

UNIVERSIDAD NACIONAL DE CÓRDOBA
FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES



**Sistema de Control Activo de Ruido Acústico
Periódico para el Interior de Automóviles**

**Proyecto Integrador
Ingeniería Electrónica**

Director: Ing. Roberto R. Rossi

Co-director: Ing. Sebastián P. Ferreyra

Alumno: Leopoldo Budde

Córdoba, Argentina
Noviembre de 2013

Agradecimientos

A mi papá, mi mamá y mi hermano, por el amor y el apoyo incondicional brindado durante toda mi vida. Gracias por ser la guía y fuente de motivación de todas y cada una de las tareas necesarias para poder alcanzar hoy este título. A mi novia Meli, por estar siempre presente y darle color a mi vida. Sin ella nada de esto hubiera sido posible.

A mis directores, Roberto R. Rossi y Sebastián P. Ferreyra, por su invaluable aporte no sólo en la dirección de este proyecto sino también en lecciones para la vida. Gracias por su apoyo constante y por el respaldo en cada decisión tomada.

Al Laboratorio de DSP (UNC), por brindar la oportunidad de desarrollar este proyecto integrador y por todos los conocimientos compartidos.

A todos y cada uno de los miembros del CINTRA (UTN), por abrir sus puertas brindándome toda la formación sin la cual hubiera sido imposible la realización de este trabajo. Gracias por darme la oportunidad de crecer, por los buenos momentos y por las comidas! En especial gracias a todos los miembros del PID 1658, por todas las tardes compartidas que hicieron posible disfrutar el trabajo juntos.

Al cuerpo docente de la Facultad de Ciencias Exactas, Físicas y Naturales de la UNC, por su excelente labor de educadores.

Al ingeniero José Amado, por ser fuente de motivación en una de las etapas más difíciles de la carrera. A los doctores Fabián Tomasini y Nicolás Wolovick, y a los ingenieros Gabriel Aguilar y Fernando González Vergara, cuyos aportes constituyeron claves esenciales durante el desarrollo del proyecto. A Matías Babenco, cuya contribución en una etapa fundamental del trabajo hizo posible reducir en varios meses el tiempo necesario para concretar el mismo. Al ingeniero Manuel Starck y al licenciado Facundo Rodríguez por su colaboración en la redacción del trabajo.

Al ingeniero Oscar Cáceres, por su gran eficiencia y apoyo brindado desde la Cátedra de Proyecto Integrador.

A todos mis familiares y amigos por darle sentido a mi vida y hacer que todo valiera la pena.

Gracias.

Resumen

En la actualidad la industria automotriz está incrementando su atención en el ruido en el interior de los vehículos, tanto por cuestiones de confort como para ajustarse a las normativas correspondientes.

Existen diferentes métodos para el control del ruido en recintos cerrados. El control pasivo de ruido (PNC: Pasive Noise Control) es eficiente para controlar (atenuar) el espectro de medias y altas frecuencias, sin embargo para obtener resultados satisfactorios en bajas frecuencias requiere incorporar materiales de masa y tamaño considerables. Por otra parte, el control activo de ruido (ANC: Active Noise Control) permite reducir la amplitud del ruido (producido por una ó múltiples fuentes sonoras) en un punto del espacio introduciendo un campo acústico secundario. El ruido secundario es de igual amplitud y fase opuesta al primario, de manera que al interferirse las ondas se produce la cancelación de ambas señales . El ANC se presenta como una alternativa eficiente en bajas frecuencias, que complementa al método de PNC. Al implementar un sistema de control activo de ruido (ANCS: Active Noise Control System) en un recinto se debe trabajar con múltiples fuentes secundarias y múltiples sensores de error (sistema multicanal), debido a la complejidad del campo sonoro en el interior del mismo . El sistema debe ser adaptativo para aplicarse a situaciones de transferencia acústica no conocidas de antemano, y para adaptarse a variaciones de la fuente de ruido.

El principal objetivo de este trabajo es desarrollar un sistema multicanal de control activo de ruido para el interior de automóviles, el cual busca atenuar las componentes espectrales de baja frecuencia del sonido proveniente del motor del mismo. Para esto se construyó una cabina de características similares a la de un automóvil y se desarrolló un sistema realimentado de 4 canales en el interior de la misma. Se implementó el algoritmo LMS de referencia filtrada x modificado (MFxLMS: Modified Filtered x Least Mean Square) en el procesador digital de señal (DSP: Digital Signal Processor) MSC7116, de la empresa Freescale Semiconductor Inc. Se lograron obtener atenuaciones mayores a 40dB en las componentes principales del ruido a cancelar.

Índice

Agradecimientos	2
Resumen.....	3
1 Introducción.....	8
1.1 Contexto	8
1.2 Objetivos	9
1.3 Modelo propuesto	10
2 Control Activo de Ruido	11
2.1 Introducción a la acústica.....	11
2.1.1 Sonido.....	11
2.1.2 Acústica de recintos	12
2.1.3 Efectos del ruido en el hombre	14
2.2 Ruidos.....	15
2.3 Concepto ANC	16
2.4 Sistemas ANC.....	16
2.4.1 Sistema prealimentado de banda ancha.....	17
2.4.2 Sistema prealimentado de banda angosta.....	18
2.4.3 Sistema realimentado	19
2.4.4 Sistema híbrido.....	20
2.5 Filtros FIR Adaptativos.....	20
2.5.1 Filtros discretos	21
2.5.2 Filtro Wiener	22
2.5.3 Filtros adaptativos	24
2.5.4 Método de máxima pendiente.....	25
2.5.5 Algoritmo LMS.....	26
2.5.6 Desempeño del algoritmo LMS	27
2.5.7 Algoritmo LMS normalizado.....	30
2.5.8 Algoritmo FxLMS	31
3 Definición del modelo	33
3.1 Selección del sistema ANC	33

3.2 Selección de control global o local	33
3.3 Diseño, construcción y medición del recinto	35
3.3.1 Diseño de la cabina	37
3.3.2 Construcción de la cabina	38
3.3.3 Cálculo de modos propios	40
3.3.4 Medición de la respuesta impulsiva de la cabina.....	42
3.4 Localización de los transductores	47
3.4.1 Principios básicos para su determinación	47
3.5 Transductores.....	49
3.5.1 Fuentes secundarias.....	49
3.5.2 Sensores	56
3.6 Amplificación de potencia.....	57
3.6.1 Amplificación para fuentes de control	57
3.6.2 Amplificación para fuente de ruido.....	58
3.6.3 Alimentación	58
4 Procesamiento digital de señales en tiempo real	60
4.1 Procesamiento digital de señales.....	60
4.1.1 Procesamiento en tiempo real.....	62
4.1.2 Formato de datos digitales.....	63
4.1.3 Relación señal a ruido de cuantificación o SQNR.....	65
4.2 Efectos de la precisión finita en el ANC.....	66
5 Simulaciones y optimización	70
5.1 Simulación Monocanal	70
5.1.1 Sistema de control.....	70
5.1.2 Camino secundario.....	71
5.1.3 Camino primario.....	72
5.1.4 Señal de ruido.....	73
5.1.5 Condiciones de simulación	76
5.1.6 Resultados	77
5.1.6.3 Señal número 3	80
5.2 Simulación multicanal 4x2.....	81
5.2.1 Sistema de control.....	83

5.2.2 Señal de ruido.....	84
5.2.3 Caminos secundarios.....	85
5.2.4 Condiciones de simulación.....	85
5.2.5 Resultados	86
5.3 Optimización	91
5.3.1 Sistema 2x2	92
5.3.2 Aumento de la absorción acústica en el recinto	95
5.3.3 Situación límite.....	98
5.3.4 Conclusiones de la optimización	100
6 Modelo experimental	101
6.1 Sistema multicanal de control activo de ruido.....	101
6.2 Sistema para adquisición de datos y medición del desempeño	103
6.3 Módulo de evaluación del DSP StarCore MSC7116	107
6.3.1 DSP StarCore MSC7116	108
6.3.2 CODEC AK4554	110
7 Implementación en el DSP	112
7.1 Descripción de SmartDSP	112
7.1.1 Arquitectura	112
7.2 Driver del periférico TDM.....	115
7.2.1 Inicialización	115
7.2.2 Proceso de transmisión y recepción de datos.....	116
7.3 Archivos del proyecto.....	117
7.4 Funciones intrínsecas	119
7.5 Programa del sistema ANC.....	120
7.6 Optimizaciones.....	125
7.6.1 Buffer circular.....	125
7.6.2 Precisión	126
7.6.3 Operación de ALUs en paralelo	129
7.6.4 Actualización parcial de coeficientes	131
8 Mediciones y resultados.....	133
8.1 Sistema ANC monocanal	133
8.1.1 Señal senoidal.....	134

8.1.2 Señal de ruido del motor.....	135
8.1.3 Análisis de filtros	139
8.2 Sistema ANC multicanal	142
8.2.1 Implementación inicial	143
8.2.2 Algoritmo LMS con actualización parcial de coeficientes	144
9 Conclusiones y trabajos a futuro.....	147
9.1 Conclusiones.....	147
9.2 Trabajos a futuro	148
9.3 Publicación	149
Bibliografía	150
Anexo – Código del programa.....	155

CAPÍTULO 1: Introducción

En este capítulo se presenta de manera global y sintética el Proyecto Integrador llevado a cabo, planteando el contexto en el cual surge el mismo y exponiendo los objetivos del trabajo.

1.1 Contexto

Los efectos del ruido en el hombre han sido estudiados por diversos investigadores, ya sea que se trate de efectos no clínicos (reducción en la eficiencia del trabajo, interferencia en la palabra) como de efectos clínicos no auditivos (reducción del rendimiento físico, pérdida de la concentración) y de efectos clínicos auditivos (hipoacusia) [3]. En este trabajo nos concentraremos en el ruido dentro de una cabina cerrada, por ejemplo, de un automóvil. Existen tres factores fundamentales que componen el sonido en el interior de un automóvil: el ruido producido por el motor (incluyendo admisión y escape), el ruido aerodinámico producido por la colisión del aire y la carrocería del vehículo (interface fluido-estructura), y el ruido de rodamiento producido por el rozamiento de los neumáticos y la superficie [4].

Si bien las técnicas de control pasivo de ruido (PNC: Pasive Noise Control) son efectivas para medias y altas frecuencias, no ocurre lo mismo para frecuencias inferiores a 500 Hz, debido a las grandes longitudes de onda involucradas y a los efectos de resonancia de la estructura. En este caso, el PNC debería contemplar la utilización de materiales aislantes con pesos y tamaños considerables, lo cual contradice con los requerimientos actuales de la industria automotriz.

En función de las limitaciones del PNC, y del gran avance de la tecnología de procesamiento digital de señales, surge el control activo de ruido (ANC: Active Noise Control). El mismo, se basa en introducir una onda sonora de *antiruido* a través de un arreglo apropiado de fuentes secundarias para lograr la cancelación del ruido en la zona deseada por interferencia destructiva de ondas. Debido a que el campo sonoro en el interior de un automóvil no presenta la simplicidad que puede encontrarse en un ducto o en una cavidad auricular, es necesario implementar un sistema multicanal (más de una fuente secundaria) para lograr la atenuación deseada en una denominada *zona de quietud*.

Resulta de suma importancia destacar la manera en que el presente proyecto integrador incluye contenidos de diversas áreas de estudio de la carrera de grado. Por un lado se aplican conceptos de física y acústica al estudiar el campo sonoro en el interior del recinto, en conjunto con herramientas propias del análisis matemático. Se utilizan en profundidad contenidos tanto de electrónica analógica como de electrónica digital, centrando finalmente la atención en el procesamiento digital de señales. A su vez se aplican fundamentos de sistemas de control, y seguridad e higiene en el trabajo.

1.2 Objetivos

El objetivo general de este trabajo es el desarrollo e implementación de un sistema multicanal de control activo de ruido acústico periódico para el interior de automóviles. Este sistema buscará atenuar el sonido proveniente del motor del vehículo, y será del tipo adaptativo para adecuarse tanto a cambios en la señal del ruido a cancelar, como a cambios en el medio. El control a implementar será local, generando una zona de quietud definida por los sensores de error del sistema.

Como objetivos particulares podemos mencionar:

- Estudio acústico para determinar frecuencia y amplitud de los modos propios en el interior del recinto.
- Selección de la cantidad y posición de las fuentes secundarias.
- Estudio de la longitud temporal de las funciones de transferencia a implementar.
- Determinación de las funciones de transferencia aproximadas que nos brinden una aceptable reducción del ruido.
- Diseño de las funciones de transferencia aproximadas adaptativas en un procesador digital de señal (DSP: Digital Signal Processor) comercial.

1.3 Modelo propuesto

Se implementó un sistema realimentado adaptativo de cuatro canales (dos sensores de error y dos fuentes secundarias) con el algoritmo MFxLMS sobre el procesador digital de señal MSC7116. El sistema fue instalado en el interior de un recinto diseñado y construido para obtener características similares a la cabina de un automóvil, y se midieron las atenuaciones obtenidas al utilizar como señal de ruido el sonido producido por un motor de cuatro tiempos y cuatro cilindros operando a 2500 rpm.

1.4 Organización del trabajo

En el Capítulo 2 del trabajo se plantean las bases teóricas involucradas en el control activo de ruido, tanto de acústica, como de filtrado FIR adaptativo. A continuación en el Capítulo 3 se detalla la definición del modelo utilizado, presentando luego en el Capítulo 4 los fundamentos del procesamiento digital de señales en tiempo real. Se describen en el Capítulo 5 las simulaciones computacionales llevadas a cabo, las cuales permitieron optimizar el sistema reduciendo el número de fuentes secundarias a utilizar, para luego detallar en el Capítulo 6 el modelo experimental utilizado. Luego se plantea en el Capítulo 7 la implementación del sistema de control activo de ruido en el DSP, tratando aspectos fundamentales como la precisión de los algoritmos y la longitud de los filtros. En el Capítulo 8 se muestran los resultados obtenidos y los factores que hicieron necesarias las distintas optimizaciones del sistema. Finalmente se presentan en el Capítulo 9 las conclusiones y trabajos a futuro.

CAPÍTULO 2: Control Activo de Ruido

En esta sección se plantean los fundamentos acústicos que constituyen el problema a resolver, y una clasificación de los distintos tipos de ruidos. A su vez se presentan los principios fundamentales del control activo de ruido y los principales sistemas ANC. Finalmente se hace una breve descripción de los filtros de respuesta impulsiva finita (FIR: Finite Impulse Response) adaptativos utilizados en la implementación del sistema.

2.1 Introducción a la acústica

2.1.1 Sonido

Puede decirse que hay sonido cuando un disturbio que se propaga por un material elástico causa una alteración de la presión, la densidad, o un desplazamiento de las partículas del material, que puedan ser reconocidos por una persona o instrumento [5]. El elemento que genera el sonido se denomina fuente sonora, y el fenómeno se produce cuando dicha fuente entra en vibración. Sin embargo, la perturbación no produce transporte neto de las partículas, sino que las mismas oscilan alrededor de su posición de equilibrio en la misma dirección en que avanza la propagación de la onda (ondas longitudinales) [6].

Una manera habitual de expresar cuantitativamente la magnitud de un campo sonoro es mediante la presión sonora, es decir la fuerza que ejercen las partículas de aire por unidad de superficie [6]. Se define el nivel de presión sonora (SPL: Sound Pressure Level) a partir de la siguiente ecuación

$$L_p \text{ ó } SPL = 20 \log \frac{P}{P_{ref}} [\text{dB}] \quad (2.1)$$

En donde p es la *presión sonora instantánea*, y p_{ref} la *presión sonora de referencia*, con un valor de 20 μPa , correspondiente al umbral de audición humano promedio cuando es excitado con un tono puro de 1 kHz.

En la Tabla 2.1 se presentan los niveles de presión sonora correspondientes a diferentes fuentes sonoras medidos a una distancia determinada, junto a su respectiva valoración perceptual ó subjetiva.

Tabla 2.1: SPL y valoración subjetiva de sonidos producidos por diferentes fuentes. Adaptada de [6]

Fuente sonora / Espacio acústico	SPL [dB]	Valoración subjetiva
Despegue avión (a 60 m)	120	Muy elevado
Edificio en construcción (a 5 m)	110	
Martillo neumático (a 2 m)	100	
Camión pesado (a 15 m)	90	Elevado
Tráfico vehicular en calle asfaltada (ciudad) (a 4 m)	80	
Interior automóvil	70	
Conversación normal (a 1 m)	60	Moderado
Oficina, aula	50	
Sala de estar	40	
Dormitorio (noche)	30	Bajo
Estudio de radiodifusión	20	

2.1.2 Acústica de recintos

La energía que irradia una fuente sonora en un recinto cerrado arriba al oyente ó receptor en dos formas: a) de manera directa a través del espacio libre, recorriendo la distancia más corta (línea recta) entre la fuente y el receptor (*sonido directo*); b) de manera indirecta, alcanzando al receptor luego de sufrir múltiples transformaciones al reflejarse sobre diferentes objetos (sonido reflejado). La energía correspondiente al sonido directo disminuye con la distancia a la fuente, mientras que la energía asociada al sonido reflejado depende además de las propiedades de las superficies implicadas, como así también de la cantidad de reflexiones. Es de particular interés para este trabajo el análisis temporal del sonido reflejado, el cual está compuesto por un primer grupo que incluye las reflexiones

que llegan al receptor inmediatamente después del sonido directo (*reflexiones tempranas*), y un segundo grupo llamado *cola reverberante* que la constituyen las *reflexiones tardías* [6].

El nivel de presión sonora en campo libre (sonido directo) disminuye al aumentar la distancia a la fuente, en un valor 6 dB por cada duplicación de la distancia. Por el contrario, el campo reverberante es aproximadamente constante en recintos cerrados, debido a que la superposición de las múltiples reflexiones resulta en una distribución prácticamente uniforme del sonido [3].

La presencia de ondas sonoras incidentes y reflejadas en recintos cerrados genera *interferencia constructiva y destructiva*, estableciendo *ondas estacionarias o modos propios de vibración* de un recinto. Cada modo posee una frecuencia denominada propia o normal, y están caracterizados por un nivel de presión sonora que varía dependiendo del punto considerado. El número de modos propios es ilimitado, si bien su distribución a lo largo del eje frecuencial es discreta, la densidad modal aumenta con el cubo de la frecuencia. La presencia de los diversos modos propios genera en cada punto concentraciones de energía alrededor de las diversas frecuencias propias, lo cual confiere un sonido característico a cada recinto. Esto recibe el nombre de “coloración” y normalmente se manifiesta en espacios de dimensiones relativamente reducidas. Los valores de las frecuencias propias dependen de la geometría y dimensiones del recinto. Cuando se trata de recintos de forma paralelepípedica con superficies cuyo coeficiente de reflexión es máximo, es posible calcular las frecuencias de los modos propios mediante la fórmula de Rayleigh

$$f_{k,m,n} = 172,5 \sqrt{\left(\frac{k}{L_x}\right)^2 + \left(\frac{m}{L_y}\right)^2 + \left(\frac{n}{L_z}\right)^2} \quad (2.2)$$

donde:

- 172,5 corresponde a la velocidad del sonido en el aire sobre 2, para una temperatura de 20 °C
- L_x , L_y y L_z representan las dimensiones de la sala (en metros)
- k , m , n pueden tomar cualquier valor entero (0, 1, 2, 3, ..)

Cada combinación de valores k, m, n da lugar a una frecuencia y modo propio asociado k, m, n [7]. Por ejemplo, la combinación: k = 1, m = 2, n = 3 da lugar al modo propio 1 2 3.

La densidad modal se define como el número de frecuencias propias por Hz a una determinada frecuencia f [8]. Como se mencionó anteriormente, la densidad de modos propios aumenta con el cubo de la frecuencia, por lo que a partir de una cierta frecuencia, el concepto de coloración del sonido deja de tener sentido, ya que una gran densidad de modos propios puede interpretarse como la ausencia de éstos, ya que dejan de existir concentraciones discretas de energía. La *frecuencia Schröeder ó de corte* define la frecuencia límite superior a partir de la cual se verifica la existencia de más de tres modos por Hz, por lo cual por encima de ella influencia es prácticamente nula, su expresión matemática es la siguiente:

$$f_{schröeder} = 2000 \sqrt{\frac{T}{V}} \quad (2.3)$$

donde:

- T es el valor del tiempo de reverberación expresado en segundos.
- V es el volumen de la sala, expresado en m^3

De la anterior expresión se desprende que el efecto de los modos propios tiene una mayor incidencia cuanto más pequeño es el recinto en consideración [6]. Esto constituye un factor fundamental a tener en cuenta a la hora de diseñar el recinto sobre el cual será aplicado el sistema ANC.

2.1.3 Efectos del ruido en el hombre

Los niveles sonoros excesivos presentan una gran variedad de efectos nocivos para el hombre, tanto efectos no clínicos, como efectos cínicos no auditivos y efectos auditivos.

Entre los efectos no clínicos el primero de ellos es la molestia. Puede decirse de manera muy general que el confort auditivo ocurre hasta los 70 a 80 dBA, mientras que por

encima de los 120dBA se percibe dolor además de un sonido ensordecedor. Otro aspecto a destacar es la interferencia en la comunicación, lo cual lleva a las personas a elevar la voz forzando sus cuerdas vocales [3].

Por otro lado, existe un gran número de efectos clínicos no auditivos del ruido, entre los cuales se hallan la hipertensión arterial pasajera, las cefaleas, el nerviosismo y estrés, la reducción del rendimiento físico y la pérdida de la concentración y la atención [3].

Dentro de los efectos auditivos, el más notorio es la pérdida de la audición o hipoacusia. Esta es resultado de la exposición a ruidos extremadamente fuertes aún durante poco tiempo, o de la exposición reiterada en el tiempo a ruidos no tan intensos [3].

2.2 Ruidos

Puede definirse el ruido como cualquier sonido indeseado. En función de esto es imposible discriminar si un determinado sonido se presenta como ruido o no, ya que ello depende las circunstancias específicas y de la actitud mental de aquellos expuestos al sonido [9]. En relación con su espectro en frecuencia, los mismos pueden clasificarse como ruidos de banda angosta y ruidos de banda ancha [2].

Los ruidos acústicos de banda angosta tienen componentes discretas de frecuencia, en las cuales está concentrada su mayor energía [10]. Estas componentes en frecuencia pueden estar relacionadas o no. Un caso en el cual están relacionadas, es aquel en donde las componentes son múltiplos de una frecuencia fundamental. Los ruidos de banda angosta se caracterizan por ser determinísticos o quasi-determinísticos del tipo periódicos. Se dice que son quasi-determinísticos ya que en la realidad las características de los ruidos no permanecen constantes, sino que varían lentamente. Sin embargo, como esta variación se produce en un tiempo mayor a los tiempos involucrados en un sistema de control activo, se puede considerar a estos ruidos como determinísticos para su análisis en dicho sistema. Una característica fundamental de las señales determinísticas es que su comportamiento es predecible en función del tiempo. Un ejemplo muy común de ruidos de banda angosta es el de los ruidos de baja frecuencia producidos por máquinas de rotación, tales como ventiladores, motores, compresores y las turbinas de un avión.

El otro tipo de ruido acústico es el ruido de banda ancha. Se lo llama así porque su espectro comprende una banda de frecuencias, en la cual está distribuida su energía de manera más o menos uniforme. Estos ruidos de banda ancha tienen la característica de ser aleatorios [10] [11]. Ejemplos de estos ruidos son los producidos por turbulencias y los ruidos impulsivos de una explosión.

2.3 Concepto ANC

El ANC se basa en un sistema electroacústico que cancela el *ruido primario* indeseado según el principio de superposición, en donde el *ruido secundario* o *antiruido* posee igual amplitud y fase opuesta al primario. Al sumarse ambas señales se obtiene la cancelación deseada. Los sistemas activos permiten atenuar ruidos de baja frecuencia, en donde las técnicas de PNC son poco eficientes y complejas en cuanto a peso y tamaño, funcionando como un suplemento moderno para dichos sistemas convencionales [2].

Todo sistema ANC debe satisfacer requerimientos de precisión en el control, estabilidad y confiabilidad, debido a que para producir un buen grado de atenuación, la amplitud y fase de ambas señales deben acoplarse con un alto grado de precisión. En función de esto es deseable que el controlador sea digital, en donde las señales provenientes de los transductores electroacústicos o electromecánicos sean muestreadas por un procesador digital de señal con la velocidad y precisión necesarias para ejecutar funciones matemáticas sofisticadas en tiempo real [2].

2.4 Sistemas ANC

La elección del sistema ANC a utilizar depende específicamente de la aplicación. En los casos en que puede considerarse que la propagación de las ondas sonoras es unidimensional, como por ejemplo en ductos cuyo radio sea considerablemente menor que la longitud de onda del ruido a cancelar, suele ser suficiente la utilización de sistemas monocanales (una única fuente secundaria y un sensor de error). Sin embargo, cuando se trabaja en recintos o estructuras con múltiples grados de libertad, la geometría del campo

sonoro primario es de gran complejidad y no es suficiente utilizar un único sensor de error para lograr la cancelación deseada [2]. En estos casos se vuelve necesario la exploración y desarrollo de sistemas multicanales adecuados para cada aplicación en particular, en donde el número y la localización de las fuentes secundarias y los sensores de error constituyen un factor fundamental en el desempeño del sistema.

Para cancelar ruidos de banda ancha, es necesario obtener una señal de referencia que brinde información suficiente acerca del ruido que se aproxima, en donde todo el ruido primario que esté correlacionado con la entrada de referencia será efectivamente cancelado. Sin embargo se debe tener en cuenta que la realimentación entre la fuente secundaria y el sensor de referencia puede traer aparejados una serie de problemas. Cuando se trabaja con ruidos periódicos las técnicas de banda angosta no dependen de la causalidad, siendo posible utilizar sensores que no sean electroacústicos para obtener la señal de referencia (por ejemplo tacómetros en máquinas rotativas). Debido a que todo el ruido se concentra en armónicos de la tasa de rotación de una máquina en particular, el sistema de control podrá cancelar el ruido en estas frecuencias específicas. Esto es necesario, por ejemplo, en la cabina de un vehículo, en donde es de suma importancia no afectar las señales de advertencia y la palabra hablada, las cuales por lo general no están sincronizadas con la rotación del motor [2].

2.4.1 Sistema prealimentado de banda ancha

Un sistema ANC *prealimentado* o *feedforward* de banda ancha monocanal emplea dos micrófonos como sensores acústicos y un altavoz como fuente secundaria de control. La Figura 2.1 muestra un sistema ANC prealimentado de banda ancha monocanal aplicado a un ducto. El primer micrófono es el sensor de referencia, encargado de proveer al controlador una buena referencia del ruido a cancelar. El controlador debe procesar esta señal y emitir un antiruido a través de la fuente secundaria en el momento en que el ruido alcanza dicha fuente, de tal manera de que se produzca la interferencia destructiva de ondas. La *zona de quietud* o *cancelación* se forma en el espacio que rodea al segundo sensor, llamado *sensor de error*, el cual es el encargado de censar el resultado obtenido de la cancelación. La señal de error es solamente usada por el algoritmo de adaptación. Para

que este tipo de sistema funcione eficientemente el ruido a cancelar debe estar altamente correlacionado con la señal de referencia [2].

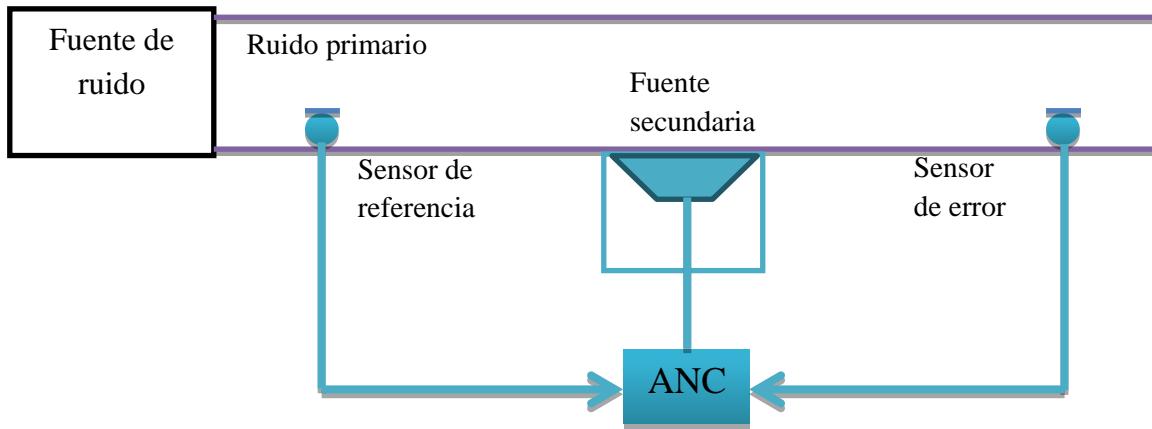


Figura 2.1: Sistema ANC prealimentado. Imagen adaptada de [2]

Un sistema de este tipo busca eliminar el ruido de banda ancha. La condición que se debe satisfacer es que el tiempo que hay desde que se toma la señal de referencia hasta que se emite un antiruido debe ser menor o igual al tiempo que se tarda el ruido en viajar desde la ubicación del sensor de referencia hasta la de la fuente secundaria de control. De no cumplirse, el sistema debería ser no causal para lograr la cancelación, lo cual resulta imposible de conseguir en la práctica. Vale decir que aun no cumpliendo con los requisitos de causalidad, el sistema puede todavía eliminar ruido de banda angosta, debido a que estos ruidos son predecibles. Un inconveniente que tiene este sistema es que parte de la señal emitida por la fuente secundaria de control se realimenta por el sensor de referencia, lo cual no es deseado [2].

2.4.2 Sistema prealimentado de banda angosta

Los sistemas ANC *prealimentados* o *feedforward* de banda angosta apuntan a eliminar los ruidos periódicos o quasi-periódicos emitidos por máquinas rotativas o repetitivas. Debido a que la información espectral de estos ruidos está fuertemente relacionada con la velocidad de rotación de las máquinas (en el caso de máquinas

rotativas), para proveer una señal de referencia al sistema utiliza un sensor no acústico (tacómetro o un sensor óptico), el cual censa la velocidad de dichas máquinas. Al no tener un sensor acústico, no tiene el inconveniente de realimentación del sistema anterior. Dado que su señal de referencia esta sincronizada a la velocidad de las máquinas, solamente los ruidos emitidos por estas serán cancelados. En este tipo de sistemas, como se busca cancelar ruidos periódicos, cumplir con la condición de causalidad no es necesario. Al igual que en el sistema anterior, la señal del sensor de error es solamente utilizada para el proceso de adaptación.

2.4.3 Sistema realimentado

Los sistemas ANC *realimentados* o *feedback* se diferencian de los prealimentados en que no tiene un sensor de referencia, razón por la cual es necesario realimentar la señal tomada por el sensor de error para realizar el control. Estos sistemas son muy usados cuando no es posible o es impráctico censar una referencia. Los sistemas ANC realimentados clásicos, los cuales no son adaptativos, procesan directamente la señal de error para generar una salida (antiruido). Al no ser adaptativos, no pueden ajustarse a los cambios de las características del ruido y del medio, por lo que son propensos de entrar en inestabilidad.

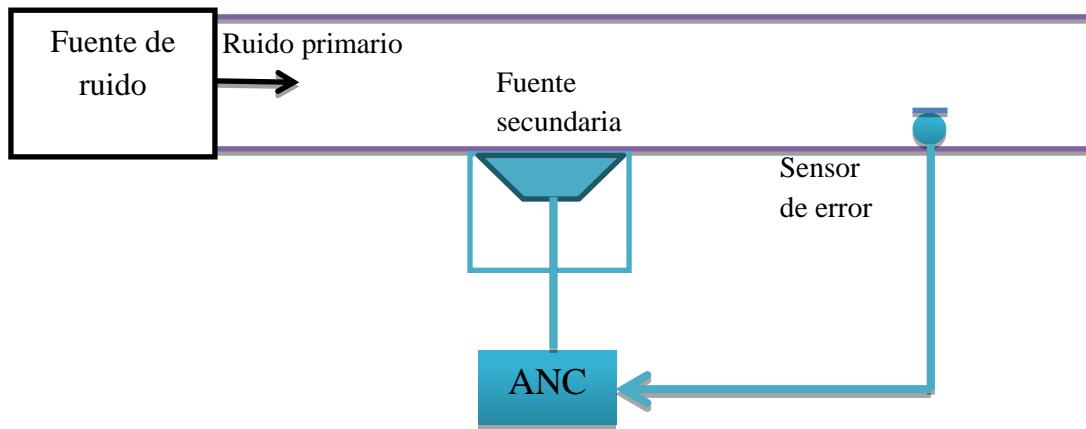


Figura 2.2: Sistema ANC realimentado. Imagen adaptada de [2]

La estabilidad de éstos depende del retardo de tiempo que hay entre la fuente secundaria de control y el sensor de error, el cual disminuye al ubicar ambos muy cerca entre sí [2].

Los sistemas ANC realimentados adaptativos, utilizan la señal de error para estimar una señal de referencia, además de usarla para el proceso de adaptación. Teniendo una referencia, el sistema toma la forma de uno prealimentado. Debido al retardo de tiempo inherente del sistema y a que se está estimando una referencia a partir del error, el controlador debe actuar en parte como predictor, por lo que el sistema puede cancelar ruidos de banda angosta solamente. La Figura 2.2 muestra un sistema ANC realimentado monocanal aplicado a un ducto.

2.4.4 Sistema híbrido

El sistema ANC híbrido combina ambos tipos de sistemas, es decir el prealimentado con el realimentado. De esta forma se beneficia de las características de los dos tipos de control. El sistema prealimentado cancela las componentes del ruido correlacionadas con la señal de referencia, mientras que el sistema realimentado cancela aquellos ruidos de banda angosta que no pudieron ser censados por el micrófono de referencia [1].

2.5 Filtros FIR Adaptativos

El principal objetivo de todo sistema ANC es lograr la atenuación de un ruido determinado mediante interferencia destructiva de ondas. Para esto el sistema debe implementar las funciones de transferencia necesarias de forma tal que, cuando las ondas sonoras generadas lleguen a los sensores de error, las mismas posean la misma amplitud y fase opuesta a las ondas correspondientes al ruido primario, produciendo la cancelación deseada. Dentro de un procesador digital de señal, estas funciones de transferencia pueden ser implementadas con filtros FIR (Finite Impulse Response) o con filtros IIR (Infinite Impulse Response). En las aplicaciones de ANC es común la utilización de filtros FIR,

debido a la estabilidad de los mismos. En las secciones siguientes se detalla el funcionamiento de los filtros FIR adaptativos.

2.5.1 Filtros discretos

Un filtro es un sistema diseñado para transformar el contenido espectral de la señal de entrada en una determinada manera [12]. Si las señales que procesa el filtro están en tiempo discreto, se dice que el mismo es discreto. Los sistemas lineales e invariantes en el tiempo (LTI: Linear Time - Invariant) son caracterizados completamente por su respuesta al impulso, ya que su salida se obtiene de realizar la convolución entre la entrada y dicha respuesta al impulso [12]. La ecuación (2.4) expresa la salida de un filtro discreto LTI en forma general [1]

$$y(n) = h(n) * x(n) = \sum_{k=-\infty}^{+\infty} h(k)x(n-k) = \sum_{k=-\infty}^{+\infty} h_k x(n-k) \quad (2.4)$$

siendo $h(n)$ la respuesta al impulso del sistema LTI, $x(n)$ la entrada e $y(n)$ la salida. Los filtros discretos lineales pueden ser de respuesta al impulso finita (FIR: Finite Impulse Response) o con respuesta al impulso infinita (IIR: Infinite Impulse Response). Por su definición la ecuación (2.4) correspondería a un filtro IIR. Las principales diferencias entre estos dos tipos de filtros son que el filtro FIR tiene una función de transferencia que contiene solamente ceros, por lo que es incondicionalmente estable y de fase lineal, en cambio el filtro IIR contiene ceros y polos, pudiendo entrar en inestabilidad. De los dos tipos de filtros, el más usado es el filtro FIR debido a su característica de estabilidad incondicional [1]. La salida de un filtro FIR causal está expresada como:

$$y(n) = h(n) * x(n) = \sum_{k=0}^{L-1} h(k)x(n-k) = \sum_{k=0}^{L-1} h_k x(n-k) \quad (2.5)$$

siendo h_k los coeficientes del filtro que son la respuesta al impulso, L la cantidad de coeficientes y $x(n)$ la entrada. La estructura de un filtro discreto tiene que ver con la manera en que se expresa matemáticamente la salida del filtro y por consiguiente la forma en que se lo implementa. Es así que podemos tener distintas maneras de expresar su salida, todas ellas equivalentes. Cada una de estas estructuras tiene sus ventajas y desventajas a la hora de su implementación. La estructura más común de un filtro FIR y la más usada también, es la transversal o estructura directa, la cual se obtiene de implementar tal cual como se expresa la ecuación (2.5) [1].

2.5.2 Filtro Wiener

Se plantea el problema de diseñar un filtro cuya respuesta al impulso sea tal, que la señal de salida del filtro sea lo más parecida posible a una señal deseada. Si la entrada al filtro y la salida deseada son procesos aleatorios estacionarios en sentido amplio (WSS: Wide Sense Stationary) ergódicos, entonces la medida del parecido entre la señal de salida del filtro y la señal deseada, puede evaluarse a través del error cuadrático medio (MSE: Mean Square Error). De esta manera la solución óptima que minimiza el valor del error cuadrático medio, da como resultado la solución óptima de Wiener o filtro de Wiener [13][14]. La Figura 2.3 ilustra este concepto.

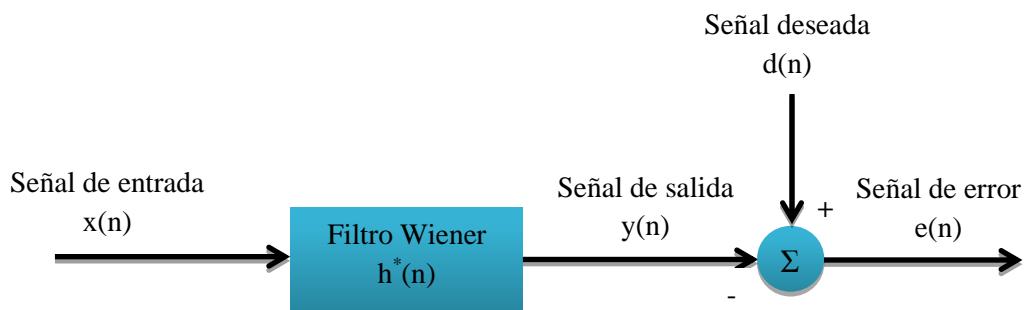


Figura 2.3: Filtro de Wiener

La señal de error $e(n)$ es la diferencia entre la salida deseada $d(n)$ y la salida del filtro $y(n)$:

$$e(n) = d(n) - y(n) \quad (2.6)$$

El MSE es la potencia de $e(n)$ y se lo conoce como la *función de desempeño*:

$$\xi(n) = MSE = E[e^2(n)] \quad (2.7)$$

Si el filtro es FIR transversal se puede demostrar que ξ en función de los coeficientes del filtro describe gráficamente una superficie cuadrática cóncava hacia abajo llamada *superficie de desempeño*, la cual tiene un solo mínimo [15]. Los coeficientes correspondientes a este mínimo son la respuesta al impulso del filtro Wiener y dependen de las propiedades estadísticas de la entrada y de la salida deseada. La Figura 2.4 muestra un ejemplo de una porción de la superficie cuadrática del MSE para un filtro FIR de dos coeficientes.

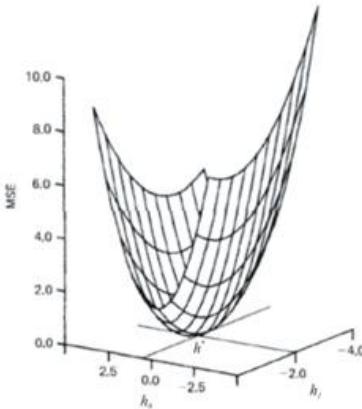


Figura 2.4: Porción de la superficie cuadrática del MSE para un filtro FIR de dos coeficientes.

Adaptado de [15].

La respuesta al impulso del filtro Wiener en forma general o forma no causal, se obtiene de minimizar la función de desempeño ξ para un filtro discreto invariante en el

tiempo expresado en la ecuación (2.8) [16]. El resultado que se obtiene es la ecuación de Wiener-Hopf:

$$\phi_{xd}(k) = h^*(k) * \phi_{xx}(k) \quad (2.8)$$

siendo $\phi_{xd}(k)$ la correlación cruzada de la entrada a la salida deseada, $\phi_{xx}(k)$ la autocorrelación de la entrada y $h^*(k)$ la respuesta al impulso del filtro Wiener. Aplicando la transformada Z a la ecuación (2.8) y reacomodando se tiene:

$$h^*(z) = \frac{\phi_{xd}(z)}{\phi_{xx}(z)} \quad (2.9)$$

siendo $\phi_{xd}(z)$ la transformada Z de la correlación cruzada de la entrada a la salida deseada, $\phi_{xx}(z)$ la transformada Z de la autocorrelación de entrada y $h^*(z)$ la transformada Z de la respuesta al impulso del filtro Wiener. La teoría del filtro Wiener, en aquellas tareas de estimación, se utiliza cuando los procesos aleatorios son estacionarios; si no lo fueran, la solución óptima estaría variando continuamente con el tiempo [17] [18] [19].

2.5.3 Filtros adaptativos

Para encontrar el filtro óptimo o filtro de Wiener se deben conocer los parámetros estadísticos de las señales involucradas. Dichos parámetros por lo general no están disponibles, por lo que es necesario estimarlos. Sin embargo, todo ese proceso de estimación es complejo, y más aún si después debe ser implementado en un DSP. Una forma más eficiente de obtener la solución óptima de Wiener, es usar un filtro adaptativo [15].

Un filtro adaptativo se compone de dos elementos: un filtro discreto encargado de realizar el procesamiento de señal y un algoritmo que modifica los coeficientes del filtro, de tal manera de aproximar la señal de salida del mismo a la señal deseada. El criterio usado en la adaptación es minimizar el valor cuadrático medio de la diferencia entre la señal deseada y la señal de salida del filtro, por lo que cuando el proceso de adaptación

finaliza el resultado que se obtiene es el filtro Wiener (si los procesos aleatorios involucrados son WSS ergódicos) [15][20]. Es necesario que la señal deseada este presente durante el proceso de adaptación, de tal manera de poder generar la señal de error. Así, la adaptación se realiza de manera recursiva, los valores de los coeficientes se van corrigiendo en el tiempo y el sistema va aprendiendo la respuesta al impulso óptima [1]. La Figura 2.5 ilustra un filtro adaptativo.

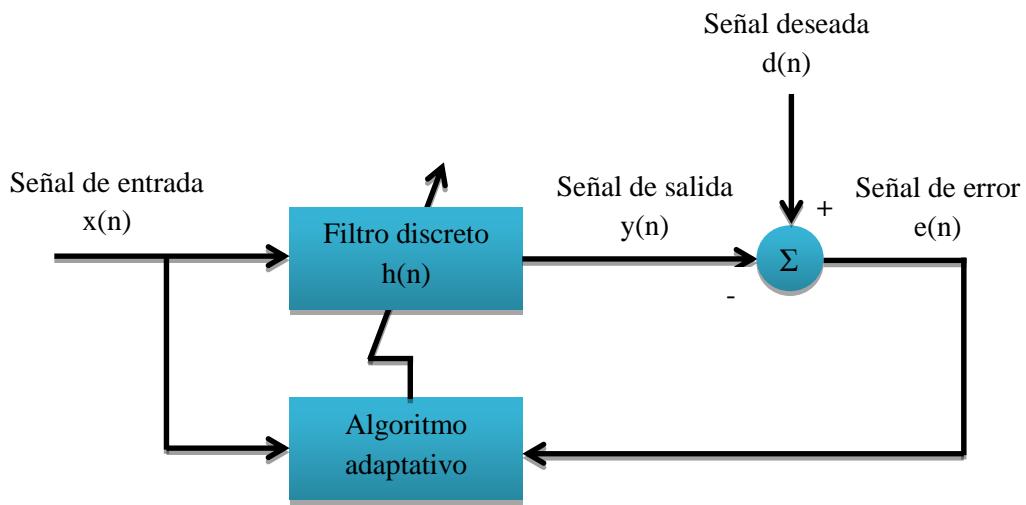


Figura 2.5: Filtro adaptativo. Imagen adaptada de [1]

Los filtros adaptativos tienen el efecto importante de poder ser aplicados donde la operación de filtrado requerida es desconocida o no estacionaria [15] [2] [20].

2.5.4 Método de máxima pendiente

Uno de los dos elementos de un filtro adaptativo, como se dijo anteriormente, es el algoritmo encargado de minimizar la función de desempeño ζ . Existen distintos tipos de algoritmos que cumplen este objetivo, cada uno aplicado a un tipo de estructura de filtro. Para el caso de la estructura transversal de un filtro FIR, ζ describe gráficamente una superficie cuadrática cuya forma en tres dimensiones es como la de un “paraboloide”. Los algoritmos desarrollados, buscan llegar al mínimo de dicho paraboloide descendiendo por

su superficie, lo que conlleva a la solución óptima de Wiener. Ejemplos de estos son el *método de Newton* y el *método de máxima pendiente* [15].

El método de máxima pendiente es un algoritmo que utiliza la dirección opuesta a la que apunta el gradiente de ζ para descender hasta el valor mínimo en el “paraboloide”. Como el gradiente apunta en la dirección de máximo crecimiento de la función, el moverse en la dirección opuesta hace que se descienda por el camino de máxima pendiente, llegando al punto mínimo rápidamente. Este método es iterativo y en cada paso se necesita calcular el valor del gradiente en el punto de la superficie donde se encuentre [1]. El algoritmo del método de máxima pendiente es el siguiente:

$$h_k(n+1) = h_k(n) - \mu \nabla \varepsilon_k(n) \quad k = 0, 1, \dots, L-1 \quad (2.10)$$

siendo $h_k(n)$ los coeficientes del filtro que ahora varían con el tiempo, L la cantidad de coeficientes, μ el tamaño de paso o factor de convergencia y $\nabla \varepsilon_k(n)$ las componentes del vector gradiente de la función de desempeño.

2.5.5 Algoritmo LMS

El método de máxima pendiente necesita calcular el valor del gradiente de ζ en cada paso de iteración. Se puede demostrar [15] que el valor de este gradiente depende de las propiedades estadísticas de las señales de entrada y deseada.

Como se dijo anteriormente estas propiedades estadísticas no se conocen en la mayoría de los casos y el proceso del cálculo de las mismas es complejo. El algoritmo LMS (Least Mean Square) es un algoritmo que emplea el método de máxima pendiente, pero en vez de usar el valor exacto del gradiente de la función de desempeño, utiliza una estimación del mismo [15][16]. Recordando que la función de desempeño es el error cuadrático medio tal cual como se expresó en la ecuación (2.7), una estimación somera del gradiente puede ser obtenida diferenciando el cuadrado del valor instantáneo del error como si fuera el error cuadrático medio [15][16]:

$$\hat{\nabla} \varepsilon(n) = \nabla[e^2(n)] \quad (2.11)$$

Siendo $\nabla\xi(n)$ la estimación del gradiente de la función de desempeño. Usando las ecuaciones (2.4), (2.5) y (2.6) se obtiene la estimación del gradiente de ξ :

$$\hat{\nabla}\xi_k(n) = -2x(n-k)e(n) \quad K = 0,1,\dots,L-1 \quad (2.12)$$

Sustituyendo la estimación del gradiente en la ecuación (2.10) se obtiene el algoritmo LMS:

$$h_k(n+1) = h_k(n) + 2\mu e(n)x(n-k) \quad k = 0,1,\dots,L-1 \quad (2.13)$$

Se observa que es un algoritmo que no requiere operaciones de cuadrado, promedio o diferenciación, y es elegante en su simplicidad y eficiencia [15][16]. Widrow junto a Hoff concibieron el algoritmo LMS, el cual es hoy en día el algoritmo de adaptación más usado en todo el mundo. Una versión más práctica del algoritmo LMS sale de incluir dentro de μ el 2 de la ecuación (2.13), por lo que finalmente el algoritmo LMS queda como:

$$h_k(n+1) = h_k(n) + \mu e(n)x(n-k) \quad k = 0,1,\dots,L-1 \quad (2.14)$$

De aquí en adelante, siempre que se refiera al algoritmo LMS, se estará refiriendo a la ecuación (2.14).

2.5.6 Desempeño del algoritmo LMS

Existen dos factores fundamentales a la hora de evaluar el desempeño de un algoritmo en particular: la estabilidad, y la velocidad de convergencia del mismo. A continuación se tratan dichos temas, considerando además el ruido en los coeficientes

2.5.6.1 Estabilidad

Uno de los principales aspectos a analizar del algoritmo LMS está referido a su convergencia. En general para que se consiga la convergencia del valor esperado de los coeficientes al filtro Wiener, el valor de μ debe estar dentro del siguiente rango:

$$0 < \mu < \frac{2}{LP_x} \quad (2.15)$$

en donde L es la cantidad de coeficientes y P_x la potencia de la señal de entrada $x(n)$. Se observa que en la inecuación (2.15) el valor máximo de μ que garantiza la convergencia del algoritmo LMS de la ecuación (2.14), es inversamente proporcional a la cantidad de coeficientes y a la potencia de la señal de entrada. En la práctica se usa típicamente la siguiente inecuación de μ [2]:

$$\frac{0.01}{LP_x} < \mu < \frac{0.1}{LP_x} \quad (2.16)$$

2.5.6.2 Velocidad de convergencia

La curva que describe la variación del MSE en función del tiempo, es conocida como la curva de aprendizaje del algoritmo LMS. Dicha curva decae exponencialmente al valor ξ_{min} a una velocidad determinada por su constante de tiempo τ_{MSE} [1]. Es posible demostrar que dicha constante de tiempo puede aproximarse por la siguiente ecuación

$$(\tau_{MSE})_k \approx \frac{T}{2\mu\lambda}(s) \quad (2.17)$$

La velocidad de convergencia del algoritmo LMS quedará determinada por la constante de tiempo $(\tau_{MSE})_k$ más grande, es decir por la que le corresponda el autovalor más chico λ_{min} .

2.5.6.3 Ruido en los coeficientes

La estimación del gradiente de ξ que emplea el algoritmo LMS, posee un error que se comporta como ruido:

$$\nabla \hat{\varepsilon}(n) = \nabla \varepsilon(n) + N(n) \quad (2.18)$$

siendo $\nabla \xi(n)$ el valor real del gradiente y $N(n)$ el ruido de la estimación.

Debido a este ruido en la estimación del gradiente, el proceso de adaptación introduce ruido en la solución de los coeficientes del filtro, y por consiguiente una pérdida del desempeño [47]. Este ruido en los coeficientes hace que en estado estacionario los valores de los mismos varíen aleatoriamente alrededor de la solución óptima.

El exceso del MSE se lo define como el promedio del incremento de ξ después de que se produce la convergencia, es decir en estado estacionario [15]:

$$\varepsilon_{exceso} = E[\varepsilon(n) - \varepsilon_{min}] \quad (2.19)$$

siendo ξ_{min} el valor del MSE correspondiente a la solución óptima. Se demuestra en [47] que el exceso del MSE para el algoritmo LMS es:

$$\varepsilon_{exceso} \approx \frac{\mu}{2} \varepsilon_{min} LP_x \quad (2.20)$$

siendo μ el tamaño de paso del algoritmo LMS de la ecuación (2.14), L la cantidad de coeficientes del filtro y P_x la potencia de la señal de entrada. La ecuación (2.20) es una aproximación válida para valores de μ pequeños. El exceso del MSE es una medida de la diferencia entre el desempeño actual y el óptimo promediada en el tiempo, del proceso adaptativo [15].

Existe una cuestión de compromiso entre el exceso del MSE y la velocidad de convergencia, ya que un aumento de μ implicaría un incremento en la velocidad de la adaptación, pero también aumentaría el ruido en los coeficientes. El desempeño del algoritmo LMS está determinado por su velocidad de convergencia y el ruido en los coeficientes que hace que varíen alrededor de la solución óptima. Dependiendo de la aplicación del filtro adaptativo y del resultado que se quiera conseguir, se decidirá los valores adecuados de los parámetros del algoritmo.

2.5.7 Algoritmo LMS normalizado

Si bien el algoritmo LMS es el más empleado a nivel internacional, existen variantes del mismo. Cada una de estas variantes es un algoritmo que mejora algún aspecto del LMS, en lo que a consideraciones prácticas se refiere [2]. En la inecuación (2.16) el valor máximo de μ que hace que el algoritmo LMS sea estable, siendo además inversamente proporcional a la potencia de la señal de entrada. Esto indica que en presencia de señales débiles se pueden usar valores grandes de μ y que en el caso de señales fuertes se deben usar pequeños valores. Algo muy usado es normalizar μ con respecto a la potencia de la señal de entrada. El algoritmo normalizado LMS o NLMS emplea esto, el cual es expresado como:

$$h_k(n+1) = h_k(n) + \mu(n)e(n)x(n-k) \quad k = 0, 1, \dots, L-1 \quad (2.21)$$

$$\mu(n) = \frac{\alpha}{L\hat{P}_x(n)} \quad (2.22)$$

siendo $\mu(n)$ el tamaño de paso normalizado, $\hat{P}_x(n)$ una estimación de la potencia de la señal de entrada, L la cantidad de coeficientes y α un factor. Para limitar el valor de $\mu(n)$ en ausencia de señal, se debe fijar un valor mínimo para $\hat{P}_x(n)$. El algoritmo NLMS es una importante técnica para optimizar la velocidad de convergencia mientras se mantiene el deseado desempeño en estado estacionario independiente de la potencia de la señal [2]. La potencia de la señal de entrada puede ser estimada usando una ventana exponencial de la siguiente manera [2]:

$$\hat{P}_x(n) = (1 - \beta)\hat{P}_x(n-1) + \beta x^2(n) \quad (2.23)$$

Siendo β un factor de alisado, el cual está expresado en función de la longitud efectiva de la ventana M :

$$\beta = \frac{1}{M} \quad (2.24)$$

Para una alisada estimación de la potencia, la longitud M debe ser grande, pero esto evita que se puedan seguir cambios rápidos en la potencia de la señal. Por lo que si la señal es estacionaria se podría usar un valor de M grande [1].

2.5.8 Algoritmo FxLMS

En la Figura 2.6 se muestra un sistema con el algoritmo LMS, en donde M es el modelo de referencia, W es el controlador, y S_e es el modelo de planta. A partir de S_e se obtienen los coeficientes de W , del cual se realiza una copia para controlar la planta S [1].

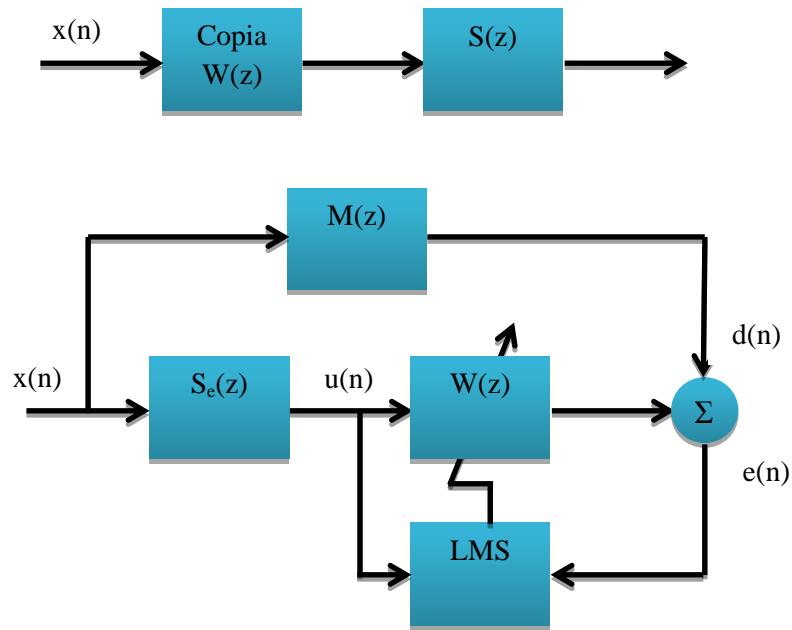


Figura 2.6: Control inverso adaptativo utilizando el algoritmo LMS. Imagen adaptada de [21].

Para poder utilizar el algoritmo LMS en el control activo de ruido es necesario realizar una extensión del mismo, ya que la salida del controlador fue restada directamente de la señal de excitación. En las aplicaciones prácticas existe una función de transferencia entre el controlador digital y el mundo físico, conteniendo el conversor digital analógico, la amplificación, la atenuación y las respuestas de los transductores. Este camino secundario

introduce modificaciones en el módulo y fase que deben ser compensados por el filtro adaptativo para asegurar la convergencia [1]. El algoritmo LMS con Filtro X (FxLMS) filtra la entrada antes de que sea utilizada por el LMS para actualizar los coeficientes del controlador [1].

En la Figura 2.7 se observa que la actualización de los coeficientes se realiza empleando el error del sistema e . Aunque el modelo de planta S_e tenga errores, el algoritmo FxLMS minimizará el error del sistema optimizando el filtro de control $W(z)$ [1].

Para un estudio más profundo de los filtros FIR adaptativos se recomienda consultar la sección 2 del proyecto integrador “*Controlador Activo Adaptativo de Ruido Acústico Periódico para un protector de Audición y o Auricular*” [1].

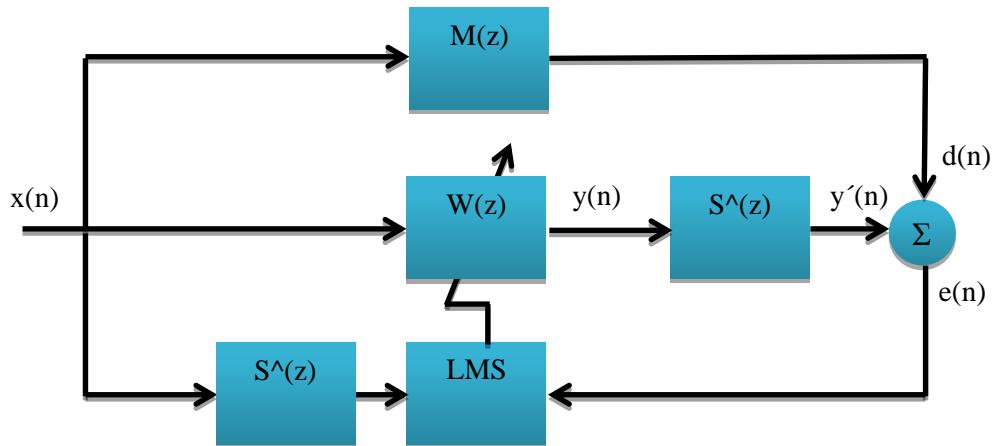


Figura 2.7: Algoritmo FxLMS. Imagen adaptada de [2]

CAPÍTULO 3: Definición del modelo

En este capítulo se presentan los fundamentos que justifican el tipo de ANC seleccionado, y el diseño y construcción del recinto sobre el cual se implementó el sistema. Además, se describe el proceso de selección y construcción de todas las partes que componen el sistema de control (excluyendo el DSP), abarcando tanto las fuentes secundarias y su etapa de potencia, como los sensores de error y el acondicionamiento de señal correspondiente.

3.1 Selección del sistema ANC

Debido a que el sistema de control fue aplicado a la cabina de un automóvil, resulta crucial que no se vieran afectadas tanto las señales de advertencia provenientes del interior o del medio externo, como la palabra hablada en el interior del recinto. En esta aplicación se buscó cancelar únicamente el sonido proveniente del motor del automóvil, el cual está relacionado con las rpm (revoluciones por minuto) del mismo, razón por la cual se optó por aplicar un sistema realimentado.

3.2 Selección de control global o local

Cuando se trata de controlar el campo sonoro en un recinto son posibles dos aproximaciones: a) *control global*; b) *control local*. El control global busca reducir la energía potencial acústica total del recinto. Por otro lado, el control local es una estrategia para crear una zona de quietud, lo cual puede causar un incremento de los niveles de ruido en los espacios más alejados de los sensores de error [2]. A continuación se presentan aspectos claves a la hora de seleccionar el tipo de control a implementar:

- Elliott y Nelson [21] implementaron un sistema ANC global en donde la fuente primaria de ruido se encontraba en una esquina del recinto, y la fuente secundaria en la esquina opuesta del mismo. Para aquellas frecuencias de

excitación en donde un sólo modo propio domina la respuesta del recinto, se lograron grandes reducciones de energía con una única fuente secundaria, ya que fue posible excitar el modo dominante en un grado igual y opuesto que la fuente primaria. Sin embargo, para frecuencias de excitación entre frecuencias naturales de los modos de resonancia de baja frecuencia, o en la mayoría de las frecuencias por encima de 200Hz, varios modos propios contribuyen la respuesta del recinto. En estos casos se observó que la fuente secundaria era incapaz de controlar cualquiera de estos modos acústicos sin aumentar la excitación de un número mayor de modos, logrando una menor reducción en la energía potencial acústica total. Sería lógico pensar que aumentando el número de fuentes secundarias aumentaría el número de modos que pueden ser controlados activamente. Sin embargo, el número de modos que contribuyen significativamente en un recinto crecen a mayores frecuencias en una proporción aproximada al cubo de la frecuencia de excitación, limitando la aplicación del ANC global a unos pocos cientos de Hz [21]. Es decir, el número de fuentes secundarias necesarias para lograr una perfecta cancelación (global) en un recinto es el mismo que el número de modos normales que han sido excitados, el cual puede ser muy grande [2]. Esto implica que *el control global no puede ser realizado con un número bajo de fuentes de control* [19].

- Al realizar un muestreo espacial del campo sonoro en un recinto con un número determinado de sensores de error, debe recordarse que el mismo presenta una exactitud reducida para representar el recinto en su totalidad, siendo posible que el nivel general de la energía potencial acústica no alcance el mínimo teórico. Esto tiene aún mayor influencia cuando se considera un ANC multicanal en un caso fuera de resonancia (es decir que no se ha excitado un único modo en particular), donde el campo sonoro resultante es más complicado. Sin embargo, un sistema ANC no siempre busca lograr una reducción de la energía sonora total del recinto, sino que *en la mayoría de los casos es suficiente con lograr zonas de quietud alrededor*

de la cabeza del usuario, siendo irrelevante un aumento del ruido en posiciones que no son de interés [2].

- Un sistema de control global es, a menudo, más complicado de implementar que un sistema local, ya que las ubicaciones tanto para los altavoces como para micrófonos son cruciales para los resultados obtenidos. Se requiere *mucho más esfuerzo para instalar tal sistema para el control global* de ruido, en oposición a un sistema para control local. En nuestra aplicación, el *control local es adecuado puesto que el conductor tiene una ubicación fija* [22].

Analizando las ventajas y desventajas presentes en cada tipo de sistema, *se optó por implementar un sistema de control local*, y trabajar con un recinto que tenga una distribución lo más uniforme posible de modos propios para evitar coloraciones intensas.

3.3 Diseño, construcción y medición del recinto

Como se mencionó en el capítulo 2, la existencia de modos propios en el interior de un recinto cerrado es inevitable. Debido a esto conviene elegir una relación entre las dimensiones del recinto de forma tal que los mismos se encuentren uniformemente distribuidos en función de la frecuencia. Si se realiza un diseño basándose en algunos de los distintos criterios utilizados para tal fin, es posible evitar que la energía se concentre en bandas estrechas de frecuencia [6].

El criterio de Richard H. Bolt [23] permite obtener distintas proporciones en las dimensiones de una sala durante la etapa de diseño, de tal forma de producir las características más suaves en baja frecuencia. Esto es aplicable en recintos rectangulares de pequeñas dimensiones. Como ejemplo, en la Figura 3.1 se representa la distribución de los modos propios más significativos de dos salas: a) con una relación óptima entre sus dimensiones (6,25 m x 3,75 m x 2,5 m), b) de forma cúbica (4 m x 4 m x 4 m) [6].

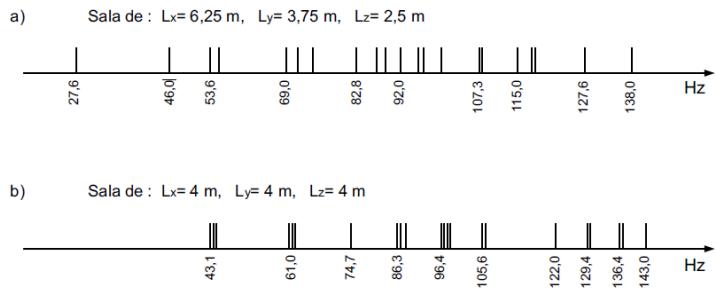


Figura 3.1: Distribución de modos propios en dos salas. Imagen tomada de [25]

En la Figura 3.2 se presenta el ábaco de Bolt, en el cual a partir de una altura unitaria $H = 1$, podemos obtener distintas relaciones dimensionales del largo y el ancho, dentro del área punteada, con valores razonables de diseño [7].

Esto verificaría que, cualquier relación que se halle dentro del área de Bolt, producirá una buena calidad en bajas frecuencias en la sala, tanto como sea posible en relación con la distribución de los modos axiales. [24]

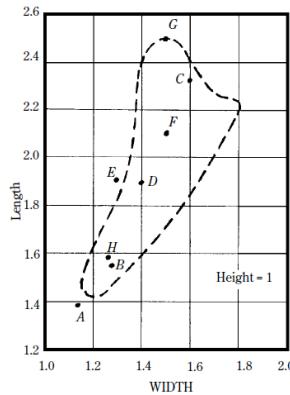


Figura 3.2: Gráfico de proporciones favorables para salas rectangulares para lograr una distribución modal de frecuencias uniforme. La línea de puntos encierra la llamada área de Bolt. Las proporciones A y E están fuera del área recomendada; G sobre el perímetro del área; C, F, D, H y B en el interior del área recomendada. Tomado de [23]

3.3.1 Diseño de la cabina

A continuación se presentan los planos de la cabina

- Vista lateral

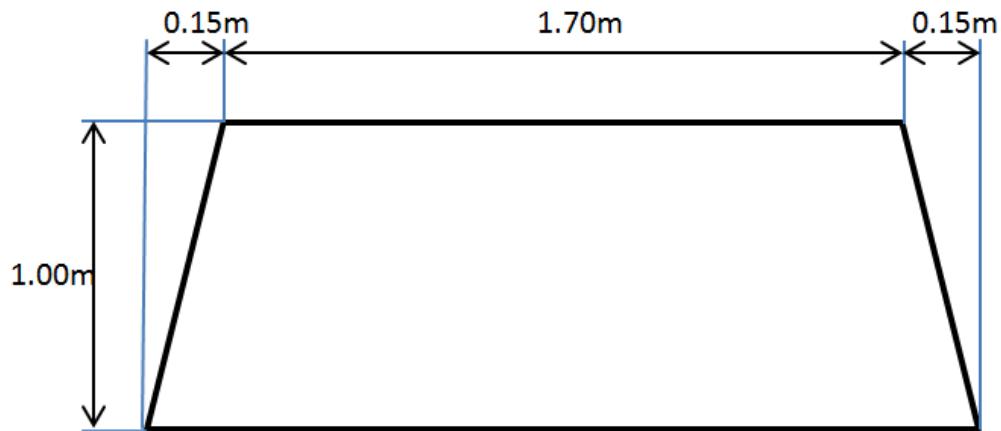


Figura 3.3: Plano de vista lateral de la cabina

- Vista frontal/posterior

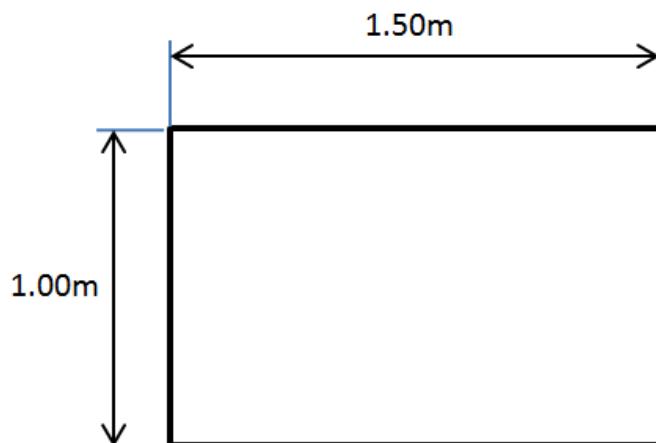


Figura 3.4: Plano de vista frontal y posterior de la cabina

- Vista superior

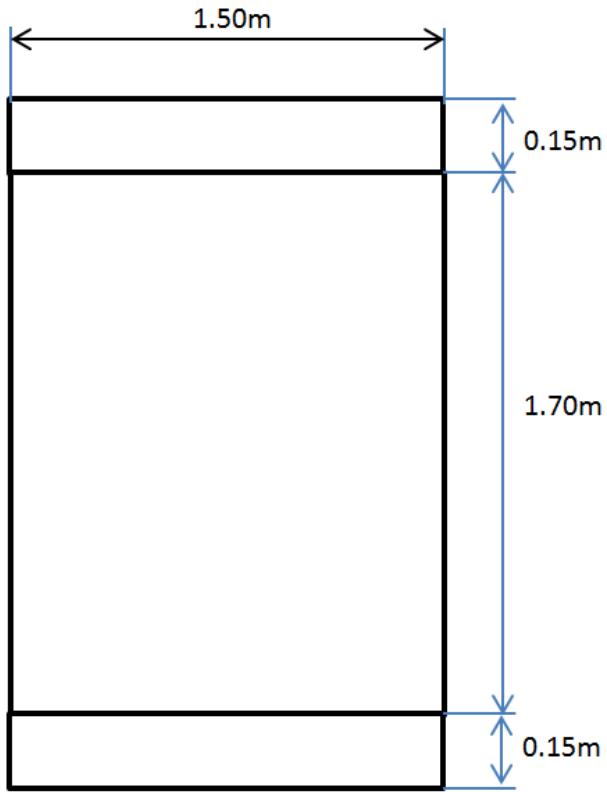


Figura 3.5: Plano de vista superior de la cabina

Las dimensiones para la verificación de las proporciones de Bolt para evitar coloraciones en baja frecuencia son: alto = 1; ancho = 1,5; largo = 1,85. Ubicando el punto en la gráfica de las proporciones se comprueba que se encuentra dentro del área de Bolt.

3.3.2 Construcción de la cabina

La cabina se construyó con madera tipo fibra de media densidad (MDF: Medium-Density Fibreboard) y fenólico, ambas de 9mm de espesor, por su costo y peso.

En el análisis del desempeño de un sistema de control activo local, [25] plantea que la reducción sonora en una cámara semi anecóica mejora comparado con un recinto

reverberante debido a la simplicidad del campo acústico. Las cámaras anecóicas son recintos cerrados construidos con el fin de verificar en su interior las características físicas de campo libre, mientras que las cámaras semi anecoicas son recintos que poseen una de sus caras interiores reflejantes, habitualmente el piso, y el resto absorbente [26]. Por otro lado, [9] define una habitación reverberante como “*aquella en donde el campo sonoro es lo más difuso posible, obteniendo tiempos de reverberación de por lo menos 5s*” [9]. Si las paredes del recinto son muy poco absorbentes, las funciones de transferencia entre las fuentes y los sensores pueden presentar grandes y agudas atenuaciones en frecuencias discretas denominadas “*notches*”. En sistemas de ANC siempre es mejor operar con caminos primarios y secundarios con una alta relación señal directa/reverberante, debido a que sus respuestas son más suaves, y permiten un mejor acoplamiento entre la energía de la fuente secundaria y los micrófonos de error [2]. De esta manera, con el objeto de reducir el tiempo de reverberación y simplificar el campo sonoro, la cabina fue recubierta en su interior con material absorbente acústico (guata y espuma de poliuretano) aumentando enormemente los coeficientes de absorción sonora de las paredes.

A continuación, en las figuras 3.6 y 3.7 se presentan fotografías de la cabina terminada.



Figura 3.6: Vista exterior del recinto construido



Figura 3.7: Vista interior del recinto construido

3.3.3 Cálculo de modos propios

Tal como se planteó en la sección anterior, las frecuencias de los modos propios en el interior de un recinto dependen de su geometría y dimensiones. En recintos paralelepípedos con superficies reflectantes, las frecuencias de los distintos modos pueden ser calculadas mediante la fórmula de Rayleigh (Ecuación (2.2), Capítulo 2).

Utilizando esta fórmula, se procedió a calcular la frecuencia de los modos propios presentes en el recinto. Sin embargo es importante tener en cuenta que esta no tiene en cuenta los materiales que componen las paredes del recinto con su correspondiente coeficiente de absorción sonora. Para poder predecir de forma adecuada el comportamiento modal del recinto, sería necesario llevar a cabo una simulación computacional que incluya tanto estos factores, como la posición y características del par fuente-receptor correspondiente. A continuación, en la Tabla 3.1, se presentan las frecuencias calculadas correspondientes a los modos propios, hasta un límite superior de 500Hz, en donde Tipo 0 se refiere a un modo oblicuo (formado como resultado de la reflexión de una onda estacionaria entre seis superficies), Tipo 1 a un modo tangencial (entre cuatro superficies) y Tipo 2 a un modo axial (entre dos superficies).

Tabla 3.1: Tipo y frecuencia de los modos propios de vibración calculados para recinto

Tipo	Frecuencia (Hz)	Tipo	Frecuencia (Hz)
2	92,70	1	388,04
2	114,33	1	389,90
1	147,19	1	389,90
2	171,50	0	394,53
2	185,41	0	398,80
1	194,95	0	406,32
1	206,12	1	408,55
1	217,82	1	412,23
0	226,00	0	422,53
2	228,67	0	424,25
1	246,74	0	425,95
1	252,56	1	435,65
0	277,24	1	441,58
1	285,83	0	452,01
1	294,39	0	456,14
0	300,49	2	457,33
1	300,69	2	463,51
1	326,74	1	466,63
0	340,70	0	468,19
2	343,00	0	473,71
2	343,00	1	477,41
0	346,16	1	485,08
1	355,31	1	488,43
1	355,31	1	493,49
1	360,05	0	493,85
1	361,55	1	494,22

A continuación en la Figura 3.8 se presenta el gráfico correspondiente a dicha tabla

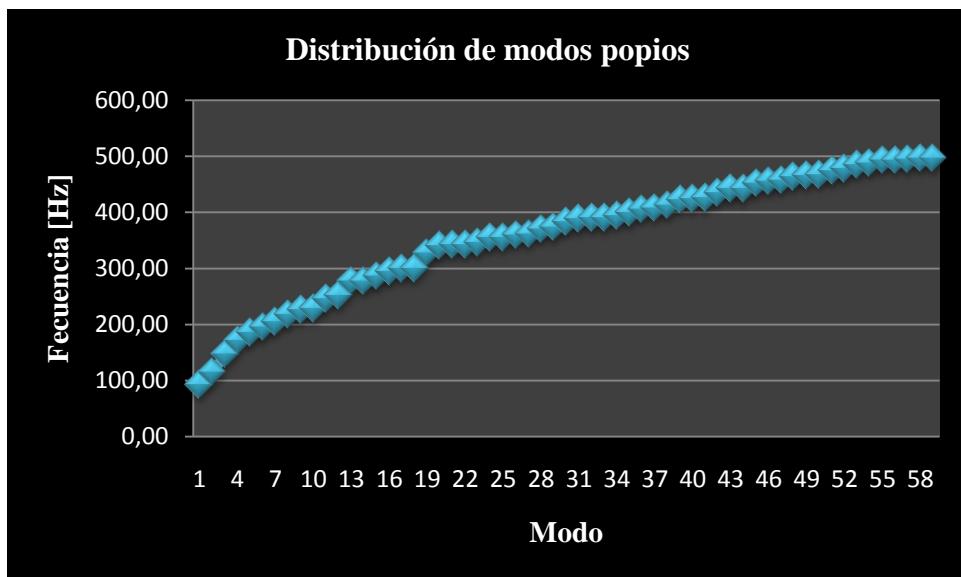


Figura 3.8: Distribución de los modos propios de vibración en el interior del recinto construido.

Se observa claramente que las frecuencias de los modos propios del recinto en el rango de interés se hallan uniformemente distribuidas en función de la frecuencia, ya que en ninguna unidad de frecuencia (Hz) por debajo de los 500 Hz, se presenta un conjunto grande de modos propios correspondiente a dicha frecuencia.

3.3.4 Medición de la respuesta impulsiva de la cabina

Se realizó la medición de la respuesta impulsiva de la cabina para poder identificar las frecuencias de los modos propios por debajo de 500Hz.

3.3.4.1 Equipamiento y configuración utilizados

Las mediciones se llevaron a cabo utilizando una computadora de escritorio con un procesador **AMD Phenom II X4** y 2GB de memoria RAM, con la placa de sonido **ESI**

Maya 44. Antes de comenzar las mediciones se realizó una evaluación del sistema conectando la entrada de audio de la placa de sonido a la salida de audio de la misma, el cual indicó que el procesador y la placa de sonido son aptos para realizar las mediciones deseadas. En la Tabla 3.2 se muestra el resultado de dicho análisis

Tabla 3.2: Resultado de la evaluación del sistema de medición

Parámetro	Valor
Respuesta en frecuencia (entre 20 y 20000 Hz)	-0,45; +0,01 dB
Interferencia y ruido	-93,1 dB
Rango dinámico	87,9 dB
Crosstalk (stereo)	-91,3 dB
THD + ruido	0,012%

Las mediciones se realizaron con una señal MLS de 23,8 segundos de duración. Estas secuencias de máxima longitud (MLS) se definen como “*secuencias discretas de ceros y unos (binarias) que tiene la propiedad de ser periódicas y su espectro continuo; de igual energía en todo el rango de frecuencias de interés*” [32].

El micrófono utilizado para la medición de la respuesta impulsiva es el **JTS PDM57**, el cual es un micrófono dinámico con un patrón de directividad de tipo cardioide. Al no ser un micrófono omnidireccional, y colocar el mismo apuntando hacia la fuente sonora, debe tenerse en cuenta que el sonido reflejado en las superficies opuestas a la dirección en donde la sensibilidad de micrófono es mayor, están siendo atenuadas.

La fuente utilizada está compuesta por dos cajas acústicas activas direccionales de dos vías **Behringer B1030A**, cuya respuesta en frecuencia de 40a 20000Hz y su potencia nominal de 50 W_{RMS} (cada una) hacen que sean óptimas para este tipo de medición.

Con el objetivo de medir el comportamiento de los modos propios en el interior del recinto, se colocaron las cajas acústicas en una esquina del recinto en dirección opuesta de modo de excitar la mayor cantidad de modos propios posibles, y se colocó el micrófono en la esquina opuesta del recinto. Es necesario aclarar que la medición realizada caracteriza únicamente el comportamiento modal para esa configuración espacial determinada del par

fuente-receptor, por lo tanto para conocer el comportamiento del recinto en las condiciones que operará el ANC, deberá medirse la respuesta impulsiva con cada par fuente-receptor correspondiente. A continuación en la Figura 3.9 se presenta un plano con las posiciones de fuente y de micrófono utilizadas

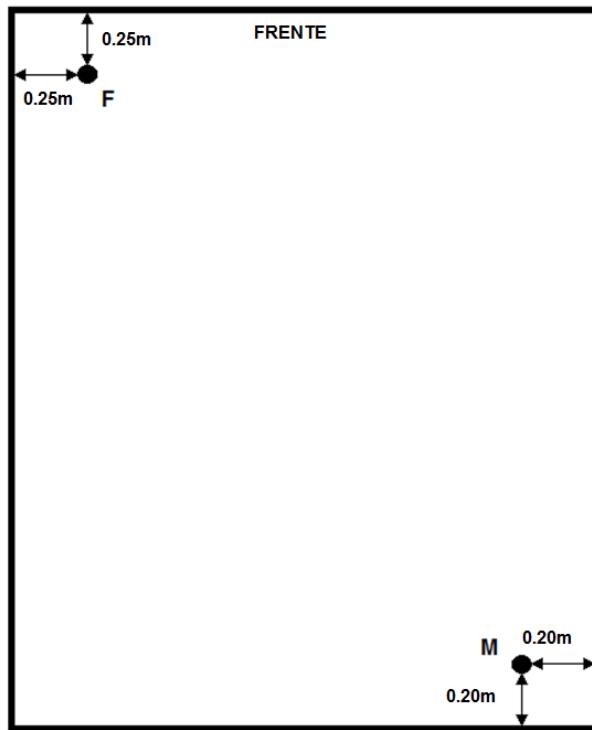


Figura 3.9: Plano de la configuración utilizada para la medición.

3.3.4.2 Resultados

En la Figura 3.10 se presenta el resultado de la medición de la respuesta impulsiva de la cabina (y toda la cadena de medición) en el dominio de la frecuencia (magnitud de la función de transferencia), utilizando como señal de excitación una secuencia MLS de 23,8 segundos de duración, para obtener una buena resolución en baja frecuencia.

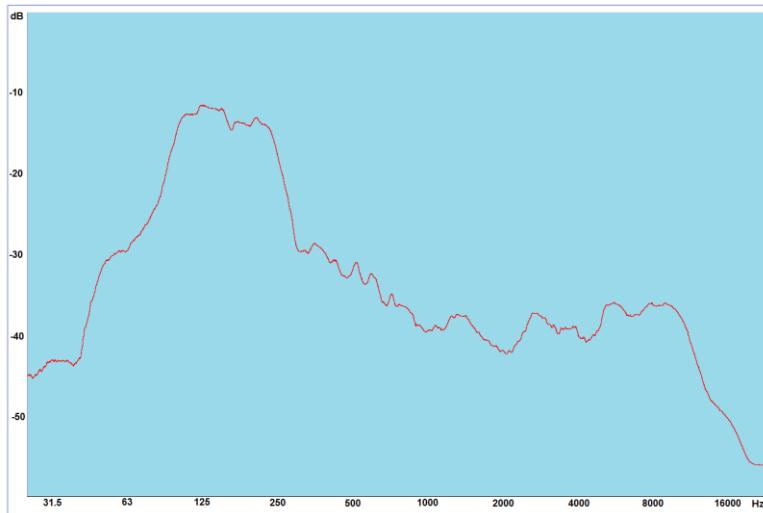


Figura 3.10: Análisis en frecuencia de la medición de la respuesta impulsiva del recinto.

A continuación, se presenta el espectrograma y el gráfico en cascada (Figura 3.11 y Figura 3.12 respectivamente) que muestran la evolución temporal de dicha respuesta impulsiva en el dominio de la frecuencia, con escala logarítmica, a modo de poder identificar mediante un cuidadoso análisis gráfico la influencia y distribución de los modos propios en el recinto por debajo de los 500 Hz. Ambos se realizaron con una FFT de 32768 puntos.

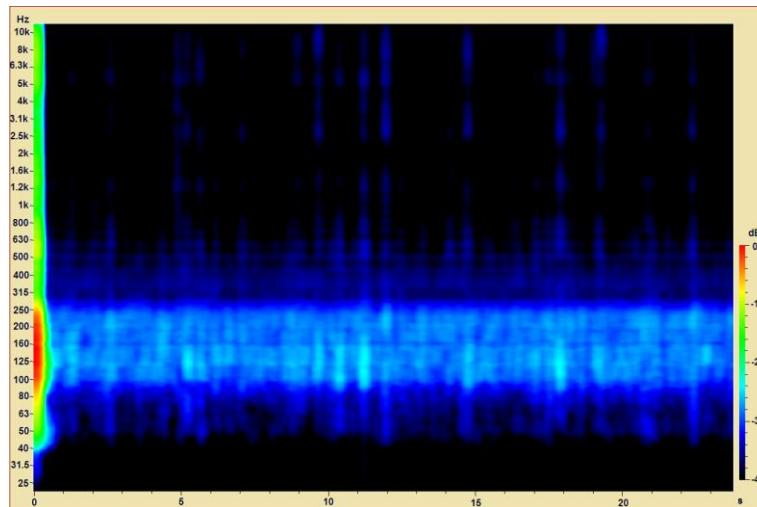


Figura 3.11: Espectrograma bidimensional de la respuesta impulsiva del recinto

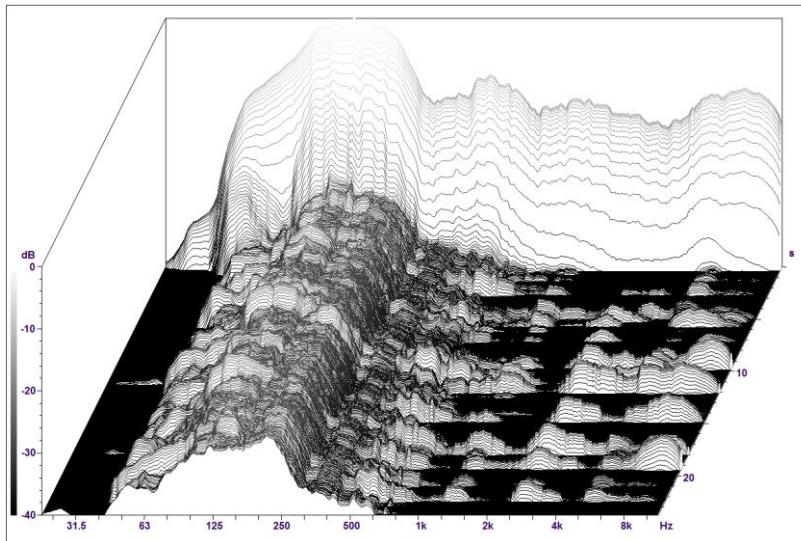


Figura 3.12: Espectrograma tridimensional (representación en cascada) de la respuesta impulsiva del recinto

A partir del análisis de los gráficos, se obtuvo información relevante respecto a la amplitud de los picos de resonancia causados por los modos propios: los picos de mayor amplitud se encuentran uniformemente distribuidos entre un límite inferior de 95Hz y un límite superior de 240Hz. En este rango de frecuencias se observa que, una vez finalizada la señal de excitación, la energía se mantiene sin atenuarse a lo largo del tiempo debido a las ondas estacionarias generadas por la interferencia constructiva y destructiva de ondas.

Esto implica que independientemente de cuál sea el espectro de la señal emitida por la fuente de ruido, la cabina resonará principalmente en el rango especificado anteriormente, lo cual hizo necesario prestar especial atención al control de ruido en dichas frecuencias.

Los resultados presentados corresponden al par fuente-receptor especificado, cuyas posiciones fueron elegidas de forma tal que se logre excitar la mayor cantidad de modos posibles para poder identificar los mismos. Además, la función de transferencia obtenida no incluye únicamente a la respuesta del recinto, sino que también incluye la función de transferencia de toda la cadena de medición utilizada. Debe recordarse que la medición se realizó con una fuente que no es omnidireccional, y con un micrófono que presenta una respuesta en frecuencia muy distante de ser plana, y patrón polar cardioide. Para obtener la respuesta únicamente de la cabina, debería filtrarse la medición obtenida

con un filtro inverso al de la cadena de medición. En la siguiente sección se especifica con mayor detalle el procedimiento que se debería llevar a cabo para tal fin.

Las posiciones de fuente y receptores a utilizar en el control activo de ruido, y las fuentes y sensores en sí mismas, difieren de las anteriores, por lo que fue necesario realizar la medición de las respuestas impulsivas correspondientes a cada uno de los pares fuente-receptor.

3.4 Localización de los transductores

3.4.1 Principios básicos para su determinación

En cuanto a la ubicación de los sensores de error y de las fuentes secundarias, Kuo [2] planteó lo siguiente:

- *“Para reducir la demora mantener distancias cortas entre los sensores de error y las fuentes secundarias.”* [2].
- *“Para que haya estabilidad cada sensor de error debe estar localizado a menor distancia de la fuente secundaria que corresponde a él, que de las otras.”* [2].
- *“Las ubicaciones en donde se encuentran los parlantes en los autos suelen ser posiciones aceptables para el control de ruido, mientras que las posiciones de los micrófonos de error pueden determinarse analizando los modos propios de la cabina.”* [2].
- *“La mayoría de los sistemas ANC confían mucho en DSP para lograr el resultado, pero si el arreglo físico no está optimizado, el sistema puede fallar. Es necesario entender el punto de vista acústico de la instalación y aplicar técnicas de diseño apropiadas para asistir al control adaptativo digital.”*[2].

Según lo planteado anteriormente, las posiciones de los sensores de error son de gran importancia. En base a esto, se procedió a realizar un análisis del comportamiento modal del recinto para determinar las localizaciones óptimas de los transductores, basándonos en los cálculos y mediciones realizados en la sección 2.

En la Figura 3.13 se presenta un gráfico a escala con las posiciones seleccionadas, en donde los puntos azules representan las fuentes, y los puntos verdes corresponden a los sensores de error.

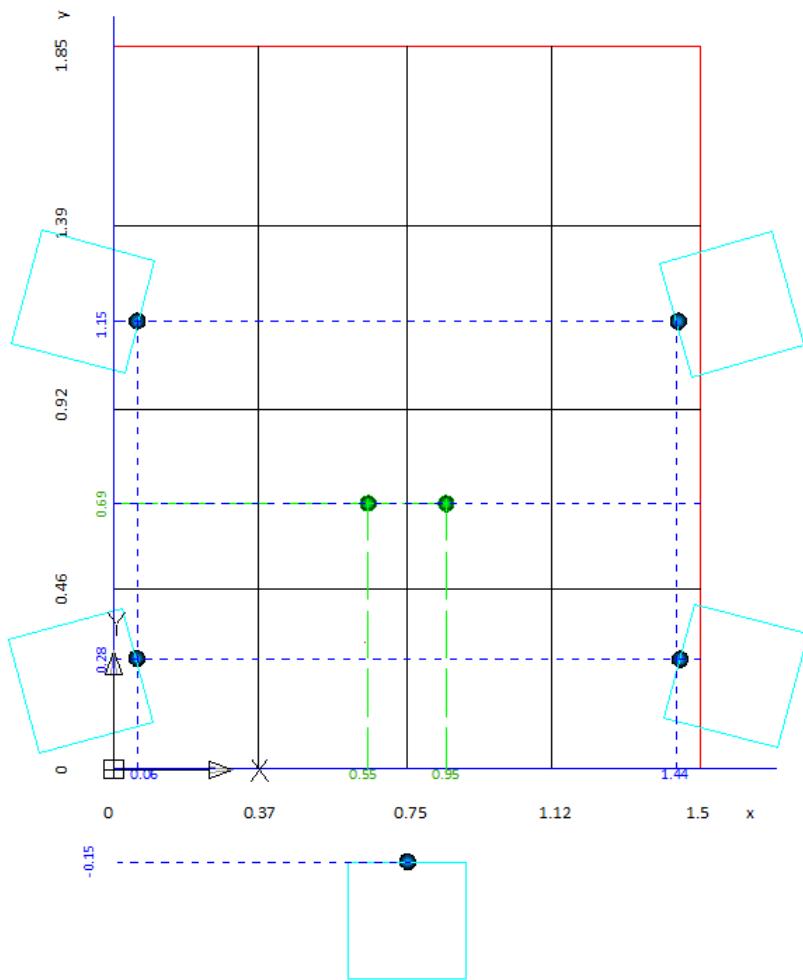


Figura 3.13: Localización de fuentes y sensores seleccionada

La altura seleccionada para ubicar las fuentes es $z = 0.20\text{m}$, mientras que la altura de los sensores es $z=0.85\text{m}$. Con respecto a las fuentes, se debe aclarar que es el centro del altavoz el que debe estar situado a dicha altura, y no la base de la caja acústica.

3.5 Transductores

Se realizó un estudio riguroso para la selección de los distintos transductores a utilizar, y se diseñaron los recintos(caja acústica tipo reflector de bajos) para los altavoces siguiendo el método propuesto por Thielle-Small en [27]. En las secciones siguientes se detallan los procedimientos llevados a cabo.

3.5.1 Fuentes secundarias

En esta sección se detalla la selección de los altavoces para las fuentes secundarias y el diseño y construcción de los recitos.

3.5.1.1 Selección de los altavoces

A continuación, se presentan en la Tabla 3.3 los diferentes modelos de altavoces analizados para la aplicación. Para la selección de los mismos se tuvieron en cuenta factores tales como la potencia, la respuesta en frecuencia y la sensibilidad de los mismos.

Se seleccionaron los altavoces Full Energy S300 por su respuesta en frecuencia, potencia nominal y valor económico.

Tabla 3.3: Comparación de las principales características de altavoces disponibles en el mercado

Modelo	Potencia [W _{PMPO}]	Potencia [W _{RMS}]	Tamaño [pulgadas]	Respuesta en Frecuencia [Hz]	Sensibilidad [dB@1W1m]	Precio 4 unidades [\$]
FOXTEX ML-80	180	45	8"	40-4500	90	360
FOXTEX ML-60	160	30	6"	60-5500	90	280
TARGA TAG5200	165	20	5.25"	70-26000	92	570
TARGA TAG6300	-	35	6.5"	50-22000	93	735
PIONEER 1344	220	35	5"	43-27000 - 20Db	89	1170
B52 WA 6181	-	30	6.5"	55-22000	92	735
BOSS CH5520	200	-	5.25"	100-18000	90	550
BOSS Bp8.8	500	-	8"	35-6000	94	-
Nipon America	-	30	6"	55-5000	-	240
Full Energy S300	150	50	6"	40-20000	-	544

3.5.1.2 Medición de parámetros TS

En la Figura 3.14 se presenta la medición de la impedancia del altavoz en campo libre, la cual se llevó a cabo con el dispositivo diseñado y construido para tal fin por el ingeniero Gabriel Aguilar [28] “Sistema de medición de parámetros de altavoces Thiele-Small”.

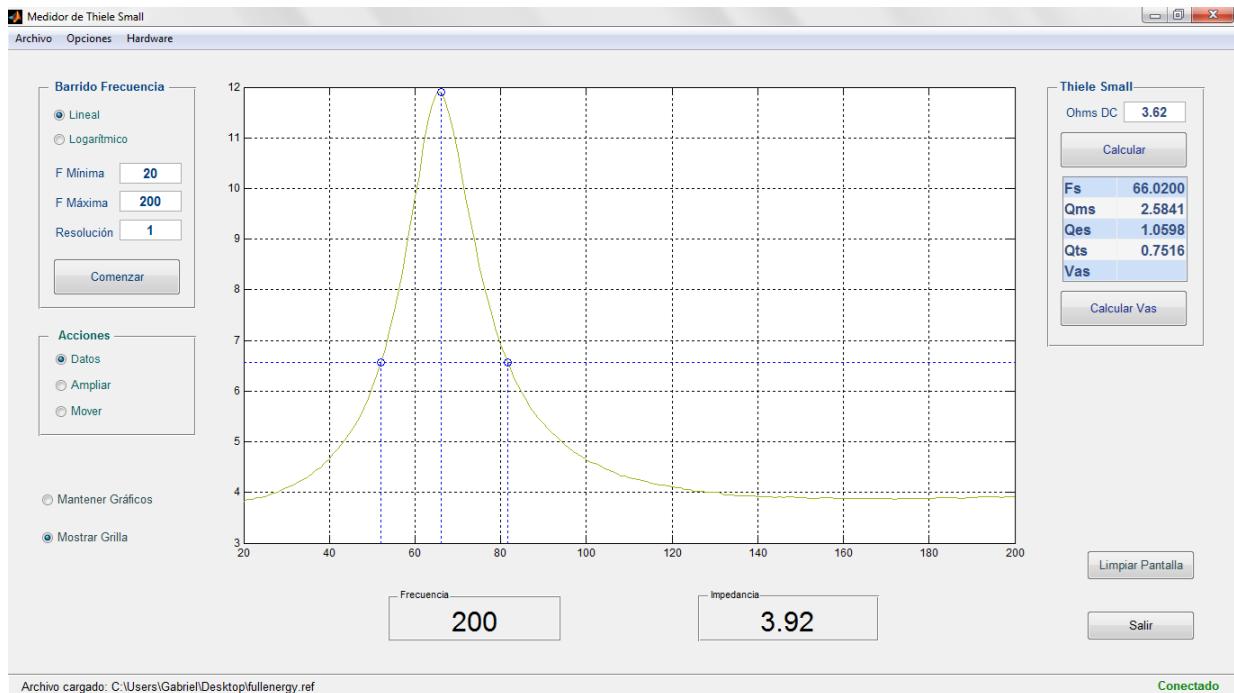


Figura 3.14: Impedanciometría del altavoz en campo libre

Los parámetros de Thiele Small medidas son:

- $f_s = 66,02 \text{ Hz}$
- $Q_{ms} = 2,5841$
- $Q_{es} = 1,0598$
- $Q_{ts} = 0,7516$

Para poder determinar el valor del parámetro V_{as} se realizó una nueva medición de la frecuencia de resonancia colocando sobre el altavoz una masa conocida, obteniendo los siguientes resultados:

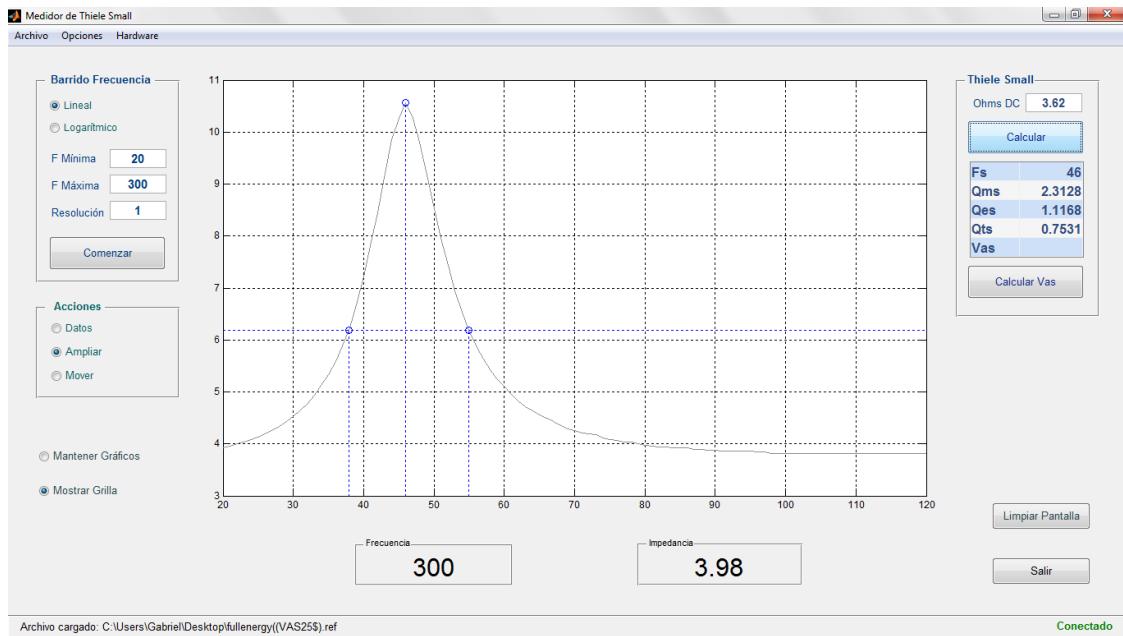


Figura 3.15: Impedancia del altavoz en campo libre con una masa conocida adherida al cono del mismo

La nueva frecuencia de resonancia es $f_s' = 46$ Hz, y la masa conocida es $m = 11.41$ g. La masa mecánica equivalente incluyendo la masa de radiación es

$$Mms = \frac{m}{\left(\frac{f_s}{f_s'}\right)^2 - 1} = \frac{11.41 \text{ g}}{\left(\frac{66.02 \text{ Hz}}{46 \text{ Hz}}\right)^2 - 1} = 10.76 \text{ g} = 0.0107 \text{ Kg}$$

El radio del pistón equivalente es igual al radio del cono (radio de la circunferencia equivalente a la base del cono, es decir a su proyección sobre un plano perpendicular a su eje) sumado a un tercio el radio de la suspensión

$$r = 5.56 \text{ cm} + \frac{1.00 \text{ cm}}{3} = 5.89 \text{ cm} = 0.0589 \text{ m}$$

Por lo que la superficie equivalente del pistón es

$$S = \pi r^2 = \pi x (5.89\text{cm})^2 = 0.0109 \text{ m}^2$$

Calculamos la masa acústica de suspensión como

$$Mas = \frac{Mms}{S^2} = 90.43 \frac{Ns^2}{m^5}$$

Y la compliancia acústica desuspensión

$$Cas = \frac{1}{(2\pi fs)^2 Mas} = 6.43 \times 10^{-8} \frac{m^5}{N}$$

Tenemos finalmente

$$Vas = Cas (\rho_0 c^2) = 6.44 \times 10^{-8} \frac{m^5}{N} \times (1.21 \frac{Kg}{m^3} \times (346.3 \frac{m}{s})^2)$$

- $V_{as} = 9.33 \times 10^{-3} \text{m}^3 = 9.33L$

Conociendo el valor de todos los parámetros TS del altavoz, es posible continuar con el diseño de los recintos utilizando el software apropiado.

3.5.1.3 Diseño de cajas acústicas

Se ingresa al software de diseño de cajas acústicas con los valores de los parámetros de Thiele Small del altavoz

- $f_s = 66,02 \text{ Hz}$
- $Q_{ms} = 2,5841$
- $Q_{es} = 1,0598$
- $Q_{ts} = 0,7516$
- $V_{as} = 9,33 \text{ L}$

Al ingresar dichos valores, el software recomienda la construcción de una caja tipo Bass-Reflex, cuyas dimensiones externas se muestran en la Figura 3.16 a(utilizando como material fenólico de 12mm)

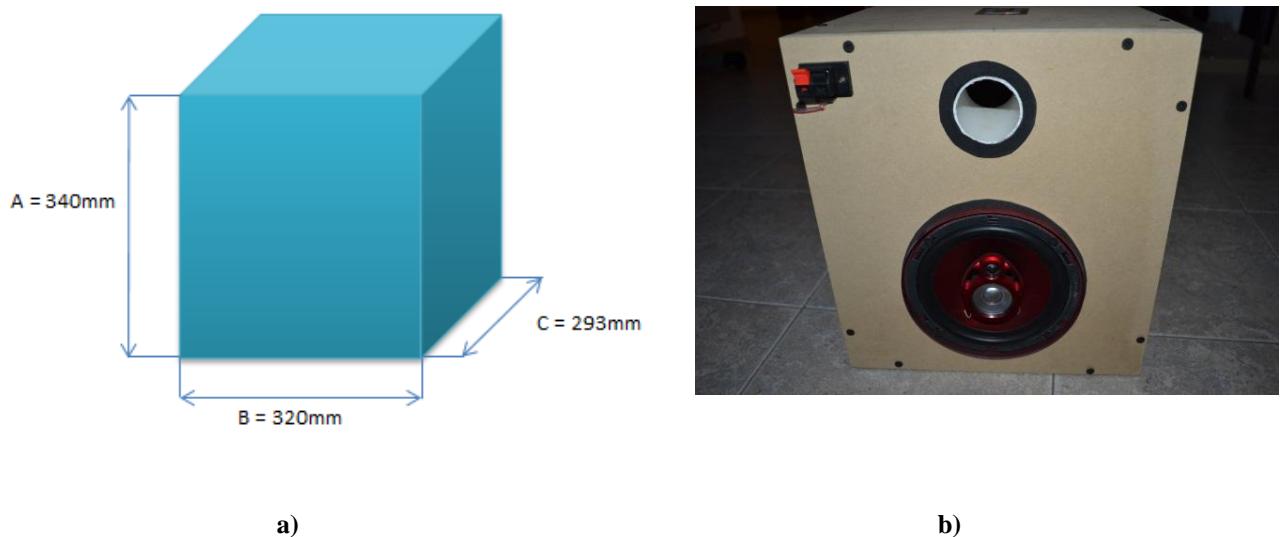


Figura 3.16: a) Esquema y dimensiones de los recintos. b) Fotografía de fuente secundaria construida

El tubo de sintonía será fabricado con tubo PVC de 50mm de diámetro, con las características indicadas en la Tabla 3.4. La caja será recubierta internamente con material absorbente (guata) para disminuir las reflexiones internas de la onda generada por la parte posterior del parlante, reduciendo así la formación de modos propios dentro del recinto que distorsionen la respuesta en frecuencia del sistema.

Tabla 3.4: Características del tubo de sintonía para los recintos ventilados

Parámetro	Valor
Número de ventilaciones	1
Forma de la sección transversal	Redonda (circular)
Fin de la ventilación	1 extremo al ras
Diámetro	50mm
Largo	85mm

Las dimensiones de las piezas para la construcción de la caja son las siguientes

Tabla 3.5: Dimensiones de las piezas de los recintos

Parte	Ancho [mm]	Largo [mm]	Espesor [mm]
Superior e inferior	293	320	12
Frontal y trasera	316	296	12
Lados	316	293	12

De acuerdo al software, la curva de respuesta en frecuencia normalizada del sistema es la siguiente (Figura 3.17)

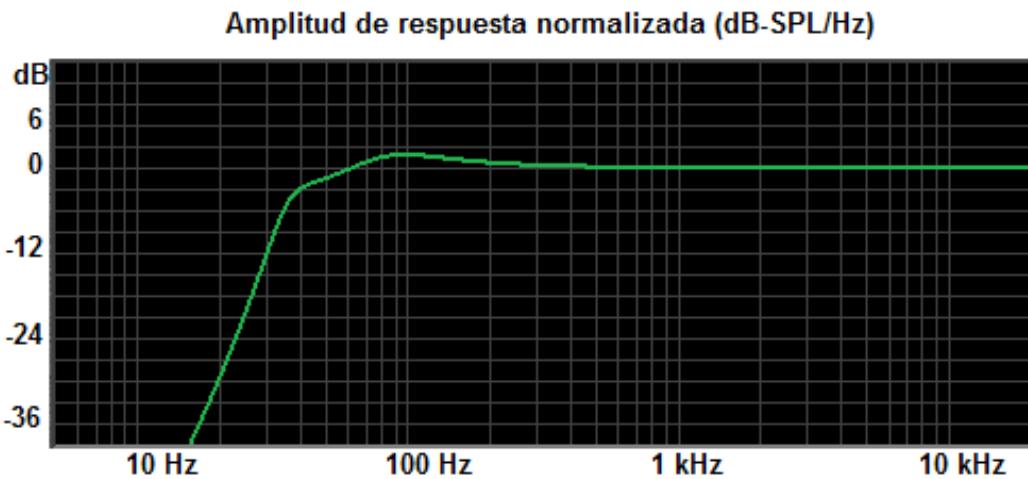


Figura 3.17: Respuesta en frecuencia normalizada de las fuentes secundarias, según el software utilizado.

3.5.2 Sensores

Para el sistema de control activo de ruido se optó por utilizar como sensores de error dos micrófonos de medición omnidireccionales marca Behringer modelo ECM8000, por presentar patrón de directividad omnidireccional, y respuesta en frecuencia plana en el rango de interés. A continuación, se presenta una breve descripción de los mismos

- Micrófono de medición del tipo capacitor prepolarizado (electret)
- Respuesta en frecuencia lineal
- Patrón de directividad omnidireccional
- Alimentación requerida de 15V a 48V
- Construcción rugosa y diseño moderno
- Protector de viento y soporte incluídos
- Impedancia: 600Ω
- Sensibilidad: -60dB
- Respuesta en frecuencia: 15Hz a 20kHz
- Peso: 120g aprox.

En la Figura 3.18 se presenta una fotografía del sensor montado en el recinto



Figura 3.18: Fotografía del micrófono de medición en el interior de la cabina

3.6 Amplificación de potencia

3.6.1 Amplificación para fuentes de control

Para la amplificación de potencia de las señales de control activo de ruido, se decidió utilizar un amplificador de audio de 4 canales de automóvil. El amplificador seleccionado fue el BOSS CE 404, debido a que ofrece su máximo rendimiento con altavoces de 4Ω , y es capaz de entregar $50 \text{ W}_{\text{RMS}}$ por canal, lo cual corresponde a la máxima potencia que son capaces de soportar los altavoces. A continuación, se presentan las especificaciones del mismo

- Potencia RMS, para una carga de 4 ohm, por canal: $50 \text{ W}_{\text{RMS}}$
- Número de canales: 4
- Respuesta en frecuencia: 12 a 20000 Hz
- Distorsión total armónica (THD), RMS: 0,05 %
- Realción señal ruido (SNR): 95 dB
- Impedancia mínima de carga por canal: 2Ω
- Dimensiones: 2.75" x 7.75" x 9.875"

En la figura 3.19 se muestra una fotografía del amplificador



Figura 3.19: Fotografía del amplificador de audiofrecuencias de cuatro canales

3.6.2 Amplificación para fuente de ruido

Para amplificar la señal de audio que representa el ruido del automóvil, se optó por implementar un amplificador con el circuito integrado TDA1562, el cual necesita una alimentación de 12 V_{DC} (al igual que el amplificador para las fuentes de control) y es capaz de entregar 70 W_{RMS} para una carga nominal de 4 Ω.

En la Figura 3.20 se presenta el diagrama esquemático del amplificador y el circuito impreso (PCB: Printed Circuit Board).

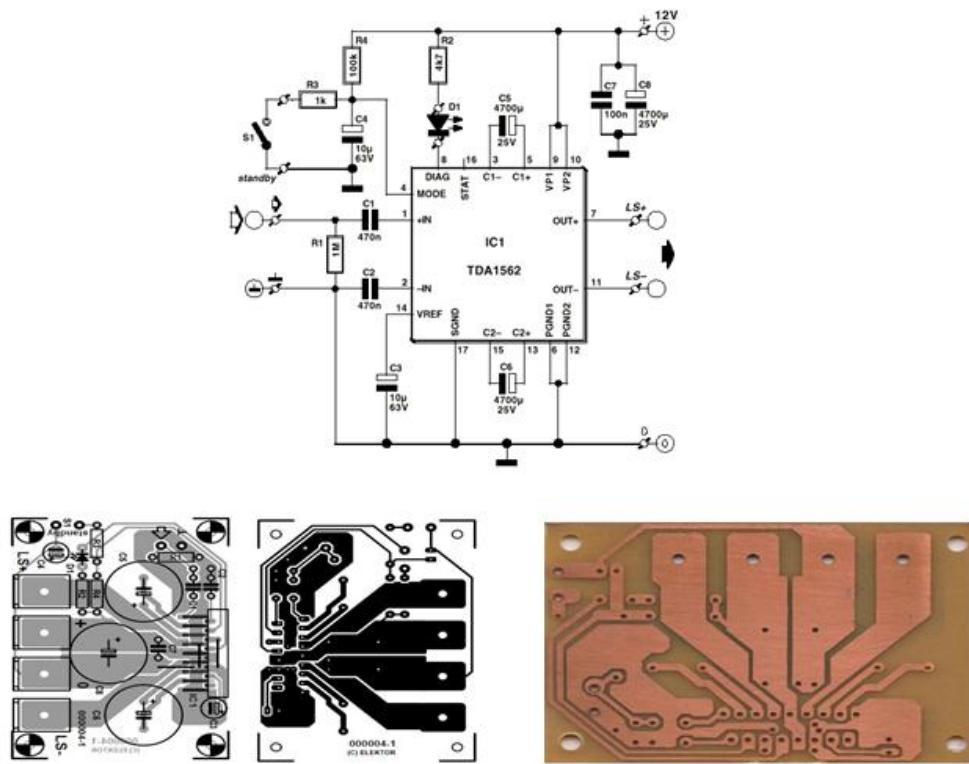


Figura 3.20: Diagrama esquemático y circuito impreso del amplificador para la fuente de ruido primario

3.6.3 Alimentación

Para alimentar ambos amplificadores se utilizó una fuente conmutada de computadora personal, la cual es capaz de entregar una intensidad de corriente de 8A para una tensión de 12V. A continuación, se presenta una fotografía del montaje completo



Figura 3.21: Amplificadores de potencia y fuente de alimentación

CAPÍTULO 4: Procesamiento digital de señales en tiempo real

En el presente capítulo se repasan los conceptos básicos del procesamiento digital de señales en tiempo real, centrándose en el procesamiento del tipo muestra por muestra utilizado en esta aplicación. Se incluyen también las cuestiones relacionadas con la longitud de palabra finita de los procesadores de señal digital, particularmente en el formato de coma fija.

4.1 Procesamiento digital de señales

El Procesamiento de señales trata de la representación, transformación y manipulación de señales. Cuando se aborda el procesado digital de señales se refiere a la representación mediante secuencias de números de precisión finita, y el procesado se realiza utilizando computadores digitales. A menudo es deseable que estos sistemas funcionen en tiempo real, lo que significa que el sistema en tiempo discreto se implementa de forma que las muestras de salida se calculan a la misma velocidad a la que se muestrea la señal en tiempo continuo. Son muchas las aplicaciones que requieren esta especificación [29].

La teoría de señales divide a las mismas en señales continuas y discretas. Las primeras se encuentran definidas sobre un rango continuo del tiempo, pudiendo tomar valores de amplitud también dentro de un rango continuo. Estas representan a la evolución temporal de algún tipo de magnitud censada como ser la perturbación que genera el sonido, las ondas electromagnéticas, o las señales eléctricas en general provenientes de sensores. Por otro lado, las señales discretas, a diferencia de las continuas, están definidas solamente para valores discretos de tiempo, comúnmente a instantes de tiempo uniformemente espaciados. Si una señal discreta presenta además valores discretos, la misma se denomina como señal digital. Las señales digitales solamente pueden tomar valores de amplitud discretos, ya que caso contrario sería imposible desde el punto de vista práctico poder almacenarlas o procesarlas en computadores, como habitualmente se hace [30]. Estas señales en tiempo discreto surgen si el sistema involucra la operación de muestreo de

señales en tiempo continuo. La señal muestreada es x_0, x_T, x_{2T}, \dots , donde T es el periodo de muestreo. Dicha secuencia de valores el que aparecen de la operación de muestreo normalmente se escribe como x_k . Si el sistema incluye un proceso iterativo realizado por una computadora digital, la señal involucrada es una secuencia de números x_0, x_1, x_2, \dots La secuencia de números normalmente se escribe como $\{x_k\}$ en donde k , entero asociado al elemento, es el índice del mismo [29].

El procesamiento digital de señal o DSP (por sus siglas en inglés Digital Signal Processing) es una disciplina que trata sobre las señales digitales y el uso de sistemas encargados de analizar, modificar, almacenar, o extraer información de este tipo de señales [56]. Para poder implementar los distintos algoritmos de la teoría de procesamiento digital de señal, es necesario utilizar un procesador digital de señal o DSP (por sus siglas en inglés Digital Signal Processor). Un DSP es un procesador especialmente diseñado y optimizado para realizar de manera rápida y eficaz las operaciones matemáticas que requieren estos algoritmos. Dichas operaciones son principalmente multiplicaciones y acumulaciones, las cuales son realizadas dentro de un DSP en un ciclo de instrucción [1].

Los elementos básicos de todo sistema que realiza un procesamiento digital de señal, son [31]:

- DSP: Encargado del procesamiento digital propiamente dicho.
- Conversor A/D: Realiza la conversión de la señal analógica de entrada, la cual proviene de un sensor (termocupla, micrófono, etc.), a una señal digital que pueda ser procesada por el DSP.
- Conversor D/A: Realiza la conversión de la señal digital de salida del DSP a una señal analógica que pueda luego aplicarse a un actuador (motor, parlante, etc.).
- Filtro antialiasing: Filtro pasa bajos que precede al conversor A/D para evitar el aliasing o solapamiento de los espectros generados en el momento de realizarse el muestreo de la señal.
- Filtro de reconstrucción: Filtro pasa bajos que sigue después del conversor D/A para reconstruir la señal analógica.
- Amplificadores: Un amplificador es necesario para levantar el nivel de la señal analógica de entrada, al necesitado por el conversor A/D. También

puede ser necesario usar un amplificador para acondicionar la señal analógica de salida a la requerida por los actuadores.

En algunas aplicaciones puede que esté presente solamente el DSP, prescindiendo de las conversiones A/D y D/A. Esto es debido a que no se necesita realizar ninguna conversión A/D porque las señales ya se encuentran en la forma digital, ni tampoco realizar la conversión D/A porque solamente se las necesita almacenar en memoria [1]. Las ventajas que tienen los sistemas con procesamiento digital frente al procesamiento analógico, son la flexibilidad, la capacidad de poder realizar procesamientos complejos en tiempo real, la invariancia de las características del sistema en el tiempo, el almacenamiento en memoria de los datos digitales (portabilidad), sistemas de tamaño físico más reducido y más económicos [1].

4.1.1 Procesamiento en tiempo real

Un sistema digital que realiza un procesamiento en tiempo real es un sistema que procesa los datos de entrada a una velocidad regular, cumpliendo requerimientos en su tiempo de respuesta [32]. El tiempo de respuesta del sistema digital es definido como el tiempo que hay entre el arribo de la muestra (o muestras) de entrada y la salida de la muestra (o muestras) procesada [32]. Cuando un procesamiento analógico es reemplazado por un procesamiento digital, este debe realizarse en tiempo real. La señal analógica es muestreada a la frecuencia de muestreo f_s , marcando la velocidad a la que los datos ingresan al sistema digital. La misma velocidad también marcará el ritmo de los datos de salida, para poder realizarse la conversión de estos en una señal analógica. Este proceso impone un requerimiento en el tiempo de respuesta del sistema, el cual dependerá de si se realiza un procesamiento muestra por muestra o un procesamiento por bloque de muestras [30][32].

En un procesamiento digital en tiempo real de muestra por muestra, las muestras de la señal analógica que ingresan a una velocidad determinada por la frecuencia de muestreo f_s , son procesadas una por vez para obtener las correspondientes muestras de

salida. Para que este procesamiento se lleve a cabo correctamente, el tiempo de respuesta t_r del sistema debe ser menor al período de muestreo $T[1]$.

El tiempo de respuesta t_r se compone del tiempo de procesamiento t_p de la muestra y el tiempo de demora t_o que hay involucrado en las operaciones de entrada y salida de cada muestra. Este tiempo t_o es el tiempo que se tarda en traer la muestra de entrada desde el conversor A/D a la memoria, más el tiempo que demora en llevar la muestra de salida desde memoria al conversor D/A.

En adición, la frecuencia de muestreo de un sistema digital determina el ancho de banda del mismo. Esto viene de la aplicación del teorema de muestreo, por lo que la máxima frecuencia f_M de la señal de entrada que debe ser procesada por el sistema es la mitad de la frecuencia de muestreo.

Disminuyendo el tiempo de procesamiento t_p y/o el tiempo de demora t_o permite que se pueda aumentar el ancho de banda del sistema. El tiempo de procesamiento t_p dependerá de la velocidad del DSP y de la eficiencia del algoritmo. Una manera de mejorar el tiempo de demora t_o es realizar un procesamiento de bloque [30][32], sin embargo, este tipo de procesamiento tiene la desventaja de introducir un retardo al sistema que dependerá del tamaño del bloque. Un sistema digital que realiza un procesamiento de muestra por muestra tiene un retardo igual al período de muestreo T , el cual es el mínimo retardo que puede introducir cualquier sistema digital que realice un procesamiento en tiempo real.

4.1.2 Formato de datos digitales

Un factor fundamental al momento de elegir un DSP para una aplicación determinada es el formato de sus datos digitales, o lo que es lo mismo el tipo de aritmética fraccional con la que opera. La aritmética es fraccional debido a que en las aplicaciones de Procesamiento de Señal Digital encontramos que la mayoría de los datos son del tipo fraccional, como por ejemplo los coeficientes de un filtro [1].

Existen dos tipos de formatos en los procesadores DSP: coma fija o fixed-point (por su traducción al inglés) y coma flotante o floating-point (por su traducción al inglés). Los DSP de coma flotante tienen una precisión y un rango dinámico superior a los de coma fija. Sin embargo, los DSP de coma fija son más rápidos, más baratos y consumen menos

potencia que los DSP de coma flotante. La mayoría de las aplicaciones embebidas de bajo costo y de alto volumen, tales como dispositivos de control, teléfonos celulares, drivers de discos duros, módems, reproductores de música y cámaras digitales, usan procesadores de coma fija [30]. Los DSP de coma flotante permiten el uso eficiente de compiladores en lenguaje C, reduciendo el costo de desarrollo y mantenimiento [30].

4.1.2.1 Coma fija

Los DSP de coma fija son usualmente dispositivos de 16 bits o 24 bits. La Figura 4.1 muestra uno de los posibles formatos de coma fija para una longitud de palabra de 16 bits, mostrando los respectivos pesos de cada bit.

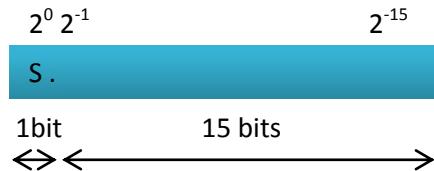


Figura 4.1: Formato digital de coma fija (16 bits)

En la Figura 4.1 se observa el bit de signo y un punto haciendo referencia a la ubicación de la coma del dato fraccional. El formato soporta números negativos en complemento a dos. La precisión del dato para 16 bits es 2^{-15} y el rango dinámico es de:

$$-1 \leq \text{Valor fraccional} \leq (1 - 2^{-15}) \quad (4.1)$$

La relación entre el valor fraccionario y el valor entero equivalente de un mismo dato digital, está dada por la siguiente ecuación para datos de coma fija de 16 bits:

$$\text{Valor fraccional} = \frac{\text{Valor entero}}{2^{15}} \quad (4.2)$$

4.1.2.2 Coma flotante

Los DSP de coma flotante son usualmente dispositivos de 32 bits (precisión simple). Existe también el formato de coma flotante de 64 bits (doble precisión). La Figura 4.2 muestra uno de los posibles formatos de coma flotante para una longitud de palabra de 32 bits, distinguiendo sus distintas partes.

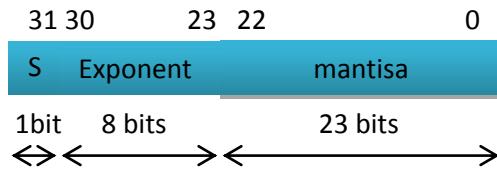


Figura 4.2: formato digital de coma flotante (32 bits)

La mantisa junto con el bit de signo forman un dato fraccional semejante al de coma fija, mientras que el exponente es un entero que se aplica a una potencia de 2 para obtener la ubicación de la coma. La siguiente ecuación explica cómo se obtiene el valor fraccional resultante (formato IEEE 754):

$$\text{Valor fraccional} = \text{mantisa con signo } 2^{(\text{exponente} - 127)} \quad (4.3)$$

La precisión para el formato de coma flotante de 32 bits es de 2^{-23} y el rango dinámico es de:

$$-5,88 \cdot 10^{-39} \leq \text{Valor fraccional} \leq 3,4 \cdot 10^{38} \quad (4.4)$$

4.1.3 Relación señal a ruido de cuantificación o SQNR

Cuando una señal analógica es convertida en una señal digital, primero debe ser muestreada para luego ser cuantificada y representada por medio de un dato digital. El proceso de cuantificación introduce un error, el cual se comporta como un ruido aditivo en

la señal resultante [1]. Este error depende del paso de cuantificación empleado por el conversor A/D, el cual es igual a:

$$\Delta = \frac{R}{2^B} \quad (4.5)$$

Siendo Δ el paso de cuantificación, R el rango máximo de la señal a cuantificar y B la cantidad de bits del conversor A/D. La relación de señal a ruido de cuantificación o SQNR (por sus siglas en inglés Signal to Quantization Noise Ratio) [30][31] es la relación entre la potencia de la señal de entrada $x(n)$ y la potencia del ruido de cuantificación $e(n)$. En decibeles es:

$$SQNR[dB] = 10\log \frac{P_x}{P_e} \quad (4.6)$$

Se demuestra [30][31] que el ruido de cuantificación tiene media igual a cero y variancia (potencia) igual a:

$$Pe = \frac{\Delta^2}{12} \quad (4.7)$$

Reemplazando las ecuaciones (4.5) y (4.6) en (4.7), y reacomodando se llega a:

$$SQNR[dB] = 10,79 + 6,02B + 10\log P_x - 20\log R \quad (4.8)$$

La ecuación (4.8) muestra que por cada incremento de un bit en el conversor A/D la SQNR aumenta 6 dB.

4.2 Efectos de la precisión finita en el ANC

En la práctica, al realizar un procesamiento de señal digital, la precisión de los datos no es infinita, debido a que estos deben tener una longitud finita para que puedan ser

almacenados y procesados por un DSP. Debido a esta precisión finita en los datos digitales, existen los siguientes errores [2]:

- Errores de cuantificación: Producidos por la cuantificación de las señales de entrada y de los coeficientes.
- Errores aritméticos: Producidos en las operaciones matemáticas de suma y multiplicación. Se los conoce como desbordamiento o overflow (por su traducción en inglés) y ruido de redondeo (o truncamiento).

En los DSP de coma fija, estos errores son importantes y deben tenerse en cuenta, debido a que los mismos tienen una precisión y un rango dinámico mucho menor que los DSP de coma flotante [1]. Al realizarse operaciones de multiplicación, se producen redondeos en los resultados, introduciéndose un error. Para evitar los redondeos intermedios como puede ser en el caso del procesamiento de un filtro digital, los DSP tienen un multiplicador con acumulador de doble precisión. De esta manera, si se desea almacenar datos de menor precisión, solamente se introduce un error de redondeo importante en el momento en que se necesita sacar el resultado fuera del acumulador para transferirlo a la memoria. En la sección anterior se trató sobre los errores de cuantificación de las señales de entrada analógicas, producidos por los conversores A/D. Al igual que las señales de entrada, los coeficientes de un filtro digital adaptativo empleando el algoritmo LMS de la Ecuación (2.14), también deben ser cuantificados, introduciéndose un error en los mismos. Insuficiente precisión en dichos coeficientes, causará problemas tales como un desplazamiento del filtro Wiener en la solución obtenida (sección 2.5.2), o que el proceso de adaptación se detenga prematuramente. En ambos casos no se llega a converger a la solución óptima, produciéndose un error en la salida del filtro [2]. Dicha precisión alta en los coeficientes del filtro adaptativo, es necesaria para el proceso de actualización de los mismos empleando el algoritmo LMS, pero no para la realización de las multiplicaciones del proceso de filtrado FIR [37]. A la hora de realizar el proceso de filtrado se utilizan los bits más significativos de los coeficientes, de tal manera de evitar multiplicaciones con factores de precisión alta [37].

En el caso de realizarse operaciones de suma con datos en formato de coma fija, como los mismos tienen un rango dinámico de aproximadamente ± 1 , se debe cuidar de que

el resultado de esa suma no supere el valor 1, de manera de evitar un overflow de la unidad aritmética del DSP. Para prevenir esto, la técnica más usada es el escalado de los datos. La misma consiste básicamente en multiplicar las señales por un factor menor a 1, esto es atenuándolas. Debido a que al atenuarlas disminuye la relación señal a ruido y causa en el caso de filtros digitales adaptativos que el proceso de adaptación se detenga prematuramente, se desea mantener las señales lo más grande que se pueda sin overflow [2]. El escalado en los coeficientes de un filtro adaptativo y por consiguiente en la salida del mismo, se puede conseguir escalando la señal deseada d del filtro de la Figura 2.5 [2]. Para el caso de los sistemas ANC, en los cuales no se puede acceder a la señal deseada para escalarla, el mismo efecto de escalar d se consigue al escalar la señal de error del sistema [2]. Luego para compensar la atenuación introducida, se deberá multiplicar la salida del filtro de control por la inversa del factor de escala empleado en la señal de error [1].

En el proceso de adaptación de un filtro digital empleando el algoritmo LMS, el término $\mu e(n)x(n - k)$ es el término de corrección o de actualización para los coeficientes del filtro. A medida que los coeficientes comienzan a converger a la solución óptima, el valor del error e empieza a disminuir. Esto produce que también el término de corrección disminuya, a tal punto de que el valor de dicho término sea menor a la precisión del dato digital. El efecto de esto hace que el proceso de adaptación se detiene prematuramente [1]. La condición para que se detenga la adaptación es:

$$|\mu e(n)x(n - k)| < 2^{-b} \quad (4.9)$$

Siendo b la cantidad de bits de la parte fraccional del dato digital en formato de coma fija. Para el caso de la Figura 4.1 es igual a 15 bits. Una aproximación de la condición (4.9) se consigue al reemplazar $e(n)$ y $x(n - k)$ por sus valores eficaces σ_e y σ_x respectivamente [2]. Por lo tanto, para prevenir que el proceso de adaptación se detenga prematuramente antes que el error alcance el nivel σ_e , el valor de μ debe ser:

$$\mu \geq \mu_{min} = \frac{2^{-b}}{\sigma_x \sigma_e} \quad (4.10)$$

La inecuación (4.10) también puede plantearse de la siguiente forma

$$\sigma_e < \mu_{min} = \frac{2^{-b}}{\sigma_x \mu} \quad (4.11)$$

El lado derecho de (4.11) se denomina error residual digital, y el algoritmo se detiene cuando ese nivel es alcanzado. De (4.11) se observa que para evitar la detención prematura del proceso adaptativo del algoritmo LMS debida a la precisión finita, se puede aumentar dicha precisión, o incrementar también el valor de μ . Esto último indica que hay una cuestión de compromiso entre el exceso del MSE y el error en los coeficientes del filtro de control debido a la detención prematura del algoritmo por precisión finita [1].

CAPÍTULO 5: Simulaciones y optimización

En el presente capítulo se profundiza sobre las simulaciones computacionales desarrolladas con el fin de verificar el funcionamiento del algoritmo seleccionado para la aplicación, y se da una primera aproximación de los resultados a obtener utilizando diferentes señales de ruido primario. A su vez, se detalla la optimización del sistema que permitió reducir el poder de cómputo necesario para el procesador al momento de implementar el sistema.

5.1 Simulación Monocanal

Se realizó la simulación del sistema de control activo de ruido en una computadora personal, en el entorno Matlab. La simulación se realizó con un procesamiento del tipo muestra por muestra (ya que de esta forma será implementado en el DSP) con aritmética de coma flotante de doble precisión, razón por la cual los errores de precisión finita son despreciables. El principal objetivo de las simulaciones fue obtener una primera aproximación de la situación propuesta, logrando estimar el largo de los filtros de control y de aprendizaje del camino secundario que deberán implementarse en el DSP, independizándose de los errores debido a la precisión finita. En primer lugar se decidió simular un sistema monocanal, es decir con una única fuente secundaria y un único sensor de error, debido a que ello permitió determinar el comportamiento ANC en la mejor situación posible (debido a la simplicidad del sistema).

5.1.1 Sistema de control

En la Figura 5.1 se presenta un esquema del sistema prealimentado con el algoritmo MFxLMS, con modelado online del camino secundario. Las aclaraciones en rojo representan las distintas señales en el sistema.

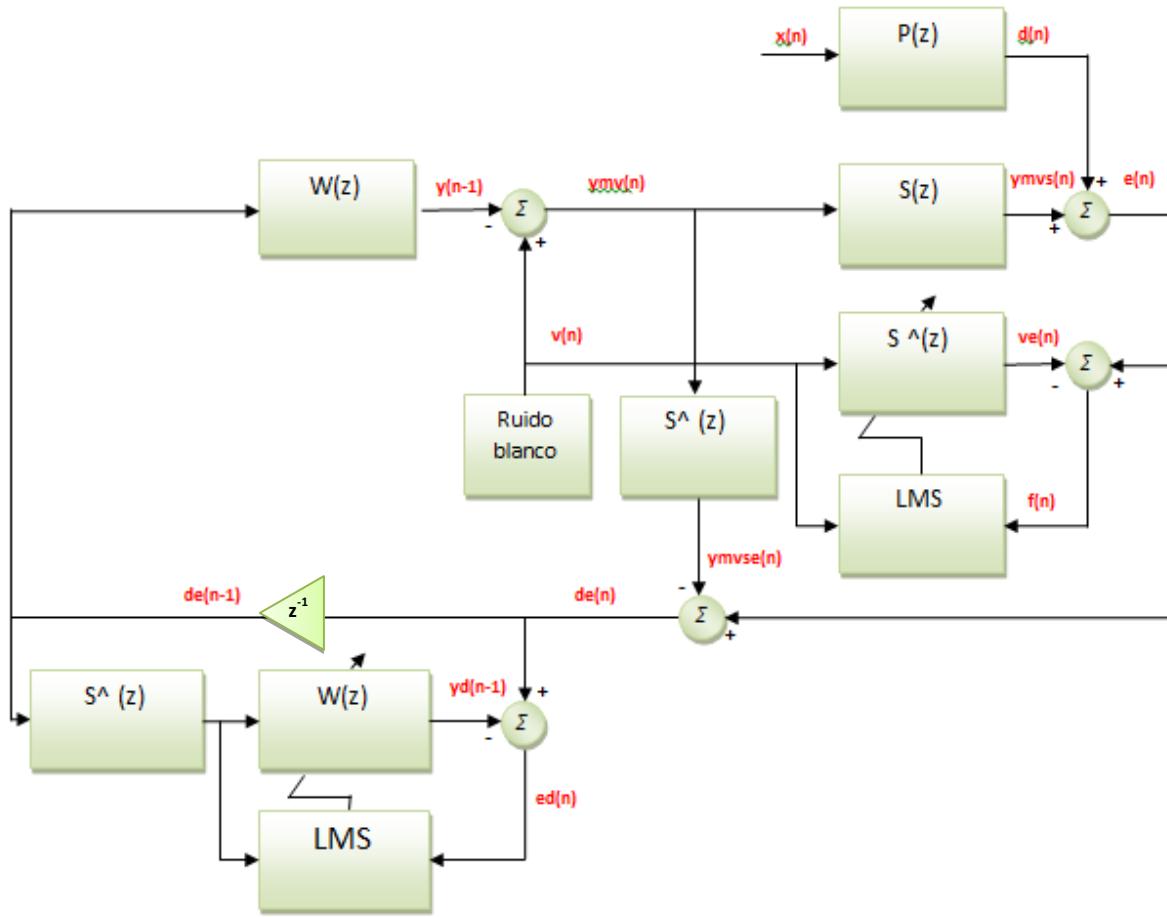


Figura 5.1: Diagrama en bloques de sistema ANC con algoritmo MFxLMS

5.1.2 Camino secundario

Se realizó la medición de la respuesta impulsiva del camino secundario en el modelo construido para el canal 3 (ver figura 5.17), y se utilizó la misma como camino secundario en la simulación. Se tuvieron en cuenta sólo los primeros 100ms (800 coeficientes) de la respuesta, debido a que en este tiempo se concentra la mayor parte de su energía. La medición se realizó con un software específico para la aplicación aplicando el método de deconvolución de señales determinísticas, usando como señal de entrada secuencias MLS de 23,8 s de duración.

A continuación, se presenta dicha respuesta (Figura 5.2).

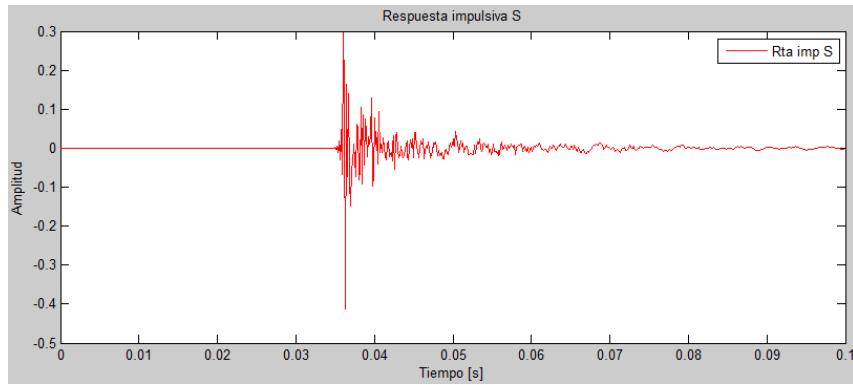


Figura 5.2: Respuesta impulsiva del camino secundario

La demora de aproximadamente 37ms que se observa al comienzo de la medición hasta la llegada del sonido directo, corresponde a los retardos introducidos por cada una de las etapas en la cadena de medición utilizada. Este retardo estará presente en cada medición de camino primario y camino secundario realizadas en esta sección.

5.1.3 Camino primario

Se midió la respuesta impulsiva del camino primario para el canal 3 del modelo, y se utilizó la misma como camino primario tomando únicamente los primeros 2 segundos. A continuación, se presenta una imagen de dicha respuesta impulsiva

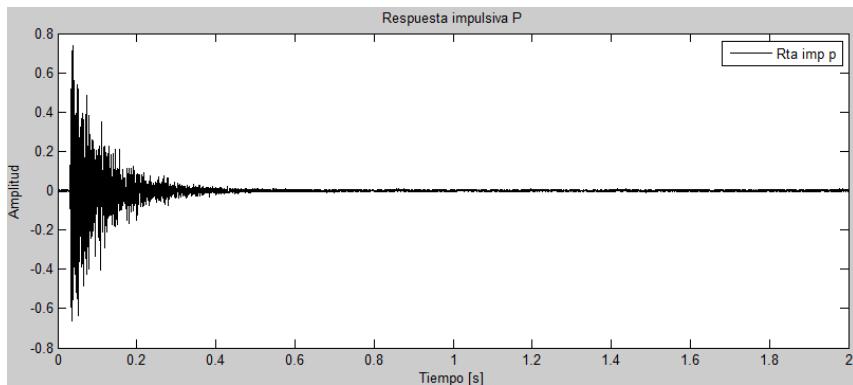


Figura 5.3: Respuesta impulsiva del camino primario

5.1.4 Señal de ruido

En la implementación del sistema se optó por trabajar con tres señales de ruido d distintas (Figura 5.1), a modo de partir de una condición ideal de operación, e ir aumentando gradualmente la complejidad hasta trabajar en una condición real de operación.

5.1.4.1 Señal número 1

La primera señal de ruido utilizada fue una señal senoidal de frecuencia 100Hz, debido a que la simplicidad de dicha señal permitió obtener altos niveles de atenuación, y apreciar el efecto de la variación de los distintos parámetros del sistema en los resultados.

5.1.4.2 Señal número 2

La segunda señal utilizada fue una señal senoidal de frecuencia 100Hz sumada a ruido blanco de media cero y varianza 0.01.

5.1.4.3 Señal número 3

La tercera señal utilizada fue una señal de un motor de combustión creada en Matlab, para lo cual se realizó un estudio previo de las distintas componentes que contribuyen al ruido en el interior de un automóvil.

El campo sonoro en el interior de un vehículo presenta una alta complejidad debido a tres razones principales. En primer lugar, se presenta el hecho de que existen numerosas fuentes sonoras distribuidas a lo largo del vehículo. En segundo lugar, muchas de estas fuentes tienen componentes complejas y no estacionarias, compuestas por una mezcla de armónicos, señales aleatorias y elementos transitorios. Finalmente, el funcionamiento de un automóvil consiste en un conjunto de condiciones distintas, desde encender el motor hasta la aceleración de las marchas [38]. Debido a la complejidad del

campo sonoro en estudio, es necesario realizar ciertas simplificaciones a la hora de implementar el control activo de ruido.

Es posible distinguir tres componentes fundamentales en el ruido de un vehículo: el ruido del motor (incluyendo admisión y escape), el ruido del viento, y el ruido de los neumáticos.

El giro del motor y el movimiento de los pistones generan un espectro de ruido armónico, cuyas principales componentes espectrales de banda angosta están directamente relacionadas a las revoluciones por minuto (rpm) del motor.

Para los autos modernos, el ruido causado por el viento tiene una influencia menor que el ruido causado por el motor y los neumáticos. El mismo toma mayor importancia solo en muy altas velocidades.

En cuanto al ruido causado por los neumáticos, puede decirse que la densidad espectral de potencia del mismo presenta una alta correlación con la velocidad del vehículo [4].

Para esta aplicación de control activo de ruido se tomará en cuenta sólo el ruido producido por el motor del automóvil. La frecuencia fundamental del sonido generado por un motor de 4 tiempos y n cilindros, que opera a una velocidad de N rpm está dada por la siguiente ecuación [33]

$$f_0 = \frac{N \times n}{2 \times 60} \text{ [Hz]} \quad (5.1)$$

A continuación, se presentan dos ejemplos del análisis espectral del nivel de presión sonora generado por motores de combustión. En el primer caso se trata de un motor de 4 tiempos, 4 cilindros, operando a una velocidad de 7650 rpm. En el segundo caso se presenta el ruido generado por un motor de 4 tiempos, 6 cilindros, corriendo a 1800 rpm. En ambas figuras puede observarse el importante contenido armónico del ruido generado.

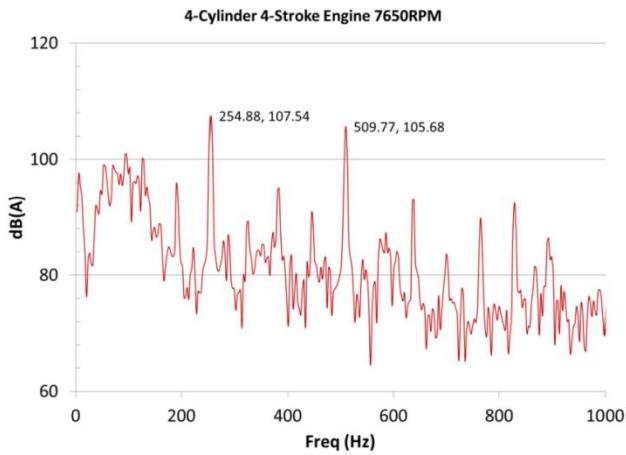


Figura 5.4: Sonido generado por un motor de combustión de 4 tiempos, 4 cilindros a 7650 rpm. Imagen tomada de [39]

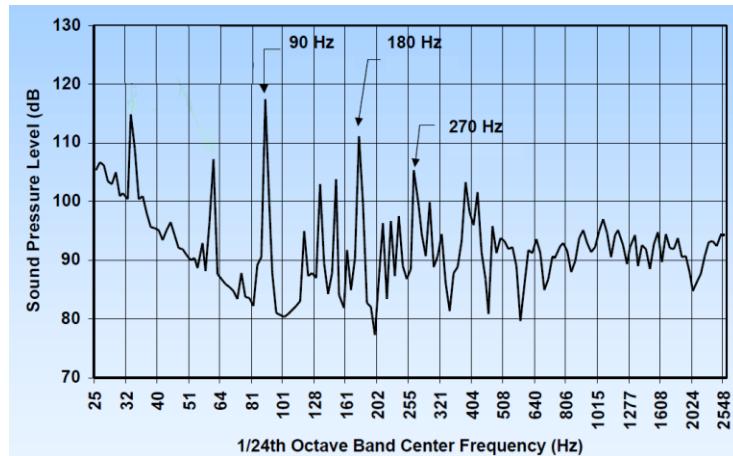


Figura 5.5: Sonido generado por un motor de combustión de 4 tiempos, 6 cilindros a 1800 rpm. Imagen tomada de [40]

Para nuestra aplicación se busca generar el ruido de un motor de 4 tiempos, 4 cilindros, operando a 2500 rpm. Mediante la ecuación (5.1) se calcula la frecuencia fundamental del ruido generado por el motor

$$f_0 = \frac{N \times n}{2 \times 60} = \frac{2500 \text{ rpm} \times 4}{2 \times 60} = 83 \text{ Hz}$$

Luego, se utilizó la función proporcionada por Matlab para simular el ruido de un motor, utilizando en este caso como componente principal la frecuencia calculada anteriormente. Se modificó la amplitud de los distintos armónicos, y la amplitud del ruido blanco incorporado en dicha señal, de modo que el ruido generado se aproxime más al de un motor de combustión real. Finalmente, se convolucionó dicha señal con la respuesta impulsiva del camino primario P_1 medido en el modelo (Figura 5.17). A continuación, se presenta en la Figura 5.6 el análisis espectral de dicha señal

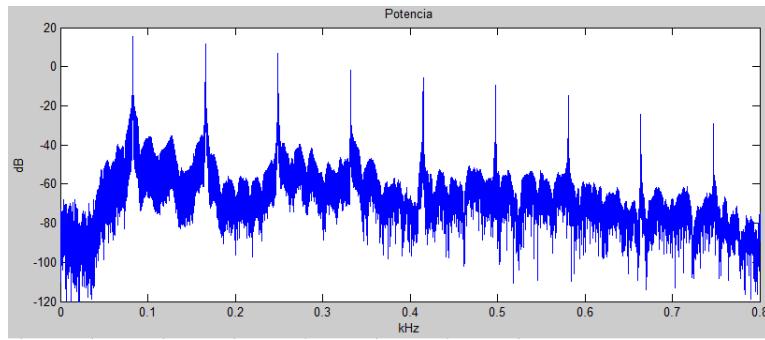


Figura 5.6: Análisis espectral de la señal de ruido primario generada

5.1.5 Condiciones de simulación

La frecuencia de muestreo utilizada fue 8kHz, y se simuló la operación del sistema durante un tiempo fijo de 20s para las dos primeras señales de ruido, y 60s para la tercera señal.

Al iniciar el funcionamiento del sistema, en primer lugar se realiza un aprendizaje del camino secundario S , utilizando como señal de entrada ruido blanco de media 0 y varianza 1. Dicho proceso de aprendizaje finalizó en el momento en que el valor medio cuadrático del error f (Figura 5.1) se estabilizó, es decir que no presentó variaciones mayores a un porcentaje determinado. El camino secundario fue implementado con un filtro FIR de 800 coeficientes, es decir que se tomaron los primeros 100ms del mismo, y el filtro utilizado para estimar el camino secundario fue un filtro FIR de 800 coeficientes. En la Figura 5.7 se presenta el camino secundario implementado S , y el camino secundario estimado S_{ev} que se obtuvo al finalizar el proceso de adaptación del mismo

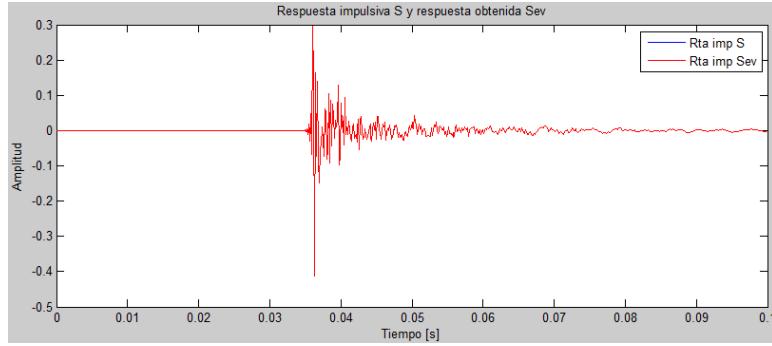


Figura 5.7: Camino secundario S, y camino secundario estimado Sev.

Para poder apreciar la diferencia en ambos es necesario ampliar una sección. En la Figura 5.8 se muestra la sección inicial de la respuesta que incluye el sonido directo, ampliada.

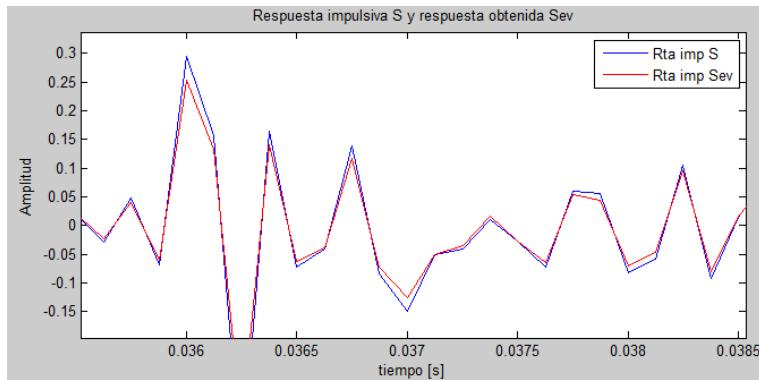


Figura 5.8: Ampliación de la Figura 5.7.

5.1.6 Resultados

A continuación, se presentan los resultados de las simulaciones para las cuatro señales de entrada. En cada caso, tanto el paso de adaptación μ , como el largo del filtro W (L_w), fueron determinados de modo de obtener las mayores atenuaciones posibles con el filtro W más corto posible.

5.1.6.1 Señal número 1

En la tabla 5.1 se presenta la configuración de los parámetros utilizada y la atenuación obtenida, y en las figuras 5.8, 5.9 y 5.10 puede apreciarse el desempeño del sistema.

Tabla 5.1: Configuración de parámetros y atenuación obtenida.

L_w	MSE final	Atenuación 100Hz
40	2 e-25	309 dB

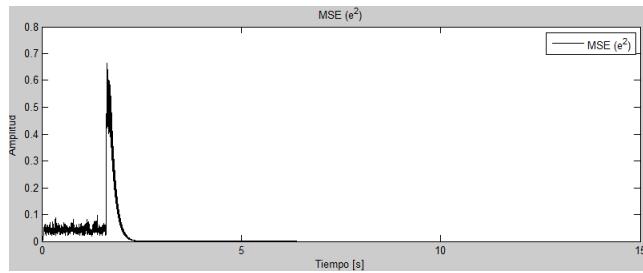


Figura 5.8: Evolución temporal del error medio cuadrático

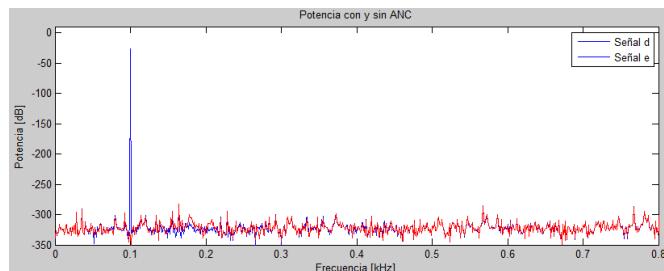


Figura 5.9: Análisis espectral de la señal de error con el ANC encendido y apagado

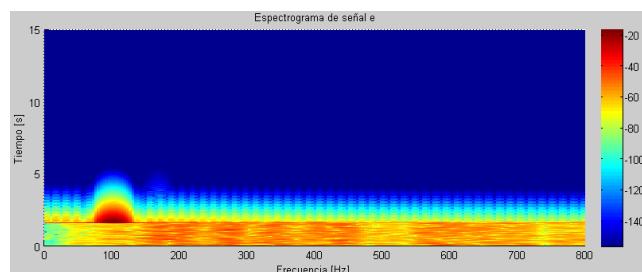


Figura 5.10: Evolución temporal de la PSD de la señal de error

5.1.6.2 Señal número 2

En la tabla 5.2 se presenta la configuración de los parámetros utilizada para la señal número 2 y la atenuación obtenida, y en las figuras 5.11, 5.12 y 5.13 puede apreciarse el desempeño del sistema.

Tabla 5.2: Configuración de parámetros y atenuación obtenida.

L_w	MSE final	Atenuación 100Hz
50	1 e-5	92 dB

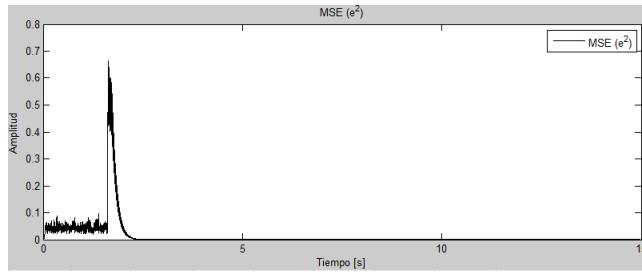


Figura 5.11: Evolución temporal del error medio cuadrático

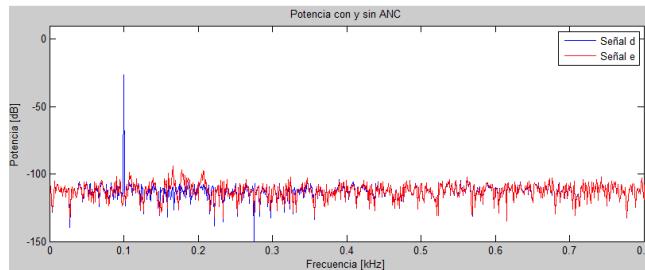


Figura 5.12: Análisis espectral de la señal de error con el ANC encendido y apagado

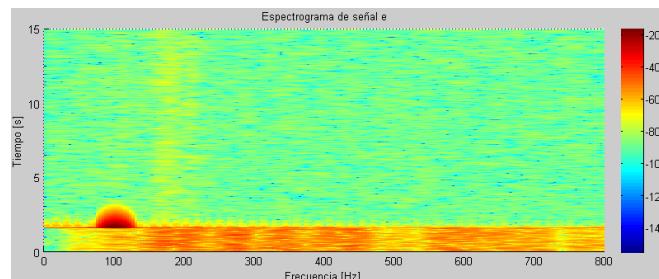


Figura 5.13: Evolución temporal de la PSD de la señal de error

5.1.6.3 Señal número 3

En la tabla 5.3 se presenta la configuración de los parámetros utilizada para la señal número 3 y la atenuación obtenida, y en las figuras 5.14, 5.15 y 5.16 puede apreciarse el desempeño del sistema ANC.

Tabla 5.3: Configuración de parámetros y atenuación obtenida.

L_w	MSE final	Att 83Hz	Att 166Hz	Att 249Hz	Att 332Hz	Att 415Hz	Att 498Hz
300	0.5	88 dB	88 dB	69dB	62 dB	38 dB	7 dB

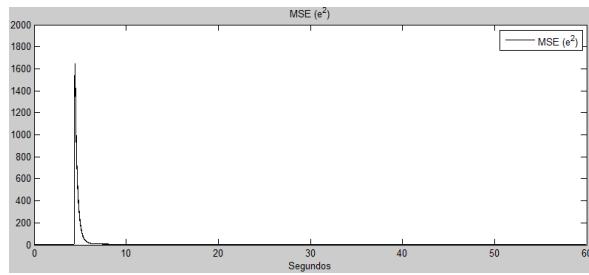


Figura 5.14: Evolución temporal del error medio cuadrático

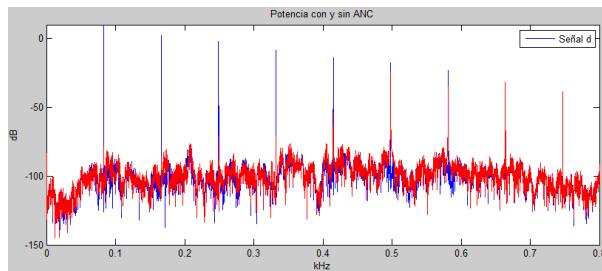


Figura 5.15: Análisis espectral de la señal de error con el ANC encendido y apagado

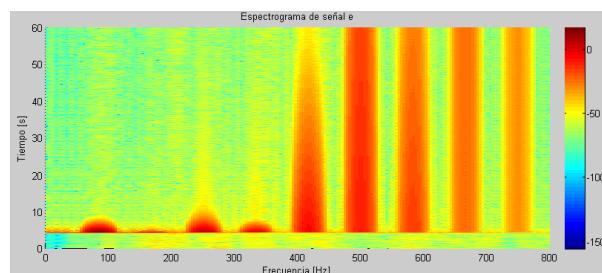


Figura 5.16: Evolución temporal de la PSD de la señal de error

Se observa en las figuras 5.15 y 5.16 que al aumentar la frecuencia las atenuaciones obtenidas por el sistema disminuyen. La señal de ruido sintetizada posee menor amplitud en los armónicos de mayor frecuencia, lo cual corresponde a los efectos de la aislación pasiva del recinto. En la Figura 5.6 se ve como la PSD de la armónica número 9 es 50 dB menor a la PSD de la primera armónica. Esta disminución de la potencia con la frecuencia, tiene como consecuencia que el sistema posea menor información acerca de la señal a cancelar en esa franja del espectro para la precisión y paso de adaptación con los que opera, razón por la cual es incapaz de obtener atenuaciones considerables.

5.1.6.4 Variación del retardo en el camino secundario

Al observar la respuesta al impulso del camino secundario S_3 medido, se vio que la misma presenta un retardo de aproximadamente 280 muestras, debido a las distintas etapas que introducen retardos en la cadena de medición. Esto hizo necesario que el filtro que estima el camino secundario tenga un número alto de coeficientes, lo cual implica que cada vez que se ejecuta el algoritmo MFxLMS deben actualizarse un gran número de coeficientes. A modo de análisis, se realizó una nueva simulación exactamente en las mismas condiciones que la anterior, excepto por el hecho de que se utilizó como camino secundario la respuesta medida quitándole el retardo. Esta modificación permitió utilizar un filtro de menor longitud para estimar el camino secundario, y de esta forma el tiempo de aprendizaje del camino secundario se redujo considerablemente. Sin embargo, los resultados finales respecto a los valores de atenuación obtenidos para las distintas señales de entrada no difieren significativamente de aquellos obtenidos en la simulación anterior, donde se utilizó el camino secundario original. Esto es lógico pues la parte quitada no posee energía real, sino que es un efecto del sistema de medición.

5.2 Simulación multicanal 4x2

Concluidas las simulaciones monocanal, se procedió a simular el ANC operando con múltiples sensores de error y fuentes secundarias. En esta etapa se buscó obtener una

aproximación de las atenuaciones que se podrían lograr con el sistema, y poder convalidar el modelo propuesto para esta aplicación.

A continuación, se presenta el diagrama del modelo implementado para el sistema multicanal de control activo de ruido con K fuentes y M sensores, indicando las principales funciones de transferencia involucradas en el mismo y la nomenclatura utilizada en la implementación (Figura 5.17 y Tabla 5.4)

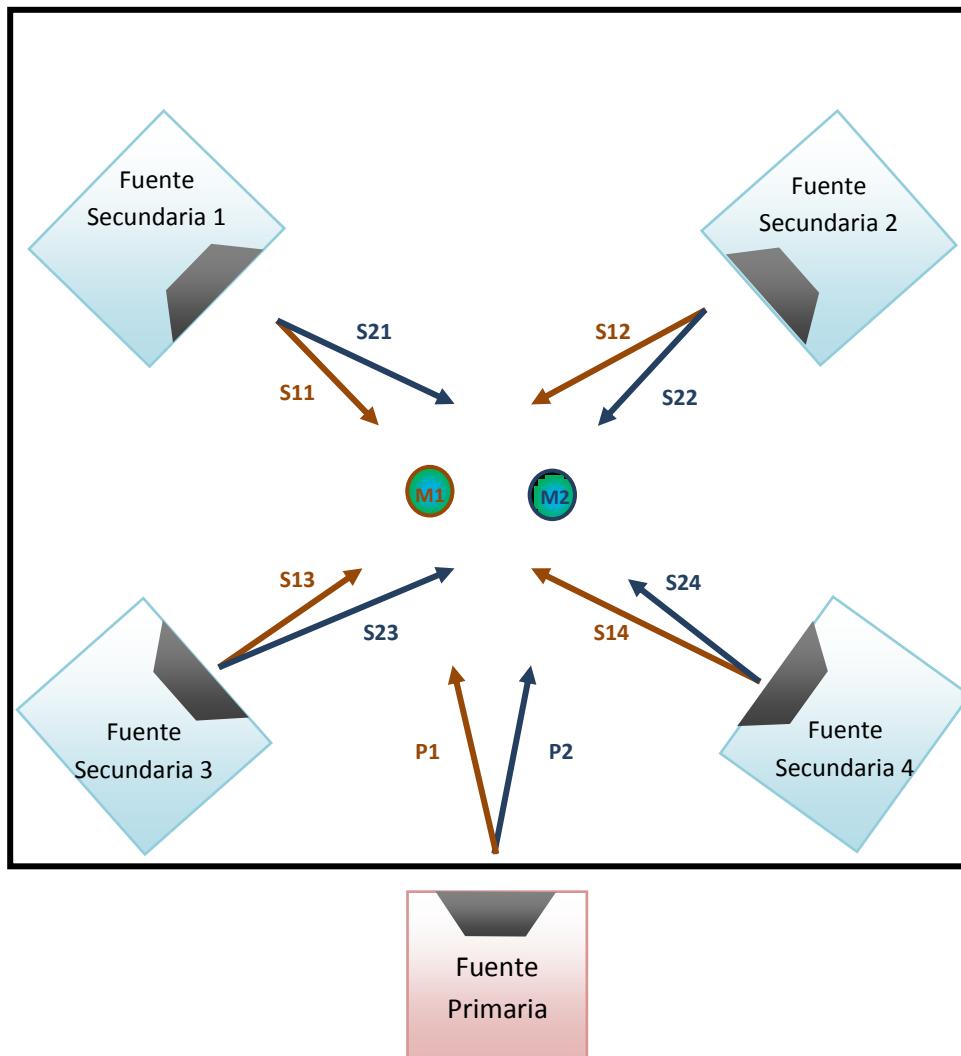


Figura 5.17: Esquema del sistema ANC 4x2

Tabla 5.4: Nomenclaturas utilizadas en la Figura 5.17

Caminos secundarios	Fuente	Sensor	Filtros W	Fuente	Referencia
$S_{11} \rightarrow S_1$	F_1	M_1	$W_{11} \rightarrow W_1$	F_1	x_1
$S_{12} \rightarrow S_2$	F_2	M_1	$W_{21} \rightarrow W_2$	F_2	x_1
$S_{13} \rightarrow S_3$	F_3	M_1	$W_{31} \rightarrow W_3$	F_3	x_1
$S_{14} \rightarrow S_4$	F_4	M_1	$W_{41} \rightarrow W_4$	F_4	x_1
$S_{21} \rightarrow S_5$	F_1	M_2	$W_{12} \rightarrow W_5$	F_1	x_2
$S_{22} \rightarrow S_6$	F_2	M_2	$W_{22} \rightarrow W_6$	F_2	x_2
$S_{23} \rightarrow S_7$	F_3	M_2	$W_{32} \rightarrow W_7$	F_3	x_2
$S_{24} \rightarrow S_8$	F_4	M_2	$W_{42} \rightarrow W_8$	F_4	x_2

5.2.1 Sistema de control

En la Figura 5.18 se presenta el diagrama del sistema de control multicanal:

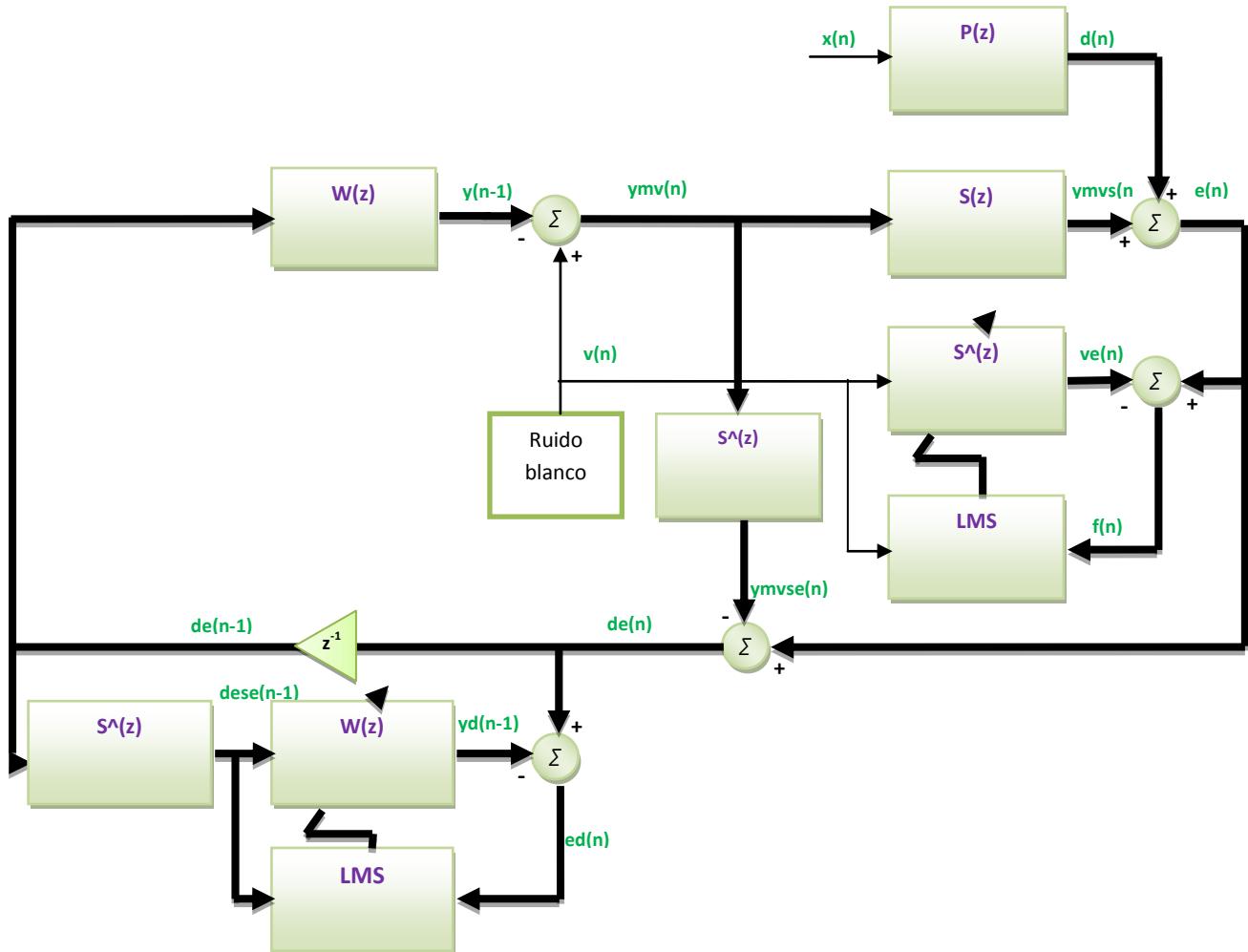


Figura 5.18: Diagrama del sistema de control

Las señales de referencia sintetizadas están dadas por [2]

$$x_m(n) = e_m(n) + \sum_{k=1}^K \hat{s}_{mk}(n) * y_k(n) \quad (5.2)$$

y las señales a las salidas de los filtros de control W son

$$y_k(n) = \sum_{m=1}^M w_{km}(n) * x_m(n) \quad (5.3)$$

En este caso en particular $M=2$ y $K=4$, por lo tanto manteniendo la nomenclatura correspondiente las señales expresadas anteriormente son

$$\begin{aligned} y_1(n) &= W_{c1}(z) * de_1(n) + W_{c5}(z) * de_2(n) \\ y_2(n) &= W_{c2}(z) * de_1(n) + W_{c6}(z) * de_2(n) \\ y_3(n) &= W_{c3}(z) * de_1(n) + W_{c7}(z) * de_2(n) \\ y_4(n) &= W_{c4}(z) * de_1(n) + W_{c8}(z) * de_2(n) \end{aligned} \quad (5.4)$$

Finalmente, las señales de error están dadas por las siguientes ecuaciones, respetando la nomenclatura actual

$$\begin{aligned} e_1(n) &= d_1(n) + y_1(n) * S_1(z) + y_2(n) * S_2(z) + y_3(n) * S_3(z) + y_4(n) * S_4(z) \\ e_2(n) &= d_2(n) + y_1(n) * S_5(z) + y_2(n) * S_6(z) + y_3(n) * S_7(z) + y_4(n) * S_8(z) \end{aligned} \quad (5.5)$$

5.2.2 Señal de ruido

En la implementación del sistema de control multicanal se utilizaron las mismas señales de ruido que en el sistema monocanal, con la diferencia de que la tercera señal

consiste en una señal diferente para cada sensor de error, ya que los caminos primarios involucrados son distintos.

5.2.3 Caminos secundarios

Se realizó la medición de la respuesta impulsiva de los caminos secundarios en el modelo propuesto, y se utilizaron como caminos secundarios en la simulación tomando solo los primeros 100ms (800 coeficientes) de cada uno. En la Figura 5.19 se presentan dichas respuestas

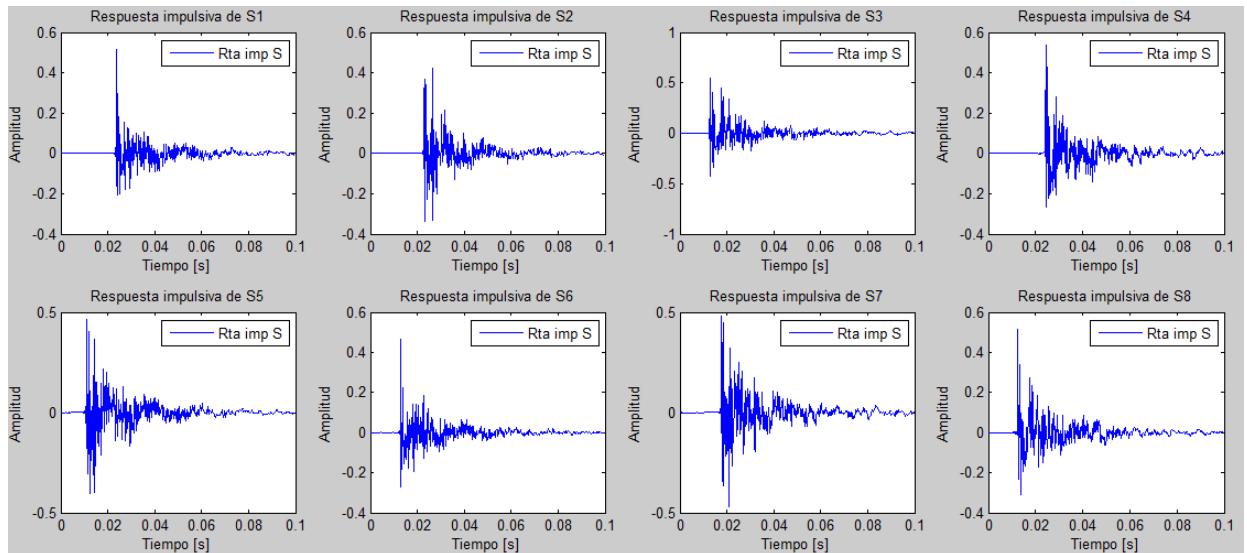


Figura 5.19: Caminos secundarios en sistema ANC multicanal

5.2.4 Condiciones de simulación

La frecuencia de muestreo utilizada fue 8kHz, y se simuló la operación del sistema durante un tiempo fijo de 30s para las dos primeras señales de ruido, y 60s para la tercera señal.

Al iniciar el funcionamiento del sistema, en primer lugar se realizó un aprendizaje de cada camino secundario S , utilizando como señal de entrada ruido blanco de media 0 y varianza 1. Dicho proceso de aprendizaje finalizó en el momento en que el valor medio cuadrático del error f de la Figura 5.18 se estabilizó, es decir que no presentó variaciones mayores a un porcentaje determinado. El camino secundario fue implementado con un filtro FIR de 800 coeficientes, es decir que se tomaron los primeros 100ms del mismo, y el filtro utilizado para estimar el camino secundario fue un filtro FIR de 600 coeficientes. A continuación, se presentan los caminos secundarios implementados S , y los caminos secundarios estimados (S_e) que se obtuvieron al finalizar el proceso de adaptación de los mismos.

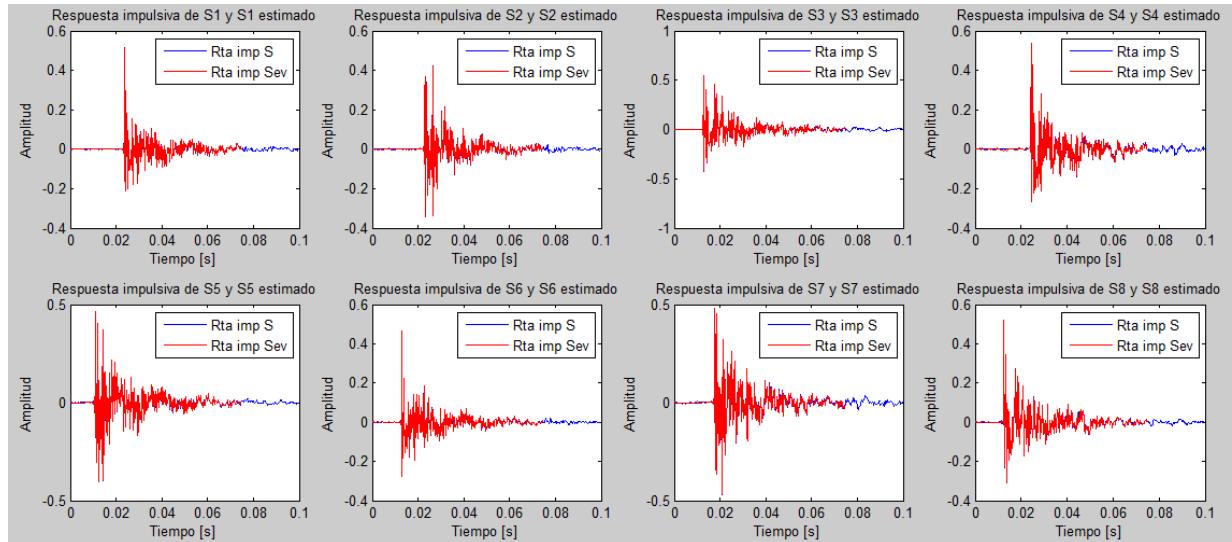


Figura 5.20: Caminos secundarios y aprendizaje de los mismos en sistema multicanal

Nuevamente, la estimación es muy exacta dentro del rango temporal cubierto por los coeficientes. Para ver alguna diferencia habría que hacer un zoom como se hizo en la Figura 5.8. Más allá del rango cubierto por los coeficientes el sistema nada puede hacer para estimar al camino secundario

5.2.5 Resultados

Aquí se presentan los resultados de las simulaciones para las tres señales de entrada. En cada caso, tanto el paso de adaptación u , como el largo del filtro $W(L_w)$, fueron

determinados de modo de obtener las mayores atenuaciones posibles con el filtro W más corto posible.

5.2.5.1 Señal número 1

En la tabla 5.5 se presenta la configuración de los parámetros utilizada para la señal número 1 y la atenuación obtenida, y en las figuras 5.21, 5.22 y 5.23 puede apreciarse el desempeño del sistema ANC.

Tabla 5.5: Configuración de parámetros y atenuación obtenida.

L_w	MSE final	Atenuación 100Hz
50	6 e-24	292 dB

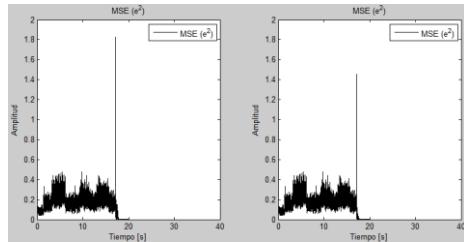


Figura 5.21: Evolución temporal del error medio cuadrático

En los primeros 17 segundos de la Figura 5.21 se observa el ruido blanco utilizado para el aprendizaje de los caminos secundarios. Luego, en el momento en que finaliza el aprendizaje de todos estos, la amplitud del error presenta un pico al comenzar la señal de ruido primario a cancelar. Una vez que comienza a actuar el sistema ANC se logra cancelar rápidamente el ruido, el cual luego de los 20s es prácticamente imperceptible. Este comportamiento se repite en las figuras 5.25, 5.28, 5.33 y finalmente 5.44. En la Figura 5.22 se observa en detalle como decae el error (ampliación Figura 5.21) cuando converge el sistema.

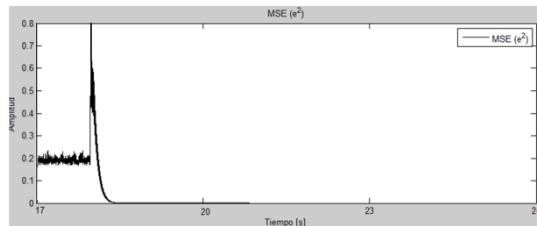


Figura 5.22: Evolución temporal del error medio cuadrático

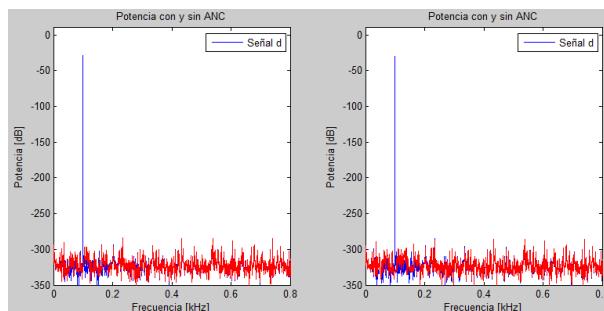


Figura 5.23: Análisis espectral de la señal de error con el ANC encendido y apagado

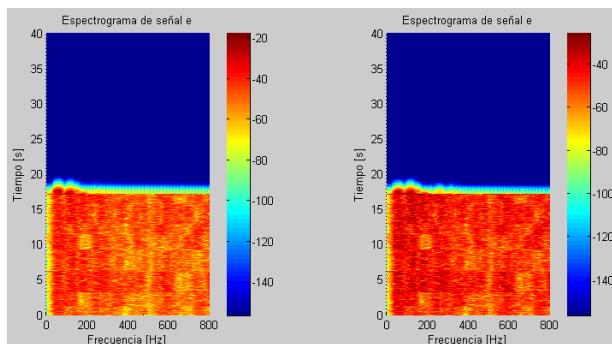


Figura 5.24: Evolución temporal de la PSD de la señal de error

5.2.5.2 Señal número 2

En la tabla 5.6 se presenta la configuración de los parámetros utilizada para la señal número 2 y la atenuación obtenida, y en las figuras 5.25, 5.26 y 5.27 puede apreciarse el desempeño del sistema ANC.

Tabla 5.6: Configuración de parámetros y atenuación obtenida.

L_w	MSE final	Atenuación 100Hz
50	1.2 e-5	98 dB

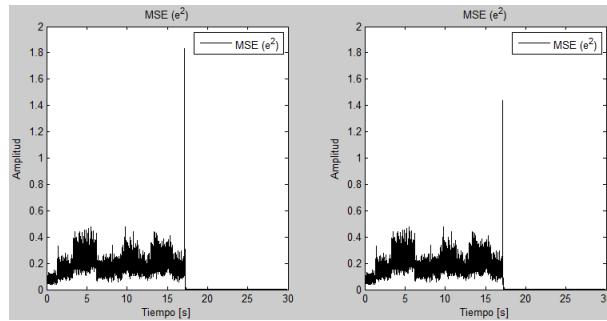


Figura 5.25: Evolución temporal del error medio cuadrático

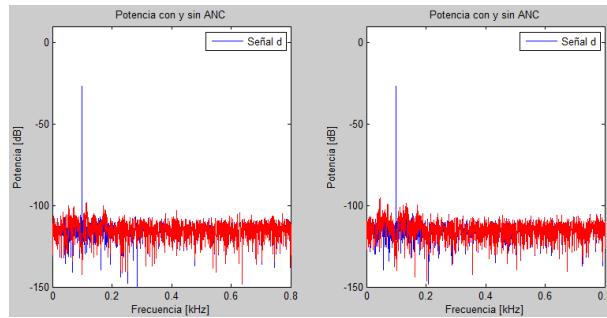


Figura 5.26: Análisis espectral de la señal de error con el ANC encendido y apagado

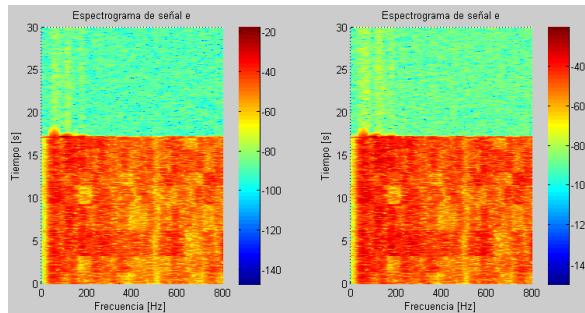


Figura 5.27: Evolución temporal de la PSD de la señal de error

5.2.5.3 Señal número 3

En la tabla 5.7 se presenta la configuración de los parámetros utilizada para la señal número 2 y la atenuación obtenida, y en las figuras 5.28, 5.29 y 5.30 puede apreciarse el desempeño del sistema ANC.

Tabla 5.7: Configuración de parámetros y atenuación obtenida.

L_w	MSE final	Att83Hz [dB]	Att166Hz [dB]	Att249Hz [dB]	Att332Hz [dB]	Att415Hz [dB]	Att498Hz [dB]	Att581Hz [dB]
200	1	97	85	67	96	72	42	11

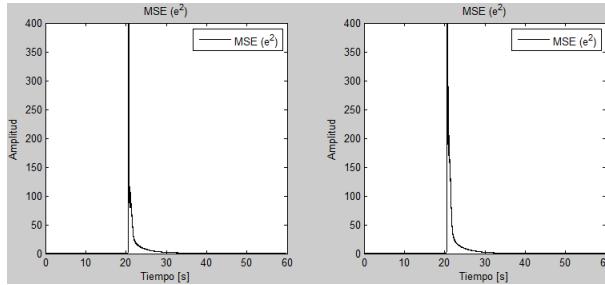


Figura 5.28: Evolución temporal del error medio cuadrático

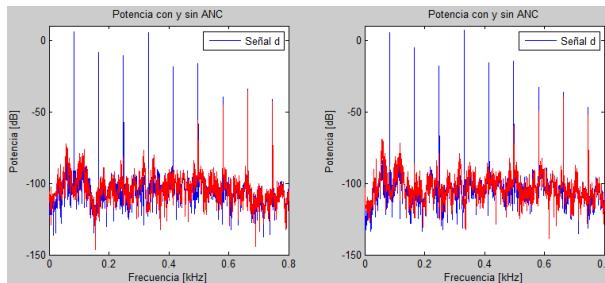


Figura 5.29: Análisis espectral de la señal de error con el ANC encendido y apagado

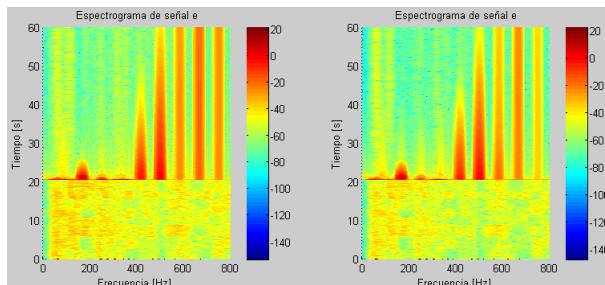


Figura 5.30: Evolución temporal de la PSD de la señal de error

5.3 Optimización

Las simulaciones llevadas a cabo muestran que el sistema multicanal de control activo de ruido operando con 4 fuentes y 2 sensores permite obtener resultados óptimos. Sin embargo, a la hora de implementar dicho sistema en un DSP debe tenerse en cuenta el elevado costo computacional exigido por el sistema. La configuración necesaria para lograr los resultados de la sección 5.2.5.3 anterior requiere que entre muestra y muestra, el DSP realice 24 filtrados FIR de 200 coeficientes (filtros W y W_{copia}), 24 filtrados FIR de 600 coeficientes (filtros S_e a y b) y la actualización de 8 filtros FIR de 200 coeficientes cada uno (filtros W) con el algoritmo LMS.

Cuando se implementa un filtro FIR en un procesador digital de señal, para obtener el resultado de la señal filtrada deben realizarse tantas multiplicaciones y acumulaciones (MAC: Multiply and Accumulate) como coeficientes tenga cada filtro. Para realizar esto dentro de un DSP es necesario ubicar los punteros en los coeficientes del filtro y en los valores de las entradas actuales y anteriores, realizar las multiplicaciones y acumulaciones, y finalmente guardar en memoria el resultado del filtrado. Dada la cantidad de filtros en juego y la longitud de cada uno de ellos, esto supone un costo computacional que puede considerarse elevado.

Sin embargo, el mayor problema lo constituye la actualización de los coeficientes de los filtros mediante el algoritmo LMS. Esto se debe a que para actualizar un coeficiente se debe obtener el valor de dicho coeficiente y de la señal de entrada correspondiente, realizar la multiplicación y acumulación, y guardar dicho resultado en memoria. Es decir que cada coeficiente debe ser actualizado y devuelto a memoria, lo que implica más instrucciones dentro del lazo. Este ciclo debe ser repetido para cada uno de los coeficientes del filtro, lo cual implica que la actualización de los coeficientes supone un costo computacional aún mayor que el filtrado FIR.

Con el objetivo de disminuir el poder de procesamiento requerido para un funcionamiento eficiente del sistema, se investigaron distintas soluciones que en conjunto permitieron disminuir el largo y la cantidad de filtros a utilizar.

5.3.1 Sistema 2x2

La primera modificación propuesta fue utilizar un sistema multicanal con 2 fuentes y 2 sensores, debido a que ello reduce a menos de la mitad la cantidad de filtros involucrados. Además, se propuso una configuración en donde las fuentes se sitúan sobre el techo, arriba de la zona deseada de quietud, de modo que esto permita disminuir la cantidad de coeficientes de los filtros encargados de estimar los caminos secundarios. Para llevar a cabo un análisis más riguroso, las respuestas impulsivas medidas de los caminos secundarios fueron editadas eliminando el retardo presente en la medición. En la Figura 5.31 puede verse el plano del nuevo sistema a utilizar

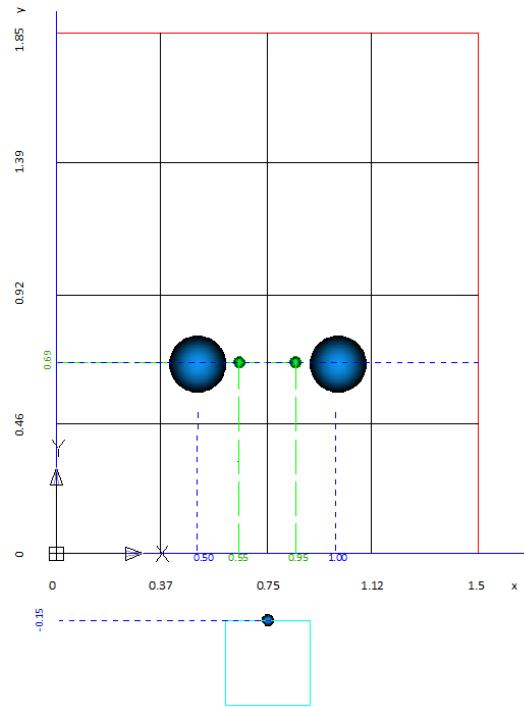


Figura 5.31: Plano del sistema multicanal 2 x 2

Los filtros involucrados y sus respectivas nomenclaturas son los siguientes

Tabla 5.8: Nomenclaturas en el sistema ANC 2 x 2

Caminos secundarios	Fuente	Sensor	Filtros W	Fuente	Referencia
$S_{11} \rightarrow S_1$	F_1	M_1	$W_{11} \rightarrow W_1$	F_1	x_1
$S_{12} \rightarrow S_2$	F_2	M_1	$W_{21} \rightarrow W_2$	F_2	x_1
$S_{21} \rightarrow S_3$	F_1	M_2	$W_{12} \rightarrow W_3$	F_1	x_2
$S_{22} \rightarrow S_4$	F_2	M_2	$W_{22} \rightarrow W_4$	F_2	x_2

Se realizó la medición de la respuesta impulsiva de los caminos secundarios en el modelo propuesto, y se utilizaron como caminos secundarios en la simulación tomando solo los primeros 100ms (800 coeficientes) de cada uno. A continuación, se presentan dichas respuestas

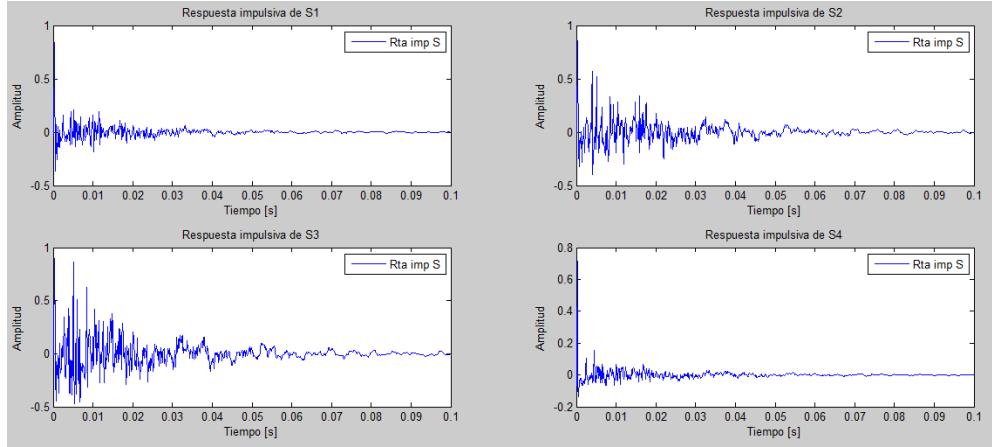


Figura 5.32: Respuestas impulsivas de los nuevos caminos secundarios

Los filtros utilizados para estimar los caminos secundarios fueron filtros FIR de 500coeficientes (en lugar de 600 como anteriormente). A continuación, se presentan los caminos secundarios S , y los caminos secundarios estimados que se obtuvieron al finalizar el proceso de adaptación de los mismos, en la Figura 5.33.

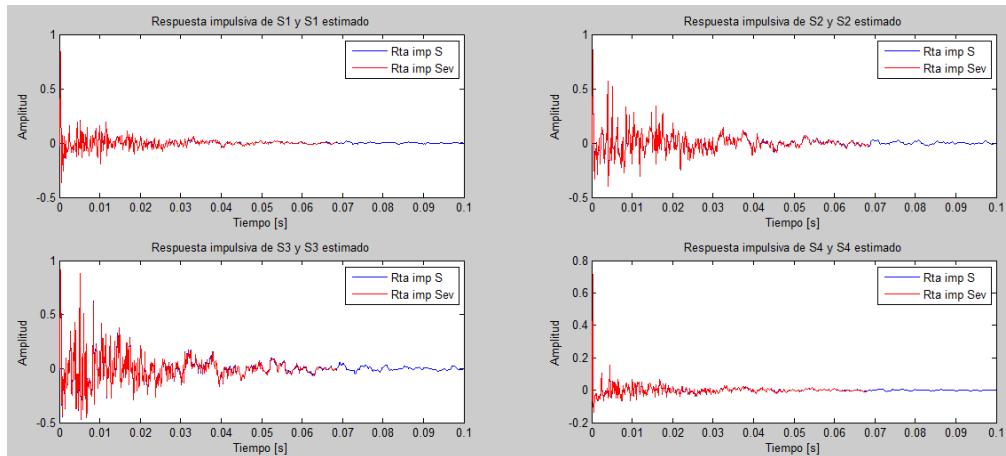


Figura 5.33: Caminos secundarios y caminos secundarios estimados en sistema 2x2

Nuevamente, la estimación es muy exacta dentro del rango temporal cubierto por los coeficientes. Para ver alguna diferencia habría que hacer un zoom como se hizo en la figura 5.7. Más allá del rango cubierto por los coeficientes el sistema nada puede hacer para estimar al camino secundario

En las tablas y figuras siguientes se presentan los resultados de las simulaciones para el ruido de motor de combustión sintetizado. En cada caso, tanto el paso de adaptación u , como el largo del filtro W (L_w), fueron determinados de modo de obtener las mayores atenuaciones posibles con el filtro W más corto posible. En la Tabla 5.10 se muestra el largo de los filtros utilizados y las atenuaciones obtenidas, y en las figuras 5.34, 5.35 y 5.36 se observa la señal de error.

Tabla 5.10: Configuración de parámetros y atenuaciones obtenidas.

L_w	L_{se}	MSE final	Att83Hz	Att166Hz	Att249Hz	Att332Hz	Att415Hz	Att498Hz	Att581Hz
150	500	0.1	81	77	62	63	13	23	1

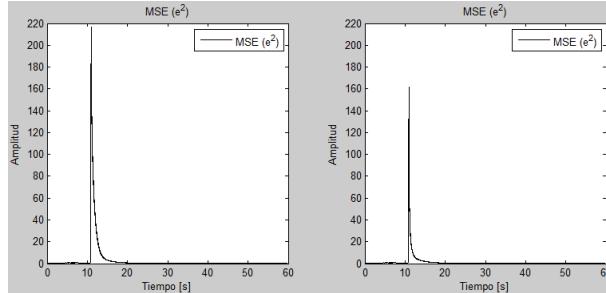


Figura 5.34: Evolución temporal del error medio cuadrático

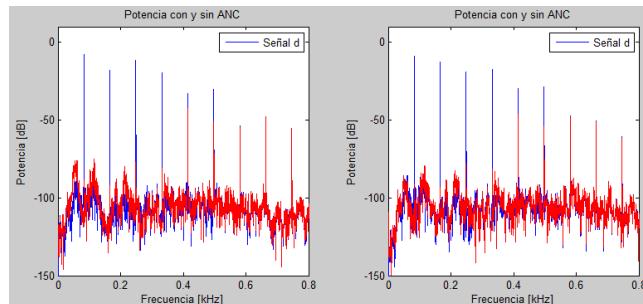


Figura 5.35: Análisis espectral de la señal de error con el ANC encendido y apagado

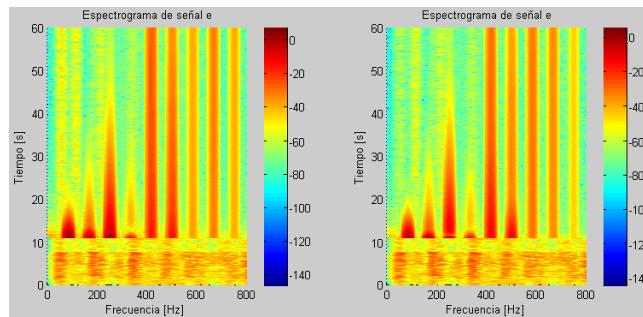


Figura 5.36: Evolución temporal de la PSD de la señal de error

5.3.2 Aumento de la absorción acústica en el recinto

Para lograr simplificar los caminos secundarios se colocaron grandes cantidades de material absorbente acústico en el interior del recinto. De esta manera se logra disminuir la energía presente en las reflexiones tempranas y tardías sin modificar el sonido directo, permitiendo estimar los caminos secundarios con filtros de menor largo.

Se presentan a continuación en la Figura 5.37 los primeros 100ms de los caminos secundarios medidos con la nueva configuración física del sistema

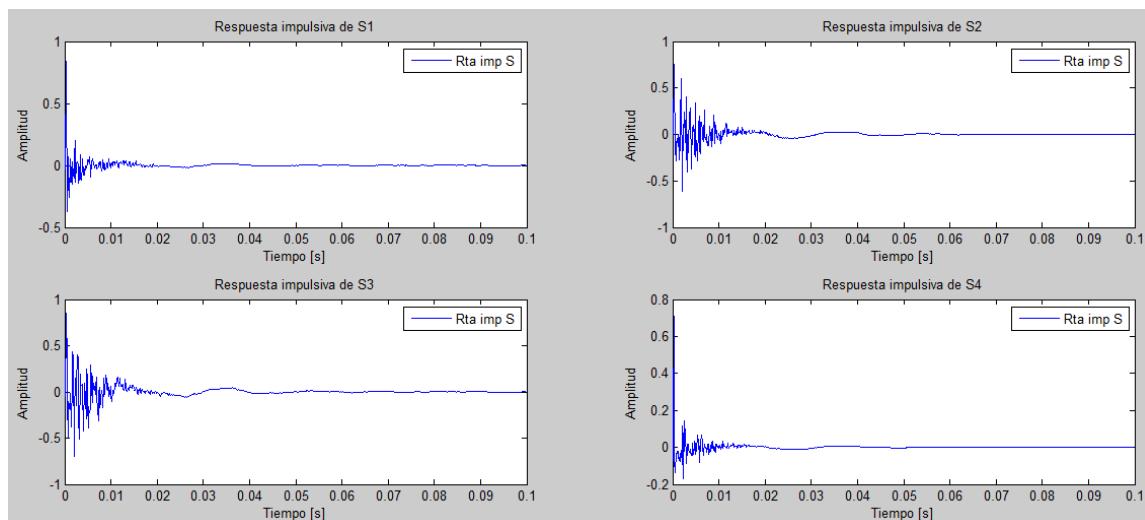


Figura 5.37: Caminos secundarios al aumentar la absorción acústica

Con esta nueva modificación se logró reducir el largo de los filtros S_e de 500 a 300 coeficientes, permitiendo además la posterior reducción del tamaño de los filtros W . Los caminos secundarios estimados son los siguientes

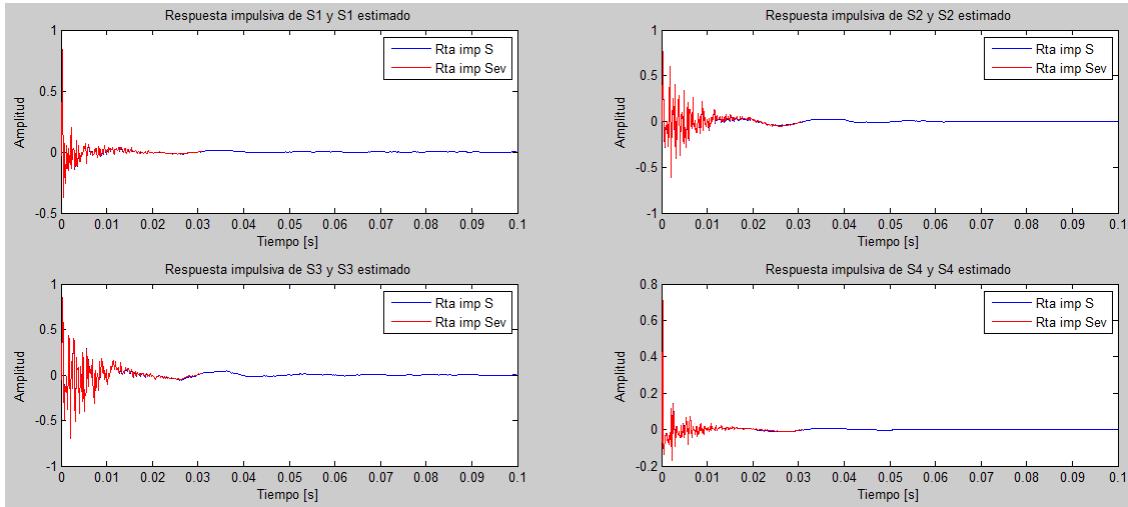


Figura 5.38: Caminos secundarios y caminos secundarios estimados al aumentar la absorción acústica

Los resultados obtenidos se plantean a continuación para $L_w = 50$ y $L_w = 100$ en la Tabla 5.10 y 5.11, y en las figuras 5.39 a 5.44.

Tabla 5.10: Configuración de parámetros y atenuaciones obtenidas.

L_w	L_{se}	Att83Hz	Att166Hz	Att249Hz	Att332Hz	Att415Hz	Att498Hz	Att581Hz
50	300	74	46	21	34	13	3	0

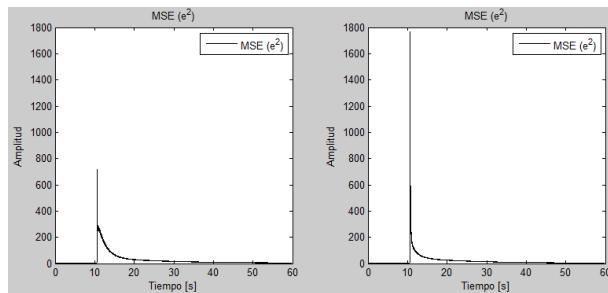


Figura 5.39: Evolución temporal del error medio cuadrático

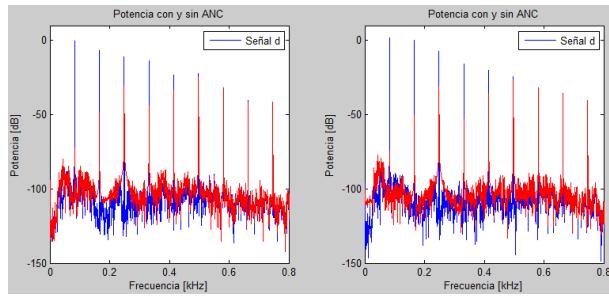


Figura 5.40: Análisis espectral de la señal de error con el ANC encendido y apagado

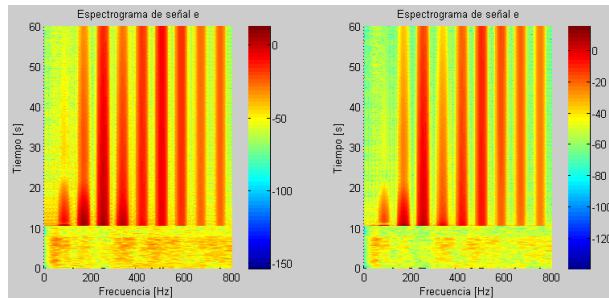


Figura 5.41: Evolución temporal de la PSD de la señal de error

Tabla 5.11: Configuración de parámetros y atenuaciones obtenidas.

L_w	L_{se}	Att83Hz	Att166Hz	Att249Hz	Att332Hz	Att415Hz	Att498Hz	Att581Hz
100	300	103	83	56	73	13	9	0

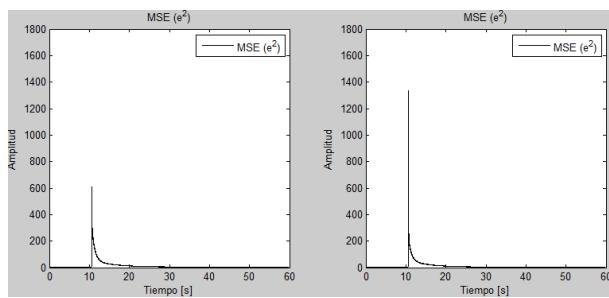


Figura 5.42: Evolución temporal del error medio cuadrático

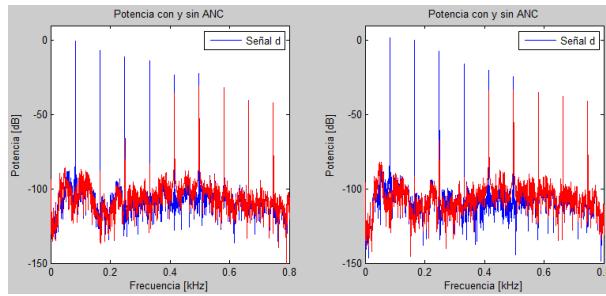


Figura 5.43: Análisis espectral de la señal de error con el ANC encendido y apagado

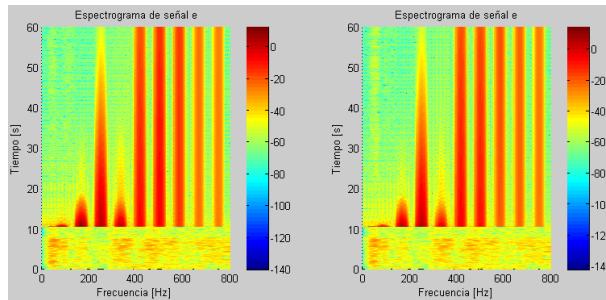


Figura 5.44: Evolución temporal de la PSD de la señal de error

5.3.3 Situación límite

Si bien en la Figura 5.39 se observa que el sistema converge para $L_w = 50$ y $L_{se} = 300$, se consideró que la longitud de los filtros de aprendizaje de los caminos secundarios podía reducirse aún más. Se decidió reducir el largo de los filtros Se hasta obtener el mínimo valor posible que mantenga la convergencia sin entrar en inestabilidad, lo cual trajo aparejado un ligero aumento en la longitud de los filtros de control W . La mejor solución de compromiso se obtuvo para $L_w = 70$ y $L_{se} = 105$. Si bien esto aumentó en 20 coeficientes a cada filtro de control, permitió reducir a un tercio la longitud de los filtros Se reduciendo considerablemente la carga computacional.

En la Tabla 5.12 y las Figuras 5.45, 5.46 y 5.47 se pueden ver los resultados obtenidos.

Tabla 5.12: Configuración de parámetros y atenuaciones obtenidas.

L_w	L_{se}	Att83Hz	Att166Hz	Att249Hz	Att332Hz	Att415Hz	Att498Hz	Att581Hz
70	105	73	61	30	30	11	3	0

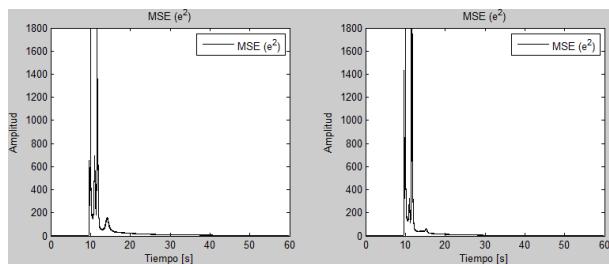


Figura 5.45: Evolución temporal del error medio cuadrático

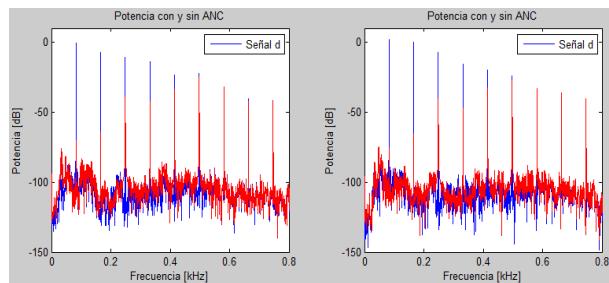


Figura 5.46: Análisis espectral de la señal de error con el ANC encendido y apagado

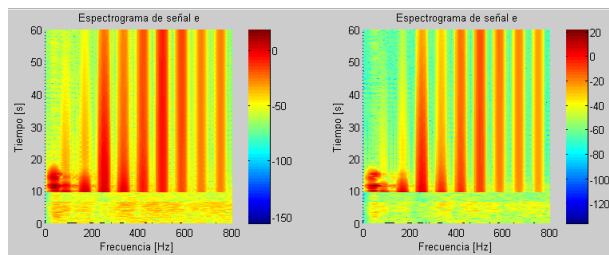


Figura 5.47: Evolución temporal de la PSD de la señal de error

5.3.4 Conclusiones de la optimización

Como conclusiones de la etapa de optimización, se decidió implementar un sistema de control activo de ruido de 4 canales, es decir dos fuentes secundarias y dos sensores de error, debido a que esto permite reducir en gran medida el costo computacional requerido para el procesador sin obtener un deterioro significativo de los resultados. En caso de que la capacidad de procesamiento del DSP no sea suficiente para implementar dicho sistema, la cabina deberá ser recubierta internamente con una mayor cantidad de material absorbente a modo de simplificar el campo sonoro y reducir la longitud necesaria de los filtros encargados de estimar el camino secundario. Al mismo tiempo, deberán utilizarse las técnicas apropiadas de programación que permitan obtener el máximo provecho del procesador seleccionado.

CAPÍTULO 6: Modelo experimental

En este capítulo se describe en detalle el prototipo desarrollado para la implementación del sistema ANC. Se especifican los diversos componentes que forman parte del lazo de control y se presenta el equipamiento y conexionado utilizado para poder realizar las mediciones en tiempo real del desempeño del sistema y el post procesamiento de los datos. Finalmente, se describe el procesador digital de señal utilizado.

6.1 Sistema multicanal de control activo de ruido

El sistema ANC está formado por los siguientes componentes:

- Módulo MSC7116EVM, que contiene el DSP StarCore 1600
- 2 sensores de error (micrófonos de medición omnidireccionales) Behringer ECM8000
- Pre amplificador de dos canales con phantom power M-Audio, modelo Audio-Buddy
- Amplificador de potencia de 4 canales (50 W_{RMS} por canal) Boss CE-404
- 2 fuentes pasivas direccionales montadas sobre recintos ventilados del tipo reflector de bajos

Los micrófonos omnidireccionales son los encargados de obtener las señales de error utilizadas en el proceso adaptativo, mientras que los preamplificadores cumplen una doble función: por un lado deben brindar la alimentación de 48V que requieren los micrófonos para su funcionamiento, ya que son del tipo electret-condensador y poseen un preamplificador interno; por otro lado deben amplificar la señal proveniente de los sensores a modo de adecuarla al rango de tensión de entrada de los ADC presentes en la placa de DSP y obtener una óptima relación señal ruido.

El amplificador de potencia es el encargado de amplificar la tensión y la corriente de la señal de antiruido generado por el DSP para que la misma sea capaz de excitar los altavoces de las fuentes secundarias, y radiar así las ondas sonoras necesarias para producir

la cancelación. Si bien se utilizan únicamente dos fuentes secundarias, la utilización de un amplificador de 4 canales corresponde a que el diseño inicial incluía 4 fuentes de control. En el momento en que se llevaron a cabo las simulaciones y se optimizó el sistema pudo determinarse que con dos fuentes secundarias los resultados no se veían afectados de manera significativa (ver sección 5.3.1).

A continuación se presenta en la Figura 6.1 un esquema de las conexiones de los distintos componentes.

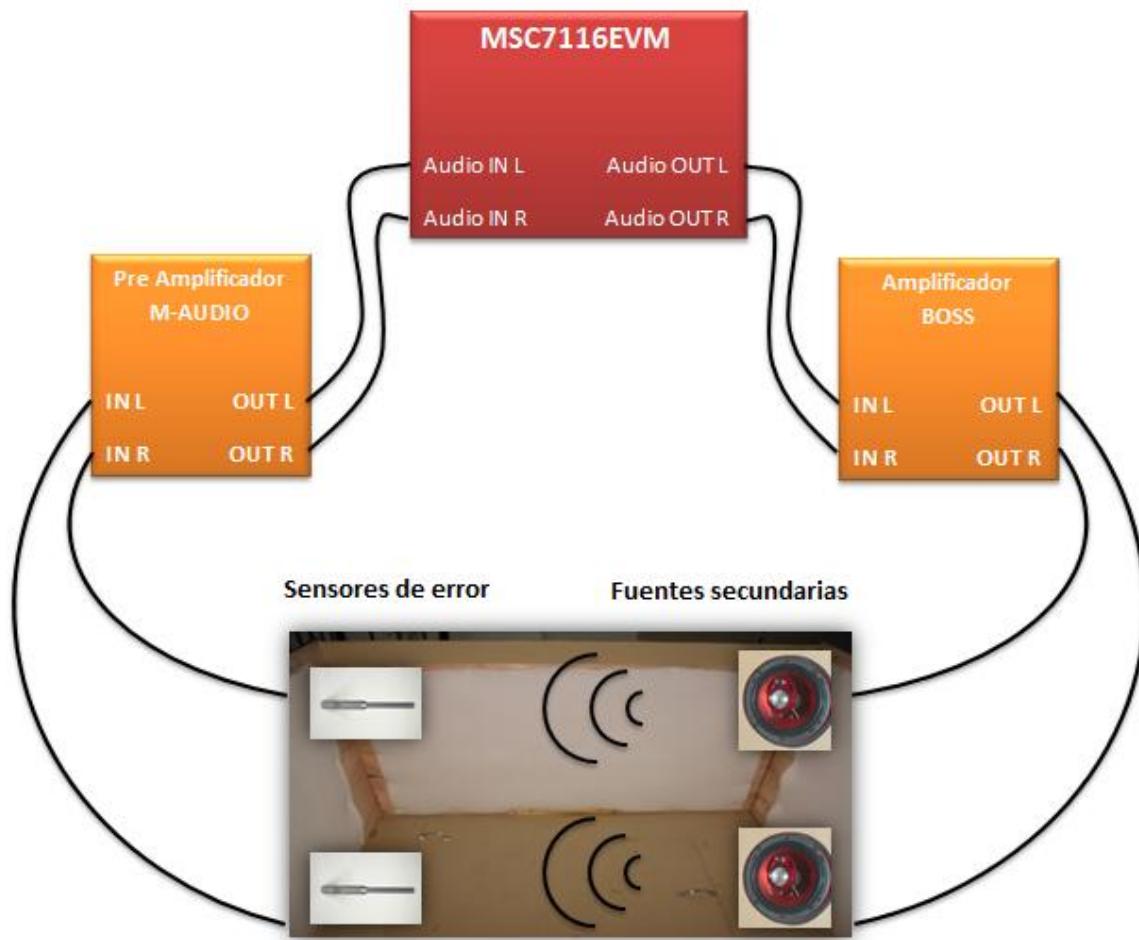


Figura 6.1: Esquema de conexiones del sistema ANC

6.2 Sistema para adquisición de datos y medición del desempeño

Para poder realizar la adquisición de las distintas señales que forman parte del sistema se utilizó la placa de adquisición de audio Presonus Audiobox 44 VSL, y una computadora portátil ACER Aspire modelo 4750. La placa de audio posee 4 canales de entrada, siendo todos ellos de alta impedancia de entrada, de modo que los mismos fueron conectados tanto a la entrada como a la salida de audio del DSP permitiendo adquirir simultáneamente las señales de error y las señales de antiruido en tiempo real. La conexión entre la placa de adquisición de audio y la computadora portátil es a través del bus USB.

Las señales fueron registradas con software profesional para grabación y edición de audio, y posteriormente se realizó el post procesamiento necesario con Matlab para realizar una evaluación precisa de las atenuaciones obtenidas en las distintas frecuencias. A continuación, se presenta un esquema del conexionado utilizado

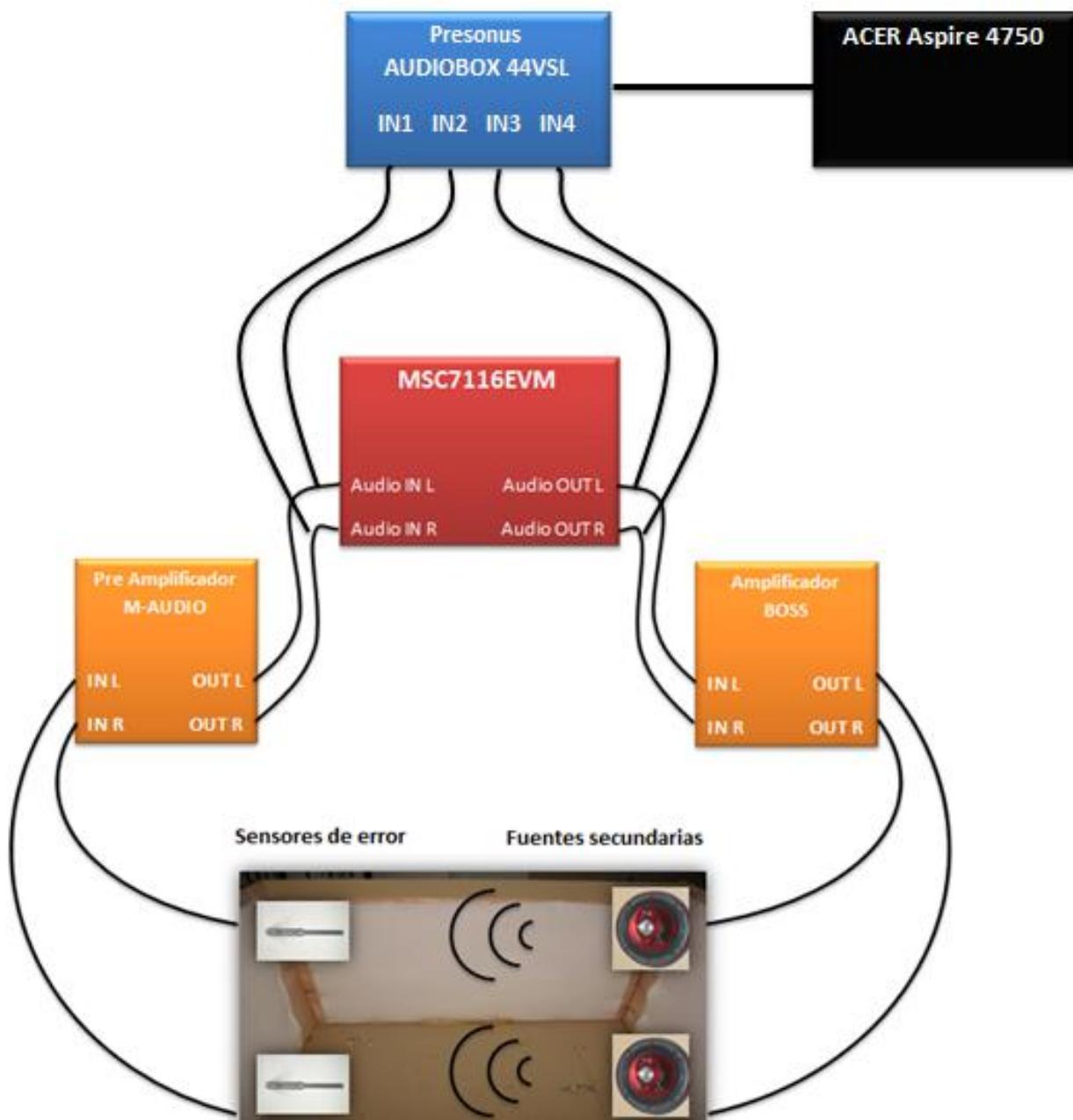


Figura 6.2: Diagrama de conexionado del sistema completo

A continuación, se presentan las fotografías del sistema completo implementado en el interior del recinto.

En la Figura 6.3 se observa el interior del recinto, en donde se encuentran instalados los sensores de error y las fuentes secundarias



Figura 6.3: Vista interior del recinto con sensores de error y fuentes secundarias

En la Figura 6.4 se aprecia la vista exterior del recinto con el sistema ANC



Figura 6.4: Vista exterior del recinto con el sistema ANC

En la siguiente figura puede verse la computadora portátil montada sobre la cabina con el módulo de amplificación de potencia.

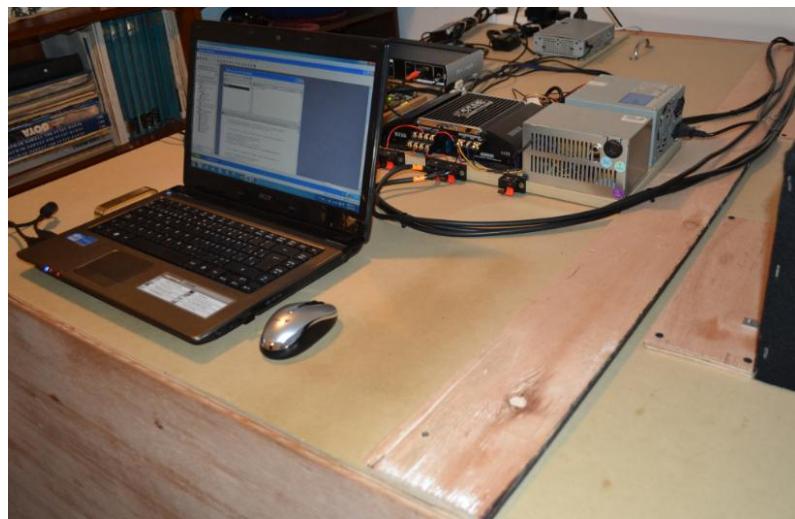


Figura 6.5: PC portátil y amplificación de potencia

Finalmente, en la Figura 6.6 se presenta el módulo MSC7116EVM en conjunto con la placa de adquisición de datos, el preamplificador para los sensores y las etapas de potencia



Figura 6.6: Etapa de adquisición y procesamiento de datos

6.3 Módulo de evaluación del DSP StarCore MSC7116

El DSP seleccionado para realizar todo el procesamiento digital e implementar el controlador fue el StarCore MSC7116 de la empresa Freescale Semiconductor Inc., cuyo módulo de evaluación es el MSC7116EVM. Este módulo 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, depuración y evaluación del software del controlador sobre el DSP, vía puerto USB a través de una PC portátil. Además, por medio del CODEC integrado en dicha placa de desarrollo, ingresan y salen las señales de los transductores electroacústicos (parlantes y micrófonos) que se procesarán a través del DSP. La Figura 6.7 muestra un diagrama en bloques del módulo de evaluación.

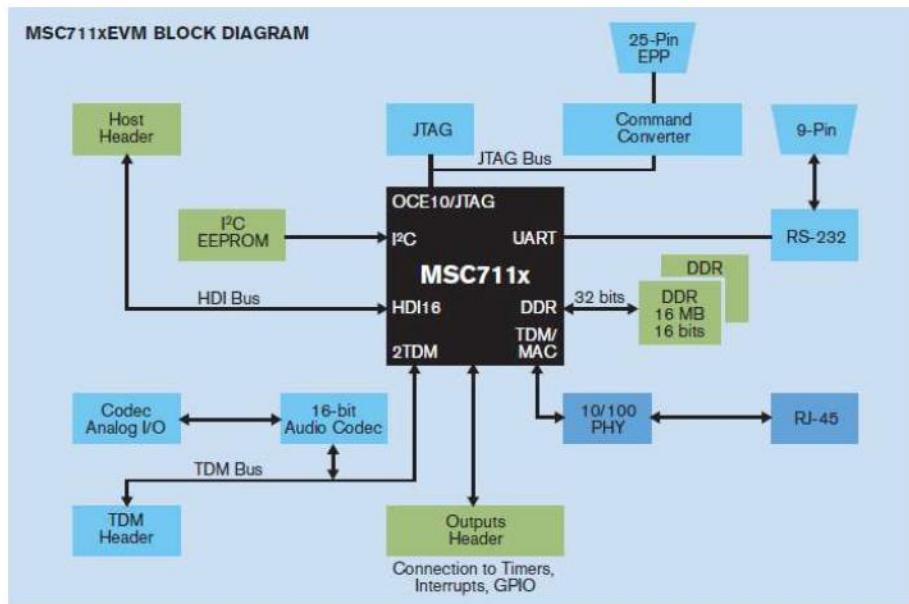


Figura 6.7: Diagrama en bloques del módulo de evaluación MSC7116EVM [68].

En la Figura 6.8 se presenta una fotografía del módulo utilizado señalando en rojo el DSP MSC7116



Figura 6.8: Fotografía del módulo MSC7116EVM, con el DSP MSC7116 resaltado en rojo.

6.3.1 DSP StarCore MSC7116

El StarCore MSC7116 es un DSP de coma fija de 16 bits de longitud de palabra, de bajo costo, con un núcleo SC1400 que tiene cuatro unidades lógicas aritméticas o ALUs (por sus siglas en inglés Arithmetic Logic Units), capaz de realizar 1000 millones de multiplicaciones y acumulaciones por segundo o MMACS a 266MHz. Cada ALU tiene una unidad de multiplicación y acumulación o MAC encargada de realizar las operaciones aritméticas. La misma ejecuta multiplicación fraccional o entera de 16 bits por 16 bits, entre operandos con signo en complemento a dos, sin signo, o mezclados, entregando un resultado de 32 bits. El acumulador de cada ALU, en el cual se almacena el resultado de 32 bits, es de 40 bits en la forma [Extensión (8 bits) - Porción alta (16 bits) - Porción baja (16 bits)]. Cada ALU tiene soporte también para operaciones aritméticas de enteros o fraccional de doble precisión, tales como multiplicación de 32 bits por 16 bits y multiplicación de 32 bits por 32 bits.

La alta capacidad de procesamiento del MSC7116, permite cumplir con la gran demanda de cálculo computacional que se requiere para la realización de los filtros adaptativos en el sistema, dentro del tiempo de un período de muestreo (ya que se realizó un procesamiento en tiempo real de muestra por muestra, como se explicó en la sección 4.1). La Figura 6.8 muestra un diagrama en bloques del MSC7116.

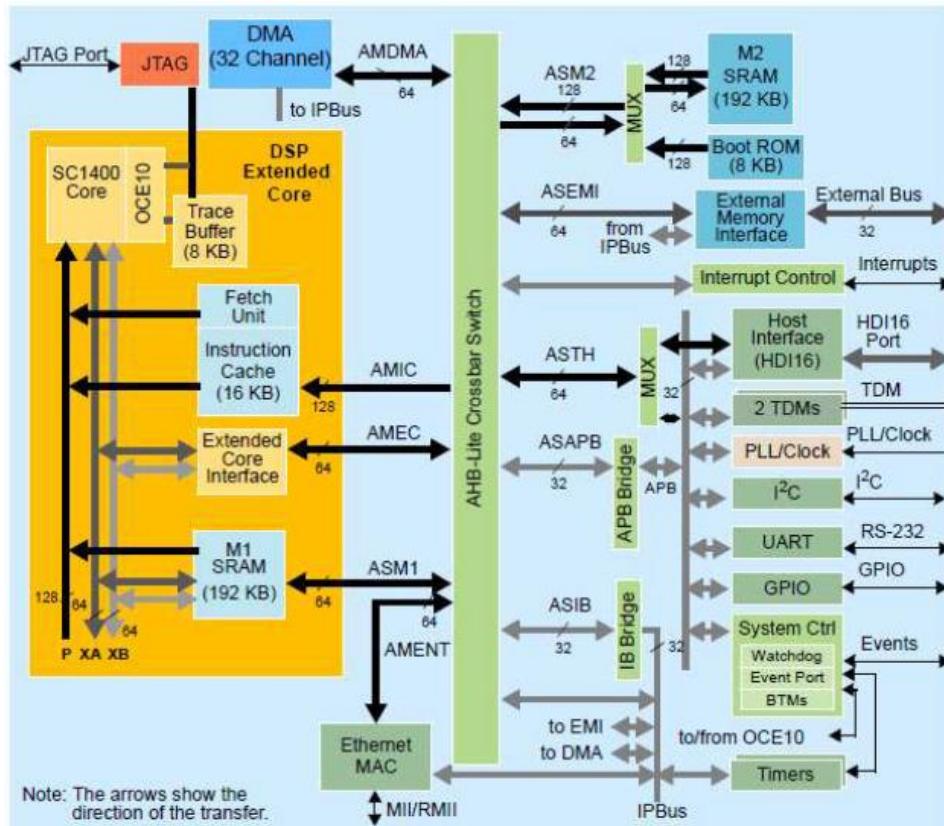


Figura 6.8: Diagrama en bloques del MSC7116 [41].

Por medio de una de sus interfaces TDM (por sus siglas en inglés Time-Division Multiplexing) se efectúa el flujo de datos de entrada/salida provenientes del CODEC dentro del DSP. La interfaz TDM es un puerto serial full-duplex por el cual el DSP se comunica con el CODEC AK4554.

6.3.2 CODEC AK4554

El AK4554 es un CODEC estéreo de 16 bits de la empresa ASAHI KASEI, el cual realiza las conversiones A/D y D/A para ambos canales empleando conversores del tipo delta-sigma, además de efectuar los filtrados “antialiasing” y de reconstrucción (filtros pasa bajos) correspondientes. El CODEC trabaja con una frecuencia de muestreo que es seleccionada automáticamente por medio de la combinación de tres relojes o clocks provistos al mismo. Estos tres clocks son: el clock maestro MCLK, el clock de entrada y salida de datos LRCK que es igual a la frecuencia de muestreo, y el clock de datos SCLK. La Figura 6.9 muestra el diagrama en bloques del CODEC AK4554.

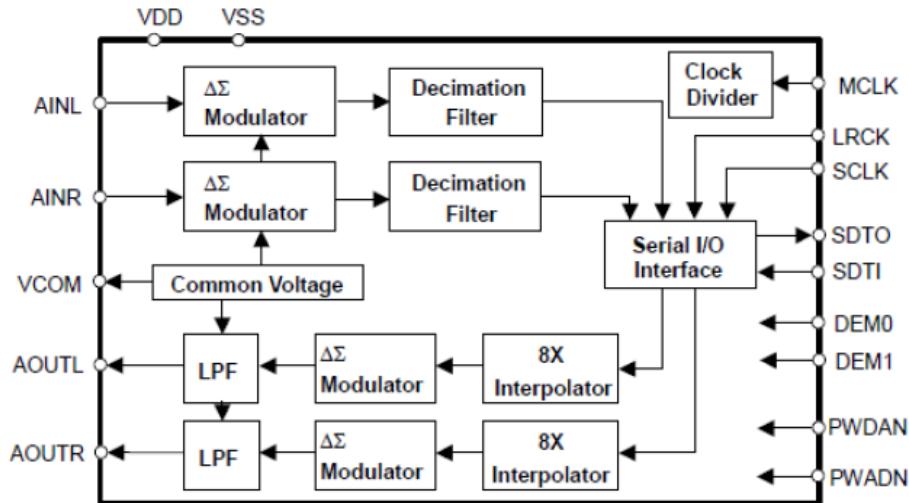


Figura 6.9: Diagrama en bloques del CODEC AK4554 [42].

El módulo de evaluación tiene un oscilador de 8,192MHz, el cual está conectado al pin EVENT0 del DSP, mientras que la señal MCLK es suplida al CODEC a través del pin EVENT1 del mismo. Esto permite que la señal del oscilador pueda ser dividida por el timer A del DSP y luego ser usada como MCLK para el CODEC. También, dicha señal de 8,192MHz es dividida por el timer B del DSP, de tal manera de proveer al periférico TDM el clock necesario para su funcionamiento. Luego la interfaz TDM entrega al CODEC los clocks LRCK y SCLK. Cambiando el valor del clock del TDM permite al usuario poder seleccionar las frecuencias de muestreo en 8kHz, 16kHz o 32kHz.

La frecuencia de muestreo del sistema se fijó en 8kHz. Este valor de frecuencia de muestreo es suficiente para la aplicación, ya que el rango de frecuencias de interés de un sistema ANC corresponde a frecuencias por debajo de los 500Hz [10]. Los valores seleccionados de cada uno de los clocks del CODEC fueron los siguientes:

$$LRCK = f_s = 8 \text{ kHz}$$

$$MCLK = 1024 f_s = 8.192 \text{ MHz}$$

$$SCLK = 32 f_s = 0.256 \text{ MHz}$$

Siendo f_s la frecuencia de muestreo. La frecuencia del clock MCLK fue seleccionada a 1024 veces la frecuencia de muestreo, ya que a ese valor se obtiene la mayor relación señal ruido del conversor D/A de 90dB. Por lo tanto, no hizo falta usar el timer A ya que no fue necesario dividir la señal del oscilador para obtener el clock MCLK. La Figura 6.10 muestra la interconexión entre el CODEC y el DSP.

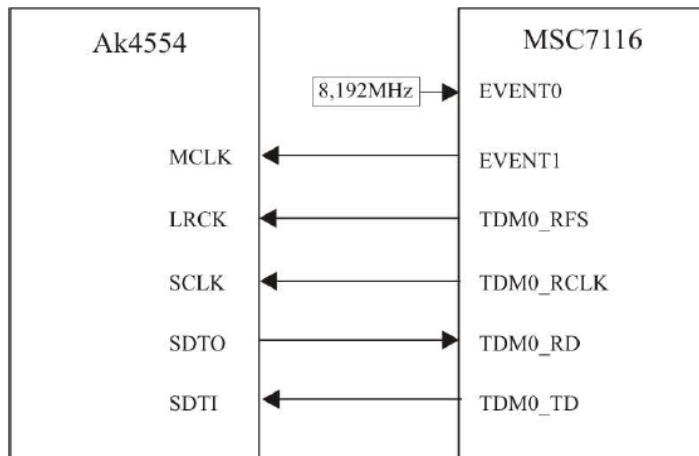


Figura 6.10: Interconexión entre CODEC y DSP. Adaptada de [1]

CAPÍTULO 7: Implementación en el DSP

En este capítulo se aborda la implementación del sistema ANC en el procesador digital de señal. Se abarca tanto la descripción del sistema operativo en tiempo real utilizado, como la optimización del código desarrollado para permitir un desempeño óptimo del sistema.

7.1 Descripción de SmartDSP

El programa desarrollado se encuentra montado sobre un sistema operativo de tiempo real o RTOS (por sus siglas en inglés Real Time Operating System), que se ejecuta en el DSP. Dicho sistema operativo es el Smart DSP de la empresa Freescale Semiconductor, diseñado especialmente para los DSP de la familia StarCore, y está integrado dentro del entorno de programación CodeWarrior usado para la depuración y programación de dichos DSPs.

La versión de este entorno que se utilizó se conoce como *Code Warrior Development Studio for Freescale MSC711X DSP Architectures v3.2.3*. El SmartDSP es un sistema operativo rápido y de bajo costo computacional, que realiza multitarea apropiativa basada en prioridades y provee baja latencia de interrupción. Su interface de programación de aplicación o API (por sus siglas en inglés Application Program Interface) [34], está formada por funciones que permiten una fácil configuración y utilización de los periféricos del DSP.

7.1.1 Arquitectura

El RTOS SmartDSP tiene la arquitectura mostrada en la Figura 7.1. En ella se observa principalmente el núcleo o kernel del sistema operativo, dos capas para los drivers (indicándose también la API de la capa superior) y los protocolos de redes soportados.

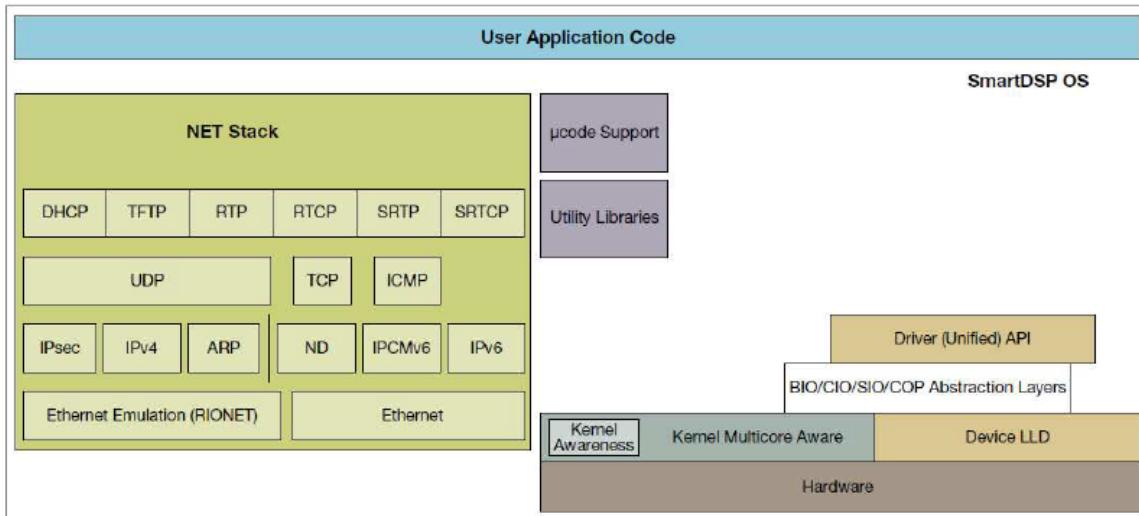


Figura 7.1: Arquitectura del RTOS SmartDSP [43].

Cada driver del RTOS SmartDSP está compuesto de dos capas:

- **Serializador**: Es la capa superior independiente del tipo de hardware de cada dispositivo o periférico; permitiendo de esta manera la abstracción del hardware para la aplicación. Por medio del uso de su API se realiza la transferencia de datos entre el programa de la aplicación y los periféricos.
- **Driver de nivel bajo o LLD** (por sus siglas en inglés Low Level Driver): Es la capa inferior, la cual es dependiente del hardware. La misma realiza las operaciones de entrada y salida correspondientes con cada dispositivo o periférico. La capa LLD debe registrarse con el serializador, de tal manera de corresponder sus funciones con las del serializador.

La transferencia de datos en un driver se realizó a través de canales. De esta manera, al transmitir datos se utilizó un canal transmisor (de escritura) o, en el caso de recibir datos, un canal receptor (de lectura). Existe un driver distinto para cada tipo de datos, con su correspondiente módulo serializador y LLD. La Tabla 7.1 muestra los distintos módulos serializadores de cada tipo de driver.

Tabla 7.1: módulos serializadores de los drivers

Módulo serializador	Funcionalidad
Módulo de Carácteres de I/O (CIO)	Orientado a la trasferencia de cadenas y caracteres, como la que ocurre con un dispositivo UART.
Módulo de Búferes de I/O (BIO)	Orientado a la transferencia de paquetes de datos, como la que ocurre con un dispositivo Ethernet.
Módulo Sincronizado de I/O (SIO)	Orientado a la transmisión y recepción de datos temporizada por hardware, como la que ocurre con un dispositivo TDM.
Módulo Coprocesador (COP)	Soporta coprocesadores. Este módulo se encuentra en la última versión del SmartDSP.

La Figura 7.2 grafica lo explicado anteriormente.

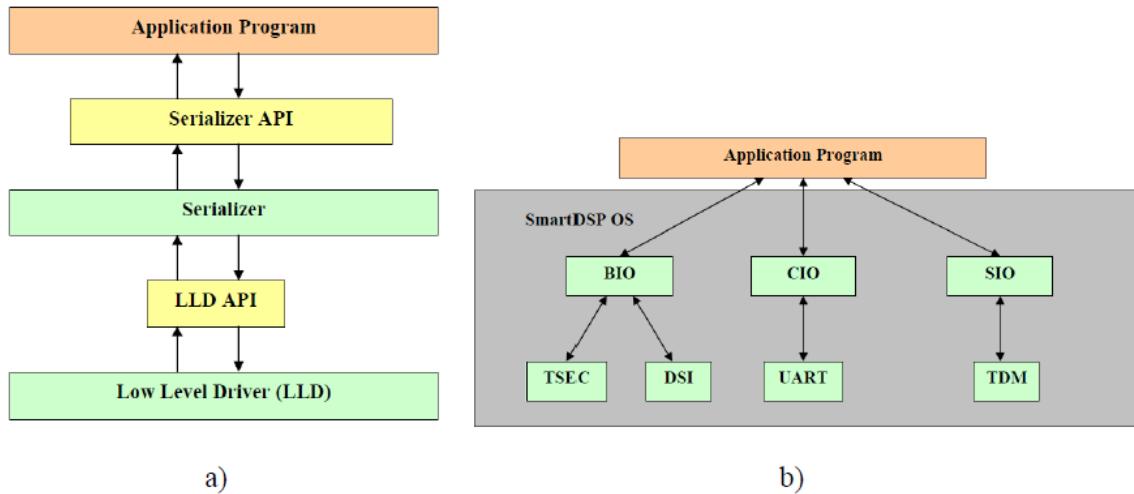


Figura 7.2: a) Modelo de dos capas del driver con sus respectivas APIs, y b) modelo especificado para cada tipo de módulo serializador [66].

Como ya se explicó en la sección 6.3.1, la entrada y salida de datos se recibirá desde el periférico TDM que se encuentra conectado al CODEC. El driver del periférico TDM se explica a continuación.

7.2 Driver del periférico TDM

El driver del periférico TDM es un driver del tipo SIO utilizado para cuando las operaciones de entrada y salida de datos son temporizadas por hardware. Desde el punto de vista de la aplicación, existen dos canales (0 y 1), cada uno ocupando un determinado espacio de tiempo o “time slot” dentro de la trama o “frame” del formato serial de datos. Mientras que desde el punto de vista del driver SIO, existe un solo canal del tipo intercalado, conteniendo los dos “time slots” de manera intercalada. Como están separadas las operaciones de transmisión y recepción, el canal intercalado existe en sus dos formas, canal intercalado transmisor y canal intercalado receptor.

Empleando la definición *sio_channel_t* se reservó memoria para la estructura interna de cada canal intercalado. Un acceso directo a memoria o DMA (por sus siglas en inglés Direct Memory Access) fue utilizado por el driver para transferir los datos entre el periférico TDM y memoria. De esta manera los datos de ambos canales de la aplicación se recibieron directamente en un búfer llamado búfer intercalado de recepción, y los datos a transmitir de dichos canales en un búfer llamado búfer intercalado de transmisión [1].

Como el procesamiento de datos en tiempo real de la aplicación fue del tipo muestra por muestra para cada canal, la transferencia de datos de ambos canales por medio del DMA se realizó a una frecuencia que es igual a la de muestreo.

7.2.1 Inicialización

Antes de poder usar el driver, el mismo debió ser inicializado durante la fase de inicialización del SmartDSP, provisto que la bandera MSC711X_TDM0 del archivo de configuración del sistema operativo este seleccionada en ON. En esta fase el periférico TDM0 fue configurado de acuerdo a los parámetros seleccionados en su estructura de inicialización *msc711x_tdm_init_params_t*, además de registrarse el driver del mismo con el serializador SIO [1]. Como resultado de esto último el programa de la aplicación pudo usar el TDM0 a través de la API del SIO.

En la fase de inicialización de la aplicación el dispositivo SIO fue abierto, para luego poder abrir canales para la realización de las operaciones de transmisión y recepción de datos. Para abrir un dispositivo SIO se utilizó la función *osSioDeviceOpen()* de la API

del SIO, indicando el nombre del dispositivo, el cual es el periférico TDM0, y entregando un puntero a su estructura de configuración *sio_dev_open_params_t*. En esta estructura, se definieron las rutinas de servicio de interrupción de recepción de datos y de eventos de error.

Como se verá más adelante, el programa de la aplicación utilizó solamente la interrupción de recepción de datos para funcionar, no empleando la interrupción de transmisión.

La función *osSioDeviceOpen()* entrega un handle del dispositivo, del tipo *sio_dev_handle*. Una vez abierto el dispositivo SIO que representa al periférico TDM0, se abrieron los canales SIO transmisor y receptor del tipo intercalados. Cada canal se abrió empleando la función *osSioChannelOpen()* de la API del SIO, especificando si es un canal de escritura (transmisor) o si es de lectura (receptor). Como parámetros de esta función, se entregaron a la misma el handle del dispositivo, un puntero a la estructura interna del canal *sio_channel_t* y otro puntero a la estructura de configuración del canal *sio_ch_open_params_t*.

Esta estructura de configuración del canal, contiene la dirección y tamaño del búfer intercalado correspondiente y la cantidad de búferes usados, la cual es igual a dos. Para obtener la dirección del búfer intercalado se utilizó la función *osSioDeviceCtrl()* de la API del SIO, empleando como parámetros los comandos TDM_CMD_TX_INTERLEAVE_BASE_GET (búfer transmisor) y TDM_CMD_RX_INTERLEAVE_BASE_GET (búfer receptor). Para obtener el tamaño del búfer intercalado se utilizó la misma función, pero ahora empleando como parámetros los comandos TDM_CMD_TX_INTERLEAVE_SIZE_GET (búfer transmisor) y TDM_CMD_RX_INTERLEAVE_SIZE_GET (búfer receptor) [1].

7.2.2 Proceso de transmisión y recepción de datos

Teniendo abiertos los canales transmisores y receptores, se pudieron realizar las operaciones de transmisión y recepción de datos. Como el proceso de recepción de datos fue temporizado por hardware, los datos recibidos debían ser leídos antes de que lleguen los siguientes, de lo contrario se produciría un error. De la misma forma durante el proceso de

transmisión los datos a enviar debían ser escritos a tiempo, es decir antes del momento en que deben ser enviados, sino un error sería producido.

La función *osSioBufferGet()* de la API del SIO, se usó mediante la estructura interna del canal receptor o transmisor, para obtener un puntero al búfer de datos recibido o por enviar, respectivamente. Después de que los datos eran leídos o escritos dependiendo del tipo de búfer, usando la función *osSioBufferPut()* de la API del SIO, se le entregaba el búfer y se le hacía saber al driver de que puede usarlo. En el caso de que un error en el proceso de recepción o de transmisión llegase a ocurrir, la rutina de servicio de interrupción de error era llamada [1].

7.3 Archivos del proyecto

Los archivos del proyecto desarrollado están agrupados en tres grupos:

- Programa del sistema ANC.
- Archivos de configuración e inicialización del RTOS SmartDSP.
- Librerías del kernel y drivers del RTOS SmartDSP.

Todo el programa del sistema ANC se encuentra en el archivo fuente *msc711x_app.c*, el cual contiene la función principal *main()*, la función de inicialización de la aplicación, las rutinas de servicio de interrupción de recepción de datos y de error, y todas las funciones desarrolladas especialmente para el funcionamiento de la aplicación, tal como la función *FIR()* que realiza el filtrado digital FIR y la función LMS que se encarga de la actualización de los coeficientes. En el archivo *cabeceratdm_interleaved_buffers.hse* encuentra la definición de los búferes de los canales intercalados de transmisión y recepción del SIO.

Los archivos de configuración e inicialización del RTOS SmartDSP son los siguientes [1]:

- *os_config.h*: es el archivo de configuración del sistema operativo, el cual es el archivo cabecera más importante. En el mismo se seleccionaron las características y módulos del SmartDSP que se utilizaron en la aplicación.

- *app_config.h*: es el archivo de configuración de la aplicación. En el mismo se configuró cuestiones relacionadas a los canales de la aplicación, tales como su tamaño, el tamaño de sus búferes y la cantidad de canales activos.
- *smartdsp_init.c*: es el archivo fuente de inicialización del SmartDSP. En el mismo se encuentra la definición de la función de inicialización del sistema operativo *osInitialize()* y de la función de inicio del mismo *osStart()*. Entre los archivos cabeceras que incluye, se encuentra el *os_config.c*, de tal manera de que solamente sean inicializados los módulos seleccionados en dicho archivo. Este archivo no necesita ser modificado por el usuario.
- *msc711x_init.c*: es el archivo fuente de inicialización de la arquitectura MSC711x. En el mismo se encuentra la definición de las funciones de inicialización de la arquitectura, las cuales son llamadas por la función de inicialización del SmartDSP. Estas funciones configuran los periféricos del DSP seleccionados en el archivo de configuración *os_config.h*. Este archivo no necesita ser modificado por el usuario.
- *msc711x_config.c*: es el archivo fuente de configuración de la arquitectura MSC711x. El mismo contiene las definiciones de las estructuras de configuración de los distintos periféricos. En este archivo se encuentra la estructura *msc711x_tdm_init_params_t*, la cual se utilizó para configurar los registros del TDM0. Las distintas estructuras son tomadas por las funciones de inicialización de los periféricos del archivo *msc711x_init.c*. Entre los archivos cabecera que incluye, se encuentran los archivos *os_config.h, app_config.h, tdm_interleaved_buffers.h*.

El kernel del SmartDSP se encuentra en la librería compilada *os_msc711x_debug.elb*(versión de depuración o debugging). Mientras que las funciones

para los distintos drivers se encuentran en la librería compilada *os_msc711x_drivers_debug.elb* (versión de depuración o debugging).

7.4 Funciones intrínsecas

Las operaciones matemáticas involucradas en el procesamiento de filtros digitales, son operaciones fraccionarias, es decir realizadas sobre números fraccionarios. Como se ha visto anteriormente, el DSP MSC7116 es un DSP que tiene una arquitectura que está diseñada y optimizada para trabajar con números fraccionarios en formato de coma fija.

Debido a que el lenguaje ANSI C no provee soporte para operaciones de números fraccionarios en ese formato, el compilador del entorno CodeWarrior tiene funciones escritas en C especialmente diseñadas para realizar operaciones fraccionarias de coma fija, a partir de datos definidos como enteros, cuyo contenido es interpretado como un valor fraccional de coma fija (de simple o doble precisión, según corresponda, en complemento a dos y con el punto ubicado como se muestra en la Figura 4.1 del capítulo 4). Dichas funciones son llamadas funciones intrínsecas, las cuales se corresponden con las instrucciones en lenguaje ensamblador del DSP, permitiendo poder optimizar el procesamiento. En otras palabras las funciones intrínsecas son una implementación en C de ciertas instrucciones del lenguaje ensamblador (ADD, MPY, MAC, etc)[36].

El sistema operativo SmartDSP usa la notación *int16_t* para definir datos enteros consigno de 16bits y la notación *int32_t* para los de 32bits. Utilizando estas notaciones se definieron las principales variables del sistema, las cuales luego se utilizaron en las funciones intrínsecas para la realización de los cálculos. Las funciones intrínsecas, se encuentran en el archivo cabecera *prototype.h*, el cual fue necesario incluir para poder utilizarlas.

Las funciones intrínsecas utilizadas a lo largo del programa de la aplicación fueron las siguientes:

- *int16_t c = add(int16_t a, int16_t b)*: suma fraccional de 16bits con saturación.
- *int16_t c = sub(int16_t a, int16_t b)*: resta fraccional de 16bits ($c = a - b$), con saturación.

- $\text{int16_t } c = \text{mult_r}(\text{int16_t } a, \text{int16_t } b)$: multiplicación fraccional de 16 bits con redondeo. Retorna los 16 bits más significativos de los 32 bits resultado de la multiplicación.
- $\text{int16_t } c = \text{round}(\text{int32_t } a)$: redondeo con saturación.
- $\text{int32_t } c = \text{L_mult}(\text{int16_t } a, \text{int16_t } b)$: multiplicación fraccional de 16bits.
- $\text{int16_t } c = \text{extract_h}(\text{int32_t } a)$: retorna los 16bits más significativos de un argumento de 32bits.
- $\text{int16_t } c = \text{div_s}(\text{int16_t } a, \text{int16_t } b)$: división fraccional de 16 bits ($c = a / b$). Ambos argumentos deben ser positivos.

7.5 Programa del sistema ANC

Al momento de comenzar el desarrollo del programa del proyecto se utilizó como guía el siguiente proyecto del tipo “demo” del CodeWarrior para el SmartDSP: *EVM_tdm_codec_demo.mcp*.

En el proyecto la función *main()*, que es la función por donde empieza la ejecución del programa, comenzó inicializando el sistema operativo SmartDSP utilizando la función *osInitialize()*. Esta función inicializó aquellos módulos del sistema operativo que fueron habilitados en el archivo de configuración *os_config.h*.

Paso siguiente fue la inicialización de la aplicación utilizando la función *appInit()*. Dentro de ella se colocaron todas las inicializaciones que requería la aplicación. Por último, la función *main()* terminó llamando la función *osStart()* para comenzar con el funcionamiento del SmartDSP. En este momento las interrupciones fueron habilitadas y pasando como parámetro la tarea de fondo de la aplicación llamada *appBackground()*, la cual consistía en una función de un lazo infinito que no realizaba nada. De esta manera el sistema quedó a la espera de que se produzca alguna interrupción.

Las funciones *osInitialize()*, *appInit()* y *osStart()*, son funciones que retornan un valor de estado del tipo *os_status*, para indicar si se realizó con éxito la función o si hubo un error en la misma. Esto permitió mediante el uso de dos macros del sistema operativo, *OS_ASSERT* y *OS_ASSERT_COND()*, detener la ejecución del programa cuando un error

aparece. OS_ASSERT detiene el programa incondicionalmente, mientras que OS_ASSERT_COND() lo hace si la condición que tiene como parámetro es no verdadera. Estas macros fueron también utilizadas en ciertas partes del programa para verificar que no hubiese errores allí.

Dentro de la función *appInit()*, un dispositivo SIO representando al periférico TDM0 fue abierto y configurado, para luego también abrir y configurar los canales intercalados SIO transmisor y receptor que realizaran las operaciones de entrada y salida de datos.

Además, en la función de inicialización de la aplicación fueron generadas las señales de clock para el CODEC, mediante las funciones de inicialización *initGPIO()*, *initsEvents()* e *initTimerB()*, dependiendo de la frecuencia de muestreo seleccionada (8kHz para esta aplicación).

La función *initGPIO()* configuró los pines EVENT0 y EVENT1 para que puedan manejar la señal del oscilador de 8,192MHz y la señal de clock MCLK respectivamente. La función *initsEvents()* configuró el puerto de eventos del DSP para que dicha señal del oscilador sea la entrada del timer B y además sea sacada al pin EVENT1 sin dividirse por el timer A. De esta manera la señal MCLK es igual a 8,192MHz y el timer B pudo ser configurado para proveer la señal de clock al periférico TDM.

Por último la función *initTimerB()*, dependiendo de la frecuencia de muestreo seleccionada, configuró el timer B para que entregue al TDM su señal de clock para su funcionamiento, el cual luego provee de las señales SCLK y LRCK al CODEC. La Figura 7.3 muestra los diagramas de flujo de las funciones *main()* y *appInit()*.

El sistema utilizó solamente la interrupción de recepción de datos del dispositivo SIO para funcionar. Cada vez que se recibían ambos datos de los canales de la aplicación, el driver interrumpía entregando los mismos. Como estos datos eran entregados en un búfer de 8 bits y los mismos en realidad eran de 16 bits (resolución del conversor A/D), fue necesario agruparlos correctamente en datos de 16 bits antes de su procesamiento.

Una vez procesados y obtenidos los datos de salida, estos últimos eran puestos en el búfer de transmisión para que sean transmitidos. Una vez en la rutina de servicio de la interrupción de recepción, lo primero que se realizó fue la identificación de los 4 caminos secundarios presentes en el sistema. Esta identificación duró unos pocos segundos hasta

que se realizó la convergencia del proceso adaptativo y se obtuvo el modelo de cada camino.

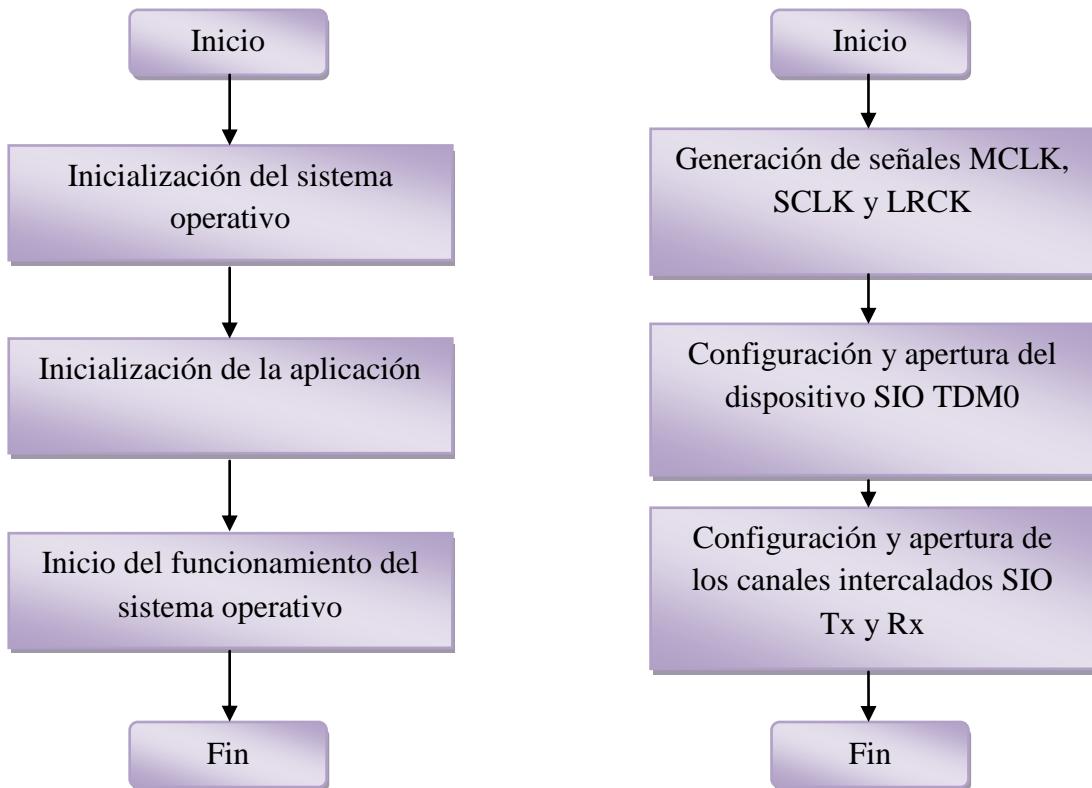


Figura 7.3: Diagramas de flujo de la función *main* (izquierda) y de la función de inicialización de la aplicación *appInit* (derecha).

Finalizada la identificación, comenzó el funcionamiento del sistema ANC realimentado adaptativo multicanal.

La Figura 7.4 muestra el diagrama de flujo de la rutina de servicio de la interrupción de recepción de datos

La identificación de los caminos secundarios es llevada a cabo para los cuatro canales de la aplicación con un esquema identificador de planta como el de la Figura 7.5

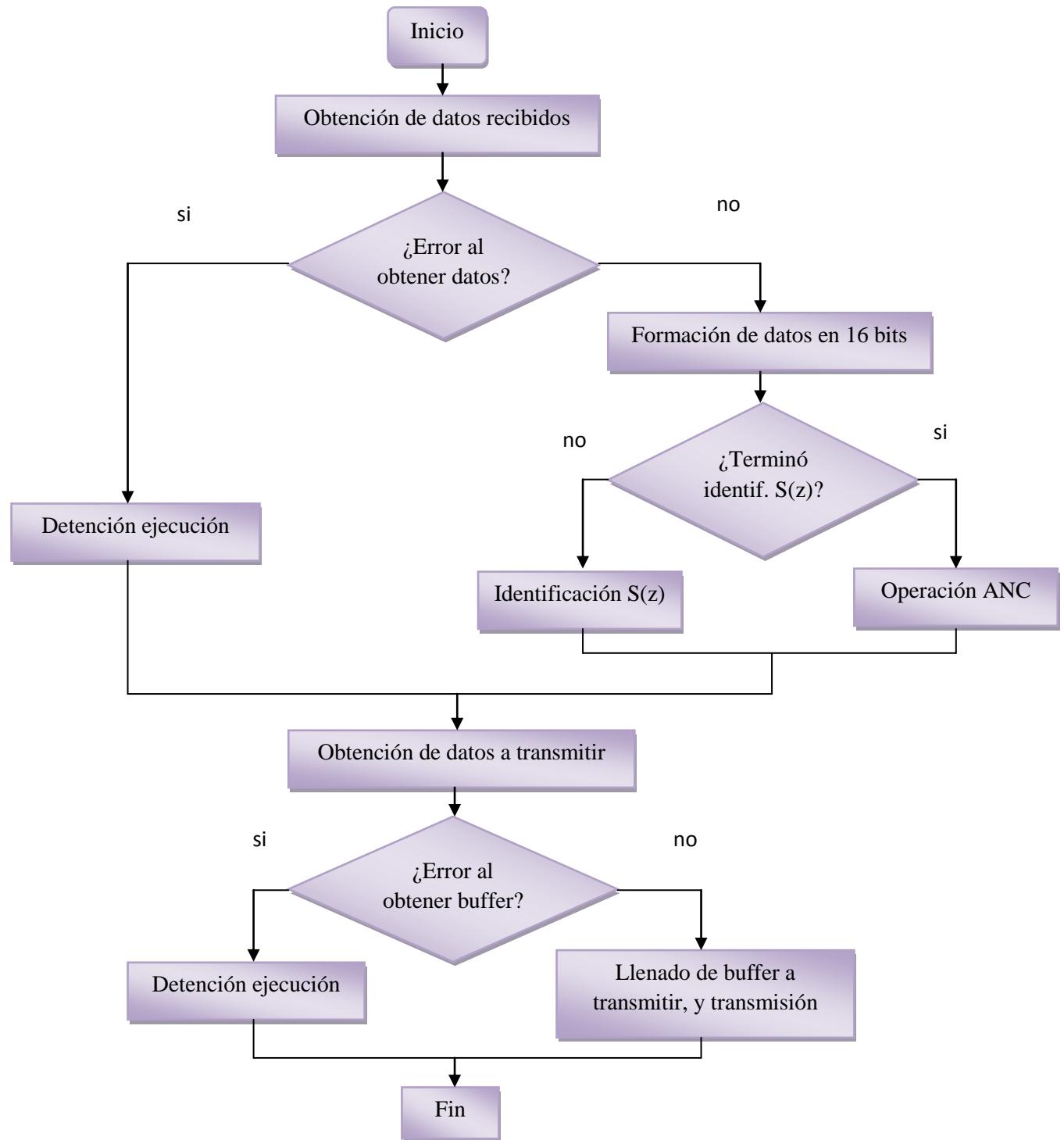


Figura 7.4: Diagrama de la rutina de recepción de datos

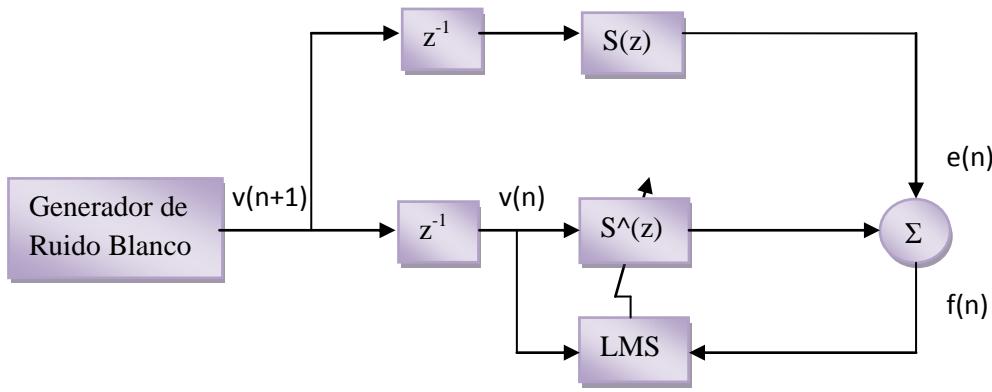


Figura 7.5: Identificación de un camino secundario

Para identificar los caminos secundarios se generó ruido blanco y se calculó el error f como la diferencia entre la señal a la salida de la planta, y la señal a la salida del filtro adaptativo. Con este error f se actualizaron los coeficientes del filtro durante unos segundos hasta que el mismo no presentó variaciones significativas.

Una vez identificados los caminos secundarios, el sistema ANC realimentado adaptativo utilizó el modelo obtenido y comenzó su funcionamiento. En la Figura 7.6 se puede observar el sistema empleando el algoritmo MFxLMS,

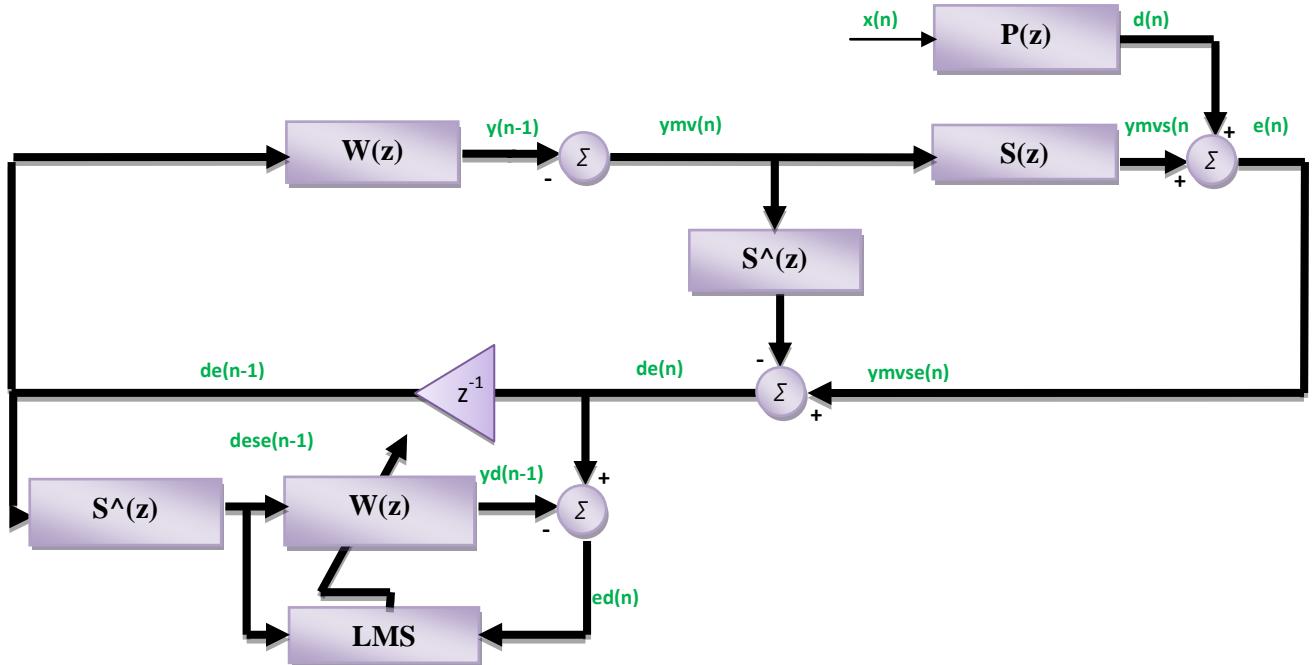


Figura 7.6: Sistema ANC con algoritmo MFxLMS

7.6 Optimizaciones

A lo largo del desarrollo del proyecto fue necesario realizar una serie de optimizaciones del programa desarrollado, a modo de obtener resultados satisfactorios para la aplicación. Para realizar un seguimiento de la necesidad de implementar dichas optimizaciones, se recomienda consultar los resultados presentados en el capítulo 8.

Inicialmente se diseñó el programa para que opere con buffers circulares. Luego se aumentó la precisión del algoritmo LMS para la actualización de los coeficientes W , para mejorar las atenuaciones obtenidas al operar con la señal de ruido del motor en el sistema monocanal. Luego se replanteó la forma en que se definían los filtros W y se operaba con ellos, de forma tal que se pueda aprovechar el funcionamiento paralelo de las cuatro ALU del DSP en el sistema multicanal. Finalmente se implementó el algoritmo LMS con actualización parcial de los coeficientes.

En las secciones siguientes se presentan en detalle las distintas etapas de optimización nombradas.

7.6.1 Buffer circular

A modo de lograr la mayor velocidad posible para las operaciones de filtrado y de actualización de los coeficientes, se desarrollaron las funciones FIR16asm(), FIR32asm(), LMS16asm() y LMS32asm() en lenguaje ensamblador. El concepto utilizado para lograr la mayor velocidad posible en el realizado de las operaciones nombradas fue el de buffer circular, también conocido como direccionamiento modular.

Para llevar a cabo esto se definió en un registro específico (registro M01) la dirección de base o inicio del buffer, y en un segundo registro se especificó el tamaño del buffer. Finalmente se configuró el registro de control de direccionamiento (registro MCTL) indicando al procesador que el modo de direccionamiento a utilizar es el de direccionamiento modular (módulo addressing).

De esta manera, tanto para el filtrado FIR como para la actualización de los coeficientes de los filtros W (procesos que requieren valores de entradas actuales y anteriores) se utilizaron punteros que avanzaron a través de cada uno de los buffer circulares correspondientes. Cuando cada uno de estos punteros llegaba a la dirección final

del buffer, debido a que el registro de control de direccionamiento se encuentra configurado para operar en forma modular, la siguiente dirección a la que apuntaban era al principio del buffer.

Al manejarse con punteros que están configurados para avanzar en la forma correspondiente en cada caso, se evitó la penalidad del tiempo necesario para el reacomodamiento de datos para el cálculo de la salida de los filtros y la actualización de los mismos, lo cual implicaría un costo computacional excesivo para esta aplicación. A su vez se evitó también cualquier tipo de instrucción para verificar si se llegó al final del buffer.

7.6.2 Precisión

A modo de facilitar la comprensión de los siguientes planteos, se presentará nuevamente el diagrama del sistema de control utilizado en la Figura 7.6.

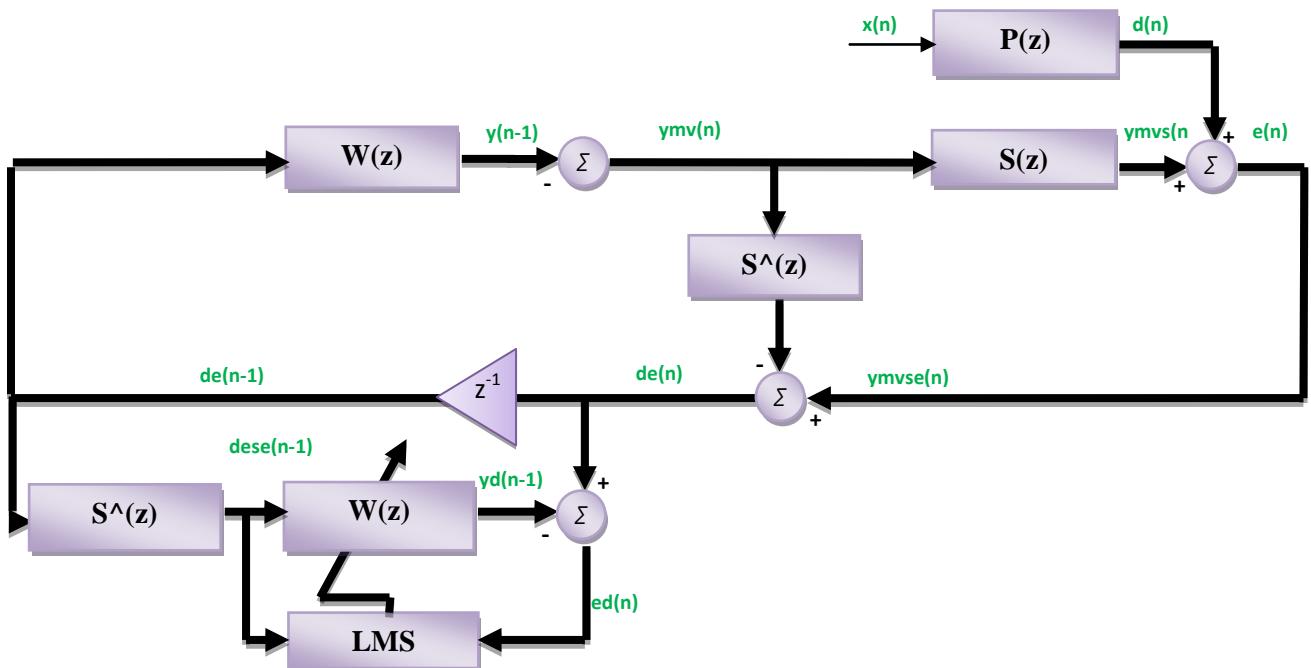


Figura 7.6: Sistema ANC con algoritmo MFxLMS

La primera implementación del sistema operaba con variables de 16 bits, tanto para los filtros de estimación del camino secundario S como para los filtros de control W

en la Figura 7.6. Esto permitía obtener resultados aceptables en el sistema monocanal utilizando como señal de ruido un tono puro senoidal, sin embargo no ocurría lo mismo con la señal de ruido del motor (tema que será tratado con mayor detalle en la sección 8).

7.6.2.1 Filtros de control W

En base a las pobres atenuaciones del ruido del motor se hizo necesario aumentar la precisión con la que se actualizaban los coeficientes de los filtros W . Así la precisión de la mayoría de las variables involucradas en la realización de la identificación de los caminos secundarios y el funcionamiento del sistema ANC continuó siendo de 16 bits, pero se fijó una precisión de 32 bits (doble precisión) en los coeficientes de los filtros de control W para permitir que los mismos converjan de manera más exacta a la solución de Wiener.

Esto introdujo una mejora importante en el desempeño del sistema en cuanto a las atenuaciones conseguidas, lo cual se verifica en la sección 8.1.2.1.

7.6.2.2 Filtros de estimación de caminos secundarios S_e

Para el caso de los coeficientes de los modelos de los caminos secundarios S_e se utilizó en cambio una precisión de 16 bits, ya que usar doble precisión en los mismos aumentaba la demanda computacional, sin mostrar una mayor incidencia en el desempeño del sistema. La razón de este último punto radicaría en que la exactitud del modelo afecta más a la estabilidad, y utilizar 16 bits sería suficiente para que el sistema sea estable [1]. Pequeñas inexactitudes de S_e terminan siendo compensados por W .

7.6.2.3 Producto $\mu \cdot e$ en filtros W

Tanto la variable μ como e son de simple precisión (16 bits), por lo tanto al realizar la multiplicación de ellas el resultado obtenido es de doble precisión (32 bits). Utilizar 16 bits de precisión para alojar dicho término de actualización de los coeficientes del filtro de control W , permitiría realizar de manera muy eficiente el proceso de

adaptación, pero la detendría prematuramente [1]. Si sólo son tomados en cuenta los 16 bits más significativos del producto, la adaptación se detendría en el momento en que estos dejen de variar. Cuando el error es lo suficientemente pequeño el resultado de dicho producto solo afecta los 16 bits menos significativos del resultado, por lo tanto se estaría ignorando dicha información esencial para el desempeño del sistema

Como solución se planteó utilizar para dicho producto la precisión del acumulador de la unidad MAC del DSP, la cual es de doble precisión. Como el resultado de este producto es constante en el momento de actualizar los coeficientes, lo que se hizo fue almacenarlo en memoria e ir utilizándolo a medida que se iba realizando el producto con cada uno de los valores de la señal x' .

Al utilizar 32 bits para almacenar el producto se tuvo que realizar luego multiplicaciones de 32 por 16 bits en la actualización de los coeficientes, incrementando el costo computacional notablemente.

7.6.2.4 Producto $\mu \cdot e$ en filtros S_e

Para el caso del proceso de adaptación de los modelos de los caminos secundarios S_e , no fue necesario utilizar doble precisión en el producto entre el paso de adaptación μ y la señal de error f , no realizándose multiplicaciones de 32 por 16 bits ni aumentándose por consiguiente la demanda computacional.

7.6.2.5 Funciones en lenguaje ensamblador

Para poder realizar los filtros digitales FIR y la actualización de los coeficientes de estos mediante el algoritmo LMS, se diseñaron funciones en lenguaje ensamblador para cada una de estas cosas.

Las funciones *FIR16asm()* y *FIR32asm()* realizan el filtrado FIR de estructura transversal, y las funciones *LMS16asm()* y *LMS32asm()* el algoritmo LMS. La diferencia entre las funciones que realizan el filtrado FIR, está en que la función *FIR16asm()* es utilizada cuando los coeficientes son de 16 bits, mientras que la función *FIR32asm()* cuando son de 32 bits. Sin embargo en esta última solamente son usados los 16 bits más

significativos de los coeficientes para realizar el filtrado, debido a que no es necesario usar los 32 bits.

Para el caso de las funciones que realizaron el algoritmo LMS, la función *LMS16asm()* fue utilizada para la actualización de los coeficientes de 16 bits del modelo del camino secundario, y la misma utilizó también 16 bits para la precisión del producto entre el paso de adaptación y la señal de error. En cambio la función *LMS32asm()* utilizó 32 bits de precisión en dicho producto y realizó el algoritmo LMS de doble precisión.

7.6.3 Operación de ALUs en paralelo

El núcleo SC1400 del DSP MSC7116 contiene cuatro ALUs, las cuales pueden funcionar realizando procesamientos en forma paralela, distribuyéndose la carga computacional. Esto implica que en un mismo ciclo de instrucción pueden realizarse cuatro operaciones simultáneas de multiplicación y acumulación (MAC), lo cual utilizado en conjunto con la instrucción MOVE.4F permite optimizar en gran manera el código.

Para poder utilizar la instrucción MOVE.4F, la dirección de memoria de los coeficientes *de los filtros* y de los valores de la señal x debían ser un múltiplo de 8. Para asegurar esto se utilizó la directiva de precompilación `#pragma align 8`, la cual coloca ciertas secciones del código en direcciones que sean múltiplos del valor especificado.

Si bien la instrucción MOVE.4F permitía traer desde memoria hasta los registros de la ALU cuatro valores de entrada x , o cuatro coeficientes de un filtro (W o S_e) según corresponda, la misma no se veía afectada por el valor del registro de control de direccionamiento MCTL. Esto implica que si bien dicho registro fue configurado para operar con buffers circulares, cuando la instrucción llegaba hasta el final del buffer se producía un error en el direccionamiento.

Para poder solucionar este conflicto se optó por definir los filtros y las variables de entrada de una manera poco convencional, pero altamente eficiente para esta aplicación. En vez de definir los cuatro filtros de control W como un arreglo bidimensional cuyas dimensiones son [4] y [L_w], lo cual se vería representado en el siguiente esquema:

$$W[4][Lw] = \left\{ \begin{array}{l} \{W_{1,1}, W_{1,2}, W_{1,3}, W_{1,4}, W_{1,5}, \dots, W_{1,Lw}\}, \\ \{W_{2,1}, W_{2,2}, W_{2,3}, W_{2,4}, W_{2,5}, \dots, W_{2,Lw}\}, \\ \{W_{3,1}, W_{3,2}, W_{3,3}, W_{3,4}, W_{3,5}, \dots, W_{3,Lw}\}, \\ \{W_{4,1}, W_{4,2}, W_{4,3}, W_{4,4}, W_{4,5}, \dots, W_{4,Lw}\} \end{array} \right\} \quad (7.1)$$

Se decidió definir a dichos filtros como arreglos bidimensionales de dimensiones $[L_w]$ y $[4]$

$$W[Lw][4] = \left\{ \begin{array}{l} \{W_{1,1}, W_{2,1}, W_{3,1}, W_{4,1}\}, \\ \{W_{1,2}, W_{2,2}, W_{3,2}, W_{4,2}\}, \\ \{W_{1,3}, W_{2,3}, W_{3,3}, W_{4,3}\}, \\ \{W_{1,4}, W_{2,4}, W_{3,4}, W_{4,4}\}, \\ \{W_{1,5}, W_{2,5}, W_{3,5}, W_{4,5}\}, \\ \vdots \\ \vdots \\ \{W_{1,Lw}, W_{2,Lw}, W_{3,Lw}, W_{4,Lw}\}, \end{array} \right\} \quad (7.2)$$

Al definir los filtros inicialmente de esta manera, fue posible optimizar el código de tal manera que utilice las cuatro ALU en paralelo y respetar al mismo tiempo el direccionamiento circular con la instrucción MOVE.4F. De esta forma en cada ciclo de 4 multiplicaciones y acumulaciones no se realizaron cuatro MAC de un filtro en particular, sino que se realizó una única MAC de cada uno de los cuatro filtros. Esto fue aplicado tanto en las operaciones de filtrado, como en las actualizaciones de los coeficientes.

A continuación se presenta la instrucción presente dentro del bucle en la función FIR16asm() en donde se observa lo planteado anteriormente

MOVE.4F (r0)+,d0:d1:d2:d3	MOVE.4F (r1)+,d4:d5:d6:d7
MAC d0,d4,d8	MAC d1,d5,d9
MAC d2,d6,d10	MAC d3,d7,d11

El puntero r(0) apunta a los valores de entradas actuales y anteriores, y el puntero r(1) apunta a los coeficientes del filtro correspondiente. Los registros d0, d1, d2 y d3 almacenan dichas entradas, mientras que los registros d4, d5, d6 y d7 almacenan los coeficientes. Finalmente los resultados de las sucesivas multiplicaciones y acumulaciones son almacenados en los registros d8, d9, d10 y d11.

Las dos instrucciones MOVE.4F en conjunto con las cuatro instrucciones MAC, fueron todas realizadas en un único ciclo de instrucción, logrando el máximo grado de optimización posible. Luego se muestran las instrucciones correspondientes al algoritmo LMS de precisión simple, en donde debe recordarse que cada coeficiente debía ser traído desde memoria hasta los registros de la ALU, calcular su actualización y ser guardado nuevamente en memoria:

MOVE.4F (r0)+,d0:d1:d2:d3	MOVE.4F (r1),d4:d5:d6:d7		
MAC d0,d8,d4	MAC d1,d9,d5	MAC d2,d10,d6	MAC d3,d11,d7
MOVE.4F d4:d5:d6:d7,(r1)			

Si bien las cuatro MAC son realizadas en un único ciclo de instrucción, se requirió en este caso dos ciclos más de instrucción para traer y guardar los datos en memoria.

7.6.4 Actualización parcial de coeficientes

Tal como se verá en la sección 8.2.1, la cantidad de filtros necesarios para la operación del sistema multicanal, en conjunto con la necesidad de utilizar doble precisión en la actualización de sus coeficientes, implicaban un gran número de operaciones a realizar entre muestra y muestra. Esto trajo como consecuencia una reducción en la máxima longitud de los filtros de control W posible de ser implementada, lo cual trajo aparejado un deterioro significativo en los resultados obtenidos. Para superar este inconveniente, se planteó la implementación del algoritmo LMS con actualización parcial de los coeficientes.

Este algoritmo implica que solo una fracción de los coeficientes de los filtros W sean actualizados en cada período de muestreo, logrando reducir el costo computacional a

cambio de una menor velocidad de convergencia [29]. De esta manera se logró aumentar en gran medida la longitud de los filtros de control, permitiendo obtener resultados óptimos en el sistema ANC multicanal cancelando el ruido correspondiente al motor de combustión interna.

CAPÍTULO 8: Mediciones y resultados

En este capítulo se presentan los resultados de las mediciones del sistema de control activo de ruido implementado efectivamente con un DSP comercial. Se siguió el mismo ciclo de pruebas llevado a cabo en las simulaciones (capítulo 5), a fin de realizar una optimización gradual del sistema real. En primer lugar se muestran los resultados del sistema funcionando con un único canal, tanto para una señal de ruido primario senoidal como para la señal equivalente al ruido de un motor. Luego se presentan los resultados del sistema multicanal operando con los dos sensores y las dos fuentes secundarias, para la señal senoidal y la señal del motor.

8.1 Sistema ANC monocanal

Se implementó el sistema de control activo de ruido para el interior del recinto en el módulo de evaluación del DSP StarCore1400 MSC7116EVM. Al igual que en las simulaciones, se realizó en primer lugar un aprendizaje del camino secundario, y posteriormente se procedió a cancelar el ruido primario generado por la fuente situada fuera de la cabina. El código del programa fue desarrollado en lenguaje C, mientras que las funciones correspondientes al filtrado FIR y al algoritmo LMS para la actualización de los coeficientes fueron desarrolladas en lenguaje ensamblador. En todos los casos las señales de error fueron grabadas simultáneas a la operación del sistema utilizando una placa de adquisición de sonido Presonus Audiobox 44VSL, y se realizó posteriormente el procesamiento de señal necesario para evaluar el desempeño del sistema. Se verá como en esta primera implementación monocanal, la precisión utilizada en los algoritmos LMS para la actualización de los coeficientes de los filtros de control W fue de 16 bits para la señal de ruido senoidal, ya que ello permitió obtener resultados satisfactorios. Sin embargo al operar el sistema con la señal de ruido del motor, se hizo necesario aumentar la precisión de los algoritmos LMS a 32 bits.

8.1.1 Señal senoidal

El primer caso que se evaluó fue utilizando una señal senoidal de frecuencia 100 Hz como señal primaria. Se implementó el sistema, y se ajustaron todos los parámetros del mismo (pasos de adaptación y largo de filtros) de manera de obtener los mejores resultados posibles. Se especifica en la Tabla 8.1 la configuración final de los parámetros, y en la Tabla 8.2 y Figura 8.1 los resultados obtenidos.

Tabla 8.1: Configuración de parámetros

Parámetros del sistema	
Largo filtro de control W	320
Paso de adaptación filtro W	6.10 e-4
Largo filtro de control S_e	280
Paso de adaptación filtro S_e	4.88 e-3

Tabla 8.2: Atenuación en sistema monocanal con señal senoidal

Número de armónico	Frecuencia [Hz]	Atenuación sonora [dB]
1	100	42

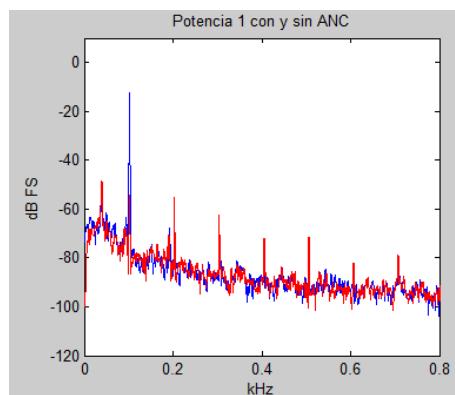


Figura 8.1: PSD en sistema con ANC (rojo) y sin ANC (azul)

8.1.2 Señal de ruido del motor

El siguiente paso fue evaluar el desempeño del sistema utilizando como señal primaria de ruido el sonido sintetizado correspondiente a un motor de 4 tiempos y 4 cilindros operando a 2500rpm. En la Tabla 8.3 se presenta la configuración utilizada, y en la Tabla 8.4 los resultados obtenidos en los primeros armónicos de la frecuencia fundamental.

Tabla 8.3: Configuración de parámetros

Parámetros del sistema	
Largo filtro de control W	640
Paso de adaptación filtro W	9.15 e-5
Largo filtro de control S_e	640
Paso de adaptación filtro S_e	3.05 e-3

Tabla 8.4: Atenuaciones en los armónicos del ruido a cancelar

Número de armónico	Frecuencia [Hz]	Atenuación sonora [dB]
1	83	17
2	166	16
3	249	16
4	332	17
5	415	3
6	498	0
7	581	3

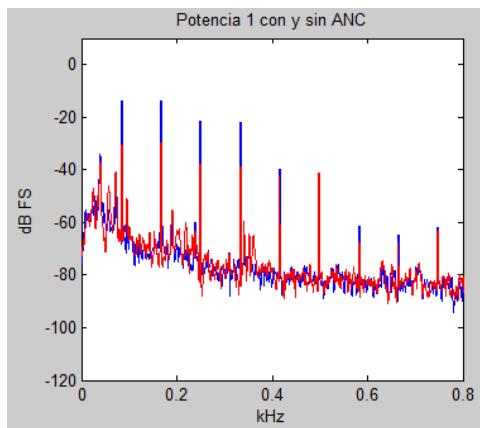


Figura 8.3: PSD en sistema con ANC (rojo) y sin ANC (azul)

Los resultados obtenidos para la señal del motor fueron notablemente inferiores a los obtenidos para la señal senoidal, como era de esperarse. Debido a esto, se optó por optimizar el sistema de modo de mejorar el desempeño del mismo.

8.1.2.1 Algoritmo LMS de doble precisión

La primera modificación implementada para mejorar el desempeño del sistema fue aumentar la precisión utilizada en los cálculos de las actualizaciones de los coeficientes de los filtros en los algoritmos LMS. A pesar de actualizar los coeficientes con 32 bits de precisión, el filtrado FIR se continuó llevando a cabo con 16 bits, utilizando los 16 bits más significativos de los coeficientes del filtro. En la Tabla 8.6 y Figura 8.4 se presentan los resultados obtenidos.

Tabla 8.5: Configuración de parámetros

Parámetros del sistema	
Largo filtro de control W	580
Paso de adaptación filtro W	6.10 e-4
Largo filtro de control S_e	640
Paso de adaptación filtro S_e	3.05 e-3

Número de armónico	Frecuencia [Hz]	Atenuación sonora [dB]
1	83	60
2	166	49
3	249	20
4	332	24
5	415	5
6	498	7
7	581	2

Tabla 8.6: Atenuaciones en los armónicos del ruido a cancelar

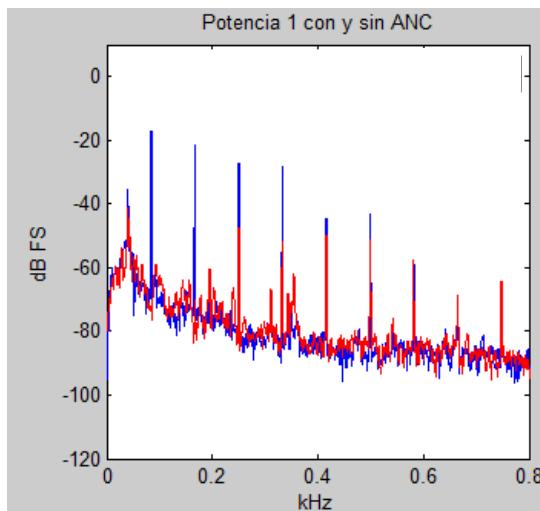


Figura 8.4: PSD en sistema con ANC (rojo) y sin ANC (azul)

Si bien los resultados mejoraron notablemente en las dos primeras armónicas de la frecuencia fundamental, los cambios no fueron de la misma magnitud en las siguientes armónicas. En base a esto, se decidió implementar la versión normalizada de los algoritmos.

8.1.2.2 Algoritmo LMS normalizado de doble precisión

En la Tabla 8.8 y Figura 8.6 se presentan los resultados obtenidos al utilizar el algoritmo normalizado, presentado en la sección 2.5.7, lo cual hizo necesario la determinación del alfa y beta (ecuación 2.22) óptimos para esta aplicación.

Tabla 8.7: Configuración de parámetros

Parámetros del sistema	
Largo filtro de control W	580
α algoritmo normalizado	3.05 e-4
β algoritmo normalizado	1.50 e-2
Largo filtro de control S_e	640
Paso de adaptación filtro S_e	3.05 e-3

Tabla 8.8: Atenuaciones en los armónicos del ruido a cancelar

Número de armónico	Frecuencia [Hz]	Atenuación sonora [dB]
1	83	55
2	166	48
3	249	66
4	332	61
5	415	11
6	498	20
7	581	0

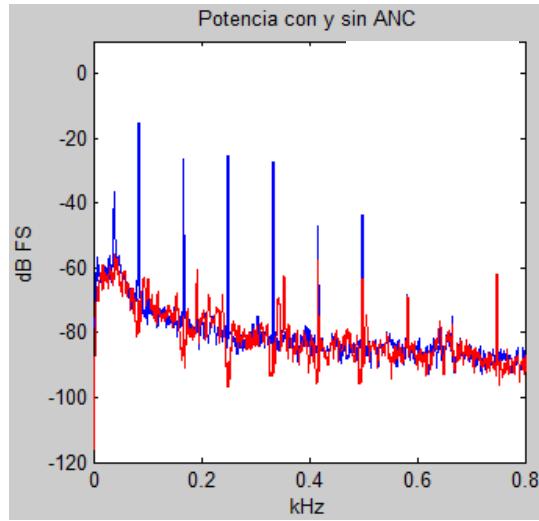


Figura 8.6: PSD en sistema con ANC (rojo) y sin ANC (azul)

8.1.3 Análisis de filtros

Una vez que el sistema se estabilizó con los resultados presentados en la sección anterior, se pausó el funcionamiento del mismo y se procedió a analizar los coeficientes de los filtros S_e (estimación del camino secundario) y W . En las figuras 8.7 y 8.8 se presentan los mismos.

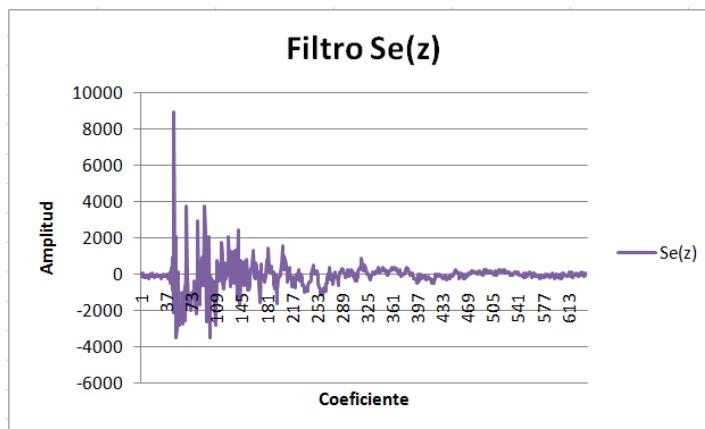


Figura 8.7: Filtro de camino secundario estimado S_e

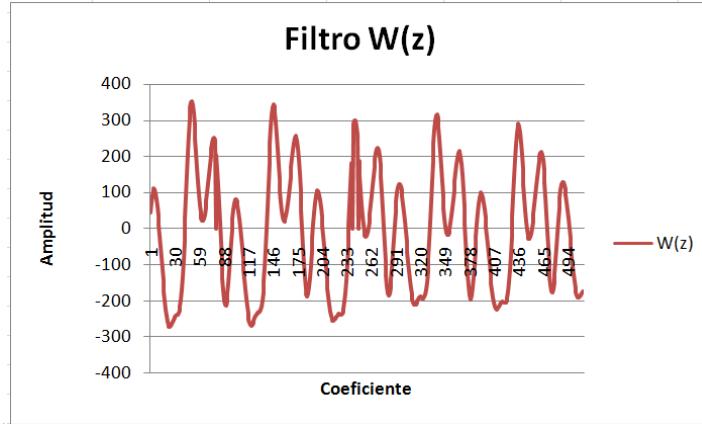


Figura 8.8: Filtro de control W

Analizando el filtro S_e , se observó que el largo del mismo fue suficiente para estimar el camino secundario. Además, puede decirse que en caso de que fuera necesario utilizar filtros de menor largo, sería muy probable que ello no implique disminuciones significativas en el desempeño del sistema. Esto se debe a que la mayor parte de la energía de la respuesta impulsiva del camino secundario se concentra en el primer cuarto de la estimación realizada.

En cuanto al filtro de control W , pudo observarse en el mismo un patrón que se repite 5 veces. Esto implica que la cantidad de coeficientes utilizado es mucho mayor (5 veces) que la mínima estrictamente hablando. Por lo tanto, en caso de reducir significativamente la longitud de dicho filtro, sería probable que el sistema funcione sin disminuir notablemente su desempeño. Para verificar esto, se implementó el sistema de control monocanal utilizando un filtro W de 240 coeficientes. En la Tabla 8.9 y 8.10 y en la Figura 8.9, se presentan la configuración y los resultados obtenidos.

Tabla 8.9: Configuración de parámetros

Parámetros del sistema	
Largo filtro de control W	240
α algoritmo normalizado	1.50 e-4
β algoritmo normalizado	1.50 e-2
Largo filtro de control S_e	640
Paso de adaptación filtro S_e	3.05 e-3

Número de armónico	Frecuencia [Hz]	Atenuación sonora [dB]
1	83	53
2	166	38
3	249	53
4	332	41
5	415	14
6	498	13
7	581	3

Tabla 8.10: Atenuaciones obtenidas en los armónicos del ruido a cancelar

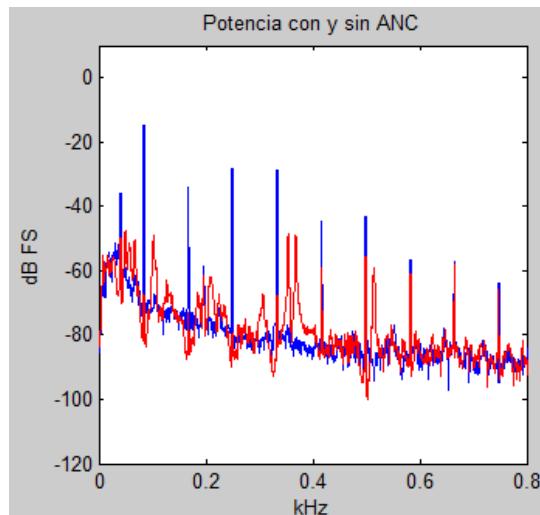


Figura 8.9: PSD en sistema con ANC (rojo) y sin ANC (azul)

En la Figura 8.10 se muestran los valores de los coeficientes del filtro W una vez que el sistema se encontraba funcionando en régimen.

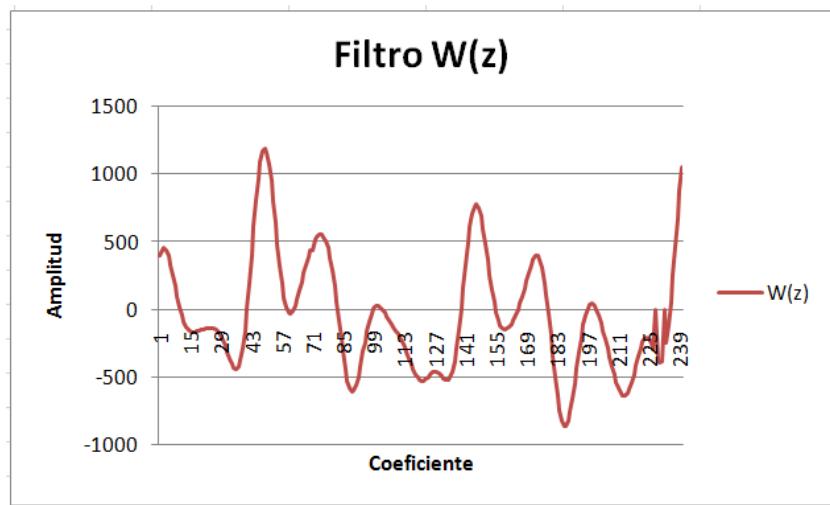


Figura 8.10: Filtro de control W

A pesar de haber reducido la longitud del filtro de control a dos tercios del largo anterior, los resultados continuaron siendo satisfactorios para esta aplicación. Sin embargo, es necesario destacar que se observaron componentes espectrales en la señal residual que no se encontraban presentes en el ruido primario a cancelar. En el caso de reducir aún más la longitud de los filtros, este hecho podría llegar a degradar significativamente el desempeño del sistema haciendo que el mismo ya no resulte adecuado para el caso bajo análisis.

A la hora de implementar la versión multicanal del sistema, la cantidad de filtros en juego aumentó notablemente, lo cual implicó que en el mismo tiempo disponible entre muestra y muestra debieron realizarse un número mayor de filtrado FIR y de algoritmos LMS. En base a esto debió reducirse la longitud de los filtros y optimizarse los algoritmos de filtrado y actualización de coeficientes, de forma tal que la máxima longitud de los filtros que sean posible implementar en función del poder de cómputo del DSP permitan alcanzar resultados satisfactorios para la aplicación.

8.2 Sistema ANC multicanal

En esta sección se presentan los resultados del sistema operando con dos sensores de error y dos fuentes secundarias, para la señal de ruido del motor. Se implementó el sistema

multicanal con los algoritmos LMS de doble precisión, utilizando la versión normalizada de los mismos.

8.2.1 Implementación inicial

Debido a la cantidad de filtrados FIR y de actualización de coeficientes que se deben realizar entre muestra y muestra por el aumento del número de canales, la longitud de los filtros W y S_e debió reducirse notablemente al implementar el sistema multicanal. Esto trajo aparejado un notable deterioro de las atenuaciones obtenidas. En la Tabla 8.12 y Figura 8.11 se presentan los resultados del sistema multicanal con el algoritmo LMS de doble precisión

Tabla 8.11: Configuración de parámetros

Parámetros del sistema	
Largo filtro de control W	160
α algoritmo normalizado	1.50 e-4
β algoritmo normalizado	1.50 e-2
Largo filtro de control S_e	400
Paso de adaptación filtro S_e	3.05 e-3

Tabla 8.12: Atenuaciones en las componentes armónicas del ruido a cancelar

Número de armónico	Frecuencia [Hz]	Atenuación sonora [dB]
1	83	30
2	166	26
3	249	8
4	332	4
5	415	0
6	498	3
7	581	0

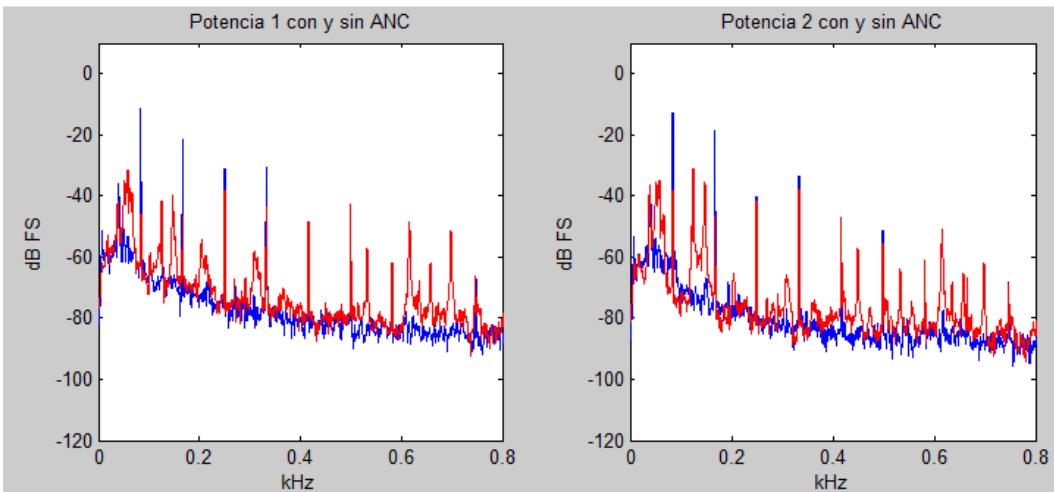


Figura 8.11: PSD en los sensores de error con ANC (rojo) y sin ANC (azul)

En los resultados presentados no solo puede apreciarse la disminución de las atenuaciones obtenidas en los distintos armónicos de la señal de ruido, sino que también se observan componentes de ruido generadas por el sistema que no corresponden a la señal de ruido primaria. Esto demuestra que el largo de los filtros que fue posible implementar en el sistema multicanal no fue suficiente para obtener resultados satisfactorios en esta aplicación. En base a esto se hizo necesario optimizar aún más el funcionamiento del sistema, de modo que sea posible implementar filtros de control W de mayor longitud para lograr una efectiva cancelación de la señal de ruido del motor.

8.2.2 Algoritmo LMS con actualización parcial de coeficientes

Debido a que en el tiempo disponible entre muestra y muestra sólo fue posible realizar el filtrado FIR y la actualización para filtros W de 160 coeficientes, se propuso la implementación del algoritmo con actualización parcial de los coeficientes propuesto por [2]. La idea fundamental de este algoritmo consiste en que en cada período de muestreo únicamente se actualicen una porción de los coeficientes de los filtros, en vez de la totalidad de los mismos.

De esta manera es posible reducir la carga computacional posibilitando la utilización de filtros de mayor longitud, teniendo como contraparte una disminución en la velocidad de convergencia.

Al utilizar este algoritmo se hizo posible aumentar la longitud de los filtros W de 160 a 500 coeficientes produciendo una mejora significativa en los resultados, alcanzando los mismos valores satisfactorios para esta aplicación. En la Tabla 8.14 y Figura 8.12 se muestran los resultados obtenidos

Tabla 8.13: Configuración de parámetros

Parámetros del sistema	
Largo filtro de control W	500
α algoritmo normalizado	1.50 e-4
β algoritmo normalizado	1.50 e-2
Largo filtro de control S_e	400
Paso de adaptación filtro S_e	3.05 e-3

Tabla 8.14: Atenuaciones obtenidas en los armónicos del ruido a cancelar

Número de armónico	Frecuencia [Hz]	Atenuación sonora [dB]
1	83	47
2	166	40
3	249	48
4	332	24
5	415	14
6	498	12
7	581	3

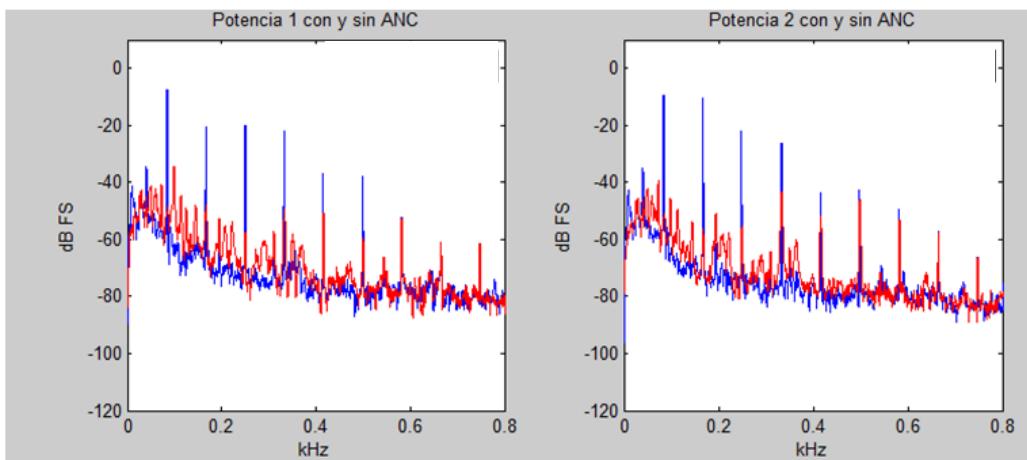


Figura 8.12: PSD en los sensores de error con ANC (rojo) y sin ANC (azul)

CAPÍTULO 9: Conclusiones y trabajos a futuro

En esta sección se exponen las principales conclusiones a las que se arribaron en este proyecto integrador. Posteriormente, se dan las sugerencias para futuras investigaciones para, finalmente, dar a conocer la publicación que surgió a partir de este trabajo.

9.1 Conclusiones

Al comienzo de este proyecto integrador se planteó desarrollar e implementar un sistema multicanal de control activo de ruido acústico periódico para el interior de automóviles, el cual debía atenuar los ruidos periódicos de baja frecuencia en tiempo real en una zona determinada del recinto. Luego de todo el trabajo realizado se puede concluir que el objetivo fue cumplido con éxito, logrando muy buenas atenuaciones para los ruidos analizados.

En la etapa de las simulaciones computacionales se determinó que un sistema con dos sensores de error y dos fuentes secundarias fue suficiente para la aplicación. Esto evitó la implementación de un sistema con cuatro fuentes y dos sensores, lo cual hubiera implicado un costo computacional mayor al doble del requerido por el sistema utilizado.

Al implementar la versión monocanal del ANC, se pudo concluir que para cancelar el ruido equivalente al del motor de un auto, fue necesario operar con doble precisión en la adaptación y en los coeficientes de los filtros de control W . A su vez se demostró que los resultados mejoran notablemente al utilizar la versión normalizada del algoritmo LMS.

Además, se llegó a la conclusión de que para poder utilizar el DSP seleccionado en esta aplicación, fue necesario realizar la actualización parcial de los coeficientes de los filtros de control. Esto permitió utilizar los filtros de la longitud necesaria para lograr las atenuaciones deseadas.

Dentro de las limitaciones del sistema implementado, es relevante destacar la incapacidad del mismo para obtener atenuaciones aceptables en frecuencias mayores a 600Hz. Esto se debe, entre otros factores, a la baja frecuencia de muestreo seleccionada en base a los requerimientos de cómputo requeridos por la aplicación. Otro factor que condiciona la operación del sistema es la imposibilidad de cancelar señales de ruido que no

sean periódicas, como por ejemplo el sonido producido por la colisión del viento y la carrocería del vehículo.

Por último, es de suma importancia destacar la diversidad de conceptos aplicados en la realización de este trabajo. La integración de conocimientos de diferentes ramas de la ingeniería como son la física, el análisis matemático, la teoría de señales y sistemas, los sistemas de control, la electrónica analógica y digital, y en particular el procesamiento digital de señales, fue necesaria para la concreción de este proyecto integrador.

9.2 Trabajos a futuro

A partir del presente proyecto, se han planteado propuestas para seguir la línea de estudio, algunas de las cuales pueden ser considerados como trabajos en sí mismos y otras como mejoras a este proyecto integrador.

Un posible proyecto integrador que surge a partir del presente, es el desarrollo e implementación de un sistema ANC multicanal global, en donde el objetivo consiste en atenuar el ruido en todo el interior del recinto en cuestión, y no únicamente en una zona de quietud reducida.

Como mejora de este proyecto se plantea la implementación del sistema con un mayor número de fuentes de control y de sensores de error, de modo que mejoren las atenuaciones obtenidas en conjunto con un aumento del tamaño de la zona de quietud.

Otro posible trabajo a futuro consiste en lograr el funcionamiento del sistema ubicando las fuentes de control en las posiciones donde normalmente se encuentran los altavoces en los automóviles comerciales. A su vez el ANC puede ser implementado con la técnica *beamforming*, de manera tal que la zona de quietud pueda ser generada en una ubicación que no necesariamente sea la de los sensores de error.

Finalmente, se puede implementar un sistema de control activo de ruido del tipo pre alimentado, de forma tal que sea posible cancelar no solo ruidos periódicos, sino cualquier tipo de ruido que esté correlacionado con la señal de referencia.

9.3 Presentación en congreso

Como se describió aquí, parte del trabajo presentado incluyó simulaciones computacionales acerca de la influencia del número y ubicación de fuentes secundarias en el desempeño del sistema ANC. A partir de esto surgió el siguiente trabajo que ha sido presentado en el congreso de mecánica computacional ENIEF 2013.

- Budde L., Rossi R., Ferreyra S. P., “Influencia de número y localización de fuentes secundarias en el desempeño de un sistema multicanal de control activo local de ruido para recintos cerrados” [44].

Bibliografía

- [1] González Vergara F. A., 2011. *Controlador activo adaptativo de ruido acústico periódico para un protector de audición y/o auricular*. Proyecto Integrador, FCEFyN, UNC. Córdoba, Argentina.
- [2] Kuo S. M., Morgan D. R., 1996. *Active Noise Control Systems: Algorithms and DSP Implementations*. Wiley Series in Telecommunications and Signal Processing. New York: Wiley.
- [3] Miyara F., 2004. *Acústica y Sistemas de Sonido*. UNR Editora, Universidad Nacional de Rosario. Rosario, Argentina.
- [4] Puder H., Steffens S.. *Improved noise reduction for hands free car phones utilizing information on vehicle and engine speeds*. Signal Theory, Darmstadt University of Technology. Darmstadt, Alemania.
- [5] Beranek L., 1954. *Acoustics*. McGraw-Hill.
- [6] Isbert A. C., 1998. *Diseño Acústico de Espacios Arquitectónicos*. Edicions de la Universitat Politècnica de Catalunya, SL. ISBN: 84-8301-252-9
- [7] Flores M. D., Martinez M. F., 2012. *Análisis de modos propios en recintos paralelepípedos con distintas dimensiones*. Cátedra de Fundamentos de Acústica y Electroacústica, Universidad Tecnológica Nacional, Facultad Regional Córdoba.
- [8] Kuttruff H., 2000. *Room Acoustics*. Spon Press – Taylor and Francis group. ISBN 0-203-18623-0.
- [9] Kuttruff H., 2007. *Acoustics*. Taylor and Francis group. ISBN 0-203-97089-6

- [10] Hansen, C. H. y Snyder, S. D., 1997. *Active Control of Noise and Vibration*. London: E & FN Spon.
- [11] Elliott, S. J., 2001. *Signal Processing for Active Control*. Signal Processing and its Applications. London: Academic Press.
- [12] Oppenheim, A. y Willsky, A. S., 1983. *Signals and Systems*. London: Prentice Hall, Inc.
- [13] Hsu, H., 1997. *Theory and Problems of Probability, Random Variables, and Random Processes*. New York: McGraw-Hill.
- [14] Papoulis, A. y Pillai, S.U., 2002. *Probability, Random Variables and Stochastic Processes*. 4th ed. New York: McGraw Hill.
- [15] Widrow, B. y Stearns, S. D., 1985. *Adaptive Signal Processing*. New Jersey: Prentice-Hall.
- [16] Widrow, B. y Walach, E., 2008. *Adaptive Inverse Control: A Signal Processing Approach, Reissue Edition*. IEEE Press Series on Power Engineering. New Jersey: Wiley-IEEE Press.
- [17] Kalman, R. E., 1960. *A New Approach to Linear Filtering and Prediction Problems*. Transactions of the ASME–Journal of Basic Engineering, 82 (Series D), 35-45.
- [18] Balakrishnan, A. V., 1984. *Kalman Filtering Theory*. New York: Optimization Software, Inc.
- [19] Haykin, S., 1996. *Adaptive Filter Theory*. Information and System Sciences Series. 3rd ed. New Jersey: Prentice Hall.

- [20] Manolakis, D. G., Ingle, V. K. y Kogon, S. M., 2005. *Statistical and Adaptive Signal Processing: Spectral Estimation, Signal Modeling, Adaptive Filtering, and Array Processing*. Boston-London: Artech House, Inc.
- [21] Elliott S. J. y Nelson P. A., 1993. *Active Noise Control. IEEE Signal Processing Magazine*, 10 (4), 12-35.
- [22] Johansson S., Winberg M., Claesson I.. *Active control of interior noise in a fork lift truck*. Department of Telecommunications and Signal Processing, Blekinge Institute of Technology. SE-372 25 Ronneby, Sweden
- [23] Bolt R. H., 1946. *Note on normal frequency Statistics for rectangular rooms*. Journal of the Acoustical Society of America. Volume 19.
- [24] Cox, D'Antonio, 2001. *Room dimensions for critical listening environments*. AES Convention: 110, pp. 5353.
- [25] Diego M., Gonzalez A., Garcia C.. *On the performance of a local active noise control system*. Universidad Politecnica de Valencia. Valencia, España.
- [26] Barrionuevo D., González J., 2012. *Criterios generales para el diseño de cámaras anecóicas*. Cátedra de Fundamentos de Acústica y Electroacústica, Universidad Tecnológica Nacional, Facultad Regional Córdoba.
- [27] Small R. H. 1973. *Vented-Box loudspeakers system, small-signal analysis, large-signal analysis, synthesis and appendices*. Journal of the Audio Engineering Society, 21:363-372, 438-444, 549-554, 635-639.
- [28] Aguilar G., "Sistema de medición de parámetros de altavoces Thiele-Small", Ing. Electrónica, FRC UTN.

- [29] *Introducción al procesamiento digital de señales*. Departamento de electrónica, automática e informática industrial.ELAI-UPM-DOCÒ001-01. Universidad Politécnica de Madrid. Madrid, España.
- [30] Kuo, S. M., Lee, B. H. y Tian, W., 2006. *Real-Time Digital Signal Processing: Implementations and Applications*. 2nd ed. London: Wiley.
- [31] Martínez Sober M., Serrano López A. J., 2009. *Introducción al procesado digital de señales*. Department d'Enginyeria Electrònica, Escola Tècnica Superior D'Enginyeria. Universidad de Valencia. Valencia, España.
- [32] Gan, W. S. y Kuo, S. M., 2007. *Embedded Signal Processing with the Micro Signal Architecture*. New Jersey: Wiley-IEEE Press.
- [33] *Vehicle Noise and Vibration Control*. Indian Institute of Technology Roorkee.
- [34] SmartDSP OS Reference Manual. Ver. 1.42, 09/2005. Metrowerks.
- [35] SmartDSP OS Drivers Developer Guide. For BIO drivers. Ver. 1.1, 05/2005. Metrowerks.
- [36] Application Note 2441. StarCore SC140 Application Development Tutorial. AN2441. Rev. 0, 2004. Freescale Semiconductors, Inc.
- [37] Bellanger, M. G., 2001. *Adaptive Digital Filters*. Signal Processing and Communications Series. 2nd ed. New York-Basel: Marcel Dekker, Inc.
- [38] Allman-Ward M., Balaam M. P., Williams R.. *Source decomposition for vehicle sound simulation*. Simulation and Test Centre, MSX International, Millbrook, Beds, MK45 2YT. Inglaterra

- [39] Morehouse C., Maier J., Zachwieja T., Bills C.. *Noise reduction for internal combustion engine*. Multidisciplinary Engineering At Rit – Senior Design. Cenco Physics.
- [40] Jerry G., Lilly P. E.. *Engine Exhaust Noise Control*. Technical Committee Sound and Vibration.
- [41] MSC7116 Data Sheet. Low-Cost 16-bit DSP with DDR Controller and 10/100 MbpsEthernet MAC. MSC7116. Rev. 13, 04/2008. Freescale Semiconductor, Inc.
- [42] AK4554 Data Sheet. MS0325-E-01. 08/2005. ASAHI KASEI.
- [43] SmartDSP OS Data Sheet. CodeWarrior for SmartDSP OS. 950-00098. Rev. E, 2010. Freescale Semiconductors.
- [44] Budde L., Rossi R. R., Ferreyra S. P.. “*Influencia de número y localización de fuentes secundarias en el desempeño de un sistema multicanal de control activo local de ruido para recintos cerrados*”. Mecánica Computacional Vol XXXII, págs. 2857-2868. Mendoza, Argentina. 19-22 Noviembre 2013.

Anexo - Código del programa

```
*****
```

Título: Proyecto integrador Ingeniería Electrónica FCEFyN UNC, ANC Multicanal

Autor: Leopoldo Budde

Directores: Ing. Roberto R. Rossi, Ing. Sebastián P. Ferreyra

```
*****
```

// Includes -----

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "smartdsp_os.h"
#include "msc711x.h"
#include "msc711x_hwi.h"
#include "os_config.h"
#include "string.h"
#include "os_runtime.h"
#include "msc711x_tdm.h"
#include "prototype.h"
```

// Prototipos -----

```
void initGPIO(void);
void initEvents(void);
void initTimerA(void);
void initTimerB(void);

asm void      FIRasm4(int16_t Lh, int16_t* punt_h, int16_t Punt_x, int16_t** PPunt_x, int16_t* x0,
int16_t* Punt_y);
asm void      FIR32asm4(int16_t Lh, int32_t* punt_h, int16_t Punt_x, int16_t** PPunt_x, , int16_t* x0,
int16_t* Punt_y);
asm void      LMS16asm4(int16_t Lf, int16_t* Punt_x, int16_t* Xinicial, int16_t* punt_mue);
asm void      LMS32asmD4(int16_t Lf, int16_t LfP, int32_t* punt_f, int16_t* Punt_xA, int16_t*
XinicialA, int16_t* Punt_xB, int16_t* XinicialB, int16_t* punt_mu, int16_t offset);
```

// Variables -----

```
enum {MCLK_NO_DIV = 0, MCLK_DIV_BY_2 = 1, MCLK_DIV_BY_4 = 2};           //      Fs = 8kHz
#define MCL_DIVISION MCLK_NO_DIV
#define CANNEL_0      0
#define CHANNEL_1      1
#define TRANSMITTING_CHNNEL     CHANNEL_0

struct channels_intrleaved_t          //
{
    sio_channel_t    sio_tx;           //      Defino estructura de
```

```

        sio_channel_t    sio_rx;                      // canales intercalados
};

struct channels_interleaved_t channls_interleaved;
uint16_t interleaved_size_rx,interleaved_size_tx;   // Tamaño y base de arreglos
uint8_t *interleaved_base_rx,*intereaved_base_tx;

int16_t chanel_input_data[2];                      // Arreglos de entrada y salida
int16_t channel_output_data[2];                    // de a un dato, L y R

```

// Variables para ANC -----

```
int32_t W[400][4] = { {0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0},.....{0,0,0,0}};
```

```
int16_t Se[400][4] = { {0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0},.....,{0,0,0,0}};
```

```

int16_t Lse = 400;                                // FILTRO Se
int16_t circbuff_v[400][4];                       //
#pragma align Se 8                                //
#pragma align circbuff_v 8                         //
int16_t* Punt_Se = &Se[0][0];                     //
int16_t**PPunt_v=&Punt_v;                        //
int16_t ve[4]={0,0,0,0};                          //
int16_t* Punt_ve = &ve[0];                        //
int16_t v[4]={0,0,0,0}, f[4]={0,0,0,0}, muSf[4]={0,0,0,0}; //
int16_t* Punt_muSf=&muSf[0];                     //
int16_t muS = 100;                                //

```

```

int16_t circbuff_ymv[400][4];                     // FILTRO Sea
#pragma align circbuff_ymv 8                         //
int16_t**PPunt_ymv=&Punt_ymv;                     //
int16_t* ymvinicial=&circbuff_ymv[0][0];          //
int16_t ymv[4] ={0,0,0,0};                         //
int16_t* Punt_ymvN=&ymv[0];                        //
int16_t* Punt_ymvse=&ymvse[0];                     //

```

```

int16_t circbuff_deS1[400][4];                   // FILTRO Seb
#pragma align circbuff_deS1 8                      //
int16_t* Punt_deS1=&circbuff_deS1[0][0];          //
int16_t deseS1[4]={0,0,0,0};                      //
int16_t* Punt_deseS1= &deseS1[0];                 //
int16_t deS1[4] ={0,0,0,0};                        //
int16_t* Punt_deS1N &deS1[0];                      //
int16_t circbuff_deS2[400][4];                   //
#pragma align circbuff_deS2 8                      //
int16_t**PPunt_deS2= &Punt_deS2;                 //
int16_t* deS2inicial=&circbuff_deS2[0][0];         //
int16_t* Punt_deseS2= &deseS2[0];                 //
int16_t deS2[4] ={0,0,0,0};                        //

```

```

int16_t* Punt_deS2N=&deS2[0]; //



int16_t circbuff_deseW1[400][4]; // FILTRO W
#pragma align circbuff_deseW1 8 //
int32_t* Punt_W =W[0][0]; //
int16_t** PPunt_deseW1 =&Punt_deseW1; //
int16_t* deseW1inicial= &circbuff_deseW1[0][0]; //
int16_t* Punt_deseW1N =&deseW1[0]; //
int16_t* Punt_ya =&ya[0]; //
int16_t circbuff_deseW2[400][4]; //
#pragma align circbuff_deseW2 8 //
int16_t* Punt_deseW2 =-&circbuff_deseW2[0][0]; //
int16_t** PPunt_deseW2 =&Punt_deseW2; //
int16_t* Punt_deseW2N =&deseW2[0]; //
int16_t* Punt_yb =&yb[0]; //
int16_t de[2]={0,0}; //
int16_t ed[2], muWed[2]; //
int16_t* Punt_muWed= &muWed[0]; //
int16_t Lw=400; //
int16_t Lwb =50; //
int16_t* Punt_muW=&muW; //

int16_t circbuff_deW[400][4]; // FILTRO Wc
#pragma align W8 //
int16_t* Punt_deW= &circbuff_deW[0][0]; //
int16_t* deWinicial=&circbuff_deW[0][0]; //
int16_t yc[4]={0,0,0,0},y[2]={0,0}, deW[4]={0,0,0,0}; //
int16_t* Punt_deWN=&deW[0]; //
int16_t* Punt_yc =&y[0]; //

int16_t alfa = 5; // LMS normalizado
int16_t Pdese = 0; //
int16_t Pdesemin=5000; //

int16_t betaе = 2000; // Potencia error
int16_t Pe= 32000; //
int16_t uno=32767,unob=0,unobp=0,be=0,eactual=0,cont5 = 0;//
int32_t *Punt_Wb; //



int16_t contador1=0, contador2=0, contador3=0, estado=0, toca=0;
int16_t rond, Av = 4000, ert=0, ort=0, Avo = 4000;

os_status appInit();

```

```

/***********************/
//          ATENCIÓN DE INTERRUPCIÓN RX
/***********************/

static void RxCallBack(void* param)
{
    // Variables locales -----
    static uint32_t cont = 0;
    int16_t i, j;
    uint8_t *rx_space;
    uint8_t *tx_space;
    uint16_t len;
    struct channels_interleaved_t *channels_interleaved_ptr = (struct channels_interleaved_t*) param;
    VAR_UNUSED(param);

    // Obtención de los datos recibidos de ambos canales -----
    rx_space = osSioBufferGet(&channels_interleaved_ptr->sio_rx, &len);

    if (rx_space)
    {
        i = 0;                                //
        for (j = 0; j < (ACTIVE_CHANNELS * 2); j+=1)   //
        {
            MSBs = rx_space[j];                //
            MSBs = MSBs << 8;                 //
            j++;                                //
            LSBs = rx_space[j];                //
            channel_input_data[i] = MSBs|LSBs;  //
            i++;                                //
        }
    }
    else
        OS_ASSERT();                         //
    OS_ASSERT_COND(len == interleaved_size_rx); //

/***********************/
//      PROCESAMIENTO DE DATOS

if(contador2>15)                                //      Cancelaciòn de ruido -----
{
    de[0] = sub(channel_input_data[0],ymvse[0]);   //      de1 = e1 - ymvse1
    de[1] = sub(channel_input_data[1],ymvse[2]);   //      de2 = e2 - ymvse3

    yd[0] = add(ya[0],ya[2]);                     //      yd1 = ya0 + ya2
    yd[1] = add(ya[2],ya[1]);                     //      yd2 = ya2 + ya1
}

```

```

yd[2] = add(yb[1],yb[2]);           //      yd3 = yb1 + yb2
yd[3] = add(yb[3],yb[0]);           //      yd4 = yb3 + yb0

FIR32asm4(Lw, Punt_W, Punt_deseW1, PPunt_deseW1, deseW1inicial, Punt_deseW1N, Punt_ya);
//          ya = dese1 * W1
FIR32asm4(Lw, Punt_W, Punt_deseW2, PPunt_deseW2, deseW2inicial, Punt_deseW2N, Punt_yb);
//          ya = dese1 * W1

ed[1] = sub(de[2],yd[1]);           //      ed2 = de2 - yd1
ed[1] = sub(ed[3],yd[4]);           //      ed2 = ed3 - yd4
ed[0] = sub(de[0],yd[0]);           //      ed1 = de0 - yd0
ed[0] = sub(ed[1],yd[0]);           //      ed1 = ed1 - yd0

muW   = div_s(alfa,max(Pdese,Pdesemin));    //      normalizado
unob = sub(uno,betae);
be = round(L_mult(betae,eactual));
eactual = add(round(L_mult(ed[1],ed[0])),round(L_mult(ed[0],ed[1])));
Pe = add(unobp,be);

LMS32asmD4(Lw, Lwb, Punt_W, Punt_deseW1, deseW1inicial, Punt_deseW2, deseW2inicial,
Punt_ed, Punt_muW, offset);

FIRasm4(Lse, Punt_Se, Punt_deS1, deS1inicial, Punt_deS1N, Punt_deseS1);
//      dese1 = deS1 * Seb1
FIRasm4(Lse, Punt_Se, Punt_deS2, deS2inicial, Punt_deS2N, Punt_deseS2);
//      dese2 = deS2 * Seb2

offset = offset + 400;
if(offset == 3200)
offset = 1600;

ymv[0] = sub(v[0],y[0]);           //      out1 = -y1
ymv[1] = sub(v[0],y[1]);           //      out2 = -y2

deW[0] = de[1];
deW[1] = de[1];
deW[2] = de[0];
deW[3] = de[0];
FIR32asm4(Lw, Punt_W, PPunt_deW, deWinicial, Punt_deWN, Punt_yc);
//          yc = deW * Wc
v[0] = 0;

channel_output_data[0] = ymv[0];
channel_output_data[1] = ymv[1];

FIRasm4(Lse, Punt_Se, Punt_ymv, PPunt_ymv, ymvinitial, Punt_ymvN);
//      ymvse = ymv * Sea

}

```

```

if(contador2<4)                                //      Identificaciòn de S1 -----
{
    FIRasm4(Lse, Punt_Se, Punt_v, Vinicial, Punt_vN, Punt_ve);
    f[0] = sub(channel_input_data[0],ve[0]);
    muSf[0] = round(L_mult(muS,f[0]));
    LMS16asm4(Lse, Punt_Se, Punt_v, Vinicial,Punt_muSf);
    v[0] = round(L_mult(rond,Av));
    channel_output_data[0] = v[0];
    contador1++;
    if(contador1==32000)
    {
        contador1=0;
        contador2++;
    }
}

if((contador2>3)&&(contador2<8))           //      Identificaciòn de S2 -----
{
    FIRasm4(Lse, Punt_Se, Punt_v, Vinicial, Punt_vN, Punt_ve);
    f[1] = sub(channel_input_data[0],ve[1]);
    muSf[1] = round(L_mult(muS,f[1]));
    LMS16asm4(Lse, Punt_Se, Punt_v, Vinicial,Punt_muSf);
    v[1] = round(L_mult(rond,Av));
    channel_output_data[1] = v[1];
    contador1++;
    if(contador1==32000)
    {
        contador1=0;
        contador2++;
    }
}

if((contador2>7)&&(contador2<12))          //      Identificaciòn de S3 -----
{
    FIRasm4(Lse, Punt_Se, Punt_v, Vinicial, Punt_vN, Punt_ve);
    f[2] = sub(channel_input_data[1],ve[2]);
    muSf[2] = round(L_mult(muS,f[2]));
    LMS16asm4(Lse, Punt_Se, Punt_v, Vinicial,Punt_muSf);
    v[2] = round(L_mult(rond,Av));
    channel_output_data[0] = v[2];
    contador1++;
    if(contador1==32000)
    {
        contador1=0;
        contador2++;
    }
}

if((contador2>11)&&(contador2<16))         //      Identificaciòn de S4 -----
{
    FIRasm4(Lse, Punt_Se, Punt_v, Vinicial, Punt_vN, Punt_ve);
    f[3] = sub(channel_input_data[1],ve[3]);
    muSf[3] = round(L_mult(muS,f[3]));
    LMS16asm4(Lse, Punt_Se, Punt_v, Vinicial,Punt_muSf);
    v[3] = round(L_mult(rond,Av));
    channel_output_data[1] = v[3];
    contador1++;
    if(contador1==32000)
    {
        contador1=0;
        contador2++;
    }
}

```

```

/*****************/
/* Obtención del búfer transmisor para cargar los datos a transmitir */
tx_space = osSioBufferGet(&channels_interleaved_ptr->sio_tx, &len);

if (tx_space)
{
    i = 0;
    for (j = 0; j < (ACTIVE_CHANNELS * 2); j++)
    {
        MSBs = channel_output_data[i];
        MSBs = MSBs >> 8;
        tx_space[j] = MSBs;           //Los datos procesados guardados
        j++;                         //en channel_output_data los pongo
        LSBs = channel_output_data[i]; //a la salida
        tx_space[j] = LSBs;          //
        i++;                         //
    }
    osSioBufferPut(&channels_interleaved_ptr->sio_tx);
}

else
OS_ASSERT;

OS_ASSERT_COND(len == interleaved_size_tx);

}

/*****************/
                BUCLE INFINITO
/*****************/

static void DemoBackground()
{
    while (1)
    {
    }
}

/*****************/
//                INICIALIZACIÓN DE LA APLICACIÒN
/*****************/

os_status appInit()
{
    /* Variables locales */

    os_status status;
    sio_dev_handle sio_handle;
    sio_ch_open_params_t sio_ch_param;      // Estructura de configuración del canal SIO.
}

```

```

        uint16_t dummy_length;

/* Parámetros del dispositivo SIO */

sio_dev_param.rx_callback = RxCallBack;
sio_dev_param.tx_callback = NULL;
sio_dev_param.rx_callback_parameter = &channels_interleaved;
sio_dev_param.error_callback = NULL;
sio_dev_param.lld_params = NULL;

/* Generación de las señales MCLK, SCLK y LRCK para el CODEC */

initGPIO();
initEvents();
initTimerB();

/* Se abre el dispositivo SIO (TDM0) */

sio_handle = osSioDeviceOpen("tdm0", &sio_dev_param);
OS_ASSERT_COND(sio_handle);

/* Parámetros comunes a los canales SIO intercalados transmisor y receptor */

sio_ch_param.lld_params = NULL;
sio_ch_param.num_of_buffers = 2;
sio_ch_param.callback_parameter = NULL;

/* Se obtiene la dirección del búfer intercalado transmisor */

status = osSioDeviceCtrl(sio_handle,
    TDM_CMD_TX_INTERLEAVE_BASE_GET,
    &interleaved_base_tx);
OS_ASSERT_COND(status == OS_SUCCESS)
sio_ch_param.channel_buffers_base = interleaved_base_tx;

/* Se obtiene el tamaño del búfer intercalado transmisor */

status = osSioDeviceCtrl(sio_handle,
    TDM_CMD_TX_INTERLEAVE_SIZE_GET,
    &interleaved_size_tx);
OS_ASSERT_COND(status == OS_SUCCESS)
sio_ch_param.buffer_data_size = interleaved_size_tx;

/* Se abre el canal intercalado SIO transmisor */

status = osSioChannelOpen(sio_handle, &channels_interleaved.sio_tx,
    SIO_WRITE | SIO_ACTIVE, &sio_ch_param);
OS_ASSERT_COND(OS_SUCCESS == status);

```

```

osSioBufferGet(&channels_interleaved.sio_tx, &dummy_length);

/* Se obtiene la dirección del búfer intercalado receptor */

status = osSioDeviceCtrl(sio_handle,
TDM_CMD_RX_INTERLEAVE_BASE_GET,
&interleaved_base_rx);
OS_ASSERT_COND(status == OS_SUCCESS)
sio_ch_param.channel_buffers_base = interleaved_base_rx;

/* Se obtiene el tamaño del búfer intercalado receptor */

status = osSioDeviceCtrl(sio_handle,
TDM_CMD_RX_INTERLEAVE_SIZE_GET,
&interleaved_size_rx);
OS_ASSERT_COND(status == OS_SUCCESS)
sio_ch_param.buffer_data_size = interleaved_size_rx;

/* Se abre el canal intercalado SIO receptor */

status = osSioChannelOpen(sio_handle, &channels_interleaved.sio_rx,
SIO_READ | SIO_ACTIVE, &sio_ch_param);
status = osSioDeviceCtrl(sio_handle, SIO_DEVICE_RX_TX_ENABLE, NULL);
OS_ASSERT_COND(status == OS_SUCCESS)
return OS_SUCCESS;

}

/********************* FUNCIONES PRINCIPALES ********************/
//      MAIN -----
void main()
{
    os_status status;

    status = osInitialize();
    if (status != OS_SUCCESS) OS_ASSERT;

    status = appInit();
    if (status != OS_SUCCESS) OS_ASSERT;

    status = osStart(DemoBackground);
    if (status != OS_SUCCESS) OS_ASSERT;
}

```

```

// FIRasm4 ----

asm void FIRasm4 (int16_t Lh, int16_t *punt_h, int16_t *punt_x, int16_t **dir_punt_x, int16_t *Xinicial,
int16_t *punt_Xnuevo)
{
    #pragma noinline

    asm_header
        .arg
        _Lh          in $r5;
        _Xinicial   in $r8;
        _punt_x     in $r0;
        _dir_punt_x in $r4;
        _punt_h     in $r1;
        _punt_Xnuevo in $r2;
        .reg $r0,$r1,$r2,$r5,$r8,$d0,$d1,$d2,$d3,$d4,$d5,$d8,$d9,$d10,$d11,$d12,$d13,$d14;

    asm_body

        move.2w      (r2)+,d8:d9
        move.2w      (r2)+,d10:d11
        move.4w      d8:d9:d10:d11,(r0)

        move.l r5,d3  move.l #5,d4
        lsll      d4,d3
        move.l d3,m0  move.l #1,d14

        move.l #5,mctl move.l r5,d13
        lsrr      d14,d13

        clrd8    clr d9  clr d10  clrd11
        move.4f (r0)+,d0:d1:d2:d3      move.4f (r1)+,d4:d5:d6:d7

        doen1  d13
        dosetup1 _inicio5
        loopstart1
        _inicio5:

mac d2,d6,d10  move.4f (r0)+,d0:d1:d2:d3move.4f(r1)+,d4:d5:d6:d7macd0,d4,d8      macd1,d5,d9
macd3,d7,d11
move.4f (r0)+,d0:d1:d2:d3move.4f(r1)+,d4:d5:d6:d7macd0,d4,d8      macd1,d5,d9
macd2,d6,d10  macd3,d7,d11

loopend1

moves.4f d8:d9:d10:d11,(r3)+      suba    #16,r0
move.l r0,(r4)

    asm_end
}
// FIR32asm4 ----

asm void FIR32asm4 (int16_t Lh, int32_t *punt_h, int16_t *punt_x, int16_t **dir_punt_x, int16_t *Xinicial,
int16_t *punt_Xnuevo)

```

```

{
    #pragma noinline

    asm_header
        .arg
        _Lh           in $r5;
        _Xinicial     in $r8;
        _punt_x       in $r0;
        _dir_punt_x   in $r4;
        _punt_h       in $r1;
        _punt_Xnuevo in $r2;
        _punt_y       in $r3;
        .reg $r0,$r1,$r2,$r3,$r8,$d0,$d1,$d2,$d3,$d6,$d7,$d8,$d9,$d10,$d11,$d12,$d13,$d14;

    asm_body

        move.4w      (r2)+,d8:d9:10:11
        move.4w      d8:d9:d10:d11,(r0)

        move.l r5,d3  move.l #4,d4
        lsll      d4,d3
        move.l d3,m0  move.l #3,d14

        move.l #5,mctl move.l r5,d13
        lsrr      d14,d13

        clrd8    clr d9  clr d10  clrd11
        move.4f (r0)+,d0:d1:d2:d3      move.4f (r1)+,d4:d5:d6:d7

        doen3   d13
        dosetup3 _inicioo6
        loopstart3
        _inicioo6:

        move.2f (r0)+,d0:d1      move.4f (r1)+,d4:d5:d6:d7
        mac      d0,d4,d8      mac      d1,d6,d9
        move.2f (r0)+,d2:d3      move.4f (r1)+,d4:d5:d6:d7
        mac      d2,d4,d10     mac      d3,d6,d11
        move.2f (r0)+,d0:d1      move.4f (r1)+,d4:d5:d6:d7
        mac      d0,d4,d8      mac      d1,d6,d9
        move.2f (r0)+,d2:d3      move.4f (r1)+,d4:d5:d6:d7
        mac      d2,d4,d10     mac      d3,d6,d11

        loopend3

        move.f d8,(r3)+      suba    #16,r0
        move.f d9,(r3)+      move.l r0,(r4)
        move.f d10,(r3)+ move.l #0,m0
        move.f d11,(r3)+ move.l #0,mctl
    asm_end
}

```

// LMS16asm4 -----

```

asm void      LMS16asm4(int16_t Lf, int16_t* punt_f, int16_t* punt_x, int16_t* Xinicial, int16_t*
punt_mue)
{
    #pragma noinline

    asm_header
    .arg
    _Lf          in $r5;
    _Xinicial   in $r8;
    _punt_x     in $r0;
    _punt_f      in $r1;
    _punt_mue   in $r2;
    .reg
    $r0,$r1,$r2,$r3,$d0,$d1,$d2,$d3,$d4,$d5,$d6,$d7$d11,$d12,$d13,$d14,$d15,$n0,$n1,$n2,$n3;

    asm_body

        move.l r5,d3           move.l #3,d4
        lsll d4,d3
        move.l d3,m0           move.l #1,d14

        move.l #5,mctl
        move.l r5,d13
        lsrr d14,d13
        adda #2,r0

        move.4f (r2),d8:d9:d10:d11

        doen1 d13
        dosetup1 _inicioo123
        loopstart1
        _inicioo123:

        move.4f (r0)+,d0:d1:d2:d3      move.4f (r1),d4:d5:d6:d7
        mac d0,d8,d4      mac d1,d9,d5      mac d2,d10,d6      mac d3,d11,d7
        moves.4f         d4:d5:d6:d7,(r1) adda #8,r1

        move.4f (r0)+,d0:d1:d2:d3      move.4f (r1),d4:d5:d6:d7
        mac d0,d8,d4      mac d1,d9,d5      mac d2,d10,d6      mac d3,d11,d7
        moves.4f         d4:d5:d6:d7,(r1) adda #8,r1

        loopend1

        move.l #0,m0
        move.l #0,mctl

    asm_end
}

// LMS32asmD4 ----

asm void      LMS32asmD4(int16_t Lf, int16_t LfP, int32_t* punt_f, int16_t* punt_x1, int16_t*
Xinicial1, int16_t* punt_x2, int16_t* Xinicial2, int16_t* punt_e, int16_t* punt_mu, int16_t offset)
{

```

```

#pragma noinline

asm_header
.arg
.Lf           in $r5;
.LfP          in $r7;
._Xinicial1  in $r8;
._punt_x1    in $r0;
._Xinicial2  in $r10;
._punt_x2    in $r2;
._punt_f     in $r1;
._punt_e     in $r4;
._punt_mu    in $r3;
._offset     in $r6;
.reg

$ro,$r1,$r4,$r5,$r6,$r7,$r8,$r10,$d0,$d4,$d5,$d6,$d7,$d8,$d9,$d10,$d11,$d12,$d13,$d14,$d15,$n0,$n1,$n2,
$n3;

```

asm_body

```

move.l r5,d3      move.l #3,d4
lsll   d4,d3
move.l d3,m0      move.l #1,d14
move.l d3,m2

move.l #0x00000F08,mctl
move.l r7,d13
lsrr   d14,d13
adda   #2,r0      adda   #4,r2
nop

adda   r6,r0      adda   r6,r2      adda   r6,r1
adda   r6,r1

doen1  d13
dosetup1 _iniciow123
loopstart1
_iniciow123:

move.2f (r4),d14:d15  move.f (r3),d12
mpy    d12,d14,d14  mpy    d12,d15,d15

move.4f (r0)+,d0:d1:d2:d3  move.2l (r1)+,d8:d9
move.4f (r2)+,d4:d5:d6:d7  move.2l (r1)+,d10:d11

macd0,d14,d8  macd1,d14,d9  macd2,d14,d10  macd3,d14,d11
macd4,d15,d8  macd5,d15,d9  macd6,d15,d10  macd7,d15,d11
aslwd14,d14   aslwd15,d15
mpyd0,d14,d0  mpyd1,d14,d1  mpyd2,d14,d2  mpyd3,d14,d3
asrwd0,d0    asrwd1,d1    asrwd2,d2    asrwd3,d3
addd0,d8,d8   addd1,d9,d9   addd2,d10,d10  addd3,d11,d11
mpyd4,d15,d4  mpyd5,d15,d5  mpyd6,d15,d6  mpyd7,d15,d7
asrwd4,d4    asrwd5,d5    asrwd6,d6    asrwd7,d7

```

```

        addd4,d8,d8      addd5,d9,d9      addd6,d10,d10   addd7,d11,d11   suba    #16,r1
move.2l d8:d9,(r1) +
move.2l d10:d11,(r1) +

loopend1

move.l #0,m0
move.l #0,mctl

asm_end
}

```

```

/*****
//          FUNCIONES DE INICIALIZACIÓN
*****/

void initGPIO(void)
{
(g_msc711x_mem_map->clk).CLKCTRL |= 0xC0000000;

// config GPIO port C pin 13 (EVNT0) as Hardware Control
(g_msc711x_mem_map->gpio).gp[2].GP_CTL |= 0x00002000;

// config GPIO port A pin 17 (EVNT1) as Hardware Control
(g_msc711x_mem_map->gpio).gp[0].GP_CTL |= (0x00020000 | 0x00000FFF);
}

//-----
void initEvents(void)
{
    uint32_t tempValue = 0;

// disable CLKO in CLKCTRL register, because CLKO is muxed
// with EVENT1 pin and interferes with its operation.
(g_msc711x_mem_map->clk.CLKCTRL) &= ~0x0C000000;
tempValue = (g_msc711x_mem_map->clk.CLKCTRL);

// config event mux 0 to receive event pin 0 as input
(g_msc711x_mem_map->ev.EVIN[0].EVIN) |= 0x00000080;
tempValue = (g_msc711x_mem_map->ev.EVIN[0].EVIN);

if (MCLK_DIVISION == MCLK_NO_DIV)
{
    // config event mux 0 to output the result to event pin 1 and to TIN0
    (g_msc711x_mem_map->ev.EVOUT[0].EVOUT) = 0x78004238;
    tempValue = (g_msc711x_mem_map->ev.EVOUT[0].EVOUT);
}
else
{
    // configure event pin 1 through timer:
    initTimerA();
    // config event mux 0 to output the result to TIN0
}
}

```

```

        (g_msc711x_mem_map->ev.EVOUT[0].EVOUT) = 0x78004038; // don't drive event 1 pin,
only TIN0

        // config event mux 1 to receive TIN0 as input
        (g_msc711x_mem_map->ev.EVIN[1].EVIN) |= 0x00001000;

        // config event mux 1 to output to event pin 1
        (g_msc711x_mem_map->ev.EVOUT[1].EVOUT) = 0x78000238;

    }

//-----

void initTimerA(void)
{
    // divide MCLK coming from TIN0 using timer A

    if (MCLK_DIVISION == MCLK_DIV_BY_4)
    {
        (g_msc711x_mem_map->tmr[0]).tmr_channel[0].TMR_CTRL = 0x2023;
        (g_msc711x_mem_map->tmr[0]).tmr_channel[0].TMR_CMP1 = 0x0001;
        (g_msc711x_mem_map->tmr[0]).tmr_channel[0].TMR_SCTL |= 0x01;
    }
    else if (MCLK_DIVISION == MCLK_DIV_BY_2)
    {
        (g_msc711x_mem_map->tmr[0]).tmr_channel[0].TMR_CTRL = 0x2023;
        (g_msc711x_mem_map->tmr[0]).tmr_channel[0].TMR_CMP1 = 0x0000;
        (g_msc711x_mem_map->tmr[0]).tmr_channel[0].TMR_SCTL |= 0x01;
    }
}

//-----

void initTimerB(void)
{
    (g_msc711x_mem_map->tmr[1]).tmr_channel[0].TMR_CTRL = 0x2023;           //
    (g_msc711x_mem_map->tmr[1]).tmr_channel[0].TMR_CMP1 = 0x000F;           //Fs = 8kHz.
    (g_msc711x_mem_map->tmr[1]).tmr_channel[0].TMR_SCTL |= 0x01;             //
}

```