UNIVERSIDAD NACIONAL DE CHIMBORAZO

FACULTAD DE INGENIERÍA

CARRERA DE ELECTRÓNICA Y TELECOMUNICACIONES

TÍTULO DEL TRABAJO DE INVESTIGACIÓN

DISEÑO E IMPLEMENTACIÓN DE UN FILTRO ADAPTATIVO MEDIANTE FILTROS WIENER PARA LA CANCELACIÓN DE RUIDO EN SEÑALES DE AUDIO

AUTOR(ES):

HENRRY PAÚL EBLA TAPIA

PAÚL ISRAEL INCA PEÑA

DIRECTOR:

ING. FABIÁN GUNSHA

AÑO

2016

Los miembros de Tribunal de Graduación del proyecto de investigación con el título: **DISEÑO E IMPLEMENTACIÓN DE UN FILTRO ADAPTATIVO MEDIANTE FILTROS WIENER PARA LA CANCELACIÓN DE RUIDO EN SEÑALES DE AUDIO**
Presentado por: **Henrry Paúl Ebla Tapia, Paúl Israel Inca Peña** y dirigida por: **Ingeniero Fabián Gunsha.**

Una vez escuchada la defensa oral y revisado el informe final del proyecto de investigación con fines de graduación escrito en la cual se ha constatado el cumplimiento de las observaciones realizadas, remite el presente para uso y custodia en la biblioteca de la Facultad de Ingeniería de la UNACH.

Para constancia de lo expuesto firman:

Ing. Paulina Vélez
Presidente del Tribunal **Firma**

Ing. Fabián Gunsha
Director del Proyecto **Firma**

Ing. Giovanny Cuzco
Miembro de Tribunal **Firma**

CERTIFICADO

El suscriptor Ing. Fabián Celso Gunsha Maji, Docente de la Facultad de Ingeniería con **C.I:0602601775**, de la ciudad de Riobamba, en calidad de tutor del proyecto de investigación **CERTIFICA:**

Que los Señores **Henrry Paul Ebla Tapia, Paul Israel Inca Peña**, de nacionalidad ecuatoriana portadores de la Cédula de Identidad N° **0604105338**, N° **0604021097,** respectivamente están en el **100%** del proyecto de investigación titulado, **"DISEÑO E IMPLEMENTACIÓN DE UN FILTRO ADAPTATIVO MEDIANTE FILTROS WIENER PARA LA CANCELACIÓN DE RUIDO EN SEÑALES DE AUDIO ".**

Es todo cuanto puedo certificar en honor a la verdad.

Riobamba, 5 de Agosto del 2016.

Atentamente.

Ing. Fabián Gunsha.
Docente de La Facultad

AUTORÍA DE INVESTIGACIÓN

La responsabilidad del contenido de este Proyecto de Graduación, nos corresponde exclusivamente a **Henrry Ebla, Paúl Inca** e **Ingeniero Fabián Gunsha;** y el patrimonio intelectual de la misma a la Universidad Nacional de Chimborazo.

Henrry Paúl Ebla Tapia

C.I 060410533-8

Paúl Israel Inca Peña

C.I 060402109-7

AGRADECIMIENTO

A Dios por ser la luz durante todo el camino de mi vida, porque junto a él lo podemos hacer todo y sin él no lograríamos nada.

A mí querida esposa Anita que ha sido el pilar que me sostuvo en los momentos difíciles, me dio ánimo para seguir adelante y nunca claudicar por muchos que sean los golpes y tropiezos que existió en el camino de mi vida universitaria.

A mí amado hijo Matías por ser mi fuente de inspiración y motivación para poder superarme cada día más y así poder luchar para que la vida nos depare un futuro mejor.

Agradezco a mi querida madre que ha sido la guía durante toda mi vida y nunca me ha desamparado, la que me ha inculcado luchar por lo que se quiere y no rendirse, a mi padre y mis hermanos que me han brindado su apoyo incondicional durante los momentos difíciles.

Un agradecimiento al Ing. Fabián Gunsha por confiar en nuestras capacidades y brindarnos sus conocimientos durante el desarrollo de este proyecto.

Henrry Ebla.

AGRADECIMIENTO

En primer lugar debo agradecer a Dios porque siempre ha estado ahí para guiarme y bendecirme en mi camino, aunque tuve muchos tropiezos siempre me dio las fuerzas para seguir adelante, me bendijo con una madre y un hermano que siempre me supieron apoyar en los momentos más difíciles de mi vida, brindándome fuerzas y esperanzas para llegar a una meta que parecía no tener un final.

Agradezco a mi tutor de Tesis Ing. Fabián Gunsha Docente el cual siempre ha demostrado ser un gran profesional y sobretodo una persona digna de admiración.

Paúl Inca

DEDICATORIA

Dedico este trabajo de graduación a mi querida familia a mi esposa Anita y mi hijo Matías que han sido la fuente de motivación e inspiración, para seguir adelante y no rendirme pese a los problemas que nos hemos enfrentado durante todo este tiempo, este gran esfuerzo es por y para ellos para forjar un mejor futuro para nuestro hijo.

A mi querida Abuelita que durante mi niñez mi enseño que en la vida hay que ser humilde y luchar por conseguir las metas que nos trazamos en nuestras vidas, por más difíciles que estas sean y estoy seguro que desde el cielo me sigue enviando sus bendiciones.

A mí amada Madre que ha sido la guía durante toda mi vida y que pese a las dificultades e injusticias que hemos pasado en la vida siempre ha estado junto a mí y ha hecho de mí una buena persona.

Henrry Ebla

DEDICATORIA

Dedico este gran éxito de mi vida a mi madre Clelia Peña y Raúl Inca los cuales siempre han estado con migo en los momentos alegres y tristes de mi vida su apoyo fue incondicional y totalmente desinteresado, me enseñaron que con esfuerzo y confianza todo se puede lograr.

Dedico este nuevo éxito de mi vida a todas las personas y familiares que pusieron ese granito de arena para que esta meta la cumpliera con satisfacción y comenzar una nueva etapa de mi vida gracias a todos.

Paúl Israel

ÍNDICE GENERAL

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABLAS

RESUMEN

El presente trabajo de investigación es el diseño e implementación de un filtro adaptativo utilizando un filtro de Wiener, para cancelar el ruido que se encuentra presente en las señales de audio. Este filtro adaptativo es programado en la tarjeta de desarrollo DISCOVERY KIT, compuesto por un procesador ARMCORTEX M7 STM32F746G, que permitirá el procesamiento de señales, es programado en lenguaje C de Keil uVision5 y STM32CubeMX, para la adaptación matemática se utiliza la teoría de Wiener mediante un algoritmo adaptativo LMS (Least-Mean-Square algorithm).

El sistema está compuesto por dos entradas, una a ser utilizada como la entrada de audio más ruido y la otra como la entrada de ruido, las señales son procesadas mediante la tarjeta de desarrollo y a la salida del sistema se tiene la señal de audio libre de ruido.

Las pruebas se realizaron con diferentes tipos de señales, la señal de entrada está dentro de la banda de frecuencias audibles, la señal utilizada como ruido será tomada desde un generador de funciones para este se empleara una señal sinusoidal, mediante el proceso de filtrado adaptativo se consigue un considerable porcentaje de cancelación del ruido en señales de audio.

UNIVERSIDAD NACIONAL DE CHIMBORAZO
FACULTAD DE INGENIERIA
CENTRO DE IDIOMAS

Lic. Geovanny Armas

04 de Septiembre del 2016

SUMMARY

This research work is the design and implementation of an adaptive filter using a Wiener filter in order to cancel the noise present in the audio signals. This adaptive filter is programmed into the DISCOVERY KIT development card, this device comprises an ARMCORTEX M7 STM32F746G processor, it will allow the signal processing, it is programmed in C Programming Language from Keil uVision5 and STM32CubeMX, for mathematical adaptation the theory of Wiener is used by means of a LMS (Least-Mean-Square algorithm) adaptive algorithm.

The system is made up by two inputs, one of them will be used as audio input and the other will be the noise input, the signals are processed by means of the development card and at the system output there is a noise-free audio signal.

The tests were carried out with several types of signals, the input signal is within the band of audible frequencies, the signal used as noise will be taken from a function generator, for this a sinusoidal signal is used. Through the process of adaptive filtering, a considerable percentage of noise cancellation in audio signals is gotten.

CENTRO DE IDIOMAS INSTITUCIONAL
UNIVERSIDAD NACIONAL DE CHIMBORAZO
RIOBAMBA ECUADOR
DIRECCION

INTRODUCCIÓN

El ruido se considera como un fenómeno universal que tiene una diversidad de orígenes que son generadas desde fuentes acústicas, hasta fuentes eléctricas. Se define como la sensación auditiva desagradable para el oído que interfiere en la comunicación entre las personas o canales de recepción.

Un problema que se presenta a diario en las salas de conferencias o entornos de discursos es la presencia de ruido, el cual puede resultar bastante molesto y frustrante, se puede generar de diferentes maneras, proveniente de equipos externos o del medio ambiente dando como resultado la contaminación de ruido en la señal original, hoy en día existen diferentes métodos para reducir o cancelar el ruido en las señales de audio.

El método utilizado en este proyecto es la aplicación de filtros adaptativos mediante filtros Wiener, que tiene como principal propósito cancelar el ruido presente en la señal de entrada de tal modo que la salida de esta sea lo más parecida a la señal original esto se realiza en base a un algoritmo matemático que es procesada por un dispositivo ARM CORTEX-M7 y gracias al software de programación MATLAB permitió simular el algoritmo matemático de forma gráfica y auditiva.

El proyecto de investigación tiene como principal objetivo el diseño e implementación un filtro adaptativo mediante filtros Wiener para la cancelación de ruido en señales de audio que se genera por la diafonía de los micrófonos, con el filtro Wiener se logrará cancelar el ruido en señales de audio, dicha hipótesis será corroborada o denegada con el desarrollo del proyecto.

CAPÍTULO I

1. FUNDAMENTACIÓN TEÓRICA

1.1. ANTECEDENTES DEL TEMA

Dentro del ámbito de filtros adaptativos existen varios trabajos investigativos nacionales e internacionales que han brindado un aporte importante en esta área.

Dentro de estos podemos mencionar los siguientes:

El proyecto realizado en enero del 2010, presentado por Walter Leopoldo Zelaya Chicas un diseño de un filtro adaptativo como cancelador de ruido basado en algoritmos LMS, las técnicas que emplea en este proyecto es un filtro Wiener basado en algoritmos LMS. La utilización de algoritmos LMS permitirá encontrar la función matemática que permita extraer los datos de interés como es la reducción de ruido acústico en voz. El proyecto detalla la realización de la cancelación del ruido acústico en voz producida por la cabina de un piloto de una nave militar, como se conoce el ruido afecta considerablemente la voz por lo que resulta muy difícil la comunicación entre pilotos dentro de esta cabina, utilizó un método de descomposición que trata de separar la voz del ruido (referencia) ocasionado por la nave militar empleando filtros Wiener basado en algoritmos LMS en donde utiliza un micrófono que es puesta en la parte trasera del casco del piloto. La simulación del filtro fue realizado en Matlab en el cual permitió visualizar mediante gráficas que tan eficiente resultó el algoritmo LMS y si es el indicado.

Giovanni Said Simón diseñó un Filtro Adaptativo y su aplicación la realizó en el reconocimiento de palabras aisladas para el control de un equipo de sonido por medio de la voz, utiliza para conseguir su objetivo filtros adaptativos ya que los algoritmos de reconocimiento de voz son muy frágiles y tienen una buena eficiencia solo en ambientes sin presencia de ruido, esta investigación intenta mejorar el porcentaje de aciertos de un sistema de reconocimiento ante la presencia de ruido, en su implementación utiliza una tarjeta de desarrollo TMS320C6711 que permite un control en tiempo real, luego de

realizar el proyecto concluye que una distancia adecuada entre el micrófono de referencia y los parlantes es entre 50 y 70cms, no se deben acercar demasiado pues los coeficientes del filtro se pueden elevar sin control produciendo desestabilización y por lo tanto un incorrecto funcionamiento.

Miguel Castillo previo a obtener su título como Ingeniero en Electrónica y Telecomunicaciones realizó una investigación acerca del tratamiento de la señal de voz usando filtros adaptativos para ambientes altamente ruidosos, el cual propone un sistema para el mejoramiento de la señal de voz, con el fin de disminuir el ruido durante el chequeo e inspección de una aeronave, dado que impide la correcta comunicación por radio entre los operadores y los pilotos, debido a la flexibilidad, robustez y versatilidad del procesamiento digital en señales acústicas, se puede decir que el diseño propuesto en este trabajo, puede ser usado para otras aplicaciones diferentes a la cancelación del ruido como por ejemplo en la eliminación de eco acústico, codificación de audio, ruido presente en auditorios, entre otros.

El proyecto de filtraje adaptativo elaborado por Rolando Pedrozo Félix se propuso a través de la utilización de un filtro adaptativo encontrar la mejor función de transferencia de un sistema lineal que sea capaz de separar a la señal del ruido. Su proyecto se basa principalmente en cancelación de ruido; supresión del ritmo cardiaco de la madre en un embarazo, se desea medir el ritmo cardiaco (electro cardiograma) de un feto durante un parto, esta señal puede ser grabada por un sensor ubicada en la región abdominal de la madre, sin embargo esta señal está alterada por ruido o distorsionada considerablemente por los latidos del corazón de la madre y además de los movimientos del feto, la idea consistiría en cancelar el ruido sustrayéndola a la señal del ritmo cardiaco del feto la señal de los latidos de la madre. Para ello se tiene que utilizar varios sensores en el abdomen, pecho y un sensor neutral que tiene por finalidad tener como referencia las señales que se desea eliminar. Las pruebas realizadas demostraron las ventajas que pueden brindar, pero no tuvieron los resultados esperados debido a que utilizó filtros analógicos adaptativos.

El proyecto realizado por el Ing. Vázquez Burgos Luis Santiago que trata de un filtro adaptativo difuso con un DSP TMS320C6713. La característica principal de este proyecto es implementar un filtro adaptativo difuso mediante hardware que se adapte a cambios de señales no lineales con la utilización de algoritmos LMS, el DSP TMS320C6713 es un dispositivo que pertenece a TEXAS INSTRUMEN son utilizados específicamente para el procesamiento de señales digitales y poseen características especiales como procesar la información en un mismo ciclo de reloj, puede operar a una frecuencia máxima de 225 Mhz, además somete a la señal de entrada a un filtro anti alias el cual elimina las componentes de frecuencia mayores a la del muestreo. Las pruebas fueron realizadas con señales de audio las cuales a través del analizador de espectros determinaron los resultados obtenidos, se utilizó el software de simulación MATLAB el cual permitió comparar el algoritmo LMS fue el adecuado en utilizar.

1.2. Presencia de ruido en ambientes cerrados

1.2.1. El ruido

En el ámbito de las telecomunicaciones el ruido es un factor que afecta de manera directa a un medio de comunicación, dicho en otras palabras el ruido es una señal no deseada, es un elemento independiente de la señal pero como consecuencia puede afectar la calidad de la misma.

Para cancelar el ruido y mejorar la calidad de las comunicaciones se han desarrollado diversas técnicas de procesamiento de señal, dentro de los cuales se puede mencionar los filtros adaptativos mediante filtros Wiener que es una técnica de varias que existen hoy en día, es importante considerar los tipos de ruidos existentes en los canales de comunicación.

De este modo, podemos definir los siguientes tipos de ruido:

- **Ruido Aditivo.-** En este caso, el ruido aditivo se puede considerar todo aquel ruido procedente de distintas fuentes que coexistan en el mismo entorno acústico.

- **Señales interferentes.-** En el caso de señales de voz, se considera señal interferente a toda aquella que proceda de otros locutores, que no sean objeto de interés.

- **Reverberación.-** Producida por la propagación multitrayectoria que se dan en los entornos acústicos cerrados o semicerrados. No se trata exactamente de ruido, sino de una forma de distorsión.

- **Eco.-** Producida generalmente por el acoplamiento entre los micrófonos y los altavoces. Igual que en el caso anterior se trata de una forma de distorsión.

- **Diafonía.-** Sonido indeseado que se origina en el receptor (Micrófono), y es la consecuencia del acoplamiento de este canal con otro, permite el paso de señales del mismo origen.

1.2.1.1. Tipos de ruido a ser cancelados.

El ruido es una onda que va desde el origen hacia un aparato receptor que detecta ondas. El ruido es captado fácilmente por el oído humano, los parámetros que lo caracterizan son:

- **La frecuencia.-** Determina si esta señal es aguda o más grave, la frecuencia es expresada en Hertz.

- **El nivel o la intensidad.-** Si los sonidos son más fuertes o débiles, se la expresa en decibeles (dB).

- **Duración.-** Si esta es breves o continua, se mide en función del tiempo.

Danny Martínez (2010) explica que "él oído humano no percibe igual todas las frecuencias solo entre el rango que va desde 20 Hz - hasta los 20000 Hz".

1.2.1.2. Ruido Blanco

Se puede describir al ruido blanco como el menos predecible y el que se genera de manera natural. El ruido blanco además de las características mencionadas, sus valores en dos momentos cual quiera no es correlativo. El ruido es motivado a tener una potencia de densidad de espectro plana

(en potencia de señal por hertzio de ancho de banda), y su parecido análogo a la luz blanca que tiene una potencia de densidad de espectro plana con respecto a la longitud de onda, de ahí su nombre. (Cruz, 2010)

1.2.1.3. Ruido Periódico.

Se conoce como ruido periódico, cuando sus componentes de frecuencia se encuentran establecidas en un rango de banda angosta. En donde se puede presentar este tipo de ruido es generalmente en motores de revoluciones periódicas, interferencias producidas por radiofónicas o ruido por las redes eléctricas. Esta clase de ruido son considerados como fácil de eliminar, siempre y cuando no se encuentren en el mismo rango de las componentes de frecuencias de la señal de interés, si esto no es así resultara más complicada la eliminación de la misma.

1.2.1.4. Relación Señal / Ruido.

La relación entre señal/ruido dependerá mucho de lo que ingrese, es decir cuan mayor sea la señal y menor el ruido a su salida se obtendrá la señal que se desee. El problema surge cuando todo esto es lo contrario, si el nivel de señal es bajo y el nivel de ruido es alto, la relación será menor y por tanto a su salida obtendremos ruido.

1.3. Filtros digitales

Los filtros en sistemas de control forman una interconexión al que se lo denominada sistema, de tal manera que el arreglo resultante sea capaz de controlarse por sí mismo. Los filtros son empleados principalmente para atenuar ruido o señales indeseables tanto en amplitud como en fase. Los filtros digitales poseen la característica de ser usados para modificar el espectro de una señal, mediante el uso de hardware o un proceso computacional (algorítmico), permitiendo de esta manera ser más efectivos, consistentes y precisos. Los filtros digitales realizan la función del filtro a través

de algoritmos numéricos. El proceso es realizado mediante el diagrama de bloques especificado en la Figura. 1. (Domingo, 2012)

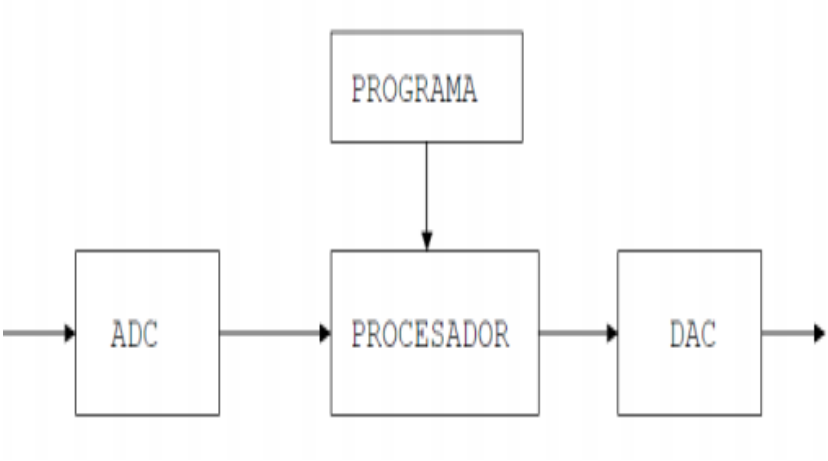


Figura. 1 Diagrama del proceso adaptativo
Fuente: David Millán Domingo.

1.4. Filtro adaptativo

Los filtros adaptativos son sistemas que se adaptan de manera automática a cambios no estacionarios de su entorno. Pueden realizar cambios en su estructura interna para ajustarse a la respuesta esperada, después de haber sido especializados en un número finito. El diagrama básico del filtro adaptativo está representado por la Figura 2. (Cruz, 2010)

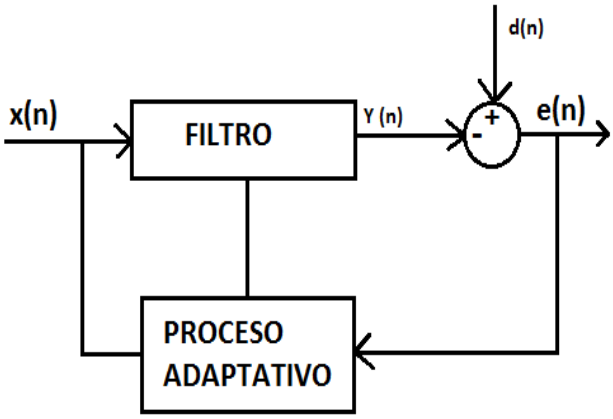


Figura. 2 Modelo de un filtro adaptativo
Fuente: Danny Pool Martínez Cruz

Las variables de la figura 2 se encuentran especificadas a continuación:

- $\boldsymbol{d(n)} = \textit{Es la señal deseada.}$
- $\boldsymbol{x(n)} = \textit{Señal que se ingresa al fitro para poder estimar } d(n).$
- $\boldsymbol{y(n)} = \textit{Salida del filtro.}$
- $\boldsymbol{e(n)} = \textit{La señal de error.}$
- $\boldsymbol{Filtro} = \textit{Proceso del filtrado.}$
- $\boldsymbol{Proceso\ adaptativo} =$
  $\textit{Algún tipo de algoritmo para adaptar los coeficientes del filtro.}$

El diseño del filtro consiste en minimizar el error de estimación, cuando la señal deseada se reduce con respecto a la señal de salida.

$$e(n) = d(n) - y(n) \quad (1)$$

1.4.1. Algoritmos LMS

Los algoritmos LMS o algoritmo mínimo cuadrado cumplen con la función de encontrar el coeficiente del filtro que es el valor mínimo del cuadrado de la señal de error. Definido como la diferencia entre la señal deseada y la señal producida en la salida del filtro. El algoritmo LMS es considerado el más simple de todos porque utiliza métodos estadísticos para cancelar el ruido que se presenta en la entrada y de esta manera obtener como resultado una señal libre de ruido.

1.4.1.1. Ventajas de los algoritmos LMS

- Posee características lineales además de ser un algoritmo sencillo de entender.
- Tiene características aleatorias en su sistema de actualización de coeficientes donde los coeficientes pasan a ser una variable aleatoria cuya media es el filtro óptimo.
- Danny Pool (2010) manifiesta que “mientras más adaptativo sea el sistema más robusto será su comportamiento y mejor su operación en ambientes estacionarios”.

1.4.1.2. Desventajas de los algoritmos LMS

- El algoritmo necesita mayor número de iteraciones para llegar a la convergencia esto se soluciona con un procesador de alta velocidad para realizar las operaciones matemáticas en el menor tiempo posible.

- El cálculo del error es inferior al de otros algoritmos adaptativos.

1.4.2. Algoritmos RLS

Los algoritmos RLS poseen una matemática más compleja, presente problemas de estabilidad numérica en comparación al algoritmo LMS, pero lo que lo diferencia de los demás es las operaciones matemáticas más que el mínimo cuadrado.

1.4.3. Algoritmo adaptativo CMA

El algoritmo adaptativo de módulo constante (CMA) es un algoritmo de igualación ciega, dicho en otras palabras que no requiere señal de referencia, ya que estos generan su propia señal de referencia desde las señales receptadas, esta es una ventaja que poseen sobre los algoritmos LMS y RLS, aunque en ocasiones no se utilice esta propiedad del algoritmo CMA, muchas veces se prefiere hacer un hibrido LMS-CMA o RLS-CMA para aprovechar las características lineales de los algoritmos LMS y RLS. (J. Alvarez, 2013)

1.4.4. Algoritmo Adaptativo DMI

Los primeros en investigar este algoritmo fueron Dailei y Blight en 1991 y por Gangass en 1994, los cuales encontraron modos de utilizarlo en circuitos eléctricos utilizando lazos, compuertas y condiciones matemáticas.

El algoritmo adaptativo de inversión de matriz directa (DMI) es un algoritmo de igualación no ciega de canal, también llamado algoritmo de matriz inversa, suele utilizarse como un algoritmo de búsqueda, en medicina es utilizado en redes neuronales al igual que el algoritmo CMA. El algoritmo DMI es un algoritmo que utiliza matrices en un lazo de

control, puede usar multicanales, en función de frecuencia, fase, etc. (J. Alvarez, 2013)

El algoritmo DMI analiza variables en un lazo de transferencia de una matriz, la matriz puede ser del tipo N por N, también puede aceptar valores negativos, puede ser usado con lazos simples o con lazos estructurados, en función de tiempo, fase, e inclusive para canales. Son utilizados en diferentes canales, para determinadas fases, y para garantizar su convergencia, ocupando una función de referencia, puede trabajar en múltiples canales en forma simultaneas independiente, en algunos casos se los calcula en zonas de seguridad para garantizar la convergencia. Algunos utilizan el algoritmo DMI en función de la frecuencia en db o rad/s o en función de fase en grados.

Sus principales ecuaciones son las siguientes:

$$R_{xx} = \sum_{i=N1}^{N2} x(i)x^{h}(i) \quad (2)$$

$$r_{dx} = \sum_{i=N1}^{N2} d^{*}(i)x(i) \quad (3)$$

$$w = R_{xx}r_{dx} \quad (4)$$

1.5. Filtro de Wiener

Los filtros Wiener son los mejores filtros lineales de mínimos cuadrados, pueden ser usados para predicción, estimación, interpolación, filtrado de señal y ruido, etc. Para diseñarlos se debe tener un conocimiento previo de señales de entrada al sistema de filtrado, el mayor problema es que la información resulta ser difícil de obtener, por esta razón los filtros adaptativos son importantes, que hacen uso de los datos de entrada para poder establecer los datos estadísticos requeridos. En su forma más general, consiste en una señal de entrada $f(k)$, una respuesta deseada $d(k)$ y un filtro lineal de respuesta impulso $h(k)$. Este filtro es

alimentado por $f(k)$ y produce a su salida $g(k)$. La diferencia entre la señal de salida del filtro $g(k)$ y la señal deseada $d(k)$, es el error de la estimación $e(k)$, tal como se muestra en la Fig.3 (L. Caiza, 2015)

El objetivo del filtrado de Wiener es determinar la respuesta impulso $h(k)$ deforma que el error $e(k)$ sea, en un sentido estadístico, "lo más pequeño posible". El criterio elegido es la minimización del valor cuadrático medio del error.

$$x = \mathrm{E}\{|\mathrm{e(n)}|^2\} \quad (5)$$

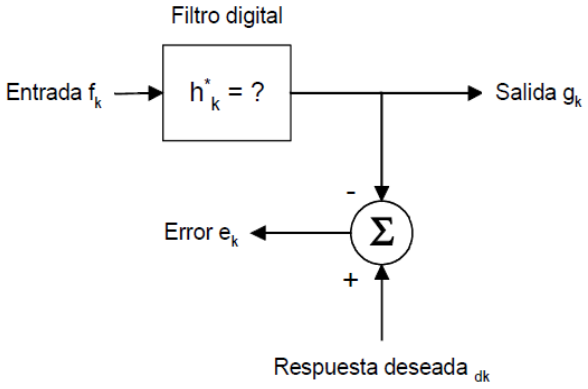


Figura. 3 Filtro Wiener
Fuente: L Caiza 2015.

El filtro digital tiene una señal de entrada y produce una señal de salida. El filtro será un filtro de Wiener si su respuesta impulsiva se elige para minimizar el error cuadrático medio. El error se define como la diferencia entre la salida del filtro y la respuesta deseada:

$$e_k = d_k - g_k \quad (6)$$

Cuando se trabaja con filtros de Wiener, generalmente la respuesta deseada existe sólo de forma conceptual. Las propiedades estadísticas de la respuesta deseada y sus relaciones estadísticas con la señal de entrada al filtro se asume que son conocidas por el diseñador. La respuesta impulsiva del filtro de Wiener se obtiene encontrando una expresión para el error

cuadrático medio y minimizándola con respecto a la respuesta impulsiva. Elevando al cuadrado en ambas partes, se obtiene: (L. Caiza, 2015)

$$e_k^2 = d_k^2 + g_k^2 - 2d_k g_k \quad (7)$$

Sabiendo que:

$$g_k = \sum_{l=0}^{\infty} f_{k-l} h_l \quad (8)$$

Se puede sustituir:

$$e_k^2 = d_k^2 + \sum_{l=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} h_l h_\mu f_{k-l} f_{k-\mu} - 2\sum_{l=-\infty}^{\infty} h_l f_{k-l} d_k \quad (9)$$

Si tomamos valor medio en ambos lados, encontramos una expresión para el error cuadrático medio (MSE, mean square error):

$$E\left[e_k^2\right] = E\left[d_k^2\right] + \sum_{l=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} h_l h_\mu E\left[f_{k-l} f_{k-\mu}\right] - 2\sum_{l=-\infty}^{\infty} h_l E\left[f_{k-l} d_k\right] = \quad (10)$$

$$= \phi_{dd}(0) + \sum_{l=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} h_l h_\mu \phi_{ff}(l-\mu) - 2\sum_{l=-\infty}^{\infty} h_l \phi_{fd}(l) \quad (11)$$

Siendo Φmm la auto correlación y Φmn la correlación cruzada de dos señales m y n. si derivamos con respecto a h, que es la respuesta impulsiva del filtro, e igualamos a cero para minimizar el error: (Cruz, 2010)

$$\sum_{l=-\infty}^{\infty} h_l^* \phi_{ff}(j-l) = \phi_{fd}(j) \quad (12)$$

Esta es la ecuación de Wiener-Hopf, que en forma de convolución queda:

$$h_l^* * \phi_{ff}(k) = \phi_{fd}(k) \quad (13)$$

Tomando transformada Z en ambas partes:

$$H^*(z)\Phi_{ff}(z) = \Phi_{fd}(z) \quad ó \quad H^*(z) = \frac{\Phi_{fd}(z)}{\Phi_{ff}(z)} \quad (14)$$

Con la solución de Wiener se puede encontrar la función de transferencia del filtro H*(z) a partir de la transformada z de la función de autocorrelación de la señal de entrada, de la correlación cruzada de la señal de entrada, y de la respuesta deseada. Si se sustituye esta ecuación en la expresión del error se obtiene el valor del mínimo MSE: (J. Alvarez, 2013)

$$E\left[e_k^2\right]\Big|_{min} = \phi_{dd}(0) - \sum_{l=-\infty}^{\infty} h_l^* \phi_{fd}(l) \quad (15)$$

1.5.1. Filtro de Wiener como filtro FIR

El filtro Wiener es FIR con M coeficientes.

$$\boldsymbol{J(w)} = \sigma^2 - pH\, w - w\, pH + H\, R\, w \quad (16)$$

p es la correlación cruzada entre la entrada y la señal deseada,

$$p = E\{\, u[n]\, d[n]\,\} \quad (17)$$

Con

$$u[n] = [u[n]u[n-1] \ldots.\ u[n-M+1]\,]\,T \quad (18)$$

Donde

$$p[-k] = E\{\, u[n-k]\, d*[n]\,\} \quad (19)$$

R es la matriz de auto correlación de la entrada:

$$R = E\{u[n]uH[n]\} \quad (20)$$

Los coeficientes del filtro óptimo que minimizan el error cuadrático medio cumplen que

$$\nabla wJ = 0 \quad (21)$$

Pablo Mosé (2015) explica "Lo que conduce al sistema M × M denominado ecuaciones de Wiener-Hopf."

$$Rw0 = p \quad \leftrightarrow \quad w0 = R - 1p \quad (22)$$

1.5.2. CANCELADOR DE RUIDO

La cancelación adaptativa de ruido está basada en la teoría de Wiener del filtro optimo, cuyo principal objetivo es eliminar el ruido de fondo que acompaña a la señal principal, es importante señalar que dicho ruido esta currelado con la señal de referencia. La utilización de la teoría de Wiener para la cancelación adaptativa de ruido requiere un filtro con un número infinito de pasos para minimizar el error. Para que la solución Wiener sea realizable, se debe construir un filtro con un número finito de pesos, es decir un filtro FIR. (L. Caiza, 2015)

En la figura. 4 la señal s(n) está compuesta por la señal deseada x(n) y una componente de ruido r(n), que no está correlado con la señal deseada. La señal de referencia del filtro ref(n), puede esta correlada con la señal deseada o bien con ruido. En el caso de la figura 4, la señal de referencia se toma de la misma fuente que el ruido que contamina la señal deseada. A la salida el filtro adaptativo se obtiene una estimación del ruido r(n) que contamina la señal deseada. Si el filtro es de orden M, r(n) es. (Pablo Mosè, 2015)

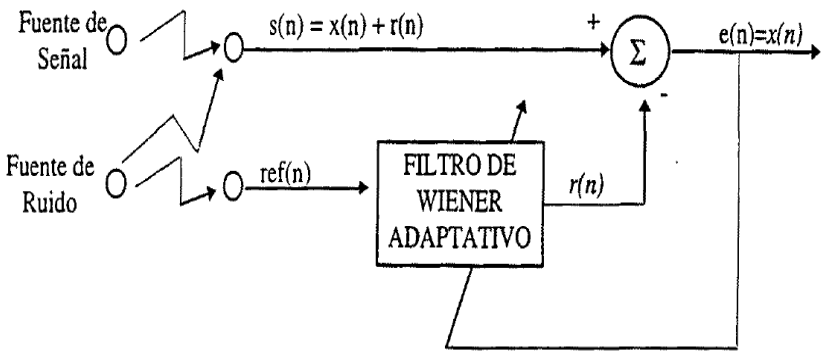


Figura. 4 Cancelador de Ruido Adaptativo
Fuente: (Fernàndez, 2012)

$$r(n) = \sum_{m=0}^{M} W_m\,(n) ref(n-m) \tag{23}$$

Donde $W_{\boldsymbol{m}}(\boldsymbol{n})$ son los pesos para el filtro adaptativo para la muestra n.

La estimación del ruido se resta a s(n) para obtener una estimación de la señal deseada x(n), que en este caso es la señal de error e(n). La señal de error a la salida el filtro es

$$e(n) = s(n) - r(n) = s(n) - W^T\,(n) r(n) \tag{24}$$

Donde

$$\text{W(n)=} \begin{bmatrix} W_0(n) \\ W_1(n) \\ . \\ W_2(n) \end{bmatrix} \qquad \text{r(n)=} \begin{bmatrix} r(n) \\ r(n-1) \\ . \\ r(n-M) \end{bmatrix} \tag{25}$$

Minimizando la expresión del error cuadrático medio respecto al peso de los filtros, se obtiene la ecuación para la obtención de dichos pesos.

Donde R es la matriz de autocorrelación de la señal de referencia y P la correlación entre la señal de entrada s(n) y la señal de referencia.

$$\frac{\partial E(e(n)^2)}{\partial w} = \begin{bmatrix} \frac{\partial E(e(n)^2)}{\partial w} \\ . \\ \frac{\partial E(e(n)^2)}{\partial M} \end{bmatrix} = -2P + 2R_w; \qquad w = R^{-1}P \tag{26}$$

La obtención del vector pesos del filtro mediante la ecuación requiere el conocimiento previo de R y P, y muchas veces esta información no se logra obtener. Widrow y Hoff encontraron una alternativa para hallar los pesos del filtro, basándose en el algoritmo LMS llegando a la siguiente ecuación.

$$w(n+1) = w(n) + 2\mu e(n) r(n) \tag{27}$$

Done μ es el parámetro de convergencia, que debe ser positivo y satisfacer la siguiente condición

$$0 < \mu < \frac{1}{\lambda_{max}} \tag{28}$$

Si λ_{max} es el máximo autoevaluador de la matriz R. En la práctica, dado que la matriz R es desconocida el parámetro μ se determina heurísticamente. Con un valor pequeño de μ se garantiza la convergencia, aunque a una velocidad muy lenta. Valores mayores de μ permiten una velocidad de convergencia más rápida aunque existe el peligro de divergir. (Fernàndez, 2012)

1.5.3. Adaptación del Filtro Wiener

Como se puede observar en la ecuación, para encontrar los coeficientes óptimos del filtrado Wiener se debe primero, hallar la matriz de autocorrelación Toeplitz y el vector de correlación cruzada entre la señal deseada d(n) y la señal recibida x(n); y posteriormente, deberá realizarse el filtrado entre dichos coeficientes y x(n). Para llevar a cabo este proceso, se debe conocer además el número de muestras n y seleccionar el grado del filtro p, respectivamente. En la figura. 5 se presentan, de forma general, las

etapas necesarias para la implementación del filtro Wiener. (Pablo Mosè, 2015)

$$R_x w = r_{dx} \tag{29}$$

Donde Rx es la matriz de autocorrelación Toeplitz p × p, w es el vector de coeficientes del filtro y rdx es el vector de correlación cruzada entre la señal deseada d(n) y la señal recibida x(n).

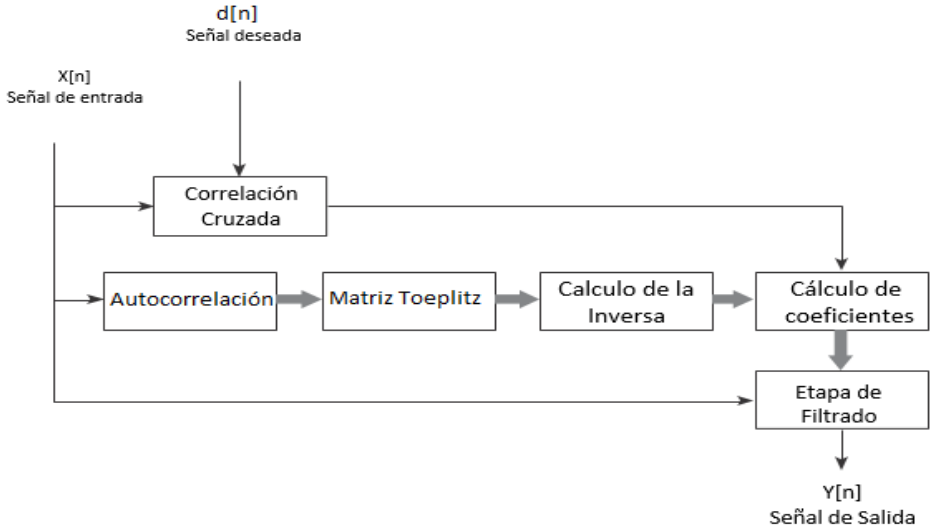


Figura. 5 Etapas del Cancelador de Ruido
Fuente: Implantación de filtros adaptativos.

1.5.4. Correlación cruzada.

Para realizar la correlación cruzada entre la señal deseada y la señal de entrada, se elaboró un algoritmo basado en la ecuación.

$$r_{xy}(1) = \sum_{n=-\infty}^{\infty} x(n+1)y(n)\ l = 0 \pm 1, \pm 2, \ldots\ldots \tag{30}$$

1.5.5. Matriz Toeplitz

En el álgebra lineal, una matriz de Toeplitz es una matriz cuadrada con todas sus diagonales, de izquierda a derecha, paralelas numéricamente.

En esta etapa se emplea la salida de la función autocorrelación, pero solo son seleccionados ciertos elementos, dependiendo del grado del filtro. Para ilustrar esta etapa se utiliza la ecuación. (Pablo Mosè, 2015)

$$\forall a_{ij} \in T \rightarrow a_{i,j} = a_{i+1,j+1} \quad (31)$$

1.5.6. Cálculo de matriz inversa.

Para encontrar los coeficientes del filtro, se calcula la matriz de autocorrelación Toeplitz, siendo necesario encontrar su matriz inversa, la siguiente ecuación queda de la siguiente manera: (Pablo Mosè, 2015)

$$R_x w = r_{dx} \quad (32)$$

Se tiene que

$$w = {R_x}^{-1}\ r_{dx} \quad (33)$$

1.5.7. Cálculo de coeficientes.

En esta etapa, simplemente se realiza una multiplicación de matrices o, más exactamente, entre un vector y una matriz.

1.5.8. Etapa de filtrado.

Luego de obtener los coeficientes del filtro, solo queda aplicarlos a la entrada x(n) para obtener la salida del filtro y(n). Dicho procedimiento se puede observar en la figura 3 y se basa en la ecuación siguiente.

$$Y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k].h[n-k] \quad (34)$$

1.6. Simulación en Matlab

Matlab es un programa que cuenta con varias herramientas matemáticas, pueden ayudar al estudiante a simular de forma real el comportamiento de señales y sistemas de control.

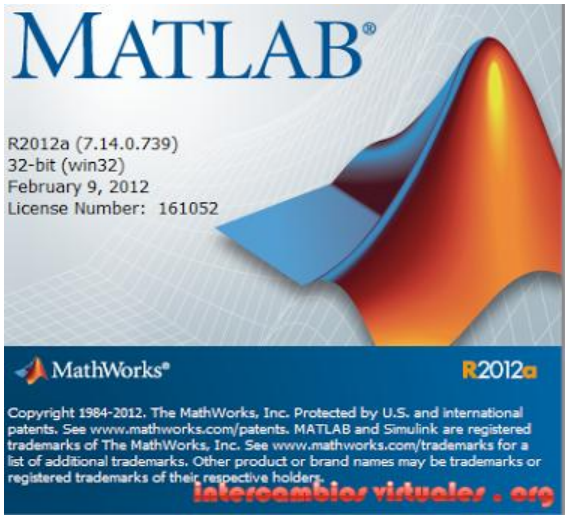


Figura. 6 Sofware Matlab
Fuente: MathWorks

Es importante el estudio minucioso del algoritmo de Wiener porque en base a las ecuaciones dadas, se procederá a realizar el cancelador de ruido, el principal propósito de Matlab es comprobar que tan eficiente resulta la ecuación matemática simulando de forma gráfica y auditiva. Para poder implementar esta aplicación en la tarjeta ARM CORTEX-32F746G de una manera más sencilla y eficiente, mediante las simulaciones se conoció el funcionamiento y los parámetros que se pueden controlar para obtener los mejores resultados en el desempeño del filtro adaptativo LMS basados en los filtros Wiener.

1.7. Procesamiento de señales digitales en la tarjeta de desarrollo ARM CORTEX

El microcontrolador STM32F746G pertenece a la familia STM Microelectronics, posee un procesador CORTEX-M7 de bajo consumo de energía, posee funcionalidad aritmética de punto flotante de características embebidas que requieren procesamiento en el menor tiempo posible.

1.7.1. Tarjeta de desarrollo STM32F746G

La tarjeta STM32F7 permite a los usuarios desarrollar y compartir aplicaciones con los microcontroladores de la Serie STM32F7 basados en ARM ® Cortex®-M7. El kit permite el descubrimiento de una amplia diversidad de aplicaciones en la que se destacan las funciones de audio, soporte de sensores múltiples, gráficos, seguridad, video y funciones de conectividad de alta velocidad. El soporte de conectividad Arduino y ARM proporciona capacidades de expansión ilimitadas con una amplia selección de complementos especializados. Este microcontrolador cuenta con cuatro I2Cs, seis SPI con tres I2S simplex multiplexados, SDMMC, cuatro USARTs UART, ADC de 12 bits, dos DAC de 12 bits, dos de las EFS, módulo de interfaz de cámara digital de 14 bits, interno 320 + 16 + 4 Kbytes de SRAM y memoria flash de 1 Mbyte, HS USB OTG, USB OTG FS, Ethernet MAC, interfaz de FMC, la interfaz Quad-SPI, soporte de depuración SWD. (Microelectronics, 2015)

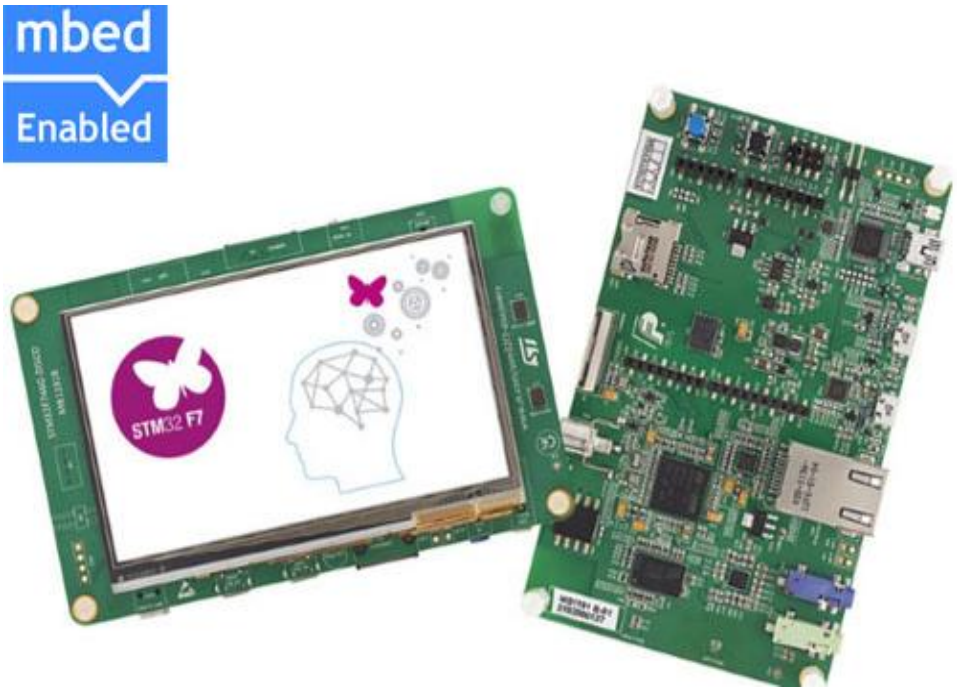


Figura. 7 STM32F746G
Fuente: Microelectronics

1.7.2. Principales Características:

- El microcontrolador STM32F746NGH6 cuenta con 1 MB de memoria flash y 340 Kbytes de memoria RAM.

- En su interior utiliza ST-LINK / V2-1 para mejorar la capacidad de re enumeración USB.
- Mbed habilitado (mbed.org).
- Funciones de USB: puerto COM virtual, almacenamiento masivo, puerto de depuración.
- De 4,3 pulgadas a color LCD de 480x272-TFT con pantalla táctil capacitiva.
- Conector de la cámara
- Códec de audio EFS
- Entrada de audio y salida de línea jack
- Salidas de los altavoces estéreo
- Dos micrófonos MEMS ST
- Conector de entrada RCA SPDIF
- Dos pulsadores (usuario y restablecer)
- Memoria de 128 Mbit Quad-SPI flash
- 128 Mbit SDRAM (64 Mbits accesible)
- Conector para tarjeta microSD
- Conector de placa hija-EEPROM RF
- USB OTG HS con conectores micro-AB
- FS USB OTG con conectores micro-AB
- Conector Ethernet compatible con IEEE-802.3-2002
- Cinco opciones de suministro de energía:
  - ST LINK / V2-1
  - Conector USB FS
  - Conector USB HS
  - VIN del conector Arduino
  - Externo 5 V del conector
- Salida de alimentación para aplicaciones externas:
- 3,3 V o 5 V
- Conectores Arduino Uno V3
- Software libre integral que incluye una variedad de ejemplos, parte del paquete STM32Cube

- Con el apoyo de una amplia variedad de entornos de desarrollo integrado. (Microelectronics, 2015)

1.7.3. Especificaciones técnicas de la tarjeta de desarrollo.

1.7.3.1. Requisitos del sistema:

- El sistema operativo Windows (XP,7,8)
- USB tipo A a mini-B Cable

1.7.3.2. Cadenas de herramientas de desarrollo

• IAR EWARM (IAR Embedded Workbench)

• Keil® MDK-ARM ™

• IDE basado en GCC (CA6 libre: SW4STM32, Atollic® TrueSTUDIO®)

• ARM mbed ™ en línea

1.7.3.3. Software de demostración

- El software de demostración está precargado en la memoria flash STM32F746NGH6.
- Las últimas versiones del código fuente de demostración y la documentación asociada se pueden descargar desde www.st.com/stm32f7-discovery.

1.7.3.4. El diseño y la configuración de hardware.

STM32F746G-disco está diseñado en torno a la STM32F746NGH6 (216 pines paquete TFBGA). El diagrama de bloques del hardware de la figura 8 muestra la conexión entre STM32F746NGH6 y periféricos (SDRAM, memoria flash Quad-SPI, módulo de cámara, pantalla LCD en color, conectores USB OTG, USART, Ethernet, audio, SPDIFRX, tarjeta microSD, escudos Arduino Uno e incrustado ST-LINK), la Figura .9 ayudarán a

localizar estas características en el tablero de descubrimiento real. (Microelectronics, 2015)

1.7.4. Características de la tarjeta.

- Audio Estéreo Speaker JP3 y JP4.
- Las bandas salidas de audio estéreo JP3 JP4 están disponibles para apoyar altavoces estéreo (izquierda y derecha).
- Toma verde de audio (salida de línea) CN10
- A3.5mm salida de audio jack estéreo verde CN10 está disponible para apoyar los auriculares.
- Jack de audio azul (línea A) CN11
- Un estéreo de audio Jack de entrada de 3,5 mm azul CN11 está disponible para apoyar la entrada de audio de línea.
- microcontrolador STM32F746NGH6 con 1 MB de memoria flash y 340 Kbytes de memoria RAM, en BGA216 paquete
- A bordo ST-LINK / V2-1 apoyar la capacidad de re-enumeración USB
- Mbed habilitado (mbed.org)
- Funciones de USB: puerto COM virtual, almacenamiento masivo, puerto de depuración
- De 4,3 pulgadas a color LCD de 480x272-TFT con pantalla táctil capacitiva
- conector de la cámara
- códec de audio EFS
- línea de audio de entrada y salida de línea
- salidas de los altavoces estéreo
- Dos micrófonos MEMS ST
- conector de entrada RCA SPDIF
- Dos pulsadores (usuario y reset)
- memoria de 128 Mbit Quad-SPI flash

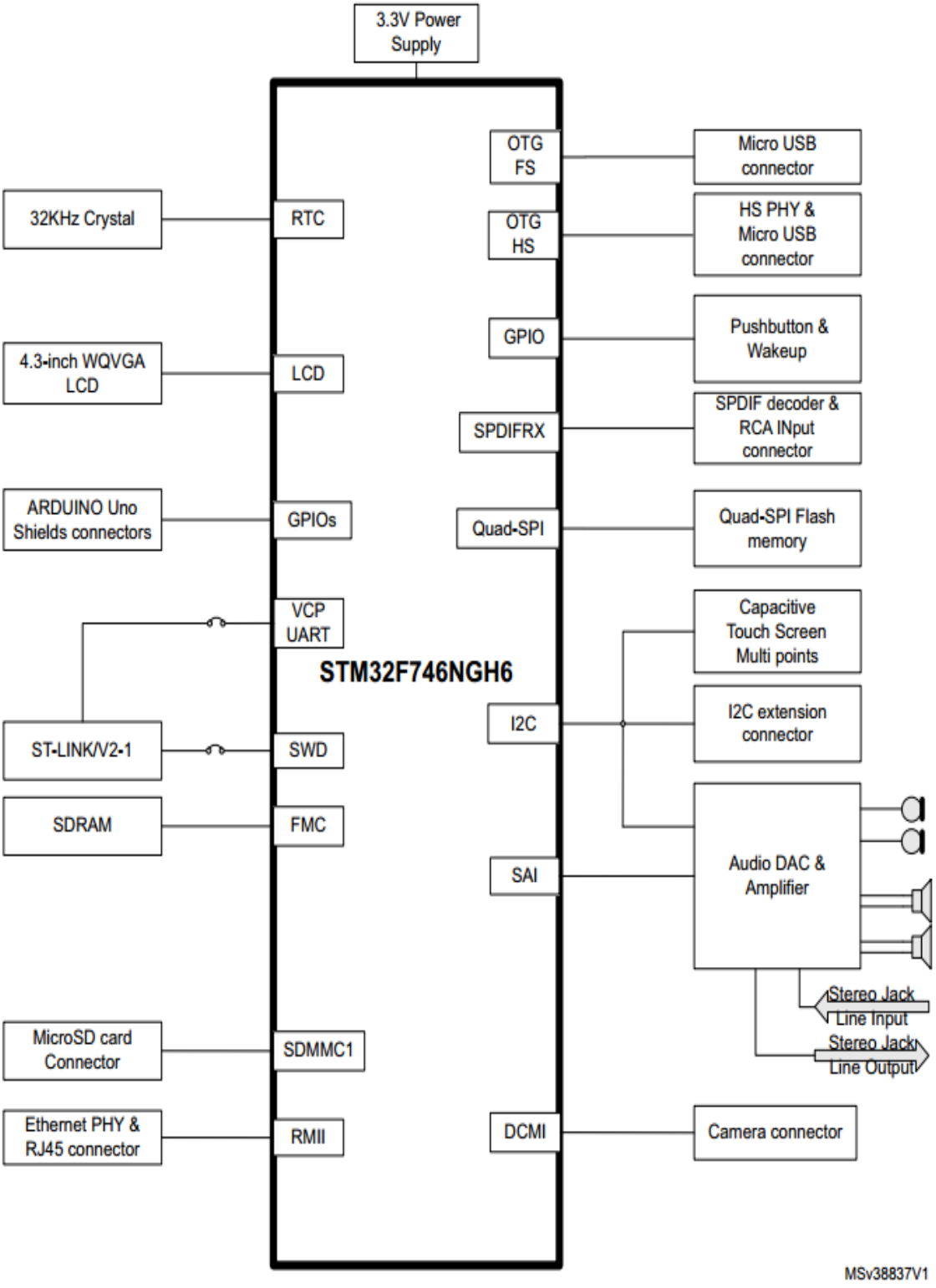


Figura. 8 Estructura de la tarjeta en bloques
Fuente: ST Microelectronics.

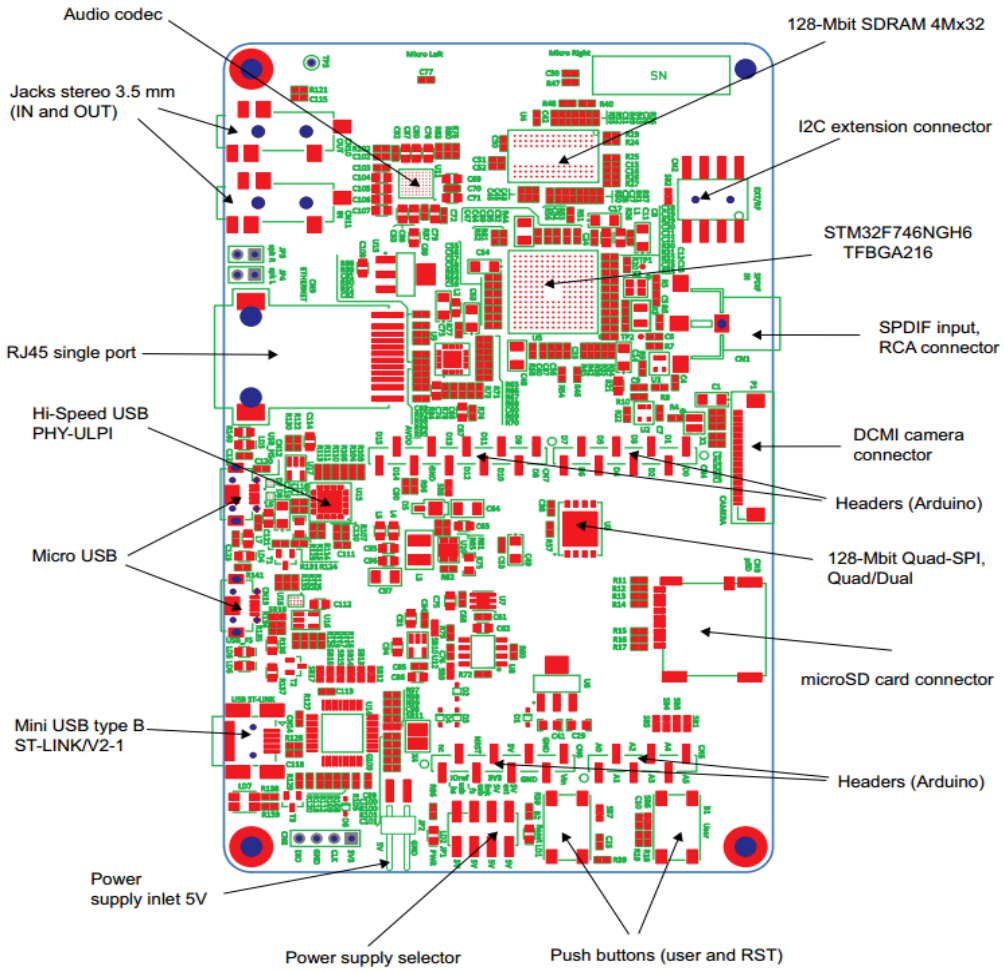


Figura. 9 Descripción de entradas y salidas
Fuente: ST Microelectronics.

1.7.5. Software de programación para dispositivos STM

Existen diferentes software de programación para los dispositivos STM, entre ellos se puede mencionar Astro Visión LifeSign Mini, Zen Visión, NVIDIA 3D Visión Video, Keil Uvisión, etc.

El software Keil Uvisión no posee ningún costo monetario y es el más fácil de utilizar, fue desarrollado originalmente por ARM y está disponible para diferentes sistemas operativos como por ejemplo Window XP/Vista/7/8/10, lo que se resalta de este entorno de programación que se desarrolla y depura en tarjetas STM de 32-bit con un procesador Cortex, su programación es de nivel alto dicho en otras palabras lenguaje C, poseen diferentes bloques e interfaces de conexión con el mundo exterior como se indica en la siguiente figura. 10.

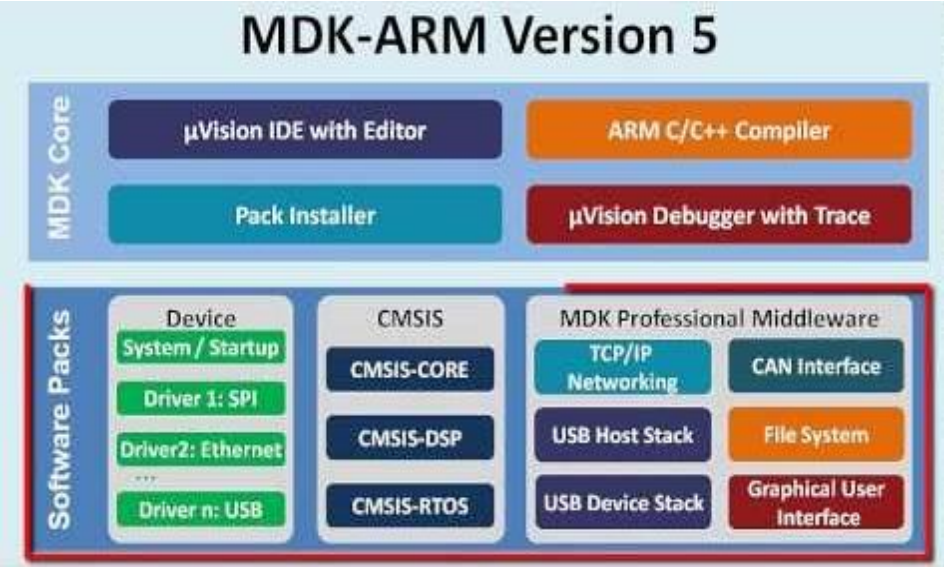


Figura. 10 Kit MDK
Fuente: ST Microelectronics.

Adicionalmente se complementa con un programa llamado STM32CubeMX que es una herramienta gráfica que ayuda a configurar cualquier STM32 generando la inicialización de un código en C visualizando de esta manera las variables en tiempo de ejecución y herramientas de ayuda en la depuración puesta a punto y validación de las solicitudes.

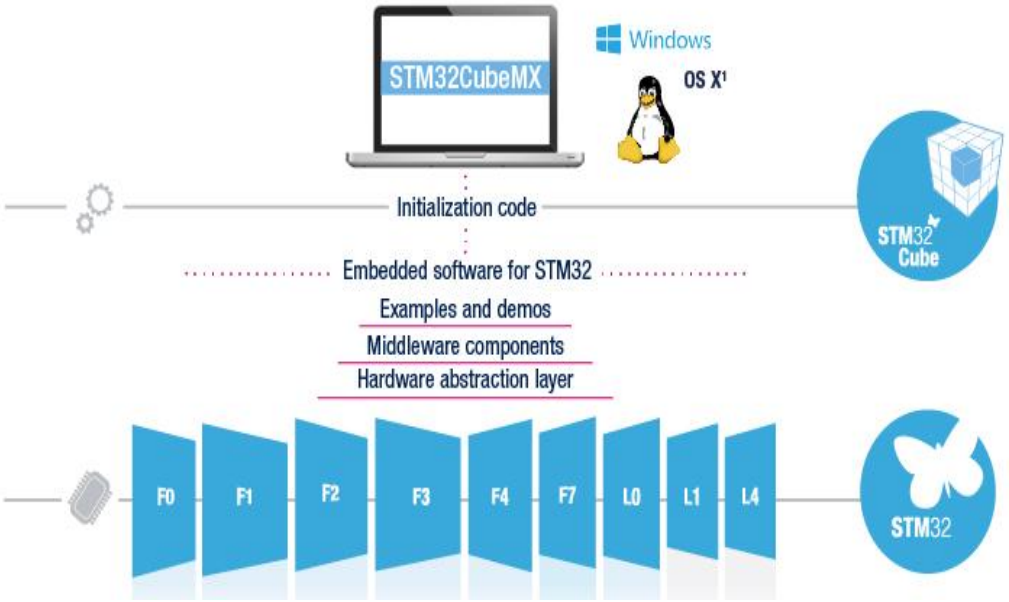


Figura. 11 Sofware stm32Cube MX
Fuente: ST Microelectronics

Los lenguajes tradicionales de C y C++ son aplicadas en STM32 ahora también su programación se puede desarrollar en Java y mezclarla con código C la interfaz que presenta es STM32CubeMX se muestra en la siguiente figura.12.

Figura. 12 Interfaz del STM32CubeMX
Fuente: ST Microelectronics

- Microcontrolador de configuración gráfica.
- Pinout con la resolución de conflictos automática.
- Árbol de reloj con la validación de configuración dinámica de periféricos y modos funcionales de middleware y la inicialización con la validación dinámica de las limitaciones de los parámetros.
- Secuencia de encendido con la estimación de los resultados de consumo.
- Proyecto de generación de código de inicialización cubierta STM32 microcontrolador compatible con IAR ™, Keil ™ y compiladores GCC.
- Disponible como un software independiente que se ejecuta en Windows, Linux y OS X ® (OS X ® es una marca registrada de Apple Inc., registrada en los países de Estados Unidos y otros) sistemas operativos, o por medio de plug-in de Eclipse. (Microelectronics, 2015)

CAPÍTULO II

2. METODOLOGÍA

2.1. Tipo De Estudio

2.1.1. Nivel de Investigación

2.1.2. Tipo de Investigación. - Se realizará una investigación experimental, debido a que se caracteriza el diseño y la implementación del proyecto de investigación, existe muy poca información de la implementación de filtros adaptativos Wiener en el tratamiento de señales de audio.

2.1.3. Diseño de la Investigación. - Es predictiva experimental debido a que se plantea el posible efecto o consecuencia de la implementación de un filtro adaptativo mediante filtros Wiener.

2.1.4. Técnicas e Instrumentación

2.2. Técnicas

Las técnicas usadas en este proyecto de investigación son la observación indirecta y la comprobación auditiva, es decir podemos visualizar los instrumentos utilizados en la investigación y escuchar la cancelación del ruido mediante la implementación de un filtro adaptativo mediante filtros Wiener implementados en la tarjeta STM32F746G.

2.2.1. Instrumentación

Los instrumentos necesarios son libros, archivos, páginas web y datasheet.

2.3. Población y Muestra

2.3.1. Población

La población comprende todas las señales de audio contaminadas con ruido que se pueden procesar y tratar, se generan por diferentes fuentes acústicas, afectando la calidad de la señal en entornos pre-grabados y en tiempo real.

2.3.2. Muestra

La muestra se determina mediante muestro intencional, es decir, se establece por criterio de los autores.

2.4. Hipótesis

Con el diseño e implementación de un filtro adaptativo mediante filtros WIENER se logrará cancelar el ruido en señales de audio.

2.5. Operacionalización de las variables

VARIABLE	CONCEPTO VARIABLE	DIMENSIONES	INDICADORES	INSTRUMENTOS
Variable independiente Diseño e Implementación de un filtro adaptativo mediante filtros Wiener	El diseño es un método específico para poder adaptar un modelo matemático partiendo de un problema determinado para resolverlo. La Implementación es poner en funcionamiento o llevar a cabo un proyecto determinado	Diseño matemático Diseño estructural Implementación del algoritmo.	- Algoritmo que mejor se adapte al filtro Wiener. - Número de coeficientes del filtro. - Tipo de lenguaje que será utilizado en la programación del dispositivo. - Determinación del porcentaje de error en el sistema. - Extraer la información o datos de interés.	- Software de simulación. - Compilador en lenguaje C.

Variable dependiente Permitirá cancelar el ruido en señales de audio.	Utilizar un método apropiado para la cancelación de ruido en señales de audio.	Porcentaje de atenuación del ruido respecto a la señal original.	- La relación señal a ruido.	- Osciloscopio.

Tabla 1. Operacionalización de las variables.
Fuente: Autores.

2.6. Procedimientos

El presente proyecto de investigación se desarrolla en cuatro etapas, las mismas que se describen a continuación en la figura 13.

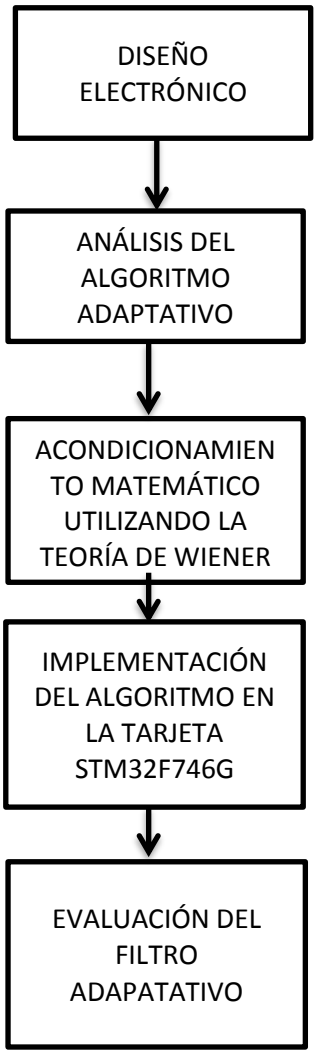


Figura. 13 Etapas de desarrollo.

Fuente: Autores

2.6.1. Diseño electrónico

En la figura 14 se muestra el diseño electrónico del sistema adaptativo de filtrado de ruido utilizando filtros Wiener, el sistema cuenta con dos entradas y una salida analógica, las entradas están conectadas a sujetadores de nivel, que garantizarán que las lecturas de los datos serán tomados únicamente en el ciclo positivo de la señal de referencia, estos circuitos se conectan a las entradas analógicas del microprocesador ARM-CORTEX M7 cuyos pines son PA0 correspondientes al ADC1M, el pin PA3 correspondiente al ADC3.

La salida del microprocesador se conecta al pin PA4, el cuál es la DAC del sistema, por este pin se obtiene la señal filtrada, esta señal es conectada a una etapa de amplificación en este caso a los parlantes del computador que cuentan con un amplificador interno.

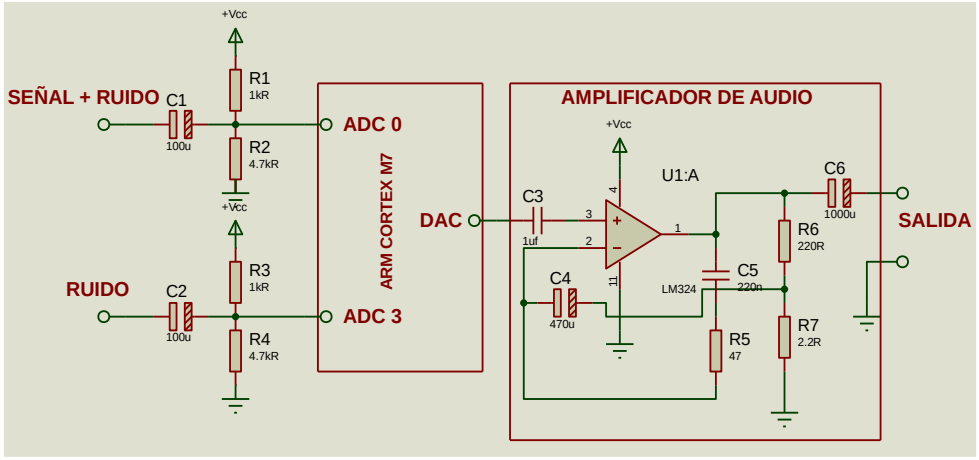


Figura. 14 Circuito electrónico.
Fuente: Autores

2.6.1.1. Diseño del sujetador de nivel

El sujetador de nivel es utilizado para desplazar la señal de audio que ingresa al ADC de la tarjeta hacia el ciclo positivo debido que el dispositivo solo toma muestra de valores que se encuentran del punto de referencia hacia arriba, los sujetadores permiten añadir un nivel de voltaje de corriente continua a un voltaje de corriente alterna a continuación se podrá observar el diseño del sujetador de nivel en la figura 15.

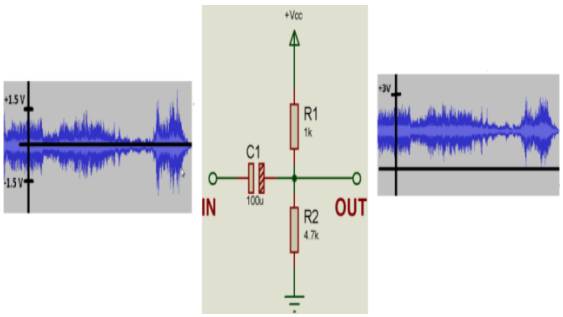


Figura. 15 Diseño del sujetador.
Fuente: Autores

2.6.2. Análisis del algoritmo adaptativo.

2.6.2.1. Filtros adaptativos LMS.

Los algoritmos LMS permitirán encontrar los coeficientes que mejor se ajusten al desarrollo del filtro de Wiener, que es el valor mínimo del cuadrado de la señal de error, dicho error está definido como la diferencia entre la señal deseada y la señal producida en la salida del filtro tal como se muestra en la figura 16.

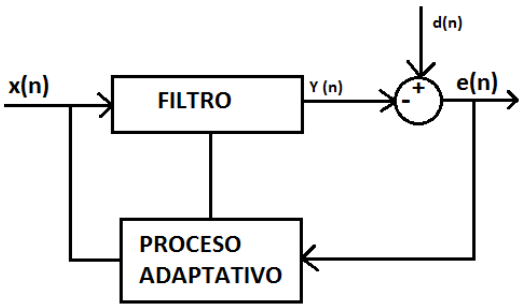


Figura. 16 Filtro adaptativo
Fuente: Danny Pool Martínez Cruz

Este modelo de adaptación matemática permite utilizar un filtro Wiener como FIR, el algoritmo LMS está regido por las siguientes ecuaciones:

$$y(n) = w^{T}(N-1)u(n) \quad (35)$$

$$e(n) = d(n) - y(n) \quad (36)$$

$$w(n) = \alpha w(n-1) + \mu e(n)u^{*}(n) \quad (37)$$

Dónde:

n= Índice de tiempo presente

u(n)= Vector de muestras de entrada asignadas según el paso n

$u^*(n)$= Conjugada compleja del vector de muestras de entrada asignadas según el paso n

w(n)= Vector del peso del filtro

y(n)= Salida Filtrada

e(n)= Error de estimación

d(n)= Respuesta deseada

μ= Tamaño de adaptación

2.6.3. Acondicionamiento matemático utilizando la teoría de Wiener

Una vez que tenemos el algoritmo que proporciona los pesos de adaptación para el filtro utilizaremos un filtro de tipo FIR, que es una restricción de causalidad del filtro de Wiener.

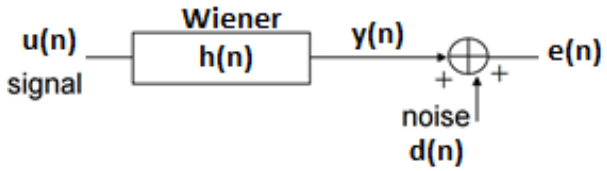


Figura. 17 Filtro Wiener FIR 1.
Fuente: David Millán.

El primer paso para el desarrollo del filtrado es calcular el error que es únicamente el resultado de una resta de muestras hasta encontrar una media que sea lo más parecida a la señal original cuya operación se muestra en la siguiente ecuación:

$$e(n)=d(n)-y(n) \quad (38)$$

Una vez calculado el error el objetivo es conseguir que este sea mínimo por el cual se aplicará la teoría de Wiener, para esto se aplica la siguiente fórmula:

$$e(n) = d(n) - \sum_{K=0}^{M} h_k \mu(n-k)$$

(39)

En donde:
k= orden del filtro
$\mu =$ pasos del filtro.

Cabe recalcar que la señal $y(n)$ es el resultado de la convolución de la señal $\mu(n)$.

Como se mencionó anteriormente el objetivo de filtrado de Wiener es minimizar al máximo el error, se debe tener encuentra que el filtro de Wiener no es un filtro adaptativo por lo cual utilizaremos métodos de adaptación LMS, para esto utilizaremos la teoría de descenso de gradiente, que permitirá la realización de la adaptación es decir iniciaremos con un filtrado deficiente, que poco a poco mejorará hasta llegar a obtener la señal deseada para esto se aplica la siguiente ecuación.

$$J(n)=e^2(n) \qquad (40)$$

$$\frac{\partial J(n)}{\partial hk(n)} = 2e(n)u(n-k) \qquad (41)$$

$$\nabla J(n) = -2e(n)\bar{\mu}(n) \qquad (42)$$

El vector $\bar{\mu}(n)$ es el vector actualizado de los datos en diferentes instantes de tiempo, por definición del algoritmo adaptativo LMS el valor de paso del filtro $u = 0.5$.

$$\bar{h}(n+1) = \bar{h}(n) - \frac{u}{2}\nabla J(n) \qquad (43)$$

Sustituyendo la ecuación 42 en la 43 tenemos:

$$\bar{h}(n+1) = \bar{h}(n) + u\, e(n)\bar{u}(n) \qquad (44)$$

La ecuación 44 corresponde al algoritmo LMS.

Una vez obtenido la ecuación adaptativa, se incluye la teoría de Wiener sustituyendo e(n) en función de d(n) y h(n) lo que da como resultado la siguiente ecuación que es la salida del filtro.

$$\bar{h}(n+1) = \bar{h}(n) + \mu\bar{\mu}(n)\, d(n) - \bar{h}(n)\bar{\mu}(n) \qquad (45)$$

Dónde:

$\bar{h}(n+1)$= vector de valores actualizados.

$\bar{h}(n)$= Vector de valores anteriores.

$\mu =$ la constante de pasos del filtro.

$\bar{\mu}(n) =$ vector de datos del filtro.

$d(n) - \bar{h}(n)\bar{\mu}(n) =$ cálculo del error.

En la figura 17 tenemos dos entradas al sistema u(n), es la entrada al sistema en este caso sería la señal contaminada, la otra entrada es d(n), que hace mención a la señal de referencia que ingresa al sistema, y debe tener el mismo tipo de datos, complejidad y dimensiones que la señal de entrada.

En el terminal de salida tenemos la señal filtrada, que es la estimación de la señal deseada. El puerto de error muestra el resultado de restar la señal de salida de la señal de entrada. En simulink el filtro LMS se muestra como en la figura 18.

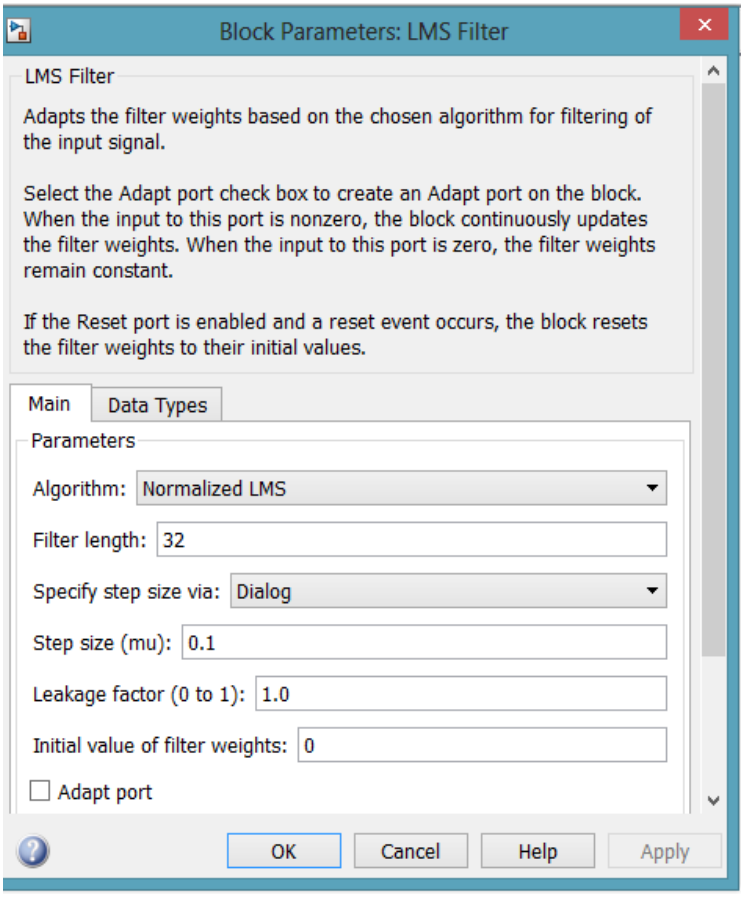


Figura. 18 Parámetros LMS
Fuente: Autores

2.6.4. Implementación del algoritmo en la tarjeta STM32F746G

La tarjeta de desarrollo STM32F746G cuenta con un procesador ARM-CORTEX M7, para la implementación del algoritmo adaptativo se utiliza el software de programación Keil uVision5 conjuntamente con el software STM32CubeMX, la explicación de la creación de un nuevo proyecto en estos dos programadores se indica en el Anexo 5.

2.6.5. Estructura de programación del algoritmo en Keil uVision5

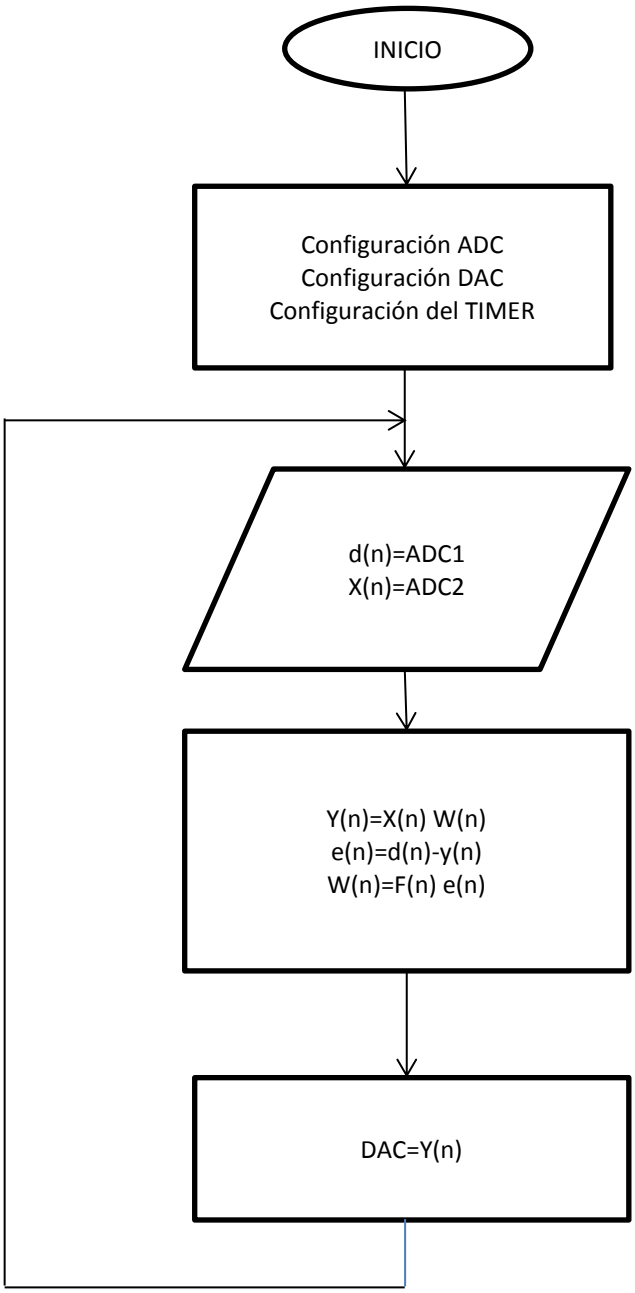


Figura. 19 Diagrama de flujo del filtro adaptativo.
Fuente: Autores

2.6.5.1. Configuración del ADC

Para configurar los parámetros de conversión analógica a digital se utiliza una resolución de 12 bits, la cual permitirá mayor procesamiento y captura de datos, además permite configurar parámetros como lectura de datos cuando se produzca la interrupción en el flanco de subida, una característica propia de este micro controlador

permite una entrada en los puertos analógicos de 3.3V como voltaje de referencia, se muestra a continuación la programación utilizada para la configuración de ADC en el software de desarrollo.

```
static void MX_ADC1_Init(void)

        {

 ADC_ChannelConfTypeDef sConfig;

          hadc1.Instance = ADC1;
          hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
          hadc1.Init.Resolution = ADC_RESOLUTION_12B;
          hadc1.Init.ScanConvMode = DISABLE;
          hadc1.Init.ContinuousConvMode = DISABLE;
          hadc1.Init.DiscontinuousConvMode = DISABLE;

    hadc1.Init.ExternalTrigConvEdge=ADC_EXTERNALTRIGCONVEDGE_RISIN
    G;
          hadc1.Init.ExternalTrigConv = ADC_EXTERNALTRIGCONV_T6_TRGO;
          hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
          hadc1.Init.NbrOfConversion = 1;
          hadc1.Init.DMAContinuousRequests = DISABLE;
          hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;


         if (HAL_ADC_Init(&hadc1) != HAL_OK)
          {

            Error_Handler();

          }

          sConfig.Channel = ADC_CHANNEL_0;
          sConfig.Rank = 1;
          sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
          if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)

          {

            Error_Handler();

          }

        }
```

2.6.5.2. Configuración de la DAC.

El sistema adaptativo una vez que realice el proceso de filtrado entregara una señal digital, a la salida del sistema se convertirá en una señal analógica para que esta pueda ser escuchada.

Al igual que en el ADC seleccionaremos una DAC de 12 bits, y además que la conversión se realizara sincronizada mente con la interrupción en este caso se utiliza el TIMER 6, a continuación se indica la configuración de la DAC en el programa KEIL.

```
/* DAC init function */
static void MX_DAC_Init(void)
{
  DAC_ChannelConfTypeDef sConfig;
    /**DAC Initialization
    */
  hdac.Instance = DAC;
  if (HAL_DAC_Init(&hdac) != HAL_OK)
  {
    Error_Handler();
  }
    /**DAC channel OUT1 config
    */
  sConfig.DAC_Trigger = DAC_TRIGGER_NONE;
  sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_ENABLE;
  if (HAL_DAC_ConfigChannel(&hdac, &sConfig, DAC_CHANNEL_1) !=
HAL_OK)
  {
```

```
    Error_Handler();

  }

}
```

2.6.5.3. Configuración del TIMER.

La implementación del filtro adaptativo utiliza un TIMER en modo de interrupción para realizar la lectura de las señales analógicas así como también la conversión de Digital a Analógico, para esto se ha seleccionado el TIMER 6 que posee la tarjeta de desarrollo.

El TIMER está configurado a una resolución de 12 bits, lo que permite tener una frecuencia de muestreo de 60 Khz, con un periodo de 16us, para esto se utilizó el fundamento teórico que indica que para asegurar una recuperación y lectura de datos correcta la frecuencia de muestreo tiene que ser 3 veces la frecuencia de la señal, a continuación se muestra la programación utilizada para configurar el TIMER6.

```
  /* TIM6 init function */

  static void MX_TIM6_Init(void)

  {

    TIM_MasterConfigTypeDef sMasterConfig;
    htim6.Instance = TIM6;
    htim6.Init.Prescaler = 0;
    htim6.Init.CounterMode = TIM_COUNTERMODE_UP;
    htim6.Init.Period = 800;
    if (HAL_TIM_Base_Init(&htim6) != HAL_OK)
    {
      Error_Handler();

    }

    sMasterConfig.MasterOutputTrigger = TIM_TRGO_UPDATE;
    sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
    if (HAL_TIMEx_MasterConfigSynchronization(&htim6, &sMasterConfig) !=
HAL_OK)
    {

      Error_Handler();

    }
```

}

2.6.5.4. Estructura de programación principal

Lo primordial en la primera línea de programación es incluir la tarjeta con la que se está trabajando.

```
#include "stm32f7xx_hal.h"
```

El sistema requiere de algunos tipos de variables que cumplan funciones específicas que se adapten al proceso. La declaración según su funcionalidad y tipo de variable viene dado por las siguientes instrucciones:

```
uint32_t In1;
uint32_t In2;
real T Out1;
```

A continuación se inicializa las configuraciones establecidas para el ADC, DAC Y TIMER.

```
/* INICIALIZO LAS PERIFERICOS DE I/O */
MX_GPIO_Init();
MX_ADC1_Init();
MX_ADC3_Init();
MX_DAC_Init();

  /* HABILITO EL ADC1 Y ADC3*/
ADC_HandleTypeDef hadc1;
ADC_HandleTypeDef hadc3;

  /* HABILITO LA DAC*/
HAL_DAC_Start (&hdac, DAC_CHANNEL_1);
```

Asignación de variables y lectura de los canales analógicos.

```
In1=HAL_ADC_GetValue(&hadc1);
In2=HAL_ADC_GetValue(&hadc3);
```

La siguiente línea de programación envía el valor hacia la DAC.

```
HAL_DAC_SetValue (&hdac,DAC_CHANNEL_1,DAC_ALIGN_8B_R,Out1);
```

Una vez establecido los comandos que se utiliza para la lectura de analógicos así como TIMER y DAC se procede a la estructura de programación del algoritmo adaptativo en lenguaje C, cabe destacar que necesitaremos librerías para la estructuración de nuestro algoritmo, estas librerías existen dentro del entorno de programación de MATLAB.

2.6.5.5. Programación del algoritmo matemático adaptativo

La figura 27 muestra la secuencia utilizada para la programación del filtro adaptativo en donde se tiene:

d(n)= Ruido.

x(n)= Señal más ruido.

e(n)= cálculo del error.

w(n)= filtrado de Wiener.

F(n)= algoritmo adaptativo LMS.

Como se puede apreciar en la implementación del filtro adaptativo

W(n) corresponde al cálculo matemático del filtro de Wiener tal como se muestra a continuación.

$$w(n) = \sum_{K=0}^{M} h_k \mu(n-k)$$

Dónde:

K= Coeficientes del filtro

$\mu =$ paso del filtro

Obtenida la ecuación del filtro de Wiener la programación en lenguaje C de dicha ecuación se muestra a continuación.

```
for (k = 0; n < (int32_T)*startIdx; n++) {
w[n] += wBuf[n1] * xBuf[n];
bufEnergy += xBuf[n] * xBuf[n];
n1++;
}
```

Esta ecuación permitirá actualizar los valores de la entrada del filtro como se muestra a continuación.

$$Y(n)=X(n)\ w(n)$$

Cuyo código de programación seria:

```
Y[n]=x[n] *w[n];
```

Una vez que se haya configurado el filtro se procede con el cálculo del error que está dada por la siguiente ecuación.

$$e(n)=d(n)-y(n)$$

A continuación se muestra el código de programación para esta ecuación.

```
e[n]=d[n] – y[n];
```

Cabe recalcar que cada valor obtenido se almacenara en una matriz, una vez calculado el error procedemos la implementación del algoritmo adaptativo como tal, para esto utilizaremos la siguiente ecuación:

$$F(n+1) = F(n) - \mu e(n)u(n)$$

F(n) será igual a la lectura de datos actualizados obtenida de la señal de entrada por lo que la secuencia de programación se presenta de la siguiente forma.

```
n1 = 0;
   for (n = (int32_T)*startIdx; k < wLen; n++)
{
    C[n1] = xBuf[n] / (bufEnergy + 2.2204460492503131E-16) * x[n] * mu + leakFac *
wBuf[j1];
    n1++;
}

   for (n = 0; j < (int32_T)*startIdx; n++)
{
    C[n1] = xBuf[n] / (bufEnergy + 2.2204460492503131E-16) * x[n] * mu + leakFac *
wBuf[n];
    n1++;
   }
  }
```

```
 n1 = wLen;
 for (n = 0; n < x(n); n++)
{
   F[n] = C(n)[n1 + 1];
   n1++;
 }
```

Finalmente se tiene que.

$$W(n)=F(n)\ e(n)$$

Lo que en código de programación representa que:

```
w[n]=F[n] * e[n];
```

2.6.6. Diseño del filtro adaptativo en Simulink.

Simulink es un entorno de simulación gráfica que permite realizar procesos matemáticos que son de alta complejidad es por esto que la simulación del filtro adaptativo se realiza en dicho en tormo, para esto se utiliza un bloque de filtro adaptativo LMS, un bloque en donde se programa la característica de filtrado de Wiener como se muestra en la figura 20.

Una vez diseñado el algoritmo adaptativo en el entorno de programación simulink se obtiene como resultado en la figura 21, se puede apreciar el error es mínimo con lo que garantizamos un filtrado casi óptimo es decir que la señal de salida es casi exacta a la señal de deseada.

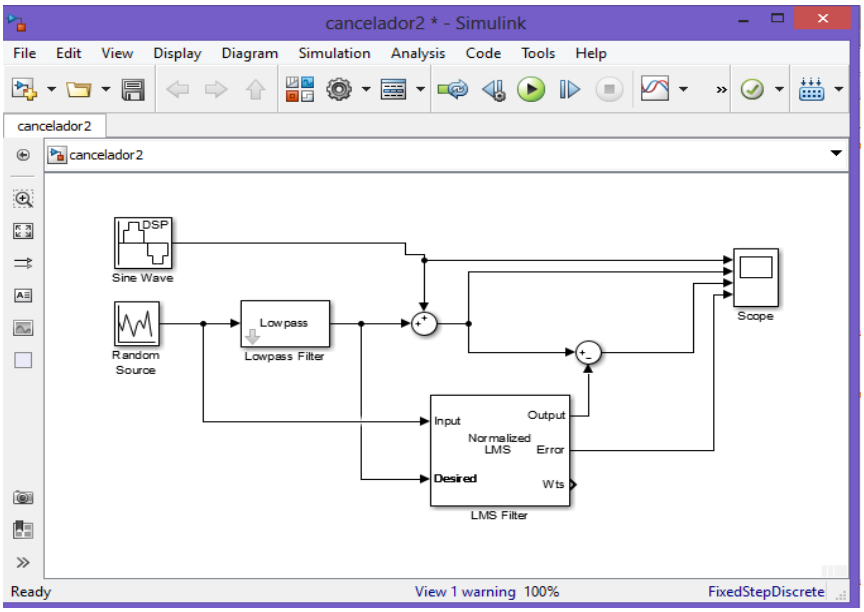


Figura. 20 Filtro Wiener en Simulink.

Fuente: Autores

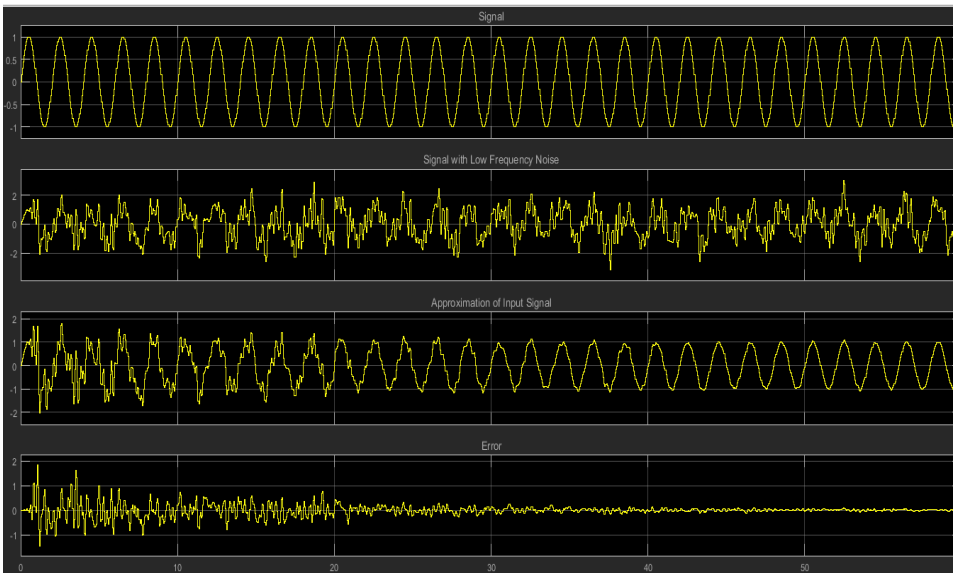


Figura. 21 Algoritmo adaptativo en Simulink.
.Fuente: Autores

2.7. Comprobación de la hipótesis.

La comprobación de la hipótesis se realizó mediante el método estadístico CHI-CUADRADO, para constatar el cumplimiento de la hipótesis.

2.7.1. Planteamiento de la hipótesis

Con el diseño e implementación de un filtro adaptativo mediante filtros WIENER se logrará cancelar el ruido en señales de audio.

2.7.2. Hipótesis Nula (Ho):

Con el diseño e implementación de un filtro adaptativo mediante filtros WIENER no se logrará cancelar el ruido en señales de audio.

2.7.3. Hipótesis Alternativa (H1):

Con el diseño e implementación de un filtro adaptativo mediante filtros WIENER no se logrará cancelar el ruido en señales de audio.

2.7.4. Establecimiento del Nivel de Significancia

Para esta investigación se eligió un nivel de significancia de 0.005, garantizando un nivel de confianza de 0.95.

Si el nivel de confianza es demasiado próximo a 1 su probabilidad de buen funcionamiento seria altísima, pero a costa de una longitud de intervalo demasiado grande, convirtiéndolo así en algo inútil. Por este motivo, suele tomarse 0.95, que representa un valor de compromiso.

2.7.5. Determinación del Valor Estadístico de Prueba

Para la comprobación de la hipótesis planteada se utiliza el método de Chi-Cuadrado mediante la Prueba de Bondad de Ajuste, en el que analiza el número de éxitos en las pruebas realizadas, para este caso utilizaremos el número de pruebas con el filtro adaptativo, determinado mediante la audición, para lo cual determinaremos la relación con los siguientes parámetros.

H0 se acepta si:

$$X_E^2 \leq X_C^2$$

H1 se acepta si:

$$X_E^2 > X_C^2$$

Para aceptar o rechazar esta hipótesis se toma en cuenta los resultados de una comparativa obtenida entre el filtro Adaptativo en la tarjeta STM23F746G Discovery con el filtro adaptativo desarrollado en Simulink – Matlab.

En la tabla 2 se muestra los valores obtenidos después de haber realizado la toma de datos entre los dos posibles casos.

	RUIDO	ERROR	SALIDA	TOTAL
TARJETA STM32F746G	0,4485782	-0,0024874	-0,0027895	0,4433013
SIMULINK-MATLAB	0,4256874	-0,0024547	-0,0038542	0,4193785
TOTAL	0,8742656	-0,0049421	-0,0066437	0,8626798

Tabla 2 Valores Obtenidos en Pruebas
Fuente: Autores

Una vez obtenidos los datos se procede a obtener las frecuencia esperadas para esto se multiplica el total de cada columna, por el total de cada fila y se divide entre el total de cada fila, y la columna, como se puede observar en la tabla 3.

	RUIDO	ERROR	SALIDA	TOTAL
TARJETA STM32F746G	0,4433013	-0,0049421	-0,0066437	0,4317155
SIMULINK-MATLAB	0,8742656	-0,0049421	-0,0049421	0,8643814

TOTAL	1,3175669	-0,0098842	-0,0115858	1,2960969

Tabla 3 Frecuencias Esperadas
Fuente: Autores

Para obtener los grados de libertad se realiza una diferencia con el número de la fila menos uno por el número de columnas menos uno, obteniendo dos grados de libertad, como se establece en la tabla 4.

Finalmente la tabla 4 muestra los valores obtenidos tras realizar los procesos antes mencionados.

Numero de fila	2
Numero de columna	3
X^2	0,4058
Grados de Libertad	2
Nivel de Significancia	0,05
Probabilidad	0,95
Valor Critico	0,103

Tabla 4 Valores Finales
Fuente: Autores

Con el resultado obtenido x^2 donde es mayor que el valor crítico, este resultado indica que se rechaza la hipótesis nula y se acepta la hipótesis alternativa que es:

”Hipótesis Alternativa (H1)”:

Con el diseño e implementación de un filtro adaptativo mediante filtros WIENER se logrará cancelar el ruido en señales de audio.

CAPÍTULO III

3. RESULTADOS

Para la cancelación de ruido en las señales de audio se realizó la implementación del filtro adaptativo utilizando la teoría de filtrado de Wiener, lo cual resulto muy eficiente, debido a que gracias a esta técnica empleada se logra cancelar el ruido en un gran porcentaje pero cabe destacar que el ruido se encuentra todavía en el sistema para ello realizamos las pruebas con varias señales para determinar la eficiencia del filtro a continuación se explicará cada uno de los resultados obtenidos y de esta manera poder evaluar su funcionamiento y capacidad figura 22.

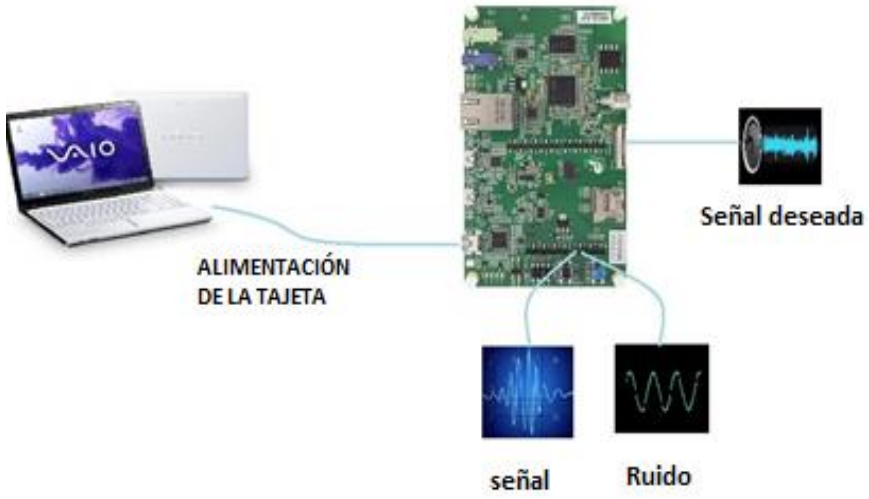


Figura. 22 Implementación general.
Fuente: Autores

3.1. Escenario 1: Señal Periódica – Señal Periódica (Ruido).

La siguiente prueba se realiza mediante la utilización de dos señales seno que serán tomadas desde el generador de funciones, a la salida del sistema, se observa la adaptación del filtro es decir la gráfica 24 muestra una perturbación de la señal hasta que esta se estabilice y tome la forma de la señal original

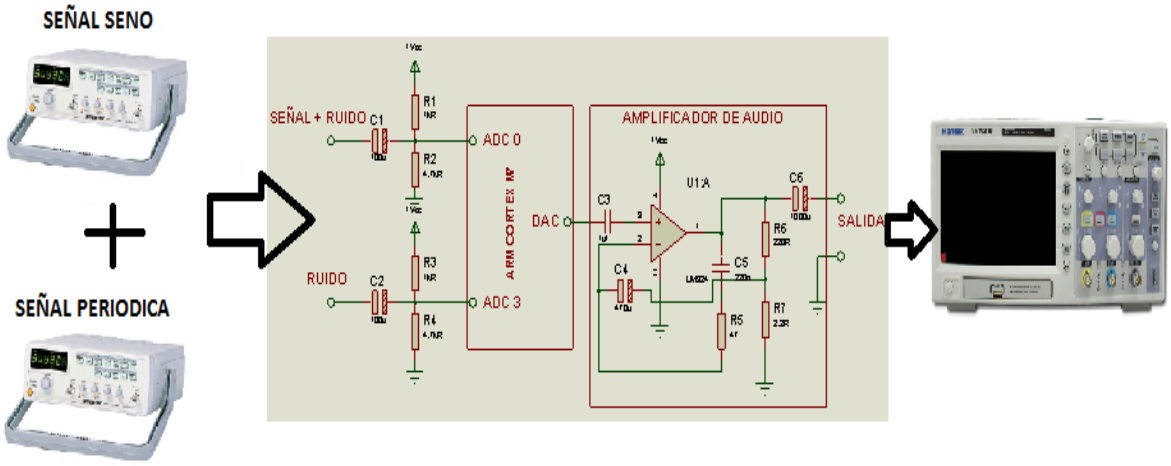


Figura. 23 Circuito electrónico escenario 1.
Fuente: Autores

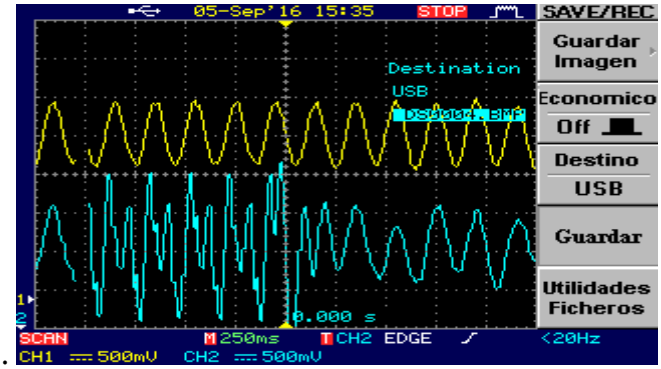


Figura. 24 Señal de audio - Señal periódica.
Fuente: Autores

En la gráfica 25 y 26 se puede observar la señal filtrada y se muestra también la señal de entrada como se puede observar las dos señales son casi idénticas por lo que el filtro adaptativo funciona correctamente.

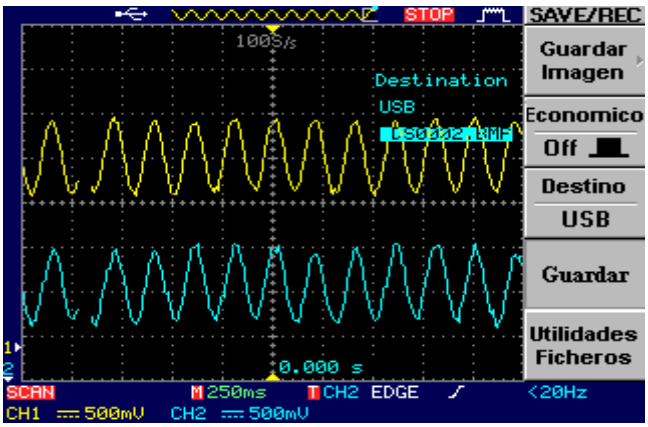


Figura. 25 Señal de audio - Señal de audio(ruido).
Fuente: Autores

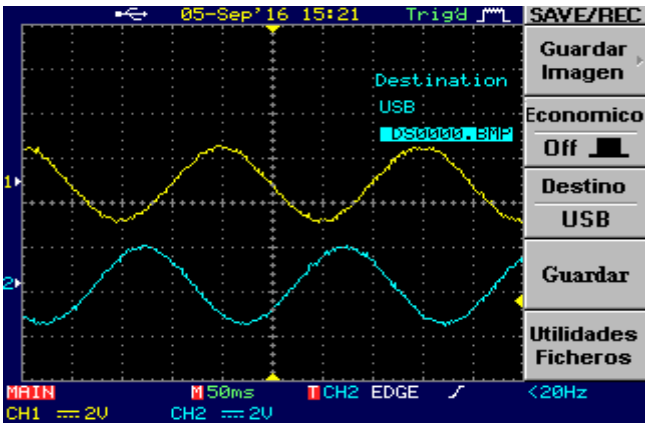


Figura. 26 Señal periódica- Señal de audio.
Fuente: Autores

3.2. Escenario 2: Cancelación del ruido de un sonido musical.

Para realizar la primera prueba de la implementación del filtro adaptativo utilizando la teoría de filtrado de Wiener lo que se necesitará es:

- 2 computadores portátiles.
- Osciloscopio.

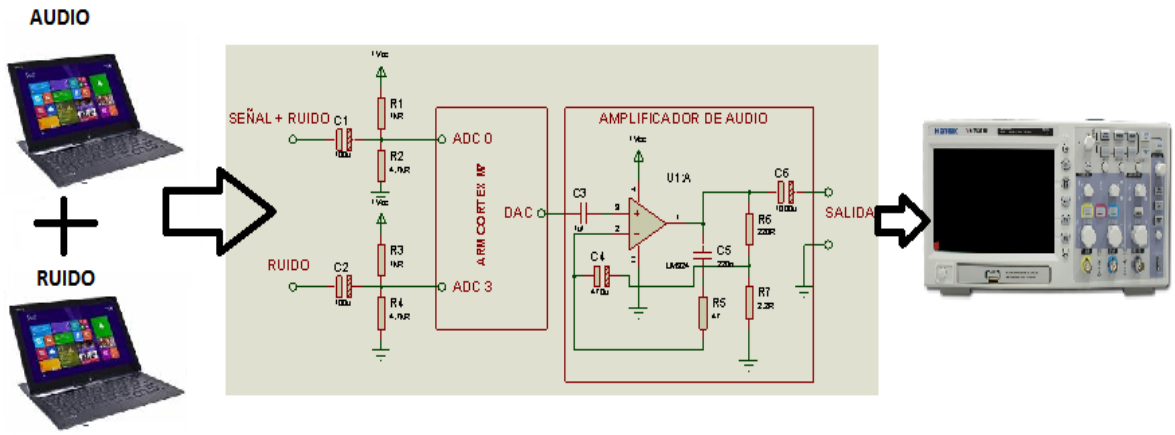


Figura. 27 Señal de audio filtrada.
Fuente: Autores

Para esta prueba se ingresa una señal de audio emitida por una pc y una señal periódica permitiendo ser esta una onda seno – coseno – cuadrada- triangular, que será igualmente proporcionada por una pc, para este caso se utilizó una onda periódica seno de 1Khz el cual es perceptible para el oído humano, al pasar por el proceso de filtrado se puede escuchar a su salida la señal de audio que es una pista musical, la cual se encuentra con un ruido casi imperceptible, este fenómeno se puede apreciar conectando a la salida del

sistema un sistema amplificador se utilizará parlantes de computadora al disminuir por completo el audio de la pista musical e incrementando el volumen de los parlantes se puede escuchar de manera muy leve el tono en este caso la señal de ruido.

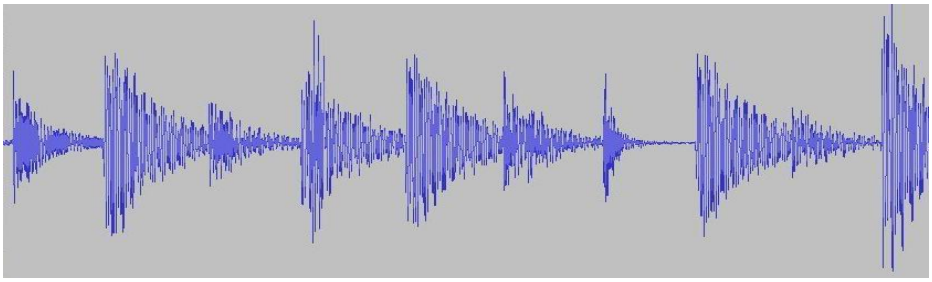


Figura. 28 Audio Filtrado
Fuente: Autores

CAPÍTULO IV

4. DISCUSIÓN

El presente trabajo de investigación tiene por objetivo el diseño e implementación de un algoritmo adaptativo mediante la utilización de la teoría de filtrado de Wiener para la cancelación de ruido en señales de audio.

Los sistemas de filtrado adaptativos son sistemas que están basados en algoritmos matemáticos, los cuales proporcionan un aprendizaje de valores numéricos pasados que permitirán una predicción de sucesos futuros, estos valores permiten calcular los pesos (w) que serán utilizados dentro del filtrado de señales.

La teoría de filtrado de Wiener basa su funcionalidad en el minimizar en valor calculado del error en este caso dicho valor será proporcionado por el algoritmo de adaptación LMS, dicho proceso se conoce como minimización del error cuadrático medio.

El ruido es una señal molesta y que se encuentra en casi todos los sistemas eléctricos pero con mayor presencia en señales de audio como en auditorios, salas de conferencia, durante llamadas telefónicas, el objetivo del estudio y diseño de algoritmos inteligentes que permitan variar automáticamente sus parámetros hacen posible la cancelación de ruido que se pueden presentar con diferentes fuentes ruidosas sin importan si a medida que varía el tiempo también lo hace en su frecuencia sin tener que realizar múltiples filtros según la frecuencia de las señal que se desea eliminar.

Mediante la programación de algoritmos adaptativos en los diferentes tipos de tarjetas de desarrollo que hoy en día existen en el mercado se ha logrado optimizar los procesos matemáticos gracias a las altas velocidades de procesamiento que estos dispositivos manejan, lo cual brinda múltiples usos en la implementación de estos procesos uno de los más utilizados es el que utilizaremos en el presente proyecto de investigación que es la cancelación de ruido en señales de audio.

CAPÍTULO V

5. Conclusiones y Recomendaciones

5.1. Conclusiones

- El periodo de adaptación transcurrido para lograr cancelar el ruido, se cuantifica en 5 segundos, dicho tiempo es el resultado del proceso computacional que realiza el microcontrolador.

- El filtro de Wiener se caracteriza por minimizar el error cuadrático medio que proporciona el algoritmo de adaptación LMS, lo cual implica un proceso matemático mucho más complejo y por ende una carga computacional muy elevada, por esta razón la implementación se realiza en la tarjeta de desarrollo STM32F746G la cual posee una capacidad de procesamiento digital de 32 bits, esta es una característica relevante en el proyecto ya que permite la realización de operaciones que manejen unidades de punto flotante.

- La cancelación de ruido no se produjo en un 100%, esto se puede apreciar disminuyendo al máximo el nivel de audio de la señal y elevando el nivel de audio de los parlantes, se logra escuchar en un nivel muy bajo el ruido a la salida del sistema.

5.2. Recomendaciones.

- Se recomienda revisar los voltajes tolerables a la entrada del ADC de la tarjeta de desarrollo STM32F746G (0V - 3.3V), para prevenir daños en al procesador interno.

- Previo al ingreso de las señales analógicas en la tarjeta de desarrollo STM32F746G se debe agregar un circuito sujetador de nivel el cual garantizara que se tomaran las lecturas de datos en el siclo positivo y así no tendremos pérdidas durante la recomposición de la señal.

- Además del software de programación Keil Uvision 5, el software STM32 CUBE MX permite configurar parámetros en el ADC, DAC, y TIMER, con mayor facilidad gracias a que maneja entornos gráficos.

CAPÍTULO VI

6. PROPUESTA

6.1. Título de la propuesta

DISEÑO E IMPLEMENTACIÓN DE UN FILTRO ADAPTATIVO MEDIANTE FILTROS WIENER PARA LA CANCELACÓN DE RUIDO EN SEÑALES DE AUDIO.

6.2. Introducción

Se pretende evaluar el nivel de ruido que se encuentra presente en las señales de audio, para ser cancelada y de esta manera tener presente solo la señal de audio original.

Mediante un filtrado, extraer la información o datos de interés en un tiempo t de una señal contaminada y emplear esos datos al mismo tiempo con el fin de actualizar la salida ya sin ruido. Para lograr esto se aplicará un filtro Wiener, el filtro Wiener es un sistema digital compuesto por un filtro lineal programable de entrada y de salida. Los coeficientes se reprograman de una muestra a la siguiente a través de un algoritmo de adaptación, estos filtros son los mejores filtros lineales de mínimos cuadrados, que pueden ser usados para predicción, estimación, interpolación, filtrado de señal y ruido, etc.

Los filtros adaptativos, son capaces de cambiar sus coeficientes, el filtro de Wiener se lo suele llamar de filtrado optimo o filtrado lineal optimo con restricciones, Para su utilización se requiere una solución iterativa de las ecuaciones normales y se suele hacer un análisis de su convergencia.

Los Algoritmos Adaptativos tienen un gran potencial en procesos de búsqueda y optimización que encuentran múltiples aplicaciones en el procesamiento digital de

señales, lo que hace que sea aplicable para la cancelación o eliminación de ruido de las señales de audio, independientemente de la magnitud que sea el ruido.

6.3. Objetivos

6.3.1. Objetivo General

Diseñar e implementar un filtro adaptativo mediante filtros WIENER para la cancelación de ruido en señales de audio.

6.3.2. Objetivos Específicos

- Caracterizar el funcionamiento de los filtros Wiener en entornos de ruido.
- Estudiar la base teórica de algoritmos Adaptativos.
- Comprender como la utilización de un filtro adaptativo mediante filtro Wiener podría cancelar el ruido en las señales de audio.
- Establecer el tipo de algoritmo matemático que mejor se ajuste a la implementación del filtro adaptativo mediante filtro Wiener para la cancelación de ruido en las señales de audio.
- Describir si se logró la cancelación del ruido de la señal de audio, mediante la implementación de un filtro adaptativo mediante filtro Wiener.
- Determinar el tipo de tarjeta programable que se utilizará en la implementación del presente proyecto de investigación así como también su lenguaje de programación.

6.4. Fundamentación Científico-Técnico

Los filtros adaptativos basado en Filtro Wiener, poseen la cualidad de alterar el contenido de una señal de entrada, logrando cancelar solo el ruido o datos no deseados, un proceso

adaptivo involucra el ajuste automático de los parámetros del filtro de acuerdo al error estimado.

La velocidad del proceso matemático dependerá del tipo de procesador utilizado en la tarjeta STM, se recomiendo utilizar el máximo valor debido a que las frecuencias de las señales de audio son altas.

6.5. Descripción de la propuesta

El presente proyecto consiste en la cancelación de ruido en señales de audio de un filtro adaptativo mediante filtros Wiener. Para lograr este objetivo se realizó diferentes procesos para llegar a este resultado:

Etapa 1: Caracterizar el diagrama del filtro Wiener.
Etapa 2: Simulación del Wiener en Matlab.
Etapa 3: Manejo y fundamentos de programación adecuado del programa Keil uVision5 para implementar en la Tarjeta STM.
Etapa 4: Pruebas de laboratorio para determinar la efectividad del filtro.

6.6. Diseño organizacional

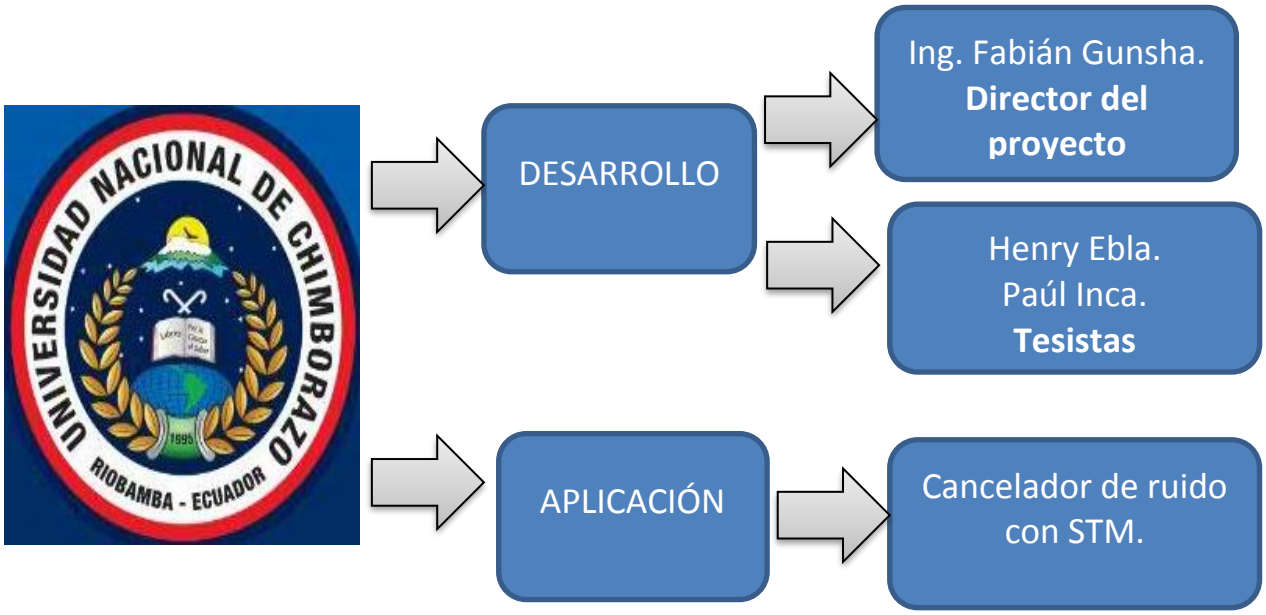


Figura. 29 Diseño Organizacional.
Fuente: Autores.

6.7. Monitoreo y Evaluación de la propuesta

La evaluación de la propuesta se realizará a través de pruebas, sometiendo a la señal de audio a diferentes tipos de ruido.

El impacto que produce la implementación de un filtro adaptativo con redes neuronales es de gran beneficio, al lograr la eliminación del ruido en la transmisión de información especialmente en la telefonía, cuando es perjudicada por un algún tipo de ruido.

BIBLIOGRAFÍA

[1] Cruz, D. P. (2001). *Implementación de filtros adaptativos LMS en tiempo real.* Quito.

[2] Delgado, J. P. (s.f.). *Equipos de sonido* . Obtenido de Departemento de Electrónica: http://213.96.251.252/sonido/temario/tema02.pdf

[3] Domingo, D. M. (2012). *Estudio y comparativa de diferentes algoritmos adaptativos para la identificaión de sistemas.*

[4] Fernàndez, M. (2012). *Obtencion de Micropotenciales Cardiacos Latido a Latido via Artificial.* Madrid.

[5] J. Alvarez, M. C. (2013). *Implementaciones en Matlab de los Algoritmos para Sistemas Lineales Inteligentes* . Guayaquil.

[6] L. Caiza, S. A. (2015). *FILTRO WIENER.* Rioamba.

[7] Microelectronics, S. (12 de 01 de 2015). *32F746GDISCOVERY.* Obtenido de 32F746GDISCOVERY: http://www.st.com/content/ccc/resource/technical/document/data_brief/b3/48/2b/e3/d2/12/45/c1/DM00179227.pdf/files/DM00179227.pdf/jcr:content/translations/en.DM00179227.pdf

[8] Pablo Mosè, E. L. (2015). *Practica 3 Filtro de Wiener.*

ENLACES WEB

[1]http://www.st.com/content/ccc/resource/technical/document/user_manual/f0/14/c1/b9/95/6d/40/4d/DM00190424.pdf/files/DM00190424.pdf/jcr:content/translations/en.DM00190424.pdf

[2]http://www.st.com/content/ccc/resource/sales_and_marketing/promotional_material/brochure/f0/93/da/5c/6b/31/4a/96/brstm32.pdf/files/brstm32.pdf/jcr:content/translations/en.brstm32.pdf

[3]http://www.st.com/content/ccc/resource/legal/legal_agreement/license_agreement/66/75/d0/96/4c/67/4e/9f/EvaluationProductLicenseAgreement.pdf/files/EvaluationProductLicenseAgreement.pdf/jcr:content/translations/en.EvaluationProductLicenseAgreement.pdf

ANEXOS

ANEXO 1

DATASHEET

STM32F746NG

DISCOVERY

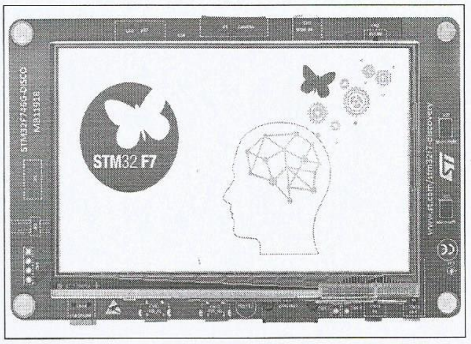
32F746GDISCOVERY

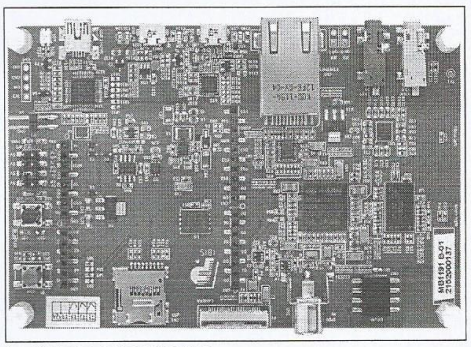
Discovery kit with STM32F746NG MCU

Data brief

Features

- STM32F746NGH6 microcontroller featuring 1 Mbytes of Flash memory and 340 Kbytes of RAM, in BGA216 package
- On-board ST-LINK/V2-1 supporting USB re-enumeration capability
- Mbed-enabled (mbed.org)
- USB functions: virtual COM port, mass storage, debug port
- 4.3-inch 480x272 color LCD-TFT with capacitive touch screen
- Camera connector
- SAI audio codec
- Audio line in and line out jack
- Stereo speaker outputs
- Two ST MEMS microphones
- SPDIF RCA input connector
- Two pushbuttons (user and reset)
- 128-Mbit Quad-SPI Flash memory
- 128-Mbit SDRAM (64 Mbits accessible)
- Connector for microSD card
- RF-EEPROM daughterboard connector
- USB OTG HS with Micro-AB connectors
- USB OTG FS with Micro-AB connectors
- Ethernet connector compliant with IEEE-802.3-2002
- Five power supply options:
  - ST LINK/V2-1
  - USB FS connector
  - USB HS connector
  - VIN from Arduino connector
  - External 5 V from connector
- Power supply output for external applications: 3.3 V or 5 V
- Arduino Uno V3 connectors
- Comprehensive free software including a variety of examples, part of STM32Cube package
- Supported by a wide choice of integrated development environments





1. Pictures not contractual

Description

The STM32F7 discovery kit allows users to develop and share applications with the STM32F7 Series microcontollers based on ARM® Cortex®-M7 core.

The discovery kit enables a wide diversity of applications taking benefit from audio, multi-sensor support, graphics, security, video and high-speed connectivity features.

The Arduino connectivity support provides unlimited expansion capabilities with a large choice of specialized add-on boards.

mbed Enabled

1 System requirements

- Windows OS (XP, 7, 8)
- USB type A to Mini-B cable

2 Development toolchains

- IAR EWARM (IAR Embedded Workbench®)
- Keil® MDK-ARM™
- GCC-based IDEs (free AC6: SW4STM32, Atollic® TrueSTUDIO®,...)
- ARM® mbed™ online

3 Demonstration software

The demonstration software is preloaded in the STM32F746NGH6 Flash memory. The latest versions of the demonstration source code and associated documentation can be downloaded from www.st.com/stm32f7-discovery.

4 Ordering information

To order the discovery kit with STM32F746NG MCU, use the order code: STM32F746G-DISCO.

5 Technology partners

MICRON:

- 128-Mbit SDRAM (64 Mbits accessible on the kit), part number MT48LC4M32B2
- 128-Mbit Quad-SPI NOR Flash memory device, part number N25Q128A

ROCKTECH:

- Color display, 4.3-inch LCD-TFT (resolution: 480x272), capacitive touch, part number RK043FN48H-CT672B

6 Revision history

Table 1. Document revision history

Date	Revision	Changes
04-Jun-2015	1	Initial release.
29-Jun-2015	2	Updated *Section : Features* adding 2 bullets: mbed-enabled, supported by a wide choice of integrated development environments. Added mbed-enabled logo. Updated *Section : Description.* Updated *Section 1: System requirements* adding OS at windows. Updated *Section 2: Development toolchains* adding ARM® mbed™ online.

ANEXO 2

DATASHEET

ARM CORTEX M7

ARM® Cortex®-M7 Processor

Revision r0p2

Technical Reference Manual

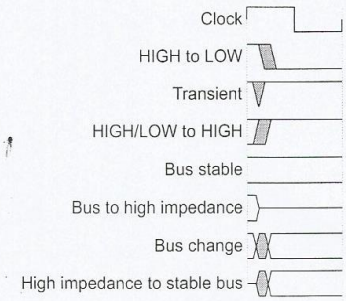
ARM®

ARM DDI 0489B (ID120914)

Timing diagrams

The figure named *Key to timing diagram conventions* explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are UNDEFINED, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.



Key to timing diagram conventions

Timing diagrams sometimes show single-bit signals as HIGH and LOW at the same time and they look similar to the bus change shown in *Key to timing diagram conventions*. If a timing diagram shows a single-bit signal in this way then its value does not affect the accompanying description.

Signals

The signal conventions are:

Signal level The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:

- HIGH for active-HIGH signals.
- LOW for active-LOW signals.

Lower-case n At the start or end of a signal name denotes an active-LOW signal.

Additional reading

This section lists publications by ARM and by third parties.

See Infocenter http://infocenter.arm.com, for access to ARM documentation.

See on ARM http://www.onarm.com, for embedded software development resources including the *Cortex Microcontroller Software Interface Standard* (CMSIS).

ARM publications

This book contains information that is specific to this product. See the following documents for other relevant information:

- *ARM® v7-M Architecture Reference Manual* (ARM DDI 0403).
- *ARM® CoreLink™ Level 2 Cache Controller L2C-310 Technical Reference Manual* (ARM DDI 0246).
- *ARM® CoreSight™ ETM-M7 Technical Reference Manual* (ARM DDI 0494).
- *ARM® AMBA® AXI and ACE Protocol Specification* (ARM IHI 0022).
- *ARM® AMBA® 3 AHB-Lite Protocol (v1.0)* (ARM IHI 0033).
- *ARM® AMBA® 3 ATB Protocol Specification* (ARM IHI 0032).
- *ARM® AMBA® 3 APB Protocol Specification* (ARM IHI 0024).
- *ARM® CoreSight™ SoC-400 Technical Reference Manual* (ARM DDI 0480).
- *ARM® CoreSight™ Architecture Specification (v2.0)* (ARM IHI 0029).
- *ARM® Debug Interface v5 Architecture Specification* (ARM IHI 0031).
- *ARM® Embedded Trace Macrocell Architecture Specification ETMv4* (ARM IHI 0064).

The following confidential books are only available to licensees:

- *ARM® Cortex®-M7 Processor Integration and Implementation Manual* (ARM DII 0239).

Other publications

This section lists relevant documents published by third parties:

- *Test Access Port and Boundary-Scan Architecture*, IEEE Standard 1149.1-2001 (JTAG).
- *IEEE Standard for Binary Floating-Point Arithmetic*, IEEE Standard 754-2008.

Feedback

ARM welcomes feedback on this product and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to `errata@arm.com`. Give:

- The title.
- The number, ARM DDI 0489B.
- The page numbers to which your comments apply.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Note

ARM tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Chapter 1
Introduction

This chapter introduces the processor. It contains the following sections:

1.1 About the Cortex-M7 processor

The Cortex-M7 processor is a highly efficient high-performance, embedded processor that features low interrupt latency, low-cost debug, and has backwards compatibility with existing Cortex-M profile processors. The processor has an in-order super-scalar pipeline that means many instructions can be dual-issued, including load/load and load/store instruction pairs because of multiple memory interfaces.

Memory interfaces that the processor supports include:

- *Tightly-Coupled Memory* (TCM) interface.
- Harvard instruction and data caches and *AXI master* (AXIM) interface.
- Dedicated low-latency *AHB-Lite peripheral* (AHBP) interface.

The processor has an optional *Memory Protection Unit* (MPU) that you can configure to protect regions of memory. *Error Correcting Code* (ECC) functionality for error detection and correction, is included in the data and instruction caches when implemented. The TCM interfaces support the implementation of external ECC to provide improved reliability and to address safety-related applications.

The Cortex-M7 processor includes optional floating-point arithmetic functionality, with support for single and double-precision arithmetic. See Chapter 8 *Floating Point Unit*.

The processor is intended for high-performance, deeply embedded applications that require fast interrupt response features.

Figure 1-1 shows the processor in a typical system.

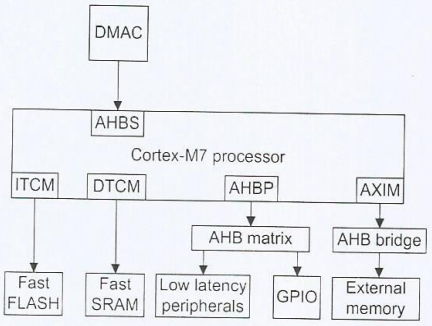


Figure 1-1 Example Cortex-M7 system

1.1.1 Features

The main features of the Cortex-M7 processor include:

- An in-order issue, super-scalar pipeline with dynamic branch prediction.
- DSP extensions.
- The ARMv7-M Thumb instruction set, defined in the *ARM® v7-M Architecture Reference Manual*.
- Banked *Stack Pointer* (SP).
- Hardware integer divide instructions, `SDIV` and `UDIV`.
- Handler and Thread modes.

- Thumb and Debug states.
- Automatic processor state saving and restoration for low-latency *Interrupt Service Routine* (ISR) entry and exit.
- Support for ARMv7-M big-endian byte-invariant or little-endian accesses.
- Support for ARMv7-M unaligned accesses.
- Low-latency interrupt processing achieved by:
  - A *Nested Vectored Interrupt Controller* (NVIC) closely integrated with the processor.
  - Supporting exception-continuable instructions, such as `LDM`, `LDMDB`, `STM`, `STMDB`, `PUSH`, `POP` and `VLDM`, `VSTM`, `VPUSH`, `VPOP` if the processor has the *Floating Point Unit* (FPU).
- A low-cost debug solution with the optional ability to:
  - Implement breakpoints.
  - Implement watchpoints, tracing, and system profiling.
  - Support `printf()` style debugging through an *Instrumentation Trace Macrocell* (ITM).
  - Bridge to an off-chip *Trace Port Analyzer* (TPA).
- Support for an optional ETM. See the *ARM® CoreSight™ ETM-M7 Technical Reference Manual* for more information.
- A memory system, which includes an optional MPU and Harvard data and instruction cache with ECC.
- An optional *Floating Point Unit* (FPU).

1.1.2 Interfaces

The Cortex-M7 processor has a number of external interfaces.

Figure 1-2 shows the external interfaces of the Cortex-M7 processor.

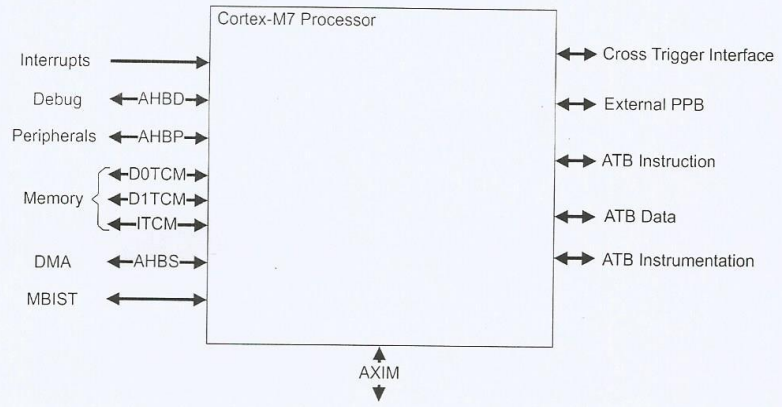


Figure 1-2 Cortex-M7 processor interfaces

1.1.3 Configuration options

The Cortex-M7 processor has configurable options that you can configure during the implementation and integration stages to match your functional requirements.

Table 1-1 shows the configurable options at build time of the processor.

Table 1-1 Implementation options

Feature	Options	Done at
Floating-point	No floating-point.	Implementation
	Single-precision floating-point only.	
	Single-precision and double-precision floating-point.	
Instruction TCM	No instruction TCM.	Integration
	4KB-16MB (powers of 2).	
Data TCM	No data TCM.	Integration
	4KB-16MB (powers of 2). The Data TCM is split equally into two TCMs, D0TCM, and D1TCM.	
Instruction cache	No instruction cache controller.	Implementation
	Instruction cache controller is included.	
Data cache	Area optimized AXIM interface, no data cache.	Implementation
	Performance optimized AXIM interface, data cache included.	
Instruction cache size	4KB, 8KB, 16KB, 32KB, 64KB.	Integration
Data cache size	4KB, 8KB, 16KB, 32KB, 64KB.	Integration
AHB peripheral size	64KB, 128KB, 256KB, 512MB.	Integration
ECC support on caches	No ECC on instruction cache or data cache.	Implementation
	ECC on all implemented caches.	
Protected memory regions	0 region, 8 regions, 16 regions.	Implementation
Interrupts	1-240 interrupts.	Implementation
Number bits of interrupt priority	Between three and eight bits of interrupt priority, between 8 and 256 levels of priority.	Implementation
Debug watchpoints and breakpoints	Reduced set. Two data watchpoints comparators and four breakpoint comparators.	Implementation
	Full set. Four data watchpoints comparators and eight breakpoint comparators.	
ITM and *Data Watchpoint and Trace* (DWT) trace functionality	No ITM or DWT trace.	Implementation
	Complete ITM and DWT trace.	
ETM	No ETM support.	Implementation
	ETM instruction trace only.	
	ETM instruction and data trace.	

Table 1-1 Implementation options (continued)

Feature	Options	Done at
Dual-redundant processor	No dual-redundant processor.	Implementation
	Dual-redundant processor included.	
Reset All Registers	Only required registers that must be initialized are reset in the RTL.	Implementation
	All registers are reset in the RTL excluding those in the ETM, if included.	
	All registers are reset in the RTL including those in the ETM, if included.	
Cross Trigger Interface (CTI)	No Cross Trigger Interface.	Implementation
	Cross Trigger Interface included.	
Wake-up Interrupt Controller (WIC)	No Wake-up Interrupt controller.	Implementation
	Wake-up Interrupt controller included.	

1.2 Component blocks

The Cortex-M7 processor has fixed and optional component blocks. Optional components are *Wake-up Interrupt Controller* (WIC), ITM, FPU, MPU, instruction cache and data cache controllers, *Cross Trigger Interface* (CTI), and ETM. All other components are fixed.

Table 1-1 on page 1-4 shows the configurable options at implementation time of the processor.

Figure 1-3 shows the components of the Cortex-M7 processor.

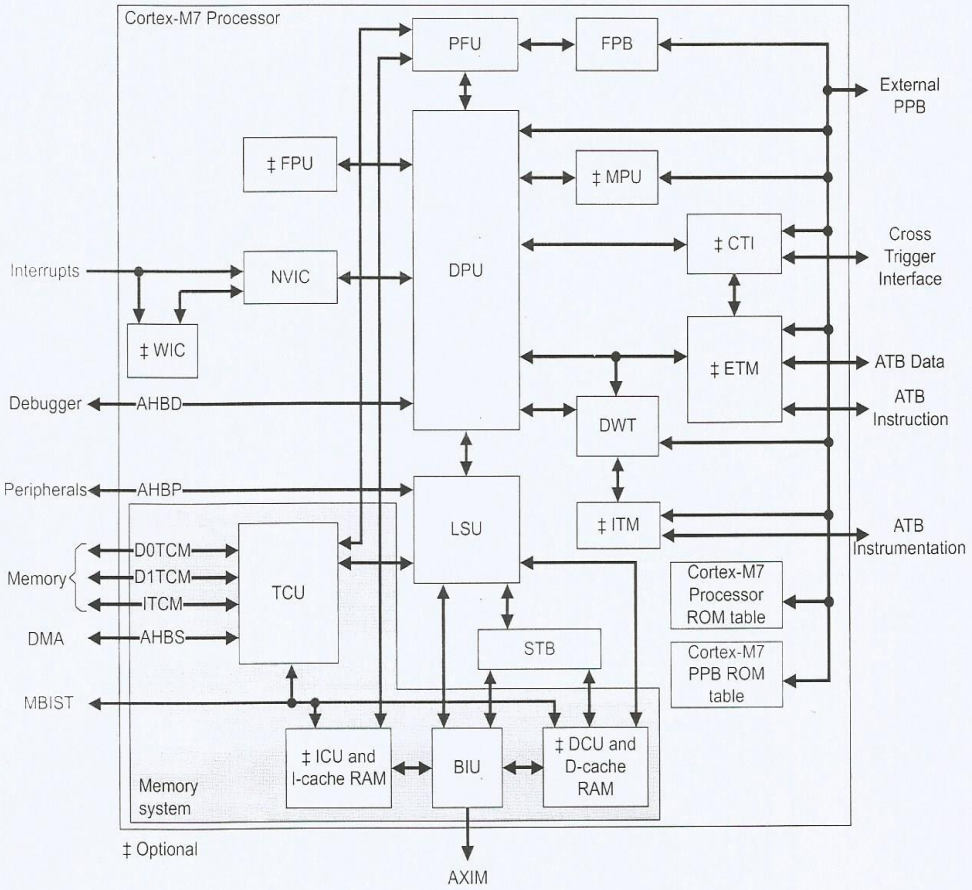


Figure 1-3 Cortex-M7 functional diagram

The Cortex-M7 processor contains the following component blocks:

- *Data Processing Unit* on page 1-7.
- *Prefetch Unit* on page 1-7.
- *Load Store Unit* on page 1-7.
- *Floating Point Unit* on page 1-8.
- *Nested Vectored Interrupt Controller* on page 1-8.
- *Wake-up Interrupt Controller* on page 1-8.
- *Memory System* on page 1-8.
- *Memory Protection Unit* on page 1-9.
- *Cortex-M7 Processor and PPB ROM tables* on page 1-9.

ANEXO 3

DESARROLLO DEL PROGRAMA EN LENGUAJE C

PROGRAMA PRINCIPAL

```
/**

******************************************************************************
  * File Name          : main.c
  * Description        : Main program body

******************************************************************************
  *
  * COPYRIGHT(c) 2016 STMicroelectronics
  *
  * Redistribution and use in source and binary forms, with or without modification,
  * are permitted provided that the following conditions are met:
  *   1. Redistributions of source code must retain the above copyright notice,
  *      this list of conditions and the following disclaimer.
  *   2. Redistributions in binary form must reproduce the above copyright notice,
  *      this list of conditions and the following disclaimer in the documentation
  *      and/or other materials provided with the distribution.
  *   3. Neither the name of STMicroelectronics nor the names of its contributors
  *      may be used to endorse or promote products derived from this software
  *      without specific prior written permission.
  *
  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
  * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  *

******************************************************************************
  */
/* Includes ------------------------------------------------------------------*/
```

```
#include "stm32f7xx_hal.h"

#include "sabado13v1.h"
#include "rtwtypes.h"


/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc1;
ADC_HandleTypeDef hadc3;

DAC_HandleTypeDef hdac;

TIM_HandleTypeDef htim6;

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
void Error_Handler(void);
static void MX_GPIO_Init(void);
static void MX_ADC1_Init(void);
static void MX_ADC3_Init(void);
static void MX_DAC_Init(void);
static void MX_TIM6_Init(void);

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/

/* USER CODE END PFP */

/* USER CODE BEGIN 0 */

volatile int IsrOverrun = 0;
static boolean_T OverrunFlag = 0;
void rt_OneStep(void)
{

//HAL_DAC_SetValue (&hdac,DAC_CHANNEL_1,DAC_ALIGN_12B_R,  rtY.Out1);
        /* Check for overrun. Protect OverrunFlag against preemption */
  if (OverrunFlag++) {
    IsrOverrun = 1;
    OverrunFlag--;
    return;
  }

  __enable_irq();
  sabado13v1_step();
```

```
  /* Get model outputs here */
  __disable_irq();
  OverrunFlag--;
}


/* USER CODE END 0 */

int main(void)
{

  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_ADC1_Init();
  MX_ADC3_Init();
  MX_DAC_Init();
  MX_TIM6_Init();

  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */
HAL_ADC_Start(&hadc1);
HAL_ADC_Start(&hadc3);
HAL_TIM_Base_Start_IT (&htim6);
/* USER CODE END 2 */

          HAL_DAC_Start (&hdac, DAC_CHANNEL_1);
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */

  /* USER CODE END 2 */
  volatile boolean_T runModel = 1;
  float modelBaseRate = 1.2;
  float systemClock = 168;

#ifndef USE_RTX

  __disable_irq();

#endif

  ;
//  stm32f4xx_init_board();
```

```
  SystemCoreClockUpdate();
 // bootloaderInit();
  rtmSetErrorStatus(rtM, 0);
  sabado13v1_initialize();
 // ARMCM_SysTick_Config(modelBaseRate);
  runModel =
    rtmGetErrorStatus(rtM) == (NULL);
  __enable_irq();
  __enable_irq();

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (runModel) {
    runModel =
      rtmGetErrorStatus(rtM) == (NULL);

            rt_OneStep();

            }

#ifndef USE_RTX

  (void)systemClock;

#endif

  ;

  /* Disable rt_OneStep() here */

  /* Terminate model */
  sabado13v1_terminate();
  __disable_irq();
  return 0;
} /* USER CODE END 3 */



/** System Clock Configuration
*/
void SystemClock_Config(void)
{

  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;

  __HAL_RCC_PWR_CLK_ENABLE();


__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_S
CALE2);

  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = 16;
```

```
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 10;
  RCC_OscInitStruct.PLL.PLLN = 210;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 2;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  RCC_ClkInitStruct.ClockType                                                    =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                      |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
  if    (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,    FLASH_LATENCY_5)    !=
HAL_OK)
  {
    Error_Handler();
  }

  HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

  HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

  /* SysTick_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}

/* ADC1 init function */
static void MX_ADC1_Init(void)
{

  ADC_ChannelConfTypeDef sConfig;

    /**Configure the global features of the ADC (Clock, Resolution, Data Alignment and
number of conversion)
    */
  hadc1.Instance = ADC1;
  hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
  hadc1.Init.Resolution = ADC_RESOLUTION_12B;
  hadc1.Init.ScanConvMode = DISABLE;
  hadc1.Init.ContinuousConvMode = DISABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_RISING;
  hadc1.Init.ExternalTrigConv = ADC_EXTERNALTRIGCONV_T6_TRGO;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
  hadc1.Init.DMAContinuousRequests = DISABLE;
  hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
```

```
    Error_Handler();
  }

    /**Configure for the selected ADC regular channel its corresponding rank in the
sequencer and its sample time.
    */
  sConfig.Channel = ADC_CHANNEL_0;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }

}

/* ADC3 init function */
static void MX_ADC3_Init(void)
{

  ADC_ChannelConfTypeDef sConfig;

    /**Configure the global features of the ADC (Clock, Resolution, Data Alignment and
number of conversion)
    */
  hadc3.Instance = ADC3;
  hadc3.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
  hadc3.Init.Resolution = ADC_RESOLUTION_12B;
  hadc3.Init.ScanConvMode = DISABLE;
  hadc3.Init.ContinuousConvMode = DISABLE;
  hadc3.Init.DiscontinuousConvMode = DISABLE;
  hadc3.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_RISING;
  hadc3.Init.ExternalTrigConv = ADC_EXTERNALTRIGCONV_T6_TRGO;
  hadc3.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc3.Init.NbrOfConversion = 1;
  hadc3.Init.DMAContinuousRequests = DISABLE;
  hadc3.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  if (HAL_ADC_Init(&hadc3) != HAL_OK)
  {
    Error_Handler();
  }

    /**Configure for the selected ADC regular channel its corresponding rank in the
sequencer and its sample time.
    */
  sConfig.Channel = ADC_CHANNEL_8;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc3, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }

}
```

```
/* DAC init function */
static void MX_DAC_Init(void)
{

  DAC_ChannelConfTypeDef sConfig;

    /**DAC Initialization
    */
  hdac.Instance = DAC;
  if (HAL_DAC_Init(&hdac) != HAL_OK)
  {
    Error_Handler();
  }

    /**DAC channel OUT1 config
    */
  sConfig.DAC_Trigger = DAC_TRIGGER_NONE;
  sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_ENABLE;
  if (HAL_DAC_ConfigChannel(&hdac, &sConfig, DAC_CHANNEL_1) != HAL_OK)
  {
    Error_Handler();
  }

}

/* TIM6 init function */
static void MX_TIM6_Init(void)
{

  TIM_MasterConfigTypeDef sMasterConfig;

  htim6.Instance = TIM6;
  htim6.Init.Prescaler = 0;
  htim6.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim6.Init.Period = 600;
  if (HAL_TIM_Base_Init(&htim6) != HAL_OK)
  {
    Error_Handler();
  }

  sMasterConfig.MasterOutputTrigger = TIM_TRGO_UPDATE;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim6, &sMasterConfig) !=
HAL_OK)
  {
    Error_Handler();
  }

}

/** Configure pins as
        * Analog
        * Input
        * Output
        * EVENT_OUT
```

```
      * EXTI
*/
static void MX_GPIO_Init(void)
{

  GPIO_InitTypeDef GPIO_InitStruct;

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOF,
GPIO_PIN_7|GPIO_PIN_6|GPIO_PIN_9|GPIO_PIN_8, GPIO_PIN_RESET);

  /*Configure GPIO pins : PF7 PF6 PF9 PF8 */
  GPIO_InitStruct.Pin = GPIO_PIN_7|GPIO_PIN_6|GPIO_PIN_9|GPIO_PIN_8;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOF, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @param  None
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler */
  /* User can add his own implementation to report the HAL error return state */
  while(1)
  {
  }
  /* USER CODE END Error_Handler */
}

#ifdef USE_FULL_ASSERT

/**
   * @brief Reports the name of the source file and the source line number
   * where the assert_param error has occurred.
   * @param file: pointer to the source file name
   * @param line: assert_param error line source number
   * @retval None
   */
void assert_failed(uint8_t* file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
```

```
  /* User can add his own implementation to report the file name and line number,
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */

}

#endif

/**
  * @}
  */

/**
  * @}
*/

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef *htim)
{



        HAL_GPIO_TogglePin  (GPIOF, GPIO_PIN_6);

        uint32_t a;
        uint32_t b;
        uint32_t c;

        rtU.In1=HAL_ADC_GetValue(&hadc1);
        rtU.In2=HAL_ADC_GetValue(&hadc3);
HAL_DAC_SetValue (&hdac,DAC_CHANNEL_1,DAC_ALIGN_12B_R,  rtY.Out1);
        ////      rtY.Out1 = rtU.In1 + rtU.In2;

// rtY.Out1 = rtU.In1 + rtU.In2;


//      suma_initialize();


        /* USER CODE BEGIN 3 */



}
```

PROGRAMACIÓN DEL CÁLCULO MATEMÁTICO

```
/*
 * File: sabado13v1.c
 *
 * Code generated for Simulink model 'sabado13v1'.
 *
 * Model version                  : 1.17
 * Simulink Coder version         : 8.10 (R2016a) 10-Feb-2016
 * C/C++ source code generated on : Sat Aug 13 16:34:23 2016
 *
 * Target selection: ert.tlc
 * Embedded hardware selection: ARM Compatible->ARM Cortex
 * Code generation objectives: Unspecified
 * Validation result: Not run
 */

#include "sabado13v1.h"
#include "sabado13v1_private.h"

/* Block signals (auto storage) */
B rtB;

/* Block states (auto storage) */
DW rtDW;

/* External inputs (root inport signals with auto storage) */
ExtU rtU;

/* External outputs (root outports fed by signals with auto storage) */
ExtY rtY;

/* Real-time model */
RT_MODEL rtM_;
RT_MODEL *const rtM = &rtM_;
void MWSPCGlmsnw_D(const real_T x[], const real_T d[], real_T mu, uint32_T
                   *startIdx, real_T xBuf[], real_T wBuf[], int32_T wLen, real_T
                   leakFac, int32_T xLen, real_T y[], real_T eY[], real_T wY[])
{
  int32_T i;
  int32_T j;
  real_T bufEnergy;
  int32_T j1;

  /* S-Function (sdsplms): '<Root>/LMS Filter' */
```

```
for (i = 0; i < xLen; i++) {
  y[i] = 0.0;
}

for (i = 0; i < xLen; i++) {
  bufEnergy = 0.0;

  /* Copy the current sample at the END of the circular buffer and update BuffStartIdx
   */
  xBuf[*startIdx] = x[i];
  (*startIdx)++;
  if (*startIdx == (uint32_T)wLen) {
    *startIdx = 0U;
  }

  /* Multiply wgtBuff_vector (not yet updated) and inBuff_vector
   */
  /* Get the energy of the signal in updated buffer
   */
  j1 = 0;
  for (j = (int32_T)*startIdx; j < wLen; j++) {
    y[i] += wBuf[j1] * xBuf[j];
    bufEnergy += xBuf[j] * xBuf[j];
    j1++;
  }

  for (j = 0; j < (int32_T)*startIdx; j++) {
    y[i] += wBuf[j1] * xBuf[j];
    bufEnergy += xBuf[j] * xBuf[j];
    j1++;
  }

  /* Ger error for the current sample
   */
  eY[i] = d[i] - y[i];

  /* Update weight-vector for next input sample
   */
  j1 = 0;
  for (j = (int32_T)*startIdx; j < wLen; j++) {
    wBuf[j1] = xBuf[j] / (bufEnergy + 2.2204460492503131E-16) * eY[i] * mu +
      leakFac * wBuf[j1];
    j1++;
  }

  for (j = 0; j < (int32_T)*startIdx; j++) {
    wBuf[j1] = xBuf[j] / (bufEnergy + 2.2204460492503131E-16) * eY[i] * mu +
```

```
      leakFac * wBuf[j1];
     j1++;
    }
  }

  j1 = wLen;
  for (j = 0; j < wLen; j++) {
    wY[j] = wBuf[j1 - 1];
    j1--;
  }

  /* End of S-Function (sdsplms): '<Root>/LMS Filter' */
}

/* Model step function */
void sabado13v1_step(void)
{
  real_T rtb_LMSFilter_o1;
  real_T rtb_LMSFilter_o2;

  /* S-Function (sdsplms): '<Root>/LMS Filter' incorporates:
   *  Inport: '<Root>/In2'
   */
  MWSPCGlmsnw_D(&rtU.In2, &rtU.In2, rtP.LMSFilter_mu,
                &rtDW.LMSFilter_BUFF_IDX_DWORK, &rtDW.LMSFilter_IN_BUFFER_DWORK
                [0U], &rtDW.LMSFilter_WGT_IC_DWORK[0U], 32,
                rtP.LMSFilter_leakage, 1, &rtb_LMSFilter_o1, &rtb_LMSFilter_o2,
                &rtB.LMSFilter_o3[0U]);

  /* Outport: '<Root>/Out1' incorporates:
   *  Inport: '<Root>/In1'
   *  Inport: '<Root>/In2'
   *  Sum: '<Root>/Sum'
   *  Sum: '<Root>/Sum1'
   */
  rtY.Out1 = (rtU.In1 + rtU.In2) - rtb_LMSFilter_o1;


}

/* Model initialize function */
void sabado13v1_initialize(void)
{
  /* (no initialization code required) */
}

/* Model terminate function */
```

```
void sabado13v1_terminate(void)
{
  /* (no terminate code required) */
}

/*
 * File trailer for generated code.
 *
 * [EOF]
 */
```

PROGRAMACIÓN DE LA MATRIZ DEL FILTRO

```
/*
 * File: sabado13v1_data.c
 *
 * Code generated for Simulink model 'sabado13v1'.
 *
 * Model version                  : 1.17
 * Simulink Coder version         : 8.10 (R2016a) 10-Feb-2016
 * C/C++ source code generated on : Sat Aug 13 16:34:23 2016
 *
 * Target selection: ert.tlc
 * Embedded hardware selection: ARM Compatible->ARM Cortex
 * Code generation objectives: Unspecified
 * Validation result: Not run
 */

#include "sabado13v1.h"
#include "sabado13v1_private.h"

/* Block parameters (auto storage) */
P rtP = {
  1.0,                          /* Mask Parameter: LMSFilter_leakage
                                 * Referenced by: '<Root>/LMS Filter'
                                 */
  0.1                           /* Mask Parameter: LMSFilter_mu
                                 * Referenced by: '<Root>/LMS Filter'
                                 */
};

/*
 * File trailer for generated code.
 *
 * [EOF]
 */
```

ANEXO 4

CREACIÓN DE UN NUEVO ARCHIVO EN MATLAB/SIMULINK

Matlab
Simulink

Considerar que la versión del software utilizado es Matlab 2016 en diferentes versiones puede variar las ubicaciones de los comandos.

Para iniciar nuestro entorno de trabajo debemos dar clic desde la ventana de Matlab para poder ingresar a Simulink como se indica en la siguiente Ilustración 1 E ilustración 2.

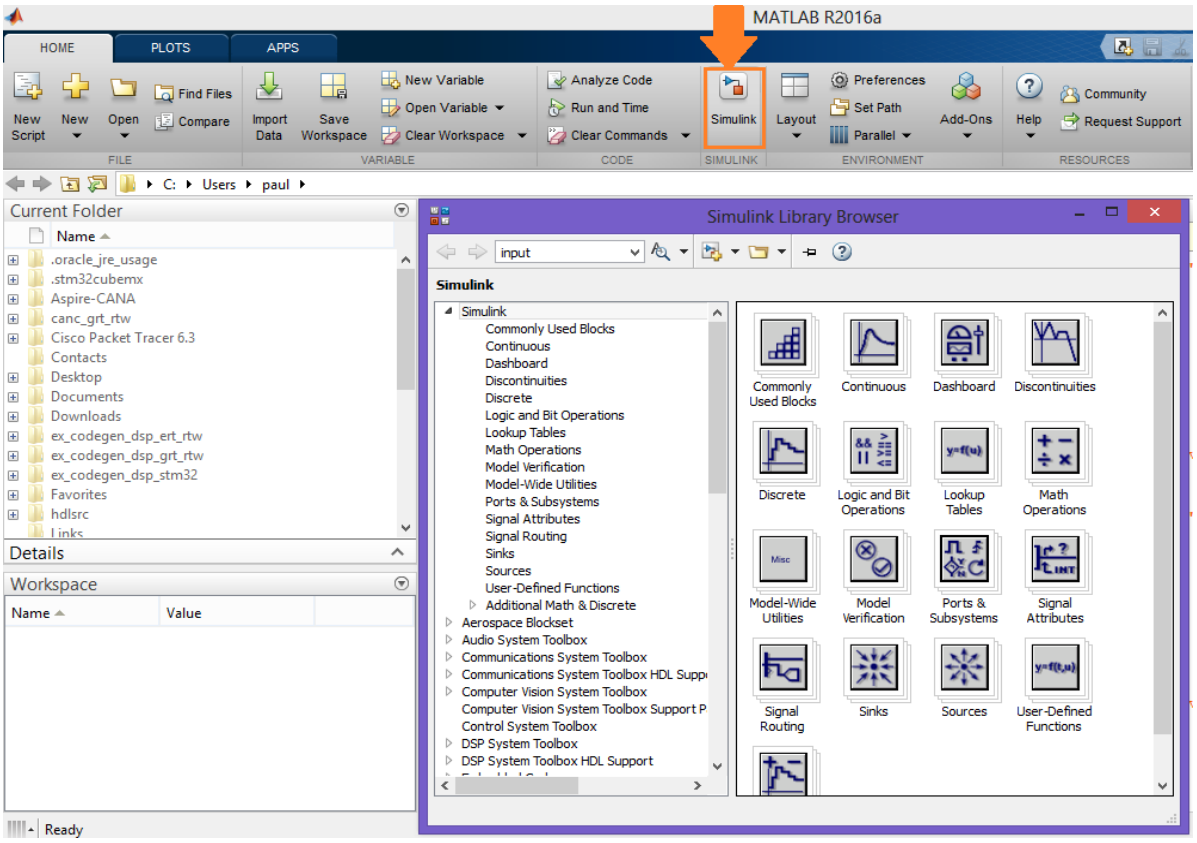


Ilustración 1 Creación del proyecto en Simulink.

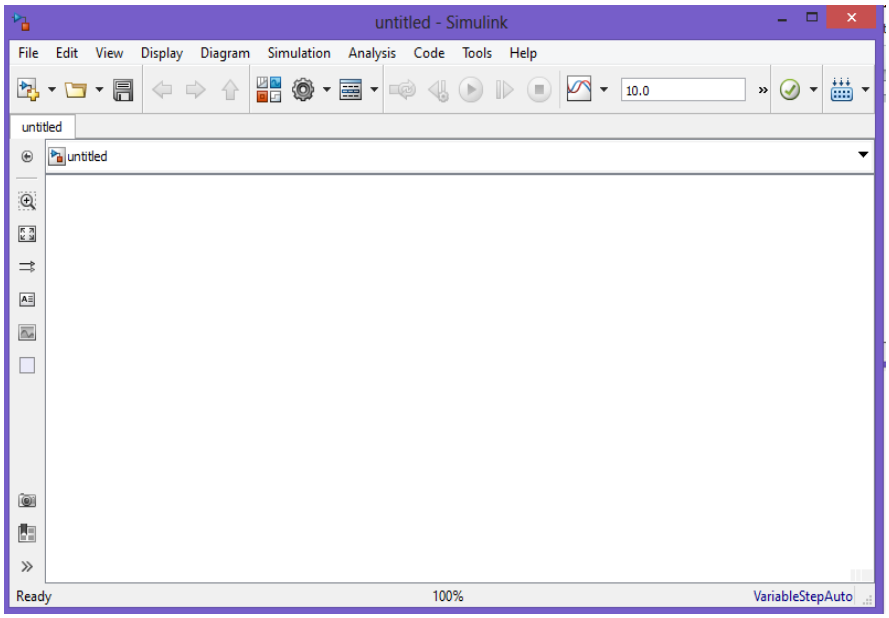


Ilustración 2 Entorno Simulink.

Para ingresar los bloques damos clic en el icono indicado en la Ilustración 3 e Ilustración 4 para desplazar la lista de componentes que posee simulink.

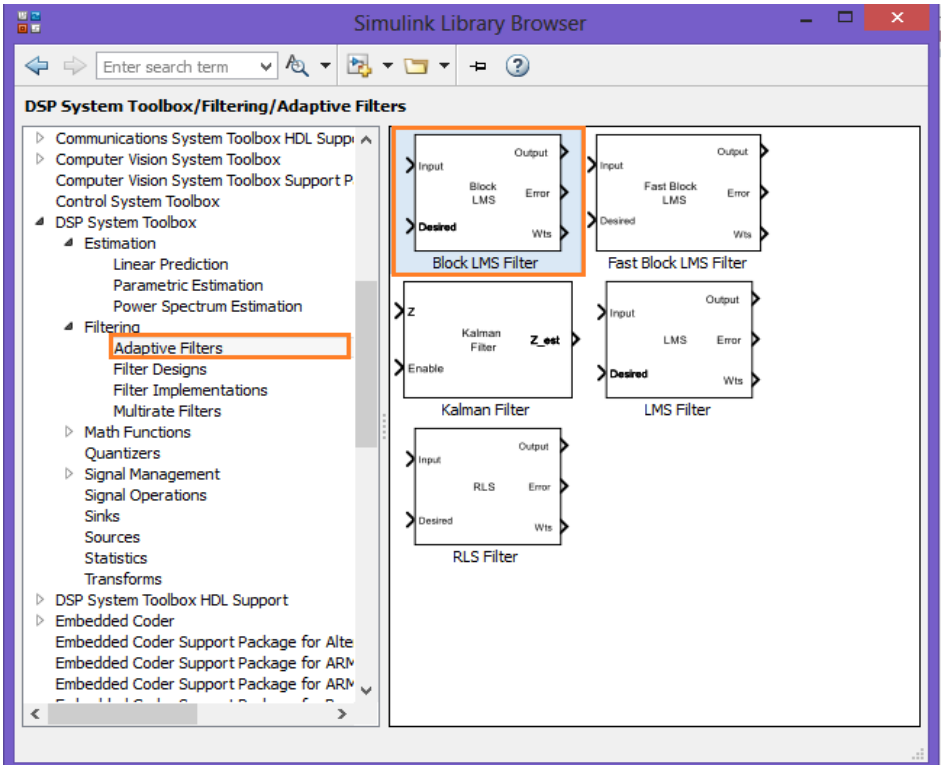


Ilustración 3 Bloque LMS.

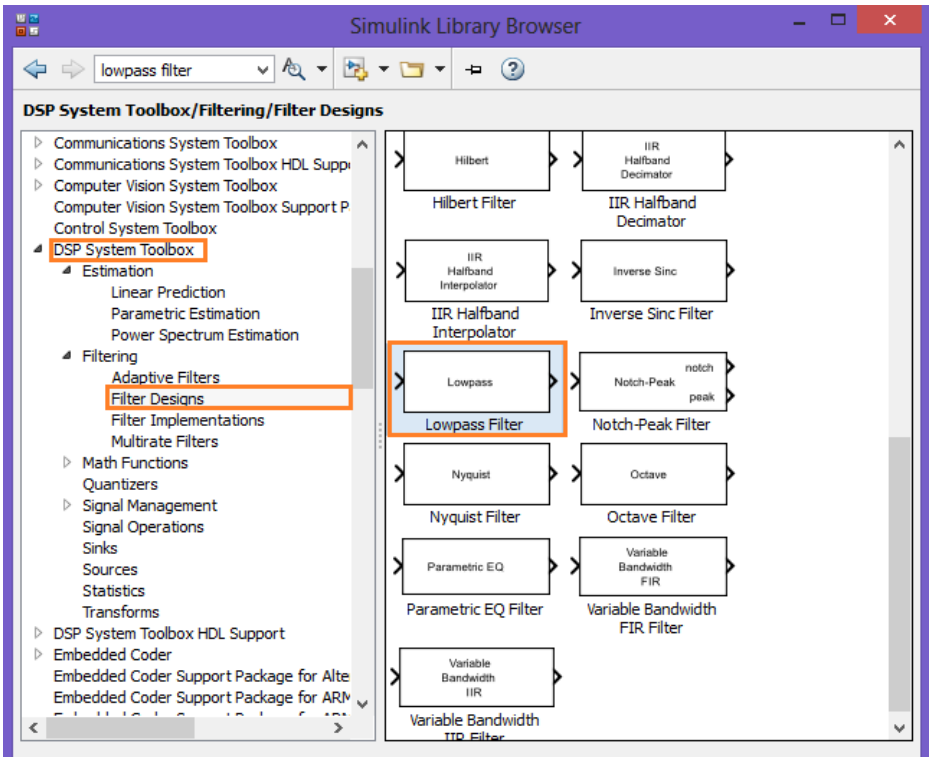


Ilustración 4 Filtro Wiener FIR.

El siguiente paso es ingresar el diagrama de bloques que representa al cancelador de ruido como se indica en la ilustración 5.

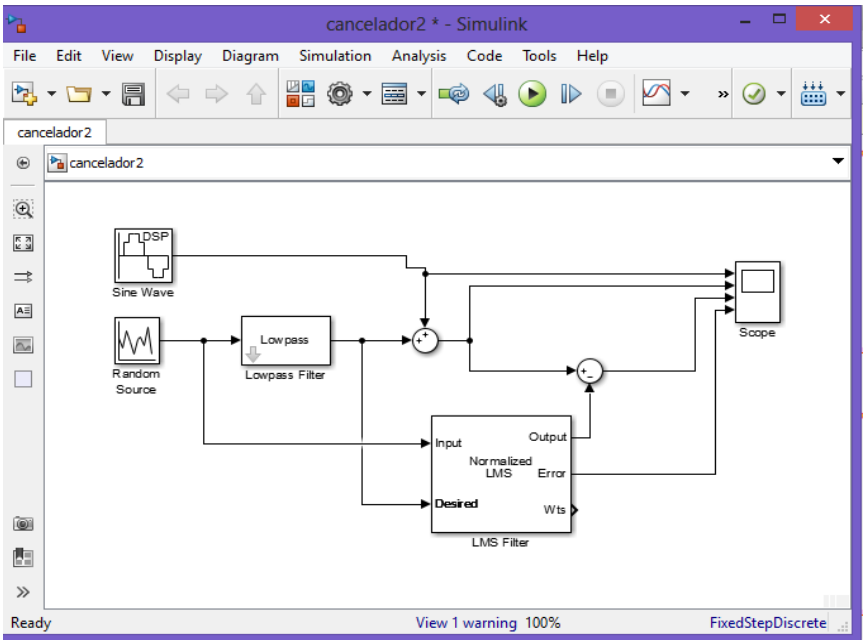


Ilustración 5 Diagrama en Simulink.

Las configuraciones que se realizaron en los siguientes bloques se indican en a continuación:

1.- Configuración del bloque Sine Wave se especifica en la ilustración 6 este bloque representa a el ingreso de la señal de audio.

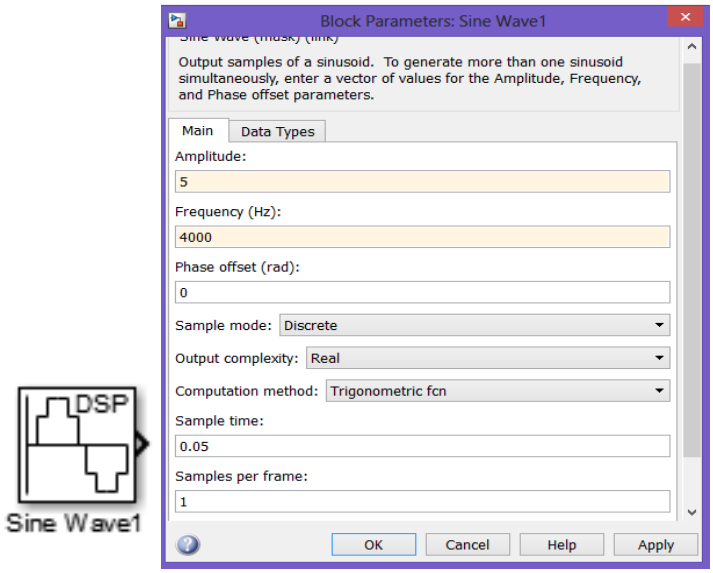


Ilustración 6 Configuración del bloque Sine Wave1.

2.- Configuración del bloque Random Source cumple la función específica de simular el ruido generado en el medio ambiente a continuación se indica en la Ilustración 7 la configuración de parámetros.

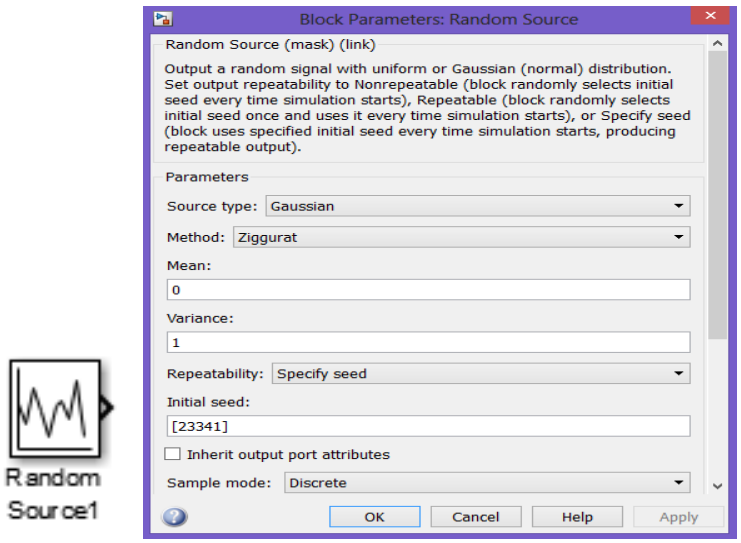


Ilustración 7 Configuración del bloque Random Source.

3.- Configuración de parámetros del Bloque Wiener Filter este permite simular al filtro Wiener FIR el cuál minimizará el error cuadrático proporcionado por el algoritmo LMS Ilustración 8.

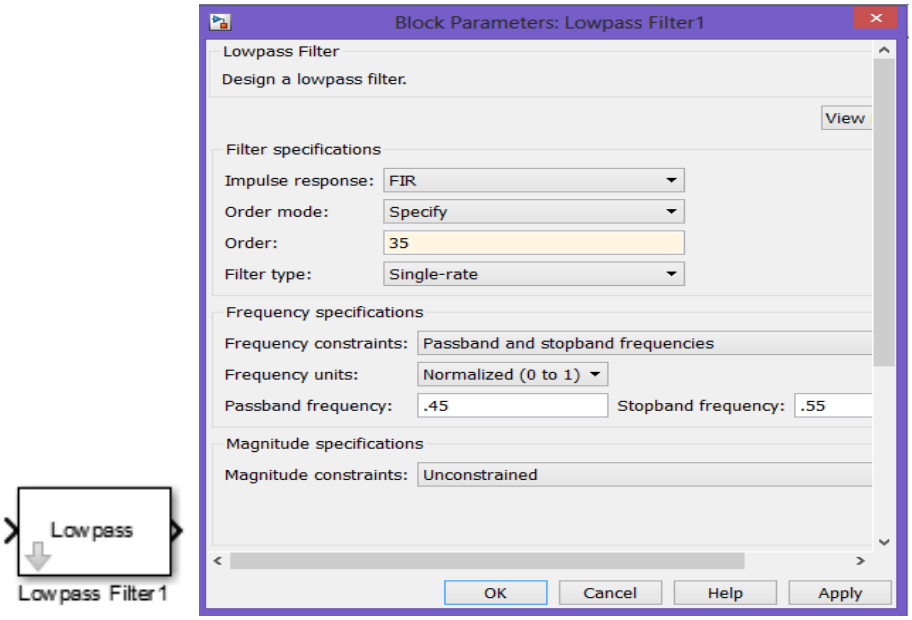


Ilustración 8 Configuración del bloque Wiener Filter.

4.- Configuración del Bloque LMS este permite describir el orden del filtro y el tipo de algoritmo que vamos a utilizar Ilustración 9.

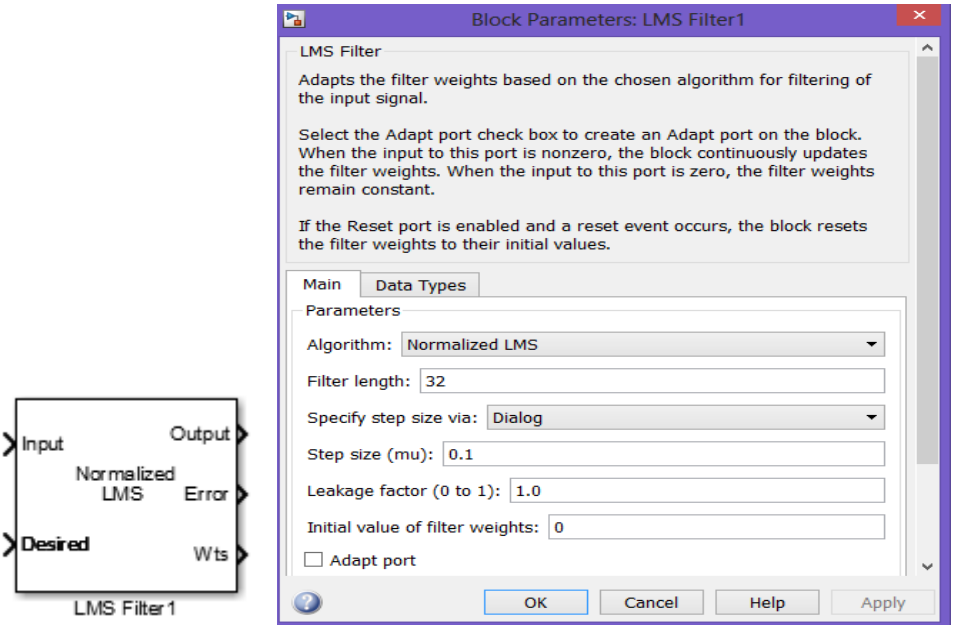


Ilustración 9 Configuración del bloque LMS.

ANEXO 5

CREACIÓN DE UN NUEVO PROYECTO EN KEIL UVISION5

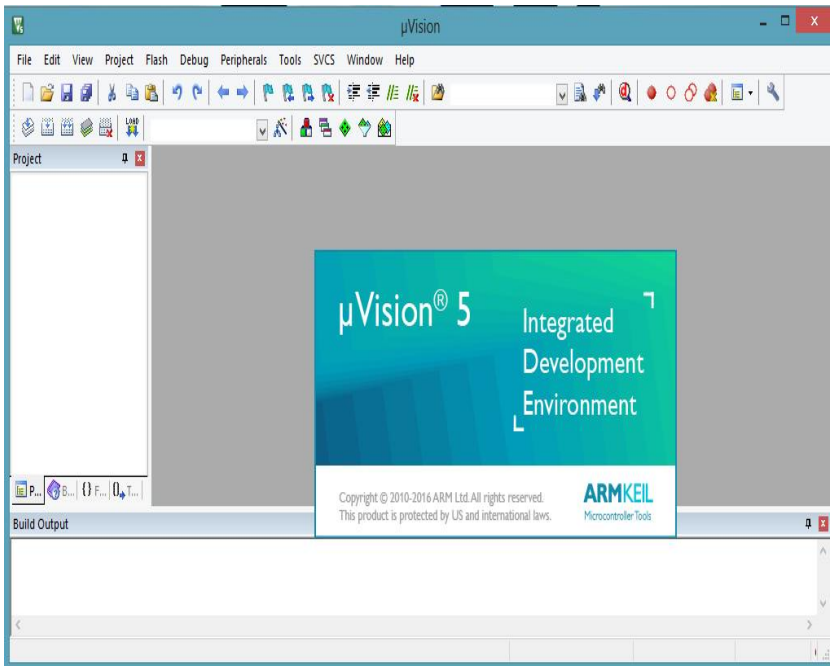


Ilustración 10 Keil uVisión5.

Una vez abierto el programa se procede a crear un nuevo proyecto seleccionando **Project**, a continuación **New uVision Project** mostrado en la Ilustración 11**.**

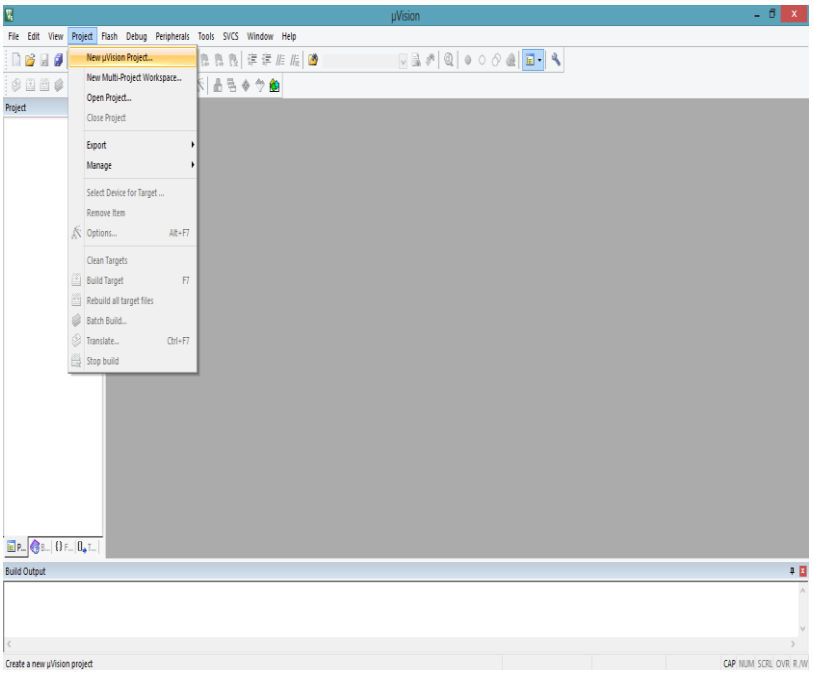


Ilustración 11 Creación de nuevo proyecto Keil.

Una vez creada pedirá guardar el proyecto recién creado Ilustracion 12.

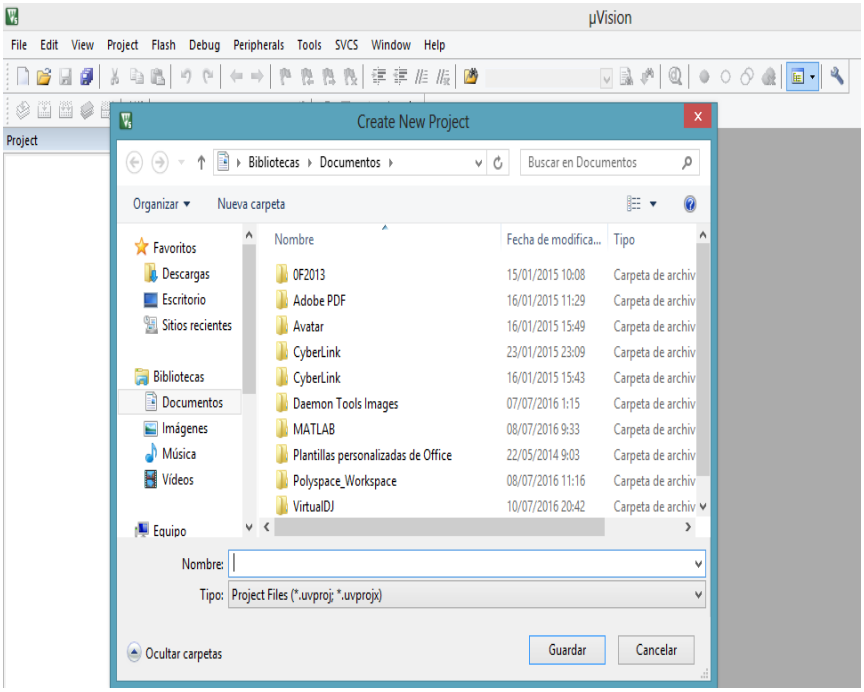


Ilustración 12 Guardar proyecto Keil uVisión.

El siguiente paso es determinar qué tipo de tarjeta de desarrollo se va a utilizar, en nuestro caso STM32F746G Ilustración 13.

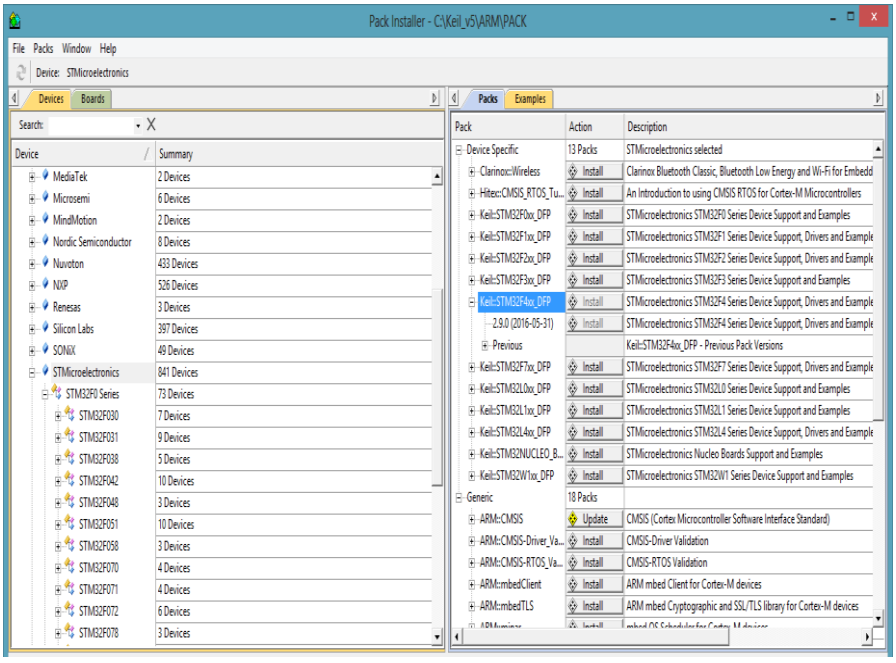


Ilustración 13 Selección de la tarjeta 1.

El entorno de programación se muestra en la Ilustración 14.

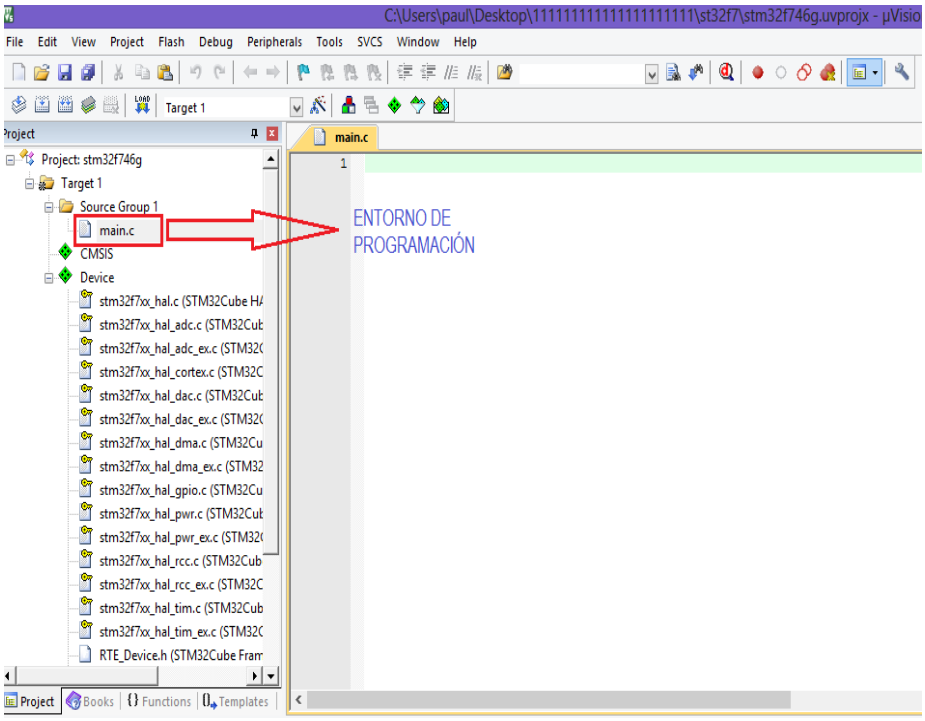


Ilustración 14 Entorno de programación.

Las librerías generadas desde Matlab/Simulink deben ser insertadas en Keil Uvision5, como se indica en la Ilustración 15 E ilustración 16.

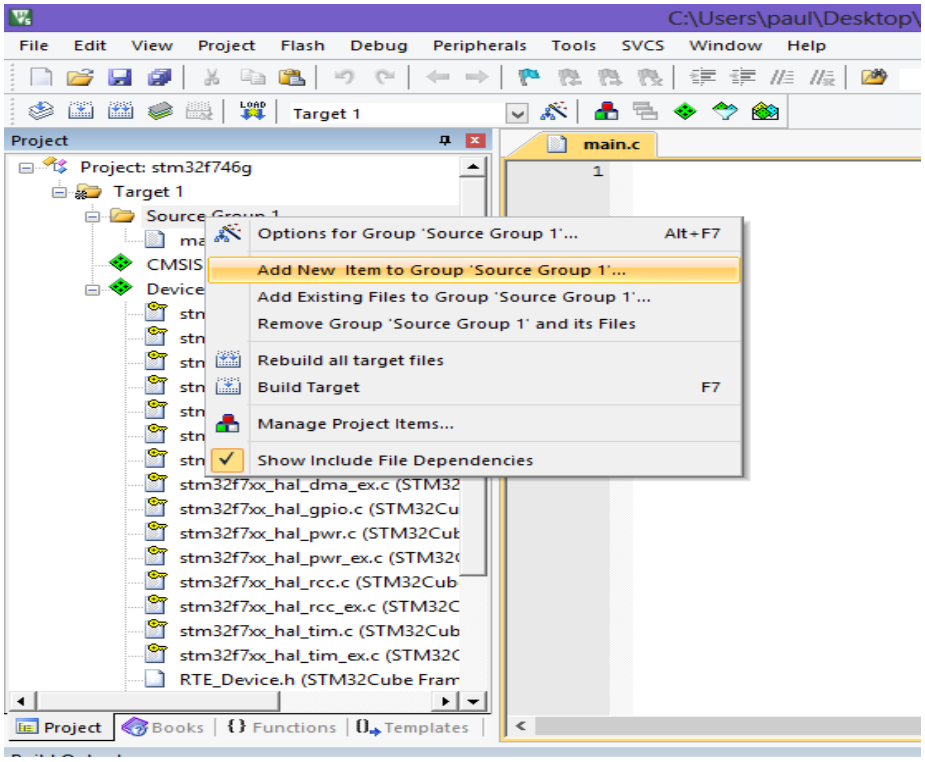


Ilustración 15 Creación de las librerías.

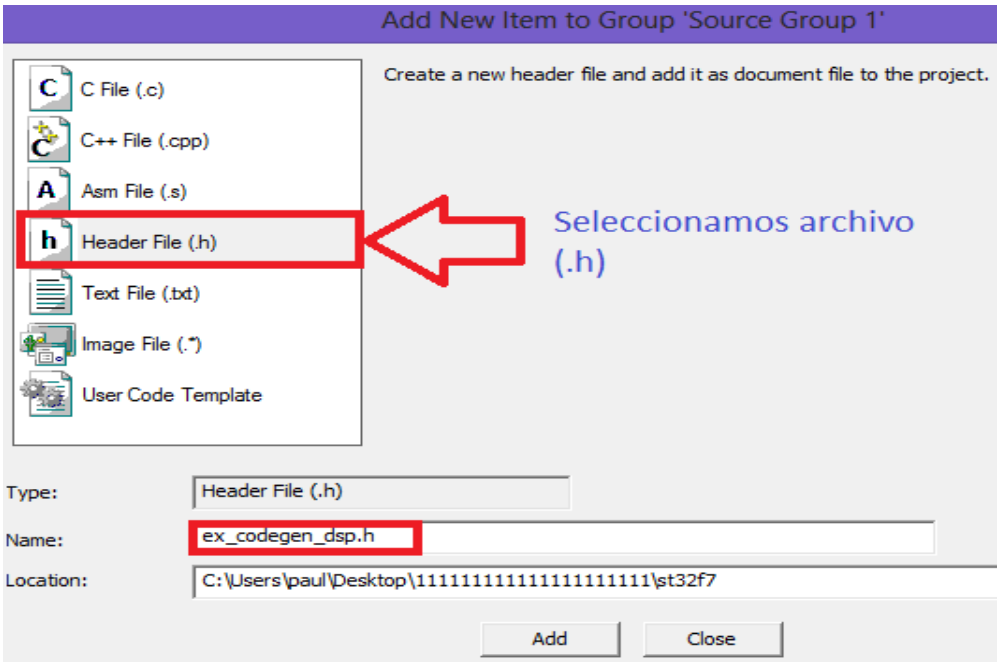


Ilustración 16 Creación de librerías.

El nombre de la librería corresponde al archivo con que se guardó desde Simulink, el cambio de nombre provocaría un error de compilación en Keil uVision5.

Las librerías a insertar se encuentran señaladas con un rectángulo de color rojo mostrado en la Ilustración 17.

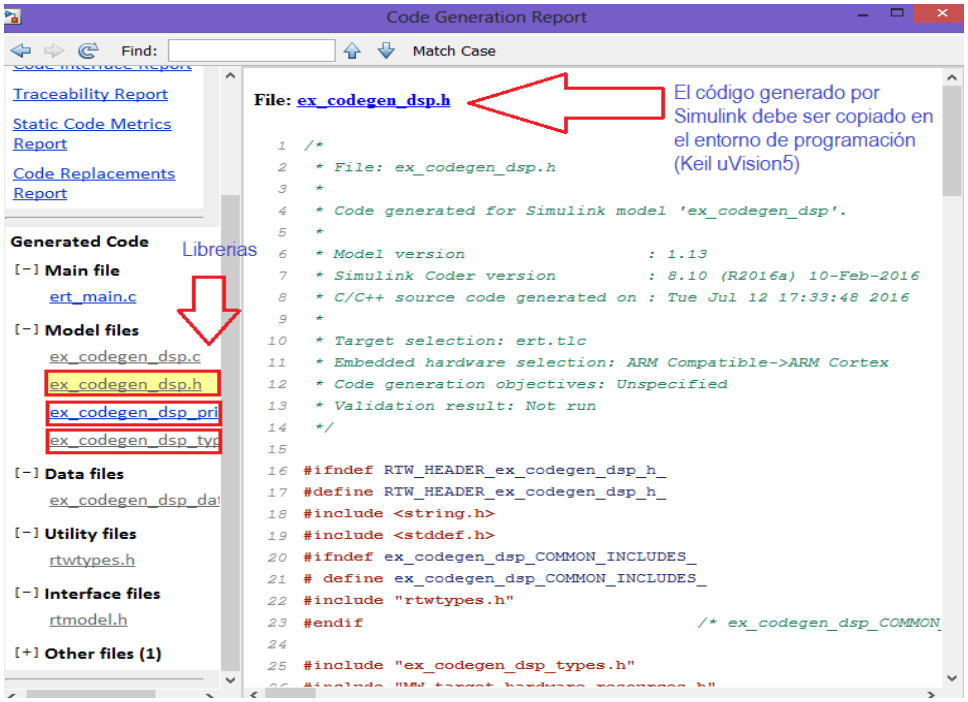


Ilustración 17 Creación de librerías desde Simulink.

CONFIGURACIÓN DE ENTRADAS Y SALIDAS EN EL SOFTWARE STM32CUBEMX

Desarrollador STM32CubeMX

STM32CubeMX es una herramienta gráfica que permite configurar los microcontroladores STM32 muy fácilmente y generar el código de inicialización C correspondiente a través de un proceso paso a paso.

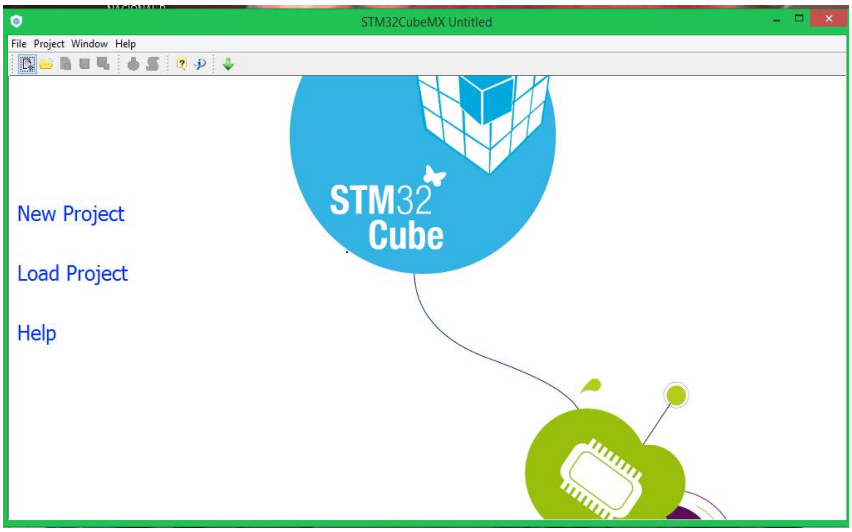


Ilustración 18 STM32CubeMx.

Posteriormente se procede a configurar los pines del dispositivo que serán utilizados en el desarrollo de este proyecto desde STM32CubeMX Ilustración 19.

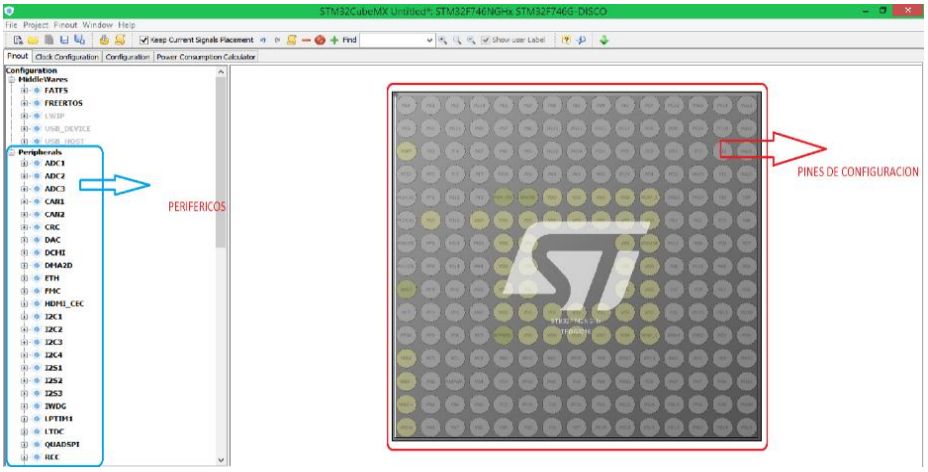


Ilustración 19 Descripción de pines.

La configuración de ADC, DAC, TIMER son de mucha importancia para la conversión y reconstrucción de la señal. En la Ilustración 20 se observa la asignación de pines a utilizar.

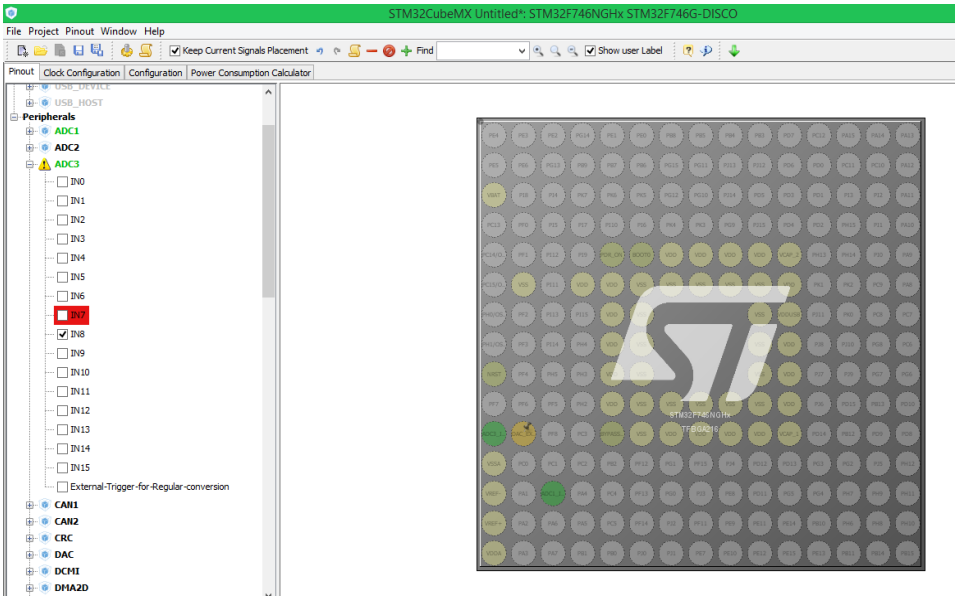


Ilustración 20 Selección de pines

El paso siguiente es configurar los parámetros de ADC, DAC, TIMER seleccionadas para esto clik en configuración, como se muestran en la Ilustración 21 y ilustración 22.

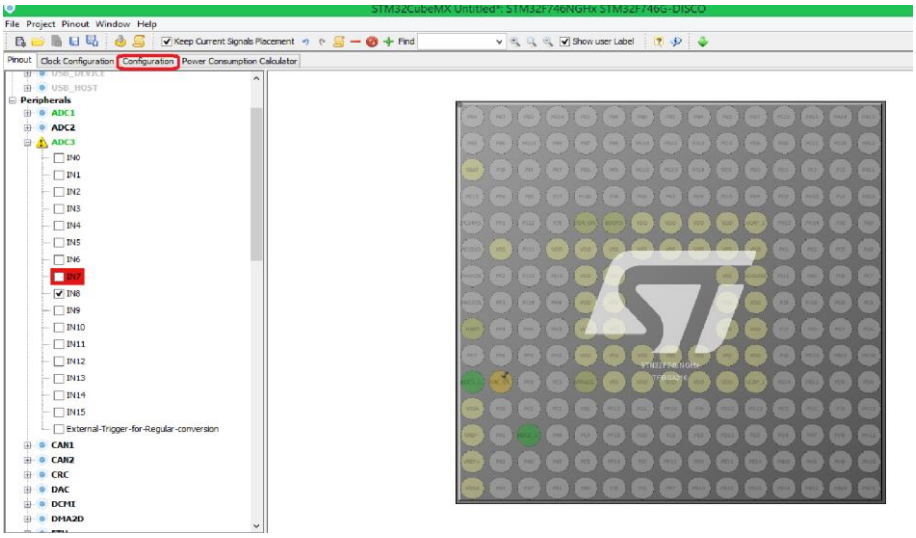


Ilustración 21 Configuración ADC, DAC y TIMER.

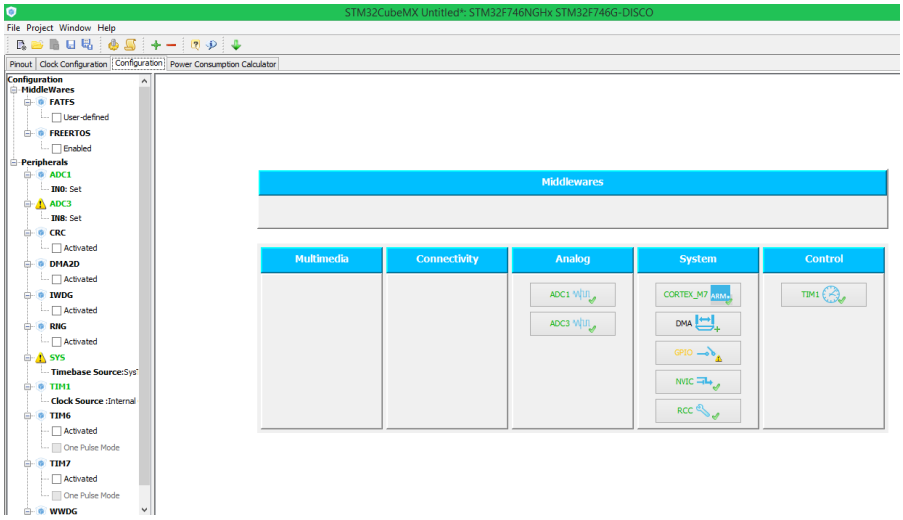


Ilustración 22 Parámetros ADC, DAC, TIMER.

El ADC cumple la función convertir un voltaje analógico de entrada en un valor digital de 12 bits, generando un resultado de 12 bits que se presenta en los registros de datos ADC (ADC Data registres). Ilustración 23.

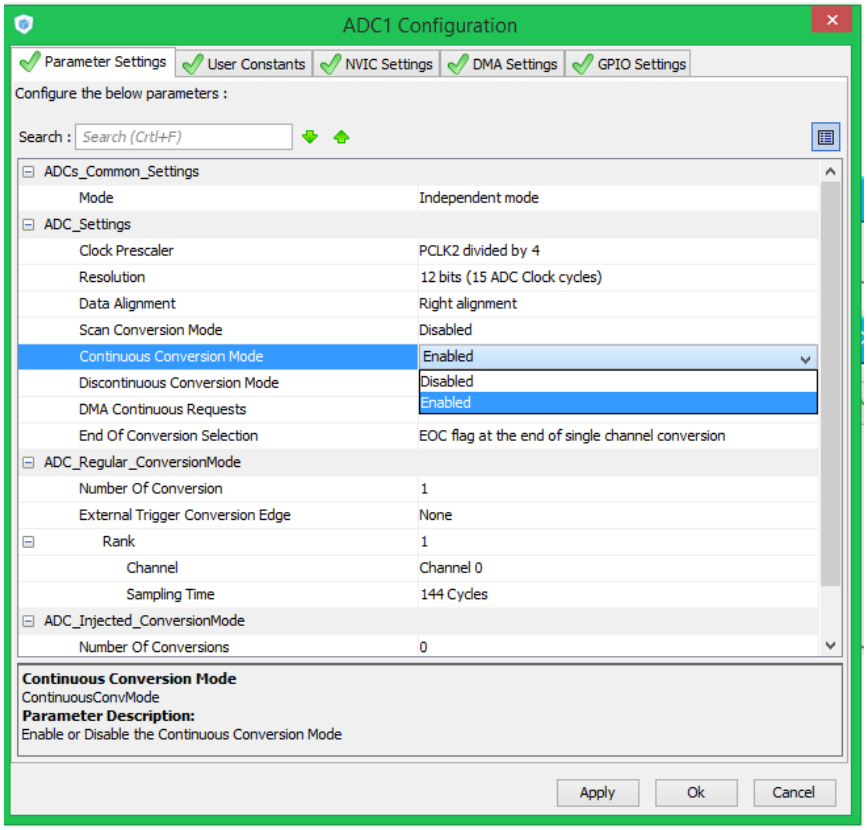


Ilustración 23 Configuración ADC.

Este software permite generar un archivo para el programa Keil uVision5, en donde proporciona el código fuente de las configuraciones y declaraciones de cada una de las librerías e interfaces a utilizar para esto seguiremos la secuencia de las siguientes imágenes Ilustración 24.

Ilustración 24 Generación del Código.

Se despliega una nueva ventana en la que va a pedir guardar el código generado y seleccionar a qué tipo de programador se transformará.

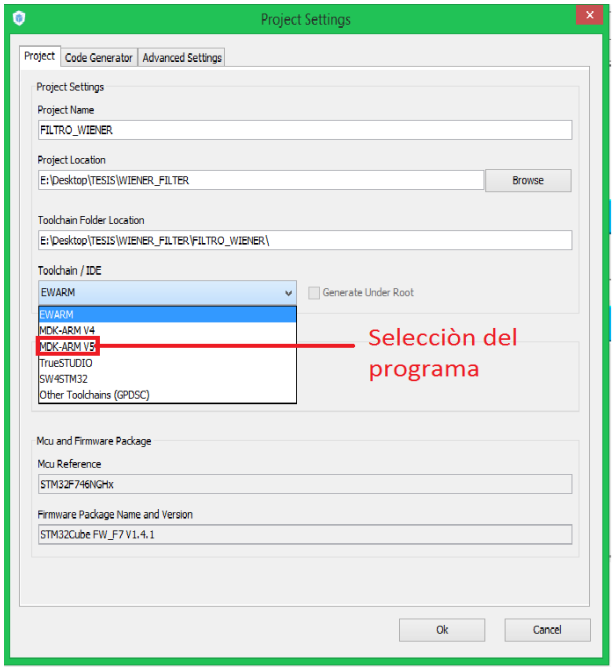


Ilustración 25 Generar Código y guardar.

Por último clik en OK para generar las configuraciones respectivas del ADC, DAC, TIMER, mostradas en la Ilustración 25, a continuación se muestra la programación en lenguaje C de los parámetros establecidos para el ADC, DAC y TIMER.

}

ANEXO 6

CREACIÓN DE UN NUEVO PROYECTO EN STM32CUBEMX

El primer paso consiste en cliquear new proyect como se muestra en la Ilustración 26.

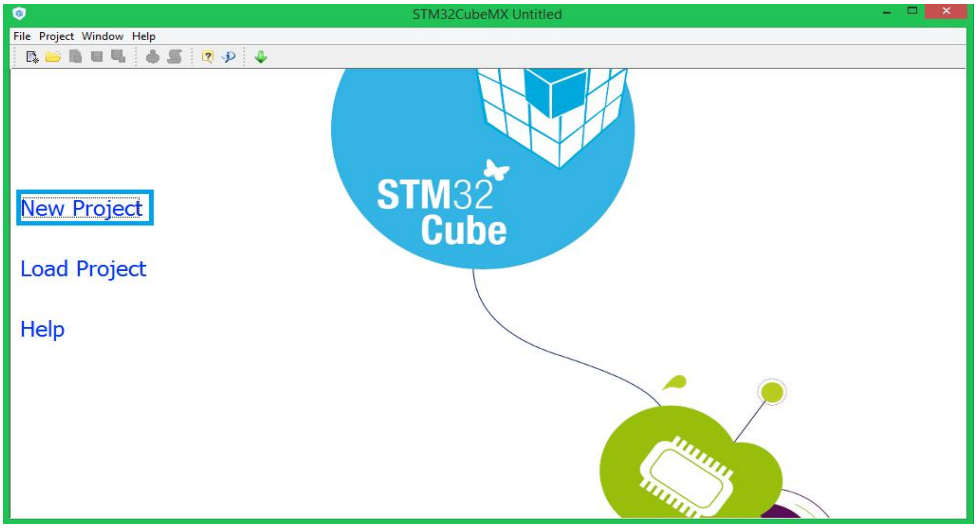


Ilustración 26 Creación de nuevo proyecto.

Una vez creado un nuevo proyecto, se presenta una pantalla idéntica a la Ilustración 27 en la que seleccionamos el dispositivo a utilizar, en este caso es el stm32F46G Discovery Kit, y finalmente en ok.

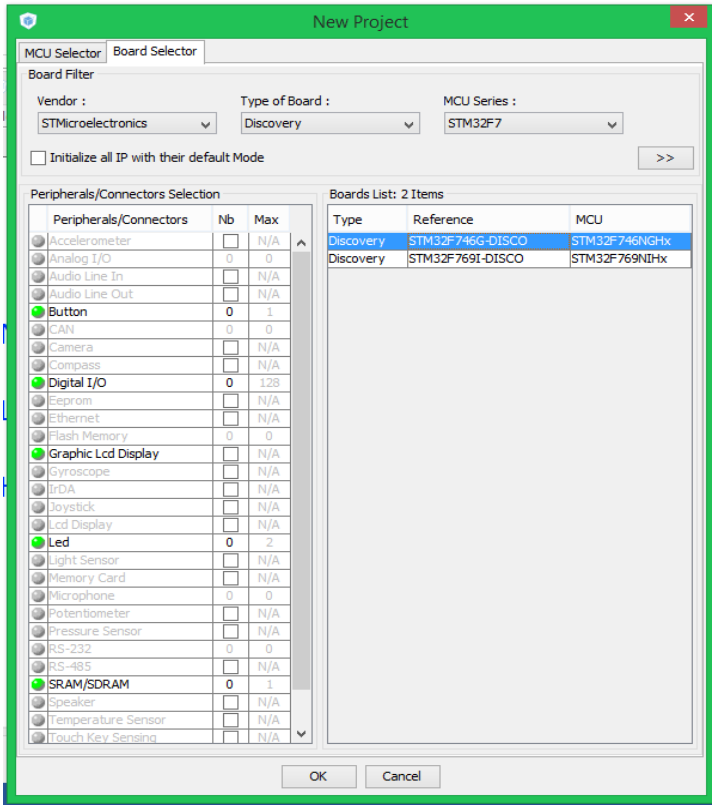


Ilustración 27 Selección del dispositivo.