UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Realización de una Tarjeta de Multisensado de Voz sobre una Plataforma DSP

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO ELÉCTRICO ELECTRÓNICO

PRESENTAN:

Daniel Fernando González Castro

Alfredo Gutiérrez Escoffié

DIRECTOR DE TESIS:

M.I. Larry Hipólito Escobar Salguero

CIUDAD UNIVERSITARIA, 2008

Agradecimientos
Daniel González Castro

Agradezco en primer lugar a mis padres, Delia y Andrés, por su apoyo toda la vida y por creer en mi siempre. Gracias por ponernos siempre a mi hermano y a mi antes que nada.

Gracias a mi hermano, Juan Pablo, por ser un ejemplo para mi, por tu apoyo y por tu amistad que en estos años han sido importantísimos para mi.

Agradezco a mis amigos y amigas por estar conmigo, por las experiencias y diversiones que han servido para distraerme un poco del trabajo y disminuir el estrés.

Muchas gracias al M. I. Larry Escobar por compartir su tiempo y sus conocimientos para el desarrollo de este proyecto; y en general a todos mis maestros por participar en mi desarrollo profesional.

Agradezco y dedico esta tesis a mi abuelo, Felipe, y a mi tía, Patricia, que siempre me apoyaron pero lamentablemente no pudieron ver la culminación de este trabajo.

Agradecimientos
Alfredo Gutiérrez Escoffié

A mi padre; porque con su apoyo propició en mi la búsqueda del conocimiento; cuando todavía era niño aprobó mis búsquedas de respuestas; aunque después no pudiera rearmar lo que había desarmado para saber cómo funcionaba. *Gracias por permitirme experimentar*.

A mi madre; que día con día acompañó mi carrera estudiantil, apoyándome en cosas pequeñas pero indispensables, *el simple hecho de saber que estás ahí es invaluable*.

A mis hermanas; que han compartido conmigo gran parte de este esfuerzo por conocer, si bien, no en lo académico, si en la vida.

A mis maestros; porque todos contribuyeron de alguna manera en este trabajo, culminación de mi vida académica. En particular, al M.I. Larry Escobar, quien proporcionó los medios para llevar acabo el proyecto, apoyándolo en todo momento con sus conocimientos.

Índice

Índice de figuras

Resumen

En este trabajo se describen los conocimientos teóricos necesarios para la implementación de una tarjeta de desarrollo para la tarjeta de desarrollo DSK del procesador digital de señales TMS320C5402, así como el proceso de diseño e implementación de dicha tarjeta.

El texto está dividido en seis capítulos y cuatro anexos. Los primeros tres capítulos corresponden al marco teórico de los procesos que se llevarán acabo en la tarjeta de expansión diseñada. El *primero* de ellos describe brevemente la tarjeta de expansión e indica los objetivos de este trabajo. El *segundo capítulo* trata sobre el análisis de señales y brinda las herramientas necesarias para el estudio y representación de estas, haciendo énfasis en la señal de interés, la voz. El *tercer capítulo* trata sobre una técnica para conocer las características espaciales de las señales llamada multisensado y de las virtudes del procesamiento adaptable; el proceso llamado formador de haz es descrito en éste capítulo. El *cuarto capítulo* describe el proceso de diseño de la tarjeta de expansión y los procesos que se pueden realizar en ella para comprobar su correcto funcionamiento. En el *quinto capítulo* se presentan los resultados obtenidos en las pruebas de laboratorio con la tarjeta diseñada. Finalmente, el *sexto capítulo* contiene las conclusiones obtenidas de este desarrollo.

Los anexos contienen información complementaria más detallada de las herramientas utilizadas en este trabajo. El *primero* de ellos, trata de las características generales del procesador digital de señales utilizado. El *segundo* presenta los códigos de los programas elaborados en C para el compilador Code Composer Studio de la tarjeta de desarrollo. El *anexo C* describe el modo de uso de la función de matlab fdatool utilizada en el cálculo de los coeficientes de los filtros digitales. Por último el *anexo D* describe el ambiente en que se diseño el circuito impreso de la tarjeta de expansión.

Capítulo 1
Introducción

Las señales acústicas son de gran importancia para los seres humanos, ya que las utilizamos como nuestra principal forma de comunicación, el habla. Cuando hablamos con otra persona a nuestros oídos llega su voz acompañada de diversos sonidos intrínsecos al ambiente que no son de nuestro interés y son considerados interferencia o ruido. En las salas de conferencias, el ponente se enfrenta con la difícil tarea de hacer llegar su voz a todos los escuchas de forma que ellos la puedan entender; esto es difícil sin la ayuda de dispositivos electrónicos, como el micrófono. Sin embargo, el micrófono tiene como defecto el convertir en señales eléctricas todas las señales acústicas que llegan a éste, es decir, transduce tanto la voz del ponente, como los otros sonidos presentes en la sala, tales como el ruido de sus pasos, o los murmullos de las personas que se encuentran cerca de este. Estos sonidos no deseados pueden eliminarse mediante el procesamiento de la señal de audio.

Existen dispositivos de propósito específico cuyo objetivo es procesar señales. En la actualidad las señales son tratadas de forma digital, porque esto simplifica los procesos, facilita su almacenamiento, su transmisión y disminuye los tiempos de procesado. Los procesadores digitales de señales (DSPs por sus siglas en inglés), son muy populares en la actualidad en el ámbito de la electrónica, por su reducido tamaño, bajo consumo de energía y su capacidad de realizar operaciones aritméticas y lógicas en paralelo. En particular la familia de DSPs TMS320C5XXX de Texas Instruments fue diseñada para procesar señales de audio.

Este trabajo tiene como *objetivo* desarrollar una tarjeta de expansión para la tarjeta de desarrollo (DSK) del DSP TMS320C5402 (C5402) de Texas Instruments (TI) que permita elegir entre diferentes procesos a realizar en una o varias señales de audio, entre estos multisensado de voz y formación de haz. El desarrollo de este trabajo surgió de la idea de tener una plataforma de un arreglo de micrófonos para la prueba de algoritmos de formador de haz en tiempo real.

La tarjeta de desarrollo DSK con la que se trabajará cuenta con una sola entrada y una sola salida de audio, por lo que para adquirir más de una señal de audio es necesario expandirla. Para ello, la tarjeta DSK cuenta con dos buses de expansión de ochenta pines cada uno, uno de los cuales cuenta con veintidós pines para dirección y treinta y dos pines para señales de entrada o salida en paralelo [2]; con los cuales se pueden manejar señales adicionales.

En la tarjeta a desarrollar se pretende adquirir simultáneamente las señales de voz que lleguen a cuatro micrófonos, para ser procesadas por el C5402; el proceso será seleccionado mediante un selector de cuatro interruptores con lo cual se podrá elegir entre dieciséis procesos.

Dentro del ámbito del procesamiento de señales existe un proceso llamado formador de haz (beamforming), cuyo objetivo es generar una señal de un solo canal a partir de arreglos de varios sensores, mediante el filtrado y la suma de las señales individuales [1]. La respuesta espacial de dicho formador de haz puede describirse como un patrón de radiación cuya ganancia depende de la dirección de incidencia de las señales acústicas.

Este proceso puede utilizarse con señales de audio, por lo que a partir de las señales obtenidas por un conjunto de micrófonos, generaríamos una señal acústica cuya ganancia dependa de la dirección de la que proviene el sonido; tendríamos así la posibilidad de dejar pasar la voz del ponente y eliminar, o atenuar de forma considerable, las señales de audio que no son de interés.

Para el desarrollo de nuestro trabajo las señales de voz serán adquiridas por medio de un arreglo de cuatro micrófonos y una vez adquiridas las señales serán amplificadas con amplificadores operacionales, ya que su amplitud es muy reducida. Las señales analógicas amplificadas se convertirán en señales digitales con un convertidor analógico digital (ADC), para después ser procesadas por el DSP C5402 (parte medular del DSK), quien también generará la salida, tal como se ilustra en la figura 1. El procesamiento que realizará el DSP dependerá de la posición en que se encuentre el selector. Entre estos procesos estarán la selección de canales de forma individual; el guardado de las señales de audio de cada canal, de dos canales o de cuatro canales; la formación de haz con dos micrófonos, y formador de haz con cuatro micrófonos.

Los micrófonos estarán dispuestos en un arreglo lineal y se utilizarán amplificadores de propósito general, el ADC será con salida paralela y cuatro entradas analógicas. El selector

constará de cuatro interruptores, con lo cual se tendrán hasta dieciséis opciones posibles de procesos.

Los DSPs son herramientas frecuentemente utilizadas en aplicaciones de audio tales como la telefonía celular, ya que su rapidez para realizar operaciones matemáticas (suma y multiplicación, principalmente) les permite realizar procesos en tiempo real. El DSP C5402 fue elegido como herramienta principal para el procesamiento de la señal de audio por su rapidez, que nos permite adquirir cuatro señales de voz muestreadas a 8 kHz, quedando tiempo suficiente para realizar alrededor de dos mil instrucciones de ensamblador para implementar el algoritmo deseado.

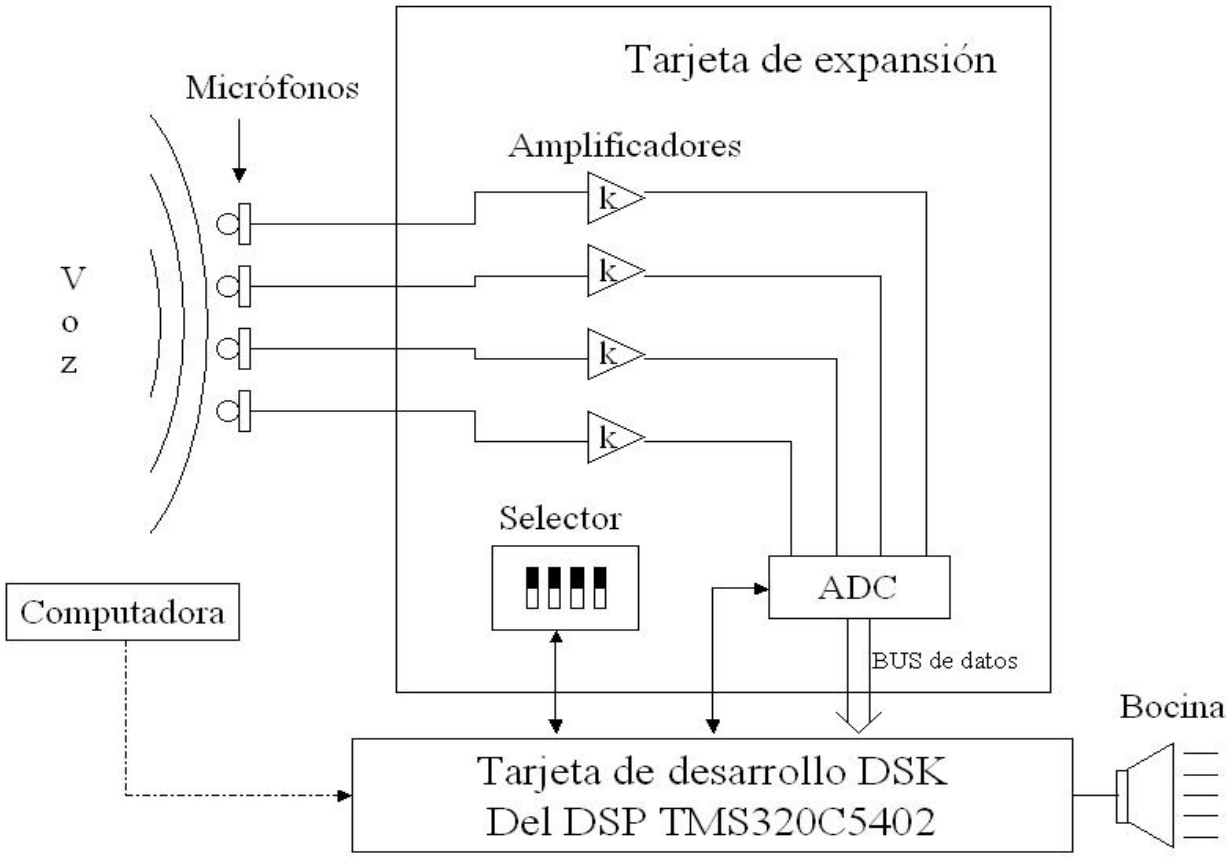


Figura 1.1. Diagrama General del sistema

En este proyecto se pretende desarrollar una tarjeta de expansión que convertirá las señales adquiridas por los micrófonos en señales digitales y las enviará al DSP; además de enviarle la información del estado de los interruptores de selección; los pines de los buses de expansión quedarán accesibles para futuras aplicaciones. Esta tarjeta podrá utilizarse en futuros proyectos en los que se requiera adquirir hasta cuatro señales simultáneas de voz.

El proceso de formación de haz deberá disminuir de forma significativa las señales que provengan de direcciones que no sean de interés así como el ruido inherente al medio ambiente. Además de la aplicación en salas de conferencias, agregando algunas modificaciones, este sistema puede usarse en ayudas auditivas para ambientes muy ruidosos o por personas con dificultades auditivas.

Este trabajo está dividido en las siguientes partes:

En el capítulo 2, "*Análisis de señales y características de la voz*", se describen las características de las señales tanto en el tiempo como en la frecuencia, también se muestran señales en su representación continua y discreta. Se mencionan señales espaciales como la voz y sus características y se explica lo que es el procesamiento de voz.

En el capítulo 3, "*Multisensado y formación de haz*", se explica cómo funcionan los arreglos de sensores y una aplicación muy importante de éstos, la formación de haz y los métodos para realizarlo.

En el capítulo 4, "*Análisis y desarrollo del sistema*", se describe el diseño de la tarjeta de expansión, tanto el hardware como el software, y se explican los procesos que se pueden realizar y cómo seleccionar cada uno de ellos.

En el capítulo, 5 "*Pruebas y resultados*", se detallan las pruebas que se realizaron para comprobar el funcionamiento de la tarjeta y se muestran gráficas de los resultados obtenidos.

Al final se presentan las conclusiones del trabajo.

Capítulo 2

Análisis de Señales y características de la voz

El *análisis de señales* se dedica al estudio y caracterización de las propiedades de las señales y ha sido desarrollado desde el descubrimiento de las señales fundamentales en la naturaleza como el campo magnético y las ondas sonoras. Generalmente, una señal es una función de varias variables, por ejemplo, el campo magnético varía en el espacio así como en el tiempo.

Las señales se caracterizan principalmente por su variación en el tiempo, pero para entenderlas mejor, es bueno estudiarlas desde representaciones diferentes. Además del tiempo, otra representación importante es la frecuencia. Las herramientas para estudiar la representación en frecuencia fueron inventadas por Fourier, quien buscaba encontrar una ecuación para el comportamiento del calor. El análisis espectral se convirtió en una de las herramientas científicas más importantes gracias a Bunsen y Kirchhoff, quienes observaron que el espectro de luz puede ser utilizado para reconocer, detectar y clasificar sustancias debido a que cada sustancia tiene su espectro propio [3].

Otra forma importante de representar las señales es la representación discreta de ellas. Esta nos permite almacenarlas sin que sufran deterioro durante el proceso de almacenamiento ni perdida de información debida al uso, permitiendo de esta forma que su transportación sea muy confiable; además, las señales digitales se pueden trabajar con procesos muy elaborados que en un procesador analógico resultarían demasiado difíciles o hasta imposibles de realizar [4].

Las señales son estudiadas principalmente en función del tiempo, pero algunas señales también se pueden estudiar en función de otras variables, como el espacio. En este caso pueden depender de una sola variable o hasta de tres variables si se trabaja con señales tridimensionales. El sonido, la voz y el audio son ejemplos de señales que pueden ser estudiadas en función del tiempo así como en función del espacio.

2.1 Clasificación de las señales

Una señal se puede considerar como una variable o cantidad física que provee información sobre el estado o evolución de un sistema o fenómeno, es decir, puede considerarse como un fenómeno físico (cantidad física) que experimenta cambios en el tiempo, espacio u otra variable independiente; una señal es una descripción de cómo un parámetro varia respecto de otro parámetro.

El hombre genera continuamente señales eléctricas para la transmisión inalámbrica de radio y televisión, señales de radares para la detección de aviones y señales de sonar. Las señales también se pueden transmitir usando cables o fibra óptica, actualmente se utiliza una combinación de cables y medios inalámbricos para transmitir señales entre puntos distantes.

Existen señales, como las acústicas, biológicas y mecánicas, que no son generadas por medios eléctricos, sin embargo, es posible modelar la cantidad física a través de una señal eléctrica como voltaje o corriente.

Una señal $f(x)$ se puede definir como una función de x que varía de cierta manera para llevar información. La mayoría de las veces se utiliza el tiempo t como la variable independiente y se define la señal como una función del tiempo que contiene información acerca de una cantidad de interés.

La variable x puede ser cualquier variable diferente al tiempo, por ejemplo, los datos grabados en una cinta magnética se pueden considerar una señal que es función de la posición x en la cinta, en relación a un punto de referencia.

2.1.1 Señales continuas y señales discretas

Una *señal continua* o *señal analógica* es aquella que está definida para cada valor del tiempo y toma valores en un intervalo continuo que puede ser desde $-\infty$ hasta ∞. Las *señales discretas* son aquellas que están definidas solamente en valores específicos de tiempo. En la Figura 2.1 se muestra un ejemplo de estas señales.

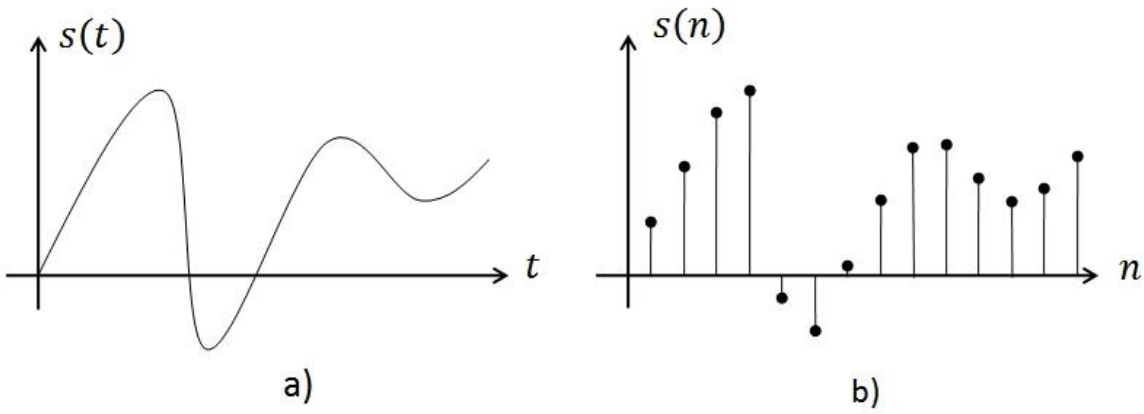


Figura 2.1. Señales: a) continua, b) discreta.

2.1.2 Señales periódicas y señales no periódicas

Las señales también se pueden clasificar en *periódicas* y *no periódicas*, siendo las señales *periódicas* aquellas en las cuales su forma se repite exactamente después de un ciclo (intervalo de tiempo o distancia, por ejemplo), matemáticamente se definen mediante la ecuación (2.1)

$$f(x) = f(x \pm nT) \qquad n = 1,2,\ldots \qquad (2.1)$$

T es el periodo en el que se repite la señal. Las señales *no periódicas* no satisfacen la ecuación (2.1). En la Figura 2.2 se muestra un ejemplo de cada una.

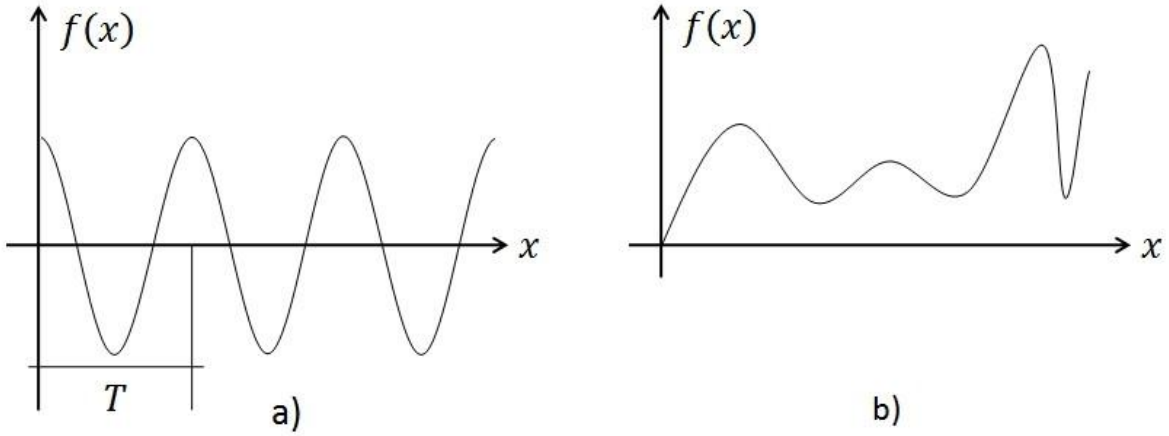


Figura 2.2. Señales: a) periódica, b) no periódica

2.1.3 Señales determinísticas y señales aleatorias

Las señales se pueden clasificar desde el punto de vista matemático en *señales determinísticas* y *señales aleatorias*. Las señales *determinísticas* son aquellas cuyos valores son especificados para cualquier punto del dominio y por ello, pueden modelarse con funciones matemáticas. Las señales *aleatorias* son aquellas que toman valores al azar en cualquier punto del dominio y deben ser caracterizadas estadísticamente [5].

2.2 Señales continuas

Este tipo de señales son las más comunes en el mundo y generalmente se trabaja con el tiempo *t* como la variable independiente, sin embargo, se pueden utilizar otras variables como la distancia.

2.2.1 Análisis temporal

Algunas señales como el campo electromagnético, la presión y el voltaje cambian respecto al tiempo, es decir, son funciones del tiempo. La forma más sencilla de este tipo de señales es la senoidal, la cual se caracteriza por una amplitud a y una frecuencia constante ω_0, como se muestra en la ecuación (2.2)

$$s(t) = a \cos \omega_0 t \qquad (2.2)$$

La energía o intensidad de una señal generalmente es $|s(t)|^2$ para un instante dado. Por ello a $|s(t)|^2$ se le llama densidad de energía o energía instantánea; entonces, en un intervalo de tiempo Δt, se necesita $|s(t)|^2 \Delta t$ cantidad de energía para producir la señal en ese tiempo [3]. Para obtener la energía total se suman o integran las energías instantáneas, $|s(t)|^2$, en todo el tiempo, dada por la ecuación (2.3)

$$\mathrm{E} = \int_{-\infty}^{\infty} |\mathrm{s(t)}|^2 \mathrm{dt} \qquad (2.3)$$

2.2.2 Transformada De Laplace

Esta transformada es una herramienta útil para el análisis y representación de sistemas lineales invariantes en el tiempo continuo, ya que relaciona las entradas y salidas de un sistema de forma algebraica.

La transformada de Laplace de una función se define como

$$\mathrm{X(s)} = \int_0^\infty \mathrm{x(t)e^{-st}dt} \qquad (2.4)$$

donde $X(s)$ es una función compleja de la variable compleja s. Para que la transformada exista es necesario que exista por lo menos un número σ tal que

$$\int_0^\infty \mathrm{|x(t)e^{-\sigma t}dt|} < \infty \qquad (2.5)$$

La transformada de Laplace cumple con varias propiedades, entre las más importantes están:

- Linealidad, es decir

$$\mathrm{ax(t) + bv(t)} \xrightarrow{\text{Laplace}} \mathrm{aX(s) + bV(s)} \qquad (2.6)$$

donde a y b son constantes, $x(t)$ y $v(t)$ son las funciones originales, $X(s)$ y $V(s)$ son las transformadas de Laplace de cada una de las funciones originales y $\xrightarrow{Laplace}$ indica transformada de Laplace.

- Corrimiento en tiempo; consiste en que si la transformada de $x(t)$ es $X(s)$ y c es una constante

$$\mathrm{x(t-c)} \xrightarrow{\text{Laplace}} \mathrm{e^{-cs}X(s)} \qquad (2.7)$$

2.2.3 Convolución

La convolución es una operación matemática que toma dos funciones $x(t)$ y $v(t)$ y produce una tercera $x(t) * v(t)$ que, en cierto modo, representa la cantidad de traslape existente entre $x(t)$ y una versión girada sobre el eje $t = 0$ y trasladada de $v(t)$, como se observa en la siguiente ecuación

$$x(t) * v(t) = \int_{-\infty}^{\infty} \mathrm{x}(\lambda)\mathrm{v}(\mathrm{t}-\lambda)\mathrm{d}\lambda \tag{2.8}$$

para dos funciones $x(t)$ y $v(t)$ causales, es decir, que valen cero para $t < 0$, el limite inferior de integración puede sustituirse por cero; ahora bien, el limite superior de integración suele ser ∞ ya que usualmente las funciones están definidas para todos los números positivos, entonces

$$x(t) * v(t) = \int_{0}^{\infty} \mathrm{x}(\lambda)\mathrm{v}(\mathrm{t}-\lambda)\mathrm{d}\lambda \tag{2.9}$$

Aplicando la transformada de Laplace a la convolución obtenemos

$$x(t) * v(t) \xrightarrow{\text{Laplace}} \int_{0}^{\infty} \left[\int_{0}^{\infty} \mathrm{x}(\lambda)\mathrm{v}(\mathrm{t}-\lambda)\mathrm{d}\lambda\right] \mathrm{e}^{-\mathrm{st}}\mathrm{dt} \tag{2.10}$$

acomodando los términos

$$x(t) * v(t) \xrightarrow{\text{Laplace}} \int_{0}^{\infty} \mathrm{x}(\lambda) \left[\int_{0}^{\infty} \mathrm{v}(\mathrm{t}-\lambda)\mathrm{e}^{-\mathrm{st}}\mathrm{dt}\right] \mathrm{d}\lambda \tag{2.11}$$

ahora considerando una nueva variable $\tau = t - \lambda$ obtenemos

$$x(t) * v(t) \xrightarrow{Laplace} \int_{0}^{\infty} x(\lambda) \left[\int_{-\lambda}^{\infty} v(\tau) exp\big(-s(\tau+\lambda)\big) d\tau\right] d\lambda$$

$$= \int_{0}^{\infty} x(\lambda) \left[\int_{0}^{\infty} v(\tau) exp\big(-s(\tau+\lambda)\big) d\tau\right] d\lambda,$$

ya que $v(\tau) = 0$ para todo $\tau < 0$.

$$= \int_{0}^{\infty} x(\lambda) \left[\int_{0}^{\infty} v(t) exp(-s\tau) exp(-s\lambda) d\tau\right] d\lambda$$

$$= \left[\int_{0}^{\infty} x(\lambda) e^{-s\lambda} d\lambda\right] \left[\int_{0}^{\infty} v(\tau) e^{-s\tau} d\tau\right] \tag{2.12}$$

finalmente

$$x(t) * v(t) \xrightarrow{Laplace} X(s)V(s) \tag{2.13}$$

Entonces la transformada de Laplace de la convolución de dos señales $x(t)$ y $v(t)$ es igual al producto de sus respectivas transformadas de Laplace $X(s)$ y $V(s)$, esto se conoce como *teorema de la convolución* [3].

2.2.4 Análisis espectral

Existen cuatro razones principales para el análisis de frecuencias o análisis espectral de las señales [3]. En *primer* lugar, al analizar espectralmente una señal podemos aprender más acerca de la fuente que la produce, esto se ha utilizado para conocer la composición de las estrellas, el papel, la sangre y casi cualquier sustancia.

La *segunda* razón es que la velocidad de propagación de las señales a través de un medio depende de su frecuencia, por ello la señal se descompone en sus diferentes frecuencias y se prueba cada frecuencia en el medio para estudiar su propagación. Para este tipo de estudios se necesita descomponer la señal en frecuencias individuales y esto se logra mediante el análisis de Fourier.

La *tercera* razón es que el análisis espectral puede simplificar mucho el entendimiento de las señales. En general, una señal parece un desorden, pero muchas veces este desorden es en realidad una superposición de ondas senoidales las cuales son más fáciles de entender y caracterizar.

Por *último*, el análisis de Fourier es una herramienta matemática muy poderosa para la solución de ecuaciones diferenciales.

2.2.5 Serie de Fourier

En 1807 Fourier anunció una tesis que resultó ser muy importante para el análisis matemático y sus aplicaciones en ingeniería [5]. Fourier enunció que una función (o señal $f(x)$) periódica de variable real x, definida en el intervalo $[-l, l]$, se puede representar en ese intervalo como una serie infinita de funciones seno y coseno

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left[a_k \cos\left(\frac{k\pi x}{l}\right) + b_k \operatorname{sen}\left(\frac{k\pi x}{l}\right) \right] \qquad (2.14)$$

en donde a_k y b_k son coeficientes reales, independientes de x, y se pueden obtener mediante las expresiones

$$a_k = \frac{1}{l}\int_{-l}^{l} f(x)\cos\left(\frac{k\pi x}{l}\right)dx \qquad k = 0,1,2,\ldots \tag{2.15a}$$

$$b_k = \frac{1}{l}\int_{-l}^{l} f(x)\text{sen}\left(\frac{k\pi x}{l}\right)dx \qquad k = 0,1,2,\ldots \tag{2.15b}$$

La única restricción que puso Fourier para la función (o señal) $f(x)$ es que sea integrable en el intervalo $[-l, l]$, esto se puede ver en la definición de los coeficientes. Esta tesis se conoce ahora como la *Teoría de Fourier*.

Estas expresiones se pueden simplificar normalizando el intervalo para el cual $f(x)$ está definido $[-l, l]$ al intervalo $[-\pi, \pi]$. Esto lo hacemos con la transformación mostrada en la ecuación (2.16)

$$x \to \left(\frac{l}{\pi}\right)\theta \tag{2.16}$$

derivando tenemos

$$dx \to \left(\frac{l}{\pi}\right)d\theta \tag{2.17}$$

y

$$\theta = \pm\pi \text{ cuando } x = \pm 1 \tag{2.18}$$

Entonces, (2.14) y (2.15) se convierten en

$$f(\theta) = \frac{a_0}{2} + \sum_{k=1}^{\infty}[a_k\cos k\theta + b_k\text{sen } k\theta] \tag{2.19}$$

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(\theta)\cos k\theta \, d\theta \tag{2.20a}$$

$$b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(\theta)\text{sen}k\theta \, d\theta \tag{2.20b}$$

La variable independiente θ es un símbolo matemático, sin embargo, las señales procesadas son una función de una variable física como el tiempo o el espacio. Suponiendo que estamos trabajando con una señal que es función del tiempo, $f(t)$, podemos utilizar las expresiones (2.19) y (2.20) si utilizamos la igualdad (2.21)

$$\theta = \omega_0 t \tag{2.21}$$

donde ω_0 es una constante cuyas unidades son *radianes por segundo*. Asumiendo que $f(t)$ es periódica en el tiempo con período *T*,

$$f(t) = f(t \pm rT) \qquad r = 1,2,\ldots \tag{2.22}$$

podemos escribir la serie de Fourier en (2.19) como se ve en (2.23)

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} [a_k \cos k\omega_0 t + b_k \operatorname{sen} k\omega_0 t] \tag{2.23}$$

y como el período en la variable θ es de 2π, y en la variable t es de T, tenemos que cuando $t = T$ y $\theta = 2\pi$,

$$\omega_0 = \frac{2\pi}{T} \tag{2.24}$$

ω_0 es llamada la *frecuencia fundamental* mientras que $k\omega_0$ con $k = 2,3,\ldots$ es llamada la *armónica de orden k*. Los coeficientes a_k y b_k los obtenemos sustituyendo (2.21) en (2.20)

$$a_k = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \cos(k\omega_0 t) dt \tag{2.25a}$$

$$b_k = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \operatorname{sen}(k\omega_0 t) dt \tag{2.25b}$$

2.2.6 Serie exponencial de Fourier

Se puede utilizar una forma equivalente y más compacta de la serie de Fourier para representar a las señales. Utilizamos las identidades (2.26) y las sustituimos en la ecuación (2.19)

$$\cos(k\theta) = \frac{1}{2} [\exp(jk\theta) + \exp(-jk\theta)] \tag{2.26a}$$

$$\operatorname{sen}(k\theta) = \frac{1}{2j} [\exp(jk\theta) - \exp(-jk\theta)] \tag{2.26b}$$

obtenemos

$$f(\theta) = \frac{a_0}{2} + \frac{1}{2} \sum_{k=1}^{\infty} [(a_k - jb_k) \exp(jk\theta) + (a_k + jb_k) \exp(-jk\theta)] \tag{2.27}$$

llamemos c_k al coeficiente complejo

$$c_k = \frac{1}{2}(a_k - b_k) \qquad k = 0,1,2,\ldots \tag{2.28}$$

y

$$c_k^* = \frac{1}{2}(a_k + b_k) \tag{2.29}$$

a su conjugado complejo. Entonces la ecuación (2.27) se puede escribir como

$$f(\theta) = c_0 + \sum_{k=1}^{\infty}[c_k \exp(jk\theta)] + \sum_{k=1}^{\infty}[c_k \exp(-jk\theta)] \tag{2.30}$$

en donde

$$c_0 = \frac{1}{2}a_0 \tag{2.31}$$

Utilizando las expresiones (2.20) en la definición de c_k y de c_k^* en las ecuaciones (2.28) y (2.29) tenemos

$$c_k = \frac{1}{2\pi}\int_{-\pi}^{\pi}\big(\cos(k\theta) - j\text{sen}(k\theta)\big)f(\theta)d\theta \tag{2.32}$$

y utilizando las identidades en (2.33)

$$\exp(jk\theta) = \cos(k\theta) + j\text{sen}(k\theta) \tag{2.33a}$$

$$\exp(-jk\theta) = \cos(k\theta) - j\text{sen}(k\theta) \tag{2.33b}$$

obtenemos

$$c_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(\theta)\exp(-jk\theta)d\theta \tag{2.34}$$

$$c_k^* = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(\theta)\exp(jk\theta)d\theta \tag{2.35}$$

$$c_k^* = c_{-k} \tag{2.36}$$

y llegamos a la ecuación (2.37)

$$f(\theta) = c_0 + \sum_{k=-\infty}^{\infty}[c_k \exp(jk\theta)] \tag{2.37}$$

que es la serie compleja de Fourier de $f(\theta)$. Cuando trabajamos en el dominio del tiempo podemos utilizar la igualdad (2.21) y (2.24) y obtenemos

$$f(t) = c_0 + \sum_{k=-\infty}^{\infty} [c_k \exp(jk\omega_0 t)] \tag{2.38}$$

$$c_k = \frac{\omega_0}{2\pi} \int_{-\pi/\omega_0}^{\pi/\omega_0} f(t) \exp(-jk\omega_0 t) d\theta \tag{2.39}$$

Como ya se dijo, ω_0 es la *frecuencia fundamental* mientras que $\omega_k = k\omega_0$ es la *armónica de orden k*. Si estudiamos c_k en función de $k\omega_0$, tenemos la representación en frecuencia de la señal $f(t)$. En la Figura 2.3 se muestra un ejemplo de una señal variante en el tiempo así como su espectro tanto en amplitud como en fase. La señal es una suma de tres señales senoidales de diferentes frecuencias (100, 150 y 300 Hz) y una constante.

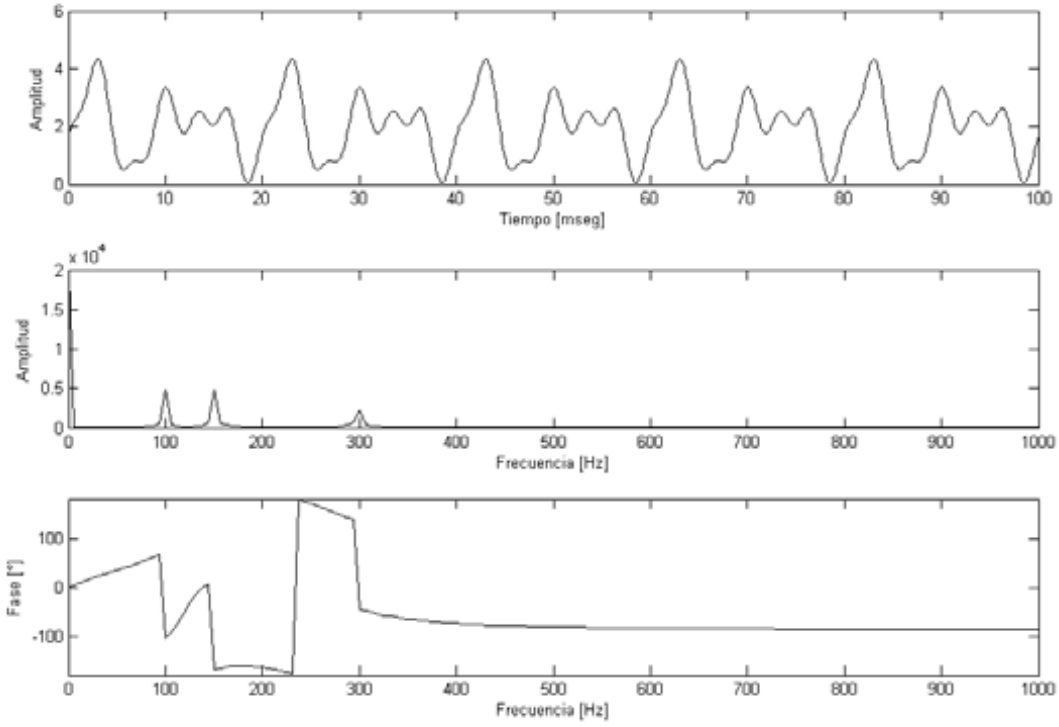


Figura 2.3. Señal variable en el tiempo, espectro en amplitud y espectro en fase.

2.3 Análisis de señales discretas

Existen muchas razones por lo que es más adecuado trabajar con señales digitales que con señales analógicas. *Primero*, un sistema programable digital es muy flexible en su reconfiguración, ya que esta se realiza simplemente cambiando el programa. En cambio para reconfigurar un sistema analógico usualmente hay que rediseñar hardware, realizar pruebas y verificar su operación.

La *exactitud* juega también un papel importante en la determinación del tipo de señales a utilizar; los procesadores digitales proveen un mejor control sobre los requerimientos de exactitud; las tolerancias de los componentes de los circuitos eléctricos hacen muy complicado al diseñador del sistema controlar la exactitud de dicho sistema. En el sistema digital la exactitud puede controlarse simplemente con la modificación del tamaño de palabra en el convertidor analógico digital, o cambiando la forma de los datos entre punto fijo o flotante.

Las señales digitales son también más *fáciles de almacenar* en dispositivos de estado sólido y medios magnéticos sin deterioro o perdida de calidad de las señales además de la introducida en la conversión analógica digital; en consecuencia una señal digital es *fácil de transportar* y por tanto se puede procesar fuera de línea en un laboratorio remoto. El procesamiento digital permite el uso de *algoritmos muy sofisticados*; usualmente es muy difícil realizar operaciones matemáticas precisas con electrónica analógica, mismas que son muy fáciles de realizar en una computadora digital mediante software.

En algunos casos la implementación de un sistema digital es además *más barata* que su contraparte analógica; esto puede deberse a que el hardware digital es más barato ya que en caso de ser necesaria una modificación, el sistema digital es más flexible [5].

2.3.1 Conversión analógica digital

Una señal en tiempo discreto solo está definida en determinados tiempos, llamados tiempos de muestreo, si estos valores son cuantizados y codificados obtenemos una señal digital. Una señal digital se obtiene mediante el proceso llamado conversión analógica digital; los errores inherentes a la conversión deben ser identificados para considerarlos posteriormente.

Un diagrama de bloques de la conversión analógica digital se muestra en la figura 2.4. Su primer componente es un muestreador cuya función es extraer muestras de la señal analógica en los tiempos de muestreo. La salida de este muestreador es una señal de tiempo discreto pero con amplitud continua, ya que los valores que la señal muestreada asume están en el mismo intervalo de la señal analógica. El segundo elemento del convertidor analógico digital es el cuantizador que asocia a la señal muestreada cantidades pertenecientes a un conjunto finito de valores que pueden ser representados por palabras digitales de tamaño definido. El codificador sirve para mapear cada uno de estos valores cuantizados en palabras digitales. A continuación se describe de manera detallada cada parte del proceso.

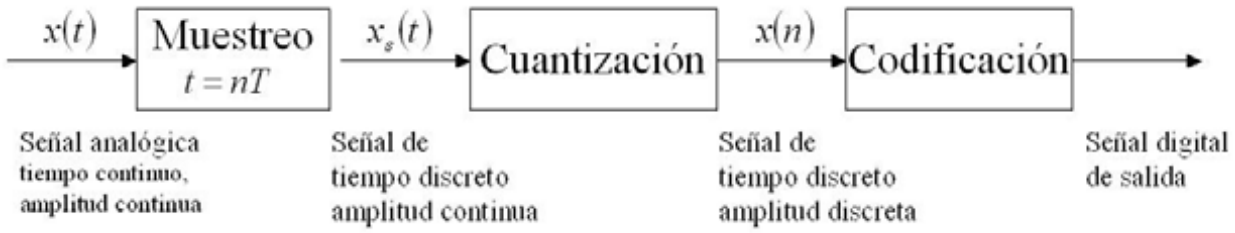


Figura 2.4. Conversión analógica digital

2.3.2 Muestreador.

Muestrear una señal continua $x(t)$ es representarla en un número discreto de puntos, en tiempos $t = nT$, donde T es el período de muestreo, que es el tiempo entre muestras, y n es un número entero que establece la posición de cada muestra. La forma más sencilla de entender este concepto es pensar en un interruptor.

Para extraer muestras de $x(t)$, el interruptor se cerrará por un tiempo muy corto cada T segundos. Obteniendo de esta forma muestras con valor $x_s(t)$ cuando el interruptor está cerrado y cero cuando el interruptor está abierto. Para que este proceso sea útil necesitamos que sea posible reconstruir $x(t)$ a partir de las muestras. Las condiciones necesarias para que esto se logre pueden obtenerse de manera sencilla en el dominio de la frecuencia.

Primero la señal muestreada de $x(t)$, llamada $x_s(t)$, se escribe como :

$$x_s(t) = x(t)p(t) \tag{2.40}$$

donde $p(t)$, llamada la función de muestreo, es el modelo de la acción del interruptor de muestreo. La función de muestreo es usualmente un tren de impulsos de periodo T. Ahora encontraremos el espectro de $x_s(t)$ y seleccionaremos los valores apropiados para T.

El primer paso para encontrar el espectro de $x_s(t)$ es darnos cuenta que $p(t)$ es una función periódica y puede ser representada como una serie de Fourier. En otras palabras.

$$p(t) = \sum_{n=-\infty}^{\infty} c_n \exp(jn2\pi f_s) \tag{2.41}$$

donde c_n es el n-esimo coeficiente de Fourier de $p(t)$ y está dado por:

$$c_n = \frac{1}{T}\int_{-T/2}^{T/2} p(t)\exp(-jn2\pi f_s t)dt \tag{2.42}$$

En las ecuaciones (2.41) y (2.42), f_s es la frecuencia fundamental de $p(t)$, que es la frecuencia de muestreo, y está dada por

$$f_s = \frac{1}{T}[Hz] \tag{2.43}$$

Como $x_s(t)$ es el producto de $x(t)$ con $p(t)$, tenemos

$$x_s(t) = \sum_{n=-\infty}^{\infty} c_n x(t)\exp(jn2\pi f_s t) \tag{2.44}$$

Ahora podemos obtener el espectro de $x_s(t)$ al que denotaremos $X_s(f)$ aplicando la transformada de Fourier a la ecuación (2.44), es decir

$$X_s(f) = \int_{-\infty}^{\infty} x_s(t)\exp(-j2\pi f t)dt \tag{2.45}$$

sustituyendo la ecuación (2.44) en lugar de $x_s(t)$ se convierte en

$$X_s(f) = \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} c_n x(t)\exp(jn2\pi f_s t)\exp(-j2\pi f t)dt \tag{2.46}$$

Intercambiando los operadores suma e integración obtenemos

$$X_s(f) = \sum_{n=-\infty}^{\infty} c_n \int_{-\infty}^{\infty} x(t)\exp(-j2\pi(f - nf_s)t)dt \tag{2.47}$$

Por la definición de la transformada de Fourier

$$X(f - nf_s) = \int_{-\infty}^{\infty} x(t)\exp(-j2\pi(f - nf_s)t)dt \qquad (2.48)$$

Por lo tanto podemos escribir la transformada de Fourier de la señal muestreada como

$$X_s(f) = \sum_{-\infty}^{\infty} c_n X(f - nf_s) \qquad (2.49)$$

Esta última ecuación muestra que el espectro de una señal de tiempo continua muestreada, es el espectro de la misma señal, $x(t)$, desplazado en múltiplos de la frecuencia de muestreo. Cada espectro trasladado está multiplicado por una constante, correspondiente a la expansión de la serie de Fourier para $p(t)$ [4]. El proceso descrito se muestra en la figura 2.5.

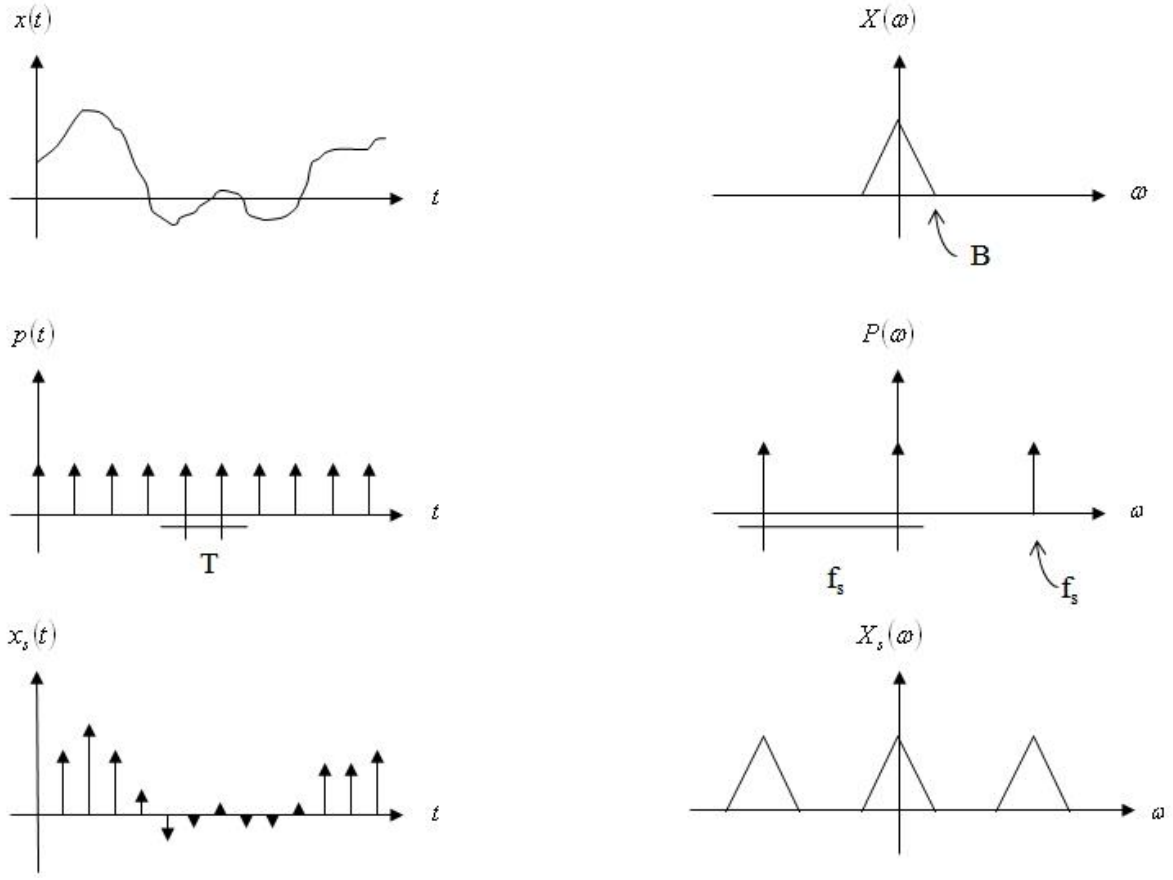


Figura 2.5. Muestreo

2.3.3 Modelo de muestreo con tren de impulsos

En el apartado anterior se consideró una señal de muestreo genérica con la característica de ser periódica. En la práctica el tiempo en que $p(t)$ no es cero, es el ancho de un pulso pequeño comparado con el periodo T. Un ancho de pulso extremadamente angosto es usado en los sistemas digitales por la simplificación matemática que resulta del modelo de la función impulso. Asumamos que nuestra función de muestreo es un tren de impulsos con periodo T. Es decir

$$p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT) \tag{2.50}$$

Entonces los valores de c_n pueden calcularse con la ecuación (2.42) como se muestra a continuación

$$c_n = \frac{1}{T} \int_{-T/2}^{T/2} \delta(t) \exp(-jn2\pi f_s t) dt \tag{2.51}$$

que es

$$c_n = \frac{1}{T} = f_s \tag{2.52}$$

por la propiedad de corrimiento de la función impulso o función delta. Entonces c_n es igual a f_s para toda n en la expresión del espectro de la señal $x(t)$ muestreada; entonces la ecuación (2.49), se convierte en

$$X_s(f) = f_s \sum_{n=-\infty}^{\infty} X(f - nf_s) \tag{2.53}$$

Al muestrear con un tren de impulsos logramos que todos los espectros trasladados tengan la misma amplitud.

2.3.4 Teorema del muestreo

Para determinar la frecuencia a la que debemos muestrear, consideremos una señal arbitraria continua, $x(t)$, limitada en banda, es decir en la que $|X(f)| = 0$ para $|f| > B$; donde $X(f)$ es la

transformada de Fourier de $x(t)$ y B es el ancho de banda de $x(t)$; el espectro de $x_s(t)$ muestreada con $p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$ se muestra en la figura 2.6.

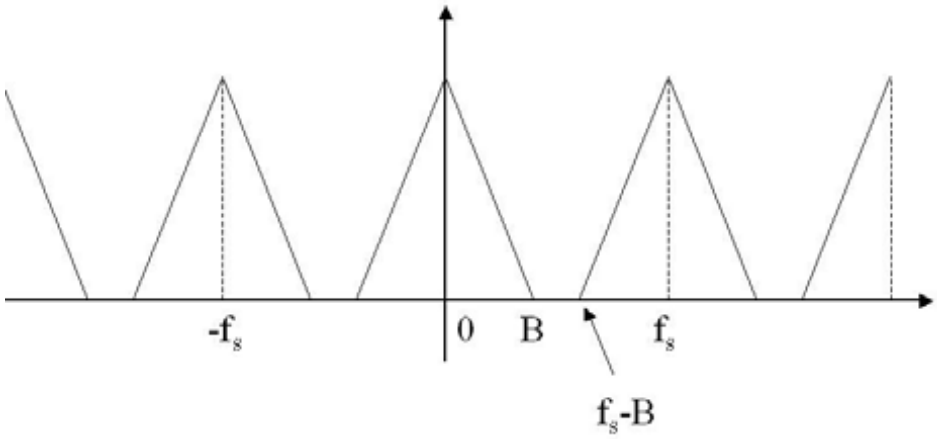


Figura 2.6. Espectro de una señal muestreada con tren de impulsos.

Es evidente en la figura 2.6 que para que $X(f)$ pueda ser reconstruida a partir de $X_s(f)$ es necesario que

$$f_s - B \geq B \tag{2.54}$$

o

$$f_s \geq 2B \tag{2.55}$$

Esto quiere decir que la frecuencia mínima de muestreo debe ser $2B$.

El teorema del muestreo suele enunciarse de la siguiente forma: *Una señal limitada en banda $x(t)$, que no tiene componentes de frecuencia mas allá de B Hz , está completamente especificada por muestras que estén tomadas con una razón uniforme mayor a $2B$ Hz* . En otras palabras, *el tiempo entre las muestras no debe ser mayor a $1/2B$ segundos*. Este teorema es llamado también *teorema de Nyquist o Shannon*.

La frecuencia $2B$ es conocida como frecuencia de Nyquist.

2.3.5 Cuantización y codificación

Cuantizar un valor muestreado es aproximarlo hacia el valor más cercano de un conjunto finito de valores permisibles. Codificar es representar cada uno de los valores permitidos en palabras digitales de un tamaño de palabra específico. Para una representación binaria tenemos que el número de niveles q y el tamaño de palabra digital n están relacionados por

$$q = 2^n \qquad (2.56)$$

Como el valor cuantizado es el único valor retenido después del muestreo, el proceso de cuantización genera errores que no se pueden eliminar mediante procesamiento posterior. Este error puede medirse fácilmente ya que el valor muestreado se redondeará hacia el nivel más cercano dando como resultado que el error por cuantizar una muestra sea de $\pm(1/2)S$, donde S es el paso de cuantización. Se observa que este error puede reducirse fácilmente aumentando el número de intervalos.

2.4 Señales espaciales

Generalmente se trabaja con señales que son funciones del tiempo, sin embargo, existen algunas señales que es útil trabajarlas como funciones del espacio, es decir, considerar la variación de su magnitud dependiendo de su posición en el espacio con respecto a un punto de referencia. Por ejemplo, una imagen bidimensional es una variación de densidad de color en función de la posición en el espacio. Si se fijan un punto y una dirección dentro de la imagen, la frecuencia será el número de aumentos y decrementos de la densidad por unidad de distancia. Por lo tanto, en lugar de frecuencia temporal, podemos hablar de *frecuencia espacial* medida en unidades de distancia, por ejemplo *ciclos/metro*. El análisis hecho para el tiempo se puede aplicar para el espacio, con los cambios apropiados de unidades. Estas señales también pueden ser bidimensionales o tridimensionales.

Algunos ejemplos de señales espaciales son la temperatura, la intensidad de campo, la tensión y el sonido.

2.4.1 El sonido

El sonido es generado por la vibración de un cuerpo material y se propaga a través de un medio. El sonido no se propaga en el vacío, por ello es necesario un medio (sólido, líquido o gaseoso) para que exista. El sonido se propaga como una onda longitudinal (la oscilación, compresiones y rarefacciones, se propaga en forma paralela a la dirección de propagación de la onda), por lo tanto se caracteriza por las propiedades de las ondas que son frecuencia, longitud de onda, período, amplitud, velocidad y dirección. La frecuencia es el número de ciclos (ondas completas) que se producen o que se reciben por unidad de tiempo, y el intervalo en el que los seres humanos podemos percibir el sonido va desde los $20\ Hz$ hasta los $20\ 000\ Hz$; la longitud de onda es la distancia que recorre la onda durante un período; el período es el tiempo que tarda cada ciclo en repetirse; la amplitud indica la cantidad de energía que contiene una señal sonora. La velocidad del sonido depende del tipo de material en el que se propaga. Cuando el sonido se desplaza en los sólidos tiene mayor velocidad que en los líquidos, y en los líquidos es más veloz que en los gases, lo cual se debe a que las partículas en los sólidos están más cercanas. La velocidad de propagación del sonido en el aire es de $343\ m/s$ a 20℃, pero varía con la temperatura.

2.5 La voz

La voz cumple con las condiciones establecidas para considerarse sonido, las cuerdas vocales son el cuerpo elástico que vibra, el aire es el medio en el que se propagan las ondas de presión del sonido y la caja de resonancia está formada por la garganta, la boca y la cavidad nasal. Los pulmones comprimen aire que excita a la glotis, haciendo vibrar los dos pares de cuerdas vocales. Las cavidades relacionadas con el sistema respiratorio y nasofaríngeo, actúan como resonadores, la variación de la intensidad depende de la fuerza de la espiración.

La voz se compone de dos tipos de sonidos: sonidos tonales y sonidos no tonales. Los sonidos *tonales* o *sonoros* son generados cuando las cuerdas vocales vibran. Todas las vocales son tonales, pero también algunas consonantes son de este tipo: “b”, “d”, “m”, etc. Cuando las

cuerdas vocales no vibran, el sonido que se genera por el paso de aire a través del tracto vocal se llama sonido *no tonal* o *sordo* [8].

En el hombre las cuerdas vocales son más largas y más gruesas que en la mujer y el niño, por lo que produce sonidos más graves. El tono y la intensidad del habla están determinados principalmente por la vibración de las cuerdas vocales y su espectro está determinado por las resonancias del tracto vocal. Para que el habla sea comprensible, es indispensable la presencia de la frecuencia fundamental de la voz y armónicos cuya frecuencia se halla entre 300 y $3400\ Hz$. Por otra parte, la energía de la voz está contenida en su mayor parte en las frecuencias bajas, su supresión resta potencia a la voz que suena delgada y con poca energía. A la frecuencia fundamental también se le llama frecuencia tonal o simplemente tono; en los hombres, el tono se encuentra entre 50 y $250\ Hz$, mientras que para las mujeres el intervalo se encuentra entre 120 y $500\ Hz$ [8]. En la Figura 2.7 se muestra una señal de voz en el tiempo y su espectro en amplitud.

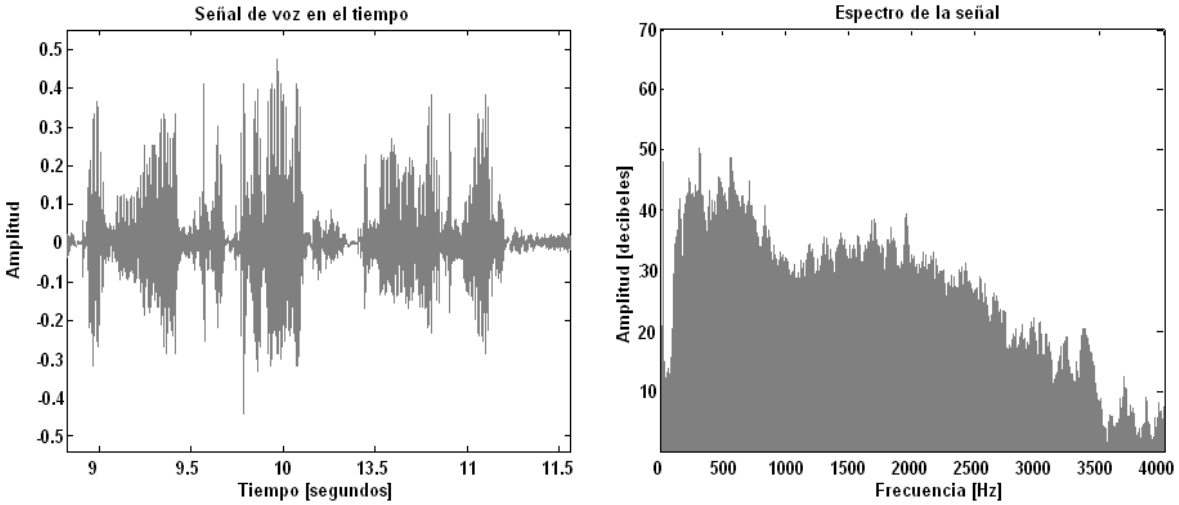


Figura 2.7. Señal de voz en el tiempo y su espectro.

2.5.1 Procesamiento de voz

Desde la invención del teléfono en el siglo diecinueve y después, en el siglo veinte, con la invención de la computadora digital, la productividad y eficiencia en el trabajo del hombre ha aumentado. Con la aplicación de diferentes tecnologías se ha podido enseñar a las computadoras a hablar e incluso a escuchar en diferentes idiomas, con lo cual se han producido nuevas

aplicaciones que permiten a las personas no solo comunicarse rápida y fácilmente a distancia, sino también permiten que las computadoras se comuniquen por teléfono para obtener y actualizar información. Esto se logra mediante un conjunto de técnicas a las cuales se les llama *procesamiento de voz* [9].

Este tipo de técnicas ha logrado que la unión entre computadoras y telecomunicaciones sea simple y de bajo costo, requiriendo solamente un teléfono básico, sin embargo, no todas las aplicaciones del procesamiento de voz implican telecomunicaciones; las mismas tecnologías se pueden encontrar en calculadoras, diccionarios y traductores, juguetes, máquinas expendedoras, automóviles y ayudas auditivas.

El procesamiento de voz se puede dividir en las siguientes categorías:

- Reconocimiento de voz; se ocupa del análisis del contenido lingüístico de una señal de voz.
- Filtrado; se encarga de mejorar la calidad de la voz, por ejemplo, reduciendo el ruido presente.
- Codificación de voz; se utiliza para compresión de datos y transmisión de la voz.
- Análisis de voz con propósitos médicos, para el análisis de disfunciones vocales.
- Síntesis de voz; la síntesis artificial del habla, lo que habitualmente significa habla generada por computadora.

2.6 Resumen

El análisis de señales se dedica al estudio y caracterización de las propiedades de las señales, las cuales son, generalmente, funciones de varias variables. Las señales se caracterizan principalmente por su variación en el tiempo, pero para entenderlas mejor, es bueno estudiarlas desde representaciones diferentes, por ejemplo, la frecuencia.

Otra forma importante de representar las señales es su representación discreta, la cual nos permite almacenarlas sin que sufran deterioro durante el proceso de almacenamiento ni pérdida de información debida al uso.

Las señales son estudiadas principalmente en función del tiempo, pero algunas señales también se pueden estudiar en función de otras variables, como el espacio. El sonido, la voz y el

audio son ejemplos de señales que pueden ser estudiadas en función del tiempo así como en función del espacio.

La voz se compone de sonidos tonales y sonidos no tonales. Los sonidos *tonales* o *sonoros* son generados cuando las cuerdas vocales vibran. Cuando las cuerdas vocales no vibran, el sonido que se genera por el paso de aire a través del tracto vocal se llama sonido *no tonal* o *sordo*.

A las técnicas que han logrado la unión entre computadoras y telecomunicaciones se les llama procesamiento de voz y, además de utilizarse en las telecomunicaciones también se utilizan en aparatos como calculadoras, traductores y ayudas auditivas.

El procesamiento de voz se puede dividir en diferentes categorías como el reconocimiento, la mejora de la señal y la síntesis de voz.

Capítulo 3

Multisensado y formación de haz

En este capítulo se describe el proceso de multisensado que consiste en la adquisición de señales por medio de un conjunto de sensores y nos sirve para estudiar las características espaciales de las señales. Además se analizan los sistemas adaptables que son sistemas que utilizan ciertos algoritmos para cambiar los coeficientes de su estructura y ajustarse a las señales de entrada y al medio en el que se encuentran. También se describe el algoritmo LMS el cual es muy usado en los sistemas adaptables debido a su simplicidad de programación y su eficacia.

Uno de los objetivos del presente trabajo es la adquisición simultánea de hasta cuatro señales de voz, provenientes de una misma fuente, a las que se les aplicará el proceso llamado formador de haz. La formación de haz o Beamforming es un proceso que se utiliza para enviar o recibir señales en alguna dirección específica e ignorar o cancelar señales provenientes de cualquier otra dirección.

3.1 Multisensado

El multisensado de señales consiste en adquirir señales por medio de varios sensores organizados en patrones o arreglos, obteniendo así información sobre el comportamiento espacial de las señales. Las aplicaciones más comunes de los arreglos de sensores se encuentran en el campo de la acústica y van desde la detección de señales en medios ruidosos hasta la determinación de la posición del emisor de la señal acústica.

3.1.1 Arreglos de sensores

Un arreglo de sensores es un grupo de sensores ubicados en puntos separados espacialmente y utilizado para filtrar señales en el espacio-tiempo mediante el aprovechamiento de sus

características espaciales. El filtrado puede expresarse como una dependencia de ángulo o número de onda y se realiza mediante la combinación de las salidas de los sensores con ganancias que permiten enfatizar o rechazar señales de acuerdo con su dependencia espacial, usualmente se busca filtrar espacialmente las señales de tal modo que para un ángulo o conjunto de ángulos sean enfatizadas mediante combinación constructiva y que el ruido proveniente de otros ángulos sea rechazado por interferencia destructiva.

El diseño de los arreglos para obtener ciertos desempeños involucra la consideración de varios criterios como son la geometría del arreglo y el número de sensores entre otros; existen dos factores principales que determinan el desempeño de los arreglos como filtros espaciales; *primero*, su geometría establece limitaciones en su forma de operación; los arreglos lineales pueden resolver solamente una componente angular, esto significa que tendremos ambigüedades entre señales que tengan el mismo ángulo respecto al arreglo pero provengan de diferentes direcciones; por otro lado, los arreglos circulares tienen patrones diferentes de los arreglos rectangulares. Frecuentemente la geometría del arreglo es establecida por restricciones físicas y el diseñador tiene libertad limitada en la especificación de la geometría del arreglo. El *segundo* aspecto es la determinación de los factores de amplificación para cada salida de los sensores, estos factores determinan las características de filtrado del arreglo para una geometría dada [10].

3.1.2 Respuestas de arreglos de sensores

Un arreglo de sensores percibe ondas que se propagan a través de un medio. La onda es recibida por todos los sensores y dependiendo de la configuración existen múltiples respuestas del arreglo, en un caso simple todas las componentes de las señales en el arreglo son replicas de la señal fuente. En otros casos las salidas individuales de los sensores están corrompidas por ruido o señales de interferencia con mucha similitud entre señales. Los algoritmos utilizados en procesamiento de señales multisensadas combinan todas las salidas de los sensores para obtener una respuesta deseada, que típicamente consiste en la recepción de la señal de una fuente deseada en forma óptima y el rechazo de otras señales. La apertura del arreglo es una extensión espacial de la distribución de los sensores y está limitada por la resolución [11].

En el análisis de arreglos de sensores es frecuente considerar que las señales recibidas por los sensores son ondas planas, es decir, que cumplen con la siguiente restricción [11]:

$$r = \frac{2L^2}{\lambda} \tag{3.1}$$

donde r es la distancia de la fuente al arreglo, L es la distancia entre los dos sensores extremos del arreglo y λ la longitud de onda de la señal emitida por la fuente.

3.1.3 Arreglo Lineal Uniforme

En la figura 3.1 se muestra un arreglo lineal de M sensores que están recibiendo ondas planas, en el cual la distancia entre sensores es d y el cambio de fase de la onda o el retardo de las señales recibidas cuando arriban a un sensor consecutivo es $d\cos\varphi$. La distancia entre sensores determina la resolución y la dimensión del arreglo.

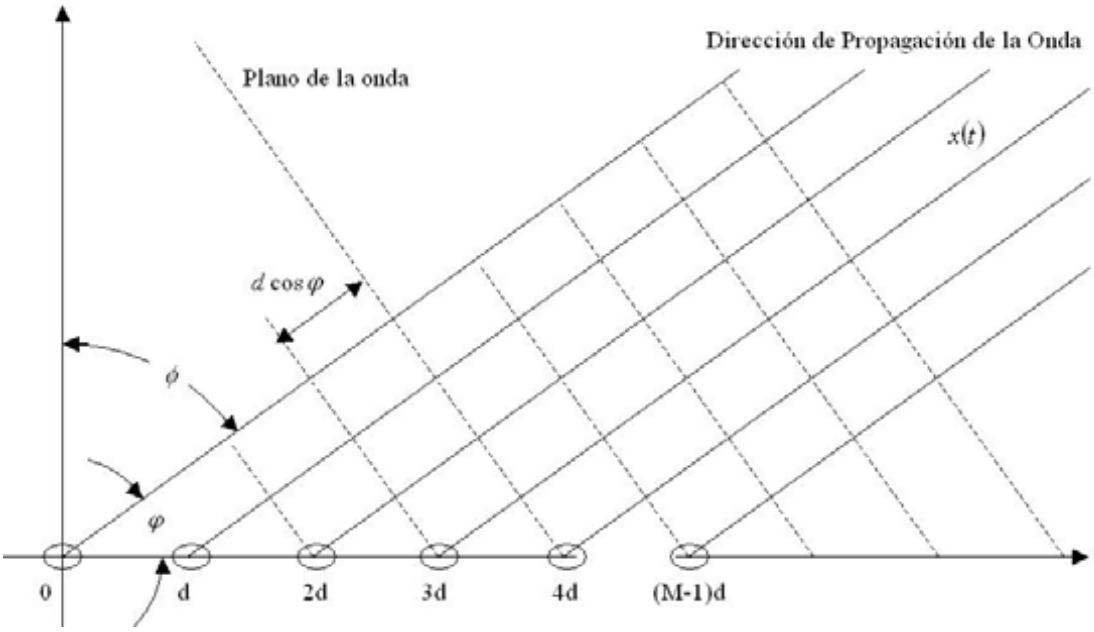


Figura 3.1. Arreglo lineal de sensores

Considerando un arreglo lineal de M sensores omnidireccionales idénticos separados una distancia d, si la señal $x(t)$ llega en cierto instante de tiempo a uno de dichos sensores, al siguiente sensor llegará la misma señal con un retardo de τ.

Las ondas que arriban a los sensores pueden modelarse como ondas planas, con un vector unitario $\bar{u}$ perpendicular al plano de la onda y un vector de posición $\bar{r}_i$ del origen del sistema de referencia a cada sensor, entonces

$$\bar{u} = \text{sen}(\phi \bar{a}_x) + \cos(\phi \bar{a}_y) \tag{3.2}$$

$$\varphi = 90° - \phi \tag{3.3}$$

$$\bar{r}_i = (i-1)d\bar{a}_x \qquad i = 1,2,3,\dots,M \tag{3.4}$$

calculando el producto punto entre los vectores $\bar{u}$ y $\bar{r}_i$ obtenemos

$$\bar{r}_i \cdot \bar{u} = (i-1)d \,\text{sen}\emptyset \qquad i = 1,2,3,\dots,M \tag{3.5}$$

que corresponde a un retardo entre sensores. Para ondas con longitud de onda λ, el cambio de fase se calcula

$$\bar{r}_i \cdot \bar{u} = \frac{2\pi}{\lambda}(i-1)d \,\text{sen}\phi \qquad i = 1,2,3,\dots,M \tag{3.6}$$

Considerando una salida discreta $y(n)$ como la suma de las señales de los sensores del arreglo, y teniendo en cuenta que cada sensor recibe la señal $x(n)$ retardada obtenemos

$$y(n) = x(n) + x(n-\tau) + x(n-2\tau) + \cdots + x(n-(M-1)\tau)$$

$$y(n) = \sum_{i=0}^{M-1} x(n-i\tau) \tag{3.7}$$

Si la señal viaja a velocidad constante v, el retardo temporal puede calcularse

$$\tau = \frac{d \,\text{sen}\phi}{v} \tag{3.8}$$

Supongamos una señal $x(n)$ de banda angosta centrada en la frecuencia f_0, el tiempo de retardo queda definido como

$$\tau = \frac{2\pi d}{\lambda_0}\text{sen}\phi \tag{3.9}$$

y la frecuencia normalizada es $\omega = 1$

Definamos el numero de onda K como

$$K = \frac{2\pi}{\lambda_0} \tag{3.10}$$

y representa la constante de fase de la onda.

Si $x(n)$ es una señal exponencial compleja $x(n) = \exp\left(\frac{j2\pi nf}{f_s}\right)$, propagándose en el medio como una onda plana en dirección $\bar{u}$, entonces el cambio de fase en cada sensor se puede interpretar como $K\bar{r}_i \cdot \bar{u}$, entonces el modelo de la onda es

$$x(n) = e^{j\left(\frac{2\pi nf}{f_s} + K\bar{r}_i \cdot \bar{u}\right)} \qquad i = 1,2,3,\dots,M \tag{3.11}$$

3.1.4 Submuestreo espacial

Una restricción importante para la distancia entre los sensores se debe a la longitud de onda de las señales, ya que si los sensores se encuentran dispuestos a distancias iguales o mayores a esta característica de las ondas incurriremos en submuestreo. Para evitarlo es necesario que la diferencia de fases entre dos sensores contiguos del arreglo sea menor o igual a π, es decir

$$2\pi\tau f \leq \pi \tag{3.12}$$

sustituyendo la ecuación (3.8) en la ecuación (3.12) y despejando d obtenemos

$$d \leq \frac{v}{2f} \tag{3.13}$$

se puede observar en (3.13) que si la frecuencia aumenta la distancia máxima disminuirá, por lo tanto esta restricción se calcula siempre con la frecuencia máxima de la señal de interés.

3.2 Formador de haz fijo

Efectuando la suma sobre todos los sensores y considerando la ecuación (3.11)

$$y(n) = \sum_{i=1}^{M} x(n) \tag{3.14}$$

$$y(n) = \sum_{i=1}^{M} \exp\left[j\left(\frac{2\pi nf}{f_s} + K\bar{r}_i \cdot \bar{u}\right)\right] \tag{3.15}$$

$$y(n) = \exp\left(\frac{j2\pi nf}{f_s}\right) \sum_{i=1}^{M} \exp(jK(i-1)d\ \text{sen}\phi) \tag{3.16}$$

definimos el patrón de haz como

$$b(f,\phi) = \sum_{i=1}^{M} \exp(jK(i-1)d\ \text{sen}\phi) \tag{3.17}$$

resolviendo la sumatoria obtenemos el patrón de haz

$$|b(f,\phi)| = \left|\frac{\text{sen}[(\pi f M d\ \text{sen}\phi)/v]}{\text{sen}[(\pi f d\ \text{sen}\phi)/v]}\right| \tag{3.18}$$

Esto corresponde con el modelo de un *formador de haz fijo* que es el más sencillo y se basa en el hecho de que, si se utiliza un arreglo lineal de sensores, la salida de cada sensor será la misma sólo que cada una tendrá un retraso diferente. Entonces, si se suman las salidas de cada sensor, las señales provenientes de fuentes ubicadas justo enfrente del arreglo se reforzarán. En la figura 3.2 se muestra un esquema de este tipo de formador de haz.

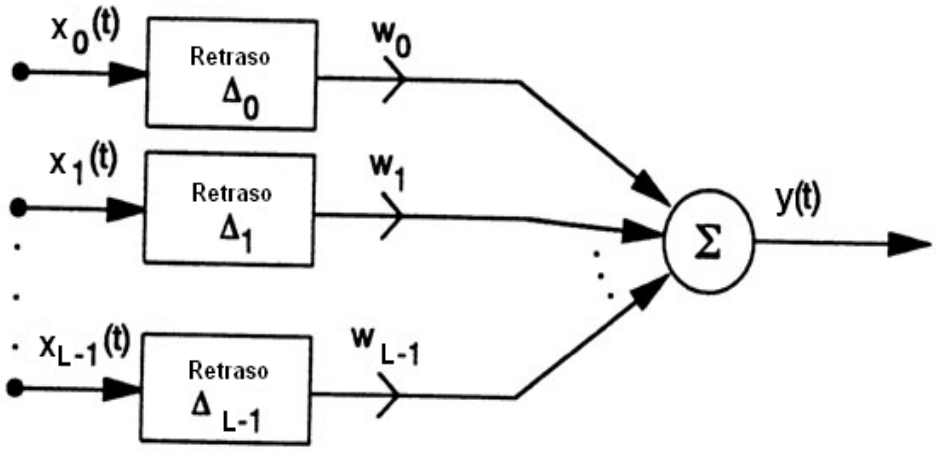


Figura 3.2. Diagrama a bloques del formador de haz de retraso y suma.

Las figuras 3.3, 3.4 y 3.5 son gráficas de el patrón de haz correspondiente a un formador de haz fijo de 10 sensores ubicados a una distancia de $3\,cm$ entre cada uno que reciben una señal acústica de $8000\,Hz$, en ellas se aprecian las características de dicha señal. La señal patrón de haz es periódica con periodo π, el pico en cero es conocido como *lóbulo principal* y los picos que se encuentran a su lado se conocen como *lóbulos secundarios o laterales*.

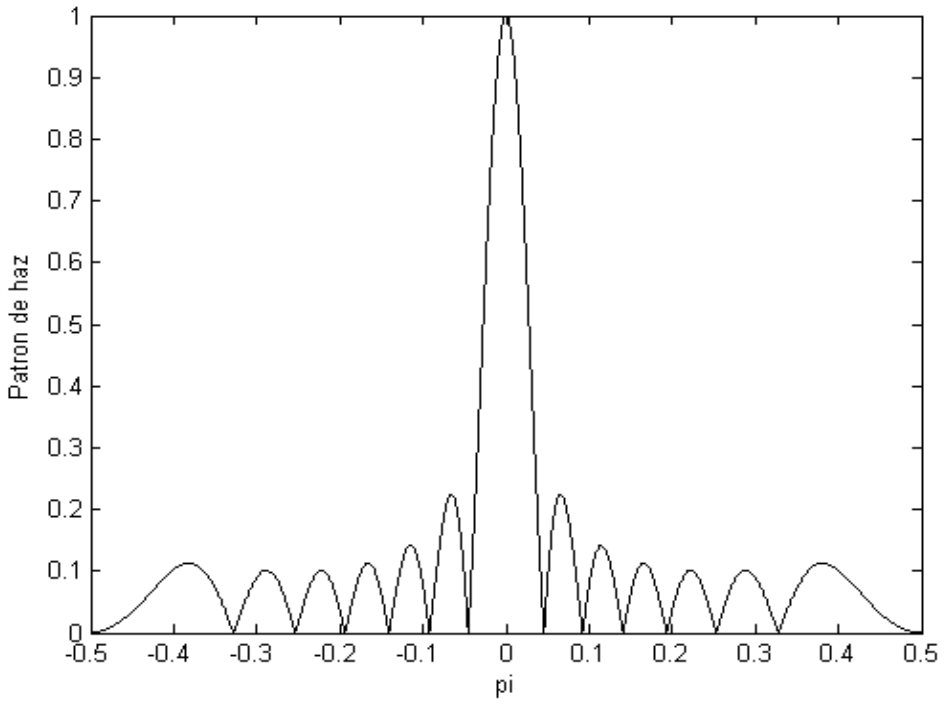


Figura 3.3. Patrón de haz de $-\pi/2$ a $\pi/2$

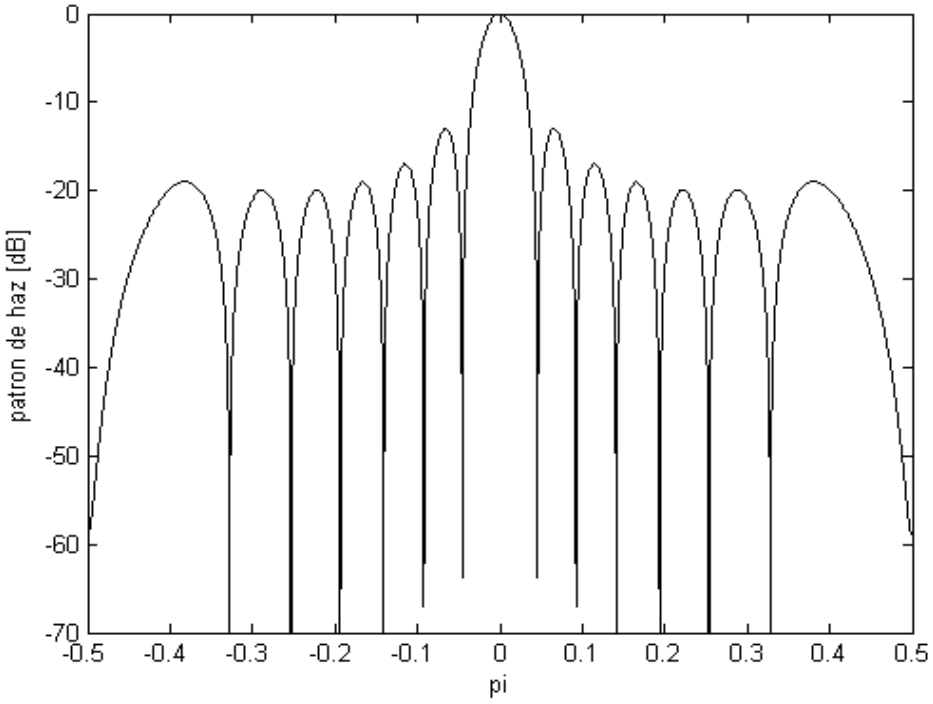


Figura 3.4. Patrón de haz en dB de $-\pi/2$ a $\pi/2$

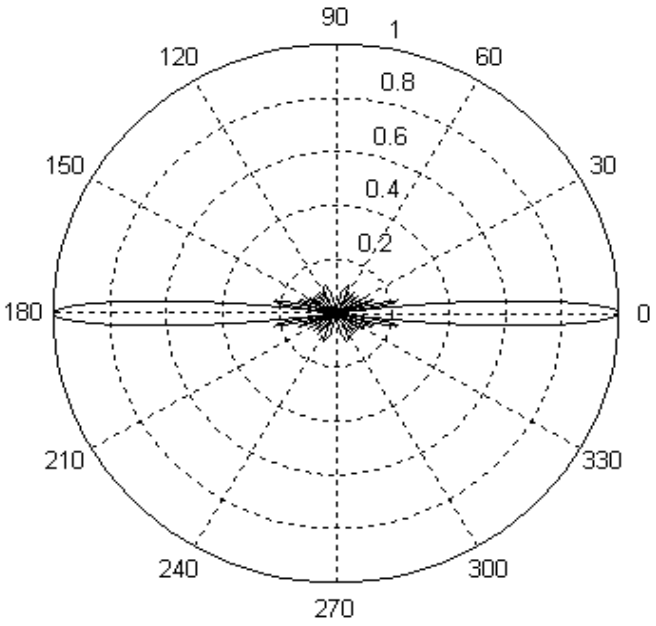


Figura 3.5. Patrón de haz en coordenadas polares

3.2.1 Uso de ventanas

Para mejorar el desempeño del formador de haz se pueden multiplicar las señales individuales de cada sensor por factores, antes de sumarlas. En la sección anterior las señales se suman solamente, lo que equivale a multiplicar las señales de los sensores por uno antes de sumarlas, lo cual corresponde a una ventana cuadrada; el desempeño del formador de haz puede mejorarse de manera significativa mediante el uso de factores correspondientes a otras ventanas, como las de Hann, Hamming o Gauss.

El calculo de los coeficientes de la ventana de Hann se realiza con la ecuación (3.19), los coeficientes correspondientes a la ventana de Hamming con la ecuación (3.20) y los de la ventana de Gauss con la ecuación (3.21).

$$W(n) = 0.5\left(1 - \cos\left(\frac{2\pi n}{M-1}\right)\right) \qquad (3.19)$$

$$W(n) = 0.53836 - 0.46164\cos\left(\frac{2\pi n}{M-1}\right) \quad (3.20)$$

$$W(n) = \exp\left(-\frac{1}{2}\left(\frac{n-\frac{M-1}{2}}{\sigma\left(\frac{M-1}{2}\right)}\right)^2\right), \sigma \leq 0.5 \quad (3.21)$$

En la figura 3.6 se muestran los patrones de haz en magnitud normalizada (izquierda) y en deciBeles (derecha) de un formador de haz fijo de 10 sensores, ubicados en una línea recta cada 3 centímetros y que reciben una señal de 8 kHz, utilizando coeficientes que corresponden a las ventanas de Hann (gráficas superiores) Hamming (gráficas de en medio) y de Gauss (gráficas inferiores) con una $\sigma = 0.4$. Las gráficas en deciBeles permiten apreciar el incremento de la caída de energía entre el lóbulo principal y el lóbulo gratinado adyacente, siendo para una ventana cuadrada, como la mostrada en la figura 3.4, de aproximadamente 13 dB, para la ventana de Hann de aproximadamente 32 dB, para la ventana de Hamming de Aproximadamente 36 dB y para la ventana de Gauss de aproximadamente 50 dB. El eje de las abscisas de todas las gráficas está dividido entre π, es decir, las gráficas son de $-\pi/2$ a $\pi/2$.

3.3 Sistemas adaptables

Otra forma de mejorar el desempeño de un formador de haz es multiplicar las señales individuales de cada sensor por coeficientes dinámicos, es decir, que cambien de acuerdo a los cambios en las señales o el ambiente. Para calcular dichos coeficientes es necesario recurrir a el procesamiento adaptable de señales.

Adaptarse significa ajustarse a diferentes condiciones o ambientes. Un sistema adaptable es aquel en cuya estructura los coeficientes son alterables o ajustables de tal manera que su comportamiento cambia mediante el contacto con el ambiente. Este tipo de sistemas se utiliza para el control y procesamiento de señales y tienen algunas o todas las características siguientes [13]:

1. Se pueden adaptar automáticamente a cambios en el ambiente.
2. Pueden ser entrenados para realizar filtros específicos y toma de decisiones, es decir, pueden ser "programados" mediante un proceso de entrenamiento.
3. No requieren procedimientos elaborados de síntesis.

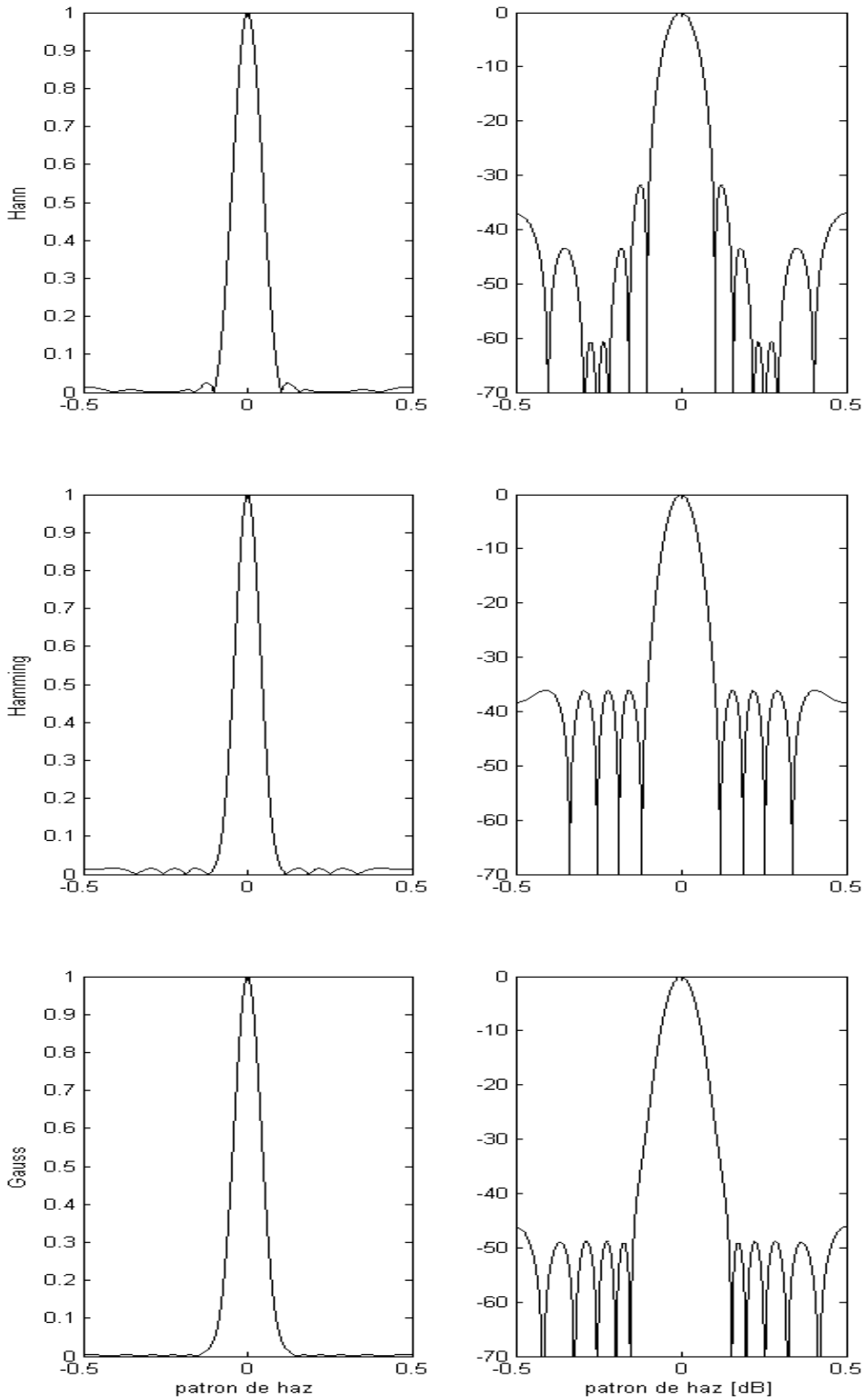


Figura 3.6. Formador de haz fijo con ventanas.

4. Pueden extrapolar el comportamiento de un modelo para manejar nuevas situaciones luego de ser entrenados en un número pequeño de señales o patrones.
5. Se pueden adaptar a ciertos defectos internos; es decir, hasta cierto punto se pueden reparar ellos mismos.
6. Se pueden describir como sistemas no-lineales con parámetros variables en el tiempo.
7. En general, son más difíciles de analizar que los sistemas no-adaptables.

Las aplicaciones actuales de los sistemas adaptables son en los campos de las comunicaciones, radar, sonar, sismología, diseño mecánico, sistemas de navegación y biomedicina.

La propiedad principal de un sistema adaptable es su comportamiento variante en el tiempo y auto ajustable. Este tipo de comportamiento es necesario ya que cuando un diseñador desarrolla un sistema que él considere óptimo, implica que ha previsto todas las posibles condiciones de entrada, por lo menos estadísticamente y sabe qué es lo que quiere que haga el sistema en cada una de estas situaciones. Muchas veces no se pueden conocer todas las posibilidades de entrada, ni siquiera estadísticamente, o las condiciones pueden cambiar en el tiempo; en estos casos un sistema adaptable que busca continuamente el comportamiento óptimo, usando un proceso de búsqueda ordenado, daría mejores resultados en comparación con un sistema no adaptable.

Los sistemas adaptables son variables en el tiempo y, generalmente, no lineales, sus características dependen de las señales de entrada. El principio de superposición no se cumple en los sistemas adaptables. Si se aplica una señal en la entrada, el sistema se adapta a esta entrada específica y cambia su forma. Por ello, los sistemas adaptables son difíciles de caracterizar en formas convencionales.

Aunque pertenecen a los sistemas no lineales, los sistemas adaptables no pertenecen a un subgrupo, sin embargo, existen dos características que los distinguen de los demás sistemas no lineales. La *primera* es que son sistemas ajustables y sus ajustes dependen de las características de la señal y no de valores instantáneos de la señal. La *segunda* es que los ajustes se hacen buscando optimizar medidas de funcionamiento específicas.

3.3.1 Adaptación de lazo abierto y lazo cerrado

Una de las formas de clasificar a los sistemas adaptables es pensando en términos de lazo abierto y lazo cerrado. El proceso adaptable de *lazo abierto* implica hacer mediciones de las características de entrada o del ambiente, utilizar esta información en una fórmula o en un algoritmo y usar los resultados para hacer los ajustes del sistema. Este tipo de adaptación se muestra en la Figura 3.7.a. La adaptación de *lazo cerrado* implica la experimentación automática con estos ajustes y el conocimiento de su salida para optimizar el funcionamiento del sistema. Este último proceso se puede llamar adaptación por "retroalimentación del funcionamiento" [13] y se muestra en la Figura 3.7.b.

Cuando se diseña un proceso adaptable, se deben tomar en cuenta varios factores para determinar si se utiliza lazo abierto o lazo cerrado. Una consideración muy importante es la disponibilidad que se tenga de señales de entrada y señales indicadoras del funcionamiento, también se toma en cuenta la capacidad y el tipo de computadora necesaria para implementar cada algoritmo.

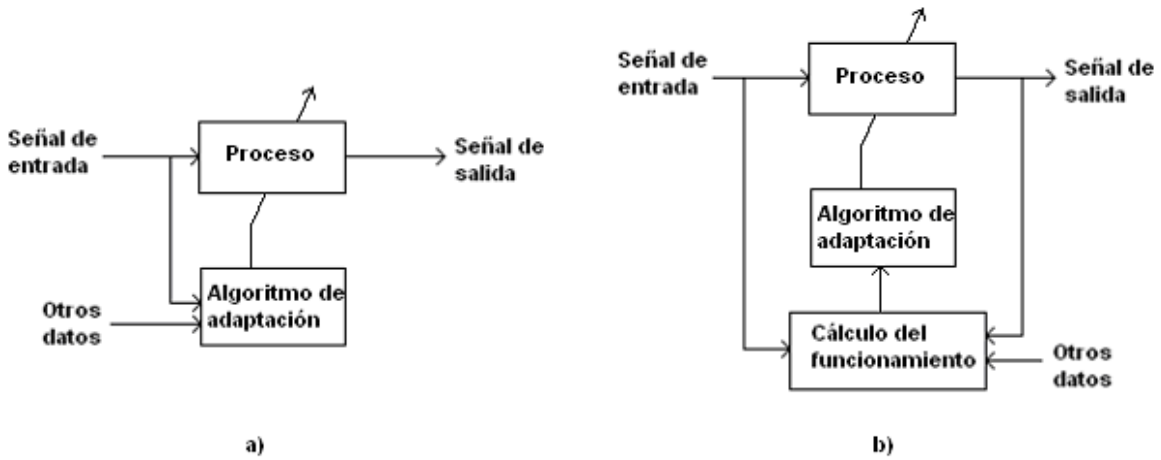


Figura 3.7. a) Adaptación de lazo abierto; b) Adaptación de lazo cerrado.

La adaptación de lazo cerrado se puede utilizar en muchos casos en los cuales no existe o no se conoce un procedimiento de síntesis analítico; por ejemplo, cuando se utiliza el criterio del error medio cuadrático, cuando los sistemas son no lineales o variables en el tiempo, cuando las

señales no son estacionarias, etc. También se puede utilizar cuando los valores de los componentes físicos del sistema son variables o no se conocen precisamente. Como resultado, se mejora la confiabilidad en el sistema utilizando la retroalimentación del funcionamiento.

3.3.2 Aplicaciones de la adaptación de lazo cerrado

En la Figura 3.8 se representa el proceso de funcionamiento. La señal de entrada se llama $x(n)$ y se define una "respuesta deseada" $d(n)$. La señal de error, $e(n)$, es la diferencia entre la señal de salida deseada y la señal de salida $y(n)$ del sistema adaptable. Un algoritmo adaptable ajusta los coeficientes de su estructura utilizando la señal de error, de esta manera se modifican las características de la respuesta minimizando el error y así se cierra el lazo [13].

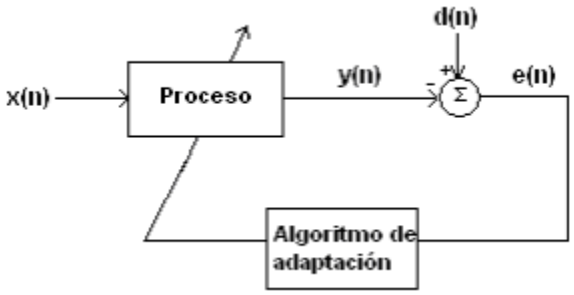


Figura 3.8. Señales en la adaptación de lazo cerrado.

Existen cuatro aplicaciones principales de la adaptación de lazo cerrado: predicción, identificación del sistema, modelado inverso y cancelador de interferencia [13].

En la aplicación de *predicción* la señal deseada es una señal de interés y un retardo de ésta es la entrada al proceso, el cual intenta "predecir" la entrada actual de manera que $y(n)$ cancele $d(n)$ y $e(n)$ tienda a un mínimo. Esta aplicación se usa en codificación de señales y reducción de ruido.

En la *identificación del sistema* una señal de banda ancha es la entrada al proceso y a una planta. Para reducir $e(n)$, el proceso adaptable trata de emular la función de transferencia de la planta y cuando el error tiende a un mínimo, se ha identificado a la planta ya que tendrá la misma función de transferencia del procesador adaptable. Esta aplicación se usa en los estudios de la vibración de sistemas mecánicos.

En el *modelado inverso* el proceso adaptable intenta recuperar la señal de entrada con un retardo mientras que ésta ha sido alterada por una planta y además se le ha añadido ruido. Este modelado se utiliza para compensar los efectos de un transductor, un canal de comunicación u otro sistema o para producir un modelo invertido de una planta desconocida, también se utiliza en el diseño de filtros digitales.

En la configuración de *cancelador de interferencia* la señal deseada es una señal con ruido, $s(n) + n(n)$, la señal de entrada del proceso es la señal de ruido distorsionada pero correlacionada, $n'(n)$. El objetivo es que la salida del proceso, $y(n)$, sea muy parecida a $n(n)$ de modo que el error, $e(n)$, sea muy parecido a la señal original, $s(n)$, y de esta forma se logre cancelar el ruido. Esta configuración es la utilizada en formadores de haz adaptables.

3.4 Combinación lineal adaptable

La combinación lineal adaptable o filtro adaptable no recursivo es fundamental para el procesamiento adaptable de señales, ya que aparece, de una forma u otra, en la mayoría de los filtros y sistemas adaptables y debido a que no es recursivo, es relativamente fácil estudiarlo y analizarlo. En la Figura 3.9 se muestra un diagrama de su forma general. Tiene un vector de la señal de entrada con elementos $x_0, x_1, \dots, x_L$, una serie correspondiente de pesos ajustables, $w_0, w_1, \dots, w_L$, una unidad sumadora y una sola señal de salida, $y(n)$. El proceso mediante el cual se ajustan los pesos se llama “ajuste de pesos”, “ajuste de ganancia” o “adaptación” y se le llama lineal porque para valores fijos de pesos la salida es una combinación lineal de los componentes de entrada; sin embargo, cuando los pesos están en el proceso de ajuste, también son funciones de los componentes de entrada y la salida no es una función lineal de las entradas.

Los elementos del vector de entrada se puede interpretar físicamente de dos maneras: pueden considerarse como entradas simultáneas de $L + 1$ fuentes diferentes en un sistema adaptable de detección acústica; por otro lado, los elementos $x_0, x_1, \dots, x_L$ pueden ser considerados como $L + 1$ muestras secuenciales de la misma fuente. Estos dos casos se conocen como *entradas-múltiples* y *entrada-sencilla* [13], y en cada caso se puede expresar el vector de entrada de una forma diferente:

$$Entradas\ múltiples: \quad \bar{X}_k = [x_{0k} \quad x_{1k} \quad \dots \quad x_{Lk}]^T \tag{3.22}$$

$$Entrada\ sencilla: \quad \bar{X}_k = [x_k \quad x_{k-1} \quad \dots \quad x_{k-L}]^T \tag{3.23}$$

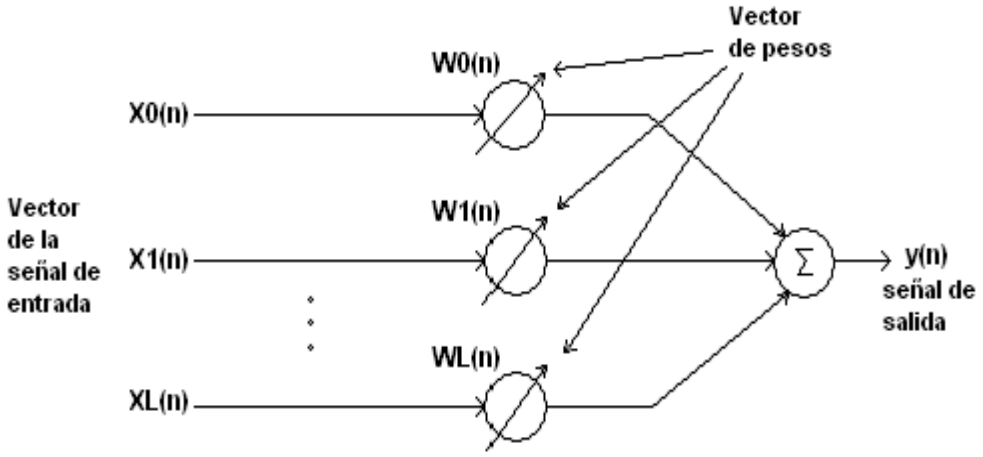


Figura 3.9. Forma general de una combinación lineal adaptable.

El subíndice k de las ecuaciones (3.22) y (3.23) se utiliza como un índice de tiempo, en el caso de la entrada sencilla, el proceso adaptable se puede implementar con una combinación lineal adaptable y elementos de retardos unitarios, como se puede ver en la Figura 3.10. Esta estructura se conoce como filtro transversal adaptable y existe en muchas aplicaciones en los campos de modelado adaptable y procesamiento adaptable de señales.

A partir de las notaciones de la señal de entrada en las expresiones (3.22) y (3.23) se pueden obtener las relaciones de entrada-salida para ambos casos como se muestra en las ecuaciones (3.24) y (3.25).

$$Entrada\ sencilla: \quad y_k = \sum_{l=0}^{L} w_{lk} x_{k-l} \tag{3.24}$$

$$Entradas\ múltiples: \quad y_k = \sum_{l=0}^{L} w_{lk} x_{lk} \tag{3.25}$$

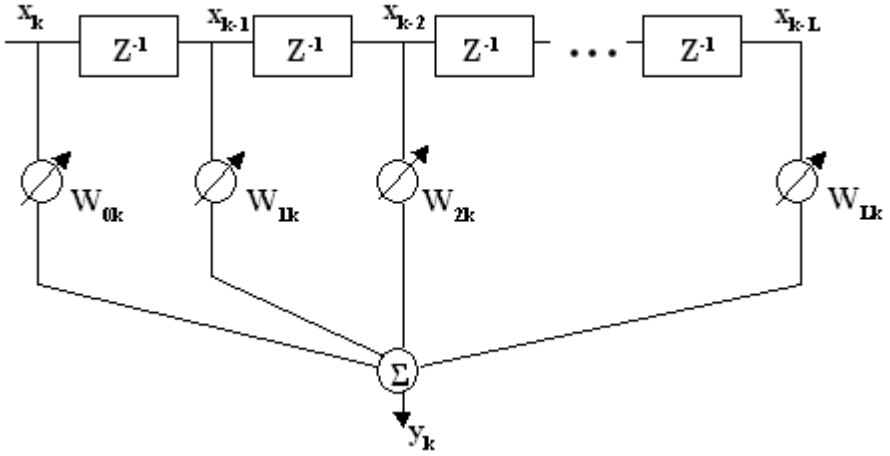


Figura 3.10. Combinación lineal adaptable en la forma de filtro transversal adaptable de entrada sencilla.

Los pesos asociados a la entrada pueden modelarse como un vector de la forma:

$$\bar{W}_k = [w_{0k} \quad w_{1k} \quad \dots \quad w_{Lk}]^T \tag{3.26}$$

De esta manera se pueden expresar las relaciones de entrada-salida en las expresiones (3.24) y (3.25) utilizando notación vectorial:

$$y_k = \bar{X}_k^T \bar{W}_k = \bar{W}_k^T \bar{X}_k \tag{3.27}$$

Respuesta deseada y error

La combinación lineal adaptable se puede utilizar tanto en sistemas adaptables de lazo abierto como de lazo cerrado. En los sistemas de lazo abierto el ajuste del vector de pesos depende únicamente de la entrada y las propiedades del ambiente, mientras que en los sistemas de lazo cerrado depende de la entrada, de valores anteriores de la señal de salida y de condiciones del ambiente.

En el proceso de adaptación retroalimentado, el vector de pesos es ajustado para hacer que la salida, y_k, se acerque lo mejor posible a la señal deseada, lo cual se logra comparando la salida con la señal deseada y obteniendo una señal de error, después se ajusta el vector de pesos para minimizar esta señal. En la Figura 3.11 se muestra el proceso que se sigue para obtener la señal

de error. La señal de salida, y_k, se resta a la señal deseada, d_k, para producir la señal de error, e_k. La fuente de la señal deseada, d_k, depende de la aplicación de la combinación adaptable.

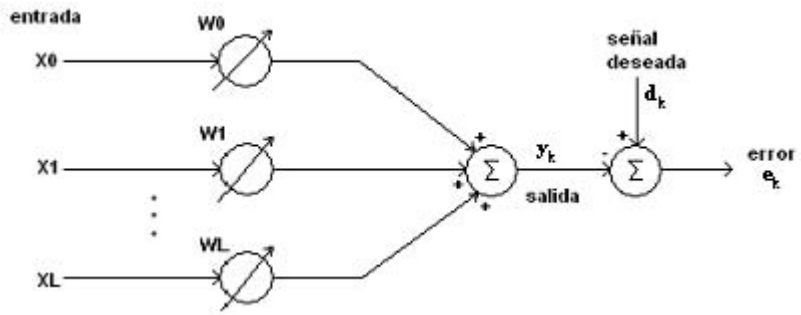


Figura 3.11. Combinación lineal adaptable de entradas múltiples.

3.4.1 Función de desempeño y gradiente

Una forma de minimizar el error, e_k, de la figura 3.11, es obteniendo los coeficientes óptimos, $\overline{W}^*$, a través del criterio de los mínimos cuadrados. En la figura 3.11 se observa que la señal de error se obtiene con

$$e_k = d_k - y_k \tag{3.28}$$

Sustituyendo (3.27) en la expresión anterior se obtiene

$$e_k = d_k - \bar{X}_k^T \overline{W}_k \tag{3.29}$$

Elevando (3.29) al cuadrado para obtener el error cuadrático instantáneo, se obtiene

$$e_k^2 = d_k^2 + \overline{W}^T \bar{X}_k \bar{X}_k^T \overline{W} - 2 d_k \bar{X}_k^T \overline{W} \tag{3.30}$$

Asumiendo que e_k, d_k y $\bar{X}_k$ son estacionarias se toma el valor esperado, $E[\,.\,]$, de (3.30) como

$$E[e_k^2] = E[d_k^2] + \overline{W}^T E[\bar{X}_k \bar{X}_k^T] \overline{W} - 2E[d_k \bar{X}_k^T] \overline{W} \tag{3.31}$$

Donde el valor esperado de una suma es la suma de los valores esperados, sin embargo, el valor esperado de un producto es el producto de los valores esperados cuando las variables son estadísticamente independientes. Las señales x_k y d_k generalmente no son independientes[13].

Para expresar la función del error medio cuadrático definimos la matriz de autocorrelación de la señal de entrada $\bar{R}$ como

$$\bar{R} = E[\bar{X}_k \bar{X}_k^T] = E\begin{bmatrix} x_{0k}^2 & x_{0k}x_{1k} & \dots & x_{0k}x_{Lk} \\ x_{1k}x_{0k} & x_{1k}^2 & \dots & x_{1k}x_{Lk} \\ \vdots & \vdots & & \vdots \\ x_{Lk}x_{0k} & x_{Lk}x_{1k} & \dots & x_{Lk}^2 \end{bmatrix} \tag{3.32}$$

Los valores de la diagonal principal son los valores medios cuadráticos de los componentes de entrada. Se define $\bar{P}$ como el vector columna que es la correlación entre d_k y x_k.

$$\bar{P} = E[d_k \bar{X}_k^T] = E[d_k x_{0k} \quad d_k x_{1k} \quad \dots \quad d_k x_{Lk}]^T \tag{3.33}$$

Ahora se designa el error medio cuadrático (MSE) en (3.31) como ξ y se expresa en términos de $\bar{R}$ y $\bar{P}$:

$$MSE = \xi = E[e_k^2] = E[d_k^2] + \bar{W}^T \bar{R} \bar{W} - 2\bar{P}^T \bar{W} \tag{3.34}$$

Se puede ver que el MSE es una función cuadrática de los componentes del vector de pesos $\bar{W}$ cuando los componentes de entrada y de la señal deseada son estacionarios.

Si se deriva (3.34) con respecto a $\bar{W}$ se obtiene el gradiente que se designa $\nabla\xi$ o simplemente ∇ y se obtiene el vector columna

$$\nabla = \frac{\partial \xi}{\partial \bar{W}} = 2\bar{R}\bar{W} - 2\bar{P} \tag{3.35}$$

Para obtener el mínimo del error medio cuadrático, el vector de pesos $\bar{W}$ debe contener un valor óptimo $\bar{W}^*$. Haciendo el gradiente igual a cero:

$$\nabla = 0 = 2\bar{R}\bar{W}^* - 2\bar{P} \tag{3.36}$$

Se puede encontrar el vector de pesos de valores óptimos $\bar{W}^*$, a este vector se le llama *vector de pesos de Wiener* [13]:

$$\bar{W}^* = \bar{R}^{-1}\bar{P} \tag{3.37}$$

Esta ecuación es una forma matricial de la ecuación de Wiener-Hopf. Sustituyendo $\bar{W}^*$ de (3.37) por $\bar{W}$ en (3.34) se obtiene el *error medio cuadrático mínimo*:

$$\begin{aligned} \xi_{min} &= E[d_k^2] + \bar{W}^{*T} \bar{R} \bar{W}^* - 2\bar{P}^T \bar{W}^* \\ &= E[d_k^2] + [\bar{R}^{-1}\bar{P}]^T \bar{R} \bar{R}^{-1} \bar{P} - 2\bar{P}^T \bar{R}^{-1} \bar{P} \end{aligned} \tag{3.38}$$

Y utilizando propiedades de las matrices, se obtiene

$$\xi_{min} = E[d_k^2] - \bar{P}^T \bar{R}^{-1} \bar{P} = E[d_k^2] - \bar{P}^T \bar{W}^* \quad (3.39)$$

Que también se puede expresar como se muestra en (3.40)

$$\xi = \xi_{min} + [\bar{W} - \bar{W}^*]^T \bar{R} [\bar{W} - \bar{W}^*] \quad (3.40)$$

Sin embargo, como se puede ver en la ecuación (3.37), para llegar al valor exacto del vector de Wiener es necesario obtener la inversa de la matriz $\bar{R}$, proceso que resulta muy laborioso y requiere de una gran cantidad de operaciones. Existen algoritmos que permiten llegar a una solución aproximada del vector de Wiener y no requieren de operaciones muy complejas, uno de ellos es el algoritmo LMS.

3.5 Algoritmo LMS

El algoritmo least mean square (LMS) fue presentado por Widrow y Hoff en 1959 y es uno de los más usados cuando se trabaja con sistemas adaptables, debido a su simplicidad y facilidad de cómputo, utilizándose para llegar a una solución aproximada de la ecuación de Wiener [13]. Si el sistema es una combinación lineal adaptable, y en cada iteración se tienen disponibles el vector de entradas $\bar{X}_k$ y la respuesta deseada d_k, el algoritmo LMS será una opción adecuada para diferentes aplicaciones de procesamiento adaptable de señales.

La combinación lineal adaptable se puede aplicar de dos maneras básicas, dependiendo de si la entrada se encuentra en paralelo (entradas múltiples) o en serie (entrada sencilla), en ambos casos se tiene la salida, y_k, como una combinación lineal de las muestras de entrada. $\bar{X}_k$ es el vector de las muestras de entrada en cualquiera de las dos configuraciones.

$$e_k = d_k - \bar{X}_k^T \bar{W}_k \quad (3.41)$$

Para obtener el algoritmo LMS, se toma el error al cuadrado, e_k^2, como una estimación de ξ_k. Luego, en cada iteración del proceso adaptable, se tiene una estimación del gradiente como se muestra en (3.42), [13].

$$\widehat{\nabla}_k = \begin{bmatrix} \frac{\partial e_k^2}{\partial w_0} \\ \vdots \\ \frac{\partial e_k^2}{\partial w_L} \end{bmatrix} = 2e_k \begin{bmatrix} \frac{\partial e_k}{\partial w_0} \\ \vdots \\ \frac{\partial e_k}{\partial w_L} \end{bmatrix} = -2e_k \bar{X}_k \tag{3.42}$$

Con esta estimación del gradiente, se puede obtener el algoritmo LMS como se expresa en (3.43).

$$\begin{aligned} \bar{W}_{k+1} &= \bar{W}_k - \mu \widehat{\nabla}_k \\ &= \bar{W}_k + 2\mu e_k \bar{X}_k \end{aligned} \tag{3.43}$$

μ es una constante de ganancia que regula la velocidad y la estabilidad de la adaptación.

De (3.41), (3.42) y (3.43) se observa que el algoritmo LMS puede ser implementado en cualquier sistema sin tener que elevar al cuadrado, obtener promedios o derivadas, por lo que es muy simple y eficiente [13].

3.5.1 Convergencia del vector de pesos

En el algoritmo LMS, como en todos los algoritmos adaptables, un asunto importante es la convergencia a la solución del vector de pesos óptimo, es decir, cuando $E[e_k^2]$ es mínimo. Para analizar esta convergencia primero notamos en (3.43) que el vector de pesos $\bar{W}_k$ es una función del error e_k, el vector de pesos anterior $\bar{W}_{k-1}$ y los vectores de entrada anteriores $\bar{X}_{k-1,}\bar{X}_{k-1}, \dots, \bar{X}_0$. Asumiendo que los vectores de entrada son independientes en el tiempo, $\bar{W}_k$ es independiente de $\bar{X}_k$. Obteniendo el valor esperado en ambos lados de la ecuación (3.43) y utilizando la ecuación (3.29) se tiene la ecuación (3.44).

$$\begin{aligned} E[\bar{W}_{k+1}] &= E[\bar{W}_k] + 2\mu E[e_k \bar{X}_k] \\ &= E[\bar{W}_k] + 2\mu(E[d_k \bar{X}_k] - E[\bar{X}_k \bar{X}_k^T \bar{W}_k]) \end{aligned} \tag{3.44}$$

Con el apoyo de las ecuaciones (3.32) para $\bar{R}$, (3.33) para $\bar{P}$ y (3.37) para el vector de pesos óptimo, la ecuación (3.44) se puede escribir como se muestra en la ecuación (3.45).

$$\begin{aligned} E[\bar{W}_{k+1}] &= E[\bar{W}_k] + 2\mu(\bar{P} - \bar{R}E[\bar{W}_k]) \\ &= (\bar{I} - 2\mu\bar{R})E[\bar{W}_k] + 2\mu\bar{R}\bar{W}^* \end{aligned} \tag{3.45}$$

Conforme k aumenta, el vector de pesos alcanza su valor óptimo, esta convergencia se consigue si se cumple la condición

$$\frac{1}{\lambda_{max}} > \mu > 0 \qquad (3.46)$$

en donde λ_{max} es el valor propio más grande de $\bar{R}$ [13]. La velocidad de adaptación está determinada por el tamaño de μ. También se puede notar que λ_{max} no puede ser más grande que la traza de $\bar{R}$, que es la suma de los elementos de la diagonal principal de $\bar{R}$.

En un filtro adaptable transversal la traza de $\bar{R}$ está dada por $(L+1)E[x_k^2]$, es decir, $L+1$ veces la energía de la señal de entrada. Por lo tanto, la convergencia está dada por (3.47), [13],

$$En\ general \qquad 0 < \mu < \frac{1}{tr[\bar{R}]} \qquad (3.47a)$$

$$Filtro\ transversal \quad 0 < \mu < \frac{1}{(L+1)(energía\ de\ la\ señal)} \qquad (3.47b)$$

Estos límites para μ son más fáciles de aplicar ya que los elementos de $\bar{R}$ y la energía de la señal pueden ser estimados más fácilmente.

3.6 Arreglos de sensores y formación de haz adaptable

La Formación de Haz (Beamforming) es una técnica de procesamiento de señales utilizada para controlar la dirección de recepción o transmisión de una señal en un arreglo de sensores. Utilizando el beamforming se puede enviar la mayoría de la energía de la señal a transmitir en la dirección que se quiera, también se puede calibrar un grupo de sensores para que reciban principalmente señales que provengan de alguna dirección especificada. Un sistema de beamforming adaptable consiste en un grupo de sensores dispuestos espacialmente y conectados a un procesador de señales adaptable de una sola entrada o de entradas múltiples.

Los arreglos adaptables se pueden utilizar para reducir o eliminar interferencias y mejorar la relación señal a ruido. Mediante algunos algoritmos adaptables se puede hacer que el arreglo de sensores se ajuste para captar una señal sin conocer de antemano su dirección de arribo y separe esta señal de interferencias que no provengan de la misma dirección. Otros algoritmos adaptables permiten determinar la dirección de arribo de la señal en la presencia de interferencia.

Utilizando otros algoritmos se puede filtrar señales débiles de señales fuertes mientras que provengan de diferentes lugares.

3.6.1 Cancelación de lóbulos laterales

La forma más simple de arreglo adaptable es el cancelador de lóbulos laterales, desarrollado por Howells y Applebaum para aplicaciones con antenas [13]; en la figura 3.12 se muestra el diagrama a bloques del esquema que utilizaron, con dos sensores omnidireccionales, uno es el primario y el otro la referencia, similar a un cancelador adaptable de ruido. Se supone que se tiene una señal y al mismo tiempo una interferencia, ambos sensores reciben tanto la señal como la interferencia pero, debido a que están separados, sus salidas no son idénticas aunque están relacionadas.

Un caso importante es cuando la interferencia es más fuerte que la señal, en este caso los pesos del filtro son completamente controlados por la interferencia. Cuando el algoritmo converge la salida del filtro adaptable es casi igual a la señal de interferencia del sensor primario, entonces la salida del sistema es igual a la señal del sensor primario menos la salida del filtro adaptable, es decir, es igual a la señal deseada con muy poca interferencia [13].

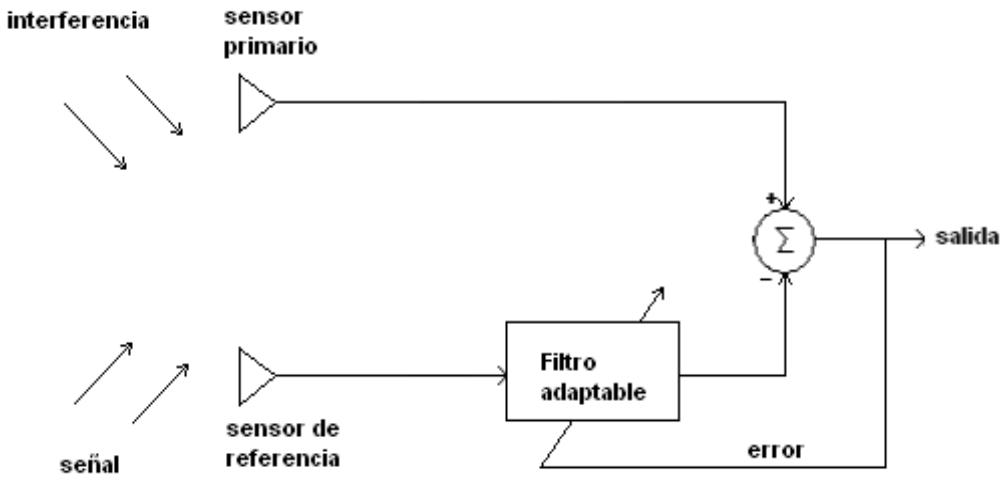


Figura 3.12. Cancelador de lóbulos laterales.

3.6.2 Beamforming con una señal piloto

Un tipo de beamforming adaptable fue desarrollado por B. Widrow, L. J. Griffiths, P. E. Mantey y B. B. Goode y está basado en un algoritmo de "señal piloto" [13], a diferencia del formador de haz Howells-Applebaum, el cual empieza siendo omnidireccional y pierde sensibilidad a las señales fuertes de interferencia, el formador de haz de señal-guía forma un haz en una dirección específica y utiliza el proceso adaptable para mantener este haz y, al mismo tiempo, anula interferencias provenientes de otras direcciones.

La señal-piloto se utiliza para entrenar al formador de haz para que tenga un lóbulo principal en la dirección deseada y pueda anular interferencias provenientes de direcciones diferentes.

Existen algunos algoritmos que producen resultados similares a los algoritmos de señal-piloto pero que en realidad son más fáciles de implementar y en ocasiones funcionan mejor, estos son los algoritmos de Griffiths y Frost [13].

El formador de haz de Frost consiste en un arreglo de K sensores y cada sensor es seguido por un filtro transversal de J pesos, el número de pesos es igual para todos los filtros, por lo tanto, el número total de pesos es JxK. La salida del formador es igual a la suma de las salidas de los filtros. Los pesos son ajustados por el algoritmo LMS que minimiza el error cuadrado de la señal de salida [14].

3.6.3 Cancelación de lóbulos laterales generalizada (GSC)

Este formador de haz se utiliza para separar señales acústicas del ruido mediante cancelación adaptable de ruido [15]. En la figura 3.13 se muestra el diagrama de este formador, las señales que llegan de diferentes sensores son tratadas como la misma señal pero con diferentes retardos y diferente ruido.

La señal que llega al arreglo de sensores contiene a la señal deseada más ruido y se aplica un retardo para que la señal deseada sea la misma en cada sensor. Luego cada señal, $x_k(n)$, se multiplica con un peso que es igual a $1/num$, siendo num el número de sensores y se suman

todas las señales; la suma, $d(n)$, es igual a la señal deseada más ruido, el objetivo es eliminar este ruido; a la señal, $d(n)$, luego se la aplica un retardo para obtener $d'(n)$.

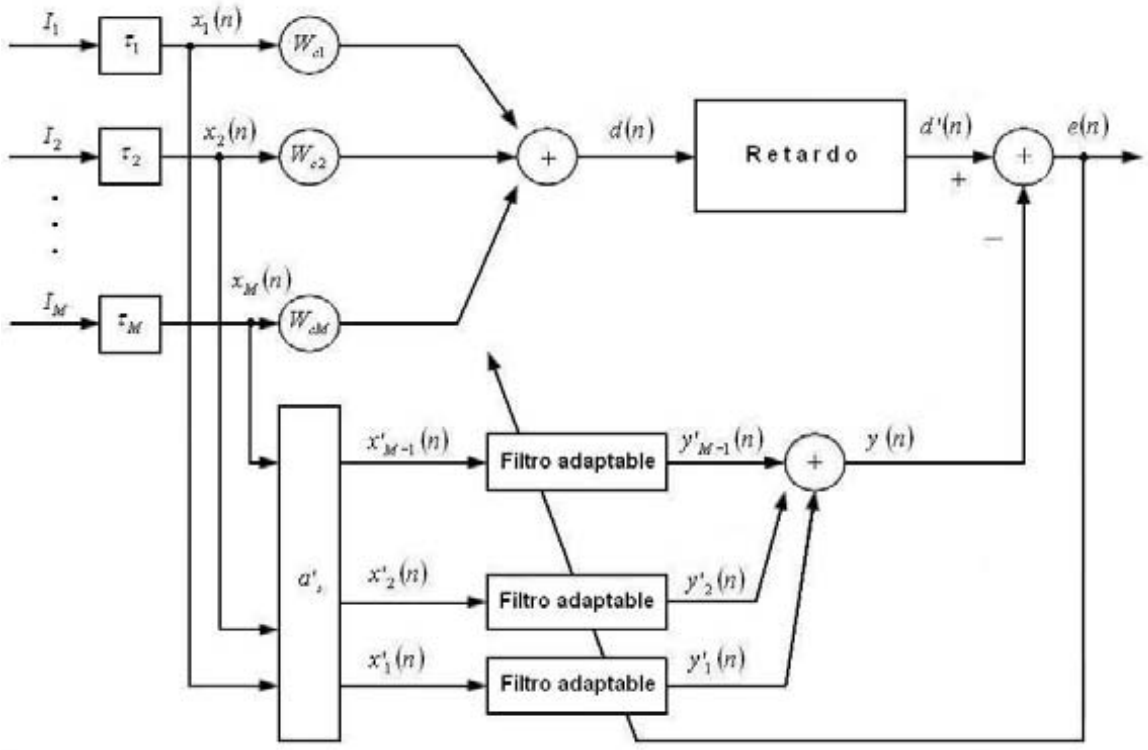


Figura 3.13. Generalized sidelobe canceller (GSC).

Al mismo tiempo, cada señal $x_k(n)$ pasa por una matriz de bloqueo, a'_s, que remueve la señal deseada y se obtiene, $x'_k(n)$, que contiene sólo el ruido de cada señal y es la entrada a cada filtro adaptable. La salida de cada filtro es $y_k(n)$ y al sumar todas estas salidas se obtiene $y(n)$ que contiene sólo ruido; a la señal $d'(n)$, que contiene la señal deseada más ruido, se le resta la señal, $y(n)$, y se obtiene la señal de error que es igual a la señal deseada.

3.7 Resumen

Los arreglos de sensores son fundamentales en el estudio de las características espaciales de las señales y permiten realizar diversos procesos valiéndose de las mismas. Una aplicación muy

importante de los arreglos de sensores son los formadores de haz, con ellos se puede conseguir enviar o recibir señales provenientes de una dirección y anular señales provenientes de direcciones diferentes. Este proceso es muy útil cuando se tienen distintas señales provenientes de diferentes direcciones pero la señal de interés proviene de una sola dirección. El desempeño de un formador de haz puede mejorarse considerablemente mediante el uso del procesamiento adaptable.

Los sistemas adaptables son muy aplicados en el procesamiento de señales en tiempo real debido a que las condiciones en las que se encuentran los sistemas y las señales de entrada pueden cambiar de un momento a otro.

Un algoritmo muy utilizado en los sistemas adaptables es el algoritmo LMS, esto es debido a su facilidad de cómputo y a su eficacia, además contiene límites para sus parámetros para conseguir la convergencia, sin embargo, no existen condiciones definitivas para ésta. Además de los algoritmos y las estructuras adaptables presentadas existen otros de mayor desempeño, complejidad y cantidad de cómputo, pero estos se encuentran fuera de los límites de nuestro trabajo.

Capitulo 4
Análisis y desarrollo del sistema

El presente capítulo tiene por objeto la descripción del diseño de la tarjeta de expansión y está dividido en cuatro partes principales, la primera sección describe brevemente las características de la tarjeta de desarrollo de importancia para el diseño de la tarjeta de expansión; la segunda sección describe los componentes utilizados y la forma en que se interconectan en la tarjeta de expansión; la tercera trata sobre los procesos que fueron programados en el DSP y en la parte final se da una descripción de la forma en que debe utilizarse el selector de la tarjeta de expansión y de los procesos programados.

Durante el diseño se debe tener en cuenta las características de las señales que va adquirir la tarjeta de expansión; en nuestro caso las señales de voz cuyo ancho de banda es de $3500\ Hz$, la frecuencia de muestreo se eligió de $8000\ Hz$ para cumplir con el teorema del muestreo.

4.1 Consideraciones de diseño

La tarjeta de expansión tiene como propósito hacer posible la selección y realización de varios procesos, entre ellos la adquisición de cuatro canales simultáneos de voz en tiempo real sobre la tarjeta de desarrollo DSK C5402; para esto es necesario adquirir la señal acústica y acondicionarla para que el DSP la pueda procesar.

La adquisición de las señales de voz se realiza utilizando un arreglo lineal, uniforme, de cuatro micrófonos instalados en una base de acrílico; empleándose la ecuación (3.13) para calcular la distancia máxima a la que se pueden encontrar los sensores, para el ancho de banda de la señal de voz, la cual es de $4.28\ cm$ (considerando una frecuencia máxima de $4000\ Hz$ y la velocidad del sonido de $343\ m/s$) y decidiéndose la colocación de los micrófonos a una distancia de $3\ cm$ entre ellos[11].

Procesadores Digitales de Señales

Un procesador Digital de señales (DSP) es un microcontrolador diseñado específicamente para el procesamiento digital de señales en tiempo real, es decir, es capaz de realizar el proceso deseado a la señal en un tiempo menor al tiempo de muestreo; para lograrlo los DSPs trabajan en paralelo utilizando intensivamente la técnica llamada 'pipeline', que les permite traer datos y realizar instrucciones al mismo tiempo; además cuentan con multiplicadores y sumadores que trabajan en paralelo, existen DSPs diseñados para trabajar en punto fijo y flotante. Para el manejo de la memoria, usualmente se cuenta con arquitecturas tipo Harvard, que tienen la capacidad de acceder a ella directamente. Sin embargo la diferencia principal entre un DSP y otro procesador convencional es que el DSP está diseñado para realizar de forma óptima la suma de los resultados de un conjunto de multiplicaciones de elemento por elemento entre dos vectores, llamada multiplicación acumulación (MAC); ésta es la operación básica del procesamiento digital de señales en la realización de convoluciones, correlaciones, filtros digitales, y transformadas diversas.

La compañía Texas Instruments (TI) ha sido el líder en el mercado de los DSPs desde 1983, año en que se produjo por primera vez el DSP TMS32010; por eso en este proyecto se utiliza el DSP TMS320C5402 fabricado por esa compañía [19].

Actualmente TI produce DSPs pertenecientes a tres familias. La familia C2000 que realiza entre 100 y 600 MIPS (millones de instrucciones por segundo), recomendada para aplicaciones industriales embebidas, como el control de motores, fuentes de poder digitales y sensores inteligentes. La familia C5000 que realiza entre 100 y 800 MIPS, con un consumo de energía muy bajo, se utiliza en aplicaciones como reproductores de música, VoIP, sistemas de localización satelital y equipos médicos portátiles. La familia C6000 que realiza entre 200 y 8000 MIPS, ideal para aplicaciones de video, infraestructuras de redes de banda ancha y audio de alta definición [20].

En este trabajo se utiliza un DSP C5402 porque, como se indica en el párrafo anterior, la familia C5000 es la más adecuada para aplicaciones de audio; además, el laboratorio de Procesamiento Digital de señales del posgrado de la Facultad de Ingeniería de la UNAM cuenta con tarjetas de esta familia, en un sistema de desarrollo llamado DSK.

La tarjeta de desarrollo DSP TMS320C5402 DSK

El DSP que utilizamos es el TMS320C5402, que trabaja hasta a 100 MHz, el cual cuenta con una unidad aritmética lógica (ALU) de 40 bits, una memoria de lectura y escritura (RAM) de acceso dual de 16k x 16 bit, una memoria de solo lectura (ROM) de 4k x 16 Bit, arquitectura multibus avanzada con tres buses separados para memoria de dato y un bus para memoria de programa, dos puertos seriales buffereados multicanal (McBSP), seis controladores de acceso directo a memoria (DMA) y dos relojes de 16 bit. Esto se muestra de forma esquemática en la figura 4.1.

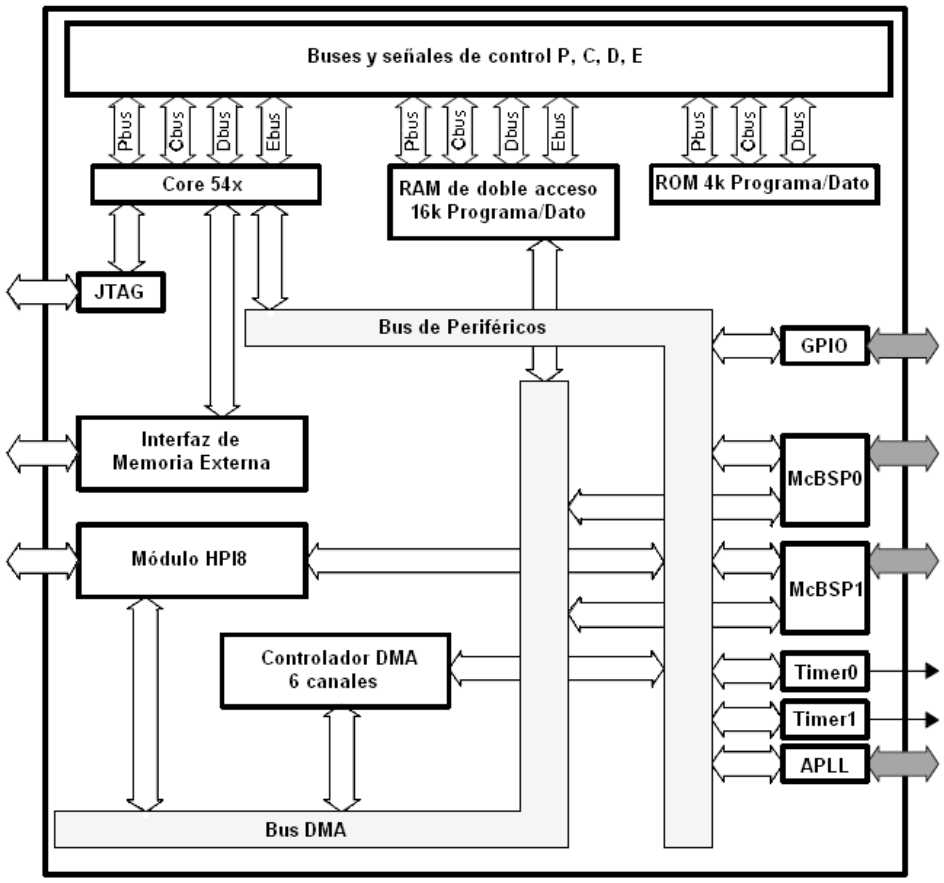


Figura 4.1. Diagrama de bloques del DSP C5402

La tarjeta de desarrollo DSP starter kit (DSK) del DSP cuenta además con un sistema de emulación embebido de JTAG, un controlador del puerto paralelo que permite utilizar una computadora personal para acceder al DSP, dos conectores de audio de 3.5 mm para micrófono y

bocina, una interfaz RJ11, un puerto de entrada y salida RS232, conectores de expansión de puertos, conectores para una interface JTAG y una fuente de $+5\ V$ universal; estos componentes se muestran en la figura 4.2

La tarjeta de desarrollo tiene las dimensiones indicadas en la figura 4.3, las cuales están en milímetros cuando no están entre paréntesis y en pulgadas cuando lo están.

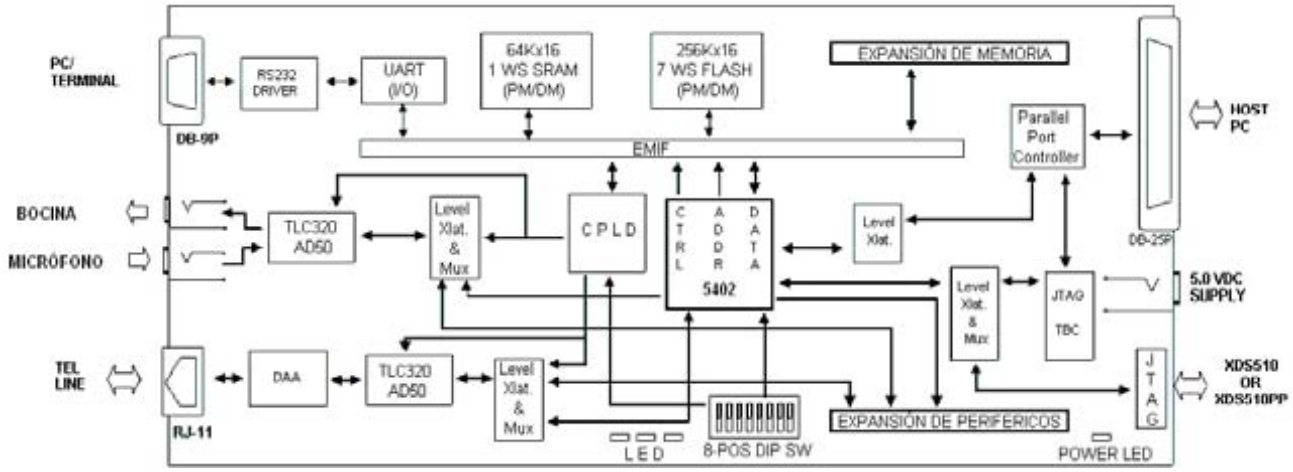


figura 4.2. Diagrama de bloques de la tarjeta de desarrollo

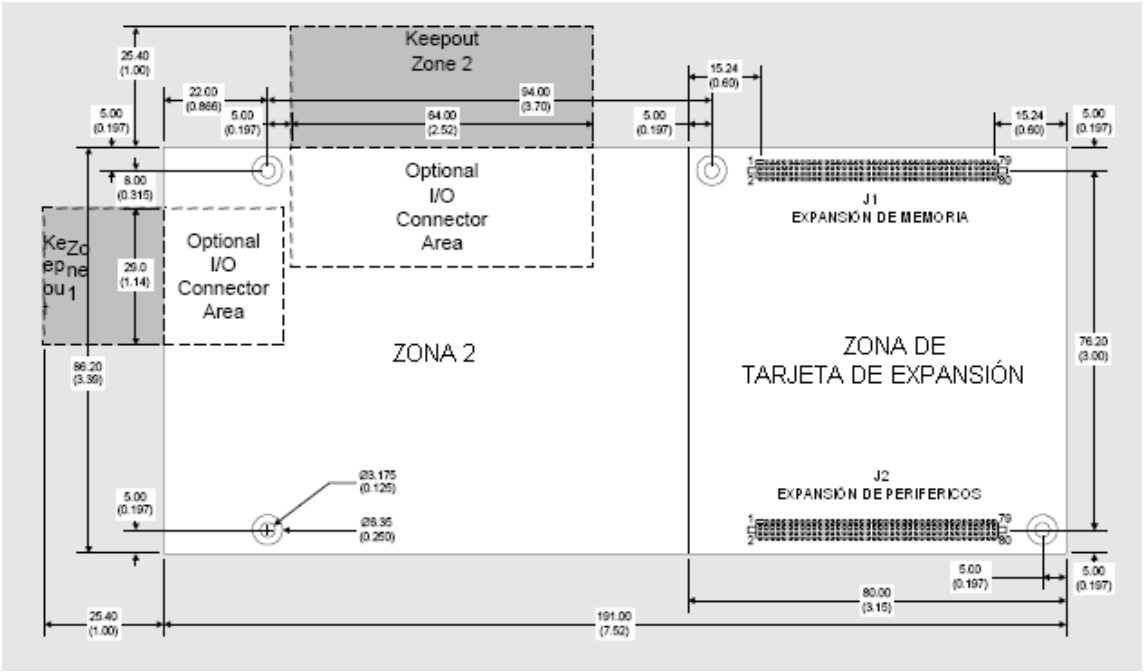


figura 4.3. Dimensiones de la tarjeta de desarrollo

4.2 Diseño de la tarjeta de expansión

La tarjeta de expansión diseñada en este trabajo se muestra en la figura 4.4 y está compuesta por las siguientes secciones:

- Conexión con la tarjeta de desarrollo
- Micrófonos
- Amplificación de la señal acústica
- Conversión analógica digital y
- Selección de procesos

Que se describen a continuación

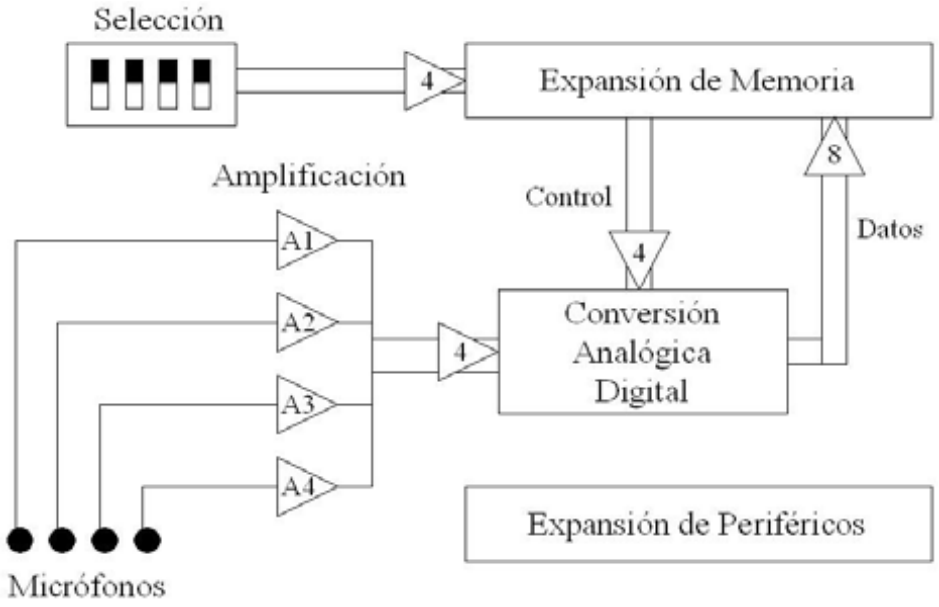


Figura 4.4. Diagrama general de la tarjeta de expansión

Conexión con la tarjeta de desarrollo

La comunicación entre la tarjeta de desarrollo y la tarjeta de expansión diseñada se lleva a cabo por medio de los conectores de expansión de puertos de la tarjeta de desarrollo señalados en la figura 4.2 y 4.3 como ‘EXPANSIÓN DE PERIFÉRICOS’ y ‘EXPANSIÓN DE MEMORIA’ que son dos conectores marca Samtec tipo SFM-140-L2-S-D-LC, por lo que en la tarjeta de

expansión requerimos de los conectores de la misma marca TFM-140-32-S-D-LC. Del conector de expansión de periféricos solo son utilizados algunos pines con el objeto de polarizar a los componentes de la tarjeta de expansión. En la figura 4.5 se muestra el diagrama electrónico de la interfaz de periféricos y en la figura 4.6 se muestra el diagrama electrónico de la interfaz de memoria.

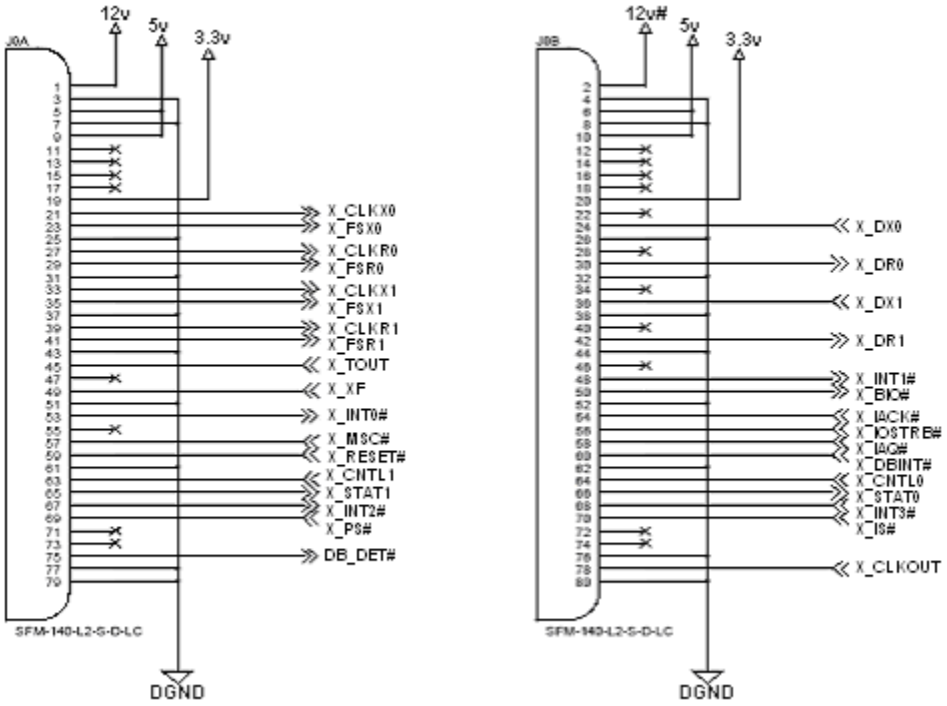


Figura 4.5. Diagrama electrónico de la interfaz de periféricos.

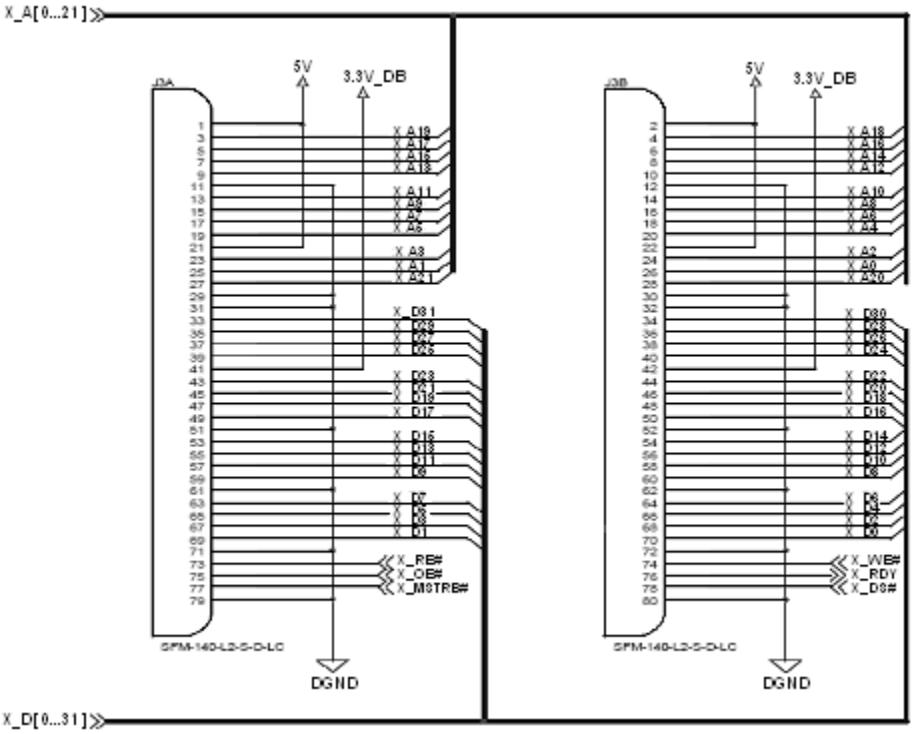


Figura 4.6. Diagrama electrónico de la interfaz de memoria.

La figura 4.7 muestra al conector SFM-140-L2-S-D-LC, conectado a la tarjeta de desarrollo en el conector de expansión 'EXPANSION DE MEMORIA'; los pines señalados como A1 a A4 (pines 25, 24, 23 y 20 del conector) son utilizados en el control del convertidor analógico digital (ADC) y están señalados en la figura 4.2 como control; los pines señalados en la figura como D0 a D7 (pines 70 al 63), son utilizados para transferir datos del ADC al la tarjeta de desarrollo y están señalados en la figura 4.2 como datos; los pines señalados en la figura como D8 a D11 (pines 60 al 57) comunican al selector con la tarjeta de desarrollo.

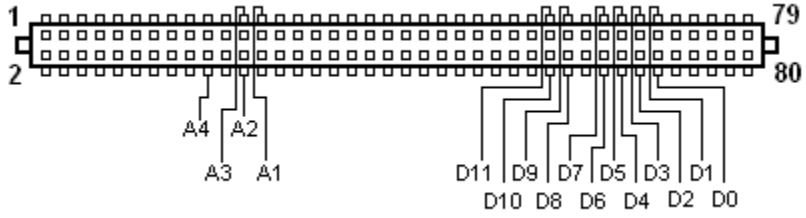


Figura 4.7. Conector Samtec conectado a Expansión de Memoria

Además de los pines señalados en la figura 4.7 utilizamos los pines 2, 11, 12 y 80 del conector 'EXPANSIÓN DE MEMORIA' y los pines 5, 6 y 61 de 'EXPANSIÓN DE PERIFÉRICOS' para polarizar la tarjeta de expansión; y los pines 75 y 79 del último se conectaron entre sí para mantener el control sobre la tarjeta de expansión.

Micrófonos

Las señales acústicas serán adquirida por medio de cuatro micrófonos económicos (marca Akteck modelo MD210) que requieren ser energizados; por ello es necesario que la tarjeta cuente con un circuito de polarización; éste circuito se diseñó con base en el circuito de polarización de la tarjeta de desarrollo. En la figura 4.8 se muestra el circuito de polarización, formado por un capacitor de $10\,\mu F$ y dos resistencias, una de $5.6\,k\Omega$ y la otra de $1\,k\Omega$. La señal Vmic corresponde a la señal obtenida directamente de uno de los micrófonos y Vo es la señal ya amplificada que se alimentará al ADC en cada uno de los pines correspondientes a las entradas analógicas.

Amplificación de señal acústica

La señal en cada micrófono tiene una amplitud de milivolts por lo que es necesario amplificarla. Para ello utilizamos el circuito integrado TLC272, que cuenta con dos amplificadores operacionales y puede ser polarizado con una sola fuente.

En la figura 4.8 se muestra el circuito del amplificador operacional. El voltaje de la terminal no inversora del amplificador operacional se obtiene mediante un divisor de voltaje formado por dos resistencias de $22\,k\Omega$.

La configuración del amplificador operacional es llamada filtro paso banda, de banda ancha, y tiene las siguientes características [17]:

$$G = -\frac{R_f}{R_i} \tag{4.1}$$

$$\omega_L = \frac{1}{R_i C_i} \tag{4.2}$$

$$\omega_H = \frac{1}{R_f C_f} \tag{4.3}$$

donde ω_L y ω_H son las frecuencias de corte baja y alta respectivamente y G es la ganancia del sistema; para los componentes utilizados en el sistema (mostrados en la figura 4.8) la ganancia es de 10 y las frecuencias de corte son de 30 Hz y 10 kHz.

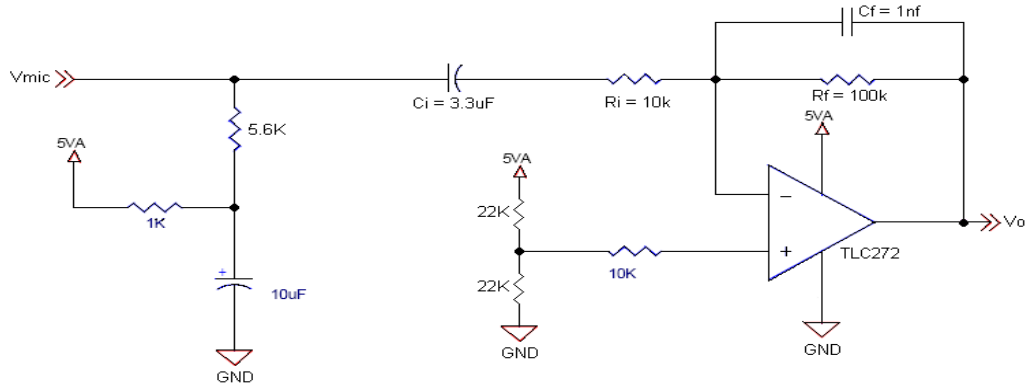


Figura 4.8 Amplificación de la señal de audio

Conversión Analógica Digital

En la elección del convertidor analógico digital hay que considerar que la salida de dicho convertidor debe ser en paralelo y que requerimos de una velocidad de conversión que nos permita convertir señales de cuatro canales en un tiempo menor al tiempo de muestreo de 125 μs por tratarse de señales de voz. Para este propósito utilizamos el convertidor analógico digital ADS7824 de TI que cuenta con cuatro canales de 12 bit con salida en paralelo y convierte las señales en 25 μs cada una.

El ADS7824 se conectó de la forma ilustrada en la figura 4.9. En ella se puede apreciar que los pines 2, 3, 4 y 5, son para señales de entrada (Vo en figura 4.8). Los pines 18 y 19 sirven para seleccionar la entrada analógica a convertir (pines 2 a 5 del ADC) y están conectados a los pines señalados como A4 y A3 en la figura 4.7; la combinación de dos niveles bajos en estos pines selecciona la entrada analógica $Vo1$, A4 bajo y A3 alto la $Vo2$, A4 alto y A3 bajo la $Vo3$ y ambos altos la $Vo4$. El pin 22 es para indicar el inicio de conversión mediante un pulso; el cual es enviado al ADC a través del pin señalado como A1 en la figura 4.7. Los pines 9 al 13 y 15 al 17, son la salida del convertidor y se comunican con el DSP a través de los pines 63 al 70 del

conector ‘EXPANSIÓN DE MEMORIA’ señalados en la figura 4.7 como D7 a D0, en estos pines se puede encontrar la parte más o menos significativa del dato convertido. El pin 21 sirve para elegir entre la parte más y la menos significativa en la salida del convertidor y está conectado al pin señalado, en la figura 4.7, como A2. Cuando el valor de A2 es alto se tiene a la salida del convertidor la parte menos significativa y cuando el valor de A2 es bajo se tiene la más significativa.

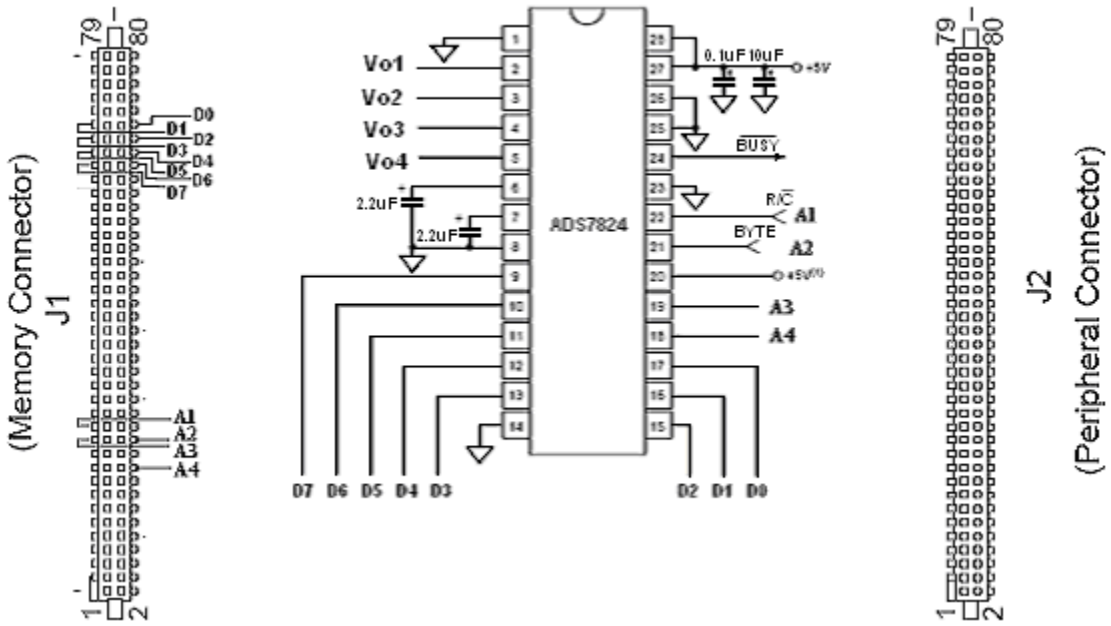
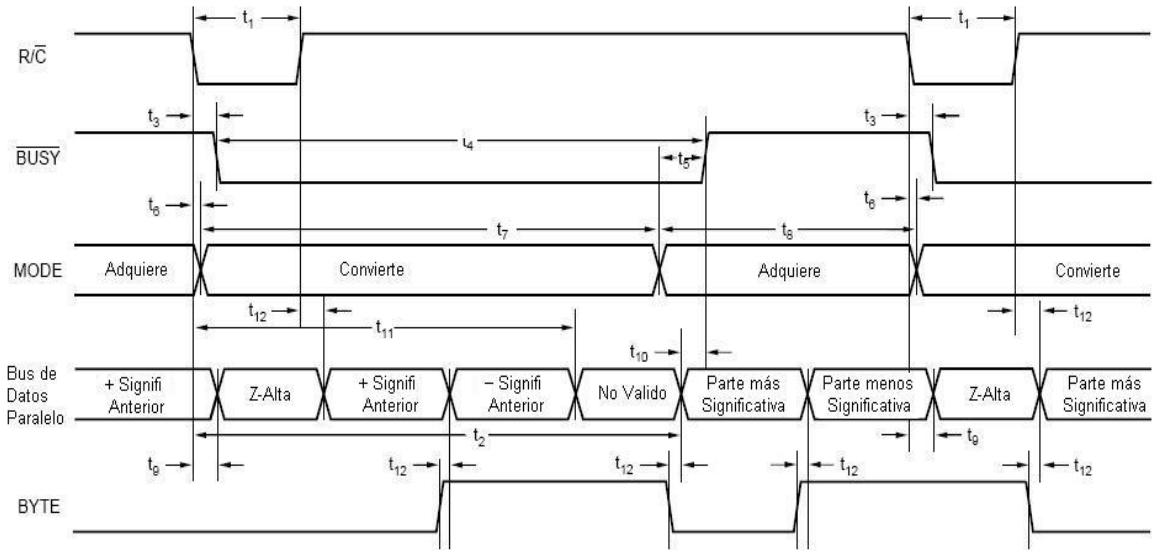


Figura 4.9. Diagrama de conexión del convertidor ADS7824

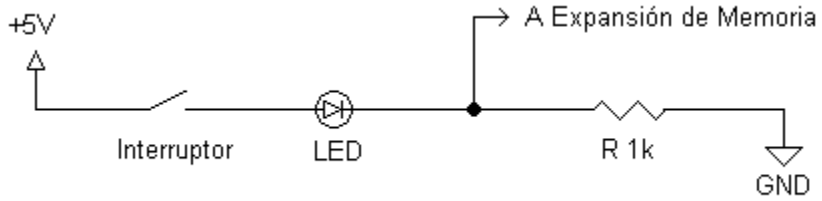
La figura 4.10 es un diagrama de tiempo del ADS7824. Los tiempos de interés para nuestro trabajo son los señalados como t_1, t_7 y t_8. t_1 es el tiempo que dura el pulso de inicio de conversión y debe durar entre $0.04\ \mu s$ y $12\ \mu s$; la duración típica de t_7, que es el tiempo de conversión, es $15\ \mu s$ y su duración máxima es de $21\ \mu s$; la duración típica de t_8, que es el tiempo de adquisición, es $3\ \mu s$ y su duración máxima es de $4\ \mu s$. Sumando las duraciones de t_7 y t_8, obtenemos un tiempo de conversión total (adquisición y conversión) de entre $18\ \mu s$ y $25\ \mu s$.



4.10. Tiempos del convertidor ADS7824

Selección

La selección de los procesos se lleva acabo mediante un conjunto de cuatro interruptores, de 1.0x0.5 *cm*, de dos posiciones (ver figura 4.4); para que el usuario visualice el estado de estos interruptores se utilizo también un LED por cada interruptor y una resistencia para limitar la corriente en el LED y el interruptor. El estado de los interruptores es leído entre cada LED y cada resistencia mediante el respectivo pin del conector 'EXPANSIÓN DE MEMORIA' que están señalados en la figura 4.5 como D8 a D11. La conexión de uno de los interruptores del selector se ilustra en la figura 4.11.



4.11. Diagrama de conexión de un interruptor del selector

4.3 Desarrollo de Software

Una vez que hemos desarrollado el hardware necesario para adaptar las señales y conociendo las señales que se encuentran en los buses de expansión, es necesario conocer la organización de la memoria del DSP, inicializar ciertos registros y conocer las instrucciones necesarias para tener acceso a los buses y diseñar el software para completar nuestro sistema.

El sistema funcionará de la forma ilustrada en la figura 4.12:

- El bloque llamado *configuración del sistema* se refiere a la habilitación de interrupciones, configuración del DSP e inicialización de registros.
- El bloque *lee selector*, realizara la lectura del estado de los interruptores en un ciclo infinito hasta que se presente una interrupción.
- Al *presentarse la interrupción* se evaluará el valor del selector y se procederá a realizar el proceso que corresponda a dicho valor; algunos de los procesos que se pueden realizar son:
    - Adquisición de 1, 2, 3 y 4 micrófonos.
    - Filtrado de la señal (filtro paso-bajas, filtro paso-altas o filtro supresor de banda)
    - Formación de haz (formador de haz fijo o formador de haz adaptable)
- *Una vez realizado* el proceso, el sistema volverá a donde se encontraba antes de la interrupción: a lee selector.

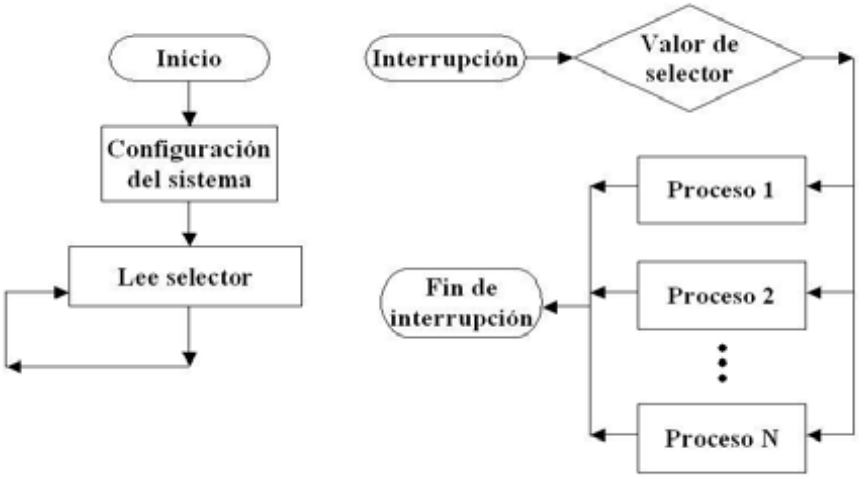


Figura 4.12. Diagrama general de procesos

4.3.1 Memoria del DSP TMS320C5402

La memoria del DSP TMS320C5402 está organizada en tres diferentes espacios: *memoria programa, memoria dato e I/O (entrada/salida)*. La memoria programa contiene las instrucciones que se van a ejecutar, la memoria dato guarda variables e información usada en la ejecución y el espacio de memoria de entrada/salida conecta con periféricos externos al DSP y puede servir para guardar datos en dispositivos externos [18].

Las variables utilizadas en los procesos se guardan en la memoria dato del DSP en memoria DARAM entre las direcciones 0x0080 y 0x4000, por otro lado, cuando se hace la adquisición de datos éstos se guardan en la memoria SRAM (ver figura 4.2) que es externa al DSP entre las direcciones 0x4000 y 0xEFFF. En la figura 4.13 se pueden ver estos espacios de memoria.

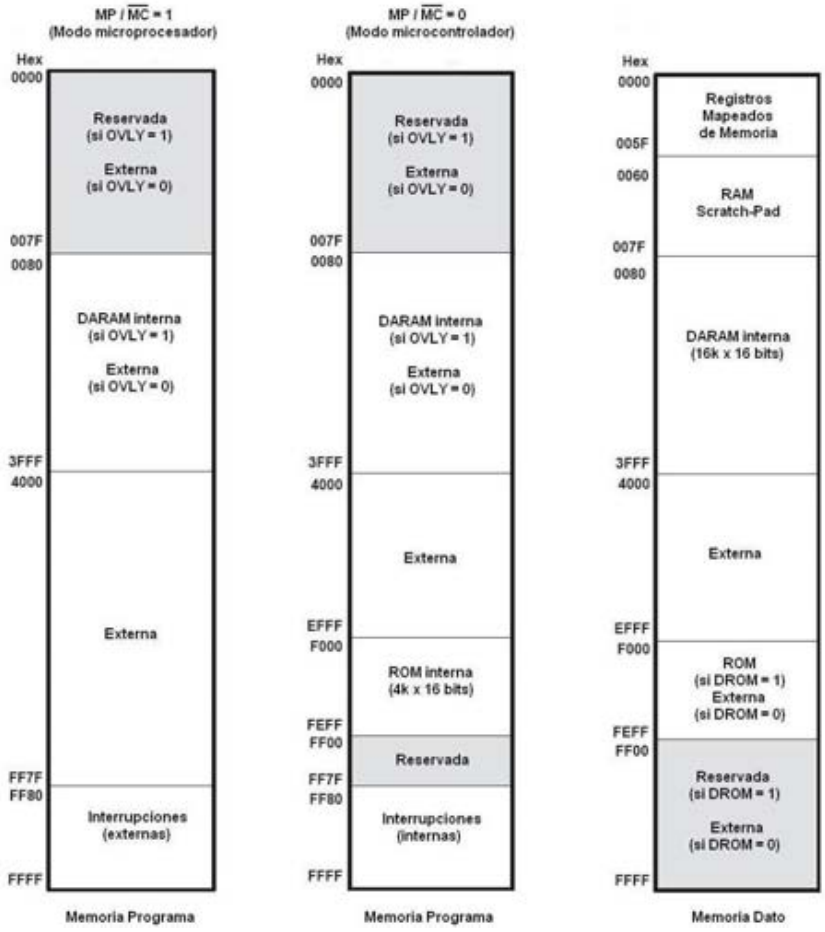


Figura 4.13. Mapa de memoria del DSP C5402

4.3.2 Inicialización de registros

Para poder utilizar la tarjeta es necesario inicializar ciertos registros, entre ellos el registro del modo del procesador (PMST) y el registro de mascara de interrupciones (IMR), además existen siete registros de dispositivo programable de lógica compleja (CPLD) mapeados a las primeras direcciones de memoria entrada/salida, sin embargo para el funcionamiento de la tarjeta solamente se modifican dos registros: el control de memoria de tarjeta de expansión (DMCTRL), en la dirección 0x0002, y el registro de control 2 (CNTL2), en la dirección 0x004.

En el registro PMST se guarda el valor 0x00A0 con lo cual:

- el bit de modo microprocesador/microcontrolador (MP/MC) es igual a 0,
- la memoria ROM se encuentra dentro del chip y
- el bit OVLY vale 1 con lo cual la memoria RAM se encuentra mapeada dentro de la memoria dato y la memoria programa.

En el registro IMR se guarda el valor 0x0800 con lo cual se habilita la interrupción de transmisión de puerto serie McBSP1 llamada BXINT1 que sirve para enviar la salida de audio a una bocina.

En la tabla 4.1 se muestra el contenido del registro CNTL2; de este registro solamente se modifica el bit 5 que sirve para seleccionar entre memoria SRAM o memoria externa o FLASH, para poder leer y escribir en los buses de expansión se utiliza la memoria externa, para tener mayor espacio de memoria se utiliza la memoria SRAM.

El registro DMCTRL se utiliza para seleccionar el tamaño de los datos así como si se van a guardar los datos en la tarjeta DSK o en una Tarjeta de expansión. Cuando se guardan los datos dentro de la tarjeta DSK, los bits 0 – 4 indican la página de memoria en la que se guardan, la tabla 4.2 muestra los bits dentro de este registro.

4.3.3 Lectura de puertos

El DSP TMS320C5402 tiene una interfaz de memoria de 16 bits disponible para una tarjeta de expansión, esta interfaz incluye tres áreas de memoria (dato, programa y entrada/salida), 20 líneas de dirección y señales de control. Las direcciones de memoria no son iguales en las tres

áreas. En la memoria dato se tiene el intervalo de 0x00000 a 0xFFFFF para la tarjeta de expansión, los datos pueden estar en 16 bits o en 32 bits, esto es, se tiene 1M x 16 o 512k x 32 bits de memoria.

BIT	NOMBRE	R/W	DESCRIPCIÓN
7	DAAOH	RW	DAA control de colgado de línea telefónica (0 = colgado, 1 = descolgado)
6	DAACID	RW	DAA identificación de llamada (0 = deshabilitado, 1 = habilitado)
5	FLASHENB	RW	Selección de FLASH o memoria externa (= 1) ; SRAM (= 0)
4	INT1SEL	RW	Interrupción 1 (0 = UART, 1 = daughterboard)
3	FC1CON	RW	Bit de control de MIC/bocinas AD50 FC (0)
2	FC0CON	RW	Bit de control de DAA AD50 FC (0)
1	BSPSEL1	RW	McBSP1 selector (0 = mic/bocina, 1 = daughterboard)
0	BSPSEL0	RW	McBSP0 selector (0 = TelSet DAA, 1 = daughterboard)

Tabla 4.1. Definición de bits del registro de control CNTL2.

BIT	NOMBRE	R/W	DESCRIPCIÓN
7	DM_SEL	RW	Selector de memoria dato (0 = memoria en tarjeta 1 = Daughterboard)
6	DB_WIDE	RW	Selector de tamaño de datos de Daughterboard (0=16, 1= 32)
5	DB_32ODD	RW	Modo de acceso de dirección para 32-bits (0=par, 1=non)
4	M_PG4	RW	FLASH/SRAM/External Data o bit 4 de página de memoria I/O.
3	M_PG3	RW	FLASH/SRAM/External Data o bit 3 de página de memoria I/O.
2	M_PG2	RW	FLASH/SRAM/External Data o bit 2 de página de memoria I/O.
1	M_PG1	RW	FLASH/SRAM/External Data o bit 1 de página de memoria I/O.
0	M_PG0	RW	FLASH/SRAM/External Data o bit 0 de página de memoria I/O.

Tabla 4.2. Definición de bits del registro de control DMCTRL.

La memoria de entrada/salida se mapea en el intervalo de 0x00000 a 0xFFFFF, en localidades de 16 bits por lo tanto se tienen 1M x 16 bits de memoria. La memoria programa tiene menos espacio debido a la memoria programa de la tarjeta de desarrollo. Se puede disponer de la mitad de la memoria externa: 0x80000 – 0xFFFFF, los datos solo pueden ser de 16 bits por lo que se tienen 512k x 16 bits de memoria.

4.4 Uso de la Tarjeta de Expansión y descripción de funciones

La tarjeta de expansión diseñada puede realizar los procesos descritos en la tabla 4.3; el proceso a realizarse será determinado por el estado de los interruptores de usuario de la interfaz, el interruptor llamado Mic1 está señalado con un LED rojo en la interfaz y se encuentra arriba del conector correspondiente al micrófono número 1; los números de micrófono e interruptor se cuentan a partir del primero en forma consecutiva. El valor de cualquiera de los interruptores es 1, cuando el interruptor está hacia arriba y el LED, entre el interruptor y el conector del mismo número, se encuentra encendido.

Numero de proceso	Estado de los interruptores Mic4			Mic1	Descripción de funciones	Categoria
1	0	0	0	1	Lee micrófono número 1 y reproduce por bocina	Lee y reproduce
2	0	0	1	0	Lee micrófono número 2 y reproduce por bocina	
3	0	1	0	0	Lee micrófono número 3 y reproduce por bocina	
4	1	0	0	0	Lee micrófono número 4 y reproduce por bocina	
5	1	1	1	1	Inicio de grabación	Grabación
6	1	1	1	0	Graba en memoria cuatro segundos de voz del canal número 1	
7	1	1	0	1	Graba en memoria cuatro segundos de cada uno de los canales 1 y 2	
8	1	0	1	1	Graba en memoria dos segundos de cada uno de los canales 1, 2 y 3	
9	0	1	1	1	Graba en memoria dos segundos de cada uno de los canales 1, 2, 3 y 4	
10	1	1	0	0	Filtro paso altas de señal de voz en micrófono 1	Filtros
11	0	0	1	1	Filtro Paso bajas de señal de voz en micrófono 1	
12	0	1	1	0	Filtro supresor de banda de señal de voz en micrófono 1	
13	0	0	0	0	Formador de haz fijo de los cuatro micrófonos	Formación de Haz
14	1	0	0	1	Formador de haz de dos micrófonos	

Tabla 4.3 Procesos de la Tarjeta de expansión

Las combinaciones 1010 y 0101 de los interruptores quedan disponibles para el uso de futuros programadores.

Las funciones que se pueden realizar a través de la tarjeta de expansión diseñada, en conjunto con la tarjeta DSK y el software se pueden clasificar en las categorías:

- Lectura y reproducción
- Grabación simultánea de señales de varios micrófonos
- Filtrado de señales
- Formación de haz

estos procesos tienen por objeto la evaluación del funcionamiento del sistema y el uso de los mismos. A continuación se presenta una descripción general de estos procesos y más adelante se describe cada proceso más detalladamente:

- Los procesos *uno* al *cuatro* hacen uso del selector de la tarjeta de expansión y de sus conectores para micrófono (con el fin de comprobar el funcionamiento de los mismos, y en su caso detectar posibles errores de conexión), así como del conector de la tarjeta de desarrollo destinado a la conexión de una bocina. La tabla 4.3 indica que estos procesos realizan la *lectura* de cada uno de los micrófonos de forma individual.

- El proceso número *cinco* sirve para *inicializar* los *registros* necesarios para la *grabación*; este proceso inicializa los apuntadores que indican la dirección en que se iniciaran a guardar los datos de los procesos *seis* al *nueve*. La transición entre el proceso cinco y cualquiera de los procesos del seis al nueve se realiza moviendo únicamente uno de los interruptores.

- Los procesos *seis* al *nueve* guardan la información de uno, dos tres o cuatro canales, la sección 4.4.2 de este capítulo describe las consideraciones que se debe tener con estos procesos. Estos datos se almacenan en memoria SARAM para su uso en futuras aplicaciones o para su procesamiento fuera de línea.

- Los procesos 10 al 12 son *filtros digitales* tipo FIR, su modo de operación y diseño se explica en la sección 4.4.3 de este capítulo.

- El formador de haz de cuatro micrófonos (proceso número 13) es un formador de haz fijo con ventana cuadrada como el descrito en la sección 3.2 de este trabajo. Por otra parte el

formador de haz de dos micrófonos (proceso 14) es adaptable y corresponde al descrito en la sección 3.6.3.

4.4.1 Lectura y reproducción

Para *leer o escribir* mediante el bus de expansión de memoria se tiene que trabajar con direcciones mayores a la 0x8000; con el fin de acceder a la tarjeta de expansión diseñada, a continuación se muestra un programa de prueba en lenguaje C que lee y escribe datos de un puerto externo a través del bus de expansión de memoria:

```
#include <type.h>
int j;
int val1,val2,val3;                    /*Variables enteras para asignar valores*/
int lect;
ioport unsigned port8002;              /*Direcciones de puertos de entrada y salida*/
ioport unsigned port8008;
ioport unsigned port800A;
void espera (void);

void main()
{
 lect=15;
 val1=0x00;                            /*Asignación de valores a las variables*/
 val2=0xAA;
 val3=0xFF;
 while(1) {
   lect=port8002;                      /*Lectura del puerto 8002*/
   port8008=val1;                      /*Escribe val1 al puerto 8008*/
   espera();
   lect=port8008;                      /*Lectura del puerto 800A*/
   port800A=val2;                      /*Escribe val3 al puerto 8002*/
   espera();
   lect=port800A;                      /*Lectura del puerto 800A*/
   port8002=val3;                      /*Escribe val3 al puerto 8002*/
   espera();
  }
 }
```

```
void espera (void)                          /*Ciclo de espera para poder distinguir */
{                                           /*los cambios*/
       int i,temp;
       for (i=0;i<30000;i++)
       {       i=i;
       }
       return;
}
```

Una vez que se probó el funcionamiento de los puertos, se utilizan las señales necesarias para el control del ADS (ver apartado 4.2 de este capítulo) y se inicializan los registros para poder leer y reproducir las señales de los micrófonos. En la figura 4.14 se muestra el esquema general de la lectura y reproducción de datos a través de la rutina de atención de interrupciones.

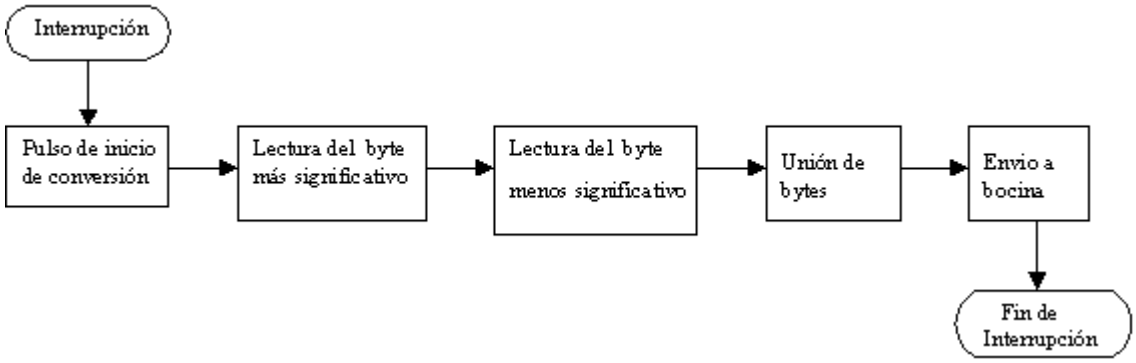


Figura 4.14. Lectura y reproducción de datos

El siguiente programa inicializa los registros IMR y PMST para después quedarse esperando una interrupción, que sucede cada 125 μs (8000 Hz), en cada interrupción se adquiere la señal de un solo micrófono y posteriormente la envía a la bocina conectada a la tarjeta DSK.

```
#include <type.h>
#include <board.h>
#include <codec.h>

#define GLOBAL_INT_ENABLE     asm( " rsbx intm ")
#define GLOBAL_INT_DISABLE    asm( " ssbx intm ")

volatile short *imr = (short *) 0x00;         /*Direcciones para el control de*/
volatile short *ifr = (short *) 0x01;         /*interrupciones*/
volatile short *pmst= (short *) 0x1d;
volatile short *dxr = (short *) 0x43;         /*Direccion de envio a bocina */
int i,j;
```

```
short data, data1;                      /*Variables utilizadas para la lectura*/
short data2;                            /*del convertidor*/

HANDLE hHandset;

ioport unsigned port8000;       /*Inicialización de las direcciones*/
ioport unsigned port8002;       /*de entrada y salida que se utilizarán*/
ioport unsigned port8006;       /*para controlar el ADS*/
void espera(void);

interrupt void mandar(void)
{
      data=port8000&0xFF;       /*Envia pulso para inicio de conversión*/
      espera();                 /*Espera entre 0.04 µseg y 12 µseg*/
      data=port8002&0xFF;       /*Adquisición del byte más significativo*/
      data1=port8006&0xFF;      /*Adquisición del byte menos significativo*/
      data2=data<<8;
      data2 |=data1;            /*Se juntan los bytes*/
      *dxr=data2;               /*envío de la señal a las bocinas*/
}

void main(void)                 /*Programa principal*/
 {
    GLOBAL_INT_DISABLE;
    brd_init_bios();
    hHandset = codec_open(HANDSET_CODEC);           /*inicialización del codec*/
    codec_dac_mode(hHandset, CODEC_DAC_15BIT);
    codec_adc_mode(hHandset, CODEC_ADC_15BIT);
    codec_ain_gain(hHandset, CODEC_AIN_6dB);
    codec_aout_gain(hHandset, CODEC_AOUT_MINUS_0dB);
    codec_sample_rate(hHandset,SR_8000);            /*Muestreo a 8000 Hz*/

    *pmst = 0x00A0;
    *imr |= 0x0800;             /*habilita interrupciones de bxint*/
    *ifr = *ifr;
    port4 = 0x21;

  GLOBAL_INT_ENABLE;            /*Habilita las interrupciones*/


   while (1) {                  /*Ciclo infinito*/
             };
}

void espera (void)              /*rutina de retardo*/
{
      int i;
      for (i=0;i<200;i++)
       {
             i=i;
       }
      return;
}
```

En el anexo B se pueden ver las versiones finales de los programas utilizados en este proyecto, en ellos se incluyen comentarios para poder comprenderlos más fácilmente.

4.4.2 Guardado en memoria

En la tarjeta se tienen cuatro diferentes opciones de guardado de señales; con la *primera* opción (proceso *seis*) se guardan cuatro segundos de la señal que llega al micrófono 1, con la *segunda* opción (proceso *siete*) se guardan cuatro segundos de los micrófonos 1 y 2, la *tercera* opción (proceso *ocho*) sirve para guardar dos segundos de los micrófonos 1, 2 y 3, la *cuarta* opción (proceso *nueve*) es para guardar dos segundos de los cuatro micrófonos.

Antes de grabar se tiene que seleccionar con los interruptores el proceso *cinco*, con ello se inicializan los valores de los apuntadores necesarios en la grabación, después se escoge el proceso dependiendo el número de micrófonos que se vayan a utilizar. El proceso cinco se selecciona teniendo todos los interruptores arriba (encendidos) y para grabar basta con bajar el interruptor cuyo número corresponde a la cantidad de canales que se adquirirán por lo que la transición es muy sencilla; además, para realizar la grabación de otro conjunto de señales basta con regresar el interruptor del número de señales que este abajo y bajar el interruptor correspondiente al numero de señales que se desee gravar a continuación.

Los datos se guardan en memoria SRAM, para ello se modifica el bit 5 del registro CNTL2 (ver Tabla 4.1), se escribe un cero en este bit para habilitar la memoria SRAM y una vez que se guardan los datos se escribe un uno para volver a habilitar la tarjeta de expansión.

Para poder guardar más datos se utilizó otra página de memoria SRAM, esto se hizo modificando el registro DMCTRL (ver Tabla 4.2), cuando se guarda la entrada de un solo micrófono todos los datos se guardan en la página cero desde la dirección 0x4004 hasta la dirección 0xBD04 (32k), cuando se guardan las entradas de dos micrófonos los datos de uno de ellos quedan en la página cero y los datos del otro se guardan en la página uno, ambos desde la dirección 0x4004 hasta la dirección 0xBD04 (2 x 32k). Cuando se guardan los datos de tres o cuatro micrófonos los datos de los micrófonos 1 y 2 se guardan en la página cero y el resto se guarda en la página uno, los datos de los micrófonos 1 y 3 se guardan desde la dirección 0x4004 hasta la dirección 0x7E83 (2 x 16k) y los datos de los micrófonos 2 y 4 se guardan desde la dirección 0x7E84 hasta la dirección 0xBD04 (2 x 16k).

Los datos guardados pueden ser utilizados para analizar las señales que llegan a los micrófonos o para procesar estas señales fuera de línea, es decir, que con el ambiente de desarrollo se pueden salvar en un archivo.

4.4.3 Filtros digitales

Entre los procesos que se pueden realizar con la tarjeta de expansión se encuentra el filtrado de señales; filtrar es el proceso de seleccionar o suprimir ciertas frecuencias de una señal. Un filtro digital es un algoritmo matemático, que se puede expresar como una ecuación en diferencias y puede atenuar o suprimir ciertas frecuencias.

Los filtros digitales tienen algunas ventajas respecto a los filtros analógicos: tienen menor sensibilidad a las condiciones ambientales como la temperatura, tienen menor tamaño físico, el costo es menor cuando son filtros de orden grande, pueden cambiarse sin tener que cambiar el hardware y todos los filtros digitales, diseñados bajo el mismo esquema, son idénticos por no depender su funcionamiento de componentes con tolerancia.

Los filtros digitales más simples son aquellos en los que solamente se toman en cuenta las entradas anteriores y la entrada actual; es decir, no existe retroalimentación, por lo tanto cuando a la entrada de estos filtros se aplica un impulso, después de cierto tiempo la salida será igual a cero, por ello se dice que la respuesta al impulso es finita. Este tipo de filtro se llama filtro de respuesta al impulso finita o filtro FIR y su ecuación en diferencias es:

$$y[n] = \sum_{k=0}^{N-1} h[k] \cdot x[n-k]$$

$$y[n] = h_0 x(n) + h_1 x(n-1) + \cdots + h_N x(n-N+1) \qquad (4.4)$$

donde N es el orden del filtro, $h(k)$ sus coeficientes, $x[n]$ la señal de entrada y $y[n]$ la señal de salida.

Existen también filtros que consideran valores anteriores de la salida. Estos filtros son llamados IIR (respuesta infinita al impulso) porque si se aplica a ellos una entrada tipo impulso, la salida será siempre diferente de cero por existir una retroalimentación de las salidas anteriores. Estos filtros, también llamados filtros recursivos, se diseñan ‘mapeando’ diseños analógicos pertenecientes a una clasificación de acuerdo a la forma de su respuesta en frecuencia. Los tipos

mas conocidos, a las que puede pertenecer un filtro IIR, son Chebyshev y Butterworth. La salida de estos filtros se modela con la ecuación 4.5

$$y[n] = \sum_{i=0}^{q} b[i] \cdot x(n-i) - \sum_{j=1}^{p} a[j] \cdot y[n-j] \qquad (4.5)$$

donde q es el orden del filtro, $b[i]$ son los coeficientes asociados a la entrada, $a[j]$ los coeficientes asociados a la salida, $y[n]$ la salida del filtro y $x[n]$ la entrada del filtro.

Otra forma de clasificar a los filtros, además de por su respuesta al impulso, es según la parte del espectro que atenúan o que dejan pasar, según este criterio se tienen:

- Filtros pasa bajas
- Filtros pasa altas
- Filtros pasa banda
- Filtros supresores de banda

Dentro de los procesos para pruebas de filtrado que se pueden realizar con la tarjeta de expansión se encuentran tres filtros: pasa bajas, pasa altas y supresor banda; las características de estos filtros serán mencionadas más adelante.

Diseño de los filtros

Puede realizarse una gran cantidad de filtros digitales con el sistema diseñado, las respuestas de estos estarán determinadas por sus coeficientes. Para mostrar su efecto en las señales se implementaron tres filtros digitales tipo FIR de los cuales dos son de orden 100, es decir, de 100 coeficientes y el otro es de orden 200. Los filtros implementados son tipo FIR, por ser esta configuración la que más recursos del sistema utiliza (un filtro IIR con respuesta similar seria de mucho menor orden) y porque buscamos explotar la capacidad del DSP.

Los coeficientes $h(n)$ de los filtros digitales se determinaron mediante la herramienta 'fdatool' de Matlab (descrita brevemente en el anexo C), considerándose para todos los diseños una ventana de Hamming. El orden de los filtros FIR paso bajas y paso altas es $N = 100$ y del supresor de banda es 200. Las frecuencias de corte fueron seleccionadas arbitrariamente y son

para los filtros paso baja y paso altas son de $1500\,Hz$ y $800\,Hz$ respectivamente; el filtro supresor de banda suprime la frecuencia $1000\,Hz$ y sus frecuencias de corte son de $925\,Hz$ y $1075\,Hz$ con frecuencia de muestreo (Fs) de $8000\,Hz$. La ganancia en la frecuencia de corte (Gpaso) de las figuras es de $-3\,dB$ y la Ganancia de supresión (Gsupresion) es de $-80\,dB$ para los filtros paso bajas y supresor de banda y de $-45\,dB$ para el paso altas. El filtro paso bajas deja pasar todas las frecuencias menores a $1438.4\,Hz$ y elimina completamente las mayores a $1614.4\,Hz$ esto puede apreciarse en la figura 4.15. El filtro paso altas elimina completamente las frecuencias menores a $729.6\,Hz$ y deja pasar las mayores a $864.8\,Hz$, como se muestra en la figura 4.16. El filtro supresor de banda elimina la banda de frecuencias entre $979.4\,Hz$ y $1023\,Hz$ y deja pasar las menores a $893.4\,Hz$ y las mayores a $1108\,Hz$, ver figura 4.17.

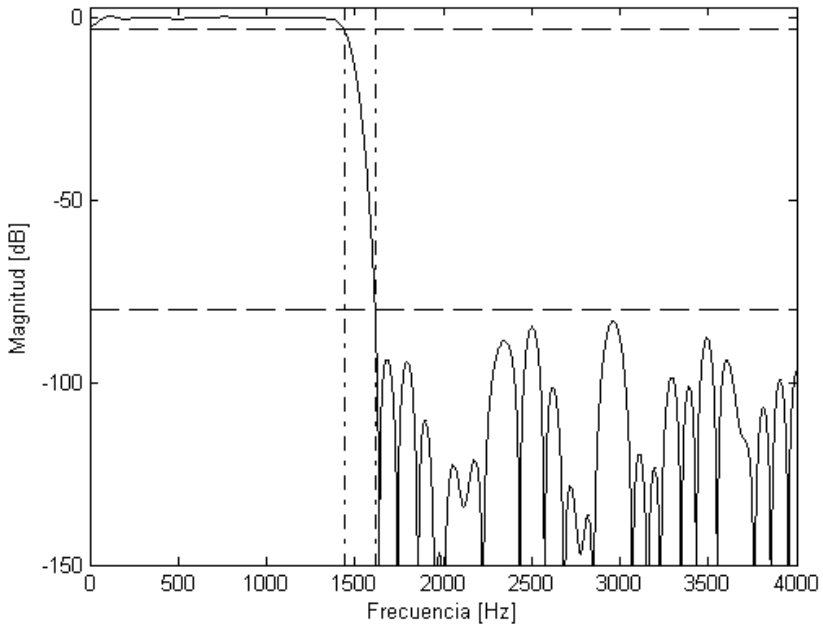


Figura 4.15. Respuesta del filtro paso bajas

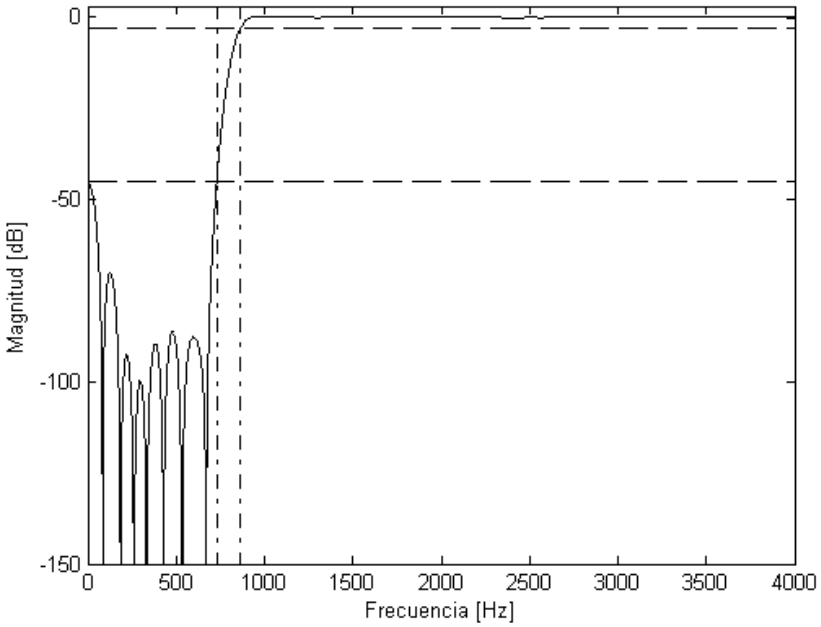


Figura 4.16. Respuesta del filtro paso altas

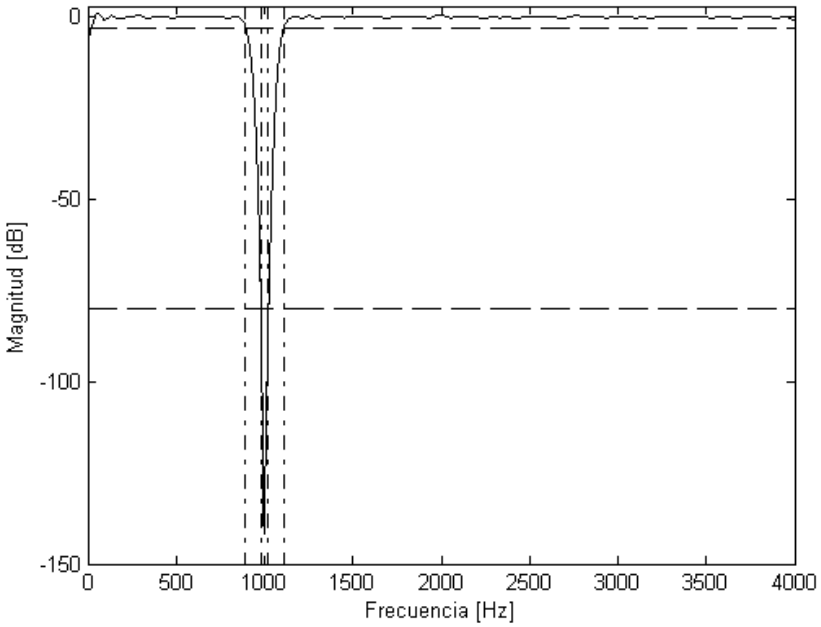


Figura 4.17. Respuesta del filtro supresor de banda

El algoritmo de estos filtros se realiza de la siguiente manera, primero se guardaron los coeficientes $h(k)$ de cada uno de ellos, después, dependiendo del estado del selector se entra a la sección de programa del filtro elegido, se hace un movimiento de 100 o 200 datos dependiendo la longitud del filtro, se obtiene un nuevo dato de la señal de entrada, se hace la multiplicación y adición de los datos con los coeficientes del filtro y finalmente se envía la salida calculada a la bocina. En la figura 4.18 se muestra un esquema general del filtrado y en el anexo B se pueden ver las versiones finales de los programas para una mejor comprensión.

Figura 4.18. Proceso de filtrado

4.4.4 Formador de haz

Los procesos 13 y 14, de la tabla 4.3, a realizar con la tarjeta de expansión son formadores de haz; el proceso 13 es un formador de haz fijo de cuatro micrófonos y el proceso 14 es un formador de haz adaptable de dos micrófonos.

El *formador de haz fijo* (FBF) se logra haciendo la adquisición y suma simultanea de las señales de los cuatro micrófonos; se puede multiplicar cada micrófono por una ganancia, en este caso se utiliza una ganancia unitaria en todos los micrófonos. El proceso es muy sencillo debido a ello es un formador de haz muy básico y su respuesta no es muy buena. La figura 4.19 muestra el diagrama general del FBF.

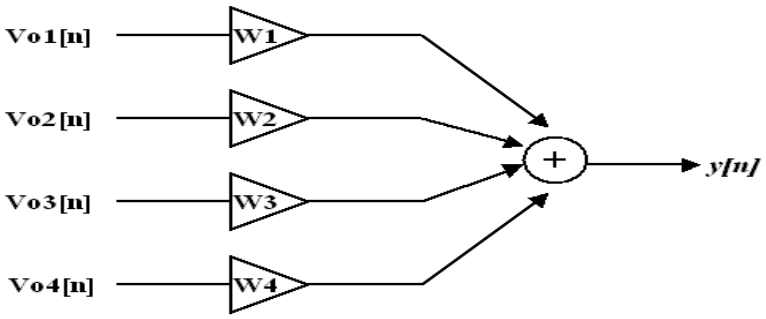


Figura 4.19. Formador de haz fijo

Las señales $Vo1[n]$, $Vo2[n]$, $Vo3[n]$ y $Vo4[n]$ corresponden a la señal de voz amplificada y digitalizada de cada micrófono; $W1$, $W2$, $W3$ y $W4$ son los coeficientes de amplificación para cada una de las señales $Vo[n]$ y para el caso particular del programa implementado todas tienen un valor unitario. Entonces la salida $y[n]$ de este sistema se puede expresar:

$$y[n] = Vo1[n] \cdot W1 + Vo2[n] \cdot W2 + Vo3[n] \cdot W3 + Vo4[n] \cdot W4$$
$$= Vo1[n] + Vo2[n] + Vo3[n] + Vo4[n] \quad (4.5)$$

El *formador de haz adaptable* que se implementó en este proyecto utiliza el algoritmo cancelador de lóbulos laterales (GSC) (ver apartado 3.6.3), sin embargo solamente se utilizaron dos entradas de micrófono debido a limitaciones del sistema. En la figura 4.20 se muestra el diagrama a bloques de este algoritmo, en este diagrama:

$$FBF[n] = Vo1[n] + Vo2[n] \quad (4.6)$$
$$U1[n] = Vo1[n] - Vo2[n] \quad (4.7)$$
$$ze[n] = y1[n] - FBF[n - d] \quad (4.8)$$

4.5 Resumen

En este capítulo se describió de forma breve el proceso que se tuvo que llevar a cabo para el diseño de la tarjeta de expansión; para ello se comenzó hablando de las características de la tarjeta de desarrollo que se deben considerar para el diseño de la tarjeta de expansión; después se describió el proceso de selección de los componentes y de la forma en que iban a estar interconectados; se continuó con la descripción del software involucrado en el funcionamiento de

la tarjeta, y se finalizó con una breve explicación de el modo de uso de la tarjeta y una breve descripción de los procesos que ésta realiza.

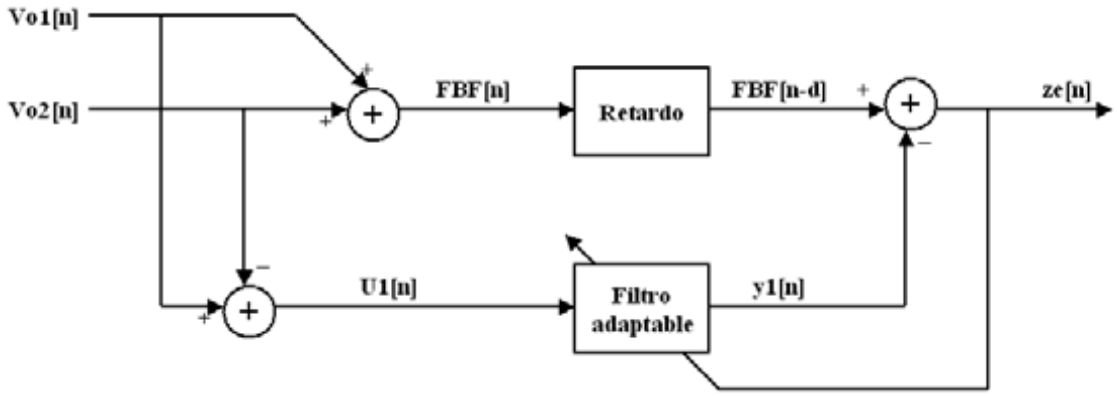


Figura 4.20. Formador de haz adaptable GSC

Capitulo 5
Pruebas y resultados

En el presente capítulo se describen las pruebas que se realizaron con la tarjeta de expansión diseñada y se muestran los resultados obtenidos en los diferentes procesos que se pueden realizar en la tarjeta. Para su mejor entendimiento se dividieron los procesos en tres bloques, el primero trata de la adquisición simultánea de una o varias señales de voz; el siguiente trata sobre los filtros digitales implementados y el final de los formadores de haz; además se realiza una evaluación de los recursos del sistema.

5.1 Adquisición simultanea de varias señales de voz de la misma fuente

Como se señaló en el capitulo 4, la tarjeta de expansión diseñada puede guardar hasta cuatro señales de voz simultáneas; la figura 5.1 muestra las cuatro señales de voz de dos segundos adquiridas en el laboratorio que provenían de la misma fuente, situada en el mismo plano de los micrófonos con un ángulo de $75°$ respecto a la normal al plano, se puede apreciar en la figura que las señales son muy similares y presentan cierto retraso. La figura 5.2 muestra la correlación de la señal del micrófono uno ($Xn1$) consigo misma y con cada una de las señales de los otros micrófonos; se puede observar que las señales están ampliamente relacionadas y que existe un retraso en los máximos de las correlaciones que es particularmente notorio en la parte central de las gráficas y es por ello que solo se muestran las partes centrales de las correlaciones.

Analizando las gráficas de correlación de la señal $Xn1$ (figura 5.2) consigo misma r_{x1x1} y con las otras señales podemos apreciar una diferencia en la posición de los máximos de dichas gráficas, encontrándose en el centro de la gráfica para la autocorrelación y desplazado en una muestra para las correlaciones con las señales $Xn2$ y $Xn3$ y en dos para la correlación con la señal $Xn4$; este desplazamiento se debe al tiempo que requiere el frente de onda para ir de un

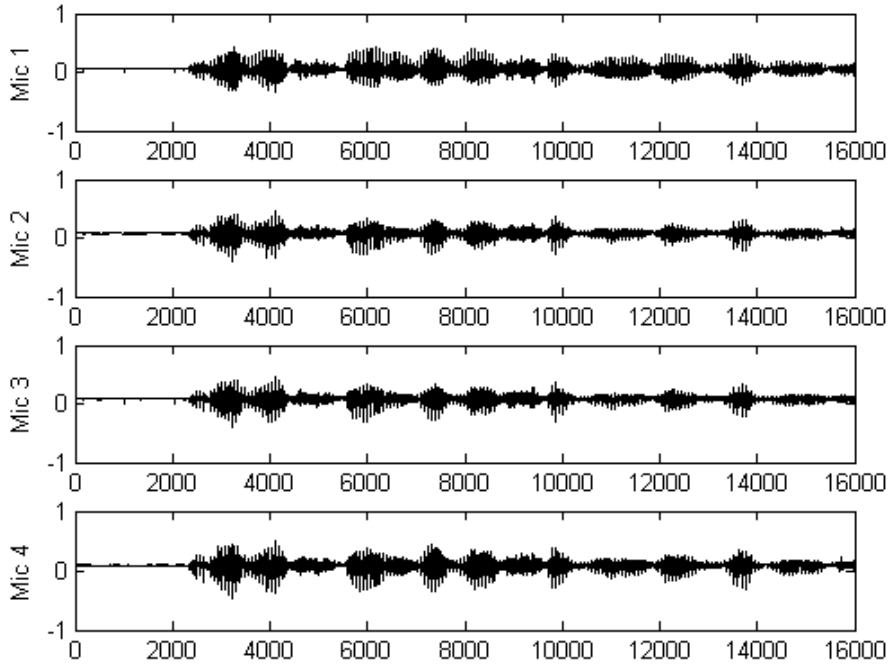


Figura 5.1. Señales de audio adquiridas

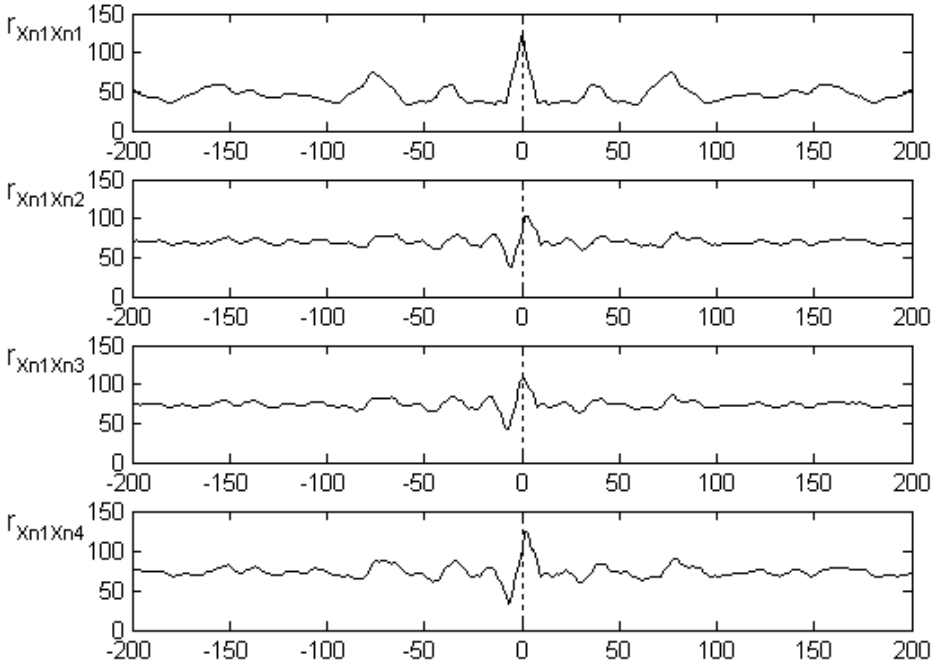


Figura 5.2. Correlación entre señales de audio

micrófono a otro, como se muestra en la ecuación (3.8). Entonces la diferencia de tiempo puede obtenerse como

$$\tau = \frac{d}{v}\text{sen}\phi \qquad (5.1)$$

donde τ es el retardo, v la velocidad del sonido, d la distancia entre los micrófonos y ϕ el ángulo respecto a la normal al plano de los micrófonos; entonces tenemos:

$$\tau = \frac{0.03[m]}{343[m/s]}\text{sen}(75°) = 84.5[\mu s] \qquad (5.2)$$

Ahora bien como la frecuencia de muestreo es de $8\ kHz$ el tiempo entre muestra y muestra es de $t_m = 1/(8\ kHz) = 125\ \mu s$; por lo que en el cuarto micrófono tendremos un retardo de:

$$\tau_4 = \frac{3\tau}{t_m} = \frac{3 \cdot 84.5}{125} = 2.3[muestra] \qquad (5.3)$$

5.2 Filtros Digitales

En la tarjeta de expansión diseñada también se pueden seleccionar tres tipos de filtros digitales, con las características señaladas en el capítulo cuatro, para evaluar su desempeño se grabaron las señales de entrada y salida del micrófono uno y se graficaron dentro del ambiente de trabajo (Code Composer Studio), las señales obtenidas se muestran de forma gráfica en las figuras 5.3 a 5.5 y son respectivamente del filtro paso bajas (de orden 100 con $f_c = 1500\ Hz$), paso altas (de orden 100 y $f_c = 800\ Hz$) y supresor de banda (de orden 200 y $f_0 = 1000\ Hz$); la gráfica de la parte superior en estas figuras muestra la señal de entrada y la inferior la señal de salida. La señal de entrada para los filtros fue un barrido de señales del generador de señales combinada con un silbido.

Las figuras 5.6 a 5.8 muestran los espectros de las señales de los filtros, paso bajas, paso altas y supresor de banda respectivamente, tanto a la entrada (gráfica superior) como a la salida (gráfica inferior) de los filtros. También se incluye, en la gráfica de la salida del filtro, con una línea discontinua, la respuesta en frecuencia asociada a los coeficientes del filtro.

En los espectros se puede observar claramente que los filtros cumplen de manera satisfactoria con su función, ya que eliminan las frecuencias de las respectivas bandas de supresión y conservan las de las respectivas bandas de paso.

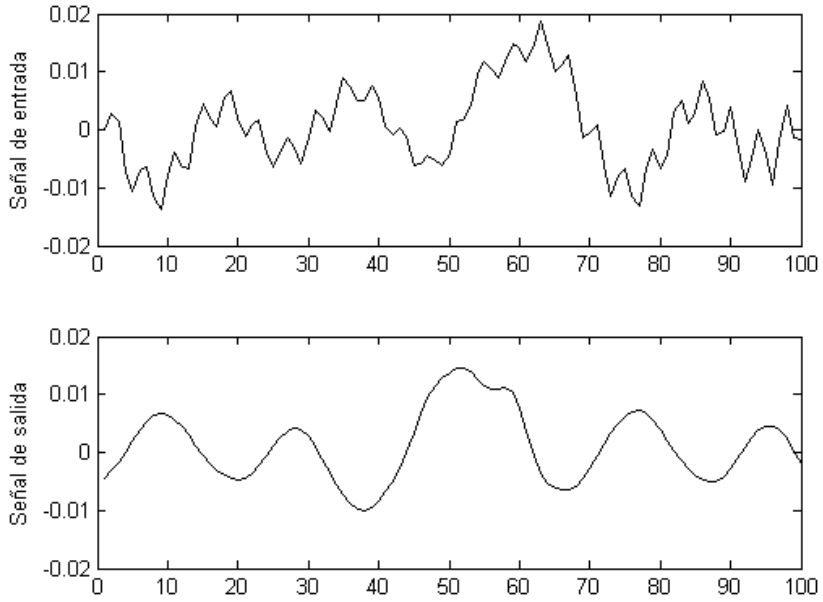


Figura 5.3. Señales de entrada y salida del filtro paso bajas

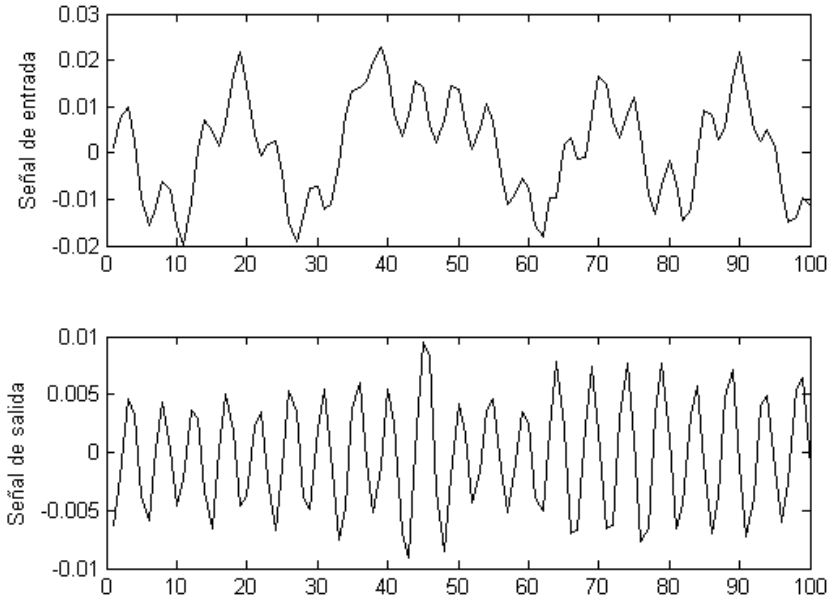


Figura 5.4. Señales de entrada y salida del filtro paso altas

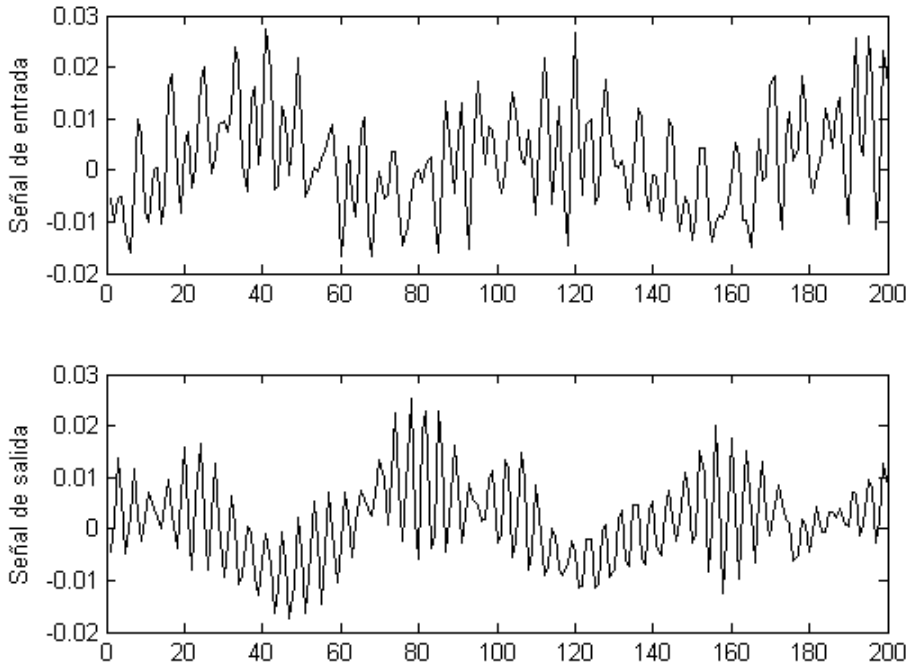


Figura 5.5. Señales de entrada y salida del filtro supresor de banda

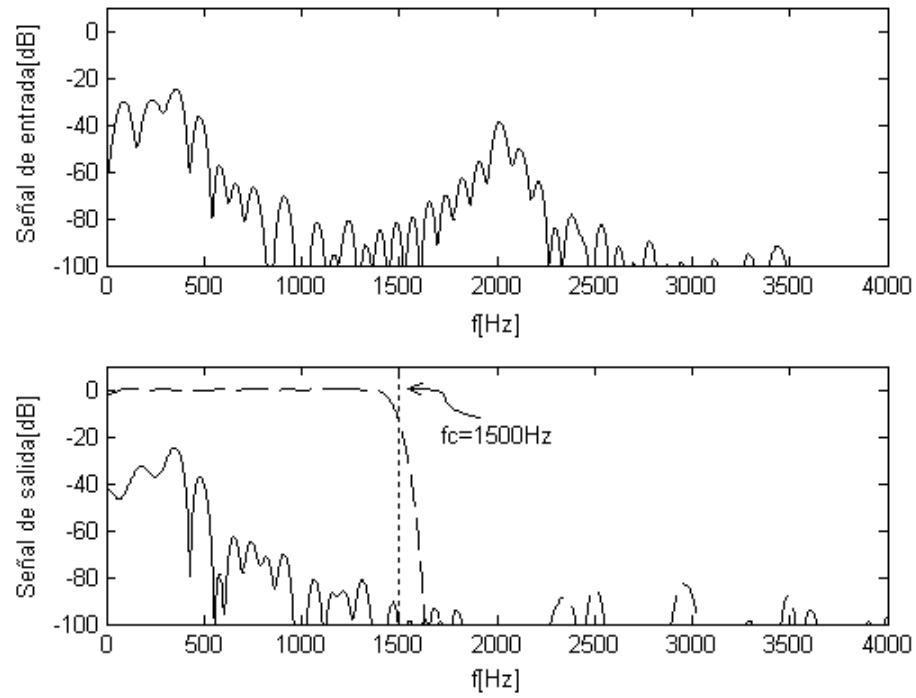


Figura 5.6. Espectro de señales de entrada y salida del filtro paso bajas.

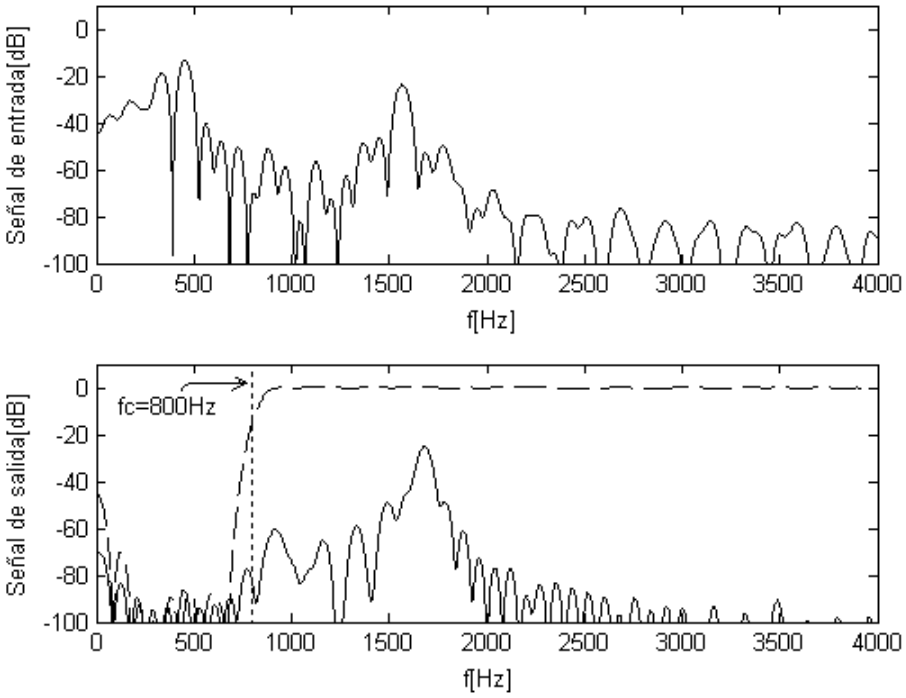


Figura 5.7. Espectro de señales de entrada y salida del filtro paso altas

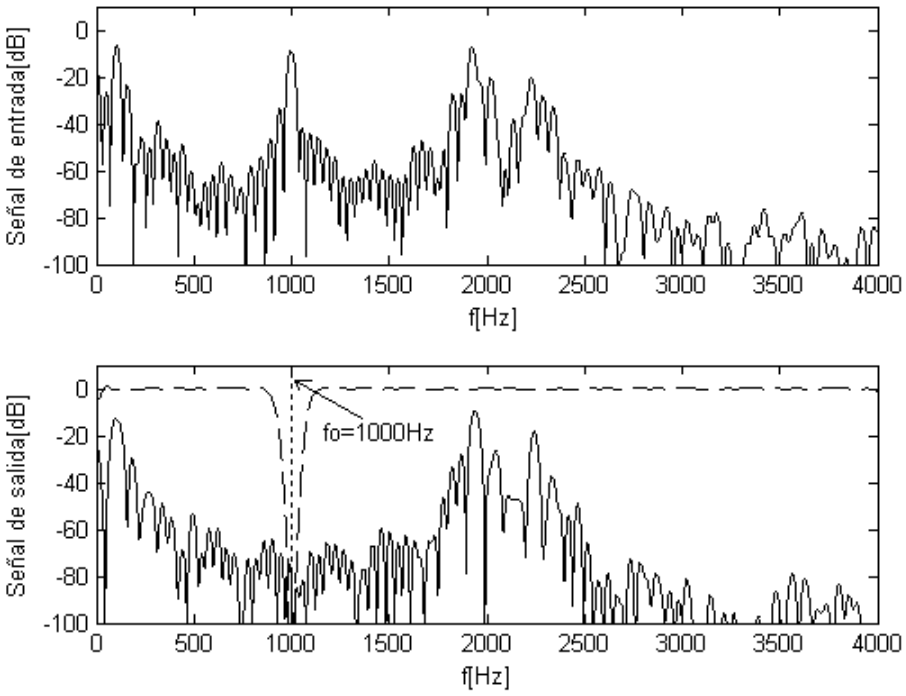


Figura 5.8. Espectro de señales de entrada y salida del filtro supresor de banda

5.3. Formadores de haz

Para evaluar el desempeño de los formadores de haz se construyó una base para los micrófonos que nos permitiera girarlos y así posicionar el ángulo de arribo ϕ de la señal fuente. Se colocó una bocina a cincuenta centímetros del arreglo de micrófonos, se alimentó una señal senoidal de frecuencias armónicas de $220\ Hz$ (frecuencia mínima que alcanzaron a percibir los micrófonos sin perderse en el ruido ambiente) y hasta $3520\ Hz$ (armónico más agudo de la frecuencia tomada como base que es capaz de adquirir la tarjeta de expansión diseñada) y se observó el comportamiento de la señal, variando el ángulo de $-90°$ a $90°$ con incrementos de $15°$. El sistema utilizado para la evaluación de los formadores de haz se ilustra en la figura 5.9.

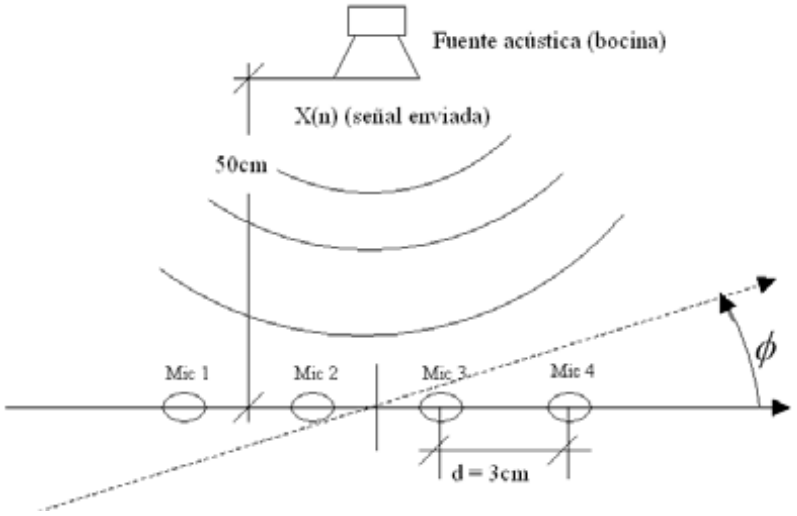


Figura 5.9. Sistema de evaluación de formadores de haz

Las figuras 5.10 a 5.14 corresponden a las gráficas de la energía de la señal total recibida en dB (medida de las gráficas generadas en el DSP para cada ángulo y cada frecuencia) contra el ángulo para las frecuencias $220\ Hz$, $440\ Hz$, $880\ Hz$, $1760\ Hz$ y $3520\ Hz$, del formador de haz fijo de cuatro micrófonos y coeficientes $W = 1$ (ventana cuadrada); además se muestra, con una línea punteada, el comportamiento teórico del formador de haz. En estas figuras se puede apreciar que el formador de haz es más selectivo con las frecuencias altas que con las bajas. La figura 5.15 Muestra el desempeño total del formador de haz fijo.

Las figuras 5.16 a 5.21 son análogas a las seis gráficas descritas en el párrafo anterior, solo que son de un formador de haz adaptable con dos micrófonos utilizando el algoritmo GSC.

En las figuras 5.10 a 5.21 se puede apreciar que el formador de haz es más selectivo con las frecuencias altas, ya que el lóbulo central es más angosto mientras más alta sea la frecuencia; también se puede observar que las gráficas no son simétricas, esto lo atribuimos a que en el laboratorio hay diferentes obstáculos físicos que generan eco, o reverberación y distribuyen asimétricamente la energía de la señal.

En las gráficas 5.10 a 5.14 y 5.16 a 5.20 se muestra, además de la señal obtenida experimentalmente, la señal teórica con línea punteada, esta última se obtuvo en simulando la ecuación (3.17); puede apreciarse que en todos los casos, excepto en la figura 5.20, la magnitud de la señal adquirida normalizada es menor que el de la señal teórica lo que demuestra que los formadores de haz implementados tienen un comportamiento satisfactorio.

$$b(f,\phi) = \sum_{i=1}^{M} \exp(jK(i-1)d\ \text{sen}\phi) \qquad (3.17)$$

Es importante notar que el comportamiento de los dos formadores de haz es muy similar a pesar de que el formador de haz adaptable con algoritmo GSC tiene la mitad de micrófonos, esto comprueba que dicho formador de haz tiene mejor desempeño.

5.4 Recursos del sistema

Un aspecto importante a evaluar en el desempeño de un sistema es el uso de los recursos disponibles de dicho sistema. Para realizar esta evaluación en la tarjeta de expansión diseñada utilizamos el menú ‘Profiler’ de Code Composer Studio y su función ‘clock’; dicha función sirve para evaluar los ciclos de reloj que transcurren entre un punto y otro del código del programa. Después de activar la función clock se colocó el contador de programa a la entrada de la interrupción y se inicializó el valor del contador de reloj en cero; posteriormente se colocó el cursor en la instrucción final del proceso seleccionado con los interruptores del selector y se ejecutó la instrucción ‘run to cursor’ (correr hasta el cursor), de esta forma se obtuvieron los ciclos reloj por proceso, además se evaluaron los tiempos en el osciloscopio, para obtener la frecuencia de reloj del DSP que resultó ser de 10 ns.

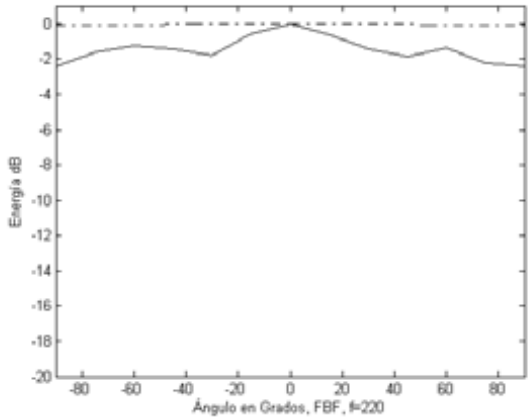


Figura 5.10. Formador de haz fijo f = 220 Hz

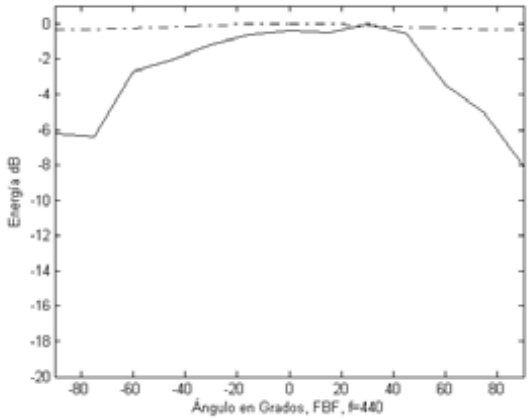


Figura 5.11. Formador de haz fijo f = 440 Hz

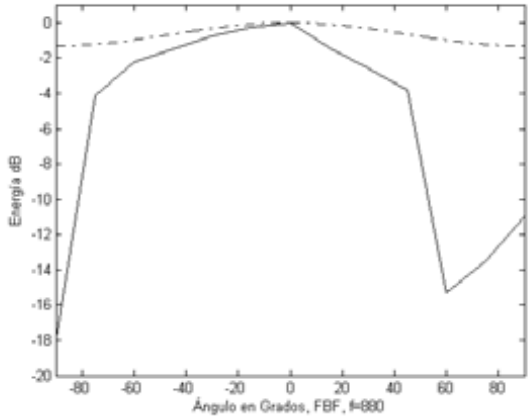


Figura 5.12. Formador de haz fijo f = 880 Hz

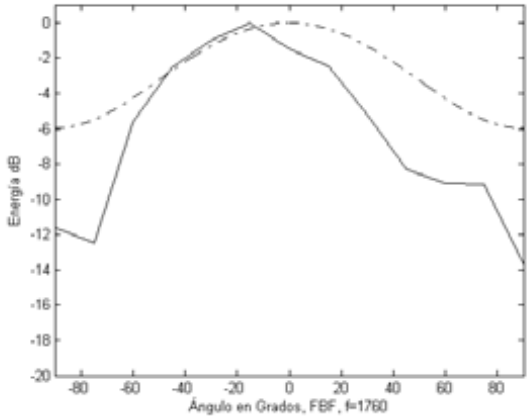


Figura 5.13. Formador de haz fijo f = 1760 Hz

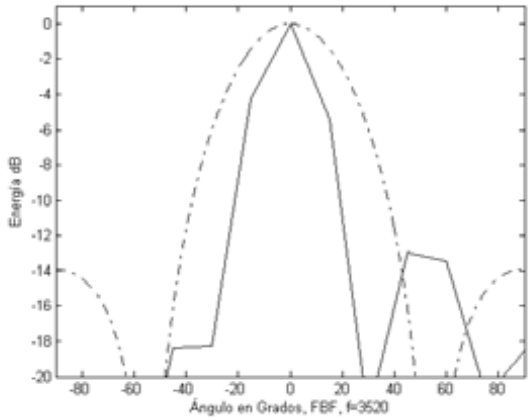


Figura 5.14. Formador de haz fijo f = 3520 Hz

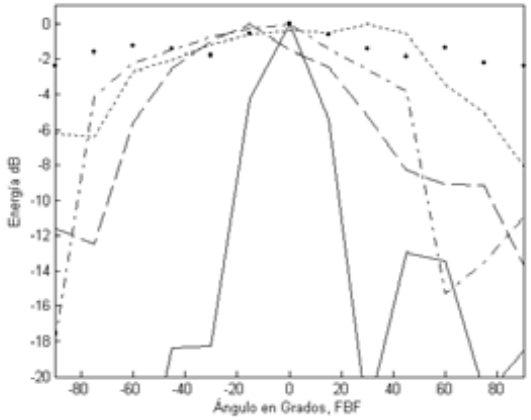


Figura 5.15. Formador de haz fijo

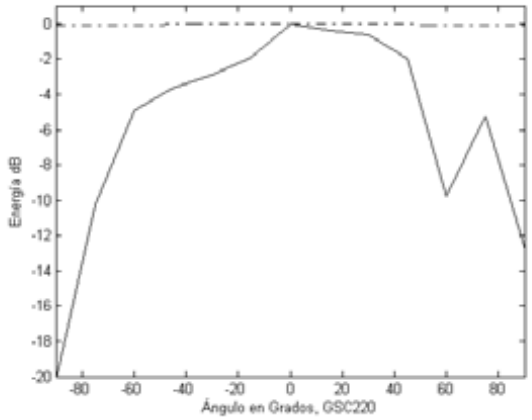


Figura 5.16. Formador de haz adaptable, f =220 Hz

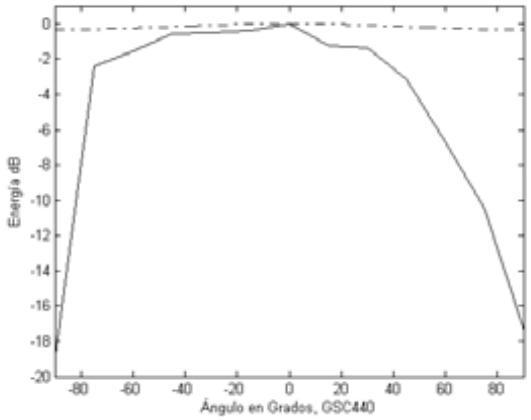


Figura 5.17. Formador de haz adaptable, f =440 Hz

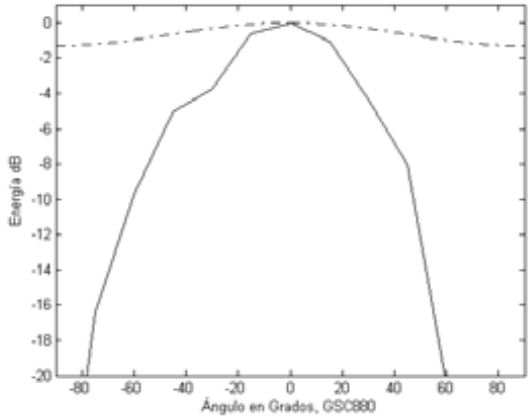


Figura 5.18. Formador de haz adaptable, f =880 Hz

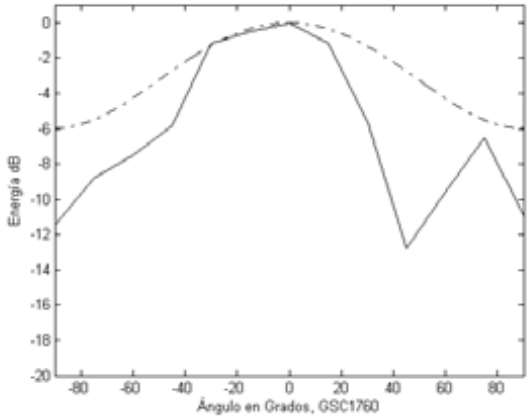


Figura 5.19. Formador de haz adaptable, f =1760 Hz

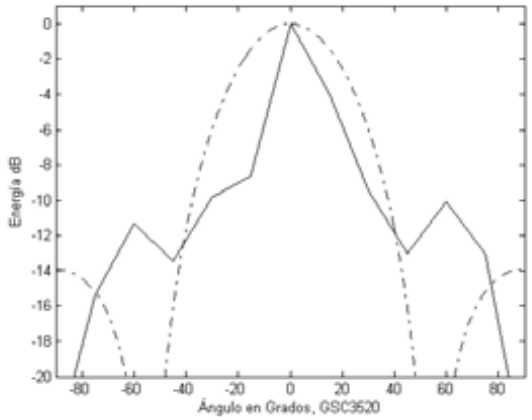


Figura 5.20. Formador de haz adaptable, f =3520 Hz

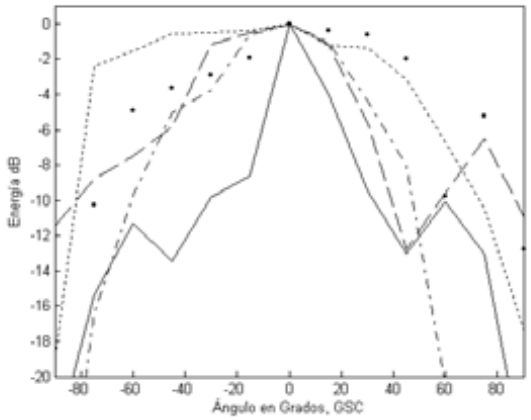


Figura 5.21. Formador de haz adaptable

En la tabla 5.1 se muestra el numero de ciclos de reloj por proceso, el tiempo al que estos ciclos equivalen y el porcentaje respecto de $125\ \mu s$ que tenemos entre muestra y muestra que utiliza cada proceso. Puede apreciarse que la adquisición de las señales de cada micrófonos requiere de un tiempo considerable ($25\ \mu s$) y por esta razón al aumentar el número de micrófonos aumenta el tiempo en forma considerable ($25\ \mu s$ por cada micrófono); también es muy claro que el formador de haz adaptable es un proceso bastante robusto ya que a pesar de que solo se utilizan dos micrófonos el tiempo de este proceso es el mayor, siendo de casi el 90% del tiempo disponible.

Proceso	**Ciclos de Reloj**	**Tiempo de proceso** [μs]	**% de tiempo utilizado**
Lectura y reproducción de cualquier micrófono	2550	25.5	20.4
Grabación 1 micrófono	2584	25.84	20.7
Grabación 2 micrófonos	5145	51.45	41.2
Grabación 3 micrófonos	7714	77.14	61.7
Grabación 4 micrófonos	10269	102.69	82.2
Filtro paso bajas	5272	52.72	42.2
Filtro paso altas	5272	52.72	42.2
Filtro supresor de banda	9625	96.25	77.0
Formador de haz fijo	10211	102.11	81.7
Formador de haz adaptable GSC	11174	111.74	89.4

Tabla 5.1. Procesos y tiempos

El programa queda alojado en las localidades de memoria programa 0x100 a 0x1468 lo que representa 4969 localidades de memoria de las 16256 (de 0x80 a 0x3FFF) que el DSP tiene en memoria programa; es decir, el proyecto completo, con todos los subprogramas ocupa únicamente el 30.6% de la memoria disponible para código con que cuenta el DSP TMS320C5402. Por otro lado de la memoria dato interna del DSP son usadas únicamente 695 que representan el 4.3% de las 16256 localidades disponibles en memoria programa interna del

DSP y 64000 de la memoria SARAM de la tarjeta de la DSK, con capacidad total de 65535, es decir el 97.7% de esta memoria externa.

5.5 Resumen

En este capítulo se presentaron los resultados obtenidos en el DSP de los procesos que puede realizar la tarjeta de expansión; en ellas observamos que dicha tarjeta realiza de forma satisfactoria los procesos que en ella fueron implementados.

Se puede apreciar que las señales de voz guardadas son muy parecidas pero no iguales, tanto por la presencia de un retardo casi inapreciable como por la presencia de diferentes ruidos inherentes al ambiente en los diferentes micrófonos; esto es debido a la posición de los micrófonos y al ángulo de arribo de la señal, pero se puede ver que los cuatro micrófonos están captando la misma señal de voz simultáneamente.

Los tres filtros funcionan muy bien y en las gráficas se puede observar que cada uno cumple con su función, dejan pasar las frecuencias esperadas y anulan las frecuencias no deseadas.

También se puede observar que los dos formadores de haz seleccionan de forma satisfactoria las señales provenientes de ángulos cercanos a 0° y suprimen aquellas provenientes de direcciones diferentes, sin embargo funcionan mejor en las frecuencias altas y el formador de haz adaptable utiliza solamente dos micrófonos, estos formadores se podrían mejorar utilizando más micrófonos o un convertidor analógico-digital más rápido.

Capítulo 6
Conclusiones

Este trabajo se enfoca en el análisis del procesamiento espacial de señales, haciendo énfasis en el proceso llamado formación de haz ya que este proceso fue implementado en la tarjeta de desarrollo.

En el desarrollo de este trabajo de tesis se presentó el proceso para la implementación de una tarjeta de expansión capaz de adquirir hasta cuatro señales de voz de forma simultánea y procesarlas con ayuda del DSP TMS320C5402 y su tarjeta de desarrollo DSK.

Para lograr la implementación es necesario conocer las características de las señales de voz; por esta razón en el capítulo dos se describe de manera general una de las formas de clasificar las señales así como las principales características de la señal de interés, la voz.

La tarjeta de expansión implementada permite adquirir hasta cuatro señales de voz y guardarlas en un archivo para después procesarlas o transportarlas, según la necesidad de la aplicación. Estas señales de voz son guardadas en dos páginas de la memoria SARAM de la tarjeta de desarrollo.

Se implementaron tres tipos diferentes de filtros digitales FIR en punto fijo cuyo desempeño es mostrado en el capítulo cinco.

También se implementaron dos formadores de haz, uno fijo y uno adaptable con algoritmo GSC. El formador de haz fijo tiene una selectividad muy similar a la teórica. El formador de haz adaptable muestra una selectividad mayor a la del formador de haz fijo, a pesar de utilizar en su algoritmo solo la mitad de micrófonos del otro; esto muestra la superioridad de este algoritmo. El desempeño de los formadores de haz puede mejorarse mediante el uso de más micrófonos; sin embargo esta posibilidad no pudo ser explorada debido a que utilizamos los recursos del DSP en más del 80% y el convertidor analógico digital (ADC) es solo de cuatro canales. En el formador de haz adaptable la limitante del número de micrófonos es la velocidad

del DSP, ya que el algoritmo utilizado en este formador es sumamente complejo, como puede apreciarse en la tabla 5.1.

Se comprobó también que los formadores de haz mejoran su selectividad a medida que aumenta la frecuencia de la señal adquirida, como se aprecia en las figuras 5.10 a 5.21, lo cual coincide con los resultados teóricos que pueden verificarse en la referencia [16].

Las aplicaciones desarrolladas e implementadas en este trabajo ocupan una parte pequeña de la capacidad total del DSP ya que el código programado ocupa solamente el 30.6% de la memoria programa disponible y un mínimo, 4.3%, de la memoria dato interna del DSP. La memoria interna del DSP fue poco utilizada porque se prefirió utilizar la memoria SARAM externa del DSP por tener esta última capacidad suficiente para grabar dos segundos de voz por cada uno de los cuatro canales con que cuenta la tarjeta desarrollada.

El trabajo desarrollado puede utilizarse como base para futuras tarjetas de expansión del DSP TMS320C5402 DSK, y puede mejorarse mediante el uso de un ADC más rápido con salida en paralelo o capaz de convertir más micrófonos de forma simultánea e incluso mediante el uso de más de un ADC.

Anexo A

Características generales del DSP TMS320C5402

La familia de DSPs TMS320C54xx (C54xx) de Texas Instruments (TI) está formada por procesadores digitales de señales orientados a aplicaciones incrustadas, en tiempo real en áreas tales como comunicaciones, instrumentación, control y filtrado. Los DSPs están basados en una arquitectura tipo Harvard, con alto grado de paralelismo, bajo consumo de potencia y modos de direccionamiento versátiles.

Su arquitectura, de buses separados para dato y programa, tipo Harvard permite el acceso simultanes a intrucciones y datos, proveyendo un alto grado de paralelismo, estos DSPs pueden ejecutar tres lecturas y una escritura en un ciclo simple. A continuación se describen brevemente las características principales de los DSPs de esta familia.

A.1 Buses

La arquitectura del DSP está construida sobre los ocho buses de 16 bits (figura A.1), descritos a continuación:

- Los buses PAB, CAB, DAB y EAB, son buses de dirección.
- El bus de programa (PB): transporta el código de instrucción y los operandos inmediatos de memoria programa. También está conectado al multiplicador como una entrada de memoria programa.
- Buses de datos: Interconectan varios elementos como el CPU, generador de direcciones de dato y programa, periféricos y memoria dato.
    - Los buses CB y DB transportan los operandos que son leídos de memoria dato. Estos buses hacen posible la búsqueda simultánea de dos operandos, o un operando doble, en un mismo ciclo de instrucción.

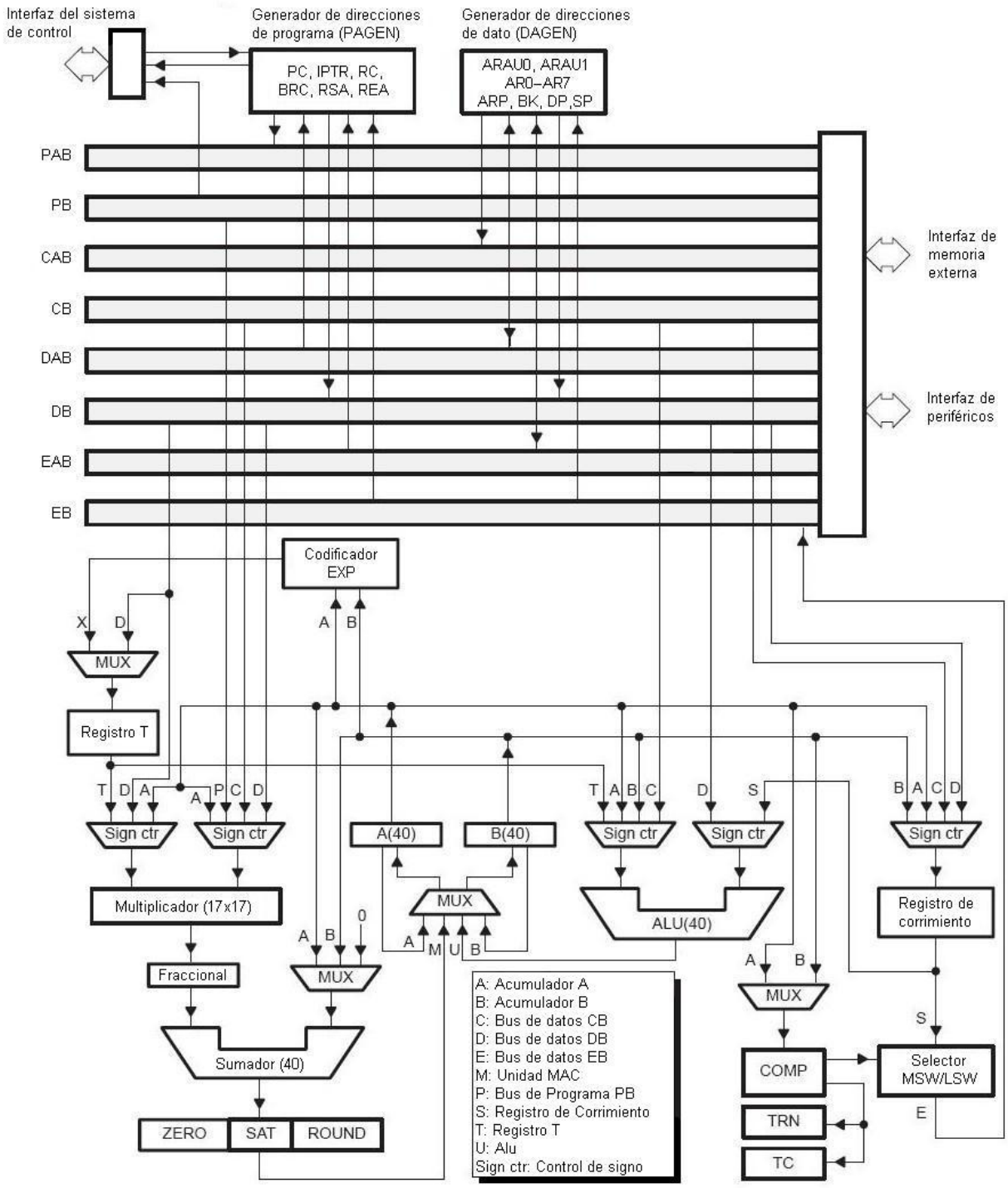


Figura A.1.Arquitectura de DSP TMS320C54xx

- El bus EB transporta los datos para ser escritos en memoria, permitiendo lecturas y escrituras para instrucciones en paralelo.

A.2 Organización de la memoria

LA memoria del C54xx está organizada en tres espacios: 64k palabras para programa, 64k palabras para datos y 64k palabras para puertos I/O; se puede elegir entre estos espacios de memoria por medio de las señales /PS, /DS e /IS en los pines externos correspondientes. La memoria puede ser de tipo RAM o ROM; la memoria RAM puede ser SARAM o DARAM. La figura A.2 muestra la distribución del mapa de memoria para el DSP TMS320C5402.

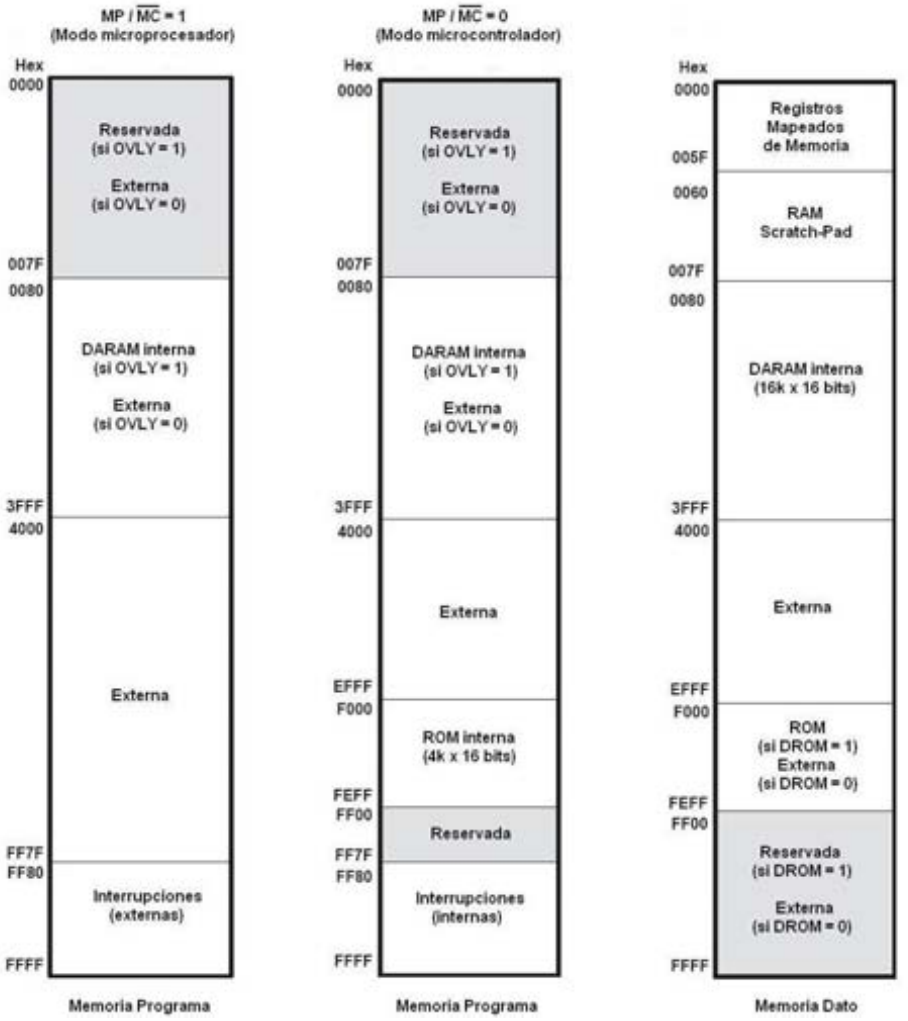


Figura A.2. Mapa de memoria del DSP C5402

A.3 Unidad central de proceso

Está encargada de realizar las operaciones aritméticas y lógicas sobre los datos y está compuesta por las siguientes partes:

- Una Unidad Aritmética lógica de 40 bits (ALU)
- Dos Acumuladores (A y B) de 40 bits, estos almacenan la salida de la ALU o del bloque multiplicador / sumador. Cada acumulador está dividido en tres partes:
  - Guarda (bits 39 a 32, AG o BG).
  - Parte alta (32 a 16, AH o BH).
  - Parte baja (15 a 0, AL o BL).
- Un registro para corrimientos de 0 a 31 bits a la izquierda (+) o de 0 a 16 bits a la derecha (-). Este registro, junto con el codificador de exponente, normalizan a cada acumulador en un ciclo de instrucción.
- Un multiplicador de 17x17 bits en complemento a dos.
- Un sumador de 40 bits.
- Una unidad de comparación, selección y almacenamiento (CSSU).
- Una unidad generadora de dirección de datos.
- Una unidad generadora de dirección de programa.
- Codificador de exponente, para calcular el exponente de un acumulador de 40 bits en un ciclo de instrucción.
- Registros de estado y control.

A.4 Otras Características

Este DSP cuenta también con las instrucciones especiales para procesamiento digital de señales:

- Para repetición de una instrucción y repetición de bloques de instrucciones.
- Para movimientos de bloques de datos y programa.

- Para operaciones con operandos de 32 bits.
- Para lectura de hasta tres operandos simultáneos.
- Para operaciones aritméticas, con almacenamiento y carga paralela.
- Para almacenamiento condicional.

También cuenta con periféricos internos para:

- Estados de espera programables por software.
- Generador de malla de fase amarrada (PLL).
- Control externo de bus para deshabilitar buses externos.
- Bus de datos con posibilidad de retención (Holder).
- Temporizador programable.
- Tres puertos serie

Además cuenta con ocho registros auxiliares mapeados (AR0 a AR7) para direccionamiento indirecto, opera en modo de bajo consumo y maneja seis niveles de pipeline: prebúsqueda, búsqueda, decodificación, acceso, lectura y ejecución.

Anexo B

Programas

En este anexo se muestran los programas utilizados para el funcionamiento de este proyecto, estos programas contienen comentarios que pueden ayudar a comprender mejor los procesos.

```
/* ********************** Programa principal ***************************** */

//include files directory: C:\ti\c5400\cgtools\include
//include drv5402.lib, dsk5402.lib,rts.lib
//
#include <type.h>
#include <board.h>
#include <codec.h>

#include <coef.h>                    /*Archivo que contiene los coeficientes de los filtros*/


#define GLOBAL_INT_ENABLE    asm( " rsbx intm ")
#define GLOBAL_INT_DISABLE   asm( " ssbx intm ")

volatile short *imr = (short *) 0x00;                    /*Definicion de apuntadores*/
volatile short *ifr = (short *) 0x01;
volatile short *pmst= (short *) 0x1d;
volatile short *xpc= (short *) 0x1e;
volatile short *drr = (short *) 0x41;
volatile short *dxr = (short *) 0x43;                    /*Envio de datos*/
volatile int *datini = (int *) 0x4004;                   /*Direccion de inicio para guardar*/
volatile int *datos = (int *) 0x4004;
volatile int *datmid = (int *) 0x7E84;
volatile int *datfin = (int *) 0xBD04;                   /*Direccion de fin para guardar*/
volatile int *datos2 = (int *) 0x7E85;
short data,data0,data1;                                  /*Definicion de variables*/
short data2,data3,data4,data5;                           /*Variables para lectura de microfonos*/
long filt;                                               /*Variable para filtros*/
short sel1,sel2,sel3,sel4;                               /*Variables para selector*/
int lect;
short salida;                                            /*Variable para salida*/
int k;
int i=1;                                                 /*Variables para contadores*/
int j=0;
int l;
```

```
#include <GSC.h>            /*Archivo que contiene la rutina del formador de haz adaptable*/
#include <lectura.h>        /*Archivo con las rutinas para recibir las señales de voz*/
short buffer1 [14000];
short buffer2 [14000];
short buffer3 [14000];
short buffer4 [14700];

HANDLE hHandset;

ioport unsigned port8000;
ioport unsigned port8002;

void espera(void);                              /*Subrutinas utilizadas*/
void ent1(void);
void ent2(void);
void ent3(void);
void ent4(void);
void ent1fil(void);

short GSC2(void);
/*******************Subrutina de atención de interrupciones ********************/
/*********************donde se realizan todos los procesos ******************/
interrupt void mandar(void)                     /*Interrupcion de envio de datos*/
{
        *dxr=salida;                            /*Envio de datos*/
        if ( sel1!=0 && sel2==0 && sel3==0 && sel4==0){          /*Proceso 1*/
                ent1();                         /*Leer y reproducir*/
                salida = data1;                 /*microfono 1*/
                }
        else if ( sel1==0 && sel2!=0 && sel3==0 && sel4==0){    /*Proceso 2*/
                ent2();                         /*Leer y reproducir*/
                salida = data2;                 /*microfono 2*/
                }
        else if ( sel1==0 && sel2==0 && sel3!=0 && sel4==0){    /*Proceso 3*/
                ent3();                         /*Leer y reproducir*/
                salida = data3;                 /*microfono 3*/
                }
        else if ( sel1==0 && sel2==0 && sel3==0 && sel4!=0){    /*Proceso 4*/
                ent4();                         /*Leer y reproducir*/
                salida = data4;                 /*microfono 4*/
                }
        else if ( sel1!=0 && sel2!=0 && sel3!=0 && sel4!=0){    /*Proceso 5*/
                datos=datini;                   /*Inicializar los apuntadores*/
                datos2=datmid;
                }
        else if ( sel1==0 && sel2!=0 && sel3!=0 && sel4!=0){    /*Proceso 6*/
                ent1();                         /*Lectura del microfono 1*/
                salida = data1;
```

```
        port4 = 0x00;                       /*Seleccion de memoria SRAM*/
        port2 = 0x00;                       /*Seleccion de pagina 0 de memoria*/
        *datos=data1;                       /*Guarda en la direccion del apuntador*/
        datos=datos+1;                      /*Aumenta el apuntador*/
        if (datos>=datfin){
        datos=datfin;
        }

        port2 = 0x00;                       /*Seleccion de pagina 0 de memoria*/
        port4 = 0x20;                       /*Seleccion de memoria externa*/
        }
else if ( sel1!=0 && sel2==0 && sel3!=0 && sel4!=0){        /*Proceso 7*/
        ent1();                             /*Lectura del microfono 1*/
        ent2();                             /*Lectura del microfono 2*/
        salida = data1+data2;
        port4 = 0x00;                       /*Seleccion de memoria SRAM*/
        port2 = 0x00;                       /*Seleccion de pagina 0 de memoria*/
        *datos=data1;                       /*Guarda en la direccion del apuntador*/
        port2 = 0x01;                       /*Seleccion de pagina 1 de memoria*/
        *datos=data2;                       /*Guarda en la direccion del apuntador*/
        datos=datos+1;                      /*Aumenta el apuntador*/
        if (datos>=datfin){
        datos=datfin;
        }

        port2 = 0x00;                       /*Seleccion de pagina 0 de memoria*/
        port4 = 0x20;                       /*Seleccion de memoria externa*/
        }
else if ( sel1!=0 && sel2!=0 && sel3==0 && sel4!=0){        /*Proceso 8*/
        ent1();                             /*Lectura del microfono 1*/
        ent2();                             /*Lectura del microfono 2*/
        ent3();                             /*Lectura del microfono 3*/
        salida = data1+data2+data3;
        port4 = 0x00;                       /*Seleccion de memoria SRAM*/
        port2 = 0x00;                       /*Seleccion de pagina 0 de memoria*/
        *datos=data1;                       /*Guarda en la direccion del apuntador*/
        *datos2=data2;                      /*Guarda en la direccion del apuntador*/
        port2 = 0x01;                       /*Seleccion de pagina 1 de memoria*/
        *datos=data3;                       /*Guarda en la direccion del apuntador*/
        datos=datos+1;                      /*Aumenta el apuntador*/
        if (datos>=datmid){
        datos=datmid;
        }
        datos2=datos2+1;                    /*Aumenta el apuntador*/
        if (datos2>=datfin){
        datos2=datfin;
        }
```

```
        port2 = 0x00;                           /*Seleccion de pagina 0 de memoria*/
        port4 = 0x20;                           /*Seleccion de memoria externa*/
        }
else if ( sel1!=0 && sel2!=0 && sel3!=0 && sel4==0){        /*Proceso 9*/
        ent1();                                 /*Lectura del microfono 1*/
        ent2();                                 /*Lectura del microfono 2*/
        ent3();                                 /*Lectura del microfono 3*/
        ent4();                                 /*Lectura del microfono 4*/
        salida = data1+data2+data3+data4;
        port4 = 0x00;                           /*Seleccion de memoria SRAM*/
        port2 = 0x00;                           /*Seleccion de pagina 0 de memoria*/
        *datos=data1;                           /*Guarda en la direccion del apuntador*/
        *datos2=data2;                          /*Guarda en la direccion del apuntador*/
        port2 = 0x01;                           /*Seleccion de pagina 1 de memoria*/
        *datos=data3;                           /*Guarda en la direccion del apuntador*/
        *datos2=data4;                          /*Guarda en la direccion del apuntador*/
        datos=datos+1;                          /*Aumenta el apuntador*/
        if (datos>=datmid){
        datos=datmid;
        }
        datos2=datos2+1;                        /*Aumenta el apuntador*/
        if (datos2>=datfin){
        datos2=datfin;          }

        port2 = 0x00;                           /*Seleccion de pagina 0 de memoria*/
        port4 = 0x20;                           /*Seleccion de memoria externa*/
        }
else if ( sel1!=0 && sel2!=0 && sel3==0 && sel4==0){        /*Proceso 10*/
        //HP                                    /*Filtro paso altas*/
        ent1fil();                              /*Lectura del microfono 1*/
        filt=0;
        for (k=102;k>0;k--){
         buffer1[k]=buffer1[k-1];               /*Corrimiento de datos*/
        }
        buffer1[0]=data1;                       /*Dato actual*/
        for (l=0;l<101;l++){                    /*Orden del filtro N=100*/
                filt=filt+buffer1[l]*HPF[l];    /*Multiplicación y suma*/
                }
        k=0;

        salida = (short)(filt>>9);              /*Corrimiento para mantener el Q*/
        }
else if ( sel1==0 && sel2==0 && sel3!=0 && sel4!=0){        /*Proceso 11*/
        //LP                                    /*Filtro paso bajas*/
        ent1fil();                              /*Lectura del microfono 1*/
        filt=0;
        for (k=102;k>0;k--){
         buffer1[k]=buffer1[k-1];               /*Corrimiento de datos*/
        }
```

```
            buffer1[0]=data1;                   /*Dato actual*/
            for (l=0;l<101;l++){                /*Orden del filtro N=100*/
                  filt=filt+buffer1[l]*LPF[l];  /*Multiplicación y suma*/
                  }
            k=0;

            salida = (short)(filt>>9);          /*Corrimiento para mantener el Q*/
            }
      else if ( sel1==0 && sel2!=0 && sel3!=0 && sel4==0){    /*Proceso 12*/
            //BS                                /*Filtro supresor de banda*/
            ent1fil();                          /*Lectura del microfono 1*/
            filt=0;
            for (k=202;k>0;k--){
             buffer1[k]=buffer1[k-1];           /*Corrimiento de datos*/
            }
            buffer1[0]=data1;                   /*Dato actual*/
            for (l=0;l<201;l++){                /*Orden del filtro N=200*/
                  filt=filt+buffer1[l]*BSF[l];  /*Multiplicación y suma*/
                  }
            k=0;

            salida = (short)(filt>>9);          /*Corrimiento para mantener el Q*/
            }
      else if ( sel1!=0 && sel2==0 && sel3==0 && sel4!=0){    /*Proceso 15*/
            //GSC 2ch                           /*Formador de haz adaptable*/
            ent1();                             /*Lectura del microfono 1*/
            ent2();                             /*Lectura del microfono 2*/
            salida = GSC2();                    /*Ir a rutina de formacion de haz*/
            }
      else{                                                 /*Proceso 14*/
            ent1();                             /*Lectura del microfono 1*/
            ent2();                             /*Lectura del microfono 2*/
            ent3();                             /*Lectura del microfono 3*/
            ent4();                             /*Lectura del microfono 4*/

      salida = (data1>>1)+(data2>>1)+(data3>>1)+(data4>>1);  /*Formador de haz fijo*/
            }

}


/*******************Programa Principal ***********************************/

void main(void)                                 /*Inicio de programa principal*/
 {



      GLOBAL_INT_DISABLE;                       /*Deshabilitar interrupciones*/
```

```
   brd_init_bios();

   hHandset = codec_open(HANDSET_CODEC);        /*inicialización del codec de envio*/

   codec_dac_mode(hHandset, CODEC_DAC_15BIT);
   codec_adc_mode(hHandset, CODEC_ADC_15BIT);
   codec_ain_gain(hHandset, CODEC_AIN_6dB);
   codec_aout_gain(hHandset, CODEC_AOUT_MINUS_0dB);
   codec_sample_rate(hHandset,SR_8000);         /*Frecuencia de muestreo 8000 Hz*/

   *pmst = 0x00A0;
   *imr |= 0x0800;                              /*Habilitación de interrupción de envio*/
   *ifr = *ifr;
   port4 = 0x20;                                /*Seleccion de memoria externa*/


   GLOBAL_INT_ENABLE;                           /*Habilitar interrupciones*/


  while (1) {                                   /*Ciclo principal de espera */
                                                /*de interrupción */

        data=port8002;                          /*El programa se mantiene checando*/
        sel1=data&0x100;                        /*los interruptores mientras espera*/
        sel2=data&0x200;                        /*la interrupcion*/
        sel3=data&0x400;
        sel4=data&0x800;

        }       // while
}  // main



/* ******************** Programa para lectura de micrófonos ********************** */
/* ****************************** lectura.h ************************************ */

int lect;                               /* Variables para guardar datos */
short data,data0,data1;
short data2,data3,data4;
ioport unsigned port8000;               /* Variables para control de puertos */
ioport unsigned port8002;
ioport unsigned port8006;
ioport unsigned port8008;
ioport unsigned port800A;
ioport unsigned port800E;
ioport unsigned port8010;
ioport unsigned port8012;
ioport unsigned port8016;
```

```
ioport unsigned port8018;
ioport unsigned port801A;
ioport unsigned port801E;

void espera (void)                  /* Ciclo de espera */
{
        int j;
        for (j=0;j<110;j++)
         {
                j=j;
         }
        return;
}

void ent1(void)                     /* Lectura de micrófono 1 */
{
        lect=port8000;              /* Manda 0 al inicio de conversión del convertidor */
        espera();
        lect=port8002;              /* Manda 1 al inicio de conversión del convertidor */
        data=port8002&0xFF;         /* Lee el byte más significativo del canal 1 */
        data0=port8006&0xFF;        /* Lee el byte menos significativo del canal 1 */
        data1=data<<8;              /* Guarda el byte más significativo */
        data1 |= data0;             /* Guarda el byte menos significativo */
        data1 = data1-8224;         /* Resta el offset */
        espera();
        return;                     /* Regresa de la subrutina */
}

void ent2(void)
{
        lect=port8008;              /* Manda 0 al inicio de conversión del convertidor */
        espera();
        lect=port800A;              /* Manda 1 al inicio de conversión del convertidor */
        data=port800A&0xFF;         /* Lee el byte más significativo del canal 1 */
        data0=port800E&0xFF;        /* Lee el byte menos significativo del canal 1 */
        data2=data<<8;              /* Guarda el byte más significativo */
        data2 |= data0;             /* Guarda el byte menos significativo */
        data2 = data2-8224;         /* Resta el offset */
        espera();
        return;                     /* Regresa de la subrutina */
}

void ent3(void)
{
        lect=port8010;              /* Manda 0 al inicio de conversión del convertidor */
        espera();
        lect=port8012;              /* Manda 1 al inicio de conversión del convertidor */
        data=port8012&0xFF;         /* Lee el byte más significativo del canal 1 */
        data0=port8016&0xFF;        /* Lee el byte menos significativo del canal 1 */
```

```
        data3=data<<8;          /* Guarda el byte más significativo */
        data3 |= data0;         /* Guarda el byte menos significativo */
        data3 = data3-8224;     /* Resta el offset */
        espera();
        return;                 /* Regresa de la subrutina */
}

void ent4(void)
{
        lect=port8018;          /* Manda 0 al inicio de conversión del convertidor */
        espera();
        lect=port801A;          /* Manda 1 al inicio de conversión del convertidor */
        data=port801A&0xFF;     /* Lee el byte más significativo del canal 1 */
        data0=port801E&0xFF;    /* Lee el byte menos significativo del canal 1 */
        data4=data<<8;          /* Guarda el byte más significativo */
        data4 |= data0;         /* Guarda el byte menos significativo */
        data4 = data4-8224;     /* Resta el offset */
        espera();
        return;                 /* Regresa de la subrutina */
}

void ent1fil(void)
{
        lect=port8000;          /* Manda 0 al inicio de conversión del convertidor */
        espera();
        lect=port8002;          /* Manda 1 al inicio de conversión del convertidor */
        data=port8002&0xFF;     /* Lee el byte más significativo del canal 1 */
        data0=port8006&0xFF;    /* Lee el byte menos significativo del canal 1 */
        data1=data<<7;          /* Guarda el byte más significativo */
        data1 |= data0>>1;      /* Guarda el byte menos significativo */
        data1 = data1-4100;     /* Resta el offset */
        return;                 /* Regresa de la subrutina */
}
```

```
/******************** Programa para formador de haz adaptable ******************** */
/* ********************************** GSC.h ************************************ */

int N = 40;
int i;
int k=0;
short mu = 5000;
short ze = 0;                                   /* salida, error */
short yest;                                     /* 'y' estimado */

short FBF[41] = {0,0,0,0,0,0,0,0,0,0,
                 0,0,0,0,0,0,0,0,0,0,
                 0,0,0,0,0,0,0,0,0,0,
```

```
        0,0,0,0,0,0,0,0,0,0,0};                 /* FBF = x1 + x2 */

short U1 [41] = {0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0};                 /* U1 = x1 - x2 */
short W1ant[40] = {0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0};                   /* W1 anterior */
long W1act[40] = {0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0};                   /* Coeficientes adaptables */

long y1;

short buffer1 [14000];
short GSC2(void)
{
   FBF[0] = data1+data2;                        /* Cálculo de FBF */
        U1[0] = data1-data2;                    /* Cálculo de U1 */
   y1 = 0;
   for(i=0;i<=N;i++)
                {
                W1act[i]=W1ant[i]+(mu*ze*U1[i]);/* Cálculo de coeficientes */
                W1act[i]=W1act[i]>>14;          /* Corrimiento para mantener el Q */
                W1ant[i]=W1act[i];              /* Copiar Wactual a Wanterior */
                y1 = y1+U1[i]*W1act[i];         /* Multiplicación y suma */
                }
        yest = y1>>14;                          /* Corrimiento para mantener el Q */
        ze = FBF[0]-yest;                       /* Cálculo de la salida */
        for(i=N;i>=1;i--)
                {
                FBF[i]=FBF[i-1];                /* Corrimiento de datos */
                U1[i]=U1[i-1];
                }
        buffer1[k]=ze;                          /* Guarda la salida en buffer1 */
        k++;
        if (k>=2000){
                k=0;
                }
        return(ze);                             /* Regresa de la subrutina con la salida */
}
```

Anexo C

Diseño de filtros digitales con Matlab®

Este anexo describe a grandes rasgos la aplicación de Matlab® 'fdatool' utilizada en este trabajo para el diseño de los filtros digitales mostrados en la sección 4.4.2.

La aplicación 'fdatool' puede ser invocada desde línea de comandos escribiendo '>>fdatool'; una vez hecho esto Matlab® desplegará la ventana que se ilustra en la figura C.1.

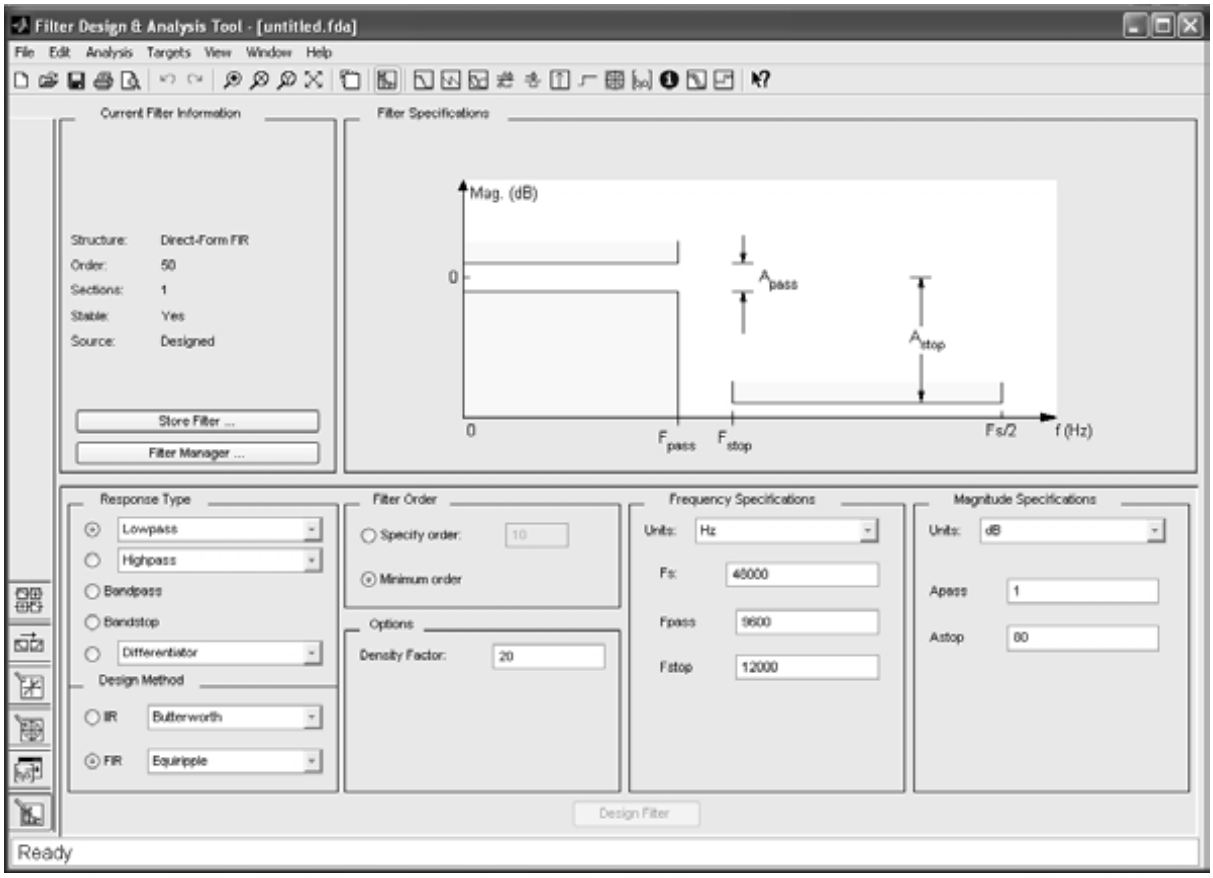


Figura C.1. Ventana de fdatool

En la parte superior izquierda de la ventana desplegada por Matlab® titulada 'Current Filter Information' se encuentra la información del filtro cuyas características se ilustran en la parte superior derecha de la pantalla titulada 'Filter Specifications'.

En la parte inferior izquierda se encuentran las secciones ‘Response Type’ en la que se encuentran las opciones de tipo de respuesta para el filtro que diseñemos, éntre estas se encuentran: paso bajas (con diferentes opciones de diseño), paso altas (con diferentes opciones de diseño), paso banda, supresor de banda y diferenciador (con diferentes opciones de diseño); y ‘Design Method’ en la que podemos seleccionar entre filtros FIR e IIR, y para cada uno de estos tipos contamos con un submenú en el que se pueden modificar otras especificaciones. En la parte inferior izquierda central encontramos la sección ‘Filter Order’ en la que se puede escoger entre el diseño de un filtro con las características especificadas de orden mínimo y el dar el orden del filtro, debajo de ‘Filter Order’ se encuentra la sección ‘Options’ en la que se pueden especificar opciones de diseño y que cambia de acuerdo a los parámetros especificados en ‘Design Method’. En la parte inferior derecha central encontramos las especificaciones de frecuencia ‘Frequency Specifications’ en la que encontramos una sección para seleccionar las unidades de la información requerida inmediatamente abajo, y que pueden estar en unidades de frecuencia o normalizadas, después se encuentran espacios para la frecuencia de muestreo y otras frecuencias de interés que variarán dependiendo del tipo de filtro. Finalmente en la parte inferior derecha se encuentra una la sección en la que encontramos las especificaciones de magnitud titulada ‘Magnitud Specifications’, en ella se debe colocar información referente a el filtro que queremos diseñar o se desplegará información sobre el filtro diseñado dependiendo de qué opciones se hayan seleccionado previamente. Una vez seleccionadas las características del filtro que se quiere diseñar, basta hacer clic en el botón que se encuentra en la parte inferior central de la ventana, que dice ‘Design Filter’, para que realice el proceso de diseño con las especificaciones dadas.

En la figura C.2 se muestra la apariencia de la ventana fdatool después de indicarle a la aplicación que diseñe un filtro supresor de bada (está seleccionada la opción ‘Bandstop’ de la sección ‘Response Type’), de respuesta finita al impulso con ventana (seleccionando FIR y en las opciones de este tipo de filtro escogiendo ’Window’), de orden 100 (en la sección ‘Order’) con ventana de Hamming (en la sección ‘Options’) considerando una frecuencia de muestreo de $8000\ Hz$ y frecuencias de corte de $900\ Hz$ y $1100\ Hz$ (especificadas en la sección ‘Frequency Specifications’)

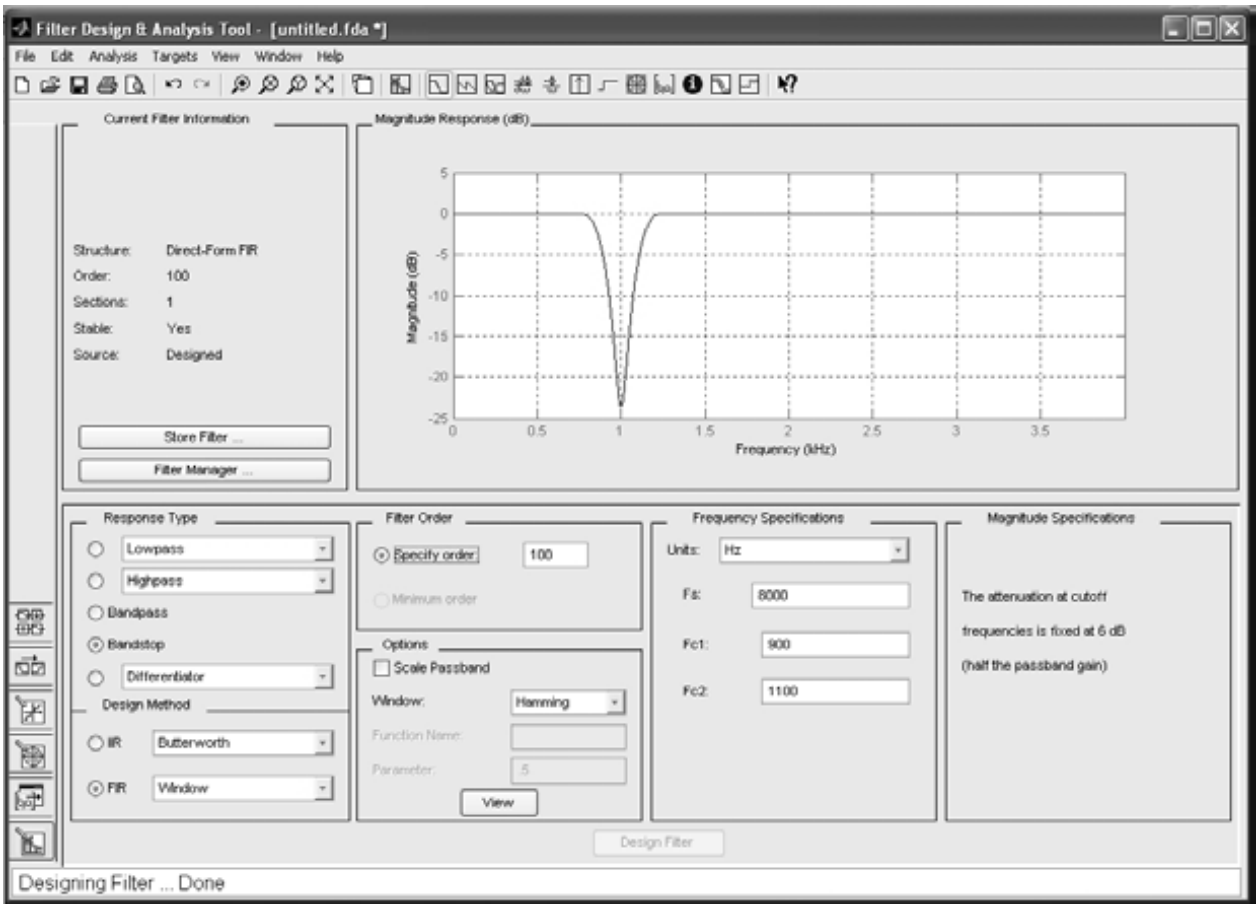


Figura C.2. Filtro paso banda

Una vez diseñado el filtro sus coeficientes pueden ser exportados mediante la opción ‘Export’ contenida en el menú ‘File’ de la ventana de fdatool; al hacer clic en ‘Export’ se nos muestra la ventana mostrada en la figura C.3, en la que se pueden seleccionar diferentes opciones para la exportación de la información. Al seleccionar ‘Workspace’ en la sección ‘Export To’, ‘Coefficients’ en ‘Export As’, dar el nombre de la variable (o las variables si el filtro es IIR) en ‘Variable Names’ y hacer clic en el botón OK, se generará una variable (o dos si el filtro es IIR) que contendrá los coeficientes del filtro diseñado en el ambiente principal de Matlab®.

Figura C.3. Ventana para exportar datos

Anexo D
El circuito impreso

Para la implementación del circuito impreso lo primero que tuvimos que definir fue qué programa utilizaríamos, ya que existen en el mercado una gran variedad de ellos, siendo el más común Eagle®. Eagle limita el espacio de los circuitos integrados, en su versión libre, a un espacio de 4x3 pulgadas, en el que resultaba imposible distribuir los tres circuitos integrados que utilizaríamos sin ocasionar conflicto entre las pistas; por ello fue necesario buscar programas alternativos.

Después de una búsqueda se encontró el editor de circuitos impresos ExpressPCB®, que no tiene costo ni restricciones en cuanto a dimensiones del circuito, este es el programa que decidimos utilizar.

ExpressPCB

El ambiente del programa ExpressSCH se muestra en la Figura D.1, donde se pueden observar un menú en la parte superior, una fila de botones inmediatamente debajo de dicho menú, una columna de botones en el extremo izquierdo del ambiente, la barra de estado en la fila extrema inferior, el ambiente en sí representado en negro y un rectángulo (de líneas amarillas), que es el límite del espacio de trabajo.

El menú de la aplicación tiene las pestañas típicas de un menú de Windows® archivo, edición, vista y ayuda; además de otras dos llamadas Component y Layout. En Component encontramos algunas acciones que se pueden realizar sobre los componentes como rotar, agrupar y guardar propiedades; en Layout encontramos acciones referentes a la compra del circuito que hemos diseñado.

La fila de botones de la parte superior, está formada por lo botones abrir, guardar y deshacer, de Windows seguidos por cuatro botones destinados a modificar la forma en que se

despliega el proyecto; a la derecha de ellos hay dos botones, el primero es para mostrar y modificar las propiedades del proyecto (entre ellas la malla o grid) y el segundo para mostrar y modificar las propiedades de los objetos; inmediatamente a la derecha se encuentran dos botones que sirven para cambiar un objeto de la capa superior a la inferior de cobre o viceversa, la capa destino es la que se encuentra hacia donde apuntan las flechas y en el proyecto el objeto cambiará de color siendo el rojo para la capa superior y el verde para la inferior. Los últimos dos botones a la derecha sirven para girar un objeto.

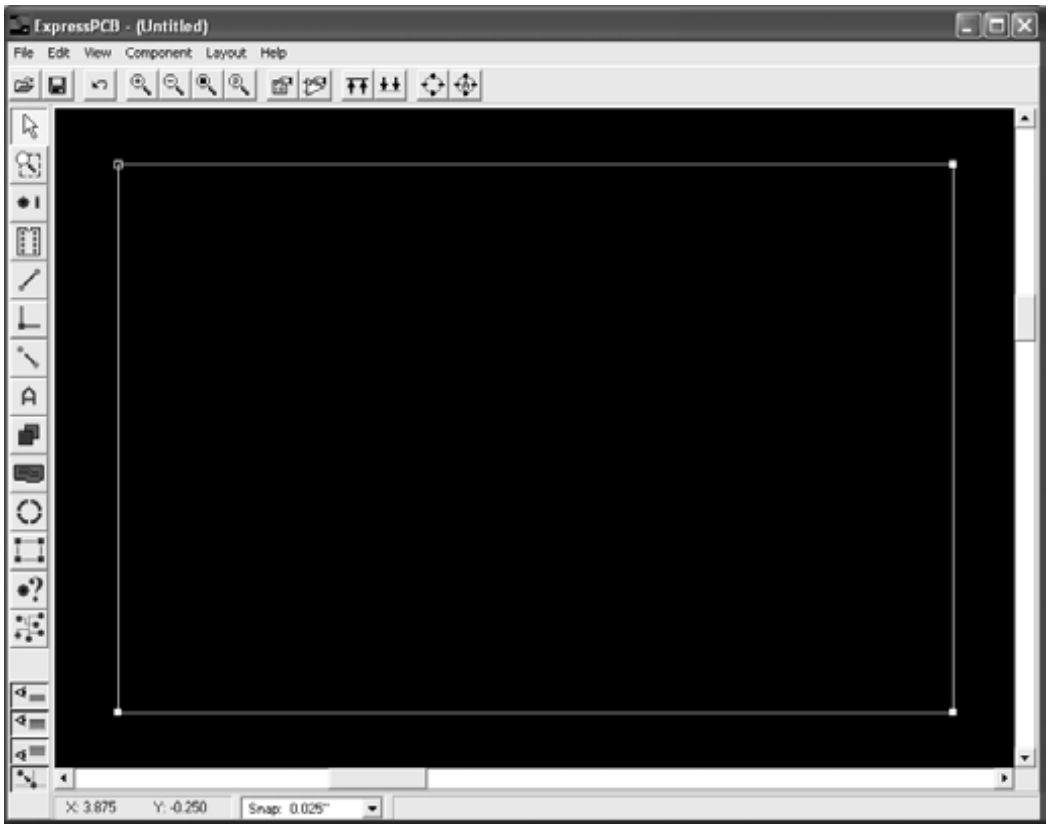


Figura D.1. Ambiente de Express PCB

Los botones de la columna son, de arriba hacia abajo, para: activar la función del mouse selección por objeto, activar la opción del mouse selección por zona, poner un punto de perforación con taladro, insertar un componente, dibujar una línea, dividir una línea (agregarle una esquina), desconectar una línea, insertar texto, insertar un rectángulo, insertar un plano relleno, insertar un círculo o arco de círculo, dividir una línea del límite del espacio de trabajo,

activar la función despliegue de información del mouse y para activar las conexiones múltiples de una red.

Separados de estos botones, en el extremo inferior de la columna de botones encontramos cuatro botones, los primeros tres de ellos modifican a los descritos en el párrafo anterior, haciendo que el objeto insertado (línea, componente, superficie, etcétera), quede como referencia (amarillo) en la capa inferior (verde) o en la capa superior (rojo); el botón restante, el de hasta abajo, sirve para activar la función que hace al mouse saltar de punto en punto de la malla (grid) o le permite moverse con completa libertad.

En la barra de estado encontramos la posición del mouse en coordenadas cartesianas, señalado con una X y una Y, u un cuadro en el que se puede seleccionar a que distancia se encontraran los objetos copiados del centro de la pantalla.

El ambiente es el espacio en el que se pueden agregar objetos, sin embargo al mandar a imprimir el proyecto, solo se imprimirán aquellos objetos que se encuentren dentro del rectángulo amarillo, límite del área de trabajo. Este espacio puede ser aumentado o disminuido de acuerdo a las necesidades del diseño simplemente arrastrando las líneas que limitan el rectángulo.

Desarrollo del circuito impreso

Ya conociendo el modo de operar el programa ExpressPCB se procedió a la distribución de los componentes descritos en la sección 4.2 de este trabajo de forma que quedaran en una sola capa. La distribución de los componentes se muestra en la figura D.2, en esta figura se señala solamente la posición de los tres circuitos integrados (un ADS7824 y dos TLC272) y de los dos conectores Samtec (Memory Int y Peripheral INT), ya que el incluir a los demás componentes (resistencias y capacitores) haría inentendible a la figura. Las terminales del conector Samtec que se conectaron con cables, ya sea a el ADS7824 o al selector (no ilustrado en esta figura) están señaladas con un pequeño cuadro negro. La dimensión de esté diseño es de $120\text{x}120\ mm$.

Este diseño fue mandado a imprimir mediante el menú archivo (file) y la opción imprimir (print) del ambiente y se seleccionó únicamente la capa inferior de cobre (bottom Copper layer), en la portada de la gaceta de la Facultad de Ingeniería, con una impresora láser. Esta Impresión se

planchó, con una plancha convencional precalentada durante cinco minutos, durante un minuto cuarenta y cinco segundos sobre una placa fenólica de 15x15 *cm*, y después se dejó enfriar.

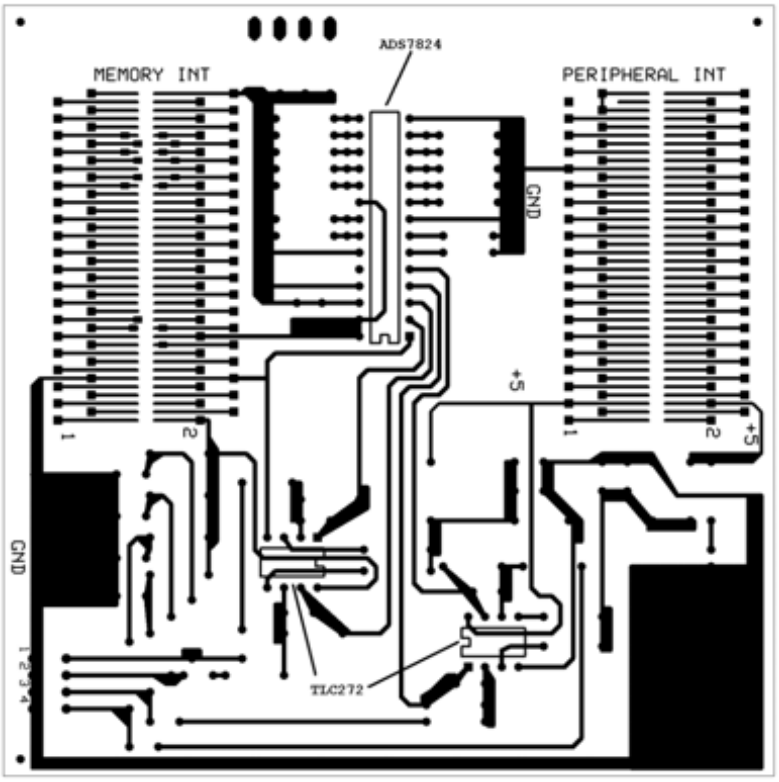


Figura D.2. Pistas del circuito impreso

La placa fenólica con la imagen planchada, se lavó para retirar el papel pegado, dejando solo el tóner, para después ser sometida a una solución de cloruro férrico manteniéndose en constante movimiento hasta que el cobre no cubierto por tóner se disolvió completamente. Después se cortó al tamaño de el proyecto y se perforó con una broca de $1\,mm$ de diámetro en los lugares necesarios.

Se procedió finalmente a soldar los componentes en los lugares correspondientes así como del los cables y conectores necesarios, obteniéndose el circuito mostrado en la figura D.3.

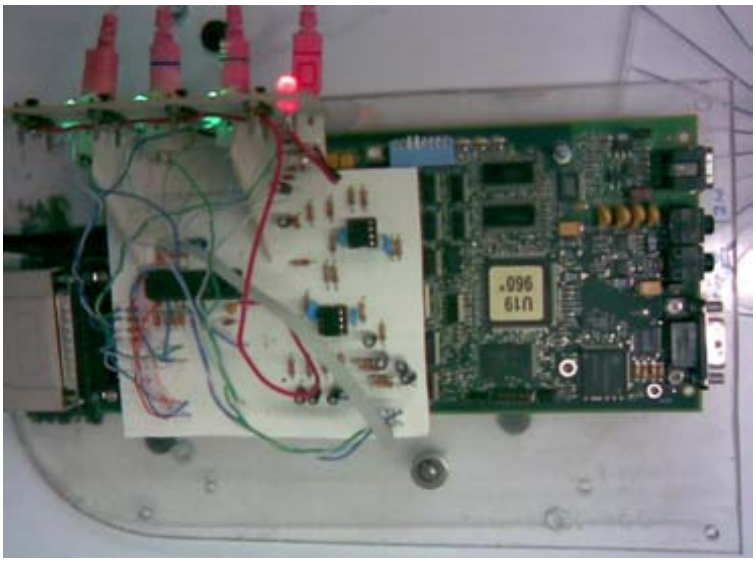


Figura D.3. Circuito Impreso

En la parte superior de la figura D.4 se observa la interfaz de usuario conformada por cuatro interruptores, cuatro conectores para micrófono de 3.5 *mm* y cuatro LEDs para indicar el estado de los interruptores.

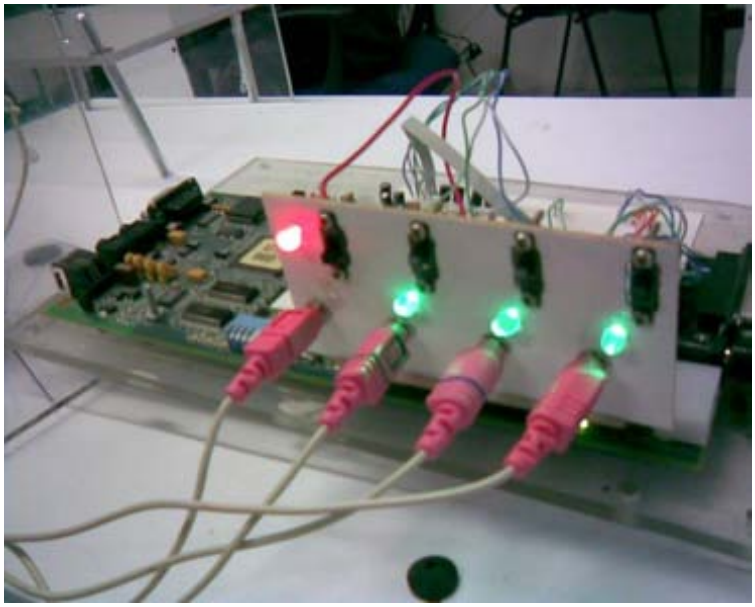


Figura D4. Interfaz de usuario

Bibliografía

[1] Peterson, Patrick, et al. ***Multimicrophone Adaptive Beamforming for interference reduction in hearing aids***. *Journal of Rehabilitation Research and Development*. Vol. 24. No. 4. 1987

[2] Texas Instruments. ***TMS320VC5402 DSK***. 1999
5402_dsk_schem.pdf

[3] Cohen, Leon. ***Time Frequency Analysis: Theory and Applications***. Prentice-Hall 1994

[4] Proakis, John. G., Manolakis, Dimitris. K. ***Digital Signal Processing: Principles, Algorithms, And Applications***. Prentice-Hall 1996

[5] Baher, Hussein. ***Analog & Digital Signal Processing***. Wiley 2001

[6] Kamen, Edward. ***Introduction To Signals And Systems***. Macmillan 1990

[7] Ziemer, Rodger, et al. ***Signals And Systems: Continous And Discrete***. Macmillan 1993

[8] Chu, Wai C. ***Speech Coding Algorithms: Foundation And Evolution Of Standardized Codes***. Wiley 2003

[9] Pelton, Gordon E. ***Voice Processing***. Mcgraw-Hill 1993

[10] Van Trees, Harry L. ***Detection Estimation And Modulation Theory Vol 4: Optimal Array Theory***. Wiley-Interscience 2002

[11] Mcpheeters, Claiborne. ***Array Signal Processing: An Introduction***. 2005
http://cnx.rice.edu/content/m12561/latest/

[12] Joiner, Sara, et al. ***Investigation Of Delay And Sum Beamforming Using A Two-Dimensional Array***. 2006
http://cnx.org/content/m13161/latest/

[13] Widrow, Bernard, Stearns, Samuel D. ***Adaptive Signal Processing***. Prentice-Hall 1985

[14] Frost, O. L. ***An Algorithm For Linearly Constrained Adaptive Array Processing***. *Proceedings of IEEE*. Vol. 60. No. 8. 1972,

[15] Griffiths L., Jim C. ***An Alternative approach to Linearly Constrained Adaptive Beamforming***. *IEEE Transaction on Antennas and Propagation*. Vol. 30. No. 1. 1982.

[16] Nielsen, Richard O. ***Sonar Signal Processing***. Artech House 1991

[17] Franco, Sergio; ***Design With Operational Amplifiers And Analog Integrated Circuits***. Mcgrawhill 2001

[18] Texas Instruments. ***TMS320C54x DSP Reference Set. Volume 1: CPU and Peripherals***. 2001
spru131g.pdf

[19] http://en.wikipedia.org/wiki/digital_signal_processor

[20] http://www.ti.com