Implementación en dsPIC de un algoritmo reductor de ruido basado en la transformada wavelet discreta

Alejandro José Uriz, Pablo Daniel Agüero, Juan Carlos Tulli, Jorge Castiñeira Moreira, Esteban González, y Roberto Hidalgo

CONICET - Facultad de Ingeniería, Universidad Nacional de Mar del Plata,
Juan B. Justo 4302, 7600 Mar del Plata, Argentina
{ajuriz@conicet.gov.ar,pdaguero@fi.mdp.edu.ar}
http://200.0.183.36/pegasus

Resumen En general uno de los primeros pasos al procesar una señal de voz es reducir el ruido que la misma contiene, para dicho propósito existen diversos algoritmos. Uno de los métodos mas eficientes para realizar esta operación es mediante la transformada wavelet discreta (TWD). Esta presenta la ventaja de poder trabajar con señales no estacionarias y, posee la capacidad de realizar un análisis multiresolución. Su principal desventaja es que la cantidad de memoria de datos que utilizan dificulta su implementación en un dispositivo dedicado al procesamiento digital de señales (DSP) de bajo costo. El objetivo de este trabajo es implementar un algoritmo reductor de ruido para señales de voz basado en la TWD, el cual forma parte de un sistema de asistencia auditiva implementado en un dsPIC. Se realizan mediciones experimentales con el fin de analizar el rendimiento del sistema.

Key words: Procesamiento Digital de Señales, Transformada Wavelet Discreta, Reducción de ruido, dsPIC.

1. Introducción

A la hora de analizar una señal es muy importante extraer la información de interés contenida en una señal que además puede estar compuesta por información que no se desee estudiar. Con este propósito se deben utilizar procedimientos capaces de filtrar las componentes de señal indeseadas.

Existen diversos algoritmos para reducir la cantidad de ruido existente en una señal. Uno de los mas utilizados es el método basado en el cálculo de los autovalores [1], el cual representa la señal en un formato matricial, al cual le calcula sus autovalores. Los autovalores de mayor valor se suponen asociados a las componentes de señal pura, mientras que los autovalores menores se asocian con componentes de ruido. Una vez descompuesta la señal, la misma es reconstruida utilizando los autovalores mayores. La cantidad de autovalores utilizados en la reconstrucción determina el nivel de ruido que se elimina. Si bien este método

permite obtener buenos resultados, es necesario generar representaciones matriciales de la señal, que consumen gran cantidad de recursos. Existe otro método de reducción de ruido basado en la **transformada wavelet discreta** (TWD) [2,3]. El término wavelet se define como una *pequeña onda* o función localizable en el tiempo, que vista desde una perspectiva del análisis o procesamiento de señal puede ser considerada como una herramienta matemática para realizar representación y segmentación de señales, mediante un análisis tiempo-frecuencia. Las características propias de la transformada wavelet otorgan la posibilidad de representar señales en diferentes niveles de resolución, así como también señales con variaciones abruptas en forma eficiente, y analizar señales no estacionarias.

El objetivo de este trabajo es implementar un algoritmo reductor de ruido [4] que permita procesar señales de voz, las cuales son no estacionarias, motivo por el cual la herramienta mas apropiada para dicha tarea es la transformada wavelet discreta.

Dado que se desea implementar el algoritmo en un dispositivo de asistencia auditiva de bajo costo basado en un DSP de Microchip[5], se debe realizar un proceso de optimización de las operaciones realizadas con el fin de reducir el uso de memoria de datos y que, ademas, el mismo pueda ejecutarse en tiempo real. El dispositivo elegido para la implementación es el dsPIC33FJ128GP802 [7].

Este trabajo está organizado de la siguiente manera. La Sección 2 describe las características mas importantes de los algoritmos de reducción de ruido utilizando la TWD. La Sección 3 desarrolla las consideraciones necesarias para implementar el algoritmo en el dispositivo seleccionado. La Sección 4 presenta las mediciones desarrolladas para evaluar el desempeño del sistema, así como también el prototipo utilizado para realizar dichas mediciones. Finalmente, la Sección 5 presenta las conclusiones del trabajo y la lineas de investigación futuras.

2. Reducción de ruido utilizando la TWD

Para el cálculo de la transformada wavelet discreta interviene una matriz que almacena los coeficientes de la ondita a utilizar. La dimensión de dicha matriz está determinada por la longitud de la señal a procesar elevada al cuadrado, por lo tanto suele consumir una gran cantidad de recursos de memoria. Es de gran importancia entonces estudiar la señal para trabajar con la menor cantidad posible de memoria de datos. Los algoritmos que permiten reducir el nivel de ruido presentan tres etapas:

1. **Análisis:** Se toma la señal a procesar y se la descompone utilizando una ondita [2]. El nivel de descomposición depende de los requisitos del filtro a implementar. En este paso, se parte de una señal $x[n]$ y se le aplica la transformada wavelet discreta, de la cual se obtienen dos señales: una señal suavizada $s_1[n]$, la cual está asociada a las componentes de baja frecuencia, y otra señal $d_1[n]$ que contiene los detalles de la señal analizada. En el siguiente nivel de análisis se descompone $s_1[n]$ en dos señales $s_2[n]$ y $d_2[n]$. Este

proceso se repite hasta que se obtiene el nivel necesario para poder aplicar el filtrado adecuado. Un esquema de las operaciones detalladas se representa en la Ecuación 1. Donde W representa la matriz de coeficientes de la ondita utilizada, en cada uno de los instantes de tiempo, y x representa la señal que está siendo analizada.

$$W.x = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ c_3 & -c_2 & c_1 & -c_0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & c_0 & c_1 & c_2 & c_3 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & c_3 & -c_2 & c_1 & -c_0 & ... & 0 & 0 & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & 0 & 0 & ... & c_0 & c_1 & c_2 & c_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & ... & c_3 & -c_2 & c_1 & -c_0 \\ c_2 & c_3 & 0 & 0 & 0 & 0 & ... & 0 & 0 & c_0 & c_1 \\ c_1 & -c_0 & 0 & 0 & 0 & 0 & ... & 0 & 0 & c_3 & -c_2 \end{bmatrix} \begin{bmatrix} x_i[1] \\ x_i[2] \\ x_i[3] \\ x_i[4] \\ ... \\ ... \\ x_i[N-3] \\ x_i[N-2] \\ x_i[N-1] \\ x_i[N] \end{bmatrix} = \begin{bmatrix} s_{i+1}[1] \\ d_{i+1}[1] \\ s_{i+1}[2] \\ d_{i+1}[2] \\ ... \\ ... \\ s_{i+1}[N/2-1] \\ d_{i+1}[N/2-1] \\ s_{i+1}[N/2] \\ d_{i+1}[N/2] \end{bmatrix} \quad (1)$$

2. **Umbralamiento:** Una vez finalizado el análisis de la señal, se aplica el filtrado sobre cada una de las señales de detalle $d_i[n]$ obtenidas de cada nivel de análisis. El umbralamiento consiste en fijar un nivel por debajo del cual las componentes de la señal residual $d_i[n]$ son descartadas.

$$W^T.y_{i+1} = \begin{bmatrix} c_2 & c_1 & c_0 & c_3 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ c_3 & -c_0 & c_1 & -c_2 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & c_2 & c_1 & c_0 & c_3 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & c_3 & -c_0 & c_1 & -c_2 & ... & 0 & 0 & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & 0 & 0 & ... & c_2 & c_1 & c_0 & c_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & ... & c_3 & -c_0 & c_1 & -c_2 \\ c_0 & c_3 & 0 & 0 & 0 & 0 & ... & 0 & 0 & c_2 & c_1 \\ c_1 & -c_2 & 0 & 0 & 0 & 0 & ... & 0 & 0 & c_3 & -c_0 \end{bmatrix} \begin{bmatrix} y_{i+1}[1] \\ y_{i+1}[2] \\ y_{i+1}[3] \\ y_{i+1}[4] \\ ... \\ ... \\ y_{i+1}[N-3] \\ y_{i+1}[N-2] \\ y_{i+1}[N-1] \\ y_{i+1}[N] \end{bmatrix} = \begin{bmatrix} \hat{s_i}[1] \\ \hat{s_i}[2] \\ \hat{s_i}[3] \\ \hat{s_i}[4] \\ ... \\ ... \\ \hat{s_i}[N-3] \\ \hat{s_i}[N-2] \\ \hat{s_i}[N-1] \\ \hat{s_i}[N] \end{bmatrix} \quad (2)$$

3. **Reconstrucción:** Una vez que el umbralamiento es finalizado, en cada una de las señales $d_i[n]$ sólo se conservan los elementos cuyo valor absoluto supera al del umbral correspondiente a cada nivel. De esta forma, se tiene una nueva señal de detalle $\hat{d}_i[n]$ la cual es una versión filtrada de $d_i[n]$. Debido a que se posee una nueva señal de detalle para cada nivel, se aplica la transformada wavelet inversa en cada uno de los niveles. Para realizar dicha operación se toman los vectores $s_{i+1}[n]$ y $\hat{d}_{i+1}[n]$, y se las reordena para generar un nuevo vector $y_{i+1}[n]$. Este nuevo vector se compone en sus posiciones impares de los elementos del vector $s_{i+1}[n]$ y en las pares de $d_{i+1}[n]$. Una vez obtenido el vector $y_{i+1}[n]$, se le aplica la TWD inversa y se obtiene la versión filtrada de

la señal suavizada del nivel previo $\hat{s_i}[n]$. El procedimiento se repite al nivel inicial de señal $x\hat{[}n]$. Una representación matricial de la operación $W^T.y_{i+1} = \hat{s_i}[n]$ se aprecia en la Ecuación 2.

Como se comentó previamente, el principal objetivo de este trabajo es implementar un algoritmo reductor de ruido para señales de voz en un dispositivo DSP, que debe operar en tiempo real. Por este motivo, se deben considerar la cantidad de operaciones necesarias para realizar la TWD, el umbralamiento y la transformada inversa con el fin de optimizar el uso de la memoria de datos, y reducir el tiempo de cálculo.

3. Optimización de recursos

El estudio comienza por las variables que mas memoria de datos utilizan, es decir las matrices de análisis W y de síntesis W^T. De acuerdo a lo planteado en [8], la transformada wavelet discreta puede definirse matricialmente como se presenta en la Ecuación 1.

Un análisis de la Ecuación 1 muestra que cada fila tiene sólo 4 elementos no nulos, los cuales además son consecutivos. Por otro lado, cada una de filas pares de la matriz se compone de la fila par inmediatamente anterior, pero desplazada dos posiciones a la derecha. Además, estos mismos patrones se repiten para las filas impares. De esta forma, es posible reescribir la matriz W completa que consta de $N.N$ elementos, usando tan sólo dos vectores de 4 coeficientes cada uno. Por lo tanto, para realizar los cálculos correspondientes sólo es necesario desplazar dichos vectores de la misma forma en que lo hacen las filas de la matriz W. El mismo criterio puede aplicarse a la matriz W^T. De esta forma, se logra reducir la memoria de datos utilizada para almacenar ambas matrices de $2.N.N$ coeficientes a tan solo 16.

Además, se pueden reutilizar cada uno de los registros para el caso de los sucesivos vectores suavizados $s_i[n]$, detalle $d_i[n]$ y de sus correspondiente versiones reconstruidas $\hat{s_i}[n]$ y $\hat{d_i}[n]$, se plantea reutilizar cada uno de los registros.

Solo se requieren 2 registros de longitud N para almacenar todas las señales filtradas $s_i[n]$ y $\hat{s_i}[n]$. Por otro lado, para las señales que contienen los detalles, se requiere un vector por cada nivel de análisis. Esto se realiza, debido a que en caso contrario, al sobreescribir se perderían componentes críticas para la reconstrucción. Por lo tanto, para el caso en que se desee implementar un algoritmo reductor de ruido, utilizando una ondita **Daubechies 4** (db4) tal como el que se presenta en [8], pero con una descomposición de cuarto nivel, se utilizan los recursos que se listan en el Cuadro 1.

En el Cuadro 1 puede apreciarse claramente que la principal mejora es la reducción en las matrices W y W^T. Además, las señales filtradas $s_i[n]$ y sus versiones reconstruidas son almacenadas solo en dos vectores de longitud N. Esto se realiza debido a que para la síntesis de la señal, cada uno de los vectores deberá ser reconstruido con las componentes filtradas. En cambio, las señales que contienen los detalles $d_i[n]$ son necesarias para la reconstrucción de la señal.

Cuadro 1. Requerimientos de memoria de datos para la primera implementación propuesta.

Variable	Cantidad de elementos
W	8
W^T	8
$s_1[n]$,$s_2[n]$,$s_3[n]$,$s_4[n]$,$\hat{s_1}[n]$,$\hat{s_2}[n]$,$\hat{s_3}[n]$	$2x(N/2)$
$d_1[n]$,$\hat{d_1}[n]$	$N/2$
$d_2[n]$,$\hat{d_2}[n]$	$N/4$
$d_3[n]$,$\hat{d_3}[n]$	$N/8$
$d_4[n]$,$\hat{d_4}[n]$	$N/16$

Con el fin de optimizar el uso de recursos, se utiliza el mismo registro para almacenar la señal original y la correspondiente señal filtrada $\hat{d_i}[n]$. Si se utiliza $N = 256$, y se utilizan datos de 16 bits, se requiere solo 1KiloByte (KB) de memoria para realizar la implementación del algoritmo. Por este motivo puede ser implementado en un dispositivo dsPIC de Microchip.

Cabe destacar que el algoritmo presentado en la Ecuación 1 tal como es implementado en [8] calcula los últimos elementos de las señales suavizadas y de detalle tal como se presenta en la Ecuación 3, allí puede verse que se utiliza tanto la información del final del vector de entrada como también la inicial. Esto trae aparejado que en las fronteras de los segmentos aparezcan discontinuidades, de la misma forma el procedimiento se repite en los sucesivos niveles de análisis.

$$\begin{aligned} s_{i+1}[N/2] &= c_2.x_i[1] + c_3.x_i[2] + c_0.x_i[N-1] + c_1.x_i[N] \\ d_{i+1}[N/2] &= c_1.x_i[1] + (-c_0).x_i[2] + c_3.x_i[N-1] + (-c_2).x_i[N] \end{aligned} \tag{3}$$

La forma mas simple de corregir este problema es utilizar un segmento de entrada $x[n]$ de mayor longitud, de forma tal que últimas muestras de las señales $s[n]$ y $d[n]$ puedan ser correctamente calculadas. Este procedimiento se detalla en la ecuación 4.

$$\begin{aligned} s_{i+1}[N/2] &= c_0.x_i[N-1] + c_1.x_i[N] + c_2.x_i[N+1] + c_3.x_i[N+2] \\ d_{i+1}[N/2] &= c_3.x_i[N-1] + (-c_2).x_i[N] + c_1.x_i[N+1] + (-c_0).x_i[N+2] \end{aligned} \tag{4}$$

La cantidad de memoria de datos requerida para implementar el algoritmo de reducción de ruido, utilizando las consideraciones previamente descriptas, se presentan en el Cuadro 2.

Bajo las condiciones detalladas en el Cuadro 2, con $N = 256$ y datos de 16 bits de longitud, se utilizan 1272 bytes de memoria. Puede apreciarse que si bien hay un aumento en el uso de memoria de datos respecto a lo planteado en el Cuadro 1, el algoritmo aún puede ser implementado en el dispositivo propuesto.

Cuadro 2. Requerimientos de memoria de datos para la segunda implementación propuesta.

Variable	Cantidad de elementos
W	8
W^T	8
$s_1[n]$,$s_2[n]$,$s_3[n]$,$s_4[n]$,$\hat{s_1}[n]$,$\hat{s_2}[n]$,$\hat{s_3}[n]$	$2x(N/2 + N/8)$
$d_1[n]$,$\hat{d_1}[n]$	$N/2 + N/8$
$d_2[n]$,$\hat{d_2}[n]$	$N/4 + N/16$
$d_3[n]$,$\hat{d_3}[n]$	$N/8 + N/32$
$d_4[n]$,$\hat{d_4}[n]$	$N/16 + N/64$

3.1. Optimización de recursos enfocada a un filtro paso bajos.

Esta segunda optimización es un caso particular del algoritmo de reducción de ruido para el cual los coeficientes de umbralamiento se fijan a un valor tal que todos los elementos de las señales $d_i[n]$ son eliminados y por ende, todos los elementos de $\hat{d}_i[n]$ tienen valor cero.

Por lo tanto, si se tiene en cuenta que todos los elementos de $d_i[n]$ van a ser descartados, se puede omitir el cálculo de dichos vectores. Esto se realiza fijando a cero todos los coeficientes de las filas pares de la matriz W. De esta forma, la Ecuación 1, puede ser reescrita como se observa en la Ecuación 5.

$$W.x = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 & 0 & 0 & ... & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_0 & c_1 & c_2 & c_3 & ... & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & 0 & 0 & ... & 0 & c_0 & c_1 & c_2 & c_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 & 0 \\ c_2 & c_3 & 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & c_0 & c_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_i[1] \\ x_i[2] \\ x_i[3] \\ x_i[4] \\ ... \\ ... \\ x_i[N-3] \\ x_i[N-2] \\ x_i[N-1] \\ x_i[N] \end{bmatrix} = \begin{bmatrix} s_{i+1}[1] \\ 0 \\ s_{i+1}[2] \\ 0 \\ ... \\ ... \\ s_{i+1}[N/2-1] \\ 0 \\ s_{i+1}[N/2] \\ 0 \end{bmatrix} \quad (5)$$

De esta forma, sabiendo que los elementos nulos del vector no van a ser utilizados, puede ser omitido su cálculo, y la Ecuación 5 puede ser reeescrita tal como se aprecia en la Ecuación 6.

La Ecuación 6 muestra que para el caso en que se aplique un filtro paso bajos solo se necesita calcular la mitad de los coeficientes respecto al caso previo. Por otro lado, para la reconstrucción de la señal, si se analiza la operación desarrollada en la Ecuación 2 para el caso donde $\hat{d}_i = 0$. Bajo estas condiciones, el vector $\hat{y}_i$ tiene sus elementos pares nulos.

$$W.x = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 & 0 & 0 & ... & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_0 & c_1 & c_2 & c_3 & ... & 0 & 0 & 0 & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & 0 & 0 & ... & 0 & c_0 & c_1 & c_2 & c_3 \\ c_2 & c_3 & 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & c_0 & c_1 \end{bmatrix} \begin{bmatrix} x_i[1] \\ x_i[2] \\ x_i[3] \\ x_i[4] \\ ... \\ ... \\ x_i[N-3] \\ x_i[N-2] \\ x_i[N-1] \\ x_i[N] \end{bmatrix} = \begin{bmatrix} s_{i+1}[1] \\ s_{i+1}[2] \\ ... \\ s_{i+1}[N/2-1] \\ s_{i+1}[N/2] \end{bmatrix} \tag{6}$$

Además, se pueden eliminar los elementos pares de cada fila de la matriz W^T, dado que al aplicarse la transformada inversa son multiplicados por elementos nulos, lo que da como resultado la matriz que se aprecia en la Ecuaciones 7 y 8, respectivamente. Entonces, también en el caso de la reconstrucción de la señal filtrada, se reduce a la mitad la cantidad de operaciones.

$$\begin{bmatrix} c_2 & 0 & c_0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 & 0 \\ c_3 & 0 & c_1 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_2 & 0 & c_0 & 0 & ... & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_3 & 0 & c_1 & 0 & ... & 0 & 0 & 0 & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & 0 & 0 & ... & 0 & c_2 & 0 & c_0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & ... & 0 & c_3 & 0 & c_1 & 0 \\ c_0 & 0 & 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & c_2 & 0 \\ c_1 & 0 & 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & c_3 & 0 \end{bmatrix} \begin{bmatrix} y_{i+1}[1] \\ 0 \\ y_{i+1}[3] \\ 0 \\ ... \\ ... \\ y_{i+1}[N-3] \\ 0 \\ y_{i+1}[N-1] \\ 0 \end{bmatrix} = \begin{bmatrix} \hat{s_i}[1] \\ \hat{s_i}[2] \\ \hat{s_i}[3] \\ \hat{s_i}[4] \\ ... \\ ... \\ \hat{s_i}[N-3] \\ \hat{s_i}[N-2] \\ \hat{s_i}[N-1] \\ \hat{s_i}[N] \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} c_2 & c_0 & 0 & ... & 0 & 0 \\ c_3 & c_1 & 0 & ... & 0 & 0 \\ 0 & c_2 & c_0 & ... & 0 & 0 \\ 0 & c_3 & c_1 & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & c_2 & c_0 \\ 0 & 0 & 0 & ... & c_3 & c_1 \\ c_0 & 0 & 0 & ... & 0 & c_2 \\ c_1 & 0 & 0 & ... & 0 & c_3 \end{bmatrix} \begin{bmatrix} y_{i+1}[1] \\ y_{i+1}[3] \\ ... \\ y_{i+1}[N-3] \\ y_{i+1}[N-1] \end{bmatrix} = \begin{bmatrix} \hat{s_i}[1] \\ \hat{s_i}[2] \\ \hat{s_i}[3] \\ \hat{s_i}[4] \\ ... \\ ... \\ \hat{s_i}[N-3] \\ \hat{s_i}[N-2] \\ \hat{s_i}[N-1] \\ \hat{s_i}[N] \end{bmatrix} \tag{8}$$

El uso de recursos para la implementación de un filtro paso bajos se desarrolla en el Cuadro 3. Para el caso en que $N = 256$ y se utilizan datos de 16 bits, la memoria de datos requerida es de 656 bytes. Lo que nuevamente, puede ser perfectamente implementado en un dispositivo DSP de bajo costo.

Cuadro 3. Requerimientos de memoria de datos para la implementación del filtro pasobajos.

Variable	Cantidad de elementos
W	4
W^T	4
$s_1[n]$,$s_2[n]$,$s_3[n]$,$s_4[n]$,$\hat{s_1}[n]$,$\hat{s_2}[n]$,$\hat{s_3}[n]$	$2x(N/2+N/8)$

4. Resultados experimentales

Con el fin de analizar el funcionamiento de las mejoras realizadas al algoritmo, se implementó el mismo en MATLAB. Una vez que se verificó el correcto funcionamiento del mismo, se lo implementó en MPLAB C30 [9] con el fin de poder utilizarlo en el DSP elegido. En las siguientes subsecciones se describe brevemente el sistema base utilizado para la implementación y las mediciones realizadas con el fin de verificar el correcto funcionamiento del sistema.

4.1. Prototipo desarrollado

El principal objetivo de este trabajo es lograr implementar el filtro mediante ondículas en un dispositivo dsPIC33FJ128GP802 de Microchip. Además, el sistema debe cumplir la condición de operar en tiempo real.

El prototipo utilizado para realizar la implementación del algoritmo se utiliza como plataforma de desarrollo de un sistema de asistencia auditiva de bajo costo desarrollado por miembros del Laboratorio de Comunicaciones de la Universidad Nacional de Mar del Plata [5,6]. Los subsistemas que lo componen se listan a continuación.

- **Acondicionador de señal**: Está compuesto por un preamplificador de micrófono, un filtro *antialiasing* Sallen Key, elíptico de octavo orden y una etapa para el control automático de ganancia. El objetivo de este subsistema es acondicionar la señal con el fin de que la misma tenga un nivel adecuado para ser adquirida y procesada por el DSP.
- **Procesador Digital de Señales (DSP)**: Este dispositivo digitaliza la señal, la procesa utilizando el algoritmo implementado y por último sintetiza la señal resultante. El dispositivo DSP utilizado es el dsPIC33FJ128GP802, el cual mas adelante es descripto con mayor nivel de detalle.
- **Etapa de salida**: Acondiciona la señal de salida del DSP, al nivel necesario para que pueda ser percibida por el usuario del sistema.

Las características más relevantes del **dsPIC33FJ128GP802**[7] son:

- **128KB de memoria de programa**. Lo cual lo hace apropiado para el uso de compiladores cruzados.
- **16KB de memoria RAM**. De los cuales 2KB son utilizados como memoria compartida para Acceso Directo a Memoria **DMA**.

- **Velocidad de procesamiento de hasta 40 millones de instrucciones por segundo (MIPS)**.
- **Bus de datos de 16 bits**.
- **Conversor analógico a digital (ADC) integrado de 12 bits@500ksps**.
- **Conversor digital a analógico (DAC) integrado de 16 bits@100ksps**.
- **Registros de entrada y salida duplicados**. Esto permite realizar operaciones de lectura y escritura a mayor velocidad. Además otorga una mayor flexibilidad al manejo de los pines de entrada y salida del dsPIC.

Además, debe tenerse en cuenta que la documentación acerca de los dispositivos Microchip y sus librerías [9] están disponibles en internet sin costo alguno.

Quizás la mayor ventaja del dsPIC33FJ128GP802 es que permite realizar en **simultáneo** dos tareas, que en este caso resultan ser el procesamiento en paralelo de los datos de un segmento y la resíntesis de la señal de audio correspondiente al segmento anterior. Esto se realiza utilizando el módulo de DMA del dispositivo [7], el cual trabaja de forma independiente al procesador principal.

Por este motivo se reduce casi a la mitad el tiempo de procesamiento de cada uno de los segmentos, lo cual es un factor crítico a la hora de obtener un dispositivo que funcione en tiempo real. Otro aspecto a tener en cuenta, es que la utilización de técnicas de DMA aumenta el rendimiento del sistema, ya que reduce al mínimo las fuentes de interrupción del programa principal. Es decir, los dispositivos realizan la transferencia de datos utilizando el módulo DMA, y por ello no agrega retardos de ejecución al programa principal. En particular los tiempos que se reducen son:

- Tiempo de procesamiento de la rutina de interrupción.
- Tiempo de acceso, almacenamiento y lectura de la pila ó *stack* del sistema.
- Tiempo de acceso a los periféricos.

En la Subsección siguiente se presentan las mediciones realizadas al sistema implementado.

4.2. Datos registrados

Una vez implementado el sistema se realizaron diferentes mediciones con el fin de analizar el desempeño del mismo. Con este propósito se implementaron cuatro configuraciones de cada uno de los algoritmos presentados en la Sección 2. Para cada una de las implementaciones se midió el uso de recursos y los tiempos de procesamiento. Los valores registrados fueron medidos utilizando una $f_{sampling} = 16{,}288KHz$, y se presentan por separado los tiempos de procesamiento y de adquisición, ya que estas tareas se procesan en paralelo, por lo que el tiempo de procesamiento total del sistema se considera como el mayor de ambos, que para el caso de segmentos de 256 muestras es de $t_{total} = 15{,}72ms$.

En las columnas dos y tres del Cuadro 4 se presentan los datos registrados para cuatro implementaciones del algoritmo de reducción de ruido. En cada una de las filas se presentan los resultados medidos para uno, dos, tres y cuatro niveles de análisis. Cabe destacar que el nivel de análisis depende de la naturaleza de

Cuadro 4. Memoria de programa, memoria de datos, y tiempo de procesamiento correspondientes a las implementaciones de los algoritmos de reducción de ruido y de filtrado pasobajos.

	Reductor de Ruido		**Filtro Pasobajos**		
Niveles de Análisis	**ROM/RAM**	**T. Proc.**	**ROM/RAM**	**T. Proc.**	**T. Adq.**
1	9 %/85 %	0,980ms	8 %/80 %	0,460μs	15,72ms
2	9 %/87 %	1,510ms	9 %/80 %	0,644μs	15,72ms
3	9 %/88 %	1,740ms	9 %/80 %	0,784μs	15,72ms
4	9 %/89 %	1,920ms	9 %/80 %	0,888μs	15,72ms

la señal que se desee filtrar. En la segunda columna del Cuadro 4 se aprecia que la memoria de programa (ROM) utilizada por los algoritmos no presenta cambios significativos para los casos bajo estudio. De la misma forma, la memoria de datos (RAM) presenta solo un leve incremento a medida que la cantidad de niveles de análisis se incrementa. Respecto a los tiempos de procesamiento de los algoritmos, la tercera columna del Cuadro muestra que a medida que aumentan los niveles de análisis, los tiempos se incrementan notablemente. Esto es consecuencia de que el incremento del nivel de análisis trae aparejado un incremento en la cantidad de iteraciones requeridas por el programa para su ejecución. Por último, debe destacarse que en todos los casos el tiempo de procesamiento es menor al tiempo de adquisición de los datos (mostrado en la sexta columna), y por lo tanto el sistema opera en tiempo real.

Por otro lado, en las columnas cuatro y cinco del Cuadro 4 se presentan las datos medidos para cuatro implementaciones del algoritmo de filtrado desarrollado en la Sección 3.1. Nuevamente, en cada fila se presentan los resultados medidos para uno, dos, tres y cuatro niveles de análisis. De la misma forma que el caso previo, la cuarta columna muestra que la memoria de programa utilizada no varía de forma considerable para cada caso, y, en este caso la cantidad de memoria de datos permanece constante en todos los casos, esto es acorde a lo planteado en el Cuadro 3, donde mediante la sobreescritura de registros se evita incrementar el uso de memoria de datos al incrementar el nivel de análisis. Por otro lado, la quinta columna muestra que los tiempos de procesamiento de los algoritmos, si bien son inferiores a los del algoritmo de reducción de ruido, también se incrementan a medida que la cantidad de niveles crece. Esto se debe a que, también en este caso, al incrementar el nivel de análisis, se incrementa la cantidad de instrucciones del programa. Nuevamente para estos casos, los tiempos de procesamiento son menores a los de adquisición, y por lo tanto funciona en tiempo real.

Si se comparan la tercera y quinta columna del Cuadro 4, se aprecia que para cada nivel de análisis el tiempo de procesamiento para el caso del algoritmo de filtrado es menor a la mitad del tiempo de procesamiento del algoritmo de reducción de ruido. Esto se debe a que en este segundo caso, al no ser necesario calcular la señal de detalle $d_i[n]$ para cada nivel, el programa reduce notablemente tiempo de ejecución. En la Figura 1 se representa gráficamente el tiempo

de procesamiento en función de la cantidad de niveles de análisis para ambas implementaciones. En dicho gráfico se representa en trazo contínuo el caso del algoritmo de reducción de ruido, mientras que en trazo rayado se presenta el caso del filtro pasobajos.

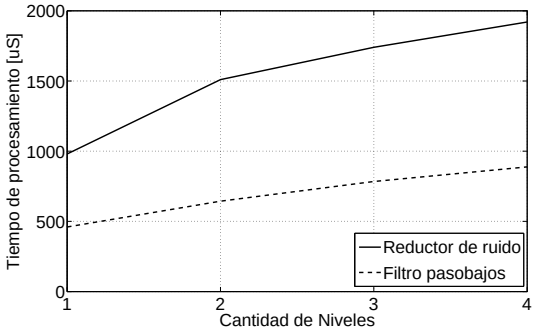

Figura 1. Tiempos de procesamiento de cada implementación.

Con el fin de analizar el rendimiento del sistema implementado, se realizaron mediciones en las cuales se excitó al sistema con señales de voz con una relación señal a ruido (SNR) de entrada conocida y se midió la relación señal a ruido a la salida del sistema.

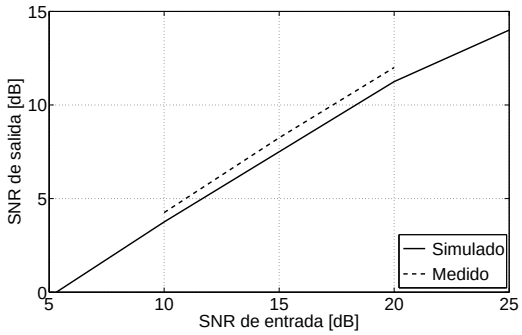

Figura 2. Curva característica del sistema.

El banco de medición está compuesto por una placa adquisidora de datos con un conversor analógico a digital con resolución de 16 bits y frecuencia de muestreo de 44100Hz. Se utilizó MATLAB tanto para generar la señal que excita el sistema bajo prueba, como también para procesar la señal adquirida. De esta forma, fue posible comparar los resultados obtenidos en las simulaciones realiza-

das en MATLAB con las mediciones realizadas en el dispositivo implementado. Se verificó en ambos casos una reducción de ruido superior a los 5dB, lo que se ajusta a los resultados obtenidos en la bibliografía [2,8]. En la Figura 2 se presenta en trazo contínuo la curva característica del sistema presentada en la literatura [8], y en trazo rayado la medida para un rango SNR de entrada entre 10db y 20dB. En la misma se puede apreciar que solo existe una leve discrepancia entre ambas curvas y que el comportamiento de ambas es similar.

5. Conclusiones

En este trabajo se realizó un estudio de la transformada wavelet discreta con el fin de implementar un algoritmo para reducción de ruido en tiempo real en un DSP de bajo costo.

Se realizó un estudio de las operaciones necesarias para implementar la TWD con el fin de reducir el uso de memoria del algoritmo propuesto. En base a las mediciones realizadas se verificó que el sistema implementado en el DSP opera en tiempo real. También se verificó que respecto al sistema base, la implementación del algoritmo reductor de ruido permite reducir el nivel de ruido en 5dB, por lo que se comporta de acuerdo a lo esperado.

Debido a que el tiempo de procesamiento es considerablemente menor al tiempo de adquisición, en el futuro se desean implementar funciones adicionales que permitan aumentar aún mas la relación señal a ruido de la señal obtenida.

Referencias

1. F. Asano, S. Hayamizu, T. Yamada y S. Nakamura, *Speech Enhancement Based on the Subspace Method*, Proceedings of IEEE Transactions on speech and audio processing, Vol.8, No.5. 2000.
2. V. Balakrishnan, N. Borgesa and L. Parchment, *Wavelet Denoising and Speech Enhancement*, 2006.
3. P. Faundez y A. Fuentes, *Procesamiento digital de señales acústicas utilizando Wavelets*, Instituto de Matemáticas UACH.
4. T. Young y W. Qiang, *The realization of Wavelet Threshold noise filtering Algorithm*,Porceedings of 2010 Conference on Meassuring Technology and Mechatronics Automation. pp 953-956. 2010.
5. F. Denk, P. Agüero, A. Uriz, J.C. Tulli, E. González, J. Garín y S. Bourguigne, *Assistive Listening Device based on a dsPIC*. Anales de VI Jornadas Argentinas de Robótica (JAR). pp.24-29, 2010.
6. A.J. Uriz, P. Agüero, J.C. Tulli, E. González y J Castiñeira, *Implementation of a noise reduction algorithm in a hearing aid device based on a dsPIC*. Anales de IEEE ARGENCON 2012, 2012.
7. Microchip Inc., *dsPIC33FJ128GPX02/X04 Data Sheet, High Performance 16-bit Digital Signal Controllers*. http://www.microchip.com/, 2009.
8. Ch. Dolabdjian, J. Fadili y E. Huertas Leyva, *Classical low-pass filter and real-time wavelet-based denoising technique implemented on a DSP: a comparison study*, The European Physical Journal Applied Physics. Vol.20, pp 135-140. 2002.
9. Microchip Inc. *16-Bit Lenguage Tools Libraries*. http://www.microchip.com/, 2005.