Desarrollo e Implementación de un Controlador Activo de Ruido de Banda Angosta Adaptativo

F. A. González, R. Rossi, G. R. Molina y G. Parlanti
Laboratorio de Procesamiento Digital de Señales
Universidad Nacional de Córdoba
Córdoba, Argentina
rrossi@efn.uncor.edu
rmolina@efn.uncor.edu

Resumen—**Se presenta en este trabajo el diseño y construcción de un sistema Controlador Activo de Ruido (CAR) adaptativo, basado en un Procesador Digital de Señales de uso comercial. El sistema apunta a cancelar el ruido de banda angosta de baja frecuencia remanente en la cavidad de un auricular. Este ruido de baja frecuencia es especialmente difícil de cancelar por medios pasivos de cancelación acústica, pero utilizando técnicas activas como la presentada, se logran atenuaciones apropiadas y eficientes para el uso comercial.**

Palabras claves: CAR, Controlador Adaptativo, FxLMS

I. INTRODUCCION

En ambientes industriales, receptáculos cercanos a motores (como en la cabina de un avión o automóvil) o en entornos ruidosos en general, los auriculares utilizados para la cancelación acústica pasiva de ruido suelen ser efectivos solamente a frecuencias superiores a los 500 Hz. Como complemento de los auriculares pasivos, los sistemas de Control Activo de Ruido (CAR), apuntan a cancelar componentes de ruido de baja frecuencia. Cuando esta cancelación se produce en un espacio reducido, como dentro de un auricular, el sistema CAR se lo denomina "unimodal" y su objetivo es producir una señal de igual amplitud y fase contraria al ruido remanente dentro de la cavidad del auricular, comúnmente llamado "antiruido"[1].

El ruido dentro de la cavidad del auricular puede variar en el tiempo, ya sea porque la fuente de ruido ha variado, o porque la transferencia acústica del auricular ha cambiado, por ejemplo por diferencias anatómicas de quien lo usa. El sistema CAR debe ser capaz de adaptarse a estas variaciones, modificando en consecuencia la señal antiruido producida. A tal fin utiliza la diferencia entre la señal producida y la deseada (el ruido a cancelar) para modificar su propia función de transferencia mediante algún algoritmo apropiado, "aprendiendo" entonces de sus errores [2]. Los filtros adaptativos de un sistema CAR suelen realizarse con filtros de respuesta al impulso finita (FIR, Finite Impulse Response) por ser más sencillos de realizar y por ser incondicionalmente estables, en contrapartida con los filtros de respuesta al impulso infinita (IIR, Infinite Impulse Response). La modificación de la función de transferencia requerida se realiza entonces modificando en forma automática los coeficientes del filtro. Los filtros FIR requieren de mayor cantidad de coeficientes que los IIR, pero con el advenimiento de Procesadores Digitales de Señales (DSP, Digital Signal Processors), de gran capacidad de procesamiento, tanto la adaptación de los coeficientes como el proceso de la señal de entrada del CAR pueden realizarse en tiempo real.

En este trabajo se presentan detalles del diseño y de la implementación de un sistema CAR en un dispositivo DSP comercial, aplicado a la cancelación de ruido dentro de la cavidad de un auricular comercial.

II. CARACTERÍSTICAS DE DISEÑO

A. *Arquitectura general de un sistema adaptativo*

En general, un sistema adaptativo modifica su propia transferencia partiendo de cuán diferente sea su salida respecto de la salida ideal o deseada. Este principio puede aplicarse a diferentes fines, en un variado campo de las ciencias. El esquema de la Fig. 1, muestra el caso de un sistema adaptativo diseñado para identificar un sistema desconocido. La señal de entrada, también llamada referencia, se representa por la señal *x(n),* la transferencia del sistema desconocido por *P(z)* y la salida del mismo, *d(n),* representa la señal que se desea cancelar. La señal de referencia se introduce al sistema adaptativo, compuesto por un controlador con transferencia *H(z)* que produce la salida *y(n)* y el algoritmo de adaptación del controlador.

De lo expuesto, se define a la señal de error como:

$$e(n) = d(n) - y(n) = x(n) * p(n) - x(n) * h(n). \qquad (1)$$

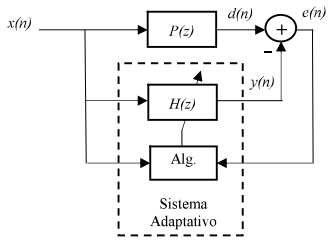

Figura 1. Sistema Adaptativo

La señal *e(n)* es quien gobierna el proceso de aprendizaje. En (1), * denota convolución, y *p(n)* y *h(n)* son las respuestas al impulso de *P(z)* y *H(z)* respectivamente. Se dice que el sistema ha convergido cuando $e(n) \approx 0$ y $H(z) \approx P(z)$, deteniéndose la adaptación. Esto sucede siempre que *x(n)* tenga suficiente contenido espectral para excitar todos los coeficientes del filtro FIR en el proceso de aprendizaje. El algoritmo de adaptación más utilizado por su sencillez y robustez es el llamado Least Mean Squares, LMS, [3] que produce la adaptación iterativa de los coeficientes del FIR *H(z)* según:

$$h_k(n+1) = h_k(n) + \mu e(n)x(n) \quad (2)$$

donde $k = 1, 2, \ldots L$, siendo L la cantidad de coeficientes del filtro FIR, y μ el paso o velocidad de adaptación entre la iteración n y la $n+1$. Un valor de μ pequeño hará el aprendizaje lento, pero logrará un error *e(n)* remanente pequeño. Un valor de μ alto hará el aprendizaje rápido, a expensas de un *e(n)* remanente más alto además de poner en riesgo la convergencia hacia el valor ideal, normalmente llamada solución de Wiener.

Una variante del algoritmo LMS, ampliamente utilizada, es el llamado Normalized Least Mean Squares, NLMS, que adapta el paso μ a la potencia de la señal de entrada según:

$$\mu(n) = \alpha / LP_x(n) \quad (3)$$

donde α es una constante y $P_x(n)$ es la potencia de la señal de referencia *x(n)* en el instante *n*. De esta forma el paso $\mu(n)$ utilizado con señales de baja potencia será mayor, acelerando el proceso de convergencia, mientras que con señales de mayor potencia $\mu(n)$ será menor, asegurando la estabilidad del sistema.

B. *Arquitectura de un CAR en un auricular*

En el caso del sistema CAR aplicado a la cavidad de un auricular, *P(z)* representa la transferencia acústica entre el exterior y el interior del auricular, y la señal de referencia *x(n)* el ruido fuera del auricular. La señal deseada *d(n)* representa el ruido a cancelar, remanente en el interior de la cavidad. En este caso, la señal de error se encuentra en el entorno acústico y no en el eléctrico. Esta situación se muestra en la Fig. 2.

En este caso, el controlador *H(z)* no podrá gobernar la señal antiruido directamente, sino a través de elementos físicos como un conversor Digital/Analógico y un parlante. Del mismo modo, el error acústico y la señal de referencia deberán ser captados por micrófonos y ser convertidas a digital por un conversor Analógico/Digital, para transformarse en la señal discreta *e(n)* y *x(n)* respectivamente. Además de estos bloques, normalmente son necesarios amplificadores y atenuadores. Todos estos bloques adicionales, se suelen representar por una sola función de transferencia llamada "camino secundario" *S(z)*.

Para compensar el efecto de *S(z)* y aplicar los mismos criterios de aprendizaje de (2), la señal de referencia *x(n)* debe afectarse o filtrarse por la misma función de transferencia *S(z)*.

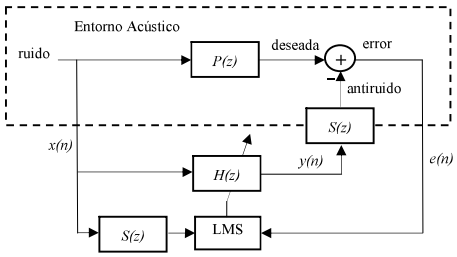

Figura 2. CAR en un auricular

El algoritmo de adaptación en este caso se llama LMS de referencia filtrada o FxLMS (por sus siglas en ingles de Filtered x Least Mean Squares) [4].

C. *Estimación del camino secundario*

La Fig. 2 supone que la transferencia *S(z)* se conoce, ya que debe usársela para filtrar la señal de referencia *x(n)*. En general esto no será así. Además la función de transferencia puede variar en el tiempo, producto del envejecimiento de los materiales utilizados, etc. Es por eso que la presencia del camino secundario desconocido o variante en el tiempo, obliga a introducir un segundo sistema adaptativo para estimar su función de transferencia. El esquema se muestra en la Fig. 3. Una vez convergido este segundo sistema adaptativo, su función de transferencia *Sp(z)*, será una buena aproximación a *S(z)*, y se utilizará una copia de la misma, *cSp(z)*, para filtrar la señal de referencia *x(n)*, produciendo la señal de referencia filtrada *x'(n)*.

En la Fig. 3, la señal *v(n)* es una señal de aprendizaje estadísticamente independiente de *x(n)* para que los procesos de aprendizaje no interfieran entre sí. La señal *f(n)* representa el error conjunto, calculado como:

$$f(n) = e(n) - v(n)*s_p(n) = x(n)*p(n) - x(n)*h(n)*s(n) + v(n)*s(n) - v(n)*s_p(n). \quad (4)$$

La señal *f(n)* es utilizada en ambos procesos de aprendizaje. Reemplaza a *e(n)* en (2) para el proceso adaptativo del camino secundario, y se usa para adaptar los coeficientes de *H(z)* como

$$h_k(n+1) = h_k(n) + \mu f(n)x'(n) \quad (5)$$

Cuando ambos sistemas han convergido, $f(n) \approx 0$, $Sp(z) \approx S(z)$, y $H(z) \approx [P(z)/S(z)]$, deteniéndose ambas adaptaciones. Si ambos procesos adaptativos se realizan simultáneamente, se dice que el camino secundario se modela "on-line".

La señal *v(n)*, requerida en una estimación on-line, siempre está presente en el error *e(n)*, siendo esta última la señal que el usuario escucha. Por ello, es conveniente disminuir o anular *v(n)* una vez que el proceso adaptativo de estimación del camino secundario ha convergido [5]. En la práctica, también es común realizar una estimación "off-line" del camino secundario. En este caso se produce primero el aprendizaje de *Sp(z)* y luego el de *H(z)*. Si *S(z)* cambia por algún motivo, el sistema adaptativo *Sp(z)* vuelve a activarse para emular esos cambios.

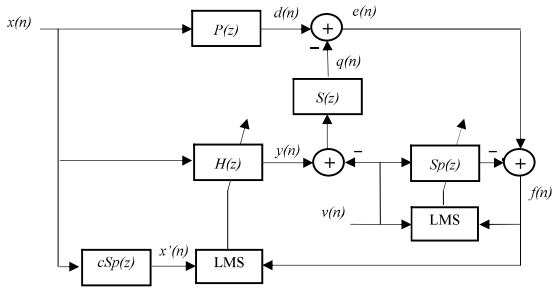

Figura 3. Sistema CAR con estimación del camino secundario

Es importante notar que para que el filtro *H(z)* sea causal el tiempo de proceso del sistema adaptativo más el retraso del camino secundario *S(z)* debe ser menor al retraso de la transferencia *P(z)*.

D. Alcance del CAR

Al diseñar un sistema CAR, lo primero que debe definirse es el tipo de ruido que se desea cancelar, dependiendo éste del ambiente donde se deba desempeñar. En ambientes con motores, turbinas, aire acondicionado, sirenas, etc, el ruido a cancelar será principalmente de banda angosta, es decir su espectro estará concentrado alrededor de frecuencias bastante definidas y su aspecto temporal será repetitivo. En cambio en ambientes donde el ruido represente música, conversaciones entre personas, ruido blanco gausseano, etc, el ruido a cancelar será de banda ancha, encontrándose un contenido espectral más distribuido. Este trabajo se concentra en el primer caso, la cancelación de ruido de banda angosta.

La Fig. 3 muestra una arquitectura que puede cancelar tanto ruido de banda ancha como de banda angosta, ya que se dispone de la señal de referencia *x(n)*, y cualquiera sea su espectro el sistema podrá aprender partiendo de ella. La señal de referencia debe tomarse desde el exterior del auricular, mediante un micrófono adicional al del error, denominado micrófono de referencia. Si se desea que el CAR sólo actúe sobre ruidos de banda angosta, como en nuestro caso, el esquema de la Fig. 3 puede simplificarse, prescindiendo del micrófono de referencia. En ese caso la señal *x(n)* podrá estimarse dentro del mismo proceso, según se muestra en la Fig. 4.

La señal de referencia *x(n)* se calcula como:

$$x(n) = y(n)*cs_p(n) + f(n) = y(n)*cs_p(n) + d(n) - y(n)*s(n) + v(n)*s(n) - v(n)*s_p(n). \quad (6)$$

Una vez convergidos ambos sistemas $cSp(z)=Sp(z)\approx S(z)$ y entonces $x(n)\approx d(n)$, por lo que el esquema de la Fig. 4, puede asimilarse a un proceso de control inverso adaptativo, con $H(z)\approx 1/S(z)$.

Es importante notar que el esquema de la Fig. 4 cancelará toda señal periódica componente de *d(n)*, mientras que el esquema de la Fig. 3 solo cancelará las componentes de *d(n)* relacionadas con *x(n)*, no afectando, por ejemplo al ruido propio de *P(z)*, o al ruido no captado por el micrófono de referencia.

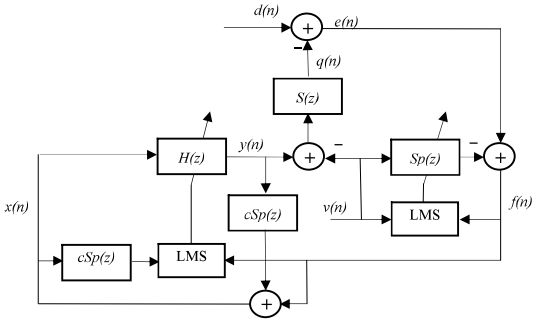

Figura 4. CAR para ruidos de banda angosta

En contrapartida, el esquema de la Fig. 4 solamente funcionará para cancelar ruidos de banda angosta (periódicos) pues si no el retraso del conjunto *H(z)S(z)* debería ser cero.

III. Implementación del Sistema

La implementación del CAR aplicado a un auricular, se realizó sobre el DSP StarCore MSC7116 de alto rendimiento de la empresa Freescale Semiconductor Inc. El StarCore MSC7116 es un DSP de punto fijo de 16 bits de ancho de palabra, de bajo costo, con un núcleo de cuatro ALUs, capaz de realizar 1000 MMACS a 266MHz. La capacidad de procesamiento del DSP es suficiente para realizar el cálculo computacional complejo que requiere la realización de los filtros adaptativos para ambos canales dentro del tiempo de un período de muestreo (procesamiento en tiempo real por muestras simples).

El software del controlador o programa de aplicación está montado sobre un sistema operativo de tiempo real o RTOS (por sus siglas en inglés Real Time Operating System), que corre sobre el DSP. Dicho sistema operativo es el SmartDSP de la empresa Freescale Semiconductor Inc, diseñado especialmente para los DSP de la familia StarCore. La interface de programación de aplicación o API (por sus siglas en inglés Application Program Interface) del SmartDSP [6], esta formada por funciones desarrolladas en lenguaje C, que permiten una fácil configuración y utilización de los periféricos del DSP. Para cada tipo de estos periféricos, existe un driver del sistema operativo, que permite la comunicación del programa de aplicación con los mismos. En el software del controlador se empleó el driver del periférico TDM (del inglés Time Division Multiplexing) para la entrada y salida de datos de ambos canales de audio (derecho e izquierdo) en el DSP. La velocidad de entrada y salida de datos es la frecuencia de muestreo del sistema, fijada en 8KHz. Este valor de frecuencia de muestreo es suficiente para la aplicación, ya que el rango de frecuencias de interés de un sistema CAR corresponde a frecuencias por debajo de los 500Hz.

El programa de aplicación fue desarrollado íntegramente en lenguaje C, empleando funciones denominadas "intrínsecas" a la hora de realizar y optimizar el procesamiento de los filtros adaptativos. Estas funciones intrínsecas son funciones en C que no pertenecen a la API del RTOS, sino que son funciones propias del compilador [7], especialmente diseñadas para realizar operaciones fraccionarias y optimizar el programa de

aplicación, aprovechando las características de procesamiento paralelo del DSP. Las funciones intrínsecas se intercalan directamente en el código en lenguaje C, permitiendo al programador aproximarse a la eficiencia propia del lenguaje ensamblador del DSP.

La precisión de la mayoría de los datos se fijó en una longitud de palabra de 16 bits en el formato de punto fijo. Para los coeficientes de *H(z)* se experimentó con 16 y 32 bits. Para mejorar la actualización de los coeficientes de los filtros en el proceso adaptativo y evitar que la adaptación se detuviera prematuramente por falta de precisión, se utilizaron 32 bits para el resultado del producto entre el paso de adaptación μ y la señal de error *f(n)*, implicando esto la realización de multiplicaciones de 32 x 16 bits en la actualización de los coeficientes de *H(z)* en el segundo sumando de (5). Esto llevó a una mejora importante en el desempeño de todo el sistema.

El modelo experimental implementado para evaluar el desempeño del CAR aplicado a un auricular, se basó en aplicar para cada canal de audio en forma independiente, un esquema como el presentado en la Fig. 4, utilizando el algoritmo FxLMS en su versión normalizada. El diagrama en bloques para cada canal del sistema CAR, se muestra en la Fig. 5. A continuación se explican cada una de las partes de este modelo.

A. *Placa de evaluación del Procesador Digital de Señales*

El módulo MSC711xEVMT [8], es una placa de desarrollo de software para aplicaciones que utilizan los DSP StarCore MSC711x. El uso de la misma permitió y facilitó la descarga y evaluación del software del controlador sobre el DSP, vía puerto paralelo con una PC portátil. Además por medio del CODEC integrado en la misma, ingresan y salen al DSP las señales analógicas de los transductores electroacústicos.

El CODEC es el AK4554 de 16 bits estéreo de la empresa AKM Semiconductor Inc., el cual realiza las conversiones A/D y D/A para ambos canales, además de efectuar los filtrados "antialiasing" y de reconstrucción (filtros pasa bajos) correspondientes. La entrada y salida de datos dentro del DSP para ambos canales del controlador, es realizada a través de su periférico TDM, el cual se comunica con el CODEC.

B. *Sistema acústico*

El auricular utilizado es el auricular circumaural estéreo SHP1900 de Philips, de una relación calidad-precio conveniente. Dada la característica de ser un auricular circumaural, las orejas de quien escucha son cubiertas totalmente por las almohadillas del auricular, creándose pequeñas cavidades acústicas en torno a cada uno de los oídos. Como sensores de error se usaron los micrófonos omnidireccionales ECM-30, los cuales son micrófonos del tipo *Electret*. Este tipo de micrófonos tienen características que son más que suficientes para aplicaciones de CAR (en lo que respecta a sus anchos de banda, sensibilidades y relaciones señal-ruido), además de ser de tamaño físico muy reducido, por lo que son ideales para nuestra aplicación.

La ubicación de los micrófonos de error dentro del auricular determina la zona de quietud, y fue sugerida por los autores de [9] y [10] como la ubicación ideal.

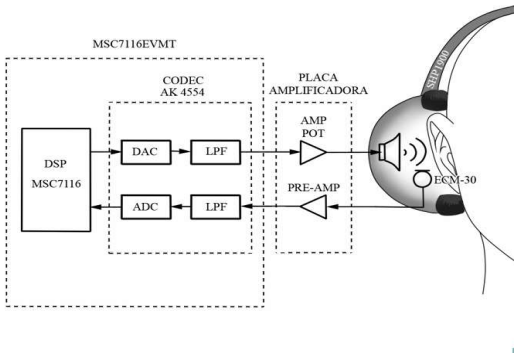

Figura 5. Diagrama en bloques de un canal del modelo experimental.

Dicha ubicación corresponde a la más cercana al tímpano del oído, además de demostrarse experimentalmente que esta configuración presenta la respuesta en frecuencia de la magnitud del camino secundario, más plana que otras configuraciones [9, 10].

C. *Placa amplificadora*

Se desarrolló un preamplificador para cada micrófono y el amplificador de potencia del auricular. Cada preamplificador es del tipo transistorizado y tiene como función el acondicionar el nivel bajo de la señal de cada micrófono hasta un nivel de señal adecuado para cada conversor A/D. El amplificador de potencia basado en el integrado LM386, entrega la potencia necesaria para excitar los parlantes del auricular.

IV. RESULTADOS

Para poder analizar y graficar los resultados obtenidos de las mediciones del sistema CAR se empleo el programa computacional MATLAB. Los datos de las mediciones se exportaron del DSP al programa MATLAB.

A. *Identificación del camino secundario S(z)*

Como hemos visto anteriormente, y se mostró en la Fig. 4, la estimación de *S(z)*, que llamamos *Sp(z),* es necesaria para estimar la señal de referencia *x(n)*. Además, se la utiliza para la actualización de los coeficientes del controlador *H(z)* con el algoritmo FxLMS, según (5). En la presente implementación se realizó una identificación del camino secundario *S(z)* de manera off-line.

Para realizar la identificación off-line de *S(z)*, *Sp(z)* se implementó con un filtro adaptativo FIR de longitud $L_s = 128$ coeficientes de tal manera de cubrir la mayor parte de su respuesta al impulso. Los coeficientes de *Sp(z)* fueron adaptados con el algoritmo LMS, según (2). Como señal de aprendizaje *v(n)* se empleó ruido blanco de media cero y varianza 0,03 generado internamente por el DSP. Se empleó un paso de adaptación $\mu = 0{,}01$ en (2).

La Fig.6 muestra la respuesta al impulso obtenida, en donde se observa un retardo de aproximadamente 40 coeficientes, o 5ms para una frecuencia de muestreo de 8kHz. La hoja de datos del CODEC indica que él mismo introduce un retardo fijo de 36 muestras, que representa casi todo el retardo presente en *Sp(z)*.

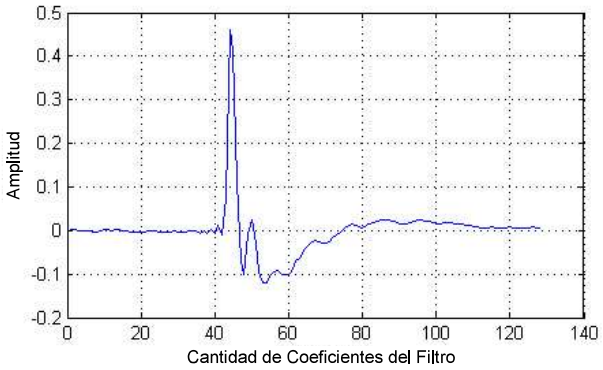

Figura 6. Respuesta al impulso de Sp(z).

De ser necesario, el retardo temporal mostrado en la Fig. 6, puede disminuirse aumentando la frecuencia de muestreo. La Fig. 7 muestra la magnitud de la respuesta en frecuencia de la estimación de *S(z)*.

Para observar la evolución temporal del proceso aprendizaje *de Sp(z),* se utilizó una ventana temporal de 16.000 muestras, es decir 2 segundos. El proceso de aprendizaje se muestra en la Fig. 8. En la misma la señal de ruido blanco filtrada por *S(z)* esta graficada en azul (en este caso la única componente de *e(n)* en la Fig. 4), el ruido blanco filtrado por el filtro adaptativo *Sp(z)* en verde y el error de identificación en rojo (en este caso la única componente de *f(n)* en la Fig. 4).

B. Desempeño del controlador H(z)

Para evaluar el desempeño del controlador, se generaron en MATLAB diferentes ruidos de prueba. Luego se los emitió con los parlantes de una PC portátil. Una persona portando los auriculares se ubicó de frente a los parlantes de la PC, recibiendo el ruido generado de manera uniforme en cada oído.

Durante el proceso de las mediciones se empleo la estimación de *S(z)* obtenida previamente de manera off-line. Para la adaptación de los coeficientes del controlador *H(z)* se utilizó la versión normalizada de FxLMS, el algoritmo FxNLMS, que combina (5) y (3). Para *H(z)* se empleó un filtro adaptativo FIR de longitud $L_h = 160$ coeficientes. En (3) se utilizó $\alpha = 0{,}016$ y para calcular de manera eficiente una aproximación P'_x de la potencia P_x se utilizó una ventana exponencial de la forma:

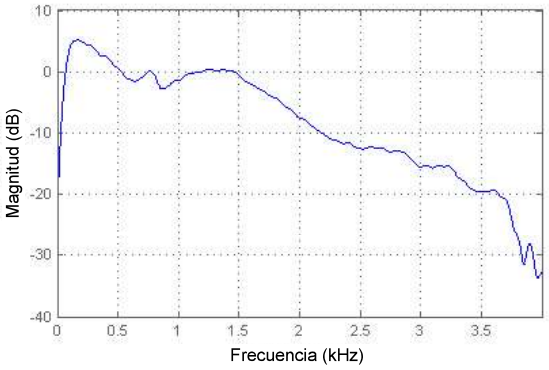

Figura 7. Magnitud de la respuesta en frecuencia de Sp(z).

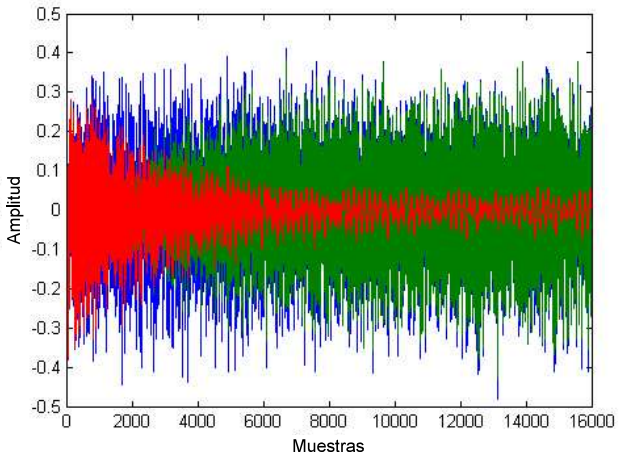

Figura 8. Evolución temporal de la identificación de S(z).

$$P'_x(n) = (1-\beta)P'_x(n-1) + \beta x^2(n) \qquad (7)$$

lo que requiere almacenar un solo valor ($P'_x(n-1)$) entre iteraciones, y utilizar un factor β que determina la rapidez en que los cambios de *x(n)* se reflejarán en $P'_x(n)$. Para ello se eligió $\beta = 0{,}002$ como compromiso entre una aproximación buena y estable de P_x y el seguimiento rápido de cambios en *x(n)*. Para evitar que valores cercanos a cero de *x(n)* produzcan μ muy grande en (3) se aseguró por software que (7) tenga un valor mínimo $P'_{xMIN}(n) = 0{,}01$.

El primer ruido de prueba generado en MATLAB fue un tono de 200Hz, con una amplitud tal que distorsione la salida de los parlantes de la PC portátil, generando así suficiente contenido armónico. La Fig. 9 muestra en azul el espectro de potencia del ruido generado (*d(n)* en la Fig. 4), en verde el espectro atenuado por el sistema CAR (*e(n)* en la Fig. 4) con coeficientes de 16 bits y en rojo con coeficientes de 32 bits. Dicha medición se obtuvo luego de 3 segundos (24.000 muestras) de comenzado el proceso de aprendizaje de *H(z)*.

En la Fig. 9 se observa el tono de 200Hz con sus armónicas, casi de la misma amplitud, y un tono de 50Hz, presente en el circuito. La atenuación que se logró para el tono de 200Hz, sus armónicas y el tono de 50Hz se muestran en la Tabla I, para ambas precisiones de los coeficientes de *H(z)*. Para armónicas siguientes a 600Hz prácticamente no se percibe atenuación.

El desempeño del controlador también fue probado con un típico ruido de maquina [11], también generado en MATLAB, y reproducido sin distorsión por los parlantes de una PC portátil. El mismo está formado por 12 componentes múltiplos de 60Hz de distinta amplitud, con la mayor potencia concentrada en las componentes de 240, 300, 360, 420 y 480Hz. La medición también fue hecha luego de haber transcurrido 3 segundos (24.000 muestras) de comenzado a actuar el aprendizaje del controlador *H(z)*.

La Fig. 10 muestra en azul el espectro de potencia del ruido, en verde el de la señal error para coeficientes de *H(z)* de 16 bits y en rojo para coeficientes de 32 bits. La atenuación de las componentes más significativas del ruido de maquina se detalla en la Tabla II para ambas precisiones de coeficientes.

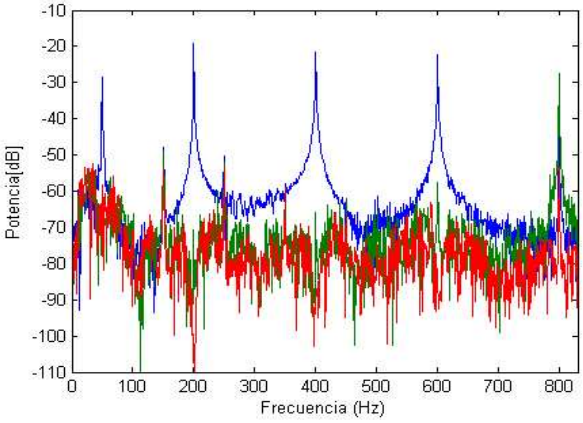

Figura 9. Espectro de potencia de ruido de 200 Hz distorsionado (línea azul), señal error con coeficientes de 16 bits (línea verde) y 32 bits (línea roja).

TABLA I. ATENUACIÓN DEL TONO Y SUS ARMÓNICAS

Frecuencia (Hz)	Atenuación coeficientes 16 bits (dB)	Atenuación coeficientes 32 bits (dB)
50	33,5	29,5
200	51	85,3
400	44	67,4
600	35,5	59,5

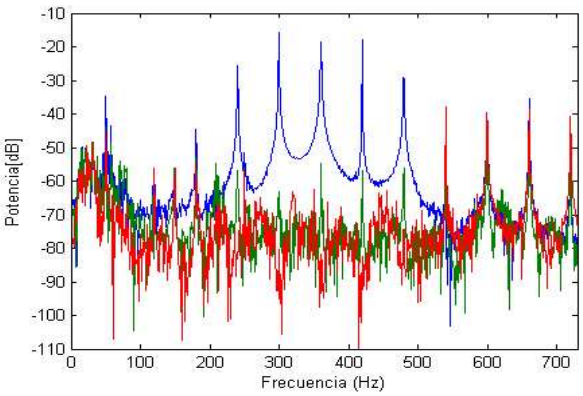

Figura 10. Espectro de potencia del ruido de maquina (línea azul), señal error con coeficientes de 16 bits (línea verde) y 32 bits (línea roja).

TABLA II. ATENUACIÓN DEL RUIDO DE MÁQUINA

Frecuencia (Hz)	Atenuación coeficientes 16 bits (dB)	Atenuación coeficientes 32 bits (dB)
240	31,3	56,1
300	46,7	64,6
360	35,2	63,1
420	38,7	64,5
480	25,6	45,2

V. CONCLUSIONES Y DIRECCIONES FUTURAS

A partir del análisis y los resultados presentados en este trabajo se resumen las siguientes conclusiones:

- El sistema CAR diseñado e implementado atenúa en forma aceptable diferentes ruidos periódicos (o de banda angosta) en la banda de frecuencia de interés.
- La precisión y capacidad de cómputo del DSP utilizado fue suficiente para realizar el procesamiento de ambos canales independientes de audio en tiempo real en forma simultánea.
- Es apreciable la mejora al utilizar coeficientes de 32 bits para *H(z)*.
- Para las señales de prueba utilizadas el usuario manifestó que el ruido remanente en la cavidad del auricular resultó en niveles confortables, y fue sensiblemente menor al percibido sin la activación del CAR.

Las direcciones futuras se orientarán al análisis, diseño e implementación de un CAR aplicado a la cancelación de ruido de banda ancha en un auricular. Asimismo, se investigarán y analizarán otros algoritmos de adaptación, implementándolos en condiciones reales y con componentes disponibles comercialmente.

AGRADECIMIENTOS

A la Secretaria de Ciencia y Tecnología de la Universidad Nacional de Córdoba, a la empresa Freescale Semiconductor, Inc.

REFERENCIAS

[1] S. M. Kuo and D. R. Morgan, Active Noise Control Systems-Algorithms and DSP Implementations, 1st ed., Hoboken, NJ: Wiley, 1996, pp.1–15.

[2] B. Widrow and S. Stearns, Adaptive Signal Processing, 1st ed., Englewood Cliffs,NJ:Prentice-Hall, 1985, pp.3–96.

[3] B. Widrow, "Adaptive filters" in Aspects of Network and Sustems Theory, R. E. Kalman and N. DeClaris, Eds. New York: Holt, Rinehart and Wiston, 1970, pp. 563–587.

[4] B. Widrow, D. Shur and S. Shaffer "On adaptive inverse control" in Proc. 15th Asilomar Conf., 1981, pp. 185-189.

[5] M. T. Akhtar, M. Abe and M. Kawamata "Noise power scheduling in active noise control systems with online secondary path modeling" in IEICE Electronic Express, Vol. 4 No. 2, 2007, pp. 66-71.

[6] *SmartDSP OS Reference Manual*, Rev. 1.42, Metrowerks, Austin, TX, Sept. 2005.

[7] CodeWarrior Development Studio for StarCore DSP Architectures: C Compiler User Guide, Freescale, Austin, TX, Aug. 2009.

[8] *MSC711XEVM User's Guide,* Rev. 0, Freescale, Austin, TX, Apr. 2005.

[9] W. S. Gan and S. M. Kuo, "Adaptive Feedback Active Noise Control Headset: Implementation, Evaluation and Its Extensions", *IEEE Transaction on Consumer Electronics,* vol. 51, no. 3, pp. 975-982, Aug. 2005.

[10] S. M. Kuo and W. S. Gan, "Active Noise Control System for Headphone Applications", *IEEE Transaction on Control Systems Technology,* vol. 14, no. 2, pp. 331-335, Mar. 2006.

[11] MathWorks. Filter design toolbox. Active Noise Control Using a Filtered-X LMS FIR Adaptive Filter. [Online]. Available: http://www.mathworks.com/products/filterdesign/demos.html?file=/products/demos/shipping/filterdesign/adaptfxlmsdemo.html