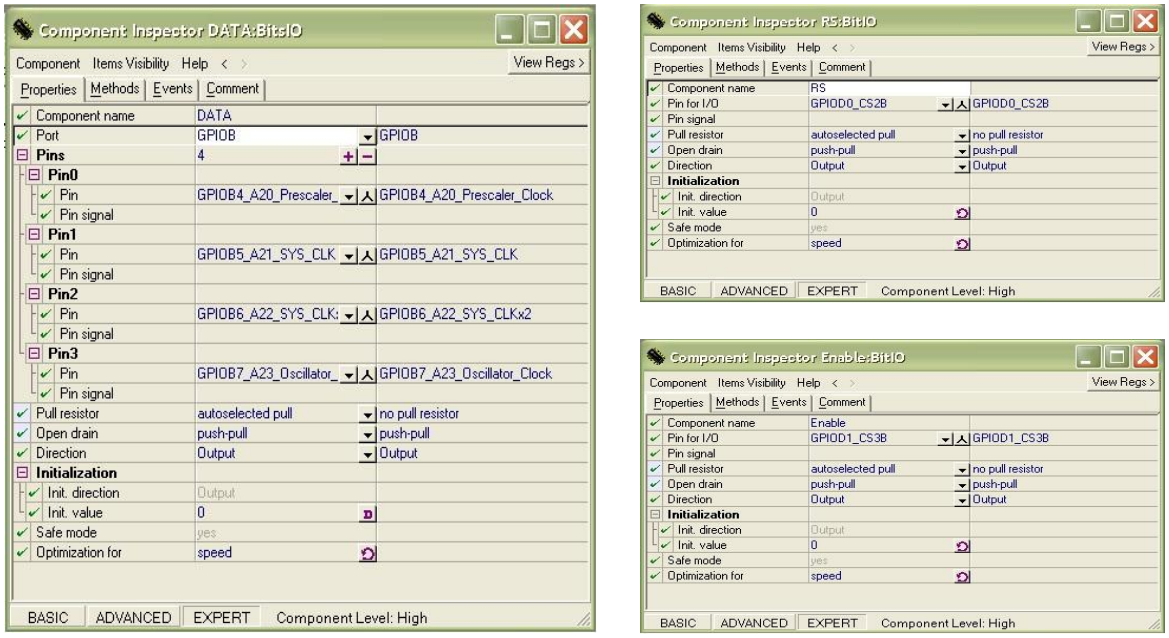

Figura B 4. Configuración de los *Bean* de conexión a la LCD

Fuente: Autores.

1.1. BitIO, ver Figura B4.

a) Nombre: Enable

b) Función: Bit que habilita el funcionamiento de la LCD.

1.2. Led, ver Figura B5.

a) Nombre: CargaMOD

b) Función: Led configurado con el fin de leer mediante osciloscopio el tiempo de procesamiento del código, en este caso el tiempo que demora en actualizar los datos de los modelos.

1.3. Led, ver Figura B5.

a) Nombre: IDENT

c) Función: Led configurado con el fin de leer mediante osciloscopio el tiempo de procesamiento del código, en este caso el tiempo que demora el algoritmo principal para la identificación de los movimientos.

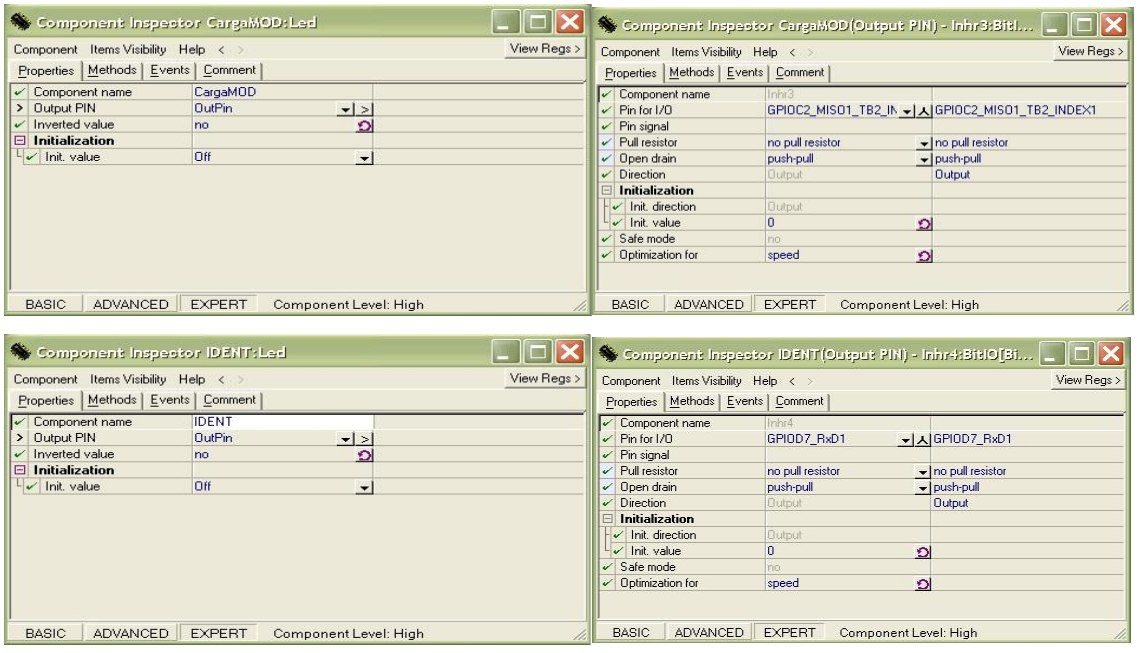

Figura B 5. Configuración de los *Bean* conectados a los led de test.

Fuente: Autores.

ANEXO C. Descripción del algoritmo de identificación.

1. Descripción general

El diagrama de flujo de la Figura C1 Representa el código planteado e implementado en el DSC. Se observa una estructura tipo for y la forma de simbolizarla indica un ciclo “infinito”, esto se debe a la programación de este tipo de dispositivos los cuales tienen un ciclo “infinito” el cual siempre retorna al inicio del algoritmo.

En el diagrama de flujo de la Figura C2después de inicializar variables, arreglos y periféricos necesarios para poder ejecutar el código que realizar la identificación de los movimientos, se pasa al ciclo infinito en donde se reciben y cargan los datos correspondientes a modelos (ACP, MSV, etc.) y señal EMG correspondiente a 2 canales por paciente, posteriormente se hace una expansión de la señal electromiográfica empleando el método reducción de efecto borde de tipo simetrización, con el fin de enfatizar información atenuada durante el filtrado[6].

1.1. Extracción de características

En este caso se reflejan los 7 primeros y 7 últimos datos de la señal (La cantidad de datos reflejados depende del tamaño de la señal) [6] para luego aplicar TWDsymlet4 de 4 niveles (después de cada nivel se aplica efecto borde), se extraen los coeficientes de detalle de cada nivel y los coeficientes de aproximación del ultimo nivel, todos los datos se almacenan en un vector que contiene 282 posiciones.

Lo anterior se realiza para la señal de cada canal (anterior y posterior), a continuación se llena un vector de 10 posiciones con el cálculo del valor eficaz de los 282 coeficientes almacenados, por cada canal se obtienen 5 datos.

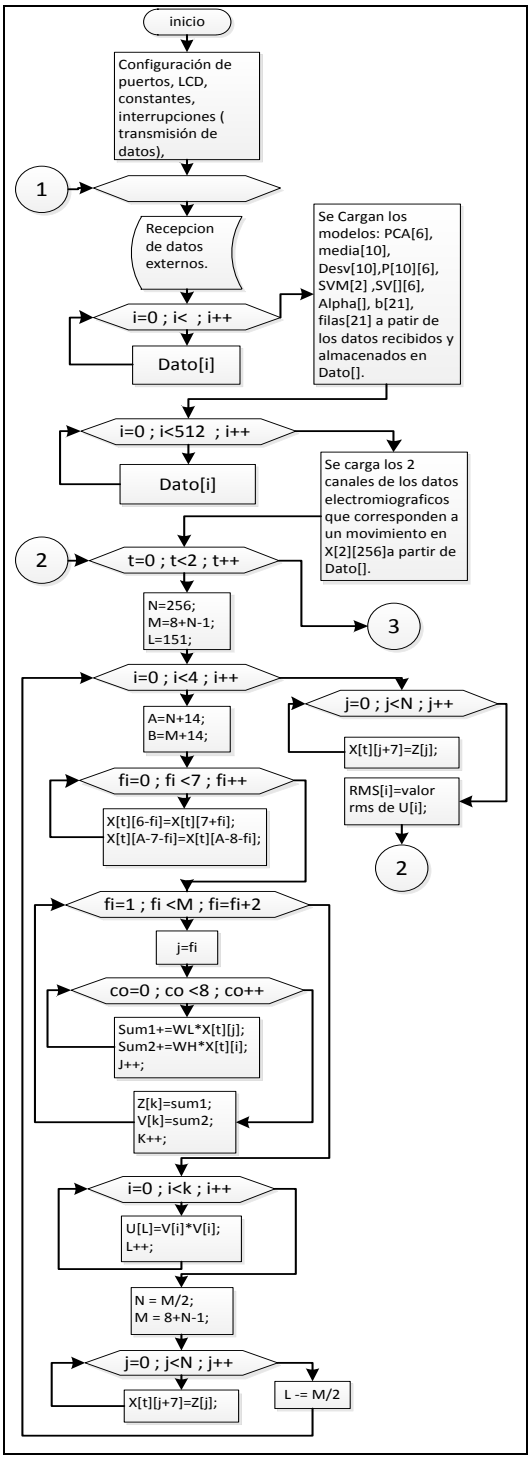

Figura C 1. Algoritmo implementado.
Fuente: Autores.

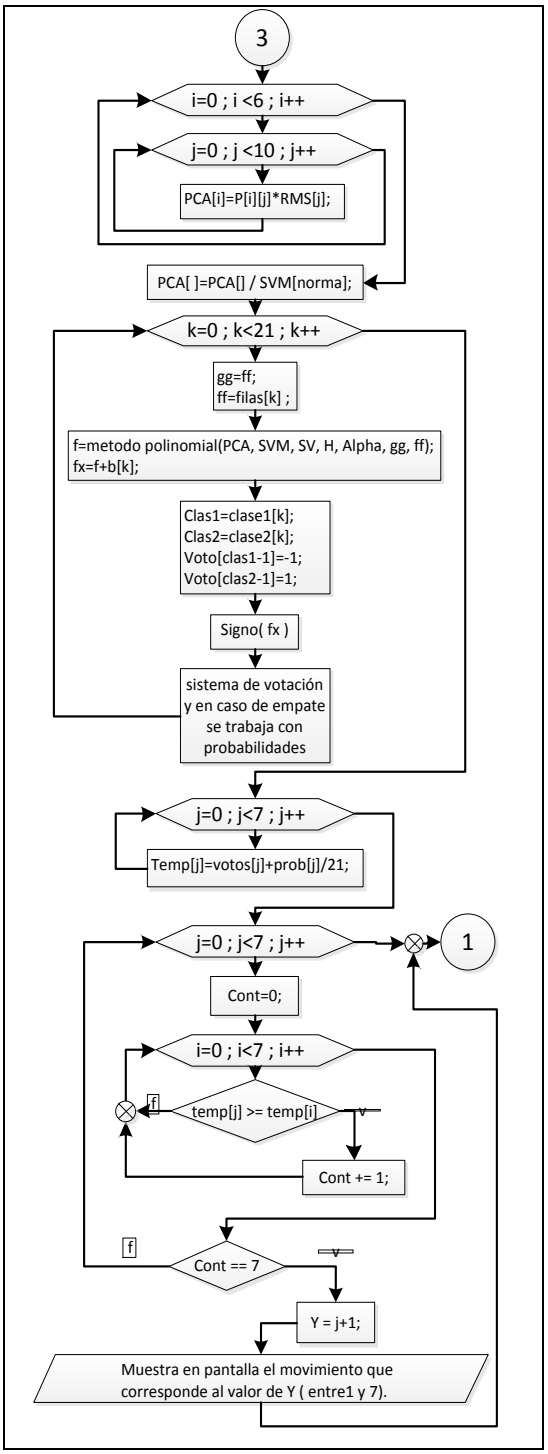

Figura C 2. Algoritmo implementado.
Fuente: Autores.

1.2. Reducción de dimensionalidad

A partir del vector de 10 datos RMS y el modelo de componentes principales cargado anteriormente se reduce la dimensionalidad de las características extraídas con el método ACP, de donde se obtienen un vector ACP con 6 datos.

1.3. Reconocimiento de patrones

Luego se aplica MSV utilizando criterio polynomial para luego hacer el reconocimiento del movimiento al cual corresponden los datos de la señal electromiografía ingresada, mediante técnicas de votación, y en caso de empate por probabilidad es identificado. Este proceso arroja un vector que contiene 7 datos, en el cual, la posición en que se encuentre el de mayor valor corresponde al número del movimiento identificado.

Posteriormente se hace la correspondencia del número identificado con el movimiento y con la previa configuración de los puertos GPIO conectados a la pantalla 2x16 se muestra en esta el movimiento al cual corresponde.

Para terminar, el programa vuelve a su punto inicial si el usuario lo desea puede enviar más datos del paciente correspondientes a otro movimiento para ser identificados, o enviar nuevamente el modelo y señal, si va a hacer reconocimiento a un nuevo paciente.

ANEXO D. Algunos *Bean's* de la herramienta *Processor Expert* del software CodeWarrior para DSP´s.

El software CodeWarrior mediante su herramienta *Processor Expert* permite crear módulos completos para utilización de los periféricos así como también módulos que realizan operaciones matriciales, trigonométricas, vectoriales, de comunicación entre otras. En la Tabla D1 se mencionan algunas de ellas.

Tabla D 1. Descripción de los Beans de la herramienta Processor Expert del Software Codewarrior.

Módulos	Descripción
Terminal	• Comunicación con terminal usando protocolo ANSI
Control	• Encapsula algoritmos de control estándar como PI PID, respuesta a entradas rampa, calculo de velocidad de acuerdo a diferencia de posición y tiempo.
Datos	• Interfaz de dos colores (*BPN*). • Interfaz para las imágenes en color RGB que se transforman en imágenes de color en el formato BMP. • Acceso a PC para leer o escribir un archivo al mismo tiempo, Inclusión de archivos externos como sonidos, base de datos etc. y los convierte a código binario para el ensamblador.

Funciones para DSP y bibliotecas matemáticas	• Operaciones básicas con matrices. • Algoritmos de procesamiento de señales como FFT, filtros FIR e IIR, correlación relacionadas. • Operaciones matemáticas básicas para tipo de datos FRAC de 16 bit y 32 bit. • Operaciones matriciales básicas para datos FRAC16. • Funciones trigonométricas para datos FRAC16. • Operaciones con vectores para datos FRAC16 como producto punto, cruz entre otros.
Administrador de Memoria	• Permite agregar bloques de memoria interna y externa, y redistribución de la misma.
Bibliotecas de Módems	• Llama y recibe llamadas de otras aplicaciones a distintas velocidades en bps
Control de Motores	• Contiene librerías para control de motores. • Control de conmutación para sensores de efecto hall. • Control para detección de cruces por cero. • Control de freno, Control en espacio de estados ETC.
Sistema Operativo	• MicroC
Bibliotecas de seguridad	• Cifrado de datos
Bibliotecas telefónicas	• Detección de tonos en procesos de llamadas. • Generación de tonos comunes. • Detección de actividad de vos. Entre otros.
Herramientas de biblioteca	• Cuenta de ciclos. • Buffers circulares. • Comprobación del paquete de software. • Soporte para pruebas y servicio.

Tutoriales	• Propiedades de las versiones básicas de los *BEANS.*

ANEXO E. Código Implementado.

En la programación se generaran dos códigos que pertenecen a: el código principal llamado *paciente1.c* donde se encuentran todas las técnicas expuestas en el artículo y un código generado por los diferentes *Bean's* llamado *Events.c*

1. Principal

```
/** ###################################################################
**     Filename   :DWT_EMG_DSC.C
**     Project    :DWT_EMG_DSC
**     Processor : 56F8357
**     Version    : Driver 01.14
**     Compiler   : Metrowerks DSP C Compiler
**     Date/Time : 06/09/2011, 07:27 p.m.
**     Abstract   :
**         Main module.
**         This module contains user's application code.
**     Settings   :
**     Contents   :
**         No public methods
**
** ###################################################################*/
/* MODULE DWT_EMG_DSC */

/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "Identifica.h"
#include "Inhr1.h"
#include "CargaMODELO.h"
#include "Inhr2.h"
#include "RxDATO.h"
#include "DATA.h"
#include "RS.h"
#include "Enable.h"
#include "CargaMOD.h"
#include "Inhr3.h"
#include "IDENT.h"
#include "Inhr4.h"

/* Including shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
#include "string.h"
#include "stdio.h"
#include "math.h"

float X[2][270]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

```
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0};

float WL[8]={0.0322, -0.0126, -0.0992,  0.2979,  0.8037,  0.4976, -0.0296, -0.0758};
float WH[8]={-0.0758,  0.0296,  0.4976, -0.8037,  0.2979,  0.0992, -0.0126, -0.0322};
int clase1[21] = {1,1,1,1,1,1,2,2,2,2,2,3,3,3,3,4,4,4,5,5,6};
int clase2[21] = {2,3,4,5,6,7,3,4,5,6,7,4,5,6,7,5,6,7,6,7,7};

float Z[131], V[131], R[8], U[282], RMS[10], ACP[6],media[10],Desv[10],P[10][6],MSV[2],
SV[128][6],Alpha[128],b[21];
intfilas[21];

/*********************************************************************************************

const byte fuente1[]="Identificador de";
const byte fuente2[]="  Movimientos   ";
const byte     uis[]="    * UIS *     ";
const byte    uis1[]="    * E3T *     ";
const byte     est[]="   Fabian Pena  ";
const byte    est1[]="  Cesar Peralta ";

const byte  fraseM[]="                ";
const byte  frase0[]="  Envie Modelo..";
const byte  frase01[]="      Luego......";
const byte  frase1[]=" ..espere aviso ";
const byte  frase11[]="     de Matlab    ";
const byte  frase2[]="-> Presione IRQB";
const byte  frase3[]=" para cargarlo  ";

const byte  frase4[]="  Enviesenales.";
const byte  frase5[]="-> presione IRQA";
const byte  frase6[]="Para Identificar";

const byte    Qmov[]="   Movimiento:  ";
const byte    mov1[]="    Apertura    ";
const byte    mov2[]="     Cierre     ";
const byte    mov3[]="   Extencion    ";
const byte    mov4[]="    Flexion     ";
const byte    mov5[]="   Pronacion    ";
const byte    mov6[]="     Reposo     ";
const byte    mov7[]="   Supinacion   ";

void MENU (const byte *frase1, const byte *frase2);
void INIC_LCD(void);
void COMANDO(byte p);
void ESCRIBIR1(const byte *c);
void RETARDO(wordRestardo);
void ESPERA(void);
void ESCRIBIR_INSTRUCCION(byte Instruccion);
void ESCRIBIR_DATO(byte Dato);
void ESCRIBIR(const byte *c);

float LE1,LE2,LE3,LE4,LE5,LE6,LE7,LE8,LE9,LE10,MOS;
float al,al1,al2,al3,al4,al5,al6,al7,al8,al9;
bytepuntos[16];

/*********************************************************************************************
*****************/
void  conv(float  U[282],  float  WL[8],  float  WH[8],  float  Z[131],float  V[131],  float
X[2][270], int N, int M, int l, int t);
intrmss(float U[282], float RMS[10], int s);
float poly(float ACP[6], float MSV[5], float SV[78][6], float H[7], float Alpha[128], intgg,
intff);
floatraiz(float x);

externbool flag1=FALSE;
externbool flag2=FALSE;
extern float num[1052];
externint jota;
externint IRQ;
```

```
void main(void)
{

  /* Write your local variable definition here */

        static float  H1=0, H[7], f, fx,  prob[7], pos=0,neg=0, votos[7], temp[7], prob1[7];
        static int N=0,M,A,B,fi,co,t,l,c,i,j,k,clas1, clas2, sign, rep1[7], rep2[7], cont,
        Y=0, ff=0, gg=0, e, s=0, ss, voto[7]; //N contiene el numero de datos a ingresar por
canal


/*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

  /* Writeyourcodehere */

        INIC_LCD();     //InicializalaLCD
        MENU(fuente1,fuente2); //Visualiza el título del proyecto
        ESPERA();
        MENU(uis,uis1);              //Visualiza los logos de la universidad y de la escuela
        ESPERA();
        MENU(est,est1);               //Visualiza los autores
        ESPERA();
        MENU(frase0,frase01);         //Visualiza los autores
        ESPERA();
        ESPERA();
        ESPERA();
        ESPERA();
        MENU(frase1,frase11);         //Visualiza lo sautores
        ESPERA();

/*************************************************************************************************
for(;;){

        if(IRQ==1)
        {
                MENU(frase2,frase3);          //Visualiza los autores
                ESPERA();
                IRQ=0;

        }
        if(IRQ==2)
                {
                        MENU(frase5,frase6);    //Visualiza indicaciones para el usuario
                        ESPERA();
                        IRQ=0;
                }

//*************************************************************************************************
        if (flag2==TRUE)
        {

        CargaMOD_Set(1);

                k=1;
for(i=0;i<num[0]*100;i++){
for(j=0;j<6;j++){
          SV[i][j]=num[k];
k++;
      }
 }
for(i=0;i<num[0]*100;i++){
    Alpha[i]=100*num[k];
k++;}

for(i=0;i<10;i++){
for(j=0;j<6;j++){
P[i][j]=num[k];
```

```
k++;
     }
}
for(i=0;i<21;i++){
b[i]=num[k];
k++;}
for(i=0;i<2;i++){
MSV[i]=num[k];
k++;}
for(i=0;i<10;i++){
Desv[i]=num[k];
k++;}

for(i=0;i<10;i++){
media[i]=num[k];
k++;}

for(i=0;i<21;i++){
filas[i]=num[k]*10;
k++;}
k=0;

for(i=0; i<1052; i++){
        num[i]=0;
}
                jota=0;
                flag2=FALSE;
                CargaMOD_Set(0);

                MENU(frase4,frase01);  //Visualiza indicaciones para el usuario
                ESPERA();
                ESPERA();
                ESPERA();
                ESPERA();
                MENU(frase1,frase11);  //Visualiza indicaciones para el usuario
                ESPERA();

 }

//*************************************************************************************************

if (flag1==TRUE)
        {
        IDENT_Set(1);
        for( i=0,j=7;i<256;i++,j++)
                {
                        X[0][j]=num[i];
                        X[1][j]=num[i+256];
                }

        for(i=0; i<1052; i++){
                num[i]=0;
        }

        f=0,fx=0,H1=0,ff=0,e=0,ss=0,clas1=0, clas2=0, sign=0,cont=0;
        ACP[0]=0,ACP[1]=0,ACP[2]=0,ACP[3]=0,ACP[4]=0,ACP[5]=0;
        votos[0]=0, votos[1]=0, votos[2]=0,votos[3]=0,votos[4]=0,votos[5]=0,votos[6]=0;
        RMS[0]=0,  RMS[1]=0,  RMS[2]=0,  RMS[3]=0,  RMS[4]=0,  RMS[5]=0,  RMS[6]=0,  RMS[7]=0,
        RMS[8]=0, RMS[9]=0;
        s=0;

        for(t=0;t<2;t++){
        fi=0;co=0;i=0;j=0;k=0;l=151;N=256;

        M=8+N-1; //M numero de filas de la matriz de coeficientes wavelet

//-------------------------------------------------------------------------

for(i=0;i<4;i++){;
        conv(U, WL, WH, Z, V, X, N, M, l, t); // función convolución
```

```
        N=(M/2);// se van reduciendo el tamaño de los datos después de cada convolución
        M=8+N-1;

for(j=0;j<N;j++){
        X[t][j+7]=Z[j];//se reemplaza los datos de X por los coeficientes de detalle de cada
convolución.
        }
l-=M/2;
        }
        for(i=0; i<N; i++){
        U[i]=Z[i]*Z[i];
        }
     // se halla el valor rms de los datos de aproximación y detalle, guardados en U y se
almacenan en el vector RMS

        ss=rmss(U, RMS, s);
        s=ss;
        }

for(i=0; i<10; i++){

                RMS[i]=(RMS[i]-media[i])/Desv[i];    //se modifican los valores RMS con la
media y desviación, med y desv datos que son cargados
    }
        for(i=0; i<6; i++){
        for(j=0; j<10; j++){

        ACP[i]+=P[j][i]*RMS[j];    // se hallan y se guardan los datos ACP, 6 datos, P es un
vector cargado
        }
        }
//-----------------  MSV

        for(i=0; i<6; i++){
       ACP[i]=ACP[i]/MSV[0];              //    datos    ACP,    MSV    vector    cargado,    con
       caracteristicasMSV[0]== 5,9187  norma
        }
        ff=0,gg=0,fx=0,clas1=0,clas2=0;
        for(k=0; k<21; k++){                                                    // se itera
21 veces que corresponde al numero de modelos

        H[0]=0,H[1]=0,H[2]=0,H[3]=0,H[4]=0,H[5]=0,H[6]=0;
        e=0;H1=0;                                                       //Implementado un
sistema de votación y en caso de empate
        gg=ff;                                                           //se trabaja con
probabilidades
// metodo polynomial
        ff+=filas[k];
        f = poly(ACP, MSV, SV, H, Alpha, gg, ff);
fx=f+b[k];

        voto[0]=0, voto[1]=0, voto[2]=0,voto[3]=0,voto[4]=0,voto[5]=0,voto[6]=0;
      //sistema de votación por parejas

        clas1=clase1[k];  // clas1 depende de el vector cargado clase1 de 21 datos
        clas2=clase2[k];   // clas2 depende de el vector cargado clase2 de 21 datos
        voto[clas1-1]=1;
        voto[clas2-1]=-1;   // dependiendo la clase asigna 1- 0 1 en el vector voto

if(fx>0)
        sign=1;
        else sign=-1;
        for(i=0; i<7; i++){
        rep1[i]=sign;
                }
                for(i=0; i<7; i++){
                rep2[i]=voto[i];
                }
                for(i=0; i<7; i++){
                        voto[i]=rep1[i]*rep2[i];
```

```
        }

prob1[0]=0, prob1[1]=0, prob1[2]=0, prob1[3]=0, prob1[4]=0, prob1[5]=0, prob1[6]=0;

        for(i=0; i<1; i++){              // función de probabilidad, para el caso de empates
        pos = 1/(1+exp(-fx));
        neg=1-pos;
        }

prob1[clas1-1]=pos;          // se llena el subvector de probabilidad
prob1[clas2-1]=neg;


        for(i=0; i<7; i++){
                votos[i]+=voto[i];
        }

        for(j=0; j<7; j++){
prob[j]+=prob1[j];    // se llena el vector de probabilidad
        }
        }

for(j=0; j<7; j++){
temp[j]=votos[j]+prob[j]/21; // temp vector de 7 datos que contiene el movimiento echo, de 7
movimientos
  }

for(j=0; j<7; j++){  // identificación del movimiento
cont=0;
for(i=0; i<7; i++){
if(temp[j]>=temp[i])      //Obtiene el valor máximo de cada fila en "temp" y la ubicación de
la columna en "Y"
cont+=1;
         }
if(cont==7){
            Y=j+1;
break;}
  }
 //asm(nop);
jota=0;
 flag1=FALSE;
IDENT_Set(0);


/*******************************************************/

switch (Y)
 {
case 1:
MENU(Qmov,mov1); //Visualizaeltítulodelproyecto
break;
case 2:
MENU(Qmov,mov2); //Visualizaeltítulodelproyecto
break;
case 3:
MENU(Qmov,mov3); //Visualizaeltítulodelproyecto
break;
case 4:
MENU(Qmov,mov4); //Visualizaeltítulodelproyecto
break;
case 5:
MENU(Qmov,mov5); //Visualizaeltítulodelproyecto
break;
case 6:
MENU(Qmov,mov6); //Visualizaeltítulodelproyecto
break;
case 7:
MENU(Qmov,mov7); //Visualizaeltítulodelproyecto
break;
}
```

```
}
}
}
/*****************************************************************************************************
*****************/
intrmss(float U[282], float RMS[10], int s){
float sum=0; int i;
for(i=0; i<22; i++){
sum+=U[i];
     }
RMS[s]=raiz(sum/22);
sum=0;
s++;
for(i=22; i<44; i++){
sum+=U[i];
}
RMS[s]=raiz(sum/22);
sum=0;
s++;
for(i=22; i<82; i++){
sum+=U[i];
}
RMS[s]=raiz(sum/38);
sum=0;
s++;
for(i=38; i<151; i++){
sum+=U[i];
}
RMS[s]=raiz(sum/69);
sum=0;
s++;
for(i=69; i<282; i++){
sum+=U[i];
}
RMS[s]=raiz(sum/131);
s++;

return s;
}
/*****************************************************************************************************
*****************/
void  conv(float  U[282],  float  WL[8],  float  WH[8],  float  Z[131],float  V[131],  float
X[2][270], int N, int M, int l, int t)
{
float sum1,sum2;
int k=0,i=0,j=0,g=0,fi,co,c=0;
       int A=N+14, B=M+14;

for( fi=0;fi<7;fi++)     {
X[t][6-fi]=X[t][7+fi]; //refleja los 7 primeros datos al principio del vector
        X[t][A-1-6+fi]=X[t][A-1-7-fi]; //refleja los 7 últimos datos al final del vector
     }
for(fi=1; fi<M; fi=fi+2){
       sum1=0,sum2=0,j=fi;
for(co=0;co<8; co++){
           sum1+=WL[co]*X[t][j];
           sum2+=WH[co]*X[t][j];

j++;

       }
       Z[k]=sum1;
       V[k]=sum2;
k++;
   }
for(i=0; i<k; i++)
{
  U[l]=V[i]*V[i];  // se van guardando los coeficientes de aproximación elevados al cuadrado

l++;
```

```
}
}
/*************************************************************************************************
float poly(float ACP[6], float MSV[5], float SV[78][6], float H[7], float Alpha[77], intgg,
intff){
inti,j,e=0; float f=0, H1=0;
for(i=gg; i<ff; i++){
for(j=0; j<6; j++){
                H1+=ACP[j]*SV[i][j];  //el vector SV contiene 21 matrices de diferentes filas
por 6 columnas que corresponden a # de modelos
          }

          H[e]=pow((H1+MSV[1]),MSV[1]);  // H guarda los datos salidos del metodo polynomial
          H1=0;

e++;
      }

      e=0;
for(j=gg; j<ff; j++){     //Alpha contiene 21 modelos MSV
                f+=H[e]*Alpha[j];     // f es un dato, para una entrada de 256 datos

e++;
          }
return f;
}
/*************************************************************************************************
******************/
floatraiz(float x){

float r=0, y=1;
while(r!=y){
    r=y;
    y=(y+(x/y))/2;

    }

return r;
}

/***************************-----LCD------
/*

  La librería utilizada a continuación pertenece a la investigación de pregrado

          GENERADOR DE SEÑALES PARA MEDICIONES DE IMPEDANCIA
                ELECTROQUÍMICA BASADO EN DSP.

       Por los autores:
              ADRIANA MARCELA MARTÍNEZ RIVERA
              EDGAR LEONARDO ARENAS RANGEL

       UIS Bucaramanga 2006
*/

void MENU (const byte *frase1, const byte *frase2)
{
COMANDO(1); // Limpia LCD
ESCRIBIR(frase1);  //Escribe la frase1 en la primera linea
COMANDO(192);  //Posiciona el cursor en la segunda linea 192
ESCRIBIR(frase2);  //Escribe la frase 2 en la segunda linea
}
/*************************************************************************************************
void INIC_LCD(void)
{
RETARDO(7000);
ESCRIBIR_INSTRUCCION(3); //Operacion con 8 bits
RETARDO(3000);
ESCRIBIR_INSTRUCCION(3); //Operacion con 8 bits
RETARDO(3000);
```

```
ESCRIBIR_INSTRUCCION(3); //Operacion con 8 bits
RETARDO(2000);
ESCRIBIR_INSTRUCCION(2); //Operacion con 4 bits
RETARDO(2000);
ESCRIBIR_INSTRUCCION(2); //Operacion con 4 bits
RETARDO(2000);
ESCRIBIR_INSTRUCCION(8); //Se fija N=1 (26 lineas) y F=0 (5X7 dot)
RETARDO(2000);
ESCRIBIR_INSTRUCCION(0); //Display Apagado
RETARDO(2000);
ESCRIBIR_INSTRUCCION(8); //Display Apagado
RETARDO(2000);
ESCRIBIR_INSTRUCCION(0); //Limpiar Display
RETARDO(2000);
ESCRIBIR_INSTRUCCION(1); //Limpiar Display
RETARDO(2000);
ESCRIBIR_INSTRUCCION(0); //Fija el Modo (Incremento/Decremento)
RETARDO(2000);
ESCRIBIR_INSTRUCCION(6); //I/D=1 SE fija en modo Incremento
RETARDO(2000);
ESCRIBIR_INSTRUCCION(0); //Display Encendido
RETARDO(2000);
ESCRIBIR_INSTRUCCION(15); //C=1 (cursor Encend) B=1 (Parpadeo Encend)
RETARDO(2000);
}
/*****************************************************************************************
void COMANDO(byte p)
{
        byte MSB, LSB;
        RETARDO(75);
        MSB=0xF0&p;
        LSB=0x0F&p;
        MSB=MSB>>4;
        ESCRIBIR_INSTRUCCION(MSB);
        RETARDO(75);
        ESCRIBIR_INSTRUCCION(LSB);
        RETARDO(75);
}
//---------------------------------------------------------------------
void RETARDO(word Retardo)
{ worda,b;
Retardo=Retardo*2;
for(a=0;a<=Retardo; a++)
  {
        for(b=0; b<=2; b++)
        {
        asm
          {
                nop;
                nop;
                nop;
                nop;
                nop;
        }
        }
  }
}
/*****************************************************************************************
void ESPERA(void) // TIEMPO DE ESPERA PARA VISUALIZAR PRESENTACION
{int i;
for(i=0; i<20; i++)
{
        RETARDO(50000);
  }
}

/*****************************************************************************************
void ESCRIBIR_INSTRUCCION(byte Instruccion)
{
        RS_ClrVal();//Para enviar una instrucción
```

```
        RETARDO(75);
        DATA_PutVal(Instruccion);
        RETARDO(75);
        Enable_SetVal();
        RETARDO(75);
        Enable_ClrVal();
        RETARDO(75);

}
/*****************************************************************************************
void ESCRIBIR_DATO(byte Dato)
{
        RS_SetVal();//Para enviar una instrucción
        RETARDO(75);
        DATA_PutVal(Dato);
        RETARDO(75);
        Enable_SetVal();
        RETARDO(75);
        Enable_ClrVal();
        RETARDO(75);

}
/*****************************************************************************************
void ESCRIBIR(const byte *c)
{byte MSB, LSB; int i;
for(i=0; i<16; i++)
  {
        RETARDO(75);
        MSB=0xF0&(c[i]);
        LSB=0x0F&(c[i]);
        MSB=MSB>>4;
        ESCRIBIR_DATO(MSB);
        RETARDO(75);
        ESCRIBIR_DATO(LSB);
        RETARDO(75);
  }

}
/* END DWT_EMG_DSC */
/*
** ###################################################################
**
**     This file was created by Processor Expert 3.00 [04.35]
**     for the Freescale 56800 series of microcontrollers.
**
** ###################################################################
*/
```

2. Eventos.

```
/** ###################################################################
**     Filename  :Events.C
**     Project   : DWT_EMG_DSC
**     Processor : 56F8357
**     Component :Events
**     Version   : Driver 01.03
**     Compiler  : Metrowerks DSP C Compiler
**     Date/Time : 06/09/2011, 07:27 p.m.
**     Abstract  :
**         This is user's event module.
**         Put your event handler code here.
**     Settings  :
**     Contents  :
**         No public methods
**
** ###################################################################*/
/* MODULE Events */

#include "Cpu.h"
```

```
#include "Events.h"
#include "stdlib.h"
#include "string.h"

bytedato=0;
int i=0;
char dato2[15];

externbool flag1;
externbool flag2;
externint jota=0;
externint IRQ=0;
extern                                                                     float
num[1052]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

/* User includes (#include below this line is not maintained by Processor Expert) */

/*
** ===================================================================
**     Event       :  Identifica_OnButton (module Events)
**
**     Component   :  Identifica [Button]
**     Description :
**         This event is called when the button is pressed.
**         If button inactivity feature (advanced view) is enabled,
**         then the next OnButton event is not generated during dead
**         time.
**     Parameters  : None
**     Returns     : Nothing
** ===================================================================
*/
#pragma interrupt called /* Comment this line if the appropriate 'Interrupt preserve
registers' property */
                         /* is set to 'yes' (#pragma interrupt saveall is generated before
the ISR)       */
voidIdentifica_OnButton(void)
{
  /* Write your code here ... */
flag1=TRUE;                       // CORRE CODIGO
}
/*
** ===================================================================
**     Event       :  CargaMODELO_OnButton (module Events)
**
**     Component   :  CargaMODELO [Button]
**     Description :
**         This event is called when the button is pressed.
**         If button inactivity feature (advanced view) is enabled,
```

```
**         then the next OnButton event is not generated during dead
**         time.
**     Parameters  : None
**     Returns     : Nothing
** ===================================================================
*/
#pragma interrupt called /* Comment this line if the appropriate 'Interrupt preserve
registers' property */
                         /* is set to 'yes' (#pragma interrupt saveall is generated before
the ISR)      */


voidCargaMODELO_OnButton(void)
{
  /* Write your code here ... */
flag2=TRUE;
}
/*
** ===================================================================
**     Event       :  RxDATO_OnError (module Events)
**
**     Component   :  RxDATO [AsynchroSerial]
**     Description :
**         This event is called when a channel error (not the error
**         returned by a given method) occurs. The errors can be
**         read using <GetError> method.
**         The event is available only when the <Interrupt
**         service/event> property is enabled.
**     Parameters  : None
**     Returns     : Nothing
** ===================================================================
*/
#pragma interrupt called /* Comment this line if the appropriate 'Interrupt preserve
registers' property */
                         /* is set to 'yes' (#pragma interrupt saveall is generated before
the ISR)      */
voidRxDATO_OnError(void)
{
  /* Write your code here ... */
}
/*
** ===================================================================
**     Event       :  RxDATO_OnRxChar (module Events)
**
**     Component   :  RxDATO [AsynchroSerial]
**     Description :
**         This event is called after a correct character is
**         received.
**         The event is available only when the <Interrupt
**         service/event> property is enabled and either the
**         <Receiver> property is enabled or the <SCI output mode>
**         property (if supported) is set to Single-wire mode.
**         Version specific information for Freescale 56800
**         derivatives ]
**         DMA mode:
**         If DMA controller is available on the selected CPU and
**         the receiver is configured to use DMA controller then
**         this event is disabled. Only OnFullRxBuf method can be
**         used in DMA mode.
**     Parameters  : None
**     Returns     : Nothing
** ===================================================================
*/
#pragma interrupt called /* Comment this line if the appropriate 'Interrupt preserve
registers' property */
                         /* is set to 'yes' (#pragma interrupt saveall is generated before
the ISR)      */

voidRxDATO_OnRxChar(void)
{
```

```
  /* Write your code here ... */

RxDADO_RecvChar(&dato);
        {
        if(dato!='\x00')
          {
        dato2[i]=dato;
        i++;
          }
if(i==15)
          {
        num[jota]=atof(dato2);
          i=0;
        jota+=1;
          }
        if(dato=='\x00' && jota>512)
        {
                IRQ=1;
        }
        if(dato=='\x00' && jota==512)
        {
                IRQ=2;
        }
}
}
/* END Events */
/*
** ###################################################################
**
**     This file was created by Processor Expert 3.00 [04.35]
**     for the Freescale 56800 series of microcontrollers.
**
** ###################################################################
*/
```