ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA EN REDES Y COMUNICACIÓN DE DATOS

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN ELECTRÓNICA EN REDES Y COMUNICACIÓN DE DATOS

TEMA: IMPLEMENTACIÓN DE UN ARREGLO SEMIESFÉRICO DE MICRÓFONOS UTILIZANDO LA METODOLOGÍA BEAMFORMING DE BANDA ANCHA PARA EL REALCE DE LA VOZ

AUTOR: CAISAPANTA LARREA, TATIANA PAOLA

DIRECTOR: ING. SÁENZ, FABIÁN

SANGOLQUÍ, JULIO DE 2015

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

INGENIERÍA EN ELECTRÓNICA EN REDES Y COMUNICACIÓN DE DATOS

CERTIFICACIÓN

Ing. Fabián Sáenz

CERTIFICO

Que el proyecto de investigación titulado **"IMPLEMENTACION DE UN ARREGLO SEMIESFÉRICO DE MICRÓFONOS UTILIZANDO LA METODOLOGÍA BEAMFORMING DE BANDA ANCHA PARA EL REALCE DE LA VOZ"** realizado por Caisapanta Larrea Tatiana Paola, ha sido guiado y revisado periódicamente y cumple normas estatutarias establecidas por la ESPE, en el Reglamente de Estudiantes de la Universidad de las Fuerzas Armadas – ESPE.

Debido a que se trata de un trabajo de investigación recomiendan su publicación.

El mencionado trabajo consta de dos documentos anillados y un disco compacto, el cual contiene los archivos en formato portátil de Acrobat (pdf). Autorizan a Tatiana Paola Caisapanta Larrea que lo entregue al Ingeniero Nikolai Espinoza, en su calidad de Director de la Carrera.

Sangolquí, julio de 2015.

Ing. Fabián Sáenz
DIRECTOR

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

INGENIERÍA EN ELECTRÓNICA EN REDES Y COMUNICACIÓN DE DATOS

DECLARACIÓN DE RESPONSABILIDAD

Yo, TATIANA PAOLA CAISAPANTA LARREA

DECLARO QUE:

El proyecto de investigación denominado **"IMPLEMENTACION DE UN ARREGLO SEMIESFÉRICO DE MICRÓFONOS UTILIZANDO LA METODOLOGÍA BEAMFORMING DE BANDA ANCHA PARA EL REALCE DE LA VOZ"**, ha sido desarrollado en base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme las citas que constan al pie de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de mi autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance científico del proyecto de grado en mención.

Sangolquí, julio de 2015.

Srta. Tatiana Paola Caisapanta Larrea

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

INGENIERÍA EN ELECTRÓNICA EN REDES Y COMUNICACIÓN DE DATOS

AUTORIZACIÓN

Yo, TATIANA PAOLA CAISAPANTA LARREA

Autorizo a la Universidad de las Fuerzas Armadas – ESPE la publicación, en la biblioteca virtual de la Institución del proyecto de investigación **"IMPLEMENTACION DE UN ARREGLO SEMIESFÉRICO DE MICRÓFONOS UTILIZANDO LA METODOLOGÍA BEAMFORMING DE BANDA ANCHA PARA EL REALCE DE LA VOZ"**, cuyo contenido, ideas y criterios son de mi exhaustiva responsabilidad y autoría.

Sangolquí, julio de 2015.

Srta. Tatiana Paola Caisapanta Larrea

DEDICATORIA

Esta tesis está dedicada a dos ángeles, mis padres, quienes son mi ejemplo y mi motor de vida; mis hermanas Amelita y Maya por el apoyo incondicional que encontrado en ellas; mi sobrina Amelí por ser quien alegra mi vida en todo momento.

A mis instructores el Ing. Paul Bernal y el Ing. Fabián Sáenz por su apoyo y enseñanzas y paciencia.

AGRADECIMIENTO

Agradezco a mis padres quienes he recibido apoyo incondicional para la culminación de este proyecto.

A mi hermana Amelia por su ayuda y su amistad brindada, desde lo más profundo de mi corazón, gracias.

Gracias al Ing. Paúl Bernal y el Ing. Fabián Sáenz por su confianza, enseñanzas y amistad brindada.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El estudio de las señales de voz es un tema de importancia, dado que una parte de la sociedad presenta algún tipo de discapacidad auditiva. La presente implementación busca ayudar a las personas con problemas de audición, mediante el realzado de la voz; utilizando un arreglo semiesférico de micrófonos con la metodología de *beamforming* de banda ancha, que pueden distinguir señales que arriban de diferentes direcciones. Se utiliza un arreglo semiesférico de micrófonos se adapta mejor a la anatomía humana, lo cual permitirá captar de mejor manera el campo del sonido. Para el desarrollo de la presente implementación se utilizan los algoritmos adaptativos de la familia *Householder* con los algoritmos *unconstrained* NLMS (*Normalized Least Mean Squares*) y CG (*Conjugate Gradient*). Se analiza el desempeño de los mismos mediante la precisión finita puesto que la implementación práctica en procesadores digitales de señales, está limitada por el número de bits que pueden ser utilizados en los cálculos matemáticos internos de la tarjeta NI myRIO dispositivo de programación. Una vez implementado y desarrollado el programa en la tarjeta se analiza cual algoritmo es el más óptimo y finalmente obtener las conclusiones y recomendaciones generadas en la presente investigación.

PALABRAS CLAVES:

BEAMFORMING,

PRESICION FINITA,

ALGORITMOS ADAPTATIVOS,

NI myRIO.

ABSTRACT

The study of the voice signals is an important issue, since a part of society has some kind of hearing impairment. This implementation aims to help people with hearing problems, through enhanced voice; using a microphone array with hemispherical methodology broadband beamforming, which can distinguish signals arriving from different directions. a semi-spherical microphone array is used it is better suited to human anatomy, which will capture better sound field. For the development of this implementation of adaptive algorithms Householder family with unconstrained algorithms NLMS (Normalized Least Mean Squares) and CG (Conjugate Gradient) are used. Performance thereof is analyzed by the finite precision since the practical implementation in digital signal processors is limited by the number of bits that can be used in mathematical calculations internal NI Myrio card programming device. Once implemented and developed the program in which the card is analyzed algorithms is the most optimal and finally get the conclusions and recommendations generated in this investigation.

KEYWORDS:

BEAMFORMING,

FINITE PRECISION,

ALGORITMOS ADAPTATIVOS,

NI myRIO.

GLOSARIO

AWGN	*Additive White Gaussian Noise.*
CCG	*Constrained Conjugate Gradient.*
CG	*Conjugate Gradient.*
CLMS	*Constrained Least Mean Squares.*
DSP	*Digital Signal Processing.*
FPGA	*Field Programmable Gate Array.*
LCMV	*Linearly Constrained Minimum Variance*.
LMMSE	*Linear Minimum Mean Square Error.*
LMS	*Least Mean Squares.*
MOE	*Minimum Output Energy.*
MSE	*Mean Squared Error.*
NCLMS	*Normalized Constrained Least Mean Squares*
NLMS	*Normalized Least Mean Squares.*
RLS	*Recursive Least-Squares.*
SO	Sistema Operativo
TIC	Tecnologías de la Información y la Comunicación.
UCA	*Uniform Circular Array.*
ULA	*Uniform Linear Array.*
VLSI	*Very Large Scale Implementation.*

CAPÍTULO I

1. INTRODUCCIÓN

1.1. Justificación e Importancia

Los sistemas de Procesamiento Digital de Señales (*Digital Signal Processing* DSP) se han venido desarrollando de forma sostenible, permitiendo la representación, transformación y manipulación de señales discretas desde el punto de vista de la información que contienen. El procesamiento digital de señales de voz posee una amplia base teórica y experimental que se ha desarrollado en conjunto con el DSP durante los últimos 80 años. Además, existe una tecnología de implementación a gran escala (*Very Large Scale Implementation* VLSI) que se encuentra bien adaptada a las demandas computacionales del procesamiento digital de señales de voz (Rabiner & R. W., 2009).

Existen varias técnicas para el procesamiento de señales sonoras, dependiendo de la aplicación que se realice, en el presente proyecto de implementación se utiliza la metodología de *beamforming* que tradicionalmente eran estudiados en la teoría de antenas, substituyendo las ondas electromagnéticas por señales acústicas y el arreglo de antenas por un arreglo de micrófonos, presentando varias aplicaciones en diferentes campos como: videoconferencias, realzado de voz, manos libre para telefonía móvil, sistemas de ayuda para la escucha, sistemas de controladores de voz, reconocimiento de voz, etc.

El *beamforming* es una técnica del DSP (*Digital Signal Processing*) en la que se produce una mejora de la señal principal, mediante combinaciones lineales de las diferentes salidas de cada uno de los sensores del arreglo, de tal manera que las perturbaciones indeseadas se atenúan por el fenómeno de la directividad o selectividad espacial del arreglo (Van Veen & Buckley, 2000).

Los arreglos de sensores pueden variar de acuerdo a la geometría de los mismos y puede ser desde arreglos de una dimensión 1-D, arreglos de dos dimensiones 2-D y arreglos de tres dimensiones 3D; esto dependerá de la aplicación en la cual se desee emplear (Yang, Cho, & Choo, 2012).

Si bien por si solo los arreglos de sensores son simples elementos pasivos sin ningún elemento activo; a la salida del arreglo es conectado un procesador digital de señales para procesar, discriminar y mejorar la señal de llegada, para lo cual se utiliza algoritmos, los mismos que se encargan de realizar tales tareas dentro de los cuales podemos nombrar (Yang, Cho, & Choo, 2012).

- Método *Delay and Sum*
- Método Capon's de Mimima Varianza
- Algoritmo MUSIC
- Algoritmo ESPRIT
- Algoritmo LCMV *Linearly Constrained Minimum Variance*

Los algoritmos mencionados tienes cualidades y defectos como es el caso del costo computacional, velocidad de convergencia, inmunidad al ruido, tipo de implementación para el hardware, etc. (Caisapanta, 2015).

El estudio de las señales de voz es un tema de importancia, dado que una gran parte de la sociedad presenta algún tipo de discapacidad auditiva. La discapacidad auditiva se considera como aquellas alteraciones cuantitativas en una correcta percepción de la audición. La presente implementación busca ayudar a las personas con problemas de audición, mediante el realzado de la voz; utilizando un arreglo semiesférico de micrófonos con la metodología de *beamforming* de banda ancha, que pueden distinguir señales que arriban de diferentes direcciones. Se utiliza un arreglo semiesférico de micrófonos se adapta mejor a la anatomía humana, lo cual permitirá captar de mejor manera el campo del sonido, como se estudió en el proyecto de tesis "Optimización de las Señales Acústicas en un Arreglo Semiesférico de Micrófonos Utilizando la Metodología *Beamforming* de Banda Ancha".

Los algoritmos adaptativos LCMV se dividen en tres familias: *Linearly Constrained* tal es el caso de los CLMS (*Constrained Least Mean Square*), entre otros. Además los algoritmos de estructura GSC (*Generalized Sidelobe Canceler*) que permite que cualquier algoritmo *unconstrained* pueda ser empleado como *constrained*. Por último se tiene los de estructura *Householder* que mediante la matriz Q y sus reflectores de *Householder* cumple como el GSC pero computacionalmente son más eficiente (Apolinário J. A., 2009).

Para el desarrollo de la presente implementación se utilizan los algoritmos adaptativos de la familia *Householder* con los algoritmos *unconstrained* NLMS (*Normalized Least Mean Squares*) y CG (*Conjugate Gradient*). El algoritmo HNCLMS (*Householder Normalized Constrained Least Mean Squares*) posee una lenta velocidad de convergencia versus al algoritmo HCCG (*Householder Constrained Conjugate Gradient*) que es de rápida convergencia. El costo computacional del algoritmo HNCLMS es menor en comparación al algoritmo HCCG, por lo que el algoritmo HNCLMS se torna deseable para implementación en dispositivos con prestaciones de hardware limitadas (Caisapanta, 2015).

1.2. Alcance del Proyecto

En el presente proyecto de graduación se implementa un arreglo semiesférico de micrófonos, para lo cual se utiliza cuatro micrófonos MEMS ADMP504. Se estudia la respuesta de los algoritmos adaptativos LCMV de la familia *Householder* utilizando los algoritmos *unconstrained* NLMS y CG en una tarjeta FPGA (*Field Programmable Gate Array*) de la marca NI myRIO 1900.

Se utiliza la herramienta de simulación Matlab® para obtener los programas de precisión finita de los algoritmos HCNLMS y HCCG y poder analizar el desempeño de los mismos, puesto que la implementación práctica en procesadores digitales de señales, está limitada por el número de bits que pueden ser utilizados en los cálculos matemáticos internos de la tarjeta FPGA y por la precisión con que se almacenan los valores utilizados.

1.3. Objetivos

1.3.1. Objetivo General

Implementar un arreglo semiesférico de micrófonos que permita realzar la voz de manera adecuada utilizando algoritmos adaptativos LCMV en un dispositivo FPGA NI myRIO-1900.

1.3.2. Objetivos Específicos

- Verificar el desempeño de los algoritmos de Precisión Finita.
- Analizar la estructura *Lattice*.
- Estudiar la respuesta de los micrófonos.
- Implementar el arreglo de micrófonos.
- Realizar la programación de los algoritmos en el dispositivo FPGA.
- Determinar el algoritmo más idóneo para su implementación en hardware.

CAPÍTULO II

2. FUNDAMENTO TEÓRICO Y CONCEPTUAL

2.1. La Voz

La voz es la principal forma de comunicación del ser humano. El proceso de la voz comienza con el pensamiento de un mensaje representado de forma abstracta en el cerebro del locutor y a través del complejo proceso de producción de la voz, la información se convierte en última instancia en una señal acústica (Rabiner & R. W., 2009).

La voz se caracteriza por ser la combinación de varias frecuencias con sus correspondientes armónicos. Sin embargo, a comparación del rango completo de la audición humana que va desde los $20\,Hz$ a los $20\,kHz$, la voz cubre un rango relativamente estrecho de frecuencias entre $100\,Hz$ hasta $6\,kHz$.

La voz es una señal no estacionaria, pero se la puede asumir mediante ventanas (pequeños bloques de muestras) como localmente estacionaria. Las ventanas entre 20 a $30\,ms$ son las más adecuadas para la mayoría de las aplicaciones.

El procesamiento de las señales de voz, para muchas aplicaciones, está relacionado al análisis de sus características significativas, como por ejemplo podemos citar al *pitch* (tono fundamental o frecuencia de vibración de las cuerdas vocales) y los formantes (picos del espectro de un sonido sonoro). La voz masculina posee un *pitch* entre 100 y $200\,Hz$ y la voz femenina típicamente se encuentra entre 150 y $300\,Hz$ (Apolinário J. , 2003).

El procesamiento digital de señales de voz, es un tema de índole interdisciplinario que involucra la fonética, fisiología, acústica, entre otras disciplinas, además de la teoría del procesamiento digital de señales. Este último ha tenido un gran avance gracias al desarrollo de la tecnología de los procesadores de

señales digitales, encontrándose innumerables aplicaciones en la vida moderna ya sea en la codificación, reconocimiento o síntesis de la voz (Apolinário J. , 2003).

2.2. *Beamforming*

Filtraje espacial o *Beamforming* es una técnica que utiliza un arreglo de sensores; donde se estima la señal principal mediante combinaciones lineales de las diferentes salidas de cada uno de los sensores del arreglo, de tal manera que las perturbaciones indeseadas que vienen de diferentes direcciones se atenúan por el fenómeno de la directividad o selectividad espacial del arreglo (Van Veen & Buckley, 2000).

2.2.1. Tipos de *Beamforming*

El *beamforming* se divide en dos tipos: de banda angosta (*Narrowband Beamforming)* y de banda ancha (*Broadband Beamforming*) (Madisetti, 2010).

- ***Narrowband Beamforming*.-** En la Figura 1 se muestra este tipo de *Beamforming,* en donde las señales se muestrean únicamente en el espacio. Este filtro es usando cuando las señales de interés son de banda angosta (*narrowband signals*). La salida del *narrowband beamforming* está representado por la Ecuación 1.

$$y[k] = \sum_{m=1}^{M} w_m x_m[k] \tag{1}$$

Donde la salida $y[k]$ se encuentra en el tiempo discreto k y es dada por la combinación lineal de los datos de los M sensores en el tiempo k, w_m son los pesos (coeficientes del filtro) y $x_m[k]$ la señal a la salida de cada uno de los sensores (Madisetti, 2010).

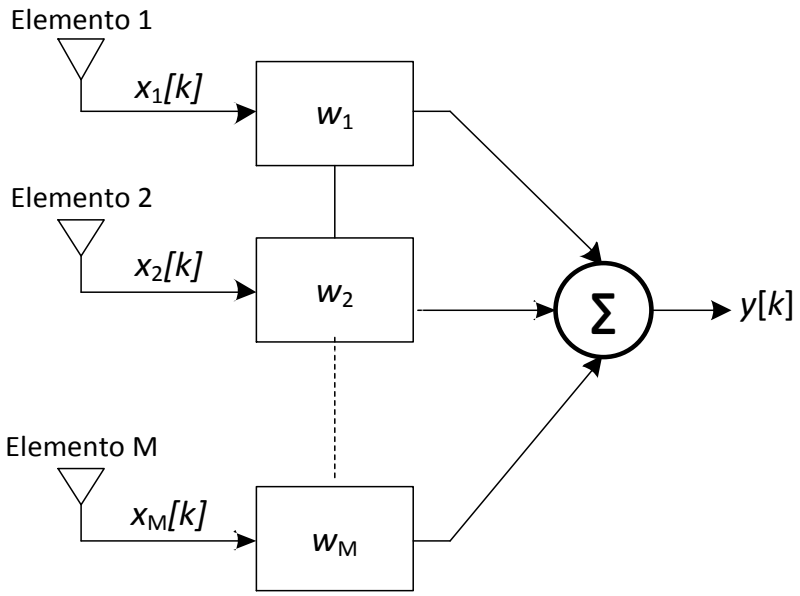


Figura 1. ***Narrowband Beamforming***
Fuente: (Madisetti, 2010)

- ***Broadband Beamforming.-*** En este tipo de *beamforming* se muestrea las señales en tiempo y espacio. Se utiliza cuando la señal de interés tienen una amplia frecuencia (*broadband signals*). La salida $y[k]$ está dada por la siguiente ecuación:

$$y[k] = \sum_{m=1}^{M} \sum_{i=0}^{N-1} w_{m,i} x_m[k-i] \tag{2}$$

Donde $N-1$ es el número de retardos que llegan en diferentes instantes a los M sensores. (Madisetti, 2010). En la Figura 2 se muestra este tipo de *beamforming*.

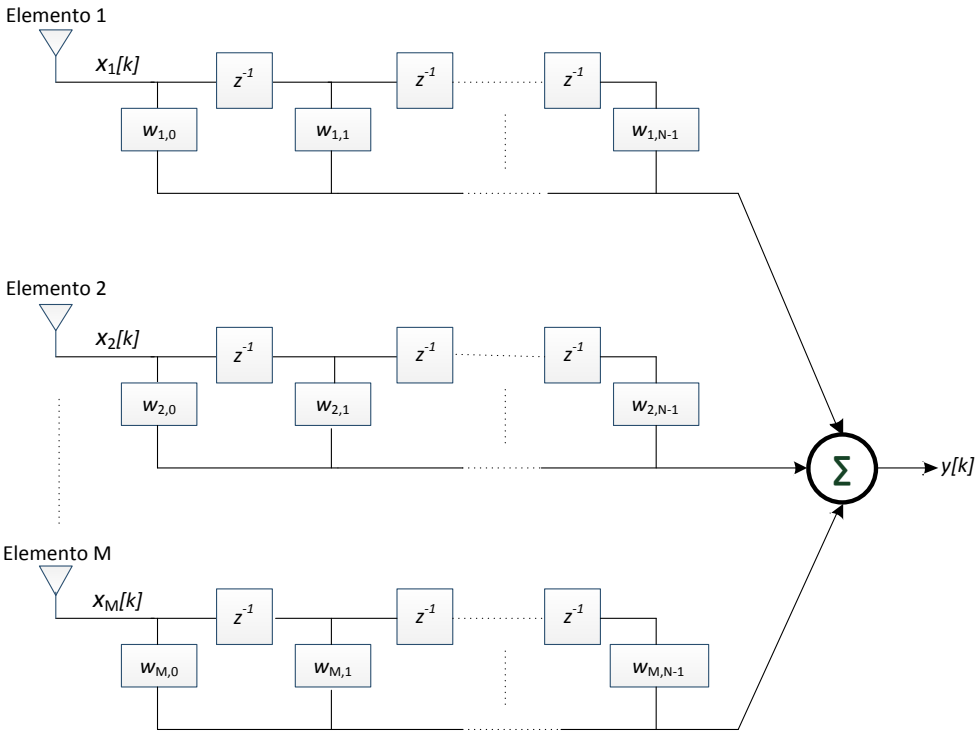


Figura 2. ***Broadband Beamforming***
Fuente: (Madisetti, 2010)

La salida del *Narrowband* y *Broadband Beamforming* se puede expresar en forma vectorial, como se puede observar en la siguiente ecuación.

$$y[k] = \mathbf{w}^H \mathbf{x}[k] \tag{3}$$

Según como se estime los pesos se tiene dos categorías: El *beamforming* fijo, en donde los pesos se encuentran asociados a cada sensor y sus pesos son fijos. La segunda categoría es el *beamforming* adaptativo, en el cual los pesos se ajustan dinámicamente dependiendo de los datos recibidos.

2.3. Arreglo de Sensores

Un arreglo de sensores, es un conjunto de elementos isotrópicos distribuidos en una determinada geometría, para así obtener información de los campos de onda en

algún medio sobre el cual se propagan. Los arreglos de sensores permiten que cada sensor tenga una mayor directividad y sensibilidad. Los arreglos de sensores más utilizados son: el arreglo lineal uniforme (*Uniform Linear Array* ULA) y el arreglo circular uniforme (*Uniform Circular Array* UCA).

Si se tiene una onda plana que llega al punto $\vec{r}$ en un tiempo t se obtiene la Ecuación 4. Una onda plana se caracteriza porque posee una misma amplitud y fase.

$$s(\mathbf{r},t) = Ae^{j(\boldsymbol{\beta}\cdot\mathbf{r}-\omega t)} \quad con \quad \begin{cases} \beta = \|\boldsymbol{\beta}\| = \dfrac{2\pi}{\lambda} = \dfrac{2\pi}{\vartheta/f} \end{cases} \tag{4}$$

Donde A es la amplitud de onda, $\boldsymbol{\beta}$ el vector de onda (propagación), β factor de onda (número de ondas), λ es la longitud de onda, ϑ velocidad de propagación de la onda, f es la frecuencia (Yang, Cho, & Choo, 2012).

Se considera que la señal de onda plana de la Ecuación 4 llega a un arreglo ULA, como se muestra en la Figura 3, en la posición (r_x, r_y, r_z) y arriba al origen $(0,0,0)$ se tiene:

$$s(\mathbf{r},t) = s(\mathbf{0},t)e^{j\beta(r_x\cos\varphi\sin\theta + r_y\cos\varphi\sin\theta + r_z\cos\varphi)} \tag{5}$$

Donde φ y θ representan el ángulos azimut y el ángulo de elevación respectivamente. (Yang, Cho, & Choo, 2012).

2.3.1. Arreglo Lineal Uniforme (ULA)

Un Arreglo Lineal Uniforme considera M sensores colocados sobre una línea recta, igualmente espaciados a una distancia d, como se observa en la Figura 3. Siendo que sobre el arreglo incide un frente de onda plana, el frente de onda es perpendicular a la dirección de propagación de la onda plana. La señal $s(t)$ arriba al m-enésimo sensor según la Ecuación (6).

$$s_m(t) = s\{(md, 0, 0), t\} = s(\mathbf{0}, t)a_m(\varphi, \theta) \quad (6)$$

Donde el factor de dirección (*steering factor*) del m-enésimo elemento es:

$$a_m(\varphi, \theta) = e^{j\beta md \cos\varphi \sin\theta} \quad (7)$$

$$a_m(\varphi, \theta) = e^{j2\pi m\frac{d}{\lambda}\cos\varphi \sin\theta} \quad (8)$$

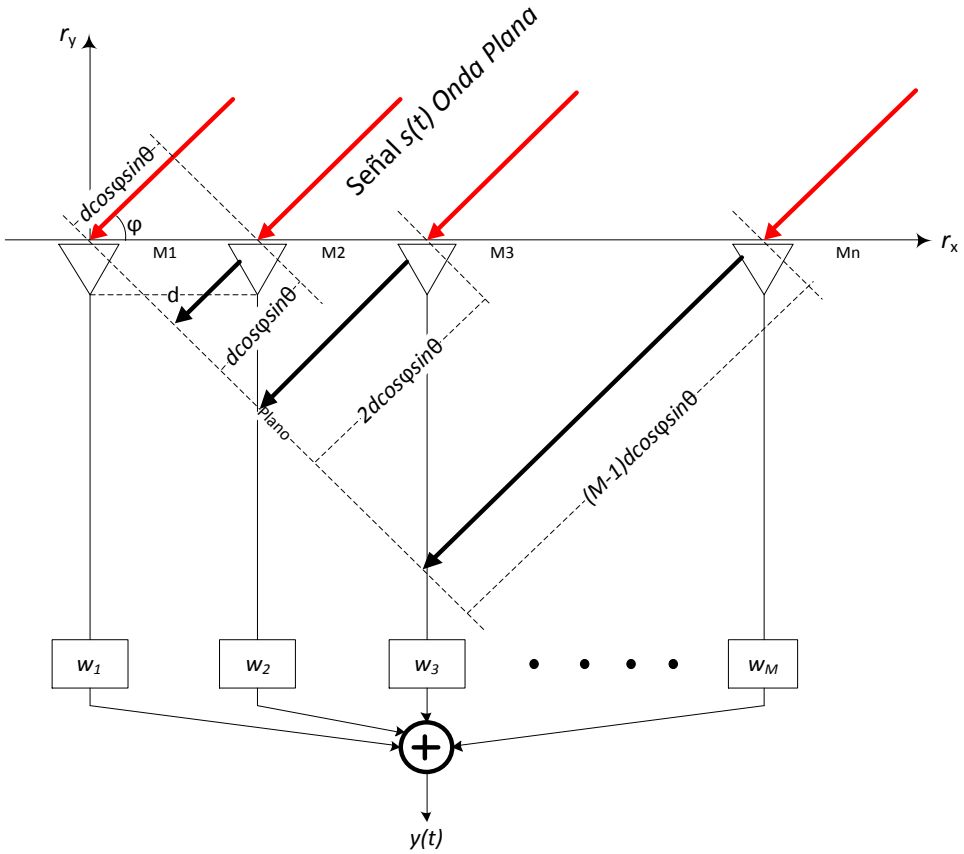


Figura 3. Arreglo Lineal Uniforme ULA
Fuente: (Yang, Cho, & Choo, 2012)

2.3.2. Arreglo Semiesférico

El arreglo semiesférico que se utiliza para la presente implementación consta de un semianillo con 4 micrófonos espaciados uniformemente, es decir, que la apertura

entre cada micrófono adyacente es de 60° y el radio ɾ tiene una dimensión de 15 cm. (Caisapanta, 2015).

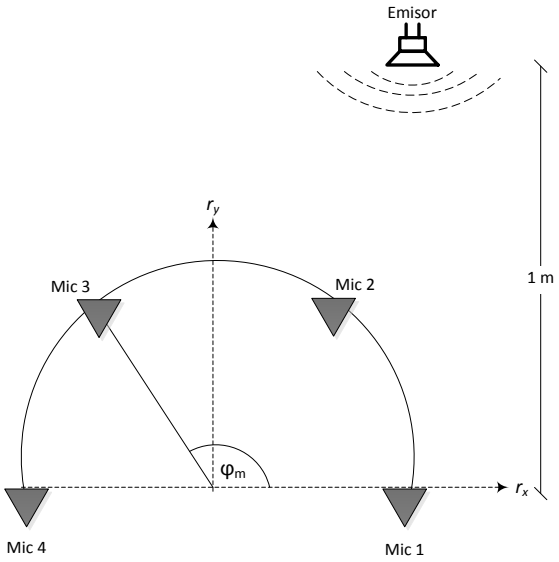


Figura 4. Arreglo Semiesférico

2.4. Algoritmos Adaptativos LCMV

Los algoritmos adaptativos LCMV *(Linearly Constrained Minimum Variance)* tiene como objetivo restringir en amplitud y fase la salida del *beamformer* a señales que llegan de una o más direcciones de no interés. Los pesos del filtro son seleccionados de tal forma que minimice la varianza o la potencia del error sujeto a un conjunto de restricciones lineales, como se muestra en la Ecuación 9.

En la Figura 5 se muestra una estructura de *beamforming* adaptativo *broadband* con M sensores y N pesos (número de coeficientes del filtro).

$$\min_{\mathbf{w}} \xi[k] \quad sujeto\ a \quad \mathbf{C}^H\mathbf{w} = \mathbf{f} \tag{9}$$

Donde la señal de error $e[k]$ está relacionada con la función objetivo $\xi[k] = E\{|e[k]|^2\} = E\{e[k]e^*[k]\}$, $\mathbf{C}$ es la matriz de restricción de dimensión MN × p, siendo p el número de restricciones y **f** el vector de ganancia de dimensión p × 1.

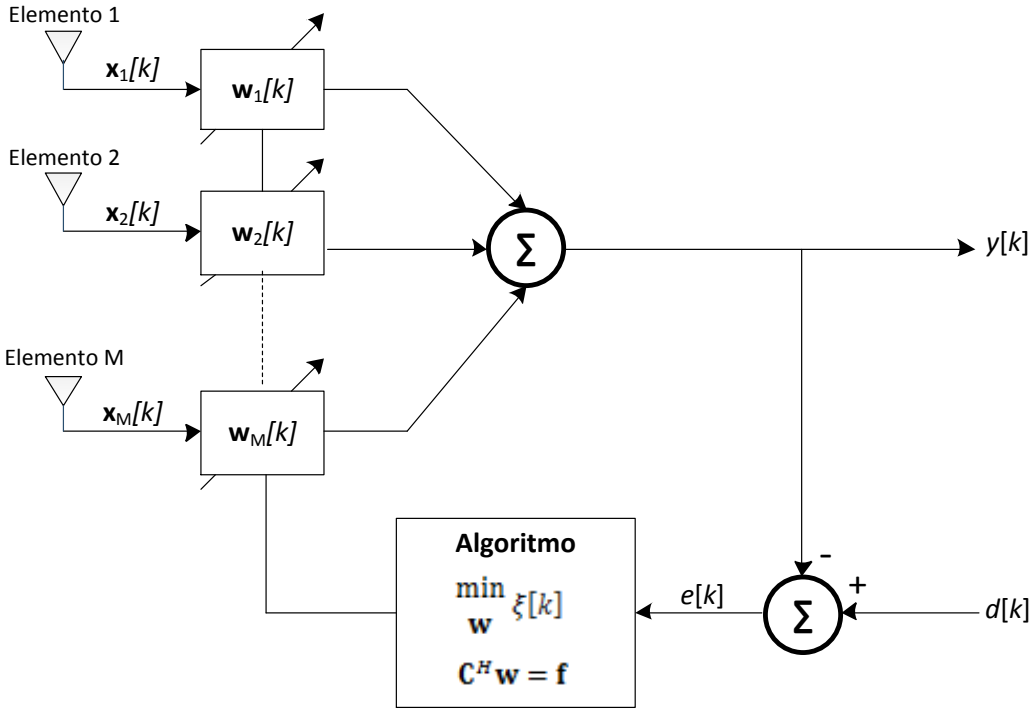


Figura 5. Estructura de *Beamforming* Adaptativo *Broadband*
Fuente: (Apolinario & de Campos, 2011)

Mediante la imposición de restricciones lineales la necesidad de una señal deseada a menudo puede ser descartada $d[k] = 0$. Los algoritmos adaptativos LCMV se dividen en tres familias: *Linearly Constrained*, de estructura GSC (*Generalized Sidelobe Canceler*) que permite que cualquier algoritmo *unconstrained* pueda ser empleado como constrained y por último se tiene los de estructura *Householder* (Apolinário J. A., 2009).

La estructura *Householder* es la que mejor desempeño presenta en comparación a las otras dos estructuras con los algoritmos adaptativos *unconstrained* NLMS (*Normalized Least Mean Squares*) y CG (*Conjugate Gradient*) para el desarrollo de la presente implementación (Caisapanta, 2015).

La estructura *Householder* es una de las mejores opciones para implementaciones prácticas por su bajo costo computacional ya que la matriz $\mathbf{Q}$ y sus reflectores de *Householder* son computacionalmente más eficientes (de Campos, Werner, & Apolinário, September 2002).

La matriz $\mathbf{Q}$ es una matriz rotacional ortogonal que se utiliza como la transformación que va a generar un vector de coeficientes modificados $\bar{\mathbf{w}}[k]$ que se relaciona a $\mathbf{w}[k]$ (de Campos, Werner, & Apolinário, September 2002).

$$\bar{\mathbf{w}}[k] = \mathbf{Q}\mathbf{w}[k] \tag{10}$$

La matriz $\mathbf{Q}$ se debe escoger de tal manera que $\mathbf{Q}\mathbf{Q}^H = \mathbf{I}$.

$$\bar{\mathbf{C}}\{\bar{\mathbf{C}}^H\bar{\mathbf{C}}\}^{-1}\bar{\mathbf{C}}^H = \begin{bmatrix} \mathbf{I}_{\mathrm{p}\times\mathrm{p}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{11}$$

Entonces $\bar{\mathbf{C}} = \mathbf{Q}\mathbf{C}$ satisface $\mathbf{f} = \bar{\mathbf{C}}^H\bar{\mathbf{w}}[k+1]$ y la transformada de la matriz de proyección está dada por:

$$\bar{\mathbf{P}} = \mathbf{Q}\mathbf{P}\mathbf{Q}^H = \mathbf{I} - \bar{\mathbf{C}}\{\bar{\mathbf{C}}^H\bar{\mathbf{C}}\}^{-1}\bar{\mathbf{C}}^H = \begin{bmatrix} \mathbf{0}_{\mathrm{p}\times\mathrm{p}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{12}$$

Si $\bar{\mathbf{w}}[0]$ se inicializa según.

$$\bar{\mathbf{w}}[0] = \bar{\mathbf{C}}\{\bar{\mathbf{C}}^H\bar{\mathbf{C}}\}^{-1}\mathbf{f} = \mathbf{Q}\mathbf{F} \tag{13}$$

Los primeros p elementos de $\bar{\mathbf{w}}[0]$ no necesitan ser actualizados. La solución $\bar{\mathbf{w}}[k]$ es basada en la transformación $\mathbf{Q}$ a la señal de salida y como consecuencia el error de salida no se modifica por la transformación.

2.4.1. *Householder* NCLMS

El algoritmo HNCLMS posee una lenta velocidad de convergencia y su costo computacional es bajo $(2\mathrm{p}+3)\mathrm{N}-(\mathrm{p}^2+\mathrm{p}-1)$ (Caisapanta, 2015), por lo que el algoritmo se torna deseable para implementación en dispositivos con prestaciones de hardware limitadas. El algoritmo HNCLMS se detalla en la Tabla 1.

Tabla 1. Algoritmo HNCLMS

Inicialización:

$\mathbf{w}_{TH}[0] = \mathbf{QF}$

$\mathbf{w}_{TH}[k] = \begin{bmatrix} \widehat{\mathbf{w}}[0] \\ -\breve{\mathbf{w}}[k] \end{bmatrix}$

$\mathbf{x}_{TH}[k] = \begin{bmatrix} \hat{\mathbf{x}}[k] \\ \breve{\mathbf{x}}[k] \end{bmatrix}$

Para cada k

$\mathbf{x}_{TH}[k] = \mathbf{Q}\mathbf{x}[k]$

$e_{TH}[k] = \widehat{\mathbf{w}}^H[0]\hat{\mathbf{x}}[k] - \breve{\mathbf{x}}^H[k]\breve{\mathbf{w}}[k]$

$\breve{\mathbf{w}}[k+1] = \breve{\mathbf{w}}[k] + \mu_n \dfrac{\breve{\mathbf{x}}^H[k] e_{TH}^*[k]}{\gamma + \breve{\mathbf{x}}^H[k]\breve{\mathbf{x}}[k]}$

2.4.2. *Householder CCG*

El algoritmo HCCG es de rápida convergencia y su costo computacional $(3\mathrm{N}^2+(10-4\mathrm{p})\mathrm{N}+\mathrm{p}^2-8\mathrm{p}+2)$ es mayor en comparación al algoritmo HNCLMS, por lo que es deseable cuando su implementación es en un hardware con alto recursos. En la Tabla 2 se detalla el algoritmo HCCG.

Tabla 2. Algoritmo HCCG

Inicialización:

$$\mathbf{w}_{TH}[\mathbf{0}] = \mathbf{QF}$$

$$\mathbf{w}_{TH}[\boldsymbol{k}] = \begin{bmatrix} \widehat{\mathbf{w}}[\mathbf{0}] \\ -\breve{\mathbf{w}}[\boldsymbol{k}] \end{bmatrix}$$

$$\mathbf{x}_{TH}[\boldsymbol{k}] = \begin{bmatrix} \hat{\mathbf{x}}[\boldsymbol{k}] \\ \breve{\mathbf{x}}[\boldsymbol{k}] \end{bmatrix}$$

$$\mathbf{g}[\mathbf{0}] = \mathbf{c}[\mathbf{0}] = \mathbf{0}$$

Para cada $\boldsymbol{k}$

$$\mathbf{x}_{TH}[\boldsymbol{k}] = \mathbf{Q}\mathbf{x}[\boldsymbol{k}]$$

$$\mathbf{R}_{TH}[\boldsymbol{k}] = \boldsymbol{\lambda_f}\mathbf{R}_{TH}[\boldsymbol{k}-\mathbf{1}] + \breve{\mathbf{x}}[\boldsymbol{k}]\breve{\mathbf{x}}^{\boldsymbol{H}}[\boldsymbol{k}]$$

$$\boldsymbol{\alpha}[\boldsymbol{k}] = \boldsymbol{\eta}\frac{\mathbf{c}^{\boldsymbol{H}}[\boldsymbol{k}]\mathbf{g}[\boldsymbol{k}-\mathbf{1}]}{\mathbf{c}^{\boldsymbol{H}}[\boldsymbol{k}]\mathbf{R}_{\boldsymbol{TH}}[\boldsymbol{k}]\mathbf{c}[\boldsymbol{k}] + \boldsymbol{\delta}}$$

$$\boldsymbol{e}_{\boldsymbol{TH}}[\boldsymbol{k}] = \widehat{\mathbf{w}}^{\boldsymbol{H}}[\mathbf{0}]\hat{\mathbf{x}}[\boldsymbol{k}] - \breve{\mathbf{x}}^{\boldsymbol{H}}[\boldsymbol{k}]\breve{\mathbf{w}}[\boldsymbol{k}]$$

$$\mathbf{g}[\boldsymbol{k}] = \boldsymbol{\lambda_f}\mathbf{g}[\boldsymbol{k}-\mathbf{1}] - \boldsymbol{\alpha}[\boldsymbol{k}]\mathbf{R}_{\boldsymbol{TH}}[\boldsymbol{k}]\mathbf{c}[\boldsymbol{k}] - \breve{\mathbf{x}}[\boldsymbol{k}]\boldsymbol{e}^{*}_{\boldsymbol{TH}}[\boldsymbol{k}]$$

$$\breve{\mathbf{w}}[\boldsymbol{k}] = \breve{\mathbf{w}}[\boldsymbol{k}-\mathbf{1}] - \boldsymbol{\alpha}[\boldsymbol{k}]\mathbf{c}[\boldsymbol{k}]$$

$$\boldsymbol{\beta}[\boldsymbol{k}] = \frac{\{\mathbf{g}[\boldsymbol{k}] - \mathbf{g}[\boldsymbol{k}-\mathbf{1}]\}^{\boldsymbol{H}}\mathbf{g}[\boldsymbol{k}]}{\mathbf{g}^{\boldsymbol{H}}[\boldsymbol{k}-\mathbf{1}]\mathbf{g}[\boldsymbol{k}-\mathbf{1}] + \boldsymbol{\delta}}$$

$$\mathbf{c}[\boldsymbol{k}+\mathbf{1}] = \mathbf{g}[\boldsymbol{k}] + \boldsymbol{\beta}[\boldsymbol{k}]\mathbf{c}[\boldsymbol{k}]$$

CAPÍTULO III

3. CÁLCULOS EN PRESICIÓN FINITA

3.1. Introducción

En este capítulo se presenta la simulación de precisión finita, una herramienta fundamental para evaluar la respuesta real de la implementación en hardware del presente proyecto, puesto que permite determinar la precisión que se encuentra limitada por el número de bits. Los efectos de precisión finita son un factor importante en la implementación en hardware debido a la limitación de la precisión del cálculo (redondeo en las multiplicaciones y desbordamiento en la sumas) y la limitación de la precisión de los datos (cuantificación de las señales y de los coeficientes utilizados) afecta el funcionamiento del sistema digital. Las representaciones equivalentes de un mismo sistema (suponiendo cálculos internos con precisión infinita) no produce los mismos resultados cuando se utiliza precisión finita. Además la precisión finita únicamente no afecta a los resultados obtenidos sino que condicionan a las prestaciones del sistema (área ocupada y velocidad de operación). (T. SANSALONI, 2000).

En particular, en una implementación digital de un algoritmo de filtrado adaptativo como encontrado en la práctica, los datos de entrada y los cálculos internos del algoritmo se realizan con precisión finita, y el costo de la implementación digital de un algoritmo está influenciado por el número de bits (precisión) disponibles para efectuar los cálculos numéricos asociados al algoritmo.

La simulación se realizara con la herramienta Matlab® que emula la operación de precisión finita; trabajando con el número de bits a la cual el hardware trabaja en este caso de 16 bits, se realizara la simulación para 8, 12 y 16 bits.

La teoría del filtraje adaptativo considera que todas las variables involucradas en los cálculos y las señales de entrada pueden ser representadas en precisión finita,

facilitando así el análisis matemático, pero la implementación práctica en procesadores digitales de señales, está limitada por el número de bits que pueden ser utilizados en los cálculos matemáticos internos y la precisión con que se almacenan los valores utilizados.

En una implementación digital de un filtro adaptativo existe esencialmente dos fuentes de errores de cuantización a ser consideradas:

- Conversión analógica – digital
- Aritmética de longitud finita de palabra

3.1.1. Conversión Análoga/Digital

La conversión análogo-digital (A/D) puede ser visualizada por un conversor continuo-discreto (C/D) ideal, seguido por un cuantizador. La operación de cuantización será representada por (Apolinário J. , 2003).

$$\bar{x}[k] = Q\{x[k]\} \tag{14}$$

Donde el operador $Q\{\,.\,\}$ efectúa una operación no lineal de redondeo del valor de $x[k]$ para el nivel de cuantización más próximo. El espaciamiento de los niveles de cuantización define el paso de cuantización, uniforme o no uniforme. Por ejemplo, una cuantización uniforme con 3 bits que se observa en la Figura 6**.** Se observa que los 3 bits corresponden a $2^3 = 8$ niveles de cuantización. Además, se observa en la figura que $x(k) > \frac{7\Delta}{2}$ o $x(k) < \frac{-9\Delta}{2}$, vamos a tener una saturación.

En el procesamiento digital de señales (PDS), tenemos una salida al conversor A/D valor perteneciente a uno de los posibles niveles de cuantización representado de acuerdo con un sistema de numeración.

También tenga en cuenta que podemos representar un rango de valores de $\left(\frac{7\Delta}{2}\right) - \left(\frac{9\Delta}{2}\right) = 8\Delta$ sin saturación por ejemplo de 3 bits, que corresponde a dos veces el valor máximo de la escala X_m.

Por lo tanto $\Delta = \left(\frac{2X_m}{2^{\#}}\right)$, donde # es el número de bits, y suponiendo que nuestro numero binario (salida del decodificador) $\widehat{x_B}(k)$ donde $-1 \leq \widehat{x_B}(k) < 1$, después $\hat{x}(k) = X_m \widehat{x_b}(k)$.

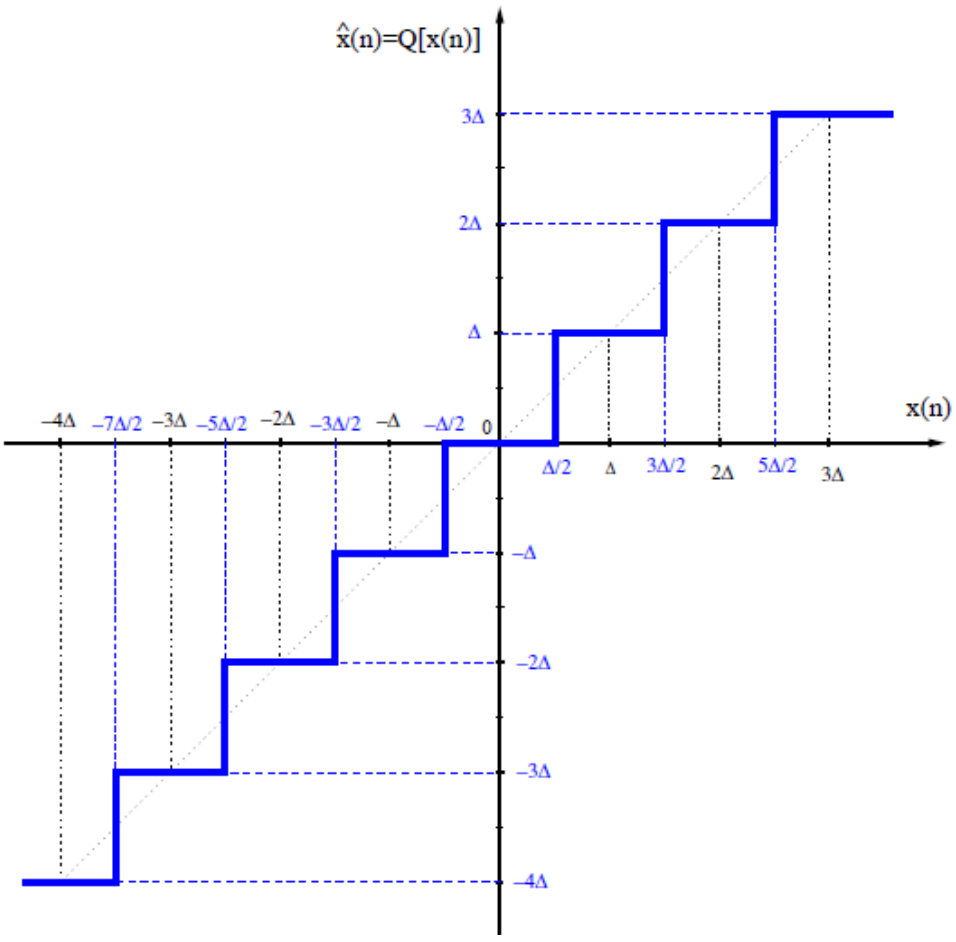


Figura 6. Ejemplo de Cuantización de 3 bits

La cuantización introduce un error (error de cuantización) definido por (Apolinário J. , 2003).

$$e[k] = x[k] - \bar{x}[k] \tag{15}$$

De modo que, en caso ejemplificado de cuantización uniforme no existe saturación ("desbordamiento"), $-\Delta/2 \leq e(k) \leq \Delta/2$.

El error promedio es cero y su varianza está dada por:

$$\sigma_e^2 = \frac{\Delta^2}{12} = \frac{4X_m^2 2^{-2\#}}{12} = \frac{X_m^2 2^{-2(\#-1)}}{12} \tag{16}$$

Donde # es el número de bits.

3.2. Programación

La herramienta de simulación Matlab® internamente trabaja con una precisión numérica elevada de 32 o 64 bits. En el presente proyecto el hardware de implementación una FPGA NI myRIO trabaja con 16 bits, por lo que para llevar a cabo el análisis de precisión finita con la función ***qround***, que emula el entero más próximo.

La función ***round*** va a permitir cuantificar la parte decimal de cada una de las variables mediante el redondeo de *b* bits, como se puede observar a continuación.

```
function Vq = qround(V,b)
   Vq=2^(-b)*round(V*2^b);
end
```

3.3. Análisis de Resultados

Al variar el número de bits entre 8, 12 y 16 para el algoritmo HNCLMS el MSE empieza a sufrir variaciones, es así que a medida que el número de bits aumenta el

algoritmo HNCLMS presenta un mejor desempeño, disminuyendo su MSE, como se puede observar en la Figura 7.

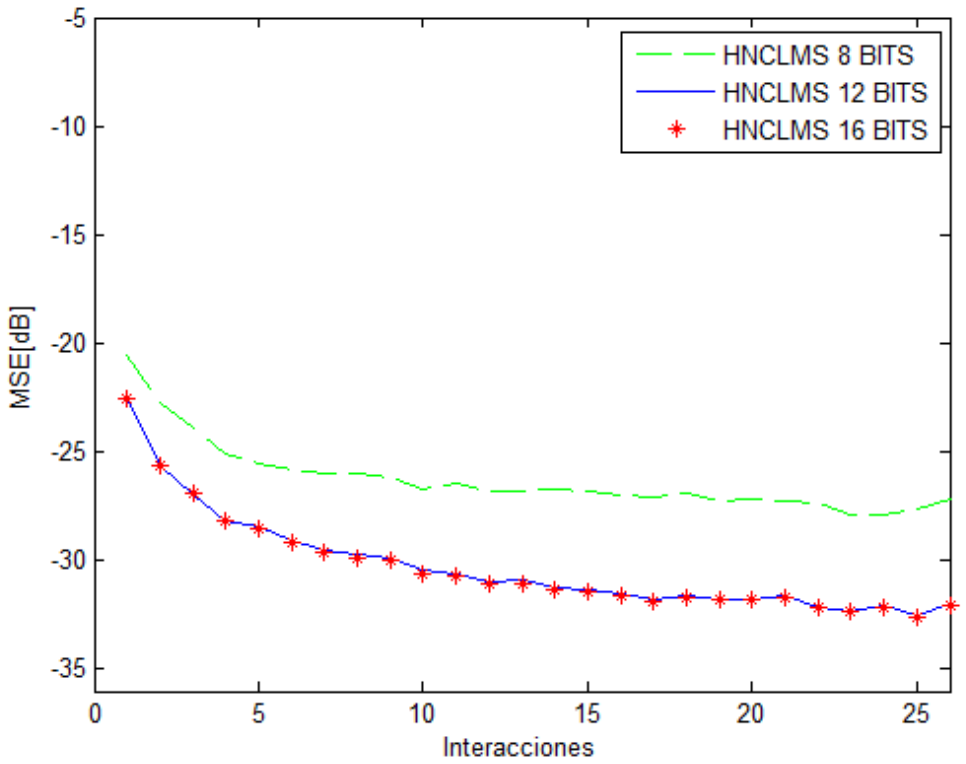


Figura 7. Algoritmo HNCLMS para varios tamaños de bits

En la Figura 8 se puede observar el comportamiento del algoritmo HCCG al variar el tamaño de los bits de 8, 12 y 16. En el algoritmo HCCG su desempeño varia al aumentar el tamaño de los bits, ya que su MSE empieza a disminuir a medida que el número de bits aumenta.

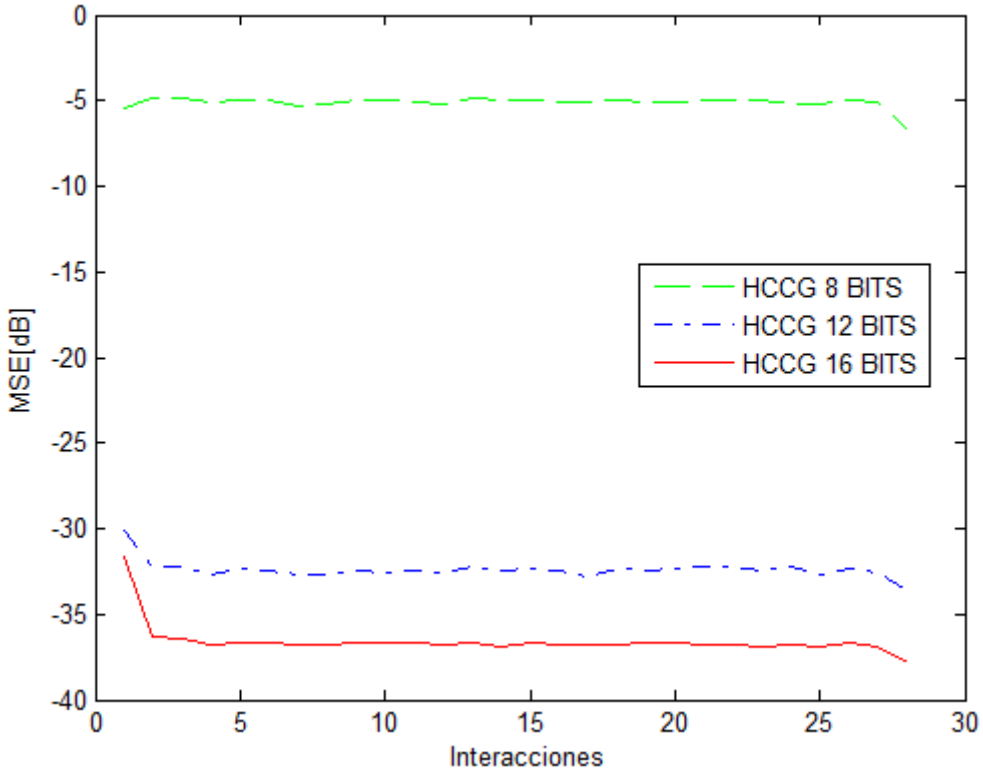


Figura 8. Algoritmo HCCG para varios tamaños de bits

3.4. Comparación de Resultados

Como se mencionó el costo de la implementación digital está influenciado por el número de bits disponibles para efectuar los cálculos numéricos asociados al algoritmo. Para el caso de los algoritmos a implementar, la complejidad computacional de los algoritmos está dada por el número de multiplicaciones que se detalla en la tabla 3.

Tabla 3. Costo Computacional de los Algoritmos LCMV

Algoritmo	
NLMS	$(2p+3)N-(p^2+p-1)$
CG	$3N^2+(10-4p)N+p^2-8p+2$

Donde se determina que N representa el número de *taps* y p el número de restricciones. En la Tabla 4 se puede analizar que los algoritmos de menor costo computacional es NLMS y el de mayor costo computaciones es CG. La estructura que tiene el menor costo computacional es la *Householder*.

En la Tabla 4 se presenta un resumen del MSE para los diferentes números de bits utilizados para los algoritmos HNCLMS y HCCG, como se puede observar cuando el número de bits aumenta el MSE es menor para cada uno de los algoritmos, debido a que al ser menor el número de bits existe un incremento de las operaciones y por ende una propagación del error.

Tabla 4. MSE en dB para los Algoritmos HNCLMS y HCCG

ALGORITMO	8 Bits	12 Bits	16 Bits
HNCLMS	-26.0071	-29.6076	-29.6826
HCCG	-5.1213	-32.3649	-36.4347

CAPÍTULO IV

4. ESTRUCTURAS DE LATTICE

4.1. Introducción

El algoritmo RLS (*Recursive Least Squares*) que frente a otros algoritmos adaptativos presenta una tasa de convergencia más rápida y una mejor respuesta en presencia de perturbación, pero con la desventaja de que su implementación en hardware es bastante difícil debido a su alta complejidad computacional y problemas con la estabilidad numérica.

Teóricamente existen varios algoritmos que resuelven el problema de mínimos cuadrados de manera recursiva. En particular la estructura *Lattice* permite reducir la complejidad computacional (al orden de N). Como consecuencia, la estructura LRLS (*Lattice Recursive Least Squares*) se considera una implementación rápida para el problema del algoritmo RLS.

El algoritmo LRLS es una estructura en cascada basada en filtros de predicción lineal *forward* y *backward* que permiten describir las propiedades de la señal de entrada, basándose en el desarrollo de mínimos cuadrados para reducir el error cuadrático de los filtros mencionados, además actualiza los coeficientes de reflexión en función del tiempo (Diniz, 2013).

El rendimiento de los algoritmos LRLS cuando se implementan con aritmética de infinita precisión es idéntico a los de cualquier algoritmo RLS, pero con la aritmética de precisión finita cada algoritmo tiene una respuesta diferente.

En este capítulo, se investigara el algoritmo LRLS mediante la imposición de la restricción lineal *Householder* que se estudió en el Capítulo 2. Además se presentan dos formas del algoritmo LRLS. El primero se basa en un error *posteriori* y el segundo en un error *a priori*.

4.2. Predicción *Recursive Least-Squares*

La predicción de RLS *forward* y *backward* son esenciales para derivar las ecuaciones de *order-updating* inherentes a los algoritmos LRLS. Para ambos casos los resultados con obtenidos siguiendo el procedimiento de derivación del algoritmo RLS convencional, puesto que la única característica distintiva de los problemas de predicción es la definición de la señal de referencia $d[k]$. En el caso de predicción *forward* se tiene que $d[k] = x[k]$, mientras que el vector de señal de entrada tiene la muestra $x[k-1]$ como los datos más recientes. Para el caso de predicción *backward* $d[k] = x[k-i-1]$, donde el índice i define la muestra del pasado que se desea predecir y la señal de vector de entrada tiene $x[k]$ como los datos más recientes. (Diniz, 2013).

4.2.1. Problema de Predicción *forward*

El objetivo de la predicción *forward* es predecir una muestra futura dado una secuencia de entrada, usando la información actual de la secuencia disponible. Por ejemplo se puede tratar de predecir el valor $x[k]$ utilizando muestras pasadas como $x[k-1]\, x[k-2]$ a través de un filtro de predicción FIR con $i+1$ coeficientes como se muestra en la siguiente ecuación:

$$y_f[k, i+1] = \mathbf{w}_f^T[k, i+1]\mathbf{x}[k-1, i+1] \tag{17}$$

Donde $y_f[k, i+1]$ es la señal de predicción de salida.

$$\mathbf{w}_f[k, i+1] = \{\mathbf{w}_{fo}[k]\mathbf{w}_{f1}[k] \dots \mathbf{w}_{fi}[k]\}^T \tag{18}$$

Es el vector de coeficientes de predicción *forward* FIR.

$$\mathbf{x}[k-1, i+1] = \{x[k-1]x[k-2] \dots x[k-i-1]\}^T \tag{19}$$

Y $\mathbf{x}[k-1, i+1]$ es el vector de señal de entrada disponible. La segunda variable incluida en el vector de la Ecuación (17) es para indicar la dimensión del vector, puesto que se requieren en las ecuaciones de *order-updating* del algoritmo LRLS.

El error instantáneo posteriori de la predicción *forward* está dado por:

$$\varepsilon_f[k, i+1] = x[k] - \mathbf{w}_f^T[k, i+1]\mathbf{x}[k-1, i+1] \tag{20}$$

Para la formulación RLS del problema de predicción *forward*, se define el vector de error ponderado *forward prediction* como:

$$\boldsymbol{\varepsilon}_f[k, i+1] = \hat{\mathbf{x}}[k] - \mathbf{X}^T[k-1, i+1]\mathbf{w}_f[k, i+1] \tag{21}$$

Donde

$$\hat{\mathbf{x}}[k] = \left\{x[k]\lambda^{1/2}x[k-1]\lambda x[k-2] \ldots \lambda^{k/2}x[0]\right\}^T \tag{22}$$

$$\boldsymbol{\varepsilon}_f[k, i+1] = \left\{\varepsilon_f[k, i+1]\lambda^{1/2}\varepsilon_f[k-1, i+1]\lambda\varepsilon_f[k-2, i+1] \ldots \lambda^{k/2}\varepsilon_f[0, i+1]\right\}^T \tag{23}$$

$$\mathbf{X}[k-1, i+1] = \begin{bmatrix} x[k-1] & \lambda^{1/2}x[k-2] & \ldots & \lambda^{[k-2]/2}x[1] & \lambda^{[k-1]/2}x[0] & 0 \\ x[k-2] & \lambda^{1/2}x[k-3] & \ldots & \lambda^{[k-2]/2}x[0] & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x[k-i-1] & \lambda^{1/2}[k-i-2] & \ldots & 0 & 0 & 0 \end{bmatrix} \tag{24}$$

El vector error $\boldsymbol{\varepsilon}_f[k, i+1]$ se puede definir como:

$$\boldsymbol{\varepsilon}_f[k, i+1] = \mathbf{X}^T[k, i+2]\begin{Bmatrix} 1 \\ -\mathbf{w}_f[k, i+1] \end{Bmatrix} \tag{25}$$

La función objetivo a ser minimizado en el problema de predicción *forward* está dada por:

$$\xi_f^d[k,i+1] = \boldsymbol{\varepsilon}_f^T[k,i+1]\boldsymbol{\varepsilon}_f[k,i+1] \tag{26}$$

$$\xi_f^d[k,i+1] = \sum_{l=0}^{k} \lambda^{k-1}\, \boldsymbol{\varepsilon}_f^2[l,i+1] \tag{27}$$

$$\xi_f^d[k,i+1] = \sum_{l=0}^{k} \lambda^{k-1} \left\{x[l] - \mathbf{x}^T[l-1,i+1]\mathbf{w}_f[k,i+1]\right\}^2 \tag{28}$$

La solución óptima para el vector de coeficientes es:

$$\mathbf{w}_f[k,i+1] = \mathbf{R}_{Df}^{-1}[k-1,i+1]\mathbf{p}_{Df}[k,i+1] \tag{29}$$

Donde $\mathbf{R}_{Df}[k-1,i+1]$ es igual a la matriz correlación determinística $\mathbf{R}_D[k-1]$ de orden $i+1$ y $\mathbf{p}_{Df}[k,i+1]$ es el vector determinista de correlación cruzada entre $x[l]$ y $\mathbf{x}[l-1,i+1]$.

El valor mínimo de $\xi_f^d[k]$ esta dada por:

$$\xi_{fmin}^d[k,i+1] = \sigma_f^2[k] - \mathbf{w}_f^T[k,i+1]\mathbf{p}_{Df}[k,i+1] \tag{30}$$

Combinando las Ecuaciones 29 y 30 se obtiene:

$$\mathbf{R}_D[k,i+2]\begin{bmatrix} 1 \\ -\mathbf{w}_f[k,i+1] \end{bmatrix} = \begin{bmatrix} \xi_{fmin}^d[k,i+1] \\ 0 \end{bmatrix} \tag{31}$$

Donde $\mathbf{R}_D[k,i+2]$ es igual a $\mathbf{R}_D[k]$de dimensiones $i+2$. La ecuación anterior se refiere a la matriz determinista de correlación de orden $i+2$.

4.2.2. Problema de predicción *backward*

El objetivo de la predicción *backward* es generar una estimación de una muestra pasada de una secuencia de entrada dada, utilizando la información disponible actual de la secuencia.

$$y_b[k, i+1] = \mathbf{w}_b^T[k, i+1]\mathbf{x}[k, i+1] \tag{32}$$

Donde $y_b[k, i+1]$ es la señal de salida de predicción *backward*, y

$$\mathbf{w}_b^T[k, i+1] = \{w_{b0}[k] w_{b1}[k] \dots w_{bi}[k]\}^T \tag{33}$$

Es el vector de coeficientes de predicción *backward* FIR. El error de predicción instantáneo posteriori *backward* está dado por:

$$\varepsilon_b[k, i+1] = x[k-i-1] - \mathbf{w}_b^T[k, i+1]\mathbf{x}[k, i+1] \tag{34}$$

El vector de error *backward* ponderado está dado por

$$\boldsymbol{\varepsilon}_b[k, i+1] = \hat{\mathbf{x}}[k] - \mathbf{X}^T[k-1, i+1]\mathbf{w}_b[k, i+1] \tag{35}$$

Donde

$$\hat{\mathbf{x}}[k-i-1] = \left\{x[k-i-1]\lambda^{1/2}x[k-i-2] \dots \lambda^{[k-i-1]/2}x[0]0 \dots 0\right\}^T \tag{36}$$

$$\boldsymbol{\varepsilon}_b[k, i+1] = \left\{\varepsilon_b[k, i+1]\lambda^{1/2}\varepsilon_b[k-1, i+1] \dots \lambda^{k/2}\varepsilon_b[0, i+1]\right\}^T \tag{37}$$

$$\mathbf{X}[k, i+1] = \begin{bmatrix} x[k] & \lambda^{1/2}x[k-1] & \dots & \lambda^{[k-1]/2}x[1] & \lambda^{[k]/2}x[0] \\ x[k-1] & \lambda^{1/2}x[k-2] & \dots & \lambda^{[k-2]/2}x[0] & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x[k-i] & \lambda^{1/2}[k-i-1] & \dots & 0 & 0 \end{bmatrix} \tag{38}$$

El vector error $\boldsymbol{\varepsilon}_b[k, i+1]$ se puede definir como:

$$\boldsymbol{\varepsilon}_b[k, i+1] = \mathbf{X}^T[k, i+2]\begin{Bmatrix} -\mathbf{w}_b[k, i+1] \\ 1 \end{Bmatrix} \tag{39}$$

La función objetivo a ser minimizado en el problema de predicción *backward* está dada por:

$$\xi_b^d[k, i+1] = \sum_{l=0}^{k} \lambda^{k-1} \{x[l-i-1] - \mathbf{x}^T[l, i+1]\mathbf{w}_b[k, i+1]\}^2 \tag{40}$$

La solución óptima para el vector de coeficientes es:

$$\mathbf{w}_b[k, i+1] = \mathbf{R}_{Df}^{-1}[k, i+1]\mathbf{p}_{Df}[k, i+1] \tag{41}$$

Donde $\mathbf{R}_{Db}[k, i+1]$ es igual a la matriz correlación determinística $\mathbf{R}_D[k]$ de orden $i+1$ y $\mathbf{p}_{Db}[k, i+1]$ es el vector determinista de correlación cruzada entre $x[l-i-1]$ y $\mathbf{x}[l, i+1]$.

El valor mínimo de $\xi_b^d[k]$esta dada por:

$$\xi_{bmin}^d[k, i+1] = \sigma_b^2[k] - \mathbf{w}_b^T[k, i+1]\mathbf{p}_{Db}[k, i+1] \tag{42}$$

Combinando las Ecuaciones 45 y 46, se obtiene

$$\mathbf{R}_D[k, i+2]\begin{bmatrix} -\mathbf{w}_b[k, i+1] \\ 1 \end{bmatrix} = \begin{bmatrix} o \\ \xi_{bmin}^d[k, i+1] \end{bmatrix} \tag{43}$$

Donde $\mathbf{R}_D[k, i+2]$ es igual a $\mathbf{R}_D[k]$de dimensiones $i+2$. La ecuación anterior se refiere a la matriz determinista de correlación de orden $i+1$.

4.2.3. Ecuaciones de Actualización

En esta sección se derivara las ecuaciones de orden de actualización del error de predicción para el *forward* y *backward*, teniendo como nuevo parámetro $\delta_f(k,i)$ que se interpreta como el determinística de correlación cruzada entre el error de predicción hacia adelante $\varepsilon_f(l,i)$ con los coeficientes fijado en $w_f(l,i)$ y la señal deseada del filtro de predicción *backward* $x(l-i-1)$

$$\delta_f(k,i) = \sum_{l=0}^{k} \lambda^{k-1}\, \varepsilon_f(l,i) x(l-i-1) \tag{44}$$

$\delta_b(k,i)$ es interpretado como el determinística de correlación cruzada entre el error de predicción *backward* $\varepsilon_b(l-1,i)$ y la señal deseada del filtro de predicción forward $x(l)$.

$$\delta_b(k,i) = \sum_{l=0}^{k} \lambda^{k-1}\, \varepsilon_b(l-1,i) x(l) \tag{45}$$

Se define que el determinística de correlación cruzada entre $\varepsilon_f(l,i)$ y $x(l-i-1)$ y entre $\varepsilon_b(l-1,i)$ y $x(l)$ son iguales, por tanto:

$$\delta_f(k,i) = \delta_b(k,i) = \delta(k,i) \tag{46}$$

4.2.4. Orden de Actualización de $\boldsymbol{\varepsilon^d_{b_{min}}(k,i)}$ y $\boldsymbol{w_b(k,i)}$

El error LS mínimo del orden de actualización y los coeficientes de predicción hacia atrás se deducen de la multiplicación por el escalar $\delta(k,i)/\varepsilon^d_{fmin}(k,i)$, obteniendo:

$$\varepsilon^d_{bmin}(k,i+1) = \varepsilon^d_{bmin}(k-1,i) - \frac{\delta^2}{\varepsilon^d_{fmin}(k,i)} \tag{47}$$

$$w_b(k,i+1) = \begin{bmatrix} 0 \\ w_b(k-1,i) \end{bmatrix} - \frac{\delta(k,i)}{\varepsilon^d_{fmin}(k,i)} \begin{bmatrix} -1 \\ w_f(k,i) \end{bmatrix} \tag{48}$$

4.2.5. Orden de Actualización de $\boldsymbol{\varepsilon^d_{f_{min}}(k,i)}$ y $\boldsymbol{w_f(k,i)}$

Similarmente, se deduce a la multiplicación del escalar$\delta(k,i)/\varepsilon^d_{bmin}(k-1,i)$ obteniendo:

$$\varepsilon^d_{fmin}(k,i+1) = \varepsilon^d_{fmin}(k,i) - \frac{\delta^2(k,i)}{\varepsilon^d_{bmin}(k-1,i)} \tag{49}$$

$$w_f(k,i+1) = \begin{bmatrix} w_f(k,i) \\ 0 \end{bmatrix} - \frac{\delta(k,i)}{\varepsilon^d_{bmin}(k-1,i)} \begin{bmatrix} w_b(k-1,i) \\ -1 \end{bmatrix} \tag{50}$$

4.2.6. Orden de Actualización del Error de Predicción

El orden de actualización del error de predicción posterior *forward* y *backward* se describe a continuación:

$$\varepsilon_f(k,i+1) = \varepsilon_f(k,i) - k_f(k,i)\varepsilon_b(k-1,i) \tag{51}$$

Donde $k_f(k,i) = \frac{\delta(k,i)}{\varepsilon^d_{bmin}(k-1,i)}$, llamado el coeficiente de reflexión *forward*.

$$\varepsilon_b(k,i+1) = \varepsilon_b(k-1,i) - k_b(k,i)\varepsilon_f(k,i) \tag{52}$$

Donde $k_f(k,i) = \frac{\delta(k,i)}{\varepsilon^d_{fmin}(k-1,i)}$, llamado el coeficiente de reflexión *backward*

4.2.7. Ecuaciones *Time-Updating*

Se requieren las ecuaciones *time-updating* para los nuevos datos de entrada disponible, definidas a continuación:

4.2.7.1. *Time-Updating* para los coeficientes de predicción

$$w_f(k,i) = w_f(k-1,i) + \phi(k-1,i)e_f(k,i) \quad (53)$$

Dónde $e_f(k,i)$ es el error de predicción a *priori forward* del orden $i-1$ (El filtro predictor es de orden $i-1$ mientras que el predictor incluyendo la señal deseada es de orden i).

$$\phi(k-1,i)=\frac{\psi(k-1,i)}{\lambda+\psi^T(k-1,i)x(k-1,i)} \quad (54)$$

$$\psi(k-1,i) = S_D(k-2,i)x(k-1,i) \quad (55)$$

$$w_b(k,i) = w_b(k-1,i) + \phi(k,i)e_b(k,i) \quad (56)$$

Donde $e_b(k,i)$ es el error de predicción a *priori backward* del orden $i-1$.

$$\phi(k-1,i)=\frac{\psi(k-1,i)}{\lambda+\psi^T(k-1,i)x(k-1,i)} \quad (57)$$

$$\psi(k-1,i) = S_D(k-2,i)x(k-1,i) \quad (58)$$

4.2.7.2. *Time Updating* para $\delta(k,i)$

Desde el punto de vista computacional, sería interesante calcular el error de predicción sin utilizar los coeficientes predictores, porque trabajar con estos coeficientes requiere el uso de productos internos. Con el objetivo de lograr esto, una se deriva $\delta(k,i)$ dando como resultado un subproducto $\gamma(k,i)$, que es un factor de conversión entre el error priori y posteriori.

$$\delta(k,i) = \lambda\delta(k-1,i) + \gamma(k-1,i)e_b(k-1,i)e_f(k,i) \quad (59)$$

Dónde:

$$\gamma(k-1,i) = 1 - \phi^T(k-1,i)x(k-1,i) \tag{60}$$

4.2.7.3. *Time Updating* para $\gamma(k,i)$

A partir de la ecuación 60 se deduce la ecuación para $\gamma(k-1,i)$ qué es la relación requerida para resolver los problemas de predicción *backward* y *forward.*

$$\gamma(k-1,i) = \gamma(k-1,i-1) - \frac{\varepsilon_b^2(k-1,i-1)}{\xi_{bmin}^d(k-1,i-1)} \tag{61}$$

4.2.8. Algoritmo LRLS con Restricción

Los algoritmo adaptativos demandan un conjunto de restricciones que se deben ejecutar en cada interacción, mediante la imposición de restricciones lineales sobre el filtro; tal es el caso de estudio de las restricciones para el algoritmo *Lattice Recursive Least Squares*. Al abordar la investigación sobre el tema, no se encuentra información al mismo.

Tabla 5. Algoritmo *Lattice* RLS Basado en Error Posteriori

Inicialización:

$\mathbf{Do\ for}\ \boldsymbol{i = 0, 1 \ldots, N}$

$\boldsymbol{\delta[-1, i] = \delta_D[-1, i] = 0}$ (asumiendo x(k)=0 for k<0)

$\boldsymbol{\xi^d_{bmin}(-1, i) = \xi^d_{bmin}(-1, i) = \epsilon}$ (una pequeña constante positiva)

$\boldsymbol{\gamma(-1, i) = 1}$

$\boldsymbol{\varepsilon_b(-1, i) = 0}$

End

Do for k≥0

$\boldsymbol{\gamma(k, 0) = 1}$

$\boldsymbol{\varepsilon_b(k, 0) = \varepsilon_f(k, 0) = x(k)}$

$\boldsymbol{\xi^d_{bmin}(k, 0) = \xi^d_{fmin}(k, 0) = x^2(k) + \lambda\xi^d_{fmin}(k - 1, 0)}$

$\boldsymbol{\varepsilon(k, 0) = d(k)}$

Do for $\boldsymbol{i = 0, 1 \ldots, N}$

$\boldsymbol{\delta(k, i) = \lambda\delta(k - 1, i) + \frac{\varepsilon_b(k-1,i)}{\gamma(k-1,i)}\varepsilon_f(k, i)}$

$\boldsymbol{\gamma(k, i + 1) = \gamma(k, i) - \frac{\varepsilon_b^2(k,i)}{\xi^d_{bmin}(k,i)}}$

$\boldsymbol{k_b(k, i) = \frac{\delta(k,i)}{\xi^d_{fmin}(k,i)}}$

$\boldsymbol{k_b(k, i) = \frac{\delta(k,i)}{\xi^d_{fmin}(k,i)}}$

$\boldsymbol{\varepsilon_b(k, i + 1) = \varepsilon_b(k - 1, i) - k_b(k, i)\varepsilon_f(k, i)}$

$\boldsymbol{\varepsilon_f(k, i + 1) = \varepsilon_f(k, i) - k_f(k, i)\varepsilon_b(k - 1, i)}$

$\boldsymbol{\xi^d_{bmin}(k, i + 1) = \xi^d_{bmin}(k - 1, i) - \delta(k, i)k_b(k, i)}$

$\boldsymbol{\xi^d_{fmin}(k, i + 1) = \xi^d_{fmin}(k, i) - \delta(k, i)k_f(k, i)}$

Feed forward filtering

$\boldsymbol{\delta_D(k, i) = \lambda\delta_D(k - 1, i) + \frac{\varepsilon_b(k,i)}{\gamma(k,i)}\varepsilon(k, i)}$

$\boldsymbol{v_i(k) = \frac{\delta_D(k,i)}{\xi^d_{bmin}(k,i)}}$

$\boldsymbol{\varepsilon(k, i + 1) = \varepsilon(k, i) - v_i(k)\varepsilon_b(k, i)}$

End

End

Tabla 6. Algoritmo *Lattice* RLS Basado en Error Priori

Inicialización:

Do for $\boldsymbol{i = 0, 1 \ldots, N}$

$\boldsymbol{\delta(-1, i) = \delta_D(-1, i) = 0}$ (asumiendo x(k)=0 for k<0)

$\boldsymbol{\gamma(-1, i) = 1}$

$\boldsymbol{\xi^d_{bmin}(-1, i) = \xi^d_{bmin}(-1, i) = \epsilon}$ (una pequeña constante positiva)

$\boldsymbol{e_b(-1, i) = 0}$

$\boldsymbol{k_f(-1, i) = k_b(-1, i)} = 0$

End

Do for k≥0

$\boldsymbol{\gamma(k, 0) = 1}$

$\boldsymbol{e_b(k, 0) = e_f(k, 0) = x(k)}$

$\boldsymbol{\xi^d_{bmin}(k, 0) = \xi^d_{fmin}(k, 0) = x^2(k) + \lambda \xi^d_{fmin}(k - 1, 0)}$

$\boldsymbol{e(k, 0) = d(k)}$

Do for $\boldsymbol{i = 0, 1 \ldots, N}$

$\boldsymbol{\delta(k, i) = \lambda \delta(k - 1, i) + \gamma(k - 1, i) e_b(k - 1, i) e_f(k, i)}$

$\boldsymbol{\gamma(k, i + 1) = \gamma(k, i) - \frac{\gamma^2(k,i) e_b^2(k,i)}{\xi^d_{bmin}(k,i)}}$

$\boldsymbol{e_b(k, i + 1) = e_b(k - 1, i) - k_b(k - 1, i) e_f(k, i)}$

$\boldsymbol{e_f(k, i + 1) = e_f(k, i) - k_f(k - 1, i) e_b(k - 1, i)}$

$\boldsymbol{k_f(k, i) = \frac{\delta(k,i)}{\xi^d_{bmin}(k-1,i)}}$

$\boldsymbol{k_b(k, i) = \frac{\delta(k,i)}{\xi^d_{fmin}(k,i)}}$

$\boldsymbol{\xi^d_{bmin}(k, i + 1) = \xi^d_{bmin}(k - 1, i) - \delta(k, i) k_b(k, i)}$

$\boldsymbol{\xi^d_{fmin}(k, i + 1) = \xi^d_{fmin}(k, i) - \delta(k, i) k_f(k, i)}$

Feedforward filtering

$\boldsymbol{\delta_D(k, i) = \lambda \delta_D(k - 1, i) + \gamma(k, i) e_b(k, i) e(k, i)}$

$\boldsymbol{e(k, i + 1) = e(k, i) - v_i(k - 1) e_b(k, i)}$

$\boldsymbol{v_i(k) = \frac{\delta_D(k,i)}{\xi^d_{bmin}(k,i)}}$

End

End

Donde $\boldsymbol{e_f(k,i)}$ que representa el error instantáneo de predicción a posteriori hacia adelante, $\boldsymbol{e_b(k,i)}$ que representa el error instantáneo de predicción a posteriori hacia atrás, $\boldsymbol{\xi^d_{fmin}}(k,i)$ y $\boldsymbol{\xi^d_{bmin}}(k,i)$ son el mínimo en el *least-square* para cada error de predicción hacia adelante y hacia atrás respectivamente. Los coeficientes $\boldsymbol{k_f(k,i)}$ y $\boldsymbol{k_b(k,i)}$ son los coeficientes de reflexión hacia adelante y hacia atrás. $\boldsymbol{\gamma(k,i)}$ es el factor de conversión entre el error a priori y posteriori. $\boldsymbol{v_i(k)}$ son los coeficientes multiplicadores de alimentación hacia adelante.

CAPITULO V

5. IMPLEMENTACION DE LOS ALGORITMO

5.1. Respuesta de los Micrófonos

Un micrófono sirve como un sensor acústico para registrar señales de audio y monitorear nivel acústico. Para el presente proyecto se utiliza los micrófonos MEMS ADMP504.

5.1.1. Características técnicas de los Micrófonos MEMS ADMP504

El ADMP504 consta de un elemento de micrófono MEMS, un convertidor de impedancia y un amplificador de salida. La especificación de la sensibilidad hace que sea una excelente opción tanto para aplicaciones de campo cercano y campo lejano.

El ADMP504 tiene muy alta relación señal/ruido y la respuesta de frecuencia de banda ancha extendida, lo que resulta en un sonido natural con una alta inteligibilidad.

Figura 9. Micrófono MEMS ADMP504

5.1.1.1.Características físicas:

- Tamaño 3.35mm.
- Respuesta Omnidireccional.
- Relación señal/ruido 65dBA.
- Sensibilidad de 38dBV.
- Frecuencia de respuesta 100Hz a 20KHZ.
- Salida analógica individual.

5.1.1.2.Diagrama de Bloques Funcional

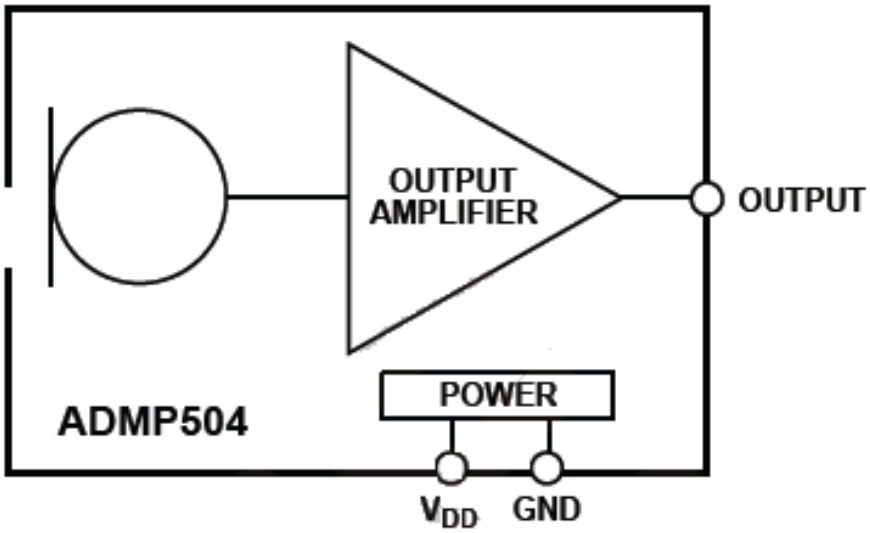


Figura 10. Diagrama de Bloque Funcional del Micrófono MEMS ADMP504

5.1.1.3.Especificaciones Técnicas

Tabla 7. Especificaciones Técnicas de los Micrófonos MENS ADM504

Parámetros	Símbolo	Condiciones de Pruebas	Min	Tipo	Max	Unit
PERFORMANCE						
Direccionalidad				Omni		
Sensibilidad		1Khz, 94dB SPL	-41	-38	-35	*dBV*
Señal/Ruido	SNR	20Hz to 20KHz		65		*dBA*
Ruido de Entrada	EIN	20Hz to 20KHz		29		*dBA*
Rango Dinámico		Entrada maxima acústica		91		*dB*
Frecuencia de Respuesta		Baja −3 *dB* Alta −3 *dB*		100 >20		*Hz* *kHz*
Power Supply Rejection	PSR	217 *Hz*, 100 *mV* p-p Onda cuadrada		-70		*dBV*
Maxima Entrada Acústica		Pico 10% *THD*		120		*dB*
POWER SUPPLY						
Tensión de Alimentación	V_{DD}		1.6		3.3	*V*
Tensión de Corriente	I_S	V_{DD} =1.8V V_{DD} =3.3V		180 200	200 225	*µA* *µA*
CARACTERÍSTICAS DE SALIDA						
Impedancia de Salida	Z_{OUT}			200		Ω
Output DC Offset				0.8		*V*
Output Current Limit				90		*µA*
Máximo Voltaje de Salida		120 *dB*		0.25		*V*
Ruido Pico		20 *Hz* a 20 *kHz*		-103		*dBV*

Tabla 8. Indice Máximo Absoluto de los Micrófonos MENS ADM504

Parámetros	Índice
Tensión de alimentación	-0.3V a +3.6V
Nivel de presión sonora (SPL)	160*Db*
Choque mecánico	10,000*g*
Rango de temperatura	-40°C to +85°C

Tabla 9. Descripción de la Funcionalidad de los PIN de los Micrófonos ADM504

Pin N°	Nombre	Descripción
1	*OUTPUT*	Señal análoga de salida
2	*GND*	Tierra
3	V_{DD}	Tensión de alimentación

5.1.2. Adquisición de Datos para análisis del Comportamiento de los Micrófonos MEMS ADMP504

Se analiza la respuesta de los micrófonos utilizados en el presente proyecto; debido a que la teoría de sensores indica que los elementos que conforman el arreglo de sensores deben tener un comportamiento isotrópico.

Para el análisis se realizó un arreglo ULA que cumple con el teorema de Nyquist, donde la frecuencia de muestreo es mayor o igual al doble de la máxima componente de frecuencia de la señal y de esta manera evitar *aliasing* (solapamiento) en frecuencia. El solapamiento se da cuando la separación entre los micrófonos no es la correcta, ocasionando que los micrófonos no puedan diferenciar las señales que arriban de diferentes direcciones.

En base al criterio de Nyquist estudiado se utilizara 4 micrófonos equidistantes entre ellos por $0{,}1m$ (Caisapanta, 2015) frente a una fuente que emite señales AWGN (*Additive White Gaussian Noise*) que por sus características matemáticas su espectro abarca todas las frecuencias (Papoulis & Pillai , 2002) distanciada a $1\,m$

con relación al arreglo de micrófonos y 90° de un micrófono a cual nominaremos M4, como se muestra en la Figura 11.

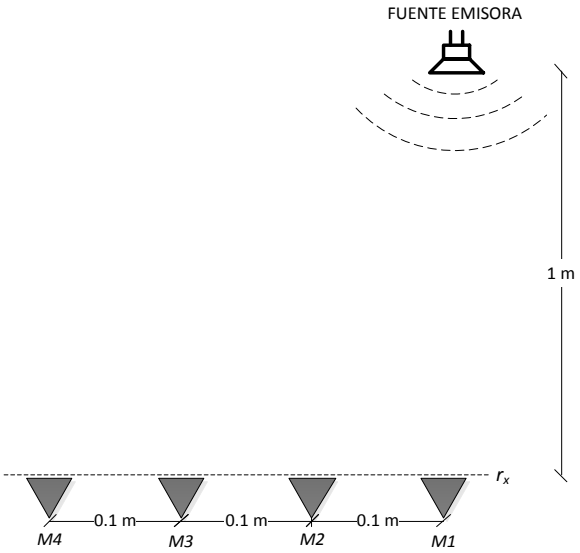


Figura 11. Arreglo ULA de micrófonos

Para la adquisición de datos, se utiliza la tarjeta NI myRIO 1900 con su interfaz en LabVIEW, estos datos serán exportados a una base de datos, para ser procesados en Matlab®, obteniendo la respuesta en frecuencia y verificar el comportamiento de los mismos.

5.1.3. Especificaciones Técnicas de la Tarjeta NI myRIO 1900

Posee un procesador *ARM® Cortex™-A9 dual core* de rendimiento en tiempo real y E/S personalizada, aprovechando la configuración FPGA predeterminados, la cual pueden personalizar conforme los proyectos, por medio de sus componentes internos, acceso a software transparente y biblioteca de recursos. Programable con *LabVIEW,* C o adaptable para diferentes niveles de programación.

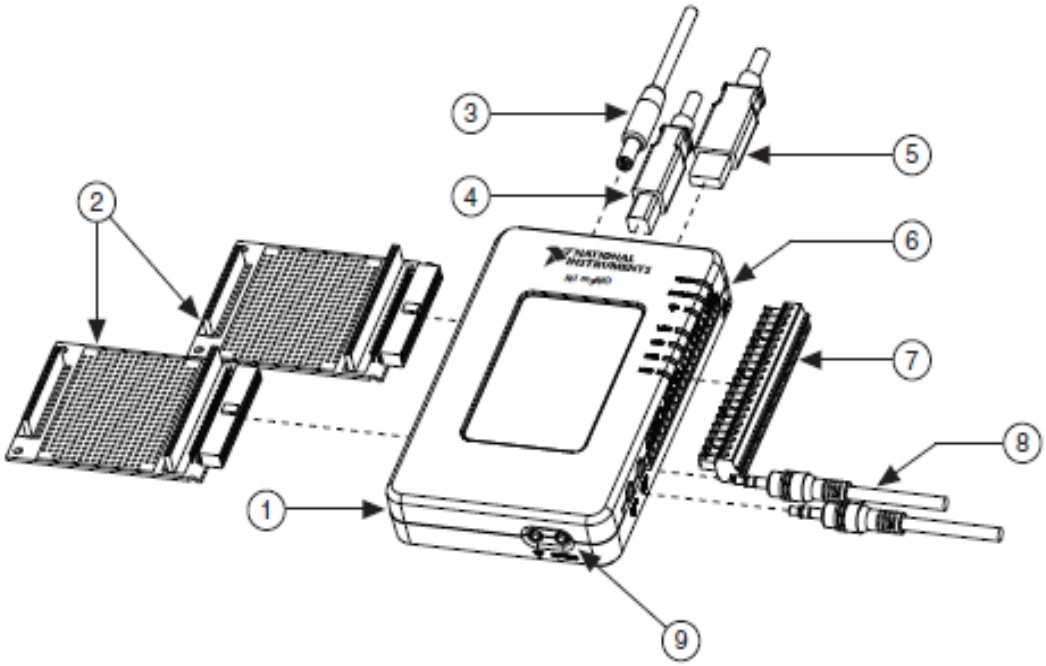


Figura 12. Micrófono MEMS ADMP504

Tabla 10. Partes NI myRIO-1900

1 NI myRIO-1900
2 myRIO *Expansion Port* (MXP)
3 *Power Input Cable*
4 USB *Device cable*
5 USB *Host Cable (Not Included in Kit)*
6 LEDs
7 *Mini System Port* (MSP)
8 *Audio In/Out Cables*
9 *Button 0*

5.1.3.1.Diagrama de Bloques del Hardware NI myRIO-1900

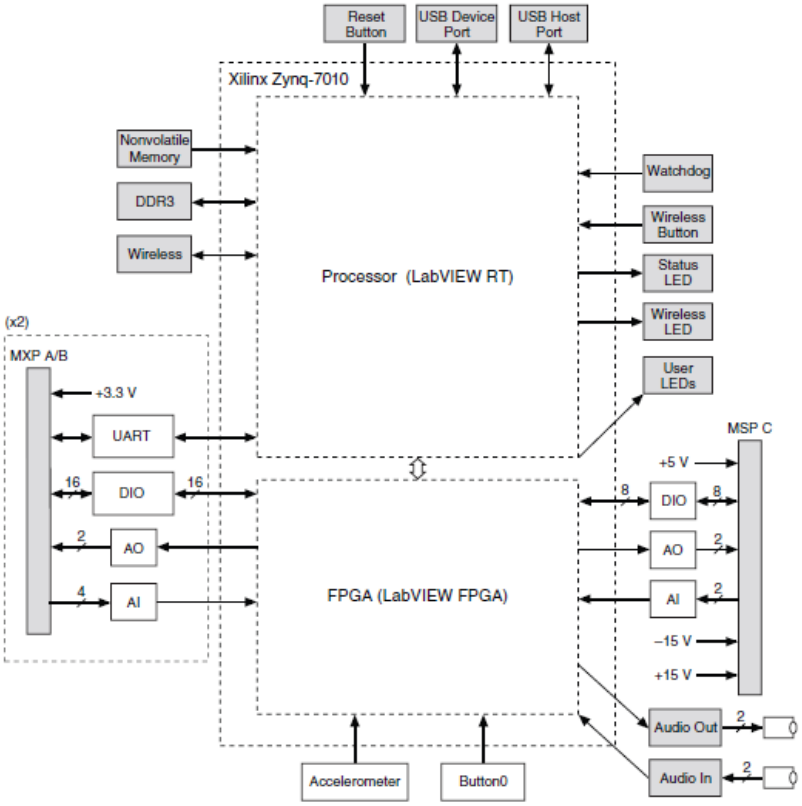


Figura 13. Tarjeta NI myRIO-1900

En la siguiente Figura 14 se muestran las señales de los conectores MXP A y B.

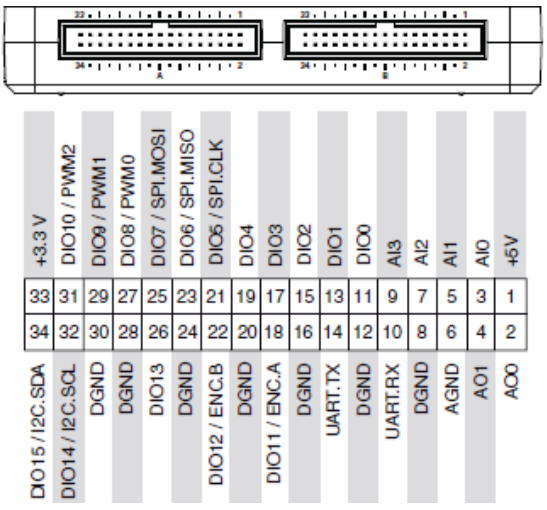


Figura 14. Conectores Primarios/Secundarios MXP

Tabla 11. Descripción de los Conectores Primarios/Secundarios MXP

Señal	Referencia	Dirección	Descripción
+5V	*DGND*	Salida	+5V, fuente de poder de salida.
AI <0..3>	*AGND*	Entrada	0-5 V, entrada del canal análogo.
AO<0..1>	*AGND*	Salida	0-5 V, salida análoga.
AGND	N/A	N/A	Referencia para la de señal análoga de entrada y salida.
+3.3 V	*DGND*	Salida	+3.3 V, fuente de poder de salida.
DIO <0..15>	*DGND*	Entrada o Salida	Líneas digitales de 3.3 V salida, 3.3 V/5 V-entrada.
UART.RX	*DGND*	Entrada	UART recibir de entrada. Las líneas UART son eléctricamente idéntica a las líneas DIO.
UART.TX	*DGND*	Salida	UART transmite la salida. Las líneas UART son eléctricamente idéntica a las líneas DIO.
DGND	N/A	N/A	Referencias para señales digitales +5V y +3.3 V.

La Figura 15 se describe los conectores del *Mini System Port* (MSP).

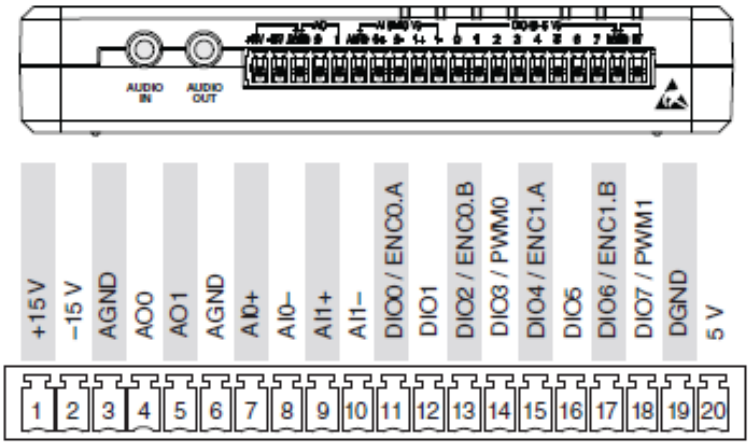


Figura 15. Conectores C Primarios/Secundarios MSP

Tabla 12. Conectores C Primarios/Secundarios MSP

Señal	Referencia	Dirección	Descripción
+15V/-15V	*DGND*	Salida	+15V/-15V Fuente de poder de salida.
AI0+/AI0-; AI1+/AIA-	*AGND*	Entrada	±10 V, Diferencial entrada del canal análogo.
AO<0..1>	AGND	Salida	±10 V, referencia, salida análoga.
AGND	N/A	N/A	Referencia para señal análoga entrada y salida y +15 V/ -15 V poder salida.
+5V	*DGND*	Salida	+5V Fuente de poder de salida
DIO <0..7>	*DGND*	Entrada o salida	Líneas digitales de 3.3 V salida, 3.3 V/5 V-entrada.
UART.RX	*DGND*	Entrada	UART recibir de entrada. Las líneas UART son eléctricamente idéntica a las líneas DIO.
DGND	N/A	N/A	Referencias para señales digitales +5V y poder de salida.

5.1.3.2.Especificaciones Técnicas

En la siguiente tabla se detalla las características técnicas de la tarjeta NI my RIO1900.

Tabla 13. Especificaciones Técnicas de la Tarjeta NI myRIO 1900

Procesador	
Tipo de procesador	Xilinx Z-710
Velocidad del procesador	667 MHz
Núcleos del procesador	2
Memoria	
Memoria NO volátil	256 MB
DDR3 memoria	512 MB
DDR3 frecuencia de reloj	533 MHz
DDR3 Ancho de bus de datos	16 Bits
FPGA	
Tipo FPGA	Xilinz Z-7010
Características Wireless	
Protocolos	IEEE 802.11 b,g,n

Continúa

Banda de frecuencia	ISM 2.4 GHz
Ancho de banda	20 MHz
Canales	USA 1-11
TX Power	+10dBm max (10mW)
Rango de salida	Up to 150 m
Directividad de la antena	Omnidireccional
Seguridad	WPA, WPA2, WPA-Enterprise.
Entrada Análoga	
Tasa de la muestra global	500 kS/s
Resolución	12 bits
Protección al sobre voltaje	±16V
MXP Conectores	
Configuración	Cuatro conectores de terminación única por conector.
Impedancia de entrada	>500 kΩ at 500 kS/s 1 MΩ powered idle 4.7 kΩ powered off
Impedancia de la fuente recomendada	3 kΩ or less
Rango nominal	0V a +5V
Precisión absoluta	±50Mv
Ancho de banda	>300 kHz
MSP Conectores	
Configuración	Dos diferentes canales
Impedancia de entrada	Up to 100 nA 4.7 kΩ powered off
Rango nominal	±10V
Precisión absoluta	±200mV
Ancho de banda	20 kHz minimum, >50 kHz typical
Entrada de Audio	
Configuración	Una entrada estéreo con dos acoplados –AC
Impedancia de entrada	10 kΩ at DC
Rango nominal	±2.5V
Ancho de banda	2 kHz a >20 kHz
Salida Análoga	
Tasas de actualización máxima agregada	Todos los canales AO con conectores MXP 345kS/s. Todos los canales AO con conectores MSP y audio 345kS/s.
Resolución	12 bits
Protección de sobra carga	±16 V
MXP Conectores	
Configuración	Dos canales

Continúa

Rango	0 V a +5 V
Precisión absoluta	50 mV
Unidad Actual	3 Ma
Velocidad de subida	0.3V/us
MSP Conectores	
Configuración	Dos canales
Rango	±10 V
Precisión absoluta	±200 mV
Unidad Actual	2 mA
Velocidad de subida	2V/us
Digital I/O	
Número de Entradas	2 Conectores MXP de 16 líneas. Una UART.RX y una UART.TX por conector. 1 Conector MSP de 8 puertos
Control de Dirección	Cada línea DIO programable individualmente como entrada o salida.
Nivel lógico	5 V entrada; 3.3V salida
Nivel lógico de Entrada	*Input low voltage*, VIL 0 V min; 0.8 V max. *Input high voltage*, VIH 2.0 V min; 5.25 V max.
Nivel lógico de Salida	*Output high voltage,* VOH *sourcing* 4 mA, 2.4 Vmin; 3.46Vmax. *Output low voltage*, VOL *sinking* 4 mA, 0 V min; 0.4 V max.
Tiempo mínimo de Pulso	20 ns

5.1.4. Especificaciones Software Informático LabVIEW

LabVIEW constituye un revolucionario sistema de programación gráfica para aplicaciones que involucran adquisición, control y presentación de datos. Es un entorno de programación destinado al desarrollo de aplicaciones, empleando programación gráfica basada en diagramas de bloques. Este poderoso software presenta las siguientes ventajas:

- Se reduce el tiempo de desarrollo de las aplicaciones al menos de 4 a 10 veces.
- Dota de gran flexibilidad al sistema, permitiendo cambios y actualizaciones tanto del hardware como del software.
- Con un único sistema de desarrollo se integran las funciones de adquisición, análisis y presentación de datos, además está dotado de un compilador gráfico para lograr la máxima velocidad de ejecución posible.
- Tiene la posibilidad de incorporar aplicaciones escritas en otros lenguajes.

Los programas desarrollados mediante LabVIEW se denominan Instrumentos Virtuales (VIs), porque su apariencia y funcionamiento imitan los de un instrumento real. Sin embargo son análogos a las funciones creadas con los lenguajes de programación convencionales. Los VIs tienen una parte interactiva con el usuario y otra parte de código fuente, y aceptan parámetros procedentes de otros VIs.
Todos los VIs tienen un panel frontal y un diagrama de bloques. Las paletas contienen las opciones que se emplean para crear y modificar los VIs.

5.1.5. Construcción del Circuito Interfaz de los Micrófonos MEMS

Se realiza la construcción del circuito amplificador para los micrófonos como se observa en la Figura 16, donde la señal de salida del micrófono se conectara directamente a las entradas analógicas MXP de la tarjeta NI myRIO.

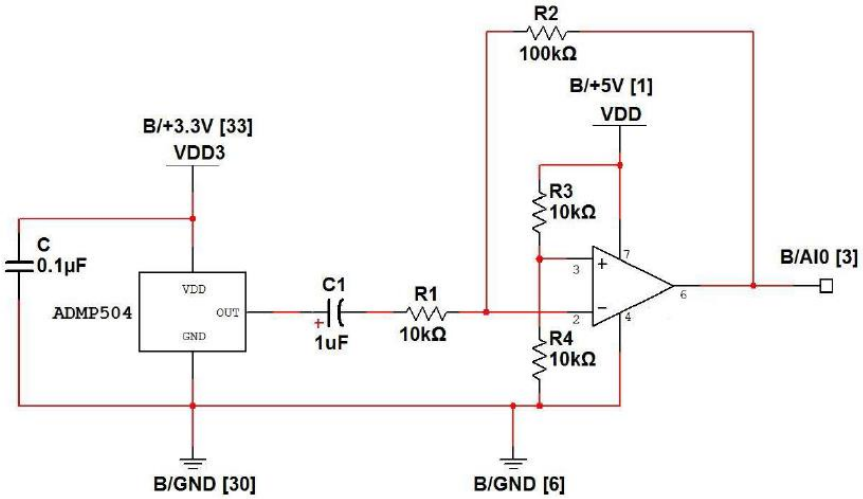


Figura 16. Circuito para Micrófono MEMS con Entrada Analógica (AI).

El circuito requiriere cinco conexiones con la tarjeta NI myRIO que se detallan a continuación:

1. Fuente de alimentación de $+5volts \rightarrow B/+5V$ (pin 1).
2. Tierra → B/GND (pin 6).
3. Fuente de alimentación de $+3.3volts \rightarrow B/+3.3V$ (pin 33).
4. Tierra → B/GND (pin 30).
5. Salida de los amplificadores

   $MIC\ 4 \rightarrow B/AIO$ (pin 3).

   $MIC\ 3 \rightarrow B/AIO$ (pin 5).

   MIC 2 → B/AIO (pin 7).

   MIC 1 → B/AIO (pin 9).

5.1.6. Interfaz con NI LabVIEW

La adquisición de datos se realizara desde la tarjeta FPGA.

Se genera un archivo FPGA.vi donde se crea las entradas análogas para los 4 micrófonos que se encuentran conectados a los conectores análogos MPX como se observa en la Figura 17.

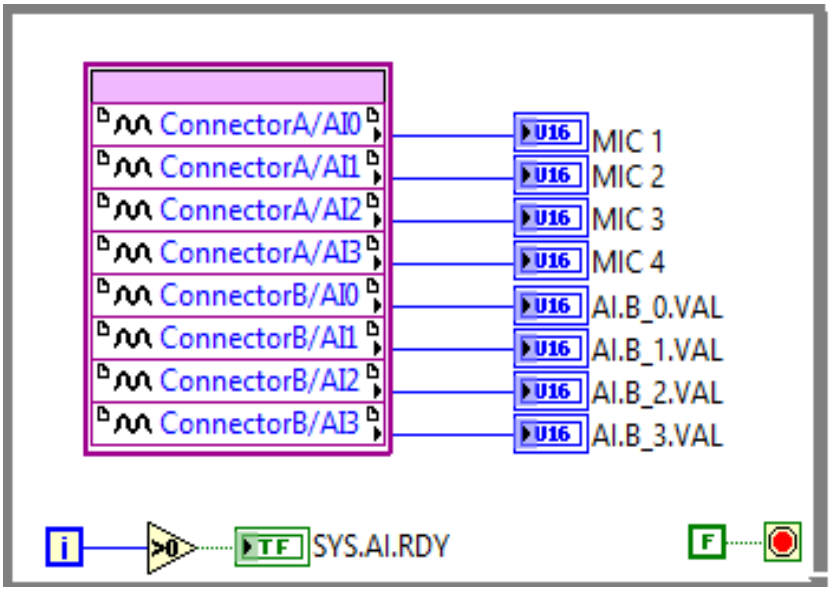


Figura 17. Entradas análogas para adquisición de datos en el FPGA

La adquisición de los datos debe ser ejecutado a 8000 *Hz*, que no es más que el proceso de codificación de la voz, se utiliza el icono *Timed Loop* que ejecuta cada iteración del bucle en el período que se especifique en este caso 125 us.

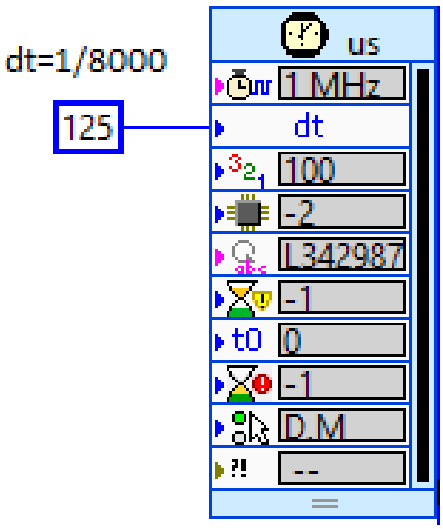


Figura 18. ***Timed loop***

Tabla 14. Parámetros de Configuración del *Timed loop*

Símbolo	Nombre	Especificaciones
	Tiempo final	Especifica el momento en que el *timed loop* debe completar la iteración. Es igual al período si no se especifica un valor. El valor límite es relativo al inicio de la *timed loop* y se especifica en unidades de la fuente de temporización.
	Error	Se propaga a través de los errores de la estructura. El *timed loop* no se ejecuta si el error recibe una condición de error.
	Modo	Especifica cómo el *timed loop* maneja las ejecuciones finales.
	Nombre de la estructura	Especifica el nombre del *timed loop.*
	*Offset*	Especifica la longitud de tiempo que un *Timed Loop* espera para comenzar la ejecución.
	Periodo	Especifica el período para el *timed loop* en las mismas unidades que la fuente de temporización.
	Prioridad	Específica la prioridad de la ejecución del *timed loop*.
	Procesador	Específica el procesador que controlará la ejecución. El valor predeterminado es -2, lo que significa que *LabVIEW* automáticamente asigna un procesador. Para asignar manualmente un procesador, introduzca un número entre 0 y 255, donde 0 representa el primer procesador. Si introduce un número que supera el número de procesadores disponibles, se genera un error en tiempo de ejecución y la estructura programada no se ejecuta.
	Nombre de la fuente	Especifica el nombre de la fuente de temporización a utilizar para el control de la estructura.
	*TimeOut*	Especifica la cantidad máxima de tiempo que un *timed loop* puede esperar para comenzar la ejecución. El valor predeterminado es -1, lo que significa que no se especifica el tiempo de espera para la siguiente trama.

Los datos adquiridos a un ritmo constante (125 μs) serán visualizados mediante el icono *Waveform Chart* que es un indicador numérico que mantiene un historial de datos. La longitud del historial depende del número de datos que se requiera para este determinado caso se tiene:

$$Longitud\ del\ historial = 8000\frac{datos}{seg} * 40seg = 320000\ datos \qquad (62)$$

Los datos serán importados al software EXCEL para su posterior exportación y procesamiento en el software Matlab®.

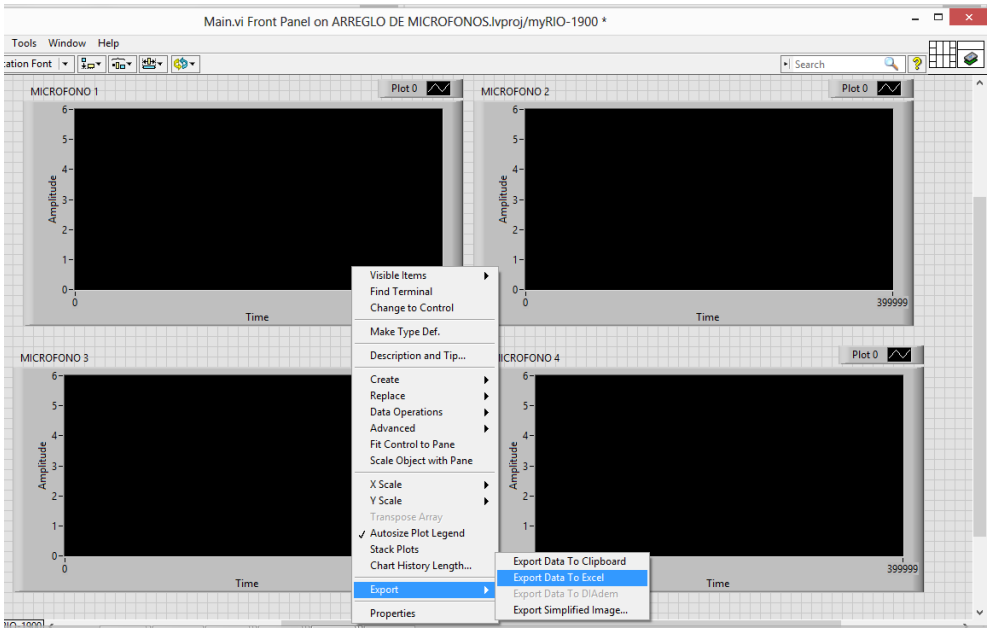


Figura 19. Exportación de información a EXCEL

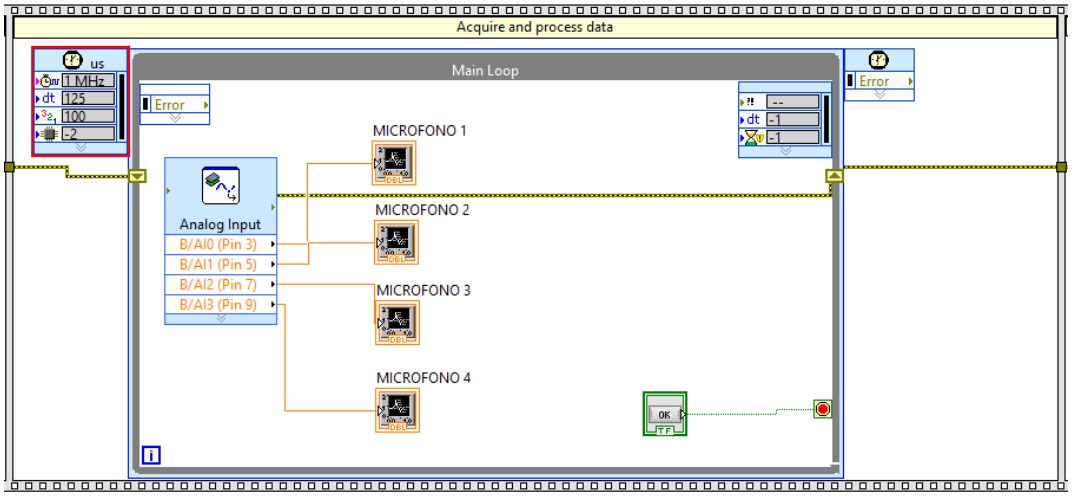


Figura 20. Diagrama de Bloque para la adquisición de las señales

5.1.7. Respuesta de los Micrófonos en Frecuencia

La respuestas de los micrófonos en frecuencia se presenta en la Figura 21, donde se visualiza que el comportamiento de los micrófonos son diferentes; para compensar las diferencias entre los micrófonos propios de su fabricación se emplea la estructura de identificación de sistema y se utiliza el M1 como referencia (Diniz, 2013).

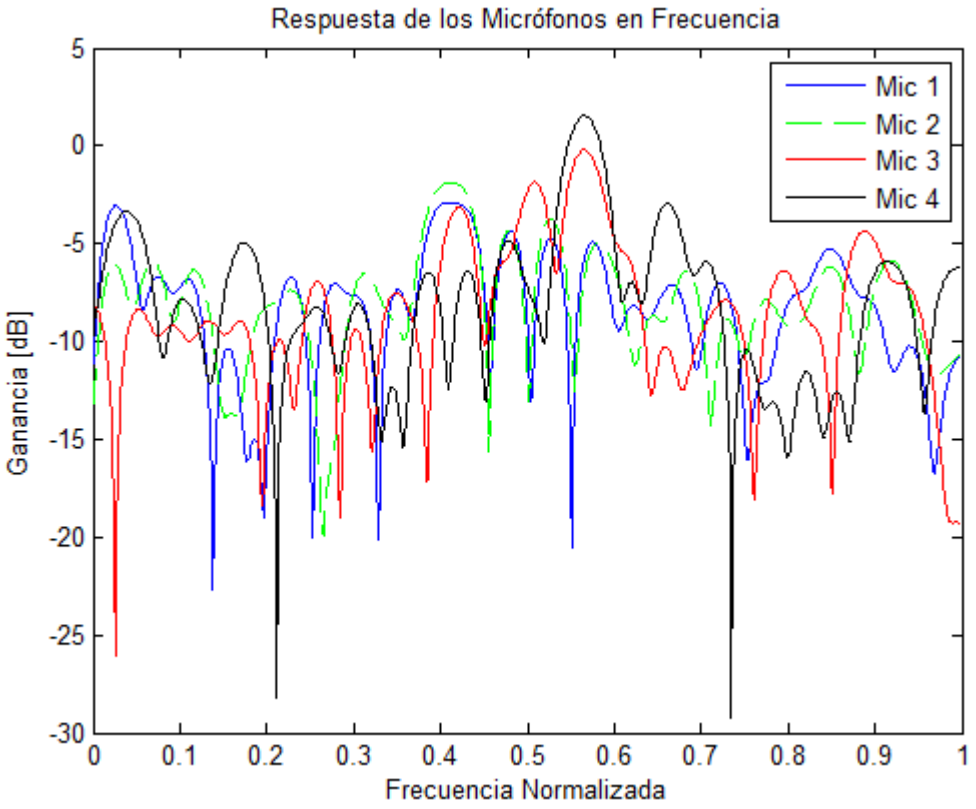


Figura 21. Respuesta de los Micrófonos en Frecuencia para un Arreglo ULA

5.1.8. Compensación de retardos en los micrófonos

Debido a que las señales no fueron adquiridas en una cabina anecoica (una sala diseñada para absorber en su totalidad las reflexiones producidas por ondas acústicas o electromagnéticas en cualquiera de las superficies que la conforman (suelo, techo y paredes laterales), a su vez, la cámara se encuentra aislada del exterior de cualquier fuente de ruido o influencia sonora externa) se ejecuta el algoritmo RLS para obtener la señal de referencia misma que es adquirida tomando la señal del ruido blanco como señal deseada y la señal de entrada la del micrófono M1, esta señal adquirida

es la señal de referencia para los M2, M3 y M4 en la estructura de identificación del sistema.

Compensando el retardo de las señales, que se producen a causa del hardware, procesamiento y propagación de la señal, se observa en la Figura 22 que la respuesta de los micrófonos en frecuencia ha cambiado y su comportamiento entre ellos es similar.

La estructura de identificación del sistema genera los coeficientes con los cuales se estimara una función de transferencia para realizar la ecualización del canal y de esta manera compensar la distorsión lineal causado por el canal (Diniz, 2013). Que se desarrolla más adelante.

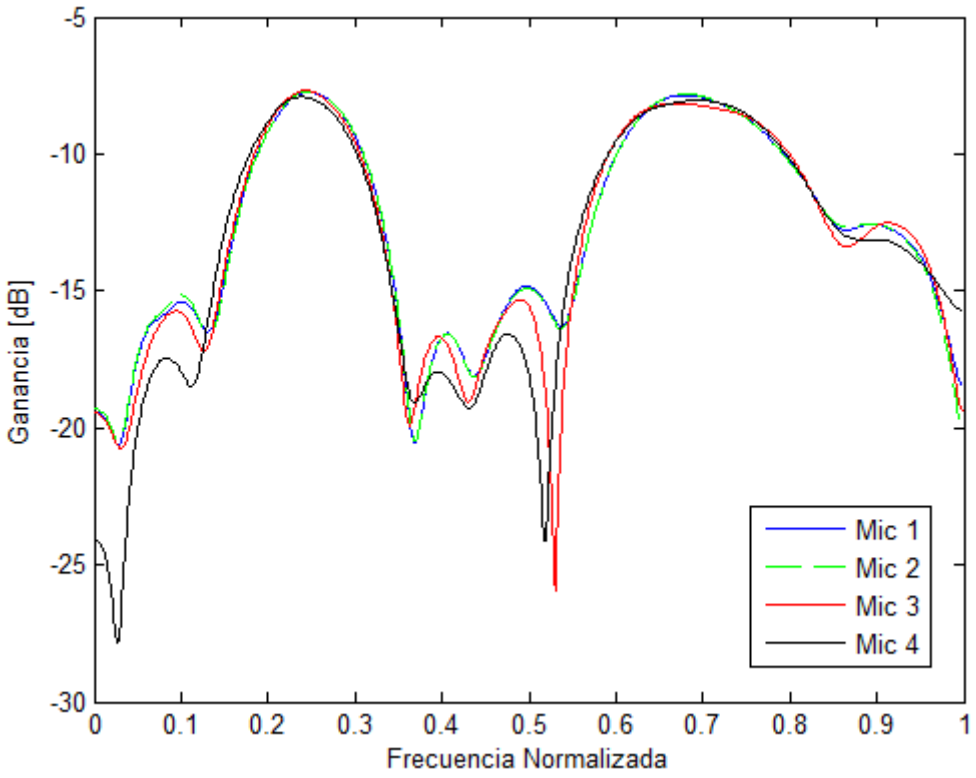


Figura 22. Respuesta Compensada de los Micrófonos en Frecuencia para un Arreglo ULA

5.2. Programación en Tarjeta

Ni myRIO es un hardware embebido diseñado específicamente para desarrollar sistemas de ingeniería avanzados, de manera más rápida. NI myRIO tiene un procesador dual-core ARM Cortex-A9 completamente programable que ejecuta un SO en tiempo real, así como un FPGA.

El software mínimo requerido para el desarrollo del presente proyecto en la tarjeta NI myRIO con labVIEW es el siguiente:

- NI LabVIEW 2013
- Módulo NI LabVIEW Real-Time
- Módulo NI LabVIEW for myRIO

Una vez instalado se tiene acceso a los recursos de la tarjeta NI myRIO directamente desde la ventana LabVIEW.

Figura 23. Ventana principal del software LabVIEW

El programa será ejecutado en el FPGA y en el procesador en tiempo real. La adquisición de los datos de los cuatro micrófonos será mediante el FPGA para luego ser transferido al procesador en tiempo real donde serán procesadas las señales con

los algoritmos desarrollados en la herramienta Matlab®, permitiendo la gestión de los datos tanto de entrada como de salida.

El código en Matlab® de los algoritmos HNCLMS (*Householder Normalized Constrained Least Mean Squares*) y el algoritmo HCCG (*Householder Constrained Conjugate Gradient*) se encuentran desarrollados en texto plano que se ejecutan de la siguiente manera:

Tabla 15. Código Plano desarrollado en Matlab para los algoritmos HNCLMS y HCCG

```
% Señales de los Micrófonos Mic (Adquiridas en el ITEM 5.1.2.)
*************************************************************
load 'microfono1.mat';  % Mic A
w=VarName4-mean(VarName4);
x1=w(20000:200000-1,1);

load 'microfono2.mat';  % Mic A
w1=VarName5-mean(VarName5);
x2=w1(20000:200000-1,1);

load('microfono3.mat');  % Mic C
w2=VarName2-mean(VarName2);
x3=w2(20000:200000-1,1);

load('microfono4.mat');  % Mic D
w3=VarName3-mean(VarName3);
x4=w3(20000:200000-1,1);

tam=max(size(x3));

%Pre-procesamiento: Filtros Equalizadores
*****************************************
fid=fopen('weq1.vet','r');
weq1=fread(fid,inf,'float');
weq1=weq1';
fclose(fid);

fid=fopen('weq2.vet','r');
weq2=fread(fid,inf,'float');
weq2=weq2';
fclose(fid);

fid=fopen('weq3.vet','r');
weq3=fread(fid,inf,'float');
weq3=weq3';
fclose(fid);

fid=fopen('weq4.vet','r');
weq4=fread(fid,inf,'float');
weq4=weq4';
```

Continúa

```
fclose(fid);

% Equalizando
********************
x1=filter(weq1,1, x1);
x2=filter(weq2,1, x2);
x3=filter(weq3,1, x3);
x4=filter(weq4,1, x4);

% Compensando el desfase introducido en las muestras
*****************************************************
x1=x1(4:tam);
x2=x2(1:(tam-3));
x3=x3(2:(tam-2));
x4=x4(2:(tam-2));
tam=max(size(x3));

% Generando Vector xk de acuerdo con notación GJ:
*************************************************
for k=1:tam
      xMk=[x1(k)
           x2(k)
           x3(k)
           x4(k)];
      xk=[xMk
          xk(1:M*N-M)];
end

%Inicialización de variables
*****************************
N=10;  % numero taps
M=4;   % numero de sensores
muHNLMS=0.30;
epsiHNLMS=5e-4;
lambdaHCCG=0.9993;
eta=0.8;
deltax=1e-4;
d=0.5;
theta = pi*[-1:0.005:1];
p=1;
xk=zeros(N*M,1);

% Matriz Constraint
*******************************
fool=ones(1,M);
C=zeros(M*N,N);
for i=1:N
  C(:,i)=[zeros(1,(i-1)*M) fool zeros(1,(N-i)*M)]';
end

% Vector de Ganancia f y Vector F
*********************************
f=[1 zeros(1,N-1)]';
F=C*inv(C'*C)*f;


 % Matriz de Transformación Q
```

Continúa

```
*******************************
L=sqrtm(inv(C'*C));
A=C*L;
[m,n]=size(A);
V=[];

 for       k=1:n
           vtemp=zeros(m,1);
           x=A(k:m,k);
           e1=eye(size(x));
           v=sign(x(1))*norm(x)*e1+x;
           v=v/norm(v);
           vtemp(k:m)=v;
           A(k:m,k:n)=A(k:m,k:n)-2*v*(v'*A(k:m,k:n));
           V=[V vtemp];
 end

Q=eye(N*M);

for hcp=1:N*M
   x=Q(:,hcp);
   bc=N*M+1;
   m=length(x);
   [mv,nv]=size(V);

           for k=1:nv
             x(k:m)=x(k:m)-2*V(k:mv,k)*(V(k:mv,k)'*x(k:m));
           end

           Q(:,hcp)=x;
end

% Inicialización HNLMS y HCCG
*******************************
wHNLMS=Q*F;
wq1=wHNLMS (1:p);
WHNLMS=wHNLMS (p+1:N*M);
MSE1=zeros(tam,1);

wHCCG=Q*F;
wq2=wHCCG(1:p);
wHCCG=wHCCG(p+1:N*M);
g=zeros(N*M-p,1);
c=g;
Rcg=zeros(N*M-p);
MSE2=zeros(tam,1);

X=Q*xk;

for k=1: tam

% Algoritmo HNLMS
*******************************
      eHNLMS=wq1'*X(1:p)-wHNLMS'*X(p+1:N*M);
     wHNLMS=wHNLMS+((+((muHNLMS*eHNLMS*X(p+1:N*M))/
     (X(p+1:N*M)'*X(p+1:N*M)+epsiHNLMS));
     MSE1(k)=MSE1(k)+eHNLMS^2;
```

Continúa

```
% Algoritmo HCCG
*********************************************************
      gm1=g;
      eHCCG=wq2'*X(1:p)-wHCCG'*X(p+1:N*M);
      Rcg=lambdaHCCG*Rcg+X(p+1:N*M)*X(p+1:N*M)';
      alfa=eta*c'*gm1/(c'*Rcg*c+deltax);
      wHCCG=wHCCG+alfa*c;
      g=(lambdaHCCG*gm1)-alfa*Rcg*c+X(p+1:N*M)*eHCCG;
      beta=(g-gm1)'*g/(gm1'*gm1+deltax);
      c=g+beta*c;
      MSE2(k)=MSE2(k)+eHCCG^2;

end
```

5.2.1. Estructura y Tiempo de Muestreo del Programa

5.2.1.1.Estructura del Programa

El programa en *LabVIEW* consta de tres bloques:

- Inicialización
- Adquisición y procesamiento de datos
- Finalización

Para el desarrollo de las tres etapas se utiliza el icono denominado *Flat Sequence Structure* que tiene uno o más sub-diagramas, o tramas, que se ejecutan secuencialmente. Se utiliza la estructura de secuencia plana para asegurar que un sub-diagrama se ejecute antes o después de otro sub-diagrama.

Cada trama en una estructura de secuencia plana se ejecuta de izquierda a derecha y cuando todos los valores de los datos conectados a la respectiva trama están disponibles. Esto significa que la entrada de una trama puede depender de la salida de otra trama.

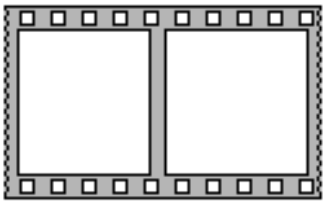


Figura 24. ***Flat Sequence Structure***

5.2.2. Adquisición de datos desde el FPGA y tiempo de muestreo

La adquisición de datos y el tiempo de muestreo se desarrollan como se describe en el ítem 5.1.6.

5.2.3. Transferencia de datos desde el FPGA al procesador en tiempo real

Adquiridos los datos en el periodo definido con el *timed loop* se procede con la transferencia de los mismos desde la FPGA (archivo audiofpga.vi) al procesador de tiempo real.

Con el icono *Open FPGA VI Reference Function* se realiza la transferencia del archivo audiofpga.vi.

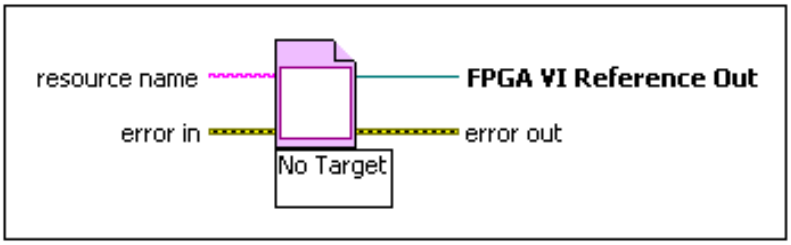


Figura 25. ***Open FPGA VI Reference Function***

Tabla 16. Parámetros de Configuración de *Open FPGA VI Reference Function*

Símbolo	Nombre	Especificaciones
	Resource name	Especifica la tarjeta FPGA en el que desea ejecutar el VI de FPGA. Se debe compilar el VI de FPGA antes.
	Error in	Describe las condiciones de error que se producen antes de que este nodo se ejecute.
	FPGA VI Reference Out	Devuelve una referencia a un FPGA.VI.
	Error out	Contiene la información del error.

Selecciona el *resource name* que en este caso es la tarjeta FPGA– RIO0.

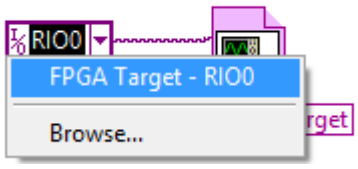


Figura 26. *Resource name*

Se carga el archivo FPGA.vi dando clic derecho en la función de referencia y selecciona Configure Open FPGA VI Reference en el menú contextual y se carga el archivo.

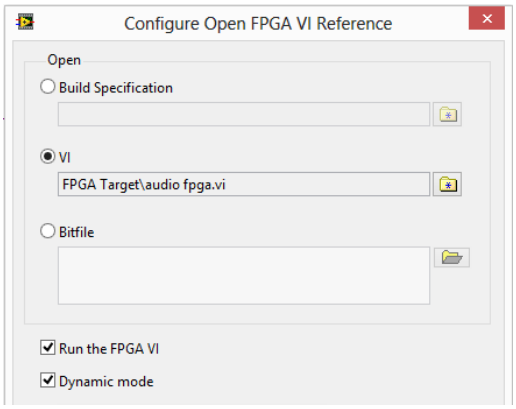


Figura 27. *FPGA VI Reference Function*

Se establece la referencia con la FPGA con el icono *Set Custom Bitfile VI* que se conecta el *FPGA VI Reference Out*

Figura 28. ***Set Custom Bitfile VI***

Tabla 17. Parámetros de Configuración de *Set Custom Bitfile VI*

Símbolo	Nombre	Especificaciones
	Referencia de entrada FPGA	Especifica la entrada de referencia personalizada del FPGA.
	Error in	Describe las condiciones de error que se producen antes de que este nodo se ejecute.
	Referencia de salida FPGA	Devuelve la referencia del FPGA.
	Error de salida	Contiene la información del error.

Finalmente se conecta al bloque *Read/Write Control* que lee un valor o escribe un valor de control o indicador en el archivo FPGA.VI de la tarjeta FPGA.

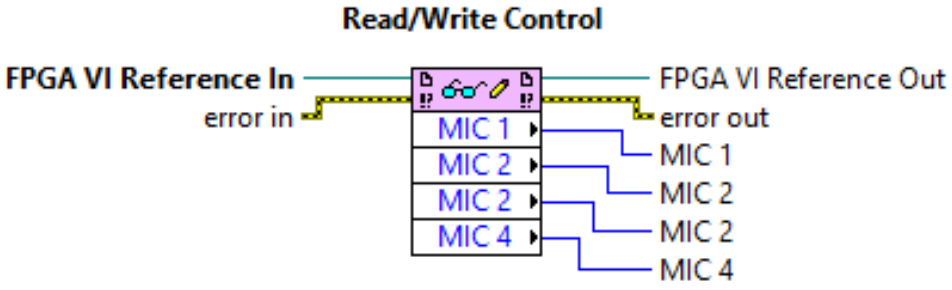


Figura 29. Conexión para la transferencia de datos

Tabla 18. Parámetros de Configuración de *Read/Write* Control

Símbolo	Nombre	Especificaciones
	Referencia de entrada FPGA	Especifica la referencia al archivo FPGA.VI
	Error in	Describe las condiciones de error que se producen antes de que este nodo se ejecute.
	Referencia de salida FPGA	Devuelve la referencia a un FPGA.VI
	Error de Salida	Contiene la información del error.

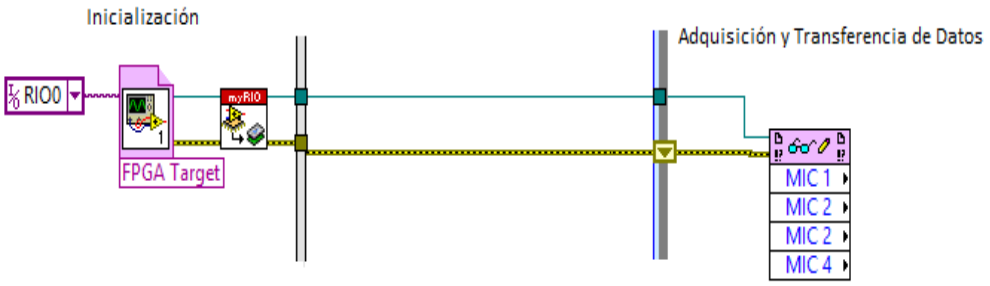


Figura 30. Conexión para la transferencia de datos

5.2.4. Ecualización de Canal

Con los coeficientes ecualizadores obtenidos en el ítem 5.1.8. se procede a generar la función de transferencia del filtro FIR para la ecualización del canal.

En Matlab® se utiliza la función integrada *filter* que se implementa con la sintaxis $y = filter(B, A, x)$

Dónde x es la entrada de señal (un vector de cualquier longitud),

- y es la señal de salida (longitud igual x),
- A es un vector de coeficientes de retroalimentación filtro, y
- B es un vector de coeficientes *feedforward* del filtro.

$$y(k) = x(k) + x(k-1), k = 1,2, \dots, N \quad (63)$$

La implementación en *LabVIEW* de los filtros FIR ecualizadores se realizara con el icono *FIR Filter PtByPt VI.*

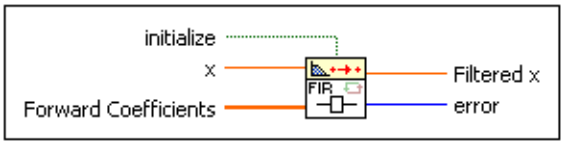


Figura 31. ***FIR Filter PtByPt VI***

Tabla 19. Parámetros de Configuración FIR Filter PtByPt VI

Símbolo	Nombre	Especificaciones
TF	Inicialización	Cuando es verdadero, inicializa el estado interno de VI.
DBL	X	Es la señal de entrada del filtro.
DBL	Coeficientes hacia adelante	Son los coeficientes de *forward* del filtro diseñado (coeficientes de ecualización).
DBL	Filtrado x	Contiene la salida del filtro resultado de la convolución.
I32	Error	Regresa el error.

Los vectores ecualizadores del filtro FIR se guardan en la memoria interna de la tarjeta NI myRIO, para ello se ingresa al sitio web browser de la tarjeta http://172.22.11.2/home/webserv/tmp donde se habilita los permisos de lectura y escritura.

Se realiza el mapeo de red desde la PC como se observa en la Figura 32.

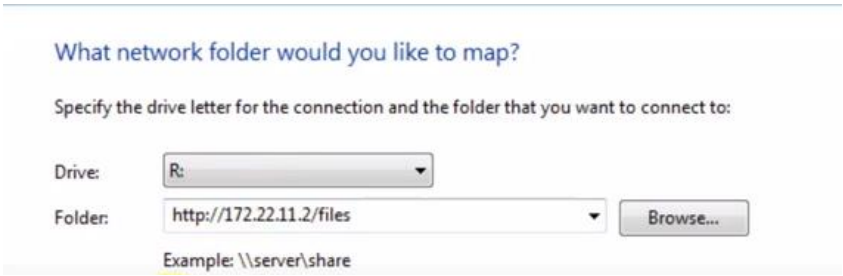


Figura 32. Mapeo de la Red

Una vez mapeado dentro de la carpeta tmp se guarda los archivos .lvm de cada micrófono.

Para la lectura de los coeficientes desde la memoria de la tarjeta se utiliza el icono *Read From Measurement File Express VI.*

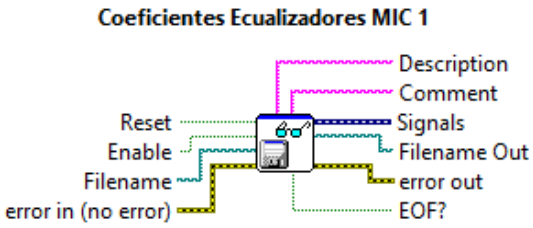


Figura 33. Read From Measurement File Express VI

Tabla 20. Parámetros de Entrada y Salida de *Read From Measurement File Express VI*

Entrada	
Enable	Activa o desactiva el VI express.
Nombre del archive	Especifica el nombre del archivo desde donde se va leer los datos (/home/lvuser/WEQD.LVM).
Reseteo	Restablece la posición del archivo de tal manera que la siguiente lectura comienza al principio del archivo.
Error de Entrada	Describe las condiciones de error que se producen antes de este nodo se ejecuta.
Salida	
Descripción	Devuelve la descripción en la cabecera del archivo .lvm o .tdm.
Comentario	Regresa el comentario en el archivo l.lvm o .tdm.
Señal	Contiene la señal de salida o las señales.
Filename OUT	Regresa el nombre del archivo.
Error de Salida	Contiene la información del error.
EOF?	Indica que el VI Express llega al final del archivo.

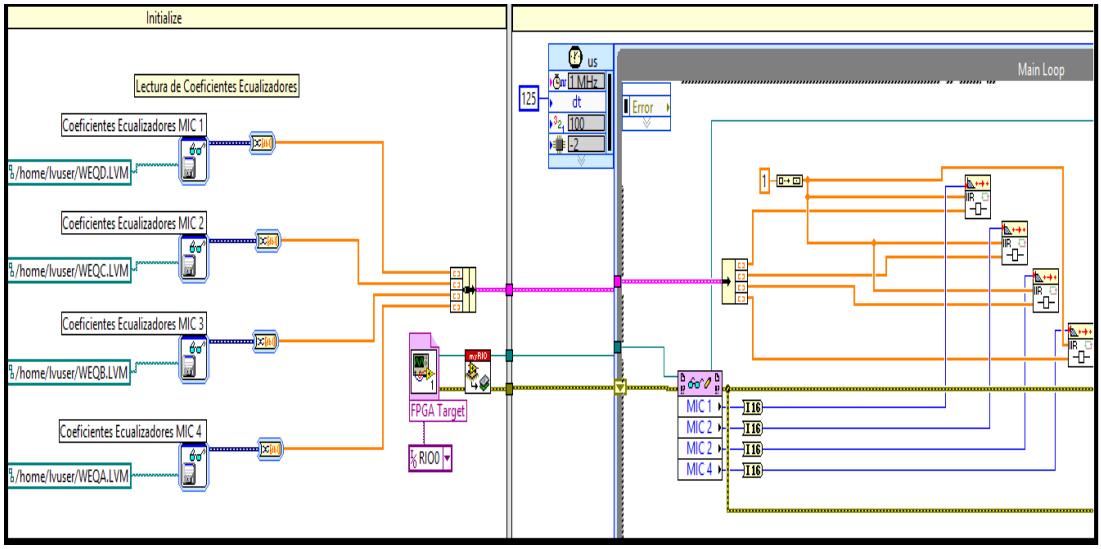


Figura 34. Diagrama de Bloque para la ecualización de los Micrófonos

5.2.5. Compensación del Desfase en las Muestras Según la Geometría Propuesta

5.2.5.1. Geometría Propuesta

Siendo la geometría propuesta un arreglo semiesférico de micrófonos que se encuentran espaciados uniformemente con una apertura entre cada micrófono adyacente de 60°, y la dimensión del radio es 15 *cm* como se observa en la siguiente Figura 35.

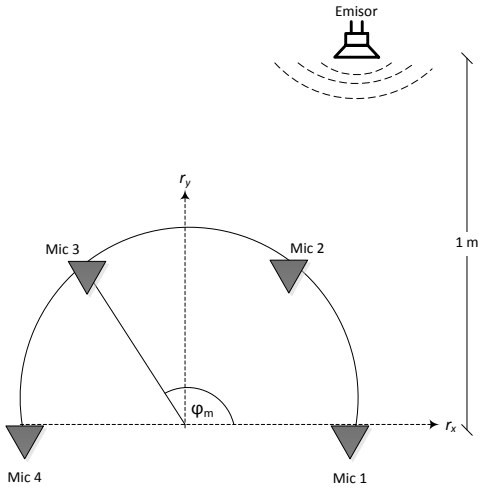


Figura 35. Arreglo Micrófonos Semiesférico

5.2.5.2.Cálculos de Retardos de las muestras

En un medio ideal la velocidad de propagación de las ondas sonoras es constante y el tiempo en el que se propagan depende solamente de la distancia recorrida, siendo el medio el aire cuyo valor es de 340 m/s. En la Figura 35 la distancia d_m establece un camino directo entre la fuente emisora y el eje r_x del arreglo de los micrófonos. La distancia $d_m = 1$m.

$$t_m = \frac{d_m}{\vartheta} \tag{64}$$

El tiempo de propagación de la fuente emisora a los micrófonos $Mic\ 1$ y $Mic\ 4$ es el siguiente:

$$t_1 = t_4 = \frac{1m}{340\ m/s} = 2{,}94\ ms \tag{65}$$

El tiempo de propagación desde la fuente emisora a los micrófonos $Mic\ 2$ y $Mic\ 3$ es:

$$t_2 = t_3 = \frac{0{,}87m}{340\ m/s} = 2{,}56\ ms \tag{66}$$

Entonces la señal que llega al micrófono $Mic\ 1$ tiene un retardo de tiempo de $0{,}38\ ms$ con relación al micrófono $Mic\ 2$ y de manera similar sucede con la señal que arriba al micrófono $Mic\ 4$ que lo hace con un retardo de $0{,}38\ ms$ con relación al micrófono $Mic\ 3$.

Las señales recibidas por los micrófonos son muestreadas a una tasa $f = 8000\ Hz$, por tanto los micrófonos $Mic\ 1$ y $Mic\ 4$ sufrirán un retardo de 3 muestras en relación con los micrófonos $Mic\ 2$ y $Mic\ 3$.

En *LabVIEW* se utilizara el registro de desplazamiento que aparece como un par de terminales directamente opuestos entre sí en los lados verticales de la frontera del bucle.

El registro de desplazamiento se ejecuta de la siguiente manera: el terminal en el lado derecho del bucle almacenan los datos que se calculan en la realización de una interacción, *LabVIEW* transfiere los datos conectados en la parte derecha hacia el terminal izquierdo del bucle, el terminal en el lado derecho del bucle devuelve el último valor almacenado en el registro de desplazamiento. Un registro de desplazamiento transfiere cualquier tipo de datos y cambia automáticamente al tipo de datos.

5.2.6. Generación del Vector xk

El vector xk se forma de la siguiente manera y su tamaño está definido por el número de micrófonos (4) y el número de TAPS (10):

$$xk = \begin{bmatrix} x1_1 \\ x2_1 \\ x3_1 \\ x4_1 \\ . \\ . \\ . \\ x1_{10} \\ x2_{10} \\ x3_{10} \\ x4_{10} \end{bmatrix}$$

En labVIEW para la generación del vector se utiliza el icono *Initialize Array Function* que inicializa el arreglo con el tamaño del vector.

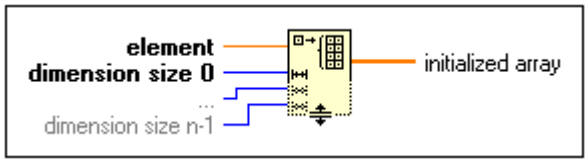


Figura 36. ***Initialize Array Function***

Tabla 21. Parámetros de Configuración *Initialize Array Function*

Icono	Nombre	Especificaciones
	Elemento	Este es un valor usado para inicializar todos los elementos del arreglo.
	Tamaño del vector	Crea una matriz vacía y de la dimensión del arreglo.
	Inicialización del arreglo	Es la matriz del mismo tipo de elemento generado.

Con el icono *Replace Array Subset Function* que sustituye un elemento o subconjunto de una matriz en el punto que especifique el índice.

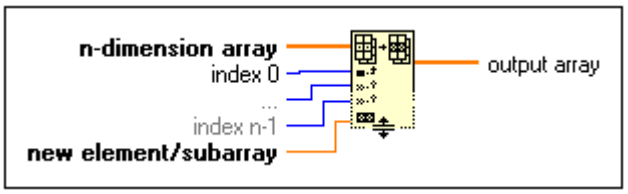


Figura 37. *Replace Array Subset Function*

Tabla 22. Parámetros de Configuración *Replace Array Subset Function*

Icono	Nombre	Especificaciones
	Dimensión del arreglo	Es la dimensión de la matriz en la que desea reemplazar un elemento (s), fila (s), columna (s). Esta entrada puede ser una matriz n-dimensión de cualquier tipo.
	Índice	Especifica el elemento, fila, columna que desea reemplazar en la matriz.
	Nuevo elemento	Es la matriz o elemento que reemplaza un elemento, fila, columna en la matriz especificada.
	Salida del arreglo	Es la matriz de salida generada.

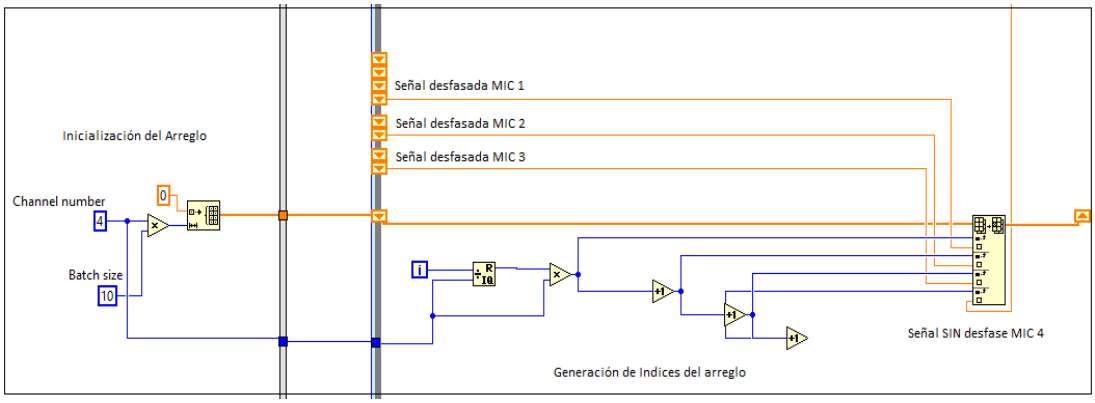


Figura 38. Diagrama de Bloques Generación del vector xk

5.2.7. Algoritmos para cálculos de constantes y constantes definidas

En el *frame* de inicialización de la estructura *flat Sequence* definimos las constantes y se ejecuta los algoritmos para el cálculo de:

- Matriz Constraint
- Vector de ganancia f y Vector F
- Matriz de transformación Q

Para ello se utiliza el *MathScript Node.*

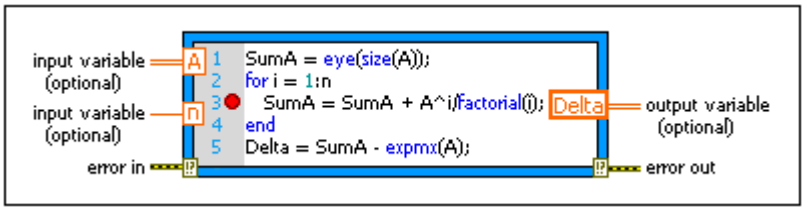
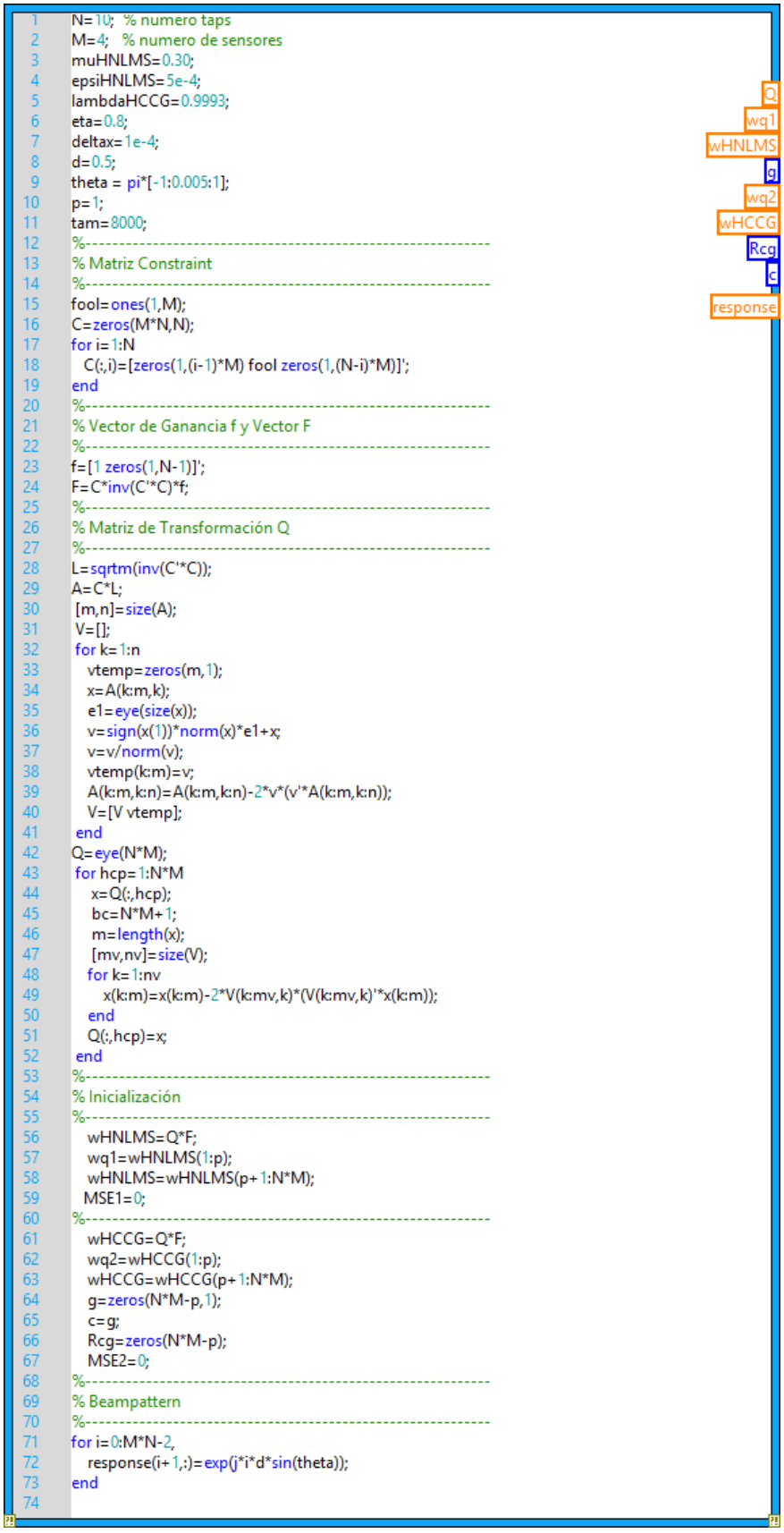


Figura 39. MathScript Node.

Se utiliza este nodo para ejecutar las secuencias de comandos, ingresar un script, que puede contener funciones incorporadas por *MathScript* o funciones definidas por el usuario.

En el borde del nodo *MathScript* se puede agregar variables de entrada y de salida.



```
N=10;  % numero taps
M=4;   % numero de sensores
muHNLMS=0.30;
epsiHNLMS=5e-4;
lambdaHCCG=0.9993;
eta=0.8;
deltax=1e-4;
d=0.5;
theta = pi*[-1:0.005:1];
p=1;
tam=8000;
%-------------------------------------------------------------
% Matriz Constraint
%-------------------------------------------------------------
fool=ones(1,M);
C=zeros(M*N,N);
for i=1:N
  C(:,i)=[zeros(1,(i-1)*M) fool zeros(1,(N-i)*M)]';
end
%-------------------------------------------------------------
% Vector de Ganancia f y Vector F
%-------------------------------------------------------------
f=[1 zeros(1,N-1)]';
F=C*inv(C'*C)*f;
%-------------------------------------------------------------
% Matriz de Transformación Q
%-------------------------------------------------------------
L=sqrtm(inv(C'*C));
A=C*L;
 [m,n]=size(A);
 V=[];
 for k=1:n
   vtemp=zeros(m,1);
   x=A(k:m,k);
   e1=eye(size(x));
   v=sign(x(1))*norm(x)*e1+x;
   v=v/norm(v);
   vtemp(k:m)=v;
   A(k:m,k:n)=A(k:m,k:n)-2*v*(v'*A(k:m,k:n));
   V=[V vtemp];
 end
Q=eye(N*M);
 for hcp=1:N*M
    x=Q(:,hcp);
    bc=N*M+1;
    m=length(x);
    [mv,nv]=size(V);
   for k=1:nv
      x(k:m)=x(k:m)-2*V(k:mv,k)*(V(k:mv,k)'*x(k:m));
   end
   Q(:,hcp)=x;
 end
%-------------------------------------------------------------
% Inicialización
%-------------------------------------------------------------
   wHNLMS=Q*F;
   wq1=wHNLMS(1:p);
   wHNLMS=wHNLMS(p+1:N*M);
  MSE1=0;
%-------------------------------------------------------------
   wHCCG=Q*F;
   wq2=wHCCG(1:p);
   wHCCG=wHCCG(p+1:N*M);
   g=zeros(N*M-p,1);
   c=g;
   Rcg=zeros(N*M-p);
   MSE2=0;
%-------------------------------------------------------------
% Beampattern
%-------------------------------------------------------------
for i=0:M*N-2,
   response(i+1,:)=exp(j*i*d*sin(theta));
end
```

Figura 40. Inicialización de constantes y cálculos de algoritmos con respuesta constante

En el borde derecho del nodo que se muestra en la Figura 40 observamos las constantes que se utilizaran para la ejecución de los algoritmos.

5.2.8. Desarrollo de los algoritmos HNLMS y HCCG

En el *frame* de adquisición y procesamiento de datos de la estructura *flat Sequence* se desarrollan los algoritmos HNLMS y HCCG en el nodo *MathScript* que se encuentra dentro del bucle infinito; en el borde izquierdo del nodo se define las entradas constantes y aquellas variables que necesitan ser almacenadas debido a que se necesitan en las siguientes interacciones por lo que se utiliza el registro de desplazamiento.

Tabla 23. Constantes y Variables de los Algoritmos HNLMS y HCCG

Algoritmo	Entradas Constantes	Entradas Variables
HNCLMS	xk, Q, wq1, response	wHNLMS
HCCG	xk, Q, wq2, response	wHCCG, Rcg, c, g

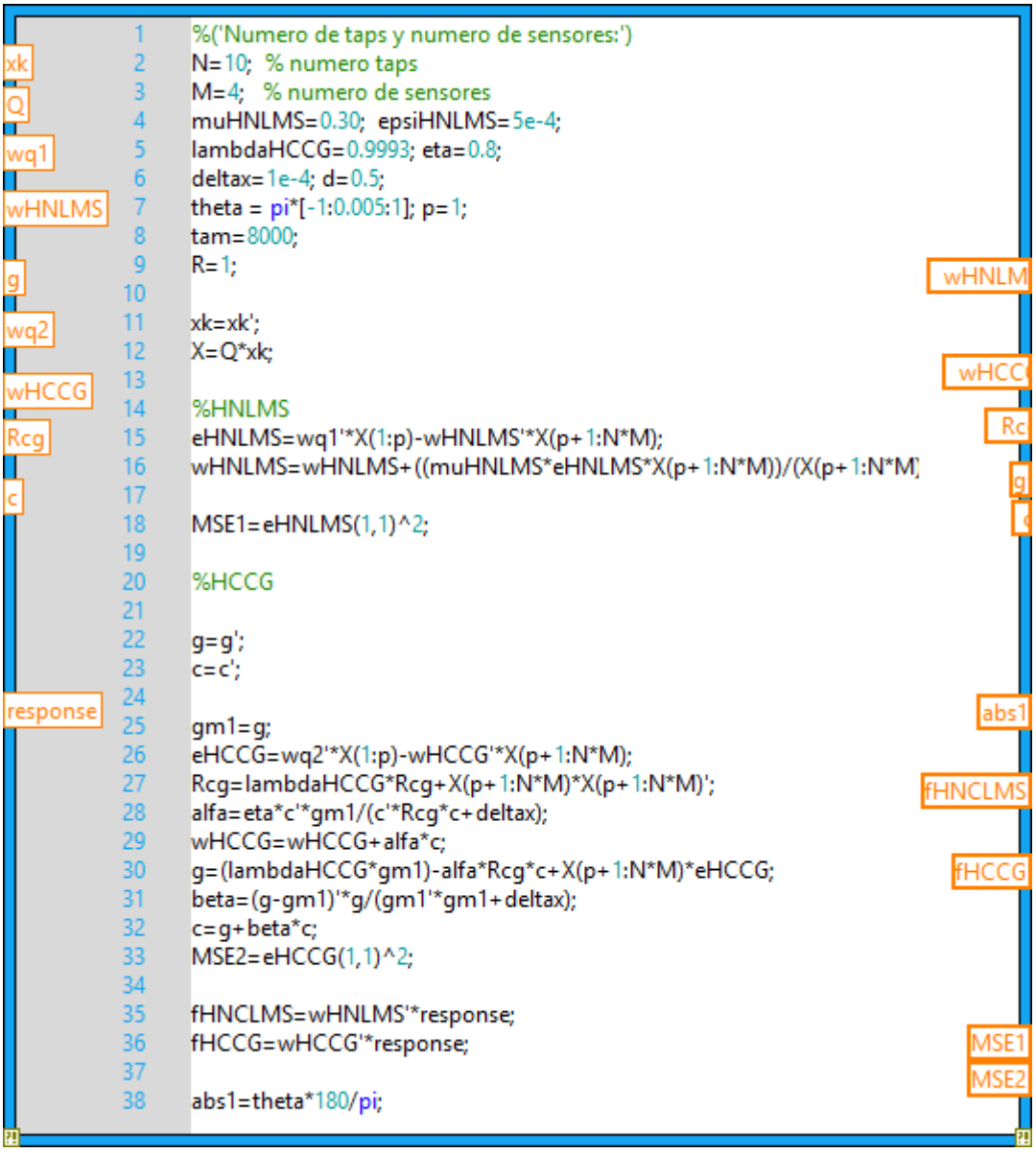
5.2.9. Salidas de los algoritmos HNLMS y HCCG

Para el análisis del comportamiento de los micrófonos se realiza las siguientes graficas:

- El error medio cuadrado MSE, que se visualizará en una escala logarítmica.
- La media del MSE menMSE.
- Los algoritmos con respecto al beampattern.

Tabla 24. Salidas para análisis de comportamiento de algoritmos HNLMS y HCCG

Algoritmo	Salida
HNCLMS	MSE1, fHNCLMS
HCCG	MS2, fHCCG




```
1  %('Numero de taps y numero de sensores:')
2  N=10;  % numero taps
3  M=4;   % numero de sensores
4  muHNLMS=0.30;  epsiHNLMS=5e-4;
5  lambdaHCCG=0.9993; eta=0.8;
6  deltax=1e-4; d=0.5;
7  theta = pi*[-1:0.005:1]; p=1;
8  tam=8000;
9  R=1;
10
11 xk=xk';
12 X=Q*xk;
13
14 %HNLMS
15 eHNLMS=wq1'*X(1:p)-wHNLMS'*X(p+1:N*M);
16 wHNLMS=wHNLMS+((muHNLMS*eHNLMS*X(p+1:N*M))/(X(p+1:N*M)
17
18 MSE1=eHNLMS(1,1)^2;
19
20 %HCCG
21
22 g=g';
23 c=c';
24
25 gm1=g;
26 eHCCG=wq2'*X(1:p)-wHCCG'*X(p+1:N*M);
27 Rcg=lambdaHCCG*Rcg+X(p+1:N*M)*X(p+1:N*M)';
28 alfa=eta*c'*gm1/(c'*Rcg*c+deltax);
29 wHCCG=wHCCG+alfa*c;
30 g=(lambdaHCCG*gm1)-alfa*Rcg*c+X(p+1:N*M)*eHCCG;
31 beta=(g-gm1)'*g/(gm1'*gm1+deltax);
32 c=g+beta*c;
33 MSE2=eHCCG(1,1)^2;
34
35 fHNCLMS=wHNLMS'*response;
36 fHCCG=wHCCG'*response;
37
38 abs1=theta*180/pi;
```

Figura 41. Procesamiento de los Algoritmos en Tiempo Real

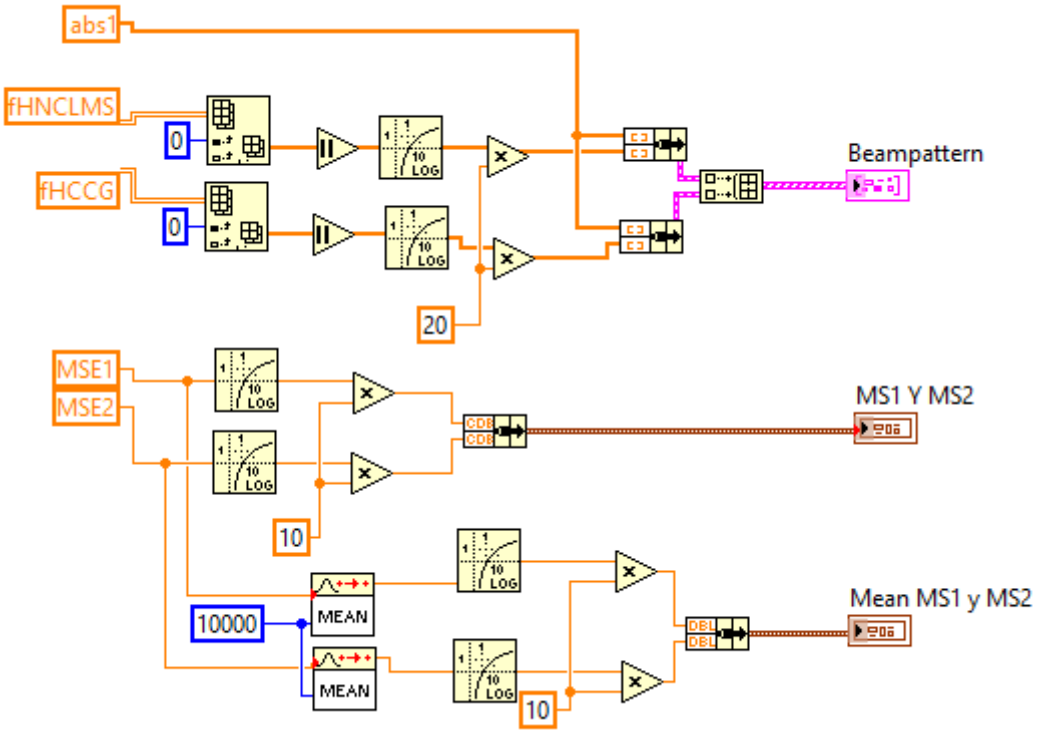


Figura 42. Diagrama de Bloques de Gráficos de Salida

5.3. Análisis de Resultados de los algoritmos en la Geometría Propuesta

El MSE se ilustra adicionalmente en la Figura 43 donde se representa los dos algoritmos con todas las interacciones, se puede observar como los algoritmos convergen alcanzando el algoritmo HNCLMS un MSE promedio de $-28.8352\ dB$, mientras que el algoritmo HCCG alcanza un MSE menor de $-33.8471\ dB$. En la Figura 45 se muestra el *beampattern* de los algoritmo HNCLMS y HCCG, según dicha figura el ángulo obtenido coincide con el de la señal de entrada de 90°. Los dos algoritmos presentan una alta resolución.

El algoritmo HCCG presenta un alto desempeño y esto se puede corroborar cuando se compara el *beampattern*, como se puede observar en la Figura 45, en la cual se tiene que la ganancia del algoritmo HCCG es aceptable.

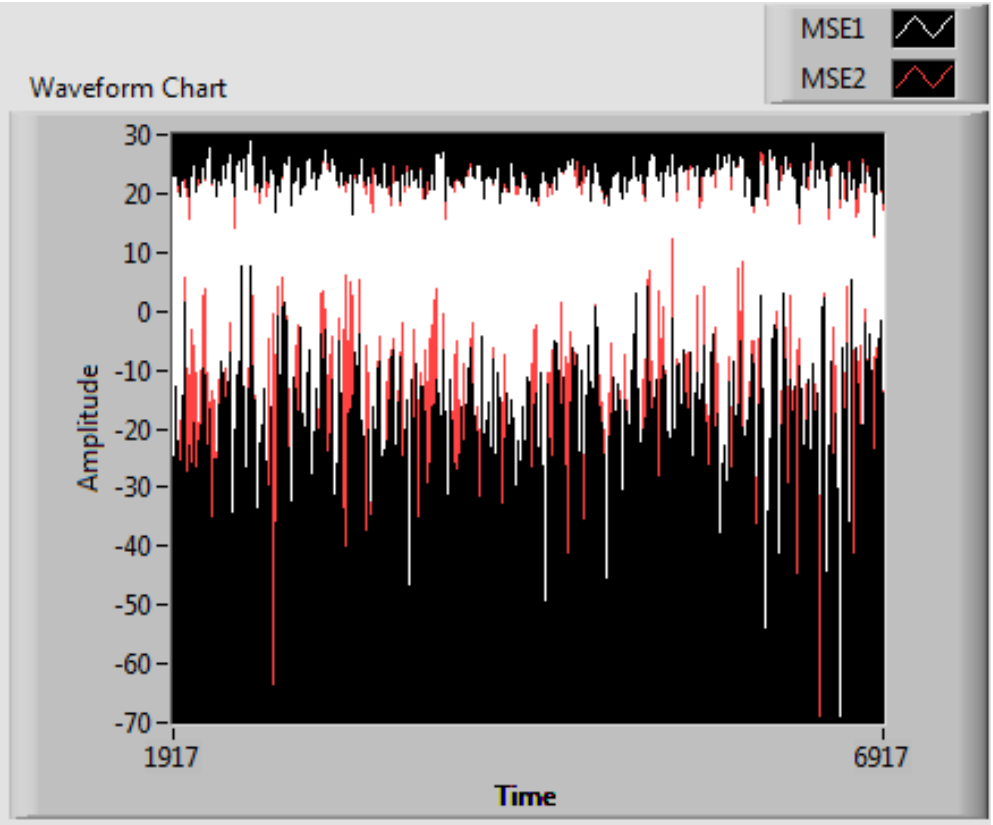


Figura 43. Comparación de los algoritmos *Householder* respecto al MSE para un arreglo semiesférico con todas las interacciones

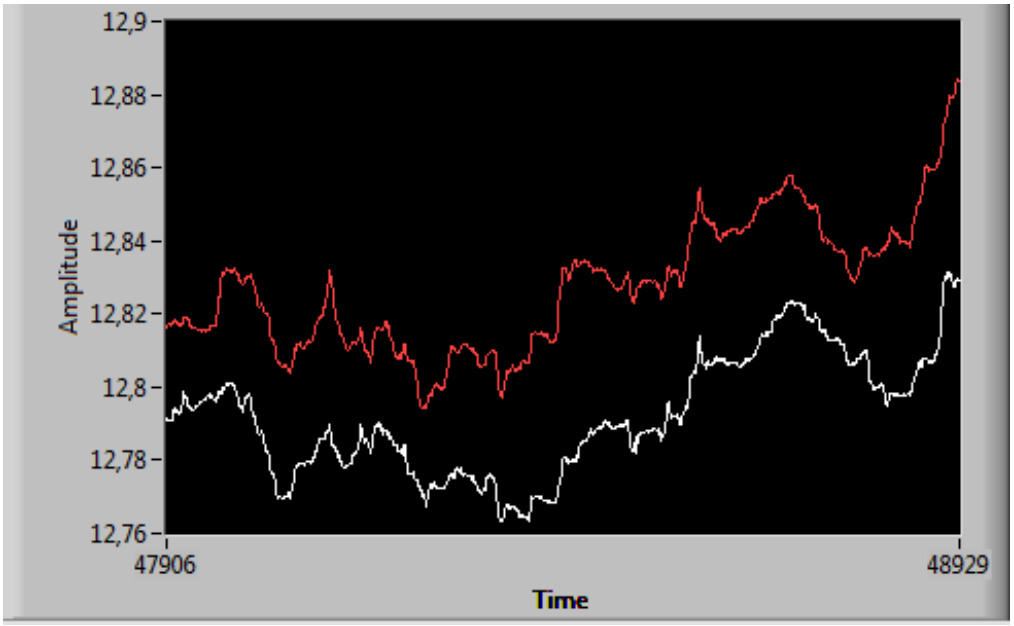


Figura 44. Comparación de los algoritmos Householder respecto al MSE para un arreglo semiesférico

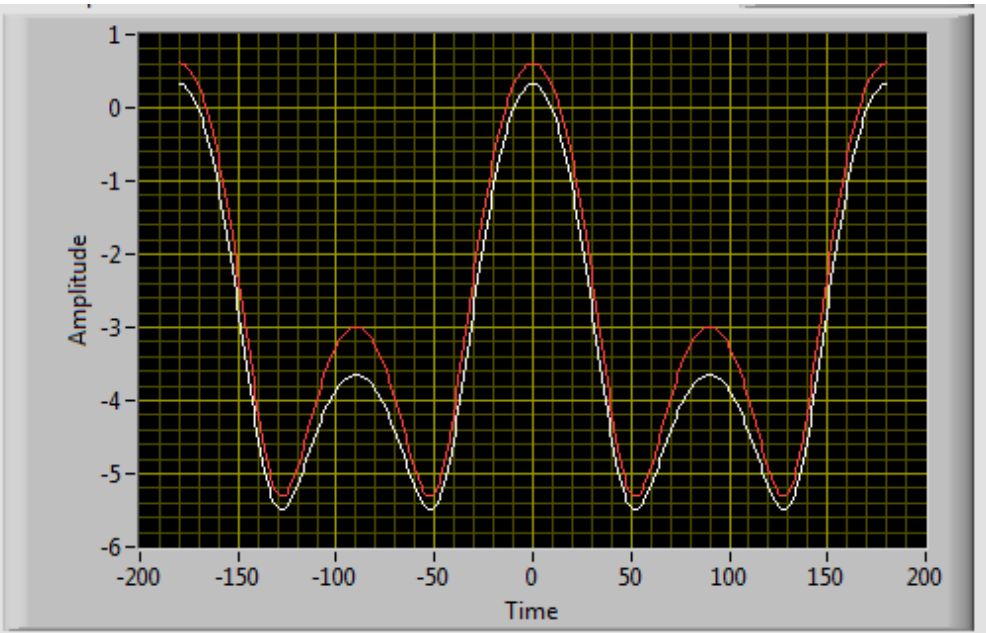


Figura 45. Comparación de los algoritmos *Householder* respecto al *beampattern* para un arreglo semiesférico

CAPÍTULO VI

6. CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

- Se cumplió con el objetivo de implementar un arreglo semiesférico de micrófonos que permita realzar la voz de manera adecuada utilizando algoritmos adaptativos LCMV en el dispositivo NI myRIO-1900.

- Se verifico el rendimiento de los algoritmos HNCLMS y HCCG para diferentes números de bits en base a la precisión finita, donde se concluye que el error MSE es menor si el número de bits aumenta, debido a que al ser menor el número de bits existe un incremento de las operaciones y por ende una propagación del error.

- Se realizó el estudio del algoritmo *Lattice Recursive Least Squares* mediante la imposición de restricciones lineales sobre el filtro; pero al abordar la investigación sobre el tema, no se encuentra información. Siendo este un tema de investigación más profundo.

- La teoría de arreglo de sensores indica que estos sensores deben tener un comportamiento isotrópicos al realizar el análisis se comprobó que en casos prácticos no cumple, esto es propio de su fabricación, de las propiedades físico-químicas de los materiales empleados, por lo que se aproximó su comportamiento mediante la ecualización de los mismos tomando como referencia el $Mic\ 1$.

- Se realizó la programación directamente desde la ventana LabVIEW en el dispositivo Ni myRIO que es un hardware embebido; la adquisición de datos se hizo mediante el FPGA y el desarrollo del algoritmo en el procesador dual-core ARM Cortex-A9 completamente programable que ejecuta un SO en tiempo real.

- Se concluye que la herramienta LabVIEW constituye un revolucionario sistema de programación gráfica para aplicaciones que involucran adquisición, control y presentación de datos, empleando programación gráfica basada en diagramas de bloques.

- El tiempo de ejecución del algoritmo HCCG es de 8*ms* y del algoritmo HCNLMS es de 6*ms;* debido a la adquisición de los datos, procesamiento de las señales, algoritmos ejecutados en el *Math Scrip* de Matlab.

- Los algoritmos NLMS y CG presentan un valor en sus MSE similar y un buen desempeño al realizar el análisis de las gráficas. Sin embargo se debe considerar que el algoritmo HCNLMS tiene un tiempo menor de ejecución por que su algoritmo tiene un menor número de operaciones.

6.2. Recomendaciones

- Se recomienda ampliar el estudio de las señales adquiridas en un ambiente idóneo, ya que en la presente investigación se utiliza señales del tipo AWGN pero estas señales no fueron adquiridas en una cabina anecoica, de esta manera se mejorara el resultado final.

BIBLIOGRAFÍA

Agilent EEsof EDA Software . (2000). *Herramienta Advanced Design System*. Recuperado el 7 de Agosto de 2013, de http://www.home.agilent.com/en/pc-1297113/advanced-design-system

Apolinário, J., de Campos, M., & Bernal, C. (2000). The Constrained Conjugate Gradient Algorithm. *IEEE SIGNAL PROCESSING LETTERS, VOL. 7, NO. 12,*.

Papoulis , A., & Pillai , U. (2002). *Probability, Random Variables and Stochastic Processes.* International Edition.

Apolinário , J. (2003). *PROCESSAMENTO DIGITAL DE SINAIS.*

Apolinário, J. A. (2009). *QDR-RLS Adaptive Filtering.* Springer.

Apolinario, J., & de Campos, M. (2011). *Instituto Militar de Engenharia*. Recuperado el 21 de 12 de 2014, de http://aquarius.ime.eb.br/~apolin

Benesty, J., Chen, J., & Huang, Y. (2008). *Microphone Array Signal Processing.* USA: Springer-Verlag Berlin Heidelberg.

Bernal Oñate, C. (2000). *Principios y Aplicaciones de CDMA "Code Division multiple access" con implementación de algoritmos para la detección de multiusuarios.* Tesis , Quito.

Bernal, P., Sáenz, F., & Romero , C. (2014). Análisis de Señales Acústicas. *Departamento de Eléctrica y Electrónica, Universidad de las Fuerzas Armadas ESPE*.

Caisapanta, A. (2015). *Optimización de las señales acústicas en un arreglo semiesférico de micrófonos utilizando la metodología de beamforming de banda ancha.* ESPE, Quito.

Chandran, S. (2004). *Adaptive Antenna Arrays Trends and Applications.* Springer-Verlag Berlin Heidelberg.

Chandran, S. (2004). *Adaptive Antenna Arrays: Trends and Applications.* Malaysia: Springer.

de Campos, M., Werner, S., & Apolinário, J. (September 2002). Constrained Adaptation Algorithms Employing Householder Transformation. *IEEE TRANSACTIONS ON SIGNAL PROCESSING VOL. 50, NO. 9,* 9.

Diniz, P. (2013). *Adaptive Filtering: Algorithms and Practical Implementation* (Fourth ed.). Rio de Janeiro, Brazil: Springer New York Heidelberg Dordrecht London.

Golub, G., & Van Loan, C. (2013). *Matrix Computations Fourth Edition.* The Johns Hopkins University Press.

Gundersen, K., & Hakon Husoy, J. (2006). Preconditioner structures for the CLMS adaptive filtering algorithm. *IEEE*.

Huang, Z., & Balanis, C. (2005). Adaptive Beamforming Using Spherical Array. *IEEE*.

Liu, W., & Weiss, S. (2010). *WIDEBAND BEAMFORMING: Concepts and Techniques.* John Wiley & Sons, Ltd.

Madisetti, V. (2010). *The Digital Signal Processing Handbook* (SECOND ed.). Taylor and Francis Group, LLC.

Medina, C. A., Rodríguez, C. V., Apolinário, J. A., & León, R. D. (2000). *IMPLEMENTACIÓN DE UN ARREGLO SUPERDIRECTIVO DE MICRÓFONOS CON MÚLTIPLES LÍNEAS DE RETARDO.* Quito.

Monzingo, R., Haupt, R., & Miller, T. (2011). *Introduction to Adaptive Arrays* (Second ed.). Published by SciTech Publishing, Inc.

Rabiner, L., & R. W., S. (2009). *Theory and Application of Digital Speech Processing.*

T. SANSALONI, J. V. (2000). Simulador de Sistemas Digitales de Preisión Finita.

Valle, S. d. (2009). *Manual Práctico de Acústica* (Terceira ed.). Rio de Janeiro : Música & Tecnología.

Van Veen, B., & Buckley, K. (2000). Beamforming Techniques for Spatial Filtering. *CRCnetBASE*.

Werner, S., Apolinário, J., & Marcello L. R. , d. (s.f.). On the Equivalence of the Constrained RLS and the GSC-RLS Beamformers. *Helsinki University of Technology, Instituto Militar de Engenharia, and Universidade Federal do Rio de Janeiro*.

Yang, W. Y., Cho, Y. S., & Choo, C. Y. (2012). *MATLAB /Simulink for Digital SIgnal Processing.* Seoul, Korea: Hongrung Publishing Company.

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

INGENIERÍA EN ELECTRÓNICA EN REDES Y COMUNICACIÓN DE DATOS

ACTA DE ENTREGA

El presente proyecto de investigación fue entregado al Departamento de Eléctrica y Electrónica y reposa en los archivos desde:

Sangolquí, 31 Julio 2015

Elaborado por

Srta. Tatiana Paola Caisapanta Larrea

Autoridad:

Dr. Nikolai Espinosa PhD

DIRECTOR DE LA CARRERA DE INGENIERÍA EN ELECTRÓNICA EN REDES Y COMUNICACIÓN DE DATOS