Revista elektron, Vol. 2, No. 2, pp. 53-62 (2018)

Implementación de un Control Activo de Ruido en Configuración Feedback basado en un Microcontrolador

Microcontroller-based Feedback Active Noise Control

Pablo Bernadí†1, Pablo Gomez†2

†*Facultad de Ingeniería, Universidad de Buenos Aires*
Paseo Colón 850, Buenos Aires, Argentina
[1]pbernadi@fi.uba.ar
[2]pgomez@fi.uba.ar

Recibido: 03/09/18; Aceptado: 09/11/18

Abstract—**This work describes an active noise control system implementation with reduced, low cost hardware using the EDU-CIAA-NXP board. The filtered-X LMS algorithm is described in its feedforward and feedback configurations, and design details for anti-alias and reconstruction filters are given. The software driving the system is described and test results in feedforward configuration are shown. Using a pure tone as the perturbing signal, a 30dB attenuation was achieved, which is adequate according to performance requirements in active noise control applications.**

Keywords: acoustics; audio; control; filtered-X LMS; EDU-CIAA-NXP.

Resumen— **Este trabajo describe la implementación de un sistema de control activo de ruido sobre la plataforma EDU-CIAA-NXP con hardware reducido y de bajo costo. Se hace una revisión del algoritmo filtered-X LMS en configuraciones feedforward y feedback, y se detalla el diseño de filtros anti-alias y de reconstrucción. Se describe el software que corre sobre la plataforma, y se muestran resultados de ensayos con la configuración feedback. Usando un tono puro como señal perturbadora se logró una atenuación de 30dB, que es adecuada según requerimientos de performance en aplicaciones de control activo de ruido.**

Palabras clave: acústica; audio; control; filtered-X LMS; EDU-CIAA-NXP.

I. INTRODUCCIÓN

Los problemas relacionados con el ruido acústico se hacen cada vez más frecuentes y evidentes a medida que aumenta el uso de equipos industriales tales como motores, ventiladores, compresores o transformadores. A su vez, el ruido urbano debido a vehículos, alarmas y maquinaria utilizada en mantenimiento residencial tiene efectos negativos sobre la calidad de sueño, concentración e incluso desarrollo en adolescentes, con impactos medibles en productividad, turismo y salud [1]. Tradicionalmente se ha encarado el problema usando métodos pasivos tales como cerramientos o paredes para atenuar el ruido, lo que da buenos resultados sobre un amplio espectro de frecuencias. Sin embargo, para el rango de frecuencias bajas, resultan menos efectivos y más costosos.

El control activo de ruido (o ANC - Active Noise Control) se basa en dispositivos electroacústicos o electromecánicos que cancelan una señal no deseada basándose en el principio de superposición, generando una señal de igual amplitud pero de fase opuesta que resulta en la cancelación de ambas señales en el punto de interés [2]. Aparecen como complemento de los métodos pasivos para bajas frecuencias [3], estando la mayoría de las aplicaciones por debajo de 1kHz. Un requerimiento de performance habitual para un sistema ANC es una atenuación de la señal perturbadora de por lo menos 10dB [4].

El diseño de sistemas ANC consiste en el uso de al menos un micrófono y un parlante alimentado con la señal de control (antiruido). Como las características de la fuente de ruido y del ambiente son no estacionarias, la implementación requiere de un sistema ANC adaptativo, lo que se traduce en un sistema digital implementado mediante filtros *FIR* o *IIR*. Los filtros adaptativos ajustan sus coeficientes en el tiempo, de modo de minimizar una señal de error, siendo el algoritmo LMS y sus variantes uno de los predilectos para esta tarea por su simpleza y bajo costo computacional [5].

En la bibliografía es más frecuente encontrar análisis de los algoritmos a nivel numérico o matemático y menos habitual la implementación de los mismos. A modo de ejemplo se analizó el trabajo de Boucher *et al.* [6] que mplementa un sistema ANC feedback sobre el DSP TMS320C50 de TI. Utilizando un microfono y un parlante de buena calidad, y muestreando con 14 bits a 600Hz, se sugiere que es posible una atenuacion de un tono puro de 40dB. También está el caso de Kuo *et al.* [7], donde se implementa un sistema similar sobre el TMS320C32 promediando la salida de 3 micrófonos de alta calidad dispuestos triangularmente y muestreando con 16 bits a 1kHz, con lo que se obtuvo una reducción de hasta 60dB. Sistemas de mayor complejidad, como los que buscan atenuar ruidos de banda ancha, requieren implementaciones en PC [8] [9].

En el presente trabajo se busca alcanzar una performance aceptable utilizando elementos de bajo costo: micrófono y parlante de bajo costo y conversores A/D y D/A integrados en el microcontrolador. Usando un tono puro como señal perturbadora, se alcanzó una atenuación de 30dB. El trabajo se estructura de la siguiente manera. En la sección II se presentan los algoritmos utilizados para problemas de control activo de ruido. En las secciones III y IV se detalla el diseño del hardware y el software involucrados, en la sección V se describen los ensayos realizados y los resultados obtenidos, y finalmente en la sección VI se presentan las conclusiones.

A. Configuraciones para ANC

Existen dos métodos clásicos para efectuar el control de ruido. El primero de ellos, llamado feedforward, más común en la mayoría de las aplicaciones, consiste en el uso de dos micrófonos y un parlante. El segundo, llamado feedback, depende de un único micrófono para identificar y cancelar el ruido.

1) Configuración Feedforward: La figura 1 muestra el diagrama en bloques para la configuración feedforward, en la cual se utiliza un micrófono para medir el ruido $r(n)$ cerca de su fuente, y un segundo micrófono que mide el error $e(n)$ en el punto donde se desea lograr la atenuación. El ruido es transformado por la respuesta impulsiva del recinto dando origen a $\tilde{r}(n)$, que interfiere con la señal deseada $x(n)$. El filtro adaptativo $\mathbf{w}(n)$ busca caracterizar la respuesta impulsiva del recinto para generar una señal de control $y(n) = -\tilde{r}(n)$ que permita cancelar el ruido.

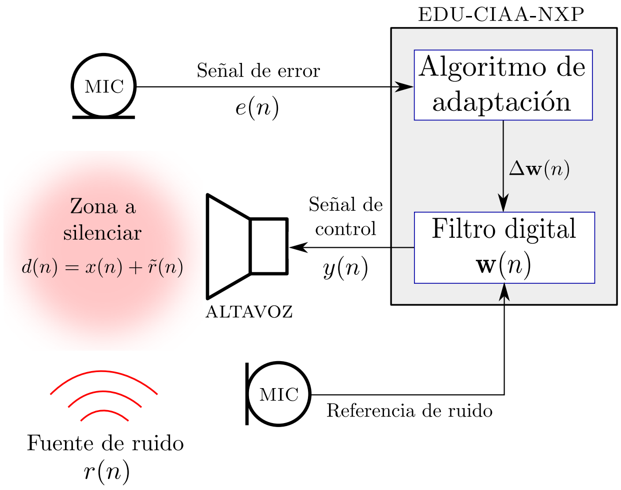

Fig. 1. Esquemático del sistema en configuración feedforward.

2) Configuración Feedback: Hay ocasiones en que no es posible medir una referencia independiente del ruido, por lo que el algoritmo adaptativo debe estimarla a partir del micrófono de error. La complejidad en el software se ve incrementada por esta razón, como se verá en la sección siguiente. En la figura 2 se muestra un esquema para configuración feedback.

II. ALGORITMOS LMS Y FILTERED-X LMS

Como se mostró en la sección I-A, en ambas configuraciones para ANC se depende generalmente del algoritmo LMS (Least Mean Square), desarrollado en 1960 por Widrow y Hoff [10], que busca minimizar iterativamente el error cuadrático medio entre la señal deseada y la medida. Esto se logra convolucionando el ruido $\mathbf{r}(n) = [r(n)\ r(n-1)\ \cdots\ r(n-M+1)]^T$ (o una estimación del mismo) con el filtro $\mathbf{w}(n) = [w(0)\ w(1)\ \cdots\ w(M-1)]^T$, y usando el método de *Steepest Descent* para optimizar sus coeficientes [11] con dicho criterio. En el problema considerado, sólo es posible medir el error $e(n)$, ya que la superposición entre la señal de interés $x(n)$, el ruido $\tilde{r}(n)$ y la señal de control $y(n)$

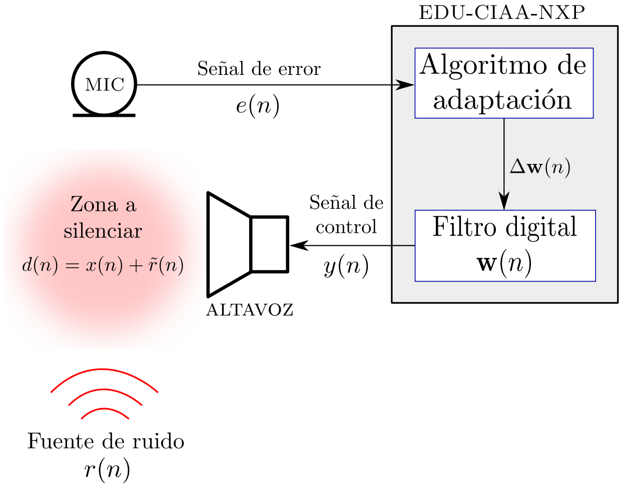

Fig. 2. Esquemático del sistema en configuración feedback.

sucede en el campo acústico. En principio, esto simplifica la implementación al no existir necesidad de realizar el cálculo del error dentro del microcontrolador.

El algoritmo se maneja sobre dos supuestos importantes

- procesos de media nula
- correlación nula entre el ruido y la señal de interés

El algoritmo LMS consiste en los siguientes pasos iterados en el tiempo:

1) Determinar el error $e(n) = x(n) + \tilde{r}(n) + y(n) = x(n) + \tilde{r}(n) + \mathbf{w}^T(n)\mathbf{r}(n)$
2) Actualizar los coeficientes del filtro $\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{r}(n)$

donde μ es el paso del algoritmo. Idealmente, se desea que $\mathbf{w}^T(n)\mathbf{r}(n) = -\tilde{r}(n)$, aunque en realidad se obtendrá su estimación. Para asegurar la convergencia del algoritmo, se establece la condición

$$0 < \mu < \frac{2}{\mathrm{tr}(E[\mathbf{r}\mathbf{r}^T])} \tag{1}$$

que depende fuertemente de la señal de entrada, por lo que existe una limitación cuando su magnitud no es conocida. Para salvar esta dificultad, se realiza una modificación en la actualización del filtro $\mathbf{w}$, dividiéndola por la norma al cuadrado de la señal de referencia. Esto resulta en el algoritmo NLMS (Normalized Least Mean Square), que consiste en los siguientes pasos:

1) Determinar el error $e(n) = x(n) + r(n) + \mathbf{w}^T(n)\mathbf{r}(n)$
2) $\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\mu e(n)}{a+||\mathbf{r}(n)||^2}\mathbf{r}(n)$

donde a es una constante de regularización. La condición sobre μ en este caso resulta

$$0 < \mu < 2 \tag{2}$$

y la convergencia mejora gracias a la independencia del nivel de señal [5].

Existe una consideración principal que impide implementar este esquema exitosamente en un contexto de cancelación de ruido. La señal de control generada digitalmente es convertida a una señal analógica, y actuada por medio de un circuito amplificador y un parlante. Más aún, la medición del error recorre un camino similar aunque inverso, siendo tomada por un micrófono, acondicionada por un circuito amplificador y luego convertida a una señal digital. Esta cadena que comienza con una señal digital, atraviesa un sistema eléctrico/acústico/eléctrico y termina nuevamente con una señal digital es llamada *camino secundario* e introduce un retardo. Si no se compensa el efecto del camino secundario sobre las señales, no será posible obtener una señal de control adecuada [3].

El algoritmo FXLMS modifica lo visto hasta ahora, identificando el camino secundario e incorporándolo al esquema. Esto se hace previo a la fase de control, en ausencia de otras señales (es decir $x(n) = r(n) = 0$), generando una secuencia de ruido blanco $y'(n)$ que excita al parlante, y es medido por el micrófono de error. En este caso es posible usar los algoritmos LMS o NLMS tal como se describieron para estimar la respuesta del camino secundario, ya que se conoce exactamente el vector de referencia $y'(n)$, como se verá en la sección IV-C. Una vez identificada esta respuesta impulsiva, comienza la fase de control, durante la cual es posible realizar la adaptación de los coeficientes del filtro mediante los algoritmos LMS o NLMS. Por simplicidad y unicidad en la presentación, de aquí en adelante se describirán configuraciones basadas en el algoritmo LMS.

A. FXLMS en configuración feedforward

La figura 3 muestra el diagrama en bloques para el algoritmo FXLMS en configuracion feedforward. $S(z)$ es la respuesta del camino secundario, mientras que $\hat{S}(z)$ es su estimación (identificada previamente de forma offline). $P(z)$ es la respuesta del recinto y $\hat{P}(z)$ su estimación online, contenida en el filtro adaptado $\mathbf{w}(n)$. La presencia del bloque $\hat{S}(z)$ se justifica por la introducción del retardo asociado al camino secundario [12], y frente a la alternativa de anteponer a la salida del sistema un bloque $\hat{S}^{-1}(z)$ que compense este efecto, inversa que no necesariamente existe.

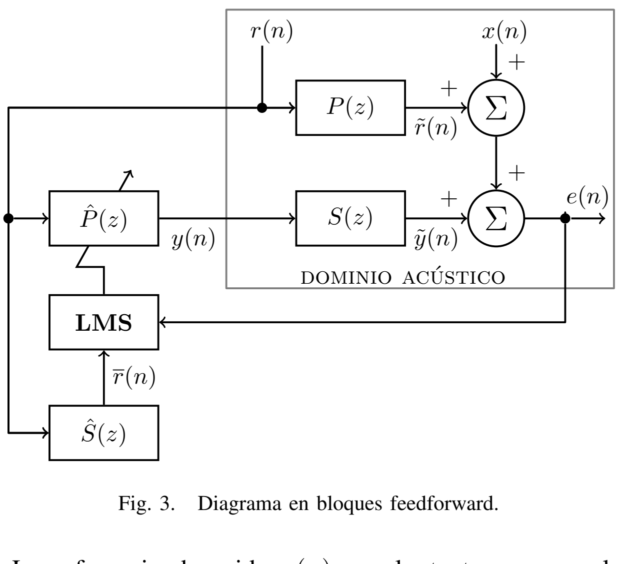

Fig. 3. Diagrama en bloques feedforward.

La referencia de ruido $\mathbf{r}(n)$ por lo tanto, es convolucionada con $\hat{\mathbf{s}}$ antes de ser utilizada para la actualización de los coeficientes de $\mathbf{w}(n)$. La medición de la superposición de todas las señales en juego es $e(n)$. El algoritmo FXLMS feedforward consiste entonces en:

1) Calcular la señal de control $y(n) = \mathbf{w}^T(n)\mathbf{r}(n)$
2) Calcular $\overline{r}(n) = \hat{\mathbf{s}}^T\mathbf{r}(n)$
3) Actualizar los coeficientes del filtro $\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\overline{r}(n)$

TABLA I
NÚMERO DE OPERACIONES DE PUNTO FLOTANTE PARA FXLMS FEEDFORWARD

Ecuación	$\times$	$+$	Operaciones
$\mathbf{w}^T(n)\mathbf{r}(n)$	M	$M-1$	$2M-1$
$\hat{\mathbf{s}}^T\mathbf{r}(n)$	L	$L-1$	$2L-1$
$\mathbf{w}(n)+\mu e(n)\overline{r}(n)$	M	$M-1$	$2M-1$
Total	$2M+L$	$2M+L-3$	$4M+2L-3$

Este algoritmo requiere entonces actualizar con cada nueva muestra los vectores $\overline{\mathbf{r}}(n)$, $\mathbf{w}(n)$ de longitud M, y mantener una copia constante de $\hat{\mathbf{s}}$, de longitud L. La cantidad de operaciones de punto flotante necesarias en esta configuración se detalla en la tabla I. Si se muestrea con una frecuencia $f_s = 2f_{max}$, entonces la cantidad de operaciones de punto flotante por segundo necesarias es

$$N_{flops} = 2f_{max}(4M + 2L - 3) \tag{3}$$

B. FXLMS en configuración feedback

Si no se dispone de la referencia de ruido $r(n)$, debe ser estimada. La adaptación de $\mathbf{w}(n)$ buscará minimizar $e(n)$, lo que se logra cuando $\tilde{y}(n) \simeq -\tilde{r}(n)$, por lo que la misma señal de control contendrá una referencia del ruido. Podemos obtener una estimación de $\tilde{y}(n)$ mediante $\overline{y}(n) = \hat{s} * y(n)$, por lo que la estimación de la referencia de error es simplemente $\hat{r}(n) = e(n) - \overline{y}(n)$. La figura 4 muestra el diagrama en bloques para esta configuración.

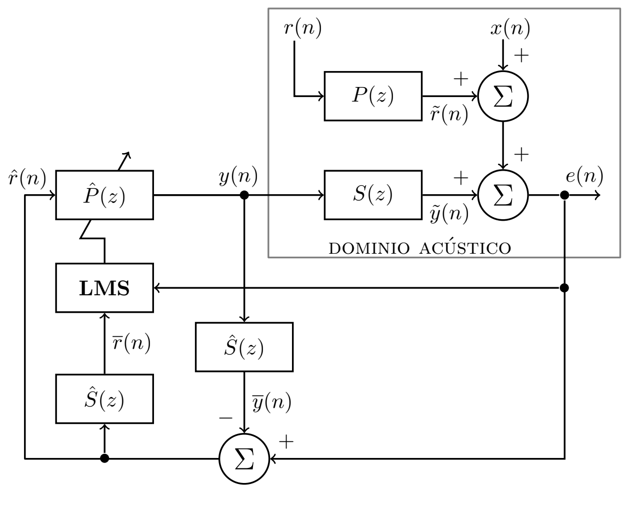

Fig. 4. Diagrama en bloques feedback.

El algoritmo FXLMS en configuración feedback resulta en los siguientes pasos:

1) Calcular la señal de control $y(n) = \mathbf{w}^T(n)\hat{\mathbf{r}}(n)$
2) Estimar $\hat{r}(n) = e(n) - \hat{\mathbf{s}}^T\mathbf{y}(n)$
3) Calcular $\overline{\mathbf{r}}(n) = \hat{\mathbf{s}}^T\hat{\mathbf{r}}(n)$
4) Actualizar los coeficientes del filtro $\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\overline{\mathbf{r}}(n)$

En este caso, además de guardar $\hat{\mathbf{s}}$, $\mathbf{w}(n)$ y $\hat{\mathbf{r}}(n)$, debe guardarse $\mathbf{y}(n)$, también de longitud L. La cantidad de operaciones de punto flotante por segundo para esta configuración es

$$N_{flops} = 8(M + L - 1)f_{max} \tag{4}$$

TABLA II
NÚMERO DE OPERACIONES DE PUNTO FLOTANTE PARA FXLMS FEEDBACK

Ecuación	$\times$	$+$	Operaciones
$\mathbf{w}^T(n)\mathbf{r}(n)$	M	$M-1$	$2M-1$
$e(n)-\hat{\mathbf{s}}^T\mathbf{y}(n)$	L	$L-1$	$2L-1$
$\hat{\mathbf{s}}^T\mathbf{r}(n)$	L	$L-1$	$2L-1$
$\mathbf{w}(n)+\mu e(n)\bar{\mathbf{r}}(n)$	M	$M-1$	$2M-1$
Total	$2M+2L$	$2M+2L-4$	$4(M+L-1)$

III. HARDWARE

Para realizar el procesamiento de las señales y el algorítmo se utilizará como plataforma la EDU-CIAA-NXP. El proyecto CIAA [13] surgió de la necesidad de mejorar la competitividad de PyMEs e industrias gracias a la incorporación de electrónica en sus procesos, lo que llevó al desarrollo de la Computadora Industrial Abierta Argentina. La placa EDU-CIAA-NXP fue concebida para ser utilizada en educación, y cuenta con un procesador Cortex M4F/M0 con FPU con una potencia de hasta 204 Mflops y una multitud de GPIO, entre los que encontramos 3 coversores A/D y uno D/A de 10 bits con frecuencias máximas de funcionamiento de 400 kHz, lo que cubre todas las necesidades de nuestro sistema.

Asumiendo una frecuencia de actuación máxima de $f_{max} = 500Hz$, según las ecuaciones 3 y 4, es posible obtener filtros de orden $M = L = 34000$ para el caso de la configuración feedforward, y de $M = L = 25500$ para la configuración feedback. Esto representa retardos de hasta 34s y 25s respectivamente, mientras que los ejemplos en la literatura lidian con retardos menores a los 500ms. Por ejemplo, en [6] se utilizan filtros de tamaños $M = 16$ Y $L = 256$ muestreando a 600Hz; en [7] se usan filtros de $M = 256$ y $L = 128$ muestreando a 1kHz. Para esta aplicación se ha considerado adecuada una frecuencia de muestreo de 1kHz, lo que permite actuar hasta 500Hz.

La performance de un sistema ANC depende fuertemente de la identificación del camino secundario [3], por lo que el tamaño del filtro $\hat{\mathbf{s}}$ se fijó en $L = 256$ coeficientes, lo que permite caracterizar un retardo del recinto de hasta 256/1000Hz=256ms. Por otro lado, se asumió una longitud para el camino principal mucho más corta, y el tamaño del filtro $\mathbf{w}$ se fijó en $M = 64$ coeficientes, lo que permite caracterizar el retardo entre la fuente de ruido y el micrófono. Según la ecuación 4 se requerirán 1,276 Mflops, lo que representa aproximadamente el 0,6% de la capacidad de cálculo disponible.

La interacción del algoritmo FXLMS sobre la EDU-CIAA-NXP y el plano acústico se realiza mediante los puertos ADC, tomando mediciones de uno o dos micrófonos, dependiendo de la implementación elegida. Por otro lado, la señal de control es alimentada a un parlante por medio del puerto DAC. Ambas interfaces requieren de acondicionamiento de señal y frecuencias de corte para evitar aliasing en la conversión A/D o imágenes en la conversión D/A. Para ambos casos, se usó un circuito basado en la nota de aplicación [14]. La figura 5 da el diagrama en bloques del sistema propuesto.

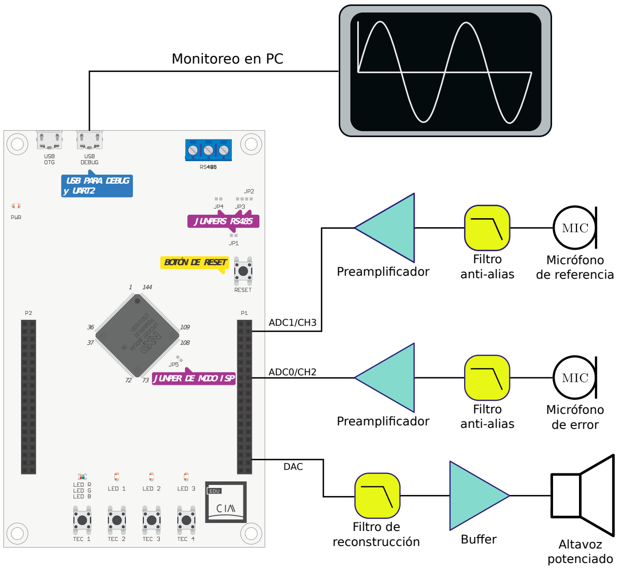

Fig. 5. Esquemático del sistema propuesto.

A. Preamplificador de micrófono para ADC

La figura 6 muestra el circuito utilizado como preamplificador de micrófono y filtro anti-alias para la entrada del ADC.

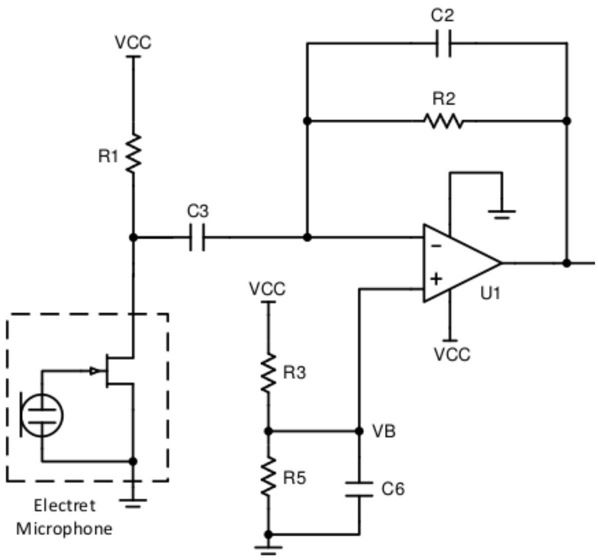

Fig. 6. Circuito preamplificador/filtro anti-alias.

Los puertos analógicos usan como referencia una tensión de 3,3V, por lo que se decidió usar esa tensión para alimentar los circuitos. Esto requiere que la señal de entrada tenga un rango de 0V a 3,3V, lo que implica usar un bias de 1,65V para el amplificador. Esto se logra si, en el circuito de la figura 6, R3=R5. Por otro lado, para que la sensibilidad del micrófono sea buena, es necesario que R1 sea grande frente a C3 en el rango de frecuencias de interés, para que la señal generada por el micrófono tenga buen nivel en la entrada del amplificador operacional.

Para evitar aliasing en el muestreo, se requiere que el polo formado por R2 y C2 tenga un valor tal que atenúe las frecuencias fuera del rango de interés hasta niveles despreciables. Con todas estas restricciones, los valores elegidos para los componentes resultaron los que se muestran en la tabla III.

TABLA III
COMPONENTES PARA EL PREAMPLIFICADOR DEL ADC

Componente	Valor	Comentarios
R1	$10k\Omega$	Da buena sensibilidad al micrófono electret
R3, R5	$330k\Omega$	Tolerancia de 1% para minimizar bias y evitar saturación
C3	$10\mu F$	El polo generado con R1 queda en $f = 1,6Hz$ (pasa-altos)
R2	$330k\Omega$	Ganancia del amplificador = 33
C2	$100nF$	El polo generado con R2 queda en $f = 4,8Hz$ (pasa-bajos)
C6	$100\mu F$	El polo generado con R3 y R5 queda en $f = 0,1Hz$ (pasa-altos)

El polo formado por R2 y C2 se ubica en 4,8Hz, una frecuencia muy baja, pero debe tenerse en cuenta que se trata de un filtro de primer orden, por lo que su atenuación es de 20dB por década. Esto da una atenuación de 40dB, o 100 veces a una frecuencia de 480Hz, lo que se considerará la frecuencia de corte.

B. *Buffer para DAC*

El diseño de la interfaz DAC/parlante (figura 7) sigue criterios similares a los utilizados en la sección anterior. En este caso, la entrada al circuito está montada sobre 1,65V, y la salida está centrada en 0V. La ganancia es unitaria para ser compatible con niveles de audio, y se implementa nuevamente un filtro pasa bajos, llamado filtro de reconstrucción, para evitar frecuencias espúreas generadas por el proceso de conversión D/A.

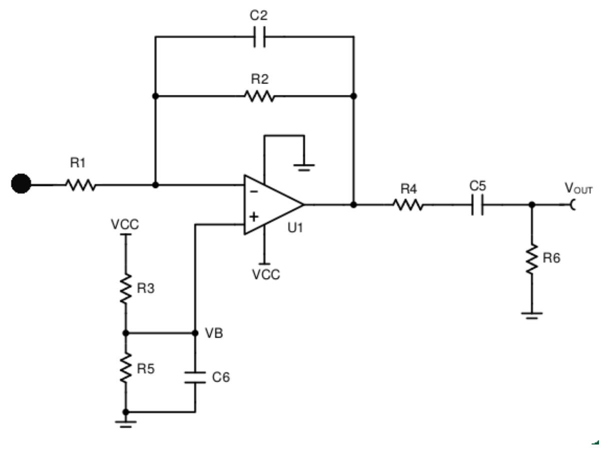

Fig. 7. Circuito preamplificador/filtro anti-alias.

Algunos componentes mantienen sus valores, como la red de bias. Se agrega un capacitor de desacople C5 a la salida, junto con una resistencia de carga R6. Los valores para los componentes se dan en la tabla IV.

Nuevamente, el polo generado por R2 y C2 queda en una frecuencia baja, pero se busca evitar problemas con frecuencias espúreas.

El amplificador operacional elegido fue el MCP6002, que es rail to rail y funciona con tensiones bajas compatibles con los 3,3V provistos por la EDU-CIAA-NXP.

TABLA IV
COMPONENTES PARA EL BUFFER DEL DAC

Componente	Valor	Comentarios
R1, R2	$10k\Omega$	Ganancia unitaria
C2	$1\mu F$	El polo generado con R2 queda en $f = 15,9Hz$ (pasa-bajos)
R3, R5	$330k\Omega$	Tolerancia de 1% para minimizar bias y evitar saturación
C3	$10\mu F$	El polo generado con R1 queda en $f = 1,6Hz$ (pasa-altos)
R4	47Ω	
R6	$100K\Omega$	Resistencia de carga, grande frente a impedancia de entrada de eventual etapa de potencia
C5	$10\mu F$	El polo generado con R6 queda en $f = 0,16Hz$ (pasa-altos)

El actuador es un parlante potenciado Microlab B-77, con altavoces de 4” y 0,75”, con una potencia de 24W RMS y $55Hz < Bw < 20kHz$.

IV. SOFTWARE

El funcionamiento del sistema ANC desde el punto de vista del software depende de dos factores principales:

- interfaces A/D y D/A
- algoritmo FXLMS

A. *Conversión A/D y D/A*

Se busca que el algoritmo FXLMS trabaje con números de punto flotante que mantengan relación con las tensiones de las señales, entre -1,65V y 1,65V, mientras que las muestras de los conversores A/D son números entre 0 (0V) y 1023 (3,3V). Para obtener valores en el rango deseado a la medición se le resta un bias de 511 y se la escala. Lo mismo se hace con la conversión D/A: los valores calculados internamente se encuentran entre -1,65 y 1,65, a los que se les suma un bias y se los escala para obtener números en el rango 0 a 1023.

Para tomar muestras de dos pines ADC distintos sin interrumpir constantemente el programa principal, se optó por usar dos buffers DMA, cada uno asignado a un pin diferente. Cuando estos buffers se llenan, generan interrupciones que activan flags independientes. Si ambos flags están activos, se promedian todos los datos en cada buffer para obtener dos mediciones: $e(n)$ (en mabas configuraciones) y $r(n)$ (en configuración feedforward). Una vez que se completa esta fase, se puede alimentar el algoritmo FXLMS con los nuevos datos.

La EDU-CIAA-NXP permite muestrear a frecuencias mucho más altas que las necessarias para esta aplicación, lo que se aprovechó para reducir el ruido de cuantización en los datos del ADC, de resolución muy limitada. El efecto teórico de sobremuestrear por un factor de 100 $100kHz/(2 \times 480Hz) \simeq 104$ permite obtener una resolución de $10 + \log_2(2 \times 104) = 13,35$ bits, lo que equivale a una relación señal a ruido teórica [15] de

$$SNR = 6,02N + 1,76dB + 10\log_{10}104 = 82,13dB \quad (5)$$

Esto agrega 100000 sumas (y 1000 multiplicaciones) por segundo por cada señal, lo que no impacta significativamente en la cantidad de operaciones realizadas.

El DAC, por otro lado, se opera en modo Polling, pasándole resultados de la función generadora de ruido o la señal de control calculada por el algoritmo FXLMS. La capacidad de cómputo de la unidad de punto flotante disponible en la EDU-CIAA-NXP permite operar el DAC de esta forma, sin necesidad de generar interrupciones.

B. Algoritmo FXLMS

El algoritmo FXLMS es esencialmente un conjunto de operaciones vectoriales, por lo que se desarrolló una pequeña biblioteca para operar con arrays de variables *float*. Las operaciones incluidas en la biblioteca son:

- `vector_initialize()`: inicializa todos los elementos de un array en un mismo valor, ambos pasados como argumento.
- `vector_copy()`: copia los elementos de un vector a otro.
- `vector_shift()`: desplaza los elementos de un vector una posición, para liberar la posición 0, que se usa para cargar un nuevo dato.
- `vector_scale()`: multiplica todos los elementos de un array por un escalar, ambos pasados como argumento.
- `vector_add()`: suma dos vectores.
- `vector_dot()`: calcula el producto escalar entre dos vectores (al pasar dos veces el mismo vector como argumento, se obtiene su norma al cuadrado).
- `vector_add_scaled()`: escala un vector y lo suma con otro.

En todos los casos, la dimensión del vector es pasada como argumento también. Esto resulta útil cuando se debe calcular el producto vectorial de vectores con longitudes diferentes (por ejemplo $\mathbf{w}^T(n)\hat{\mathbf{r}}(n)$), ya que evita tener que realizar copias de igual longitud, pudiendo operar sobre los primeros elementos del de mayor longitud.

El conjunto de operaciones vectoriales desarrollado permite implementar fácilmente todos los algoritmos mencionados (LMS, NLMS, FXLMS, y FXNLMS)

C. Programa Principal

El diagrama de flujo del programa principal (figura 8) consiste en tres fases. La primera de ellas consiste en la identificación del nivel de bias de los conversores ADC en reposo, esencial para asegurar que las señales de entrada tengan media nula. Una vez completada, se procede a identificar el camino secundario $\hat{\mathbf{s}}$, y finalmente se pasa a la fase de control, que consiste en un loop infinito.

La fase de medición de bias en el ADC es necesaria porque el algoritmo FXLMS asume media nula en todas sus señales. Se pudo observar que, de no compensarse, el sesgo en la señal de control crece sin control. Esta fase consiste simplemente en el promedio de un número grande (~ 10000) de muestras de los pines ADC de la EDU-CIAA-NXP.

1) Identificación del camino secundario: La fase de identificación del camino secundario consiste en la generación de un ruido blanco $y'(n)$, medición del micrófono $m(n)$ y posterior aplicación del algoritmo LMS para la actualización de los coeficientes del filtro $\hat{\mathbf{s}}$. Una vez que termina este proceso de identificación, el filtro hallado no se modifica nuevamente, por lo que no es conveniente mover el micrófono de error o el parlante de lugar. En la figura 9 se muestra el diagrama de flujo para esta etapa.

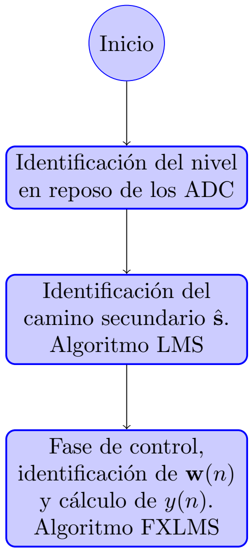

Fig. 8. Diagrama de flujo del programa principal.

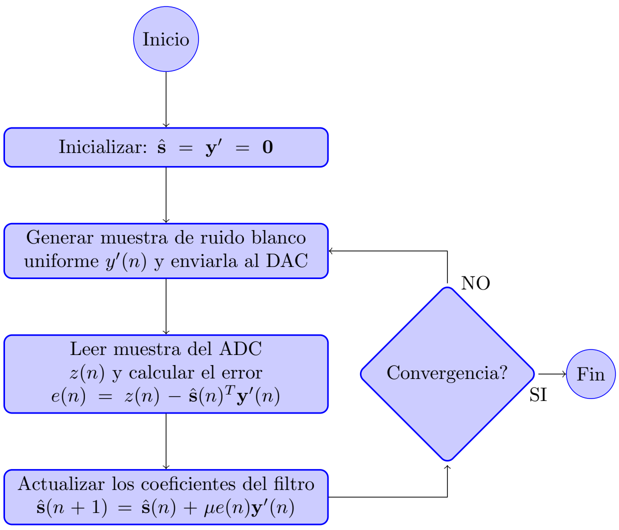

Fig. 9. Diagrama de flujo para la identificación del camino secundario.

Debe mencionarse que al generar la secuencia de ruido blanco con la funcion `rand()`, comúnmente utilizada para obtener muestras aleatorias, se encontró que la señal de salida era altamente armónica, es decir, la secuencia se repetía periódicamente, lo que se evidenciaba en tonos constantes en el parlante. También se pudo verificar esto realizando la FFT sobre la secuencia, observando picos en frecuencias muy definidas.

Se encontró en foros de este tema una función en C que genera ruido blanco uniforme [16], lo que se verificó analizando la FFT sobre la nueva secuencia, confirmando que la potencia era plana para todas las frecuencias. La diferencia al oído también fue muy marcada, ya que no se identificaron tonos dominantes.

2) Fase de control: Durante la fase de control, se corre un loop infinito que toma mediciones de los pines ADC, genera la señal de control $y(n)$ que es pasada al DAC, y actualiza los coeficientes del filtro $\mathbf{w}(n)$ usando el algoritmo FXLMS. En la figura 10 se muestra el diagrama de flujo de este proceso usando la configuración feedback.

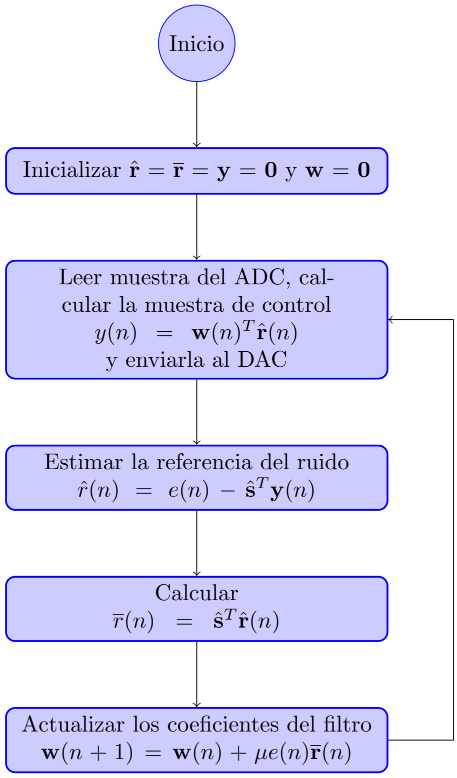

Fig. 10. Diagrama de flujo para la fase de control.

V. Ensayos y Resultados

En esta sección se describen los ensayos realizados para validar el hardware y el software, detallando los resultados logrados y algunos problemas resueltos.

A. *Pruebas funcionales del software*

En una primera instancia se realizó una validación de la librería de álgebra y de los algoritmos LMS y NLMS sobre la EDU-CIAA-NXP simulando una señal por el puerto de debug. Mediante un script de Python sobre una PC se generó una señal senoidal $x(n)$ de amplitud 1 contaminada por ruido gaussiano $g(n) \sim N(0,1)$ transformado por una respuesta impulsiva aleatoria h de 4 coeficientes (figura 11). Los únicos datos transmitidos a la EDU-CIAA-NXP fueron la señal ruidosa, y la referencia del ruido. Transmitiendo datos de la EDU-CIAA-NXP a la PC y utilizando un software de visualización en tiempo real (en ventanas de 10000 muestras) también escrito en Python se verificó que los algoritmos LMS y NLMS funcionaron correctamente, identificando la respuesta impulsiva $\hat{h}$ que afectaba al ruido y filtrándolo, como se muestra en la figura 12.

Las funciones para manejo de ADC se validaron fijando en un principio tensiones constantes y conocidas en los canales de interés, y modificando los ejemplos provistos en [17] para leer esos datos. En el caso del DAC, se comenzó utilizando el ejemplo, que genera un diente de sierra, que pudo medirse con un osciloscopio. Luego se lo modificó para generar ruido blanco y otras señales.

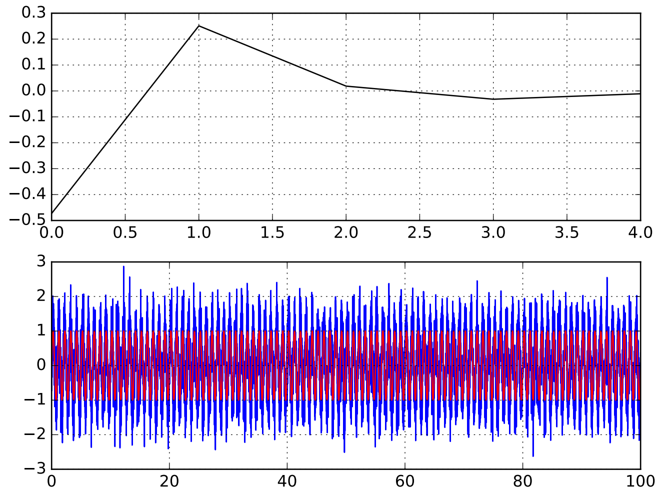

Fig. 11. Respuesta impulsiva aleatoria $\mathbf{h}$ y señales $x(n)$ (rojo) y $x(n) + g(n) * h(n)$ (azul).

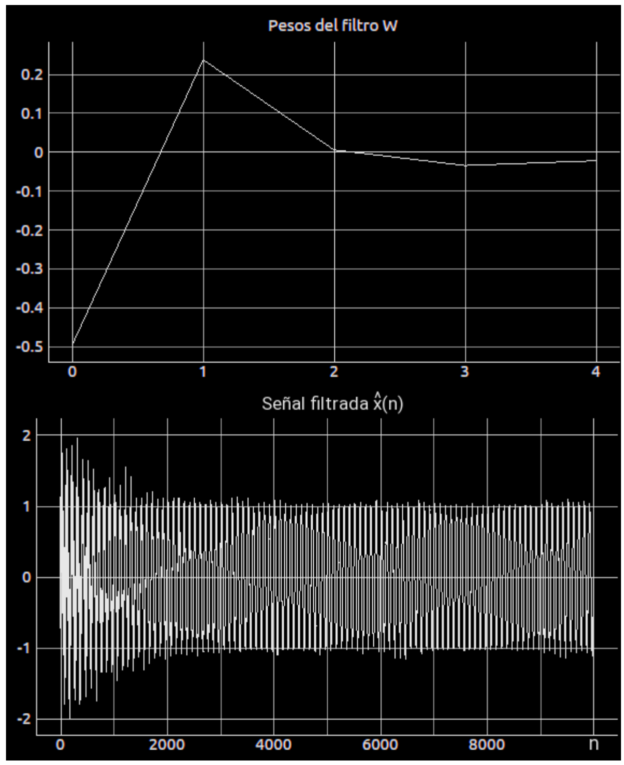

Fig. 12. Respuesta impulsiva estimada $\mathbf{w} \simeq \mathbf{h}$ y señal filtrada $\hat{x}(n)$ por el algoritmo LMS.

B. *Pruebas funcionales del hardware*

Para probar el hardware, se contó con las funciones de lectura de canales ADC y generación de datos por DAC, lo que simplificó las pruebas en forma considerable. Inicialmente, con los micrófonos conectados a ADC0/CH2 y ADC1/CH3 y usando el visualizador en la PC, se verificó que los micrófonos respondían a estímulos.

El paso siguiente fue intentar reproducir las mediciones de los micrófonos en el parlante, lo que utiliza el DAC y el buffer de salida. En este caso, la EDU-CIAA-NXP funciona sólo como una repetidora. Al realizar esta prueba se encontró que los valores elegidos en un primer momento para las frecuencias de corte en los operacionales no eran adecuados, ya que el efecto del aliasing era muy importante. Se decidió reducir considerablemente el ancho de banda del sistema, lo que llevó a los valores calculados en la sección III-A. Esto dio buenos resultados, pudiendo verificar con tonos variables en frecuencia que la entrada al micrófono era reproducida en el parlante.

En la figura 13 se muestra la conexión de la EDU-CIAA-NXP con el preamplificador de micrófono y el buffer del DAC, ambos en la misma placa. Esta configuración corresponde a un sistema ANC de tipo feedback.

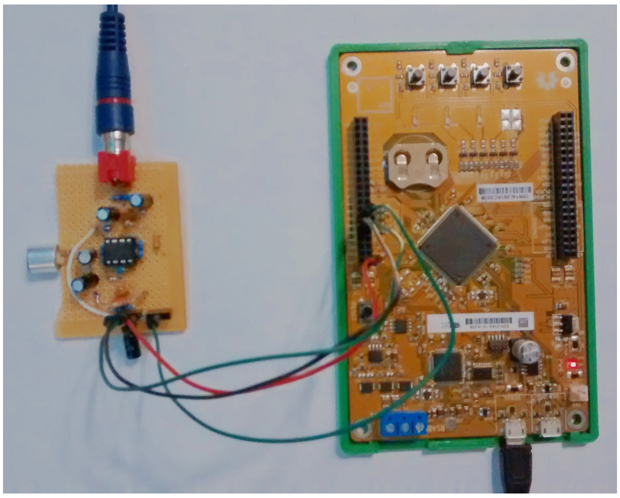

Fig. 13. EDU-CIAA e interfaces micrófono/ADC y DAC/parlante.

C. Ensayos y ajuste de parámetros

Durante la fase de identificación del camino secundario, se colocó el micrófono frente al parlante de control a distancias de entre 20 y 30 cm. Se utilizó el algoritmo NLMS con buenos resultados fijando el paso para esta etapa en $\mu = 0,05$ dado el buen compromiso de este factor entre velocidad de convergencia y error estacionario. La identificación de la respuesta impulsiva del camino secundario resultó ser repetible, lo que confirmaba la robustez del método utilizado. En la figura 14 se puede ver la señal de error cayendo en amplitud a medida que se identifica el camino secundario, y en la figura 15 la respuesta en frecuencia estimada. Las escalas en dB están referidas al nivel de contínua de la señal de error en reposo.

Luego de la identificación comienza la fase de control, en que la señal de prueba o ruido a cancelar fue un tono de 75Hz, que daba buen nivel de señal (aproximadamente 0,4V pico) en el micrófono aún a distancias de varios metros. Se encontraron diversos problemas para ajustar el paso en el algoritmo FXNLMS en la configuración feedforward, mientras que el FXLMS en configuración feedback funcionó sin mayores complicaciones. Se observó que una cantidad de coeficientes menor a 64 para $\mathbf{w}$ impedía la convergencia de la señal de control. El algoritmo convergía para $\mu < 0,0005$, efectivamente logrando una atenuación del tono de prueba casi hasta valores de reposo.

Con $\mu = 5 \times 10^{-6}$ el nivel de la señal de control se estabilizó en torno a 0,65V pico. Esto es el 40% de la amplitud disponible, por lo que el sistema tiene en principio la capacidad de controlar ruidos con mayor potencia. El filtro

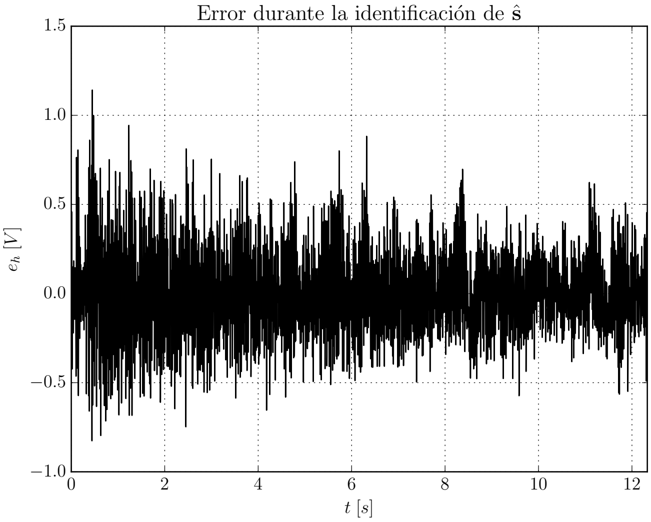

Fig. 14. Señal de error $e(n)$ durante la identificación del camino secundario

Fig. 15. Respuesta en frecuencia $\hat{S}$ estimada para el camino secundario.

$\mathbf{w}(n)$ estimado muestra un pico en torno a 69Hz (figura 16), identificando aproximadamente la frecuencia a cancelar. La discrepancia en la frecuencia estimada puede explicarse por la baja cantidad de coeficientes usados. En la figura 17 se muestra la respuesta en frecuencia de la señal de control, generada correctamente como una senoidal en torno a 75Hz.

La figura 18 muestra la respuesta en frecuencia de la señal de error en tres instancias. E_r es el nivel de señal en reposo, en ausencia del tono de prueba y de la señal de control; E_n es el nivel de señal cuando se enciende la fuente de ruido, y E_c es el nivel de señal durante la fase de control, luego de la convergencia del algoritmo. Se logró una atenuación del tono de prueba de 30dB, a la vez que baja el nivel de ruido a través del espectro. No se logra atenuar el armónico presente en 150Hz y se amplifica levemente el armónico presente en 225Hz, lo que puede adjudicarse a la poca sensibilidad del filtro anti-alias en altas frecuencias. El resultado satisface los requerimientos para sistemas ANC prácticos, aún con un conversor A/D de baja resolución integrado al controlador y con un micrófono electret.

Se destacan las siguientes observaciones luego de diversas

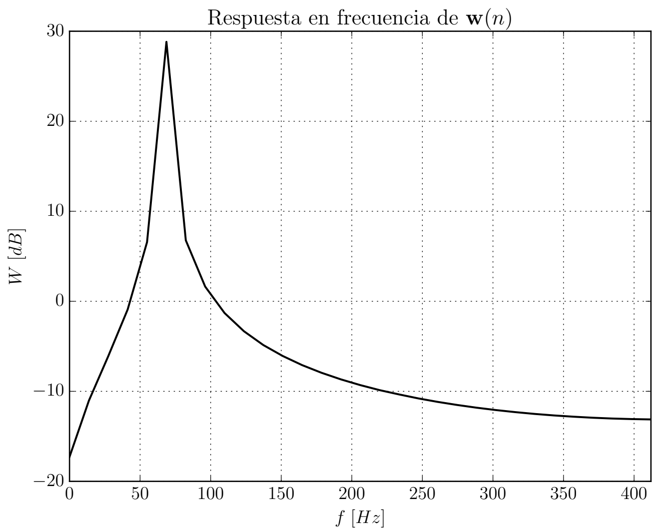

Fig. 16. Respuesta en frecuencia del filtro $\mathbf{w}$ luego de la convergencia.

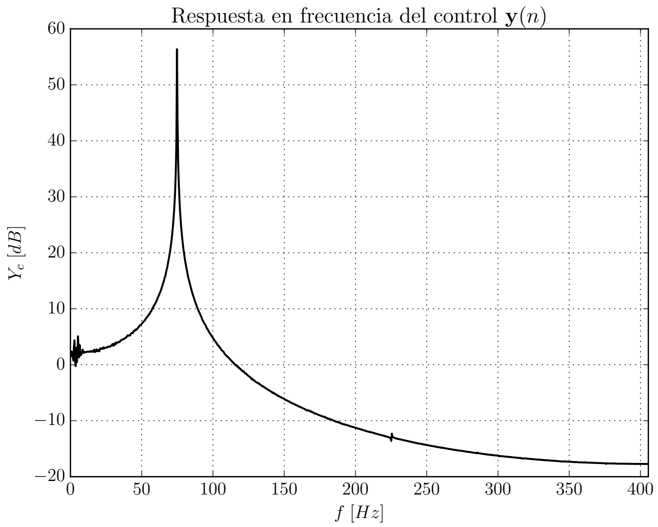

Fig. 17. Respuesta en frecuencia de la señal de control $\mathbf{y}(n)$ luego de la convergencia. La señal de control coincide en frecuencia con la señal perturbadora.

pruebas:

- Aumentar el tamaño del filtro $\mathbf{w}(n)$ acelera la convergencia pero no la asegura.
- El valor del paso μ es crítico para el algoritmo, ya que pequeños cambios en él pueden producir divergencia u oscilaciones.
- El algoritmo NLMS produce mejores (mayores) valores iniciales para la señal de control $\mathbf{y}(n)$, pero no siempre mayor velocidad de convergencia.
- La convergencia depende fuertemente de las condiciones del recinto; se han realizado pruebas idénticas con resultados muy dispares en cuanto a la convergencia. Un momento particularmente crítico es cuando el nivel de señal del parlante comienza a ser comparable al de la perturbación. A veces el nivel de la señal de control $\mathbf{y}(n)$ no se estabiliza y continúa creciendo hasta que el algoritmo diverge, lo que posiblemente tenga que ver con la longitud del vector $\mathbf{w}$ y con la respuesta del parlante utilizado en bajas frecuencias

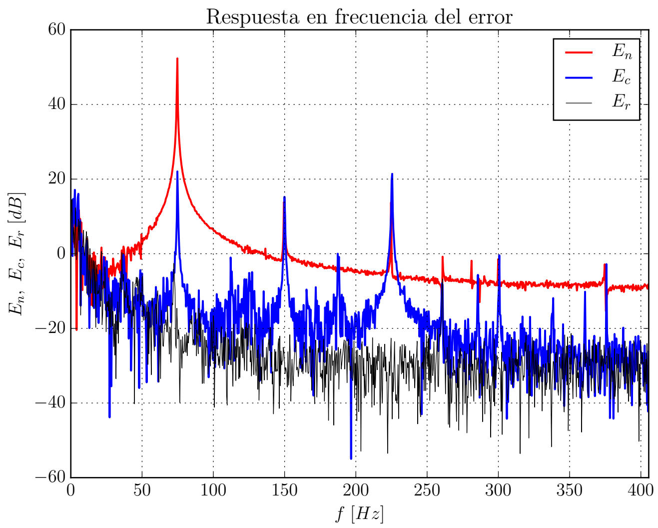

Fig. 18. Respuesta en frecuencia de la señal de error en reposo (E_r), en presencia de ruido (E_n) y con el control encendido luego de la convergencia (E_c). Se obtuvo una atenuación de 30dB a 75Hz.

VI. Conclusiones

Se logró una versión funcional del sistema de control de ruido activo en configuración feedback, aprovechando puertos analógicos de baja resolución integrados a la EDU-CIAA-NXP y utilizando un micrófono y un parlante de bajo costo. La configuración testeada permite reducir aún más el hardware, al requerir un único micrófono para su funcionamiento. En los ensayos se obtuvo una atenuación de 30dB que, aunque menor a la de otros sistemas que utilizan conversores A/D, micrófonos y parlantes de mayor calidad, es superior a la requerida en aplicaciones de este tipo. Esto sugiere que, dado el requisito de atenuación de por lo menos 10dB [4], es posible alcanzarlo con hardware de mucho menor costo que el utilizado en la bibliografía consultada. Análogamente, mejorando la calidad del hardware, debería ser posible lograr atenuaciones mayores sin modificaciones adicionales en el software.

El prototipo desarrollado servirá como plataforma para continuar experimentando con parámetros y algoritmos, habiendo resuelto cuestiones importantes como la lectura de canales ADC minimizando el uso del procesador, la escritura en DAC y las operaciones vectoriales.

A. *Próximos pasos*

Entre las mejoras que pueden hacerse al sistema se destacan:

- Mejorar el filtro anti-alias, buscando una respuesta en frecuencia plana hasta la frecuencia de corte. Opciones para lograr esto son configuraciones Sallen-Key en cascada para construir filtros de segundo orden. Esto aplica también al filtro de reconstrucción.
- La librería CMSIS-DSP [18] cuenta con todas las funciones para procesamiento de señales utilizadas en este trabajo, incluyendo filtros LMS y NLMS. Estas funciones se encuentran optimizadas para trabajar sobre Cortex-M con posibilidad de utilizar la unidad de punto flotante, por lo que un paso natural es comenzar a utilizarlas.

- Entender las variables que influyen sobre la convergencia del algoritmo, modificando el código para robustecer el sistema.
- Experimentar con el algoritmo FXNLMS, ya que posee mejores características que su versión no normalizada, y tiene potencial para reemplazarlo.
- Realizar pruebas con ruidos no senoidales, como por ejemplo motores, para conocer los alcances del sistema corriendo sobre EDU-CIAA-NXP.
- Considerar el uso de FreeRTOS para garantizar la regularidad de la señal de control.
- Agilizar la modificación de parámetros, permitir ajustarlos en tiempo real desde una PC.

REFERENCIAS

[1] R. P. King and J. R. Davis, "Community noise: Health effects and management," *International Journal of Hygiene and Environmental Health*, vol. 206, no. 2, pp. 123 – 131, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1438463904702023

[2] P. Lueg, "Process of silencing sound oscillations," U.S. Patent 2 043 416, June 9, 1936.

[3] S. M. Kuo and D. R. Morgan, "Active Noise Control: A Tutorial Review," in *Proceedings of the IEEE*, vol. 87, no. 6, Jun. 1999, pp. 943–973.

[4] L. Wu, X. Qiu, and Y. Guo, "A simplified adaptive feedback active noise control system," vol. 81, pp. 40–46, 07 2014.

[5] S. O. Haykin, *Adaptive Filter Theory*. Prentice Hall, 1995.

[6] Stéphane Boucher, Martín Bouchard, André L'esperance and Bruno Paillard, "Implementing a Single Channel Active Adaptive Noise Canceller with the TMS320C50 DSP Starter Kit," Department of Electrical and Computer Engineering, Faculty of Applied Sciences, University of Sherbrooke, Sherbrooke, Quebec, Application Report, Nov. 1997.

[7] S. M. Kuo, X. Kong, and W. S. Gan, "Applications of adaptive feedback active noise control system," *IEEE Transactions on Control Systems Technology*, vol. 112, no. 2, pp. 216–2202, March 2003.

[8] P. Peretti, S. Cecchi, L. Romoli, and F. Piazza, "Adaptive feedback active noise control for yacht enviuronments," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 2, pp. 737–744, 2014.

[9] C. K. Chen, T.-D. Chiueh, and J.-H. Chen, "Active cancellation system of acoustic noise in mr imaging," *IEEE Transactions on Biomedical Engineering*, vol. 46, no. 2, pp. 186–191, Feb 1999.

[10] B. Widrow and M. E. Hoff, "Neurocomputing: Foundations of Research," J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, ch. Adaptive Switching Circuits, pp. 123–134. [Online]. Available: http://dl.acm.org/citation.cfm?id=65669.104390

[11] A. Goldstein, "Ons Steepest Descent," *Journal of the Society for Industrial and Applied Mathematics Series A Control*, vol. 3, no. 1, pp. 147–151, 1965.

[12] J. C. Burgess, "Active adaptive sound control in a duct: A computer simulation," vol. 70, 09 1981.

[13] Proyecto CIAA. [Online]. Available: http://www.proyecto-ciaa.com.ar/

[14] J. Caldwell, "Single Supply, Electret Microphone Pre-Amplifier Reference Design," Texas Instruments, Sherbrooke, Quebec, Application Report, Jan. 2015.

[15] "Improving ADC Resolution by Oversampling and Averaging - AN118," Silicon Labs, Application Report, 2013.

[16] Why is the rand() function not the least bit random in the LPC11u24 MBED? [Online]. Available: http://os.mbed.com/questions/2886/Why-is-the-rand-function-not-the-least-b/

[17] LPCOpen Libraries and Examples. [Online]. Available: http://www.nxp.com/support/developer-resources/software-development-tools/lpc-developer-resources-/lpcopen-libraries-and-examples:LPC-OPEN-LIBRARIES

[18] CMSIS DSP Software Library. [Online]. Available: http://arm-software.github.io/CMSIS_5/DSP/html/index.html