UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS

# IMPLEMENTACIÓN DE UN DETECTOR DE FRECUENCIA FUNDAMENTAL DE VOZ EN TIEMPO REAL USANDO LA PLATAFORMA STM32F407 DISCOVERY DE STMICROELECTRONICS.

FRANCISCO ALEJANDRO RODRÍGUEZ ARTEAGA
TULIO RAFAEL LORETO PORTILLO

Bárbula, 27 de Noviembre del 2014

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE
TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS

# IMPLEMENTACIÓN DE UN DETECTOR DE FRECUENCIA FUNDAMENTAL DE VOZ EN TIEMPO REAL USANDO LA PLATAFORMA STM32F407 DISCOVERY DE STMICROELECTRONICS.

TRABAJO ESPECIAL DE GRADO PRESENTADO ANTE LA ILUSTRE UNIVERSIDAD DE CARABOBO PARA OPTAR AL TÍTULO DE INGENIERO DE TELECOMUNICACIONES

FRANCISCO ALEJANDRO RODRÍGUEZ ARTEAGA
TULIO RAFAEL LORETO PORTILLO

Bárbula, 27 de Noviembre del 2014

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS


## CERTIFICADO DE APROBACIÓN

Los abajo firmantes miembros del jurado asignado para evaluar el trabajo especial de grado titulado «IMPLEMENTACIÓN DE UN DETECTOR DE FRECUENCIA FUNDAMENTAL DE VOZ EN TIEMPO REAL USANDO LA PLATAFORMA STM32F407 DISCOVERY DE STMICROELECTRONICS.», realizado por los bachilleres FRANCISCO ALEJANDRO RODRÍGUEZ ARTEAGA, cédula de identidad 18.102.824, TULIO RAFAEL LORETO PORTILLO, cédula de identidad 20.261.034, hacemos constar que hemos revisado y aprobado dicho trabajo.

**Firma**
Prof. EDUARDO GONZÁLEZ
TUTOR

**Firma**
Prof. CARLOS MEJÍAS
JURADO

**Firma**
Prof. AHMAD OSMAN
JURADO

Bárbula, 27 de Noviembre del 2014

# Dedicatoria

*A mi esposa e hijos.*

**FRANCISCO ALEJANDRO RODRÍGUEZ ARTEAGA**

*A Dios, mi Señor y guía.*

*A mi familia.*

*A mis amistades.*

**TULIO RAFAEL LORETO PORTILLO**

# Agradecimientos

Agradecemos a Dios y a nuestras familias, por el apoyo brindado durante la realización de este proyecto.

A nuestro tutor, Eduardo González, por la ayuda y guía prestada a lo largo de la investigación e implementación.

A todos aquellos profesores y compañeros de estudio que de alguna forma influyeron en la realización de este trabajo especial de grado.

# Índice general

# Índice de figuras

# Indice de tablas

# Acrónimos

| | |
|---|---|
| **ADC** | **A**nalog to **D**igital **C**onverter |
| **CMSIS** | **C**ortex **M**icrocontroller **S**oftware **I**nterface **S**tandard |
| **CRC** | **C**yclic **R**edudancy **C**heck |
| **DAC** | **D**igital to **A**nalog **C**onverter |
| **DMA** | **D**irect **M**emory **A**ccess |
| **DSP** | **D**igital **S**ignal **P**rocessing |
| **FPU** | **F**loating **P**oint **U**nit |
| **GPIO** | **G**eneral **P**urpose **I**nput **O**utput |
| **PC** | **P**ersonal **C**omputer |
| **RTC** | **R**eal **T**ime **C**lock |
| **SNR** | **S**ignal to **N**oise **R**atio |
| **UC** | **U**niversidad de **C**arabobo |

**IMPLEMENTACIÓN DE UN DETECTOR DE FRECUENCIA FUNDAMENTAL DE VOZ EN TIEMPO REAL USANDO LA PLATAFORMA STM32F407 DISCOVERY DE STMICROELECTRONICS.**

por

FRANCISCO ALEJANDRO RODRÍGUEZ ARTEAGA y TULIO RAFAEL LORETO PORTILLO

Presentado en el Departamento de Señales y Sistemas
de la Escuela de Ingeniería en Telecomunicaciones
el 27 de Noviembre del 2014 para optar al Título de
Ingeniero de Telecomunicaciones

## RESUMEN

Los sistemas de comunicaciones AM-SSB presentan errores de frecuencia a la hora de demodular la señal debido a la falta de sincronía entre los mezcladores del modulador y del demodulador, por lo que la detección de los mismos permite realizar correcciones para lograr una mejor comunicación. El cálculo de la frecuencia fundamental de la voz es una de las etapas requeridas para la elaboración de algoritmos de corrección de errores de corrimiento de frecuencia, por lo que el presente

proyecto tiene por objetivo la implementación de un detector de frecuencia fundamental de la voz en tiempo real en la plataforma de desarrollo STM32F407, la cual aprovecha el lenguaje de programación C y módulos propios de ADC y FPU para facilitar la implementación. Para ello, se escogió el algoritmo de CEPSTRUM como base del programa de estimación de la frecuencia fundamental; luego se diseñó el programa final en lenguaje C, aprovechando las herramientas y librerías provistas por el fabricante de la plataforma, incorporando etapas de preprocesamiento y postprocesamiento de la señal de voz. Se implementó el programa en la plataforma de desarrollo mediante el entorno de desarrollo IAR Embedded Workbench 6.30 para la escritura y compilación del código, adaptando un display para la visualización de los resultados y un circuito de elevación de nivel de señal para modificar la señal de entrada, de modo que pudiera ser muestreada correctamente a través del ADC. Seguidamente, se realizaron pruebas estadísticas para la estimación de la frecuencia fundamental de señales de voz con parámetros conocidos, determinando que el detector funciona en tiempo real y tiene mayor precisión para voces masculinas que femeninas, arrojando errores de estimación menores en el caso de las primeras. Se determinó que, en general, se requiere una SNR mayor a 2 dB para realizar estimaciones confiables de la frecuencia fundamental de voces masculinas, y mayor a 3 dB para voces femeninas. Finalmente, se realizaron estimaciones de frecuencia fundamental a señales de voz con frecuencia desconocida, para sujetos femeninos y masculinos.

Palabras Claves: CEPSTRUM, frecuencia, fundamental, plataforma, desarrollo, tiempo, real

Tutor: EDUARDO GONZÁLEZ
Profesor del Departamento de Señales y Sistemas
Escuela de Telecomunicaciones. Facultad de Ingeniería

# Capítulo I

# Introducción

## 1.1. Motivación

En un sistema de comunicaciones AM-SSB existen errores de frecuencia a la hora de demodular la señal debido a la falta de sincronía entre los mezcladores del modulador y del demodulador. Pequeñas diferencias en la frecuencia de referencia del transmisor respecto a la frecuencia de referencia del receptor ocasionan que el filtro de la etapa demoduladora no se encuentre centrado de manera exacta en la frecuencia de la señal mensaje, produciendo esto que la información llegue distorsionada, con la posibilidad de que el usuario receptor no entienda el mensaje. Los receptores AM-SSB tradicionales no son capaces de corregir automáticamente esta diferencia en la frecuencia de las señales, sino que cuentan con perillas o instrumentos que deben ser manejados manualmente hasta lograr corregir al máximo el error de frecuencia.

La mayoría de los algoritmos que se han diseñado para la detección del corrimiento se pueden dividir en grupos independientes que tienen sus propias dificultades, una de las cuales es la detección de la frecuencia fundamental de la voz. A. Michael Noll (1966) propuso la utilización del CEPSTRUM como técnica para la estimación de la frecuencia fundamental de señales de voz[1]. En su investigación,

Noll realizó el modelado matemático del cálculo del CEPSTRUM de una señal y demostró que el mismo muestra un espectro, equivalente al dominio del tiempo, que tiene picos marcados en el período fundamental. Métodos como el SIFT, basado en autocorrelación y propuesto por Markel[2], algoritmos basados en espectrograma propuestos por Díaz et al[3] y técnicas basadas en filtrado armónico escalado, propuestas por Roa et al[4], también han sido diseñados para abordar el problema de cálculo de la frecuencia fundamental de la voz. En la universidad de Hertfordshire (Reino Unido), Al-Langawi (2004) realizó un proyecto con la finalidad de investigar una forma de sintonizar un receptor de banda lateral única (SSB) automáticamente con la mínima intervención posible del usuario-operador, necesitando para ello guardar las señales recibidas en el receptor SSB en un formato compatible con el software para el análisis, calcular la frecuencia fundamental de voz de dicha señal y ajustar el receptor hacia arriba o hacia abajo, de modo que se sintonice a la frecuencia calculada[5]. Se calculó la frecuencia fundamental utilizando tres métodos implementados en el software MATLAB: cepstrum, autocorrelación y algoritmo de cálculo de pitch, llegando a la conclusión de que este último arrojaba los resultados más fieles y cercanos a la realidad.

A pesar de la existencia de múltiples algoritmos, el cálculo de la frecuencia fundamental sigue siendo uno de los problemas más difíciles de resolver en los análisis acústicos de voz, debido principalmente a diferentes factores que generan dificultades en la determinación precisa de la posición y altura de sus máximos, entre los cuales se pueden citar: los concernientes al enventanamiento y muestreo de las señales de voz, la presencia de perturbaciones en el registro electrónico de las mismas, la no estacionalidad para algunos segmentos, la falta de periodicidad en la excitación glotal, la dificultad para ejecutar segmentación de inicio y final en zonas sonoras, y el amplio margen de variación para la frecuencia fundamental.

En vista de lo anterior, se requiere crear un detector de frecuencia fundamental de voz utilizando para ello la plataforma STM32F407 Discovery de STMicroelectronics, basado en la arquitectura ARM, que a través de un lenguaje de programación de alto nivel y librerías que provee el fabricante es posible integrar y acoplar con otros módulos de procesamiento digital de señales (DSP: *Digital Signal Processing*)

para formar un dispositivo capaz de detectar el error de frecuencia en un sistema de comunicación AM-SSB y de corregir automáticamente los errores.

ARM es una arquitectura con conjunto de instrucciones reducidas (RISC*: Reduced Instruction Set Computer*) de 32 bits desarrollada por ARM Holdings. Con la diversa variedad de propiedad intelectual de la empresa y el amplio conjunto de tecnologías de semiconductores y software para soluciones basadas en esta arquitectura, los principales fabricantes de equipos (OEM: *Original Equipment Manufacturers*) usan esta tecnología en una gran variedad de aplicaciones que abarcan terminales móviles, decodificadores digitales, enrutadores de red, entre otros. Actualmente la tecnología ARM se usa en el 95 % de los teléfonos inteligentes, 80 % de cámaras digitales y 35 % del total de dispositivos electrónicos[6].

La utilización de dispositivos desarrollados por software y DSP, como el detector de frecuencia fundamental de voz planteado, forman parte de las bases necesarias para la creación de lo que se denomina Radio Definida por Software (SDR: *Software Defined Radio*), lo cual consiste en la implementación de radios por medio de software y procesamiento digital de señales en lugar de utilizar hardware convencional.

El presente proyecto, de carácter factible, aprovecha la plataforma de desarrollo STM32F407 Discovery para realizar un detector de frecuencia fundamental de voz de relativo bajo costo, logrando de esta manera obtener una etapa importante en la elaboración de receptores AM-SSB capaces de realizar cálculo de errores de corrimiento de frecuencia de manera automática, integrando para ello técnicas de procesamiento digital de señales.

## 1.2. Objetivos

### 1.2.1. General

Implementar un detector de frecuencia fundamental de voz en tiempo real usando la plataforma STM32F407 Discovery de STMicroelectronics.

### 1.2.2. Específicos

- Seleccionar un algoritmo de detección de frecuencia fundamental de voz a partir de un conjunto de algoritmos más conocidos.
- Desarrollar el diagrama de bloques y de flujo del algoritmo seleccionado.
- Seleccionar las herramientas disponibles más adecuadas, tales como librerías y métodos que provee el fabricante del hardware.
- Diseñar el programa a utilizar para el cálculo de la frecuencia fundamental de voz y su visualización a través de una unidad gráfica elemental, basado en el algoritmo y las librerías seleccionadas.
- Implementar el algoritmo seleccionado en el kit de desarrollo.
- Verificar el funcionamiento del dispositivo utilizando archivos de prueba estandarizados.
- Realizar pruebas con señales reales para verificar el correcto funcionamiento del dispositivo.

## 1.3. Alcance

Se implementó un detector de frecuencia fundamental de voz en tiempo real usando la plataforma STM32F407 Discovery de STMicroelectronics, la cual utiliza una unidad microcontroladora (MCU: *Microcontroller Unit*) ARM Cortex-M4 de 32 bits que opera a 168 MHz, con co-procesador matemático (FPU: *Floating-Point Unit*), 192 kB de memoria RAM y 1 MB de memoria flash; para ello se empleó uno de los algoritmos ya creados y disponibles para el análisis acústico de voz: CEPSTRUM, utilizando librerías y herramientas en lenguaje C disponibles y ofrecidas o recomendadas por los fabricantes. La frecuencia detectada se muestra en una unidad gráfica elemental modelo Parallax Serial LCD 27977 que se incorporó al kit de desarrollo.

## 1.4. Recursos utilizados

- Plataforma STM32F407 Discovery de STMicroelectronics.
- Unidad gráfica elemental.
- Software IAR Embedded Workbench for ARM 6.30 de IAR Systems.
- Libros.
- Artículos de investigación.
- Computadoras.
- Ingeniero Eduardo González.

## 1.5. Resultados esperados

Se espera detectar correctamente en tiempo real la frecuencia fundamental de la señal de voz, de forma tal que esta información pueda ser utilizada por otros módulos para determinar el corrimiento en frecuencia que tuvo la señal AM-SSB y poder corregirlo, sintonizando así la frecuencia correcta de manera automática.

# Capítulo II

# Marco conceptual

## 2.1. Bases Teóricas

### 2.1.1. Sistemas de comunicación basados en modulación en banda lateral única (SSB)

#### 2.1.1.1. Introducción

Un sistema de comunicación se refiere a cualquier sistema cuyo objetivo principal sea el de transmitir información de un punto a otro.

Couch (2008) establece que los elementos fundamentales que conforman un sistema de comunicaciones son el transmisor (constituido generalmente por un codificador y un modulador), el canal (el cual es el medio de transmisión) y el receptor (constituido por un demodulador y un decodificador)[7]; la figura 2.1 muestra el diagrama de bloques de un sistema de comunicación genérico.

Los sistemas de comunicación se pueden dividir en dos tipos, según el canal que utilizan para la transmisión: cableados e inalámbricos. Los sistemas cableados son aquellos que requieren de líneas de transmisión entre el transmisor y el receptor para transportar la información deseada, mientras que los sistemas de comunicación inalámbricos son capaces de transportar la información a través del espacio libre,

utilizando para ello diversas técnicas, entre las cuales se encuentra la modulación en amplitud en banda lateral única.

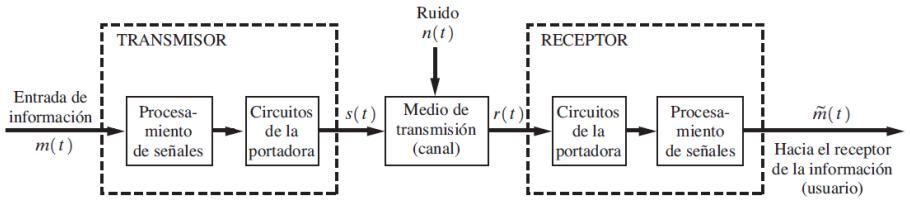


**Figura 2.1:** Diagrama de bloques de un sistema de comunicaciones genérico. *Fuente: Couch (2008) "Sistemas de Comunicación Digitales y Analógicos".*

#### 2.1.1.2. Modulación en amplitud en banda lateral única (SSB-AM)

La modulación en banda lateral única se deriva a partir de una modulación en amplitud convencional, la cual consiste en imponer la información contenida en una señal de baja frecuencia hacia una señal de alta frecuencia, por medio de la variación de la amplitud de esta última.

La señal de baja frecuencia o de información se denomina modulante, la señal de alta frecuencia se denomina portadora y la señal resultante se denomina señal modulada. Cuando se modula una señal en amplitud de manera convencional se forman dos bandas laterales alrededor de la señal portadora, cada una de las cuales posee exactamente la misma información.

La modulación AM en banda lateral única consiste en suprimir la potencia de la señal portadora y de una de las bandas laterales de la señal modulada, lo cual trae consigo varias ventajas: menor consumo de potencia, menor cantidad de ruido (debido a la disminución del ancho de banda con respecto a AM convencional), optimización del espectro disponible para la transmisión, entre otros[7].

Couch (2008) define dos tipos de señales de banda lateral única: superior e inferior. Una señal de banda lateral única superior (USSB) tiene un espectro con valor

de cero para $|f| < f_c$, mientras que una señal de banda lateral única inferior (LSSB) tiene un espectro con valor de cero para $|f| > f_c$, donde $f_c$ es la frecuencia de la portadora[7].

Una señal de banda lateral única se obtiene utilizando la envolvente compleja

$$g(t) = A_c \cdot [m(t) \pm j \cdot \widehat{m}(t)] \tag{2.1}$$

de la cual resulta la forma de onda de señal SSB s(t)

$$s(t) = A_c \cdot [m(t) \cdot \cos(\omega_c t) \pm j \cdot \widehat{m}(t) \cdot sen(\omega_c t)] \tag{2.2}$$

donde la USSB se obtiene empleando el signo (-) y la LSSB se obtiene empleando el signo (+); $m(t)$ es la señal moduladora; $\widehat{m}(t)$ denota la transformada de Hilbert de $m(t)$, la cual se obtiene de

$$\widehat{m}(t) = m(t) \star h(t) \tag{2.3}$$

donde

$$h(t) = \frac{1}{\pi t} \tag{2.4}$$

y $H(f) = \mathcal{F}[h(t)]$ corresponde a una red de corrimiento de fase de $-90^o$

$$H(f) = \begin{cases} j, & f > 0 \\ -j, & f < 0 \end{cases} \tag{2.5}$$

La figura 2.2 muestra el espectro de una señal de banda lateral única superior obtenida a partir de la ecuación 2.2. Si la señal moduladora $m(t)$ tiene un espectro como el de la figura 2.2 (a), para obtener una señal USSB es necesario que la envolvente compleja $g(t)$ tenga un espectro de cero para las frecuencias negativas, como

se ve en la figura 2.2 (b). Finalmente, la señal modulada s(t) posee un espectro como el de la figura 2.2 (c), el cual corresponde a una señal USSB.

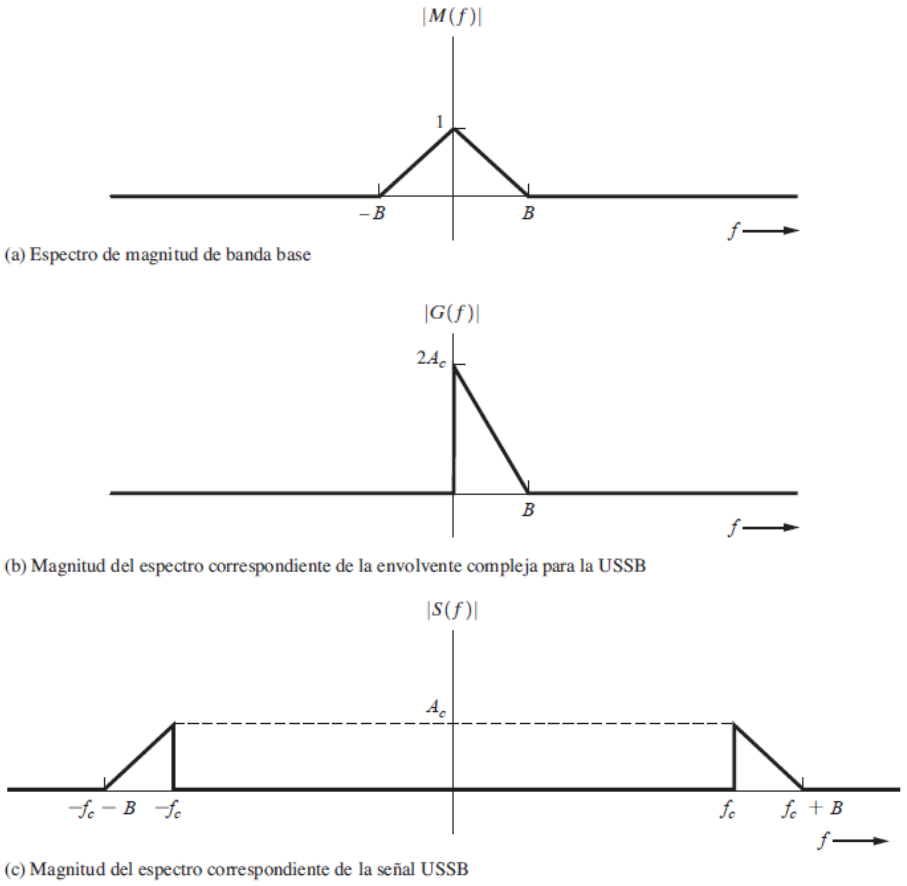


**Figura 2.2:** Espectro para una señal de banda lateral única superior. *Fuente: Couch (2008) "Sistemas de Comunicación Digitales y Analógicos".*

Según el propio Couch (2008), existen dos técnicas principales para la generación de señales AM de banda lateral única, el *método de puesta en fase*, basado en un generador de señales en cuadratura y el cual trabaja en banda base, y el *método de filtrado*, que realiza procesamientos en RF y utiliza un filtro de banda lateral para generar una señal SSB equivalente sin necesidad de trabajar en banda base. Este último es el más popular entre los métodos de generación de señales AM-SSB, ya que se pueden crear filtros de gran supresión de banda lateral al utilizar cristales

para su elaboración, los cuales resultan relativamente económicos al producirse en cantidad a frecuencias IF estándares[7]. La figura 2.3 (a) muestra el diagrama de bloques para la generación de una señal SSB utilizando el método de puesta en fase y la figura 2.3 (b) muestra el diagrama de bloques del método de filtrado de banda lateral.

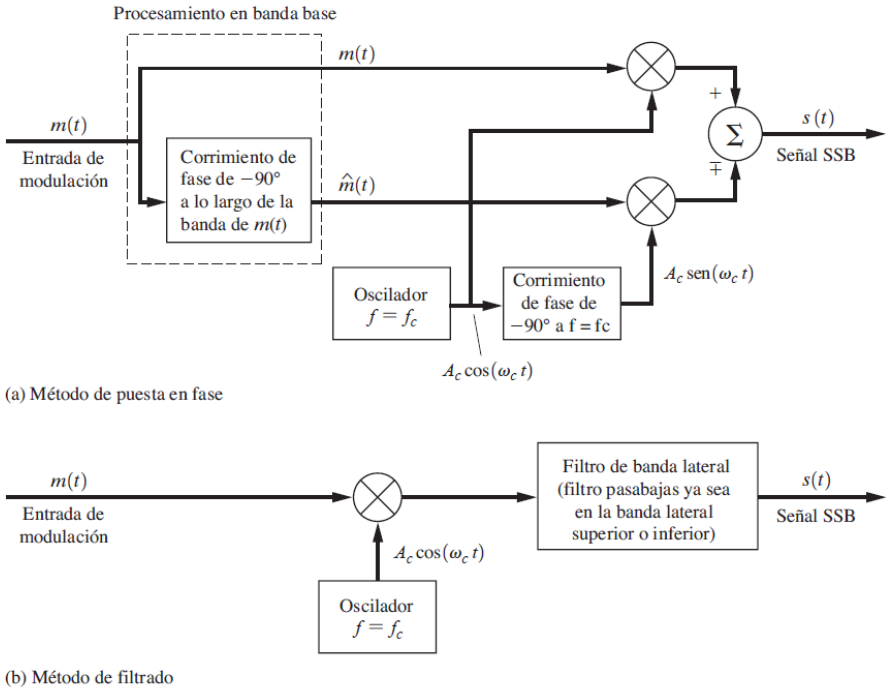


**Figura 2.3:** Métodos de generación de señales AM de banda lateral única. *Fuente: Couch (2008) "Sistemas de Comunicación Digitales y Analógicos".*

#### 2.1.1.3. Receptores de señales moduladas en banda lateral única

Un receptor SSB es un equipo que recibe una señal modulada en una banda lateral de la portadora y la procesa de modo tal que se pueda extraer la información de la misma; pueden ser receptores de banda superior o de banda inferior y los mismos demodulan solamente la señal de banda lateral correspondiente, esto es, un receptor de banda superior no es capaz de demodular una señal de banda lateral inferior y uno de banda inferior no demodula señales de banda lateral superior[5].

La figura 2.4 muestra el diagrama de bloques de un receptor SSB; se observa que al recibir la señal SSB por medio de la antena, la misma es amplificada y mezclada con un oscilador local que lleva la señal a frecuencia intermedia para luego ser filtrada y amplificada nuevamente; luego la señal es demodulada utilizando un detector de producto, el cual es un tipo de mezclador de frecuencia que toma el producto de la señal modulada y un oscilador local BFO (Beat Frequency Oscillator) para producir una copia del audio original. Finalmente la señal de audio es amplificada con un amplificador de audiofrecuencia y llevada a la salida correspondiente[5].

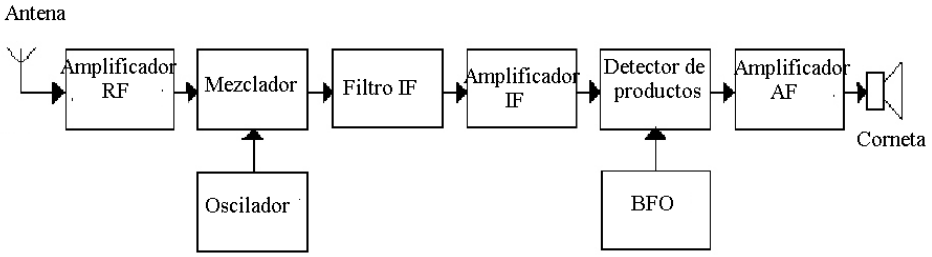


**Figura 2.4:** Diagrama de bloques de un receptor SSB. *Fuente: Al-Langawi (2004) "Automatic Tuning of an SSB Receiver".*

### 2.1.2. Acústica

#### 2.1.2.1. Producción del habla

Cheng (2009) establece que el generador del habla de un humano consiste de: pulmones, tráquea, laringe (cuerdas vocales), garganta (cavidad faríngea), boca (cavidad oral), nariz (cavidad nasal) y articuladores (labios, paladar suave, lengua, dientes), como se muestra en la figura 2.5[8].

El habla se produce mientras el aire viaja desde los pulmones hasta la tráquea, y luego hasta la laringe. El volumen del aire determina la amplitud del sonido. El

flujo de aire continúa viajando hasta la la boca y/o la nariz, y es *soplado* en los labios o nariz para convertirse en una onda acústica de presión. Esta onda se denomina *habla* y se origina del movimiento voluntario de los componentes del generador de habla. La combinación de garganta, boca y nariz se denomina tracto vocal; un tracto vocal tiene una cierta resonancia característica que depende del tamaño y forma del mismo. Esta forma y tamaño puede ser modificada cuando se colocan los articuladores en diferentes posiciones, lo que permite generar un amplio espectro de ondas[8].

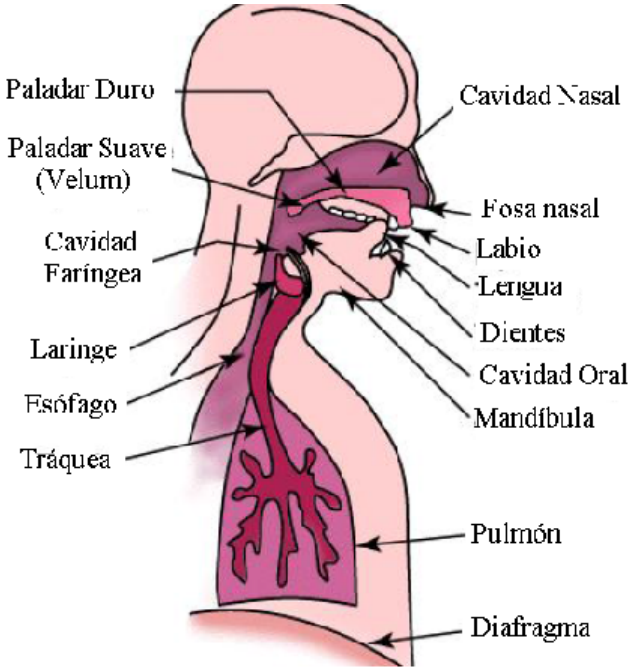


**Figura 2.5:** Generador de habla de un humano. *Fuente: Cheng (2009) "Introductory speech processing".*

Las dos componentes principales en la producción del habla son el tracto vocal y la excitación. La excitación crea un sonido al mover el aire rápidamente y el tracto vocal da forma al sonido para agregarle características únicas al mismo.

La excitación tiene tres formas principales: fonación, fricativa y oclusiva. La fonación se produce por la vibración de las cuerdas vocales o membrana glotis; los

sonidos fricativos se originan por flujos de aire turbulento y los oclusivos se producen por el cierre del tracto vocal, seguido por una liberación repentina de aire[8].

El tracto vocal consiste en una serie de cavidades resonantes bordeadas por estructuras anatómicas de naturaleza fijas o móviles. Un modelo realista del mismo consiste en un tubo no uniforme cerrado al final del glotis y abierto en la boca. Un modelo simplificado del tracto vocal consiste en un cilindro cerrado en el extremo del glotis y abierto en el extremo de la boca[8].

#### 2.1.2.2. Clases de sonidos

Según Al-Langawi (2004), los sonidos, de acuerdo a los modos de excitación del sistema vocal que pueden ser controlados por la fuerza del flujo del aire en las cuerdas vocales, pueden clasificarse en tres clases: sonidos expresivos (*voiced sounds*), sonidos sordos o fricativos (*unvoiced sounds*) y sonidos oclusivos (*plosive sounds*).

**Sonidos expresivos o sonoros** Se refiere a los sonidos producidos por la vibración de las cuerdas vocales debido al flujo de aire de los pulmones; estas vibraciones de las cuerdas producen ondas triangulares y periódicas, cuya frecuencia es rica en armónicos múltiplos de la frecuencia fundamental de vibración.

La tasa de vibración de la cuerda vocal depende de la presión de aire desde los pulmones, la cual puede ser controlada por el hablante. Los sonidos expresivos incluyen las vocales y algunas consonantes, como la [m, n, l]. La figura 2.6 (a) muestra una forma de onda típica de este tipo de sonidos.

**Sonidos sordos o fricativos** Esta clase de sonidos se generan sin vibración alguna de las cuerdas vocales, y se produce al hacer una constricción del tracto vocal en algunos de sus puntos, obligando al aire a fluir a través de él a una alta velocidad para producir turbulencia. La forma de onda típica de estos sonidos se muestran el la figura 2.6 (b).

**Sonidos oclusivos o plosivos** Se producen al cerrar completamente el flujo de aire hacia el tracto vocal, seguido de una liberación repentina del aire, generando así el sonido.

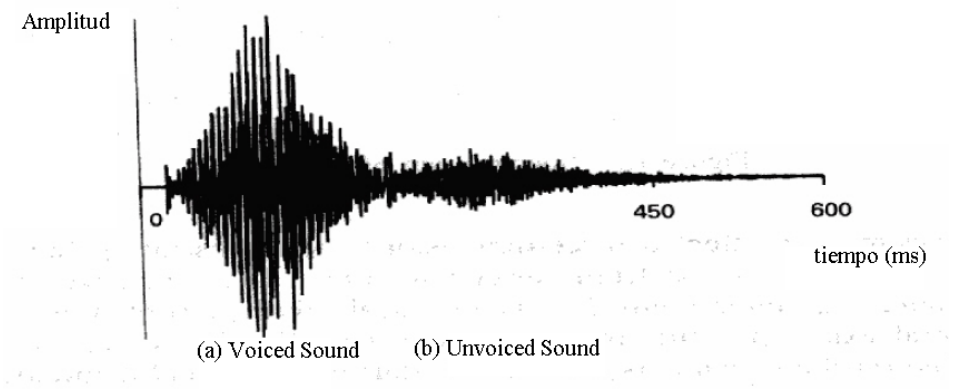


**Figura 2.6:** Formas de ondas típicas de: (a) Sonidos expresivos y (b) Sonidos fricativos. *Fuente: Al-Langawi (2004) "Automatic Tuning of an SSB Receiver".*

### 2.1.3. Frecuencia fundamental de la voz y pitch

Todo ser humano tiene un rango de frecuencia que utiliza para comunicarse, y el mismo está determinado por la física relacionada con la faringe y tracto vocal del individuo. Para los hombres, este rango usualmente se encuentra entre los 50 y 250 Hz, mientras que para las mujeres normalmente va desde los 120 a los 500 Hz[8]. La frecuencia fundamental de la voz puede variar en un individuo como respuesta a factores como el estrés, entonación y las emociones[9].

La frecuencia fundamental es uno de los parámetros de mayor importancia en el procesamiento y análisis de señales de voz, y su estimación constituye un paso fundamental en la implementación de receptores SSB autosintonizables. Algunas de las definiciones, así como técnicas y algoritmos de gran utilidad a la hora de estimar la frecuencia fundamental de la voz, son descritos a continuación.

#### 2.1.3.1. Definiciones de pitch y frecuencia fundamental de la voz.

Es importante conocer la diferencia que existe entre frecuencia fundamental de la voz y pitch, ya que es común que estos términos se intercambien entre sí de manera errónea.

Existen varias definiciones de pitch, y las mismas dependen del contexto de quién esté empleando la terminología.

En física acústica, la utilización del término pitch se refiere a la frecuencia fundamental percibida de un sonido, sin importar si dicho sonido se encuentra presente o no en la forma de onda[9]. La frecuencia fundamental de la voz se puede definir como la frecuencia a la cual las cuerdas vocales vibran durante un sonido de voz.

Según Deller, Hansen y Proakis (2000), cuando se refiere a pitch en una señal de voz cualquiera, se está hablando de la frecuencia fundamental de dicha señal[9].

Cheng (2009) define el pitch como la frecuencia fundamental del sonido que se produce en el ciclo de apertura y cierre de las cuerdas vocales de un individuo[8].

Talkin (1995) define la frecuencia fundamental de la voz como el inverso del menor período verdadero en un intervalo de señal de voz analizado, y establece que es una propiedad propia de las señales periódicas que tiende a tener una alta correlación con el pitch[10]. El mismo Talkin (1995) define el pitch como la percepción auditiva que tiene un oyente de un tono acústico.

Desde el punto de vista de ingeniería es conveniente, entonces, definir al pitch como la frecuencia fundamental de la señal acústica estudiada.

#### 2.1.3.2. Técnicas para el análisis de la frecuencia fundamental de la voz.

El análisis de señales de voz se divide en dos enfoques principales: análisis temporal y análisis espectral. Existe un gran número de técnicas para cada uno de estos enfoques. A continuación se describen brevemente algunas de las técnicas más utilizadas para el análisis y estimación de la frecuencia fundamental de la voz o pitch.

**Análisis en el dominio del tiempo**

**Autocorrelación** El objetivo del método de correlación es el de encontrar similitudes entre la señal de voz $u[n]$ y una versión desfasada de la misma; la autocorrelación se logra al multiplicar la señal estudiada por una versión retardada de la misma, lo cual se expresa matemáticamente como:

$$Y[n] = \sum_{k=1}^{M} u[k] \cdot u[k+n] \tag{2.6}$$

donde $k$ es el retardo.

La señal resultante tendrá una gráfica de amplitud en función del tiempo, y el período fundamental es identificado como el primer mínimo de la señal autocorrelacionada[11]. Al conocer el período fundamental $T_0$, se calcula la frecuencia fundamental $F_0$ de la señal como el inverso del período fundamental $T_0$:

$$F_0 = \frac{1}{T_0} \tag{2.7}$$

Está técnica es muy confiable para señales altamente periódicas. Sin embargo, presenta problemas de falsas detecciones (falsos positivos), muchas veces presenta errores cuando la señal es muy ruidosa y requiere una gran cantidad de cómputo, por lo que su implementación en aplicaciones en tiempo real es virtualmente nula[11].

**Tasa de cruce por cero (zero-crossing rate)** La tasa de cruce por cero ($zcr$) es la medida del número de veces que la amplitud de la señal cruza el cero de referencia en una ventana de tiempo dada. Es un algoritmo simple de computar y robusto frente a ruidos de alta energía, aunque es muy limitado debido a que no es muy confiable frente a señales complejas y polifónicas. La tasa de cruce por ceros viene dada por:

$$zcr = \frac{1}{T-1}\sum_{t=1}^{T-1} II\{s_t \cdot s_{t-1} < 0\} \tag{2.8}$$

donde $s$ es una señal de longitud $T$ y la función indicadora $II\{A\}$ es $1$ si el argumento $A$ es verdadero y $0$ si no lo es[11].

Como la tasa de cruce por ceros simplemente cuenta el número de veces que la señal de muestras de voz cambia dentro de la misma ventana, es posible determinar rudimentariamente la frecuencia de la señal por medio del número de veces que la misma cruzó por cero. Por ejemplo, para una sinusoide de $100$ Hz se espera tener $4$ cruces por cero dentro de una ventana de medición de $20$ $ms$.

**Función de magnitud de diferencias (MDF)** Debido a que la técnica de autocorrelación requiere una gran cantidad de cómputo, especialmente multiplicaciones, es muy común utilizar la técnica de función de magnitud de diferencias para el cálculo del pitch. Es una técnica simple que consiste en restar muestras consecutivas de la señal de voz y sumar estas diferencias a lo largo de la ventana de estudio, como ilustra la ecuación 2.9. Cuando esta expresión es computada para diferentes valores de retraso correspondientes a los diferentes períodos del pitch, es posible estimar el período del pitch. El retraso puede ser ajustado iterativamente para obtener el período del pitch más apropiado.

$$MDF[l, m] = \sum_{n=m-N+1}^{m} |s[n] - s[n-1]| \tag{2.9}$$

donde $N$ es el número de muestras y $l$ es el retraso[11].

### Análisis en el domino de la frecuencia

**Algoritmo de pitch** Introducido por Schroeder[12], consiste en medir la frecuencia de cada uno de los componentes armónicos más altos de la señal y calcular

el máximo común divisor de estas frecuencias. Para detectar la frecuencia fundamental de la señal es necesario medir el pico más alto del primer armónico, para el segundo y tercer armónico la frecuencia se debe dividir entre dos y tres, respectivamente, y así sucesivamente para el resto de los armónicos. Es un método que ofrece buenos resultados y además es inmune al ruido aditivo y al multiplicativo.

**CEPSTRUM** El *habla* está conformada por dos componentes: fuente de excitación y sistema de tracto vocal. Para lograr modelar ambos componentes de manera independiente y utilizarlos en aplicaciones de tratamiento de señales de audio, es necesario que ambas componentes sean separadas del *habla*. Es aquí donde entra el análisis *cepstral*, cuyo objetivo principal es el de separar el *habla* en su componente de fuente de excitación y su componente de sistema de tracto vocal.

Según el modelo de la fuente y el filtro, base de la teoría acústica de la producción del habla, los sonidos se producen al excitar el sistema de tracto vocal con secuencias de impulsos periódicos (*voiced sounds*) o con secuencias de ruido aleatorio (*unvoiced sounds*). El *habla* resultante puede considerarse como la convolución de la secuencia de excitación respectiva (ruido o impulsos) y las características del tracto vocal, por lo que su representación en el domino frecuencial sería lineal[13][14][15][16]. El primero en proponer la utilización del CEPSTRUM como técnica para la estimación de la frecuencia fundamental de la voz fue A. Michael Noll[1].

Cepstrum es una modificación de la palabra inglesa “spectrum” y consiste simplemente en el espectro de un espectro[17]. La señal de tiempo original es transformada utilizando un algoritmo de transformada rápida de Fourier (FFT) y el espectro resultante es llevado a potencia y luego a una escala logarítmica. Este espectro en escala logarítmica es transformado nuevamente utilizando el algoritmo de transformada inversa rápida de Fourier (IFFT) y su magnitud elevada al cuadrado, para obtener el cepstrum de potencia, el cual es una especie de representación en escala del tiempo donde la variable t ya no es tiempo, sino el término “quefrency”. La figura 2.7 muestra las operaciones básicas requeridas para el cálculo del cepstrum de una señal de voz cualquiera. La figura 2.8 muestra una gráfica del cepstrum de

una señal de voz llevada al espectro logarítmico de potencia y la quefrency correspondiente al pitch de la señal.

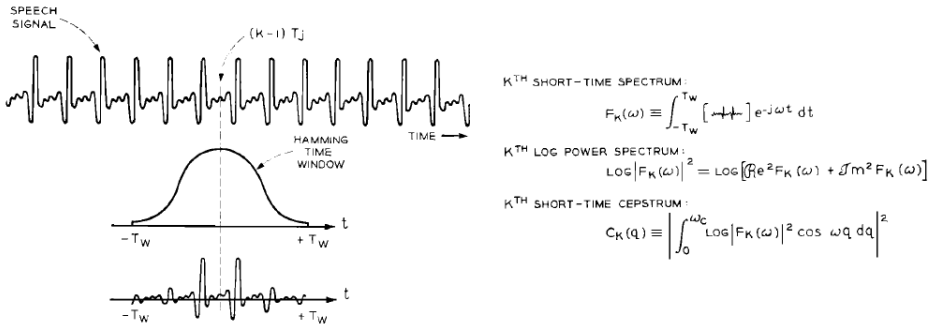


**Figura 2.7:** Operaciones básicas para el cálculo del cepstrum de una señal de voz. La ventana de Hamming de longitud $T_W$ se mueve en saltos de $T_j$ segundos. *Fuente: A. Michael Noll (1966) "Cepstrum pitch determination".*

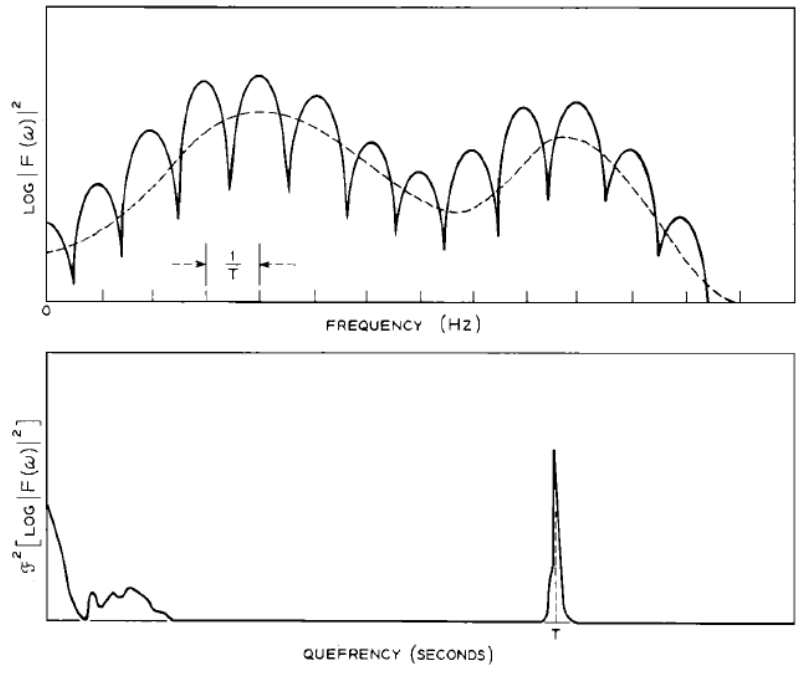


**Figura 2.8:** Espectro de potencia y CEPSTRUM de una señal. Arriba se observa el espectro de potencia logarítmico de una señal de voz, el cual muestra una periodicidad espectral resultante de la periodicidad del pitch de la señal original. Abajo se observa el cepstrum de la señal, el cual tiene un pico correspondiente a esta periodicidad espectral. *Fuente: A. Michael Noll (1966) "Cepstrum pitch determination".*

La frecuencia fundamental de la voz vendrá dada en función de la quefrency de mayor pico en su magnitud del cepstrum de potencia.

$$f_0 = \frac{1}{quefrency} \tag{2.10}$$

En términos generales, cuando la IFFT es tomada, la parte de la señal con poca variación resulta en una componente cepstral a baja quefrency (valores menores en el eje del "tiempo"), y la componente con rápidas variaciones resulta en una componente cepstral a alta quefrency (valores mayores en el eje del "tiempo"). Por lo tanto, la componente cepstral de baja quefrency representa una aproximación de la respuesta impulsiva del sistema vocal (también llamada envolvente espectral, asociada a la energía de la señal) y la componente de alta quefrency del cepstrum corresponde a la excitación de la señal (i.e. el pitch o frecuencia fundamental de la misma)[18][19].

### 2.1.4. Plataforma de desarrollo STM32F407 Discovery

En general, una plataforma o kit de desarrollo es aquella estructura de software y/o hardware sobre la cual algunas aplicaciones son capaces de ser ejecutadas[20]. La plataforma STM32F407 Discovery es una plataforma desarrollada por STMicroelectronics basada en núcleos microcontroladores Cortex-M4 de arquitectura ARM RISC de 32 bits.

#### 2.1.4.1. Descripción general

La STM32F407 Discovery es una placa de evaluación de bajo costo para el rango de microcontroladores de STM32F4 ARM Cortex-M4 basada en el microcontrolador STM32F407VGT6, que incluye una herramienta de depuración ST-LINK/V2, dos MEMS de ST (acelerómetro digital y micrófono digital), una DAC de audio con

controlador de altavoz integrado clase D, LEDs y botones pulsadores y un conector USB OTG micro-AB. La serie STM32F4 y la plataforma STM32F407 Discovery son producidas a partir de septiembre de 2011. Es el primer grupo basado en ARM Cortex-M4F[21].

#### 2.1.4.2. Características

La plataforma STM32F407 Discovery de STMicroelectronics ofrece las siguientes características[22]:

1. Núcleo ARM Cortex-M4F con una frecuencia de operación máxima de 168 MHz.
2. CPU de 210 DMIPS.
3. 192 kB de RAM estática, 64 kB de CCM (Core Coupled Memory).
4. Microcontrolador STM32F407VGT6 con 1 MB de memoria flash en encapsulado LQFP100.
5. Herramienta de depuración ST-Link/V2.
6. Alimentación vía USB o desde fuente externa de 3,3 V o 5 V.
7. Acelerómetro digital de 3 ejes ST MEMS LIS302DL.
8. Micrófono digital omnidireccional ST MEMS MP45DT02.
9. Coversor digital a analógico CS43L22 con controlador de altavoz integrado clase D.
10. Ocho LEDs: LD1 (rojo/verde) para comunicación USB; LD2 (red) para indicador de fuente de 3,3 V; cuatro LEDs de usuario, LD3 (naranja), LD4 (verde), LD5 (rojo) y LD6 (azul); dos LEDs para USB OTG, LD7 (verde) y LD8 (rojo).
11. Dos botones pulsadores (user y reset).
12. USB OTG FS con conector micro-AB.

13. 16 DMA, 6 USART, 3 SPI, 2 I2C, 3 ADC, 2 DAC, RTC, unidad CRC, 2 USB OTG, varios temporizadores, 2 watchdogs independientes.

14. 80 pines de propósito general.

#### 2.1.4.3. Pinout

La figura 2.9 muestra el pinout general de la plataforma STM32F407 Discovery y las funciones que pueden asociarse a cada uno de los pines de la misma.

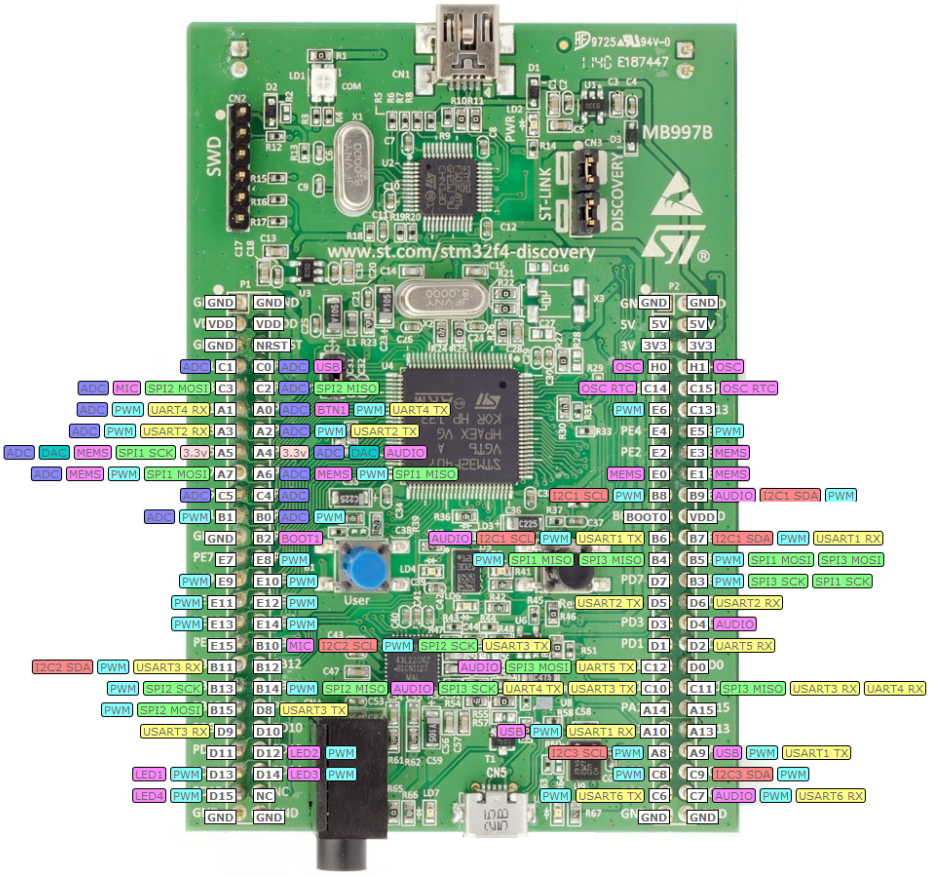


**Figura 2.9:** Pinout de la plataforma de desarrollo STM32F407 Discovery, con las funciones y componentes que pueden asociarse a cada pin. *Fuente: STMicroelectronics.*

Cabe destacar que los pines de la STM32F407 Discovery pueden trabajar bajo el estándar GPIO, esto es, pueden asignarse como interfaz de entrada o de salida, dependiendo de la aplicación que se desee.

#### 2.1.4.4. Arquitectura ARM

El término ARM se refiere a una arquitectura de diseño de CPU de 32 bits desarrollada por la empresa ARM Holdings. Es el acrónimo del inglés *Advanced RISC Machine.* Está basada en la arquitectura de diseño RISC[23], por lo que incorpora sus características principales. Entre ellas se encuentran:

- Instrucciones de tamaño fijo mostradas en un reducido número de formatos.
- Ejecución de instrucciones en paralelo.
- Una arquitectura de registros uniforme, donde el procesamiento de datos se realiza solo en el contenido de los registros y no directamente en el contenido de la memoria, por lo que se reduce el acceso a la misma.

El mejoramiento de la arquitectura RISC básica permite a los procesadores ARM lograr un buen balance de alto rendimiento, tamaño de código reducido y bajo consumo de potencia[23].

La plataforma STM32F407 Discovery utiliza un procesador ARM Cortex-M4, el cual presenta las siguientes características[24][21]:

- Núcleo procesador.
- Un controlador de interrupciones anidado integrado con el procesador para alcanzar procesos de interrupción de baja latencia.
- Múltiples interfaces bus de alto rendimiento.
- Herramienta de debugging que permite implementar breakpoints, watchpoints, rastreo, entre otros.

- Unidad de protección de memoria (MPU).
- Unidad de punto flotante (floating point unit o FPU), la cual se especializa en el cálculo de operaciones en punto flotante.
- Unidad microcontroladora (MCU) con facilidad de programación en lenguaje C y C++.
- Procesador digital de señales (Digital Signal Processor o DSP) con arquitectura Harvard, esto es, utiliza dispositivos de almacenamiento físicamente separados para datos e instrucciones.
- Otros.

La figura 2.10 muestra un diagrama esquemático de los procesadores ARM Cortex-M4. La figura 2.11 muestra un resumen de las tres principales características y funcionalidades de los procesadores ARM Cortex-M4; en la misma se pueden apreciar las principales características de la MCU, FPU y DSP.

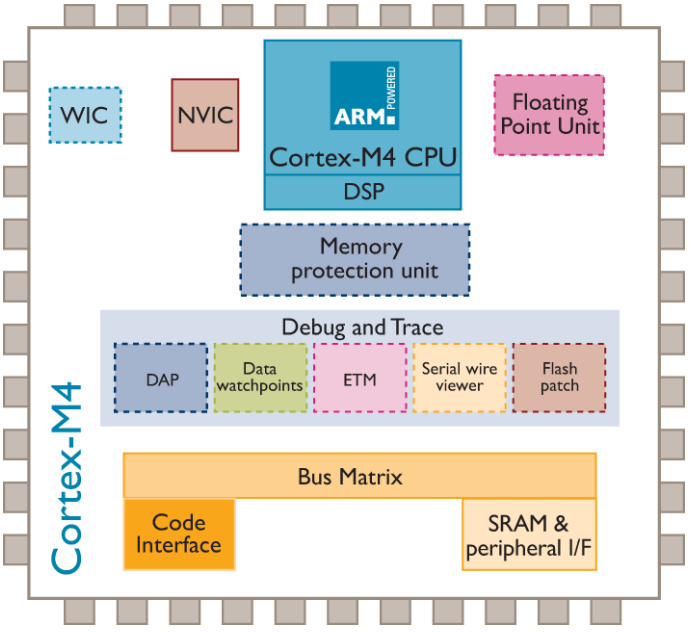


**Figura 2.10:** Diagrama esquemático de los procesadores ARM Cortex-M4, utilizados por la STM32F4 Discovery Board. *Fuente: STMicroelectronics.*

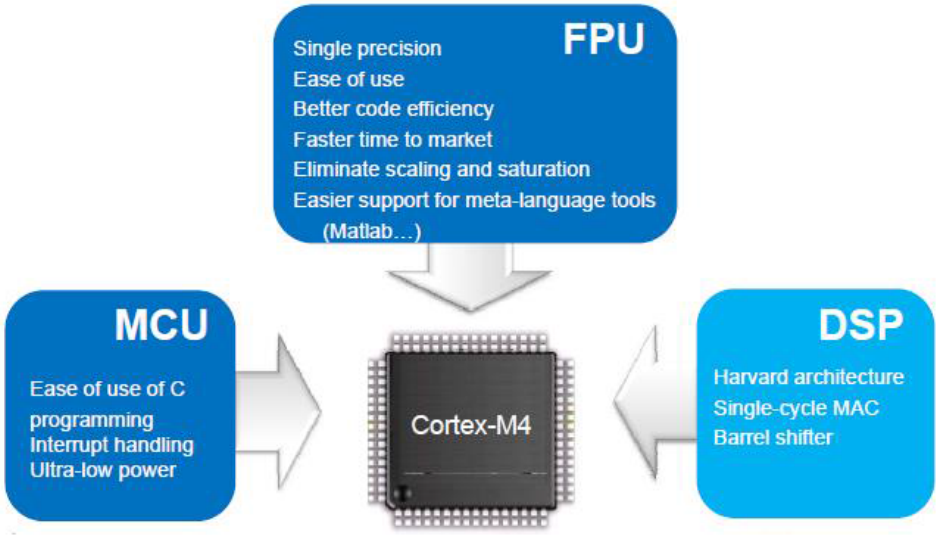


**Figura 2.11:** Procesador ARM Cortex-M4 y principales características. *Fuente: ST-Microelectronics.*

.

## 2.2. Definición de términos

- ARM: arquitectura basada en RISC desarrollada por ARM Holdings.
- I2C: es un bus de comunicaciones en serie. Su nombre viene de Inter-Integrated Circuit (Inter-Circuitos Integrados). Utilizado en la industria, principalmente para comunicar microcontroladores y sus periféricos integrados (embedded systems) y para comunicar entre sí circuitos integrados alojados en el mismo circuito impreso.
- NVIC: nested vectored interrupt controller (controlador de interrupciones anidado). Módulo de la plataforma STM32F407 Discovery que permite controlar interrupciones, administrar el consumo de potencia e implementar registros de control de sistema.

- RISC: reduced instruction set computing (Computador con Conjunto de Instrucciones Reducidas). Es una arquitectura computacional diseñada con el paradigma de que un set de instrucciones simplificadas permite un rendimiento superior al combinarse con microcontroladores capaces de ejecutar dichas instrucciones utilizando una menor cantidad de ciclos de microprocesador por instrucción.

- SPI: protocolo de datos en serie utilizado por microcontroladores para comunicarse con uno o más dispositivos periféricos rápidamente en distancias cortas, o bien para la comunicación entre dos microcontroladores.

- UART: transmisor/receptor asíncrono universal. Dispositivo que provee una interfaz que permite la comunicación entre la plataforma de desarrollo y otros dispositivos seriales.

- USART: transmisor/receptor asíncrono/síncrono universal. Similar al UART, pero permite, además, la utilización de un modo síncrono.

# Capítulo III

# Procedimientos de la investigación

Para la realización de los diferentes objetivos planteados en el presente proyecto, se dividió el mismo en cinco fases descritas a continuación.

## 3.1. Selección del algoritmo para la estimación de la frecuencia fundamental de la voz.

Basado en recomendaciones y conclusiones derivadas de estudios previos de otros autores e investigadores, se selecciona una técnica o algoritmo de estimación que pueda ser programado e implementado mediante la tarjeta STM32F407 Discovery. En este sentido, se favorece especialmente la técnica de estimación que no requiera de una gran capacidad de cálculo computacional, debido a que el kit de desarrollo utilizado tiene limitaciones en este aspecto. Para ello, se escoge una técnica basada en CEPSTRUM, la cual tiene un rendimiento bueno, debido a que el pico en el dominio *"cepstral"* tiene una alta correspondencia con el período del pitch del segmento de señal de voz que está siendo analizado[1], además de que sigue siendo uno de los indicadores de pitch más efectivos que existen[25].

Las técnicas basadas en Cepstrum pueden utilizar transformadas rápidas de Fourier (FFT) en sus cálculos, por lo que la carga computacional que conllevan es

menor que en algunas otras técnicas[26]. Por esto, la utilización del CEPSTRUM como técnica de estimación es apropiada en la implementación del detector de frecuencia fundamental de la voz utilizando la plataforma STM32F407 Discovery y las mismas se favorecen para la implementación con respecto al resto.

## 3.2. Diseño del programa en lenguaje C a partir del algoritmo seleccionado.

Una vez seleccionado el algoritmo basado en CEPSTRUM a utilizar, se diseña el diagrama de bloques y de flujos que sirve de referencia para la elaboración del código utilizado para la implementación del programa en la plataforma de desarrollo. A partir de la técnica de estimación seleccionada para la resolución del problema, y utilizando las herramientas de software, librerías y métodos provistos por el fabricante, se procede a diseñar el programa de estimación de frecuencia fundamental de la voz que posteriormente se implementa en la tarjeta DSP, aprovechando para ello la capacidad de la plataforma de desarrollo de utilizar el lenguaje de programación C y basándose en el diagrama de bloques diseñado.

Además del diseño del algoritmo en lenguaje C, es necesario incorporar las etapas de inicialización de todos los elementos que permiten la recopilación de la señal de datos que debe ser procesada, i.e., incluir los controladores de los conversores analógico-digital, entradas y salidas analógicas y digitales, entre otros, además de las variables utilizadas para el almacenamiento y procesamiento de la información recibida por la plataforma.

La mayoría de las librerías a utilizar fueron diseñadas o aprobadas para su uso por el mismo fabricante de la plataforma, por lo que facilitan la elaboración del código final en C ya que disminuyen el tamaño final del mismo, permiten una mejor organización de las etapas que lo componen, optimizan su funcionamiento y aprovecha las funcionalidades de la plataforma de desarrollo. De esta manera, la carga computacional final utilizada para la implementación del algoritmo de CEPSTRUM

en la plataforma de desarrollo permite la estimación de frecuencia fundamental en tiempo real.

Además, la utilización de librerías y métodos predefinidos permite al usuario modificar el código con mayor facilidad, si así lo requiere en un futuro, de modo que se deja la opción a largo plazo de optimizar aún más el programa diseñado.

## 3.3. Implementación del algoritmo en el hardware a utilizar.

Para el desarrollo del programa y escritura del código en el kit de desarrollo se utiliza un editor de entorno de desarrollo integrado (IDE), el cual es el IAR Embedded Workbench 6.30, compatible con la tarjeta STM32F407 Discovery. En este software se realiza la escritura del código e incorpora las librerías utilizadas en el diseño del programa final, y se compila en la plataforma de desarrollo mediante el puerto miniUSB de la misma.

Luego se conecta una unidad gráfica elemental (Display) para la visualización de los resultados, la cual se acopla a la tarjeta de desarrollo mediante un código en lenguaje C que permite comunicar ambos elementos, y se prueba la rapidez, precisión y confiabilidad de las estimaciones en dos etapas: una primera, donde se utilizan archivos de prueba estándar para el cálculo de la frecuencia fundamental de dichos archivos, y la segunda, que consiste en la utilización de señales de voz en tiempo real con parámetros desconocidos como señales de prueba.

Los errores de compilación que se presentan en esta etapa se resuelven a medida que los mismos sean detectados, aprovechando la facilidad que brinda el software de precisar la posición en el código en la cual se presentan los posibles errores.

## 3.4. Pruebas con archivos estandarizados. Correcciones necesarias.

Se utiliza una serie de archivos de prueba estandarizados, los cuales se grabaron en formato *.wav* y cuyos parámetros son previamente conocidos, para su procesamiento por parte de la tarjeta. Se utilizan archivos con audio de voz femenina y archivos con audio de voz masculina, a varios niveles de relación señal a ruido, de modo que se pueda determinar con mayor precisión la calidad de las estimaciones realizadas. Los archivos de audio de prueba se reproducen desde una PC y se transmiten desde la tarjeta de audio de la misma hasta el conversor analógico digital de la plataforma de desarrollo.

Se comparan los resultados obtenidos mediante la estimación con los parámetros ya conocidos de los archivos y se determina si fueron o no coherentes y confiables. De no suceder esto, se replantea el programa utilizado hasta lograr la mayor precisión posible; para lograr tal objetivo se modifica en el código el tamaño de la FFT e IFFT usadas, la tasa de muestreo del ADC, el tipo de ventana utilizada, entre otros posibles factores, y se compila nuevamente el programa hasta obtener resultados confiables y en concordancia con los parámetros de las señales estandarizadas.

## 3.5. Pruebas en tiempo real con señales de voz de frecuencia fundamental desconocida.

Se realizan pruebas en tiempo real con señales de voz de parámetros desconocidos, lo cual consiste en la lectura de una serie de vocales, palabras o escritos genéricos por parte de los sujetos de prueba en un ambiente genérico. Se realizan grabaciones de voces femeninas y de voces masculinas en una PC y se procede a enviar la señal hacia el detector, de modo que pueda hacer las estimaciones de frecuencia y se pueda verificar el correcto funcionamiento del detector de frecuencia fundamental de la voz implementado.

En la figura 3.1 se muestra el flujograma de las fases de investigación descritas anteriormente.

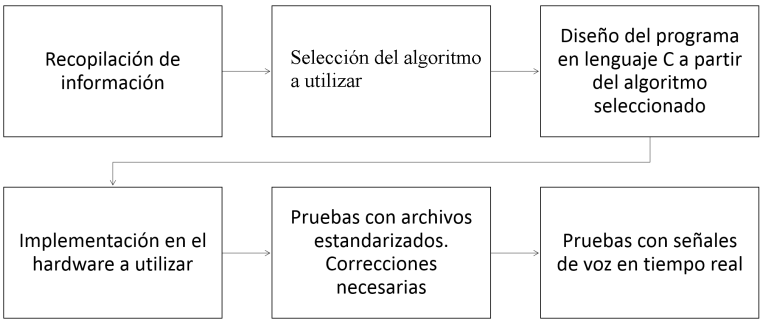


**Figura 3.1:** Flujograma de fases de investigación

# Capítulo IV

# Análisis, interpretación y presentación de los resultados

A continuación se presentan de forma detallada los resultados obtenidos a lo largo de la realización del presente proyecto en sus diferentes fases, así como la interpretación, análisis y procesamiento de los mismos con el fin de dar cumplimiento al objetivo general planteado.

## 4.1. Selección del algoritmo de estimación de la frecuencia fundamental de la voz

La estimación de la frecuencia fundamental de la voz se puede realizar a partir de una diversa cantidad de técnicas y algoritmos que se aplican en los dominios de estudio más utilizados en el ámbito del procesamiento digital de señales: el dominio temporal y el espectral. Cada una de ellas posee sus propias ventajas y desventajas, las cuales se debieron tomar en cuenta para determinar cuál algoritmo es más conveniente para la aplicación planteada. En este sentido, un detector de frecuencia fundamental de la voz en tiempo real requiere de una implementación capaz de realizar una estimación precisa utilizando la menor carga computacional posible, de modo que no exista retraso perceptible entre la aplicación de la señal

de entrada y el resultado dado por el detector. Técnicas como la autocorrelación (en el dominio temporal) ofrecen una gran precisión en los resultados obtenidos, especialmente cuando la señal tiene una alta periodicidad, pero requieren una gran carga computacional, por lo que su utilización es contraproducente para el presente proyecto[11]. Otras técnicas del dominio temporal, como la tasa de cruces por cero y la función de magnitud de diferencias, requieren una menor carga computacional que la autocorrelación, pero ofrecen resultados con una menor precisión debido a que son más propensas al ruido[11].

Aunque estas dos últimas técnicas constituyen alternativas viables para la implementación de un detector de frecuencia fundamental de la voz en tiempo real, se decidió utilizar un algoritmo sencillo basado en el cálculo del CEPSTRUM de la señal de entrada, debido a que es una técnica que ofrece uno de los estimadores de pitch más efectivos utilizados en la actualidad, ya que el pico resultante en el dominio *cepstral* tiene una alta correspondencia con la frecuencia fundamental de la señal estudiada[1][25]. Además de esto, la utilización del CEPSTRUM como algoritmo de estimación de la frecuencia fundamental de la voz implica el uso de transformadas de Fourier y transformadas inversas de Fourier, lo cual permitió aprovechar en un mayor grado las herramientas y librerías provistas por el fabricante y desarrolladores de la plataforma STM32F407 Discovery, pues las mismas incluyen algoritmos de transformadas rápidas de Fourier y transformadas inversas rápidas de Fourier optimizados. De este modo, se determinó que la utilización del CEPSTRUM genera un balance en cuanto a carga computacional, disponibilidad de herramientas y librerías de desarrollo, confiabilidad de los resultados y velocidad de procesamiento matemático en la plataforma de desarrollo utilizada, por lo que se escogió este algoritmo como fundamento para el programa de estimación de frecuencia fundamental de la voz en tiempo real implementado.

## 4.2. Diseño del programa en lenguaje C a partir del algoritmo seleccionado

### 4.2.1. Descripción del algoritmo seleccionado

Para diseñar el programa a implementar en la plataforma de desarrollo se requiere, en primera instancia, conocer el funcionamiento del algoritmo que se escogió como fundamento del programa final. La figura 4.1 muestra el flujograma del algoritmo de estimación de frecuencia fundamental a partir del CEPSTRUM que fue seleccionado.

En primer lugar, se aplica la transformada rápida de Fourier a la señal de voz utilizada como entrada, conformada por la convolución de las componentes de excitación [e(n)] y respuesta impulsiva del sistema vocal [h(n)], de modo que se obtiene el espectro en frecuencia de dicha señal. En el dominio espectral se tiene una combinación lineal de ambas componentes, por lo que se facilita el manejo de ambas. Luego de calculada la FFT se aplica el cuadrado del valor absoluto al espectro y se hace una representación logarítmica del mismo, de modo que el espectro de magnitud de la señal es ahora una combinación lineal de ambas componentes, i.e., se llevó la operación *producto* a *adición* en el dominio frecuencial. La utilización del cuadrado del valor absoluto permite amplificar las componentes espectrales con niveles de potencia apreciables, permitiendo separarlas aún más del piso de ruido del espectro, lo cual facilita su procesamiento por la transformada inversa de Fourier.

La separación de ambas componentes de la señal se realiza mediante la transformada inversa de Fourier de esta representación logarítmica; la IFFT de un espectro logarítmico genera un dominio análogo al dominio temporal, el cual se llama quefrency o dominio cepstral. Se aplica valor absoluto nuevamente y se selecciona el quefrency de mayor magnitud, para el cálculo de la frecuencia correspondiente mediante la ecuación 2.10. A partir de este algoritmo se realizó el diseño del programa final implementado.

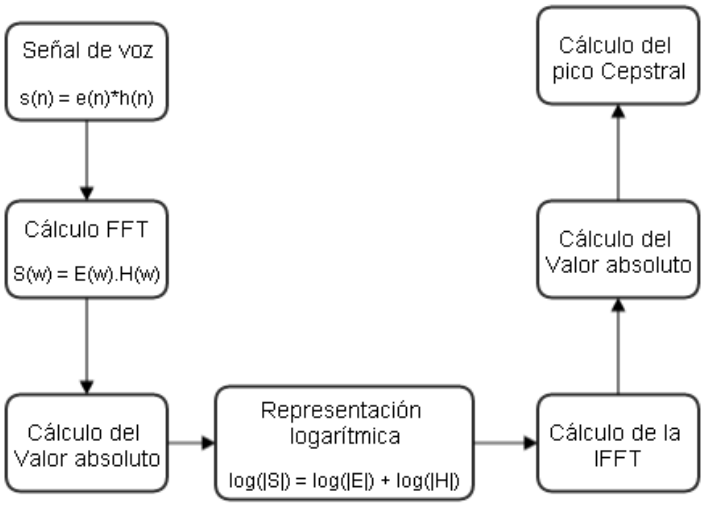


**Figura 4.1:** Diagrama de bloques del algoritmo de estimación de la frecuencia fundamental de una señal de voz utilizando CEPSTRUM.

### 4.2.2. Librerías y herramientas utilizadas

El lenguaje de programación C permite una programación estructurada, modular y recursiva, cuyo código está contenido en subrutinas (llamadas *funciones*). Permite, además, la utilización de operaciones aritméticas y lógicas, asignaciones de valores múltiples en una misma sentencia, utilización de *procedimientos* (funciones que no retornan valor), conversiones implícitas entre varios tipos de variables, uso eficiente de apuntadores, integración de lenguaje *assembler*, entre otros[27]. Es necesario conocer la sintaxis y funcionamiento de este lenguaje, así como de las librerías y herramientas provistas por los desarrolladores de la plataforma STM32F407 y terceros, para el diseño de un código eficiente que posteriormente fue implementado como programa final.

Entre las librerías y herramientas utilizadas se encuentran: *CMSIS DSP Library v. 1.4.1, STM32F4-Discovery support library for Digital Signal Processing, ARM Core Cortex M3/M4 Support Library, STM32F4xx Peripheral Libraries, STM32F4-Discovery Board Support Libraries*, además de ciertas modificaciones de las mismas y la creación de librerías y funciones propias diseñadas específicamente para este trabajo.

La tabla 4.1 muestra un resumen de algunas características y funcionalidades de las librerías disponibles y las propias. La librería *CMSIS DSP* contiene las funciones de procesamiento digital de señales y operaciones matemáticas necesarias para la implementación del algoritmo, tales como FFT, IFFT, valor absoluto y otros. La librería *STM32F4-Discovery support for Digital Signal Processing* contiene subrutinas de inicialización de los conversores analógico-digital, funciones de traslado y ordenamiento de los datos adquiridos hacia arreglos o buffers de trabajo utilizados para el posterior procesamiento de la señal, y otros. La librería *ARM Core Cortex M3/M4* contiene pre-procesadores y funciones complementarias de procesamiento de señales que permiten acoplar y utilizar en conjunto la librería *CMSIS DSP* con procesadores de núcleo M3 y M4, como es el caso de la plataforma de desarrollo utilizada. La librería *STM32F4xx Peripheral* contiene funciones y subrutinas asociadas a la inicialización, configuración y uso de los periféricos de la plataforma de desarrollo, como los controladores del ADC, DAC, micrófono, módulo SPI, módulo UART, módulo DMA, memoria flash, entre otros. La librería *STM32F4-Discovery Board Support* contiene subrutinas y funciones utilizadas para la inicialización y uso de los LEDs, botones de pulso, salidas de audio, entre otros, presentes en la plataforma de desarrollo.

Se creó una librería para la utilización de una pantalla LCD modelo Parallax Serial LCD 27977 como herramienta de visualización de los resultados de la estimación de la frecuencia fundamental. Esta librería se denominó *stm32f4xx_serlcd library*, y la misma contiene funciones y subrutinas que permiten enviar comandos desde la plataforma de desarrollo hasta la unidad gráfica elemental; entre estos se encuentran comandos de envío de sentencias y caracteres ASCII, movilización del cursor en la pantalla, reinicio de la pantalla (borrado de los caracteres mostrados en la misma), entre otros. También se crearon archivos complementarios que contienen funciones necesarias para la implementación del programa final, tales como funciones ventana (Hamming y Hann), filtros, relleno con ceros, entre otros.

En general, estas librerías trabajan con funciones que utilizan variables de tipo *integer* y *float*, por lo que, en conjunto con el módulo de unidad de punto flotante de la plataforma de desarrollo, permitió la creación de un código con mayor capacidad

de cómputo y eficiencia en comparación a uno equivalente utilizando funciones con punto fijo. La mayoría de las funciones provistas por estas librerías están optimizadas para trabajar con arreglos de longitudes de potencias de base 2, por lo que los arreglos utilizados en el código fueron creados con longitudes de este tipo.

Los anexos A, B, C y D muestran los contenidos de algunas de estas librerías, así como ciertos archivos utilizados.

### 4.2.3. Programa final diseñado en lenguaje C

El programa final fue diseñado en cinco fases generales: inicialización de variables y subrutinas de las librerías, muestreo y pre-procesamiento de la señal de voz, aplicación del algoritmo de CEPSTRUM, post-procesamiento de la señal de voz y visualización de los resultados. La figura 4.2 muestra el flujograma correspondiente.

En primer lugar se diseñó el código de inicialización de las variables globales y subrutinas de inicialización necesarias para el funcionamiento de los componentes de la plataforma de desarrollo (conversor analógico digital, transmisor/receptor asíncrono universal), así como de las funciones necesarias para el procesamiento de la señal de entrada.

Se diseñó una etapa de adquisición de datos (muestreo de la señal de entrada) por medio del ADC con una tasa de muestreo de 8 kHz, la cual almacena la señal en un bloque de datos inicial de 4000 muestras que será procesado posteriormente, aprovechando el conversor analógico digital propio de la plataforma de desarrollo y las funciones de inicialización del mismo provistas por las librerías nombradas en la sección anterior.

**Tabla 4.1:** Librerías utilizadas en el diseño del programa final en lenguaje C para la implementación de un detector de frecuencia fundamental de la voz en tiempo real.

| Librería | Descripción | Funciones | Comentarios |
|---|---|---|---|
| CMSIS DSP Library | Librería que contiene las funciones para el procesamiento digital de señales y otras operaciones matemáticas. | FFT, IFFT, valor absoluto, búsqueda del máximo, otros. | Provista por los desarrolladores. |
| STM32F4-Discovery Support Library for DSP | Librería encargada de la inicialización y configuración de las funciones relacionadas con la adquisición y ordenamiento de datos. | Inicialización del ADC; arreglo del buffer de trabajo; selección de la tasa de muestreo; otros. | Modificada para el presente trabajo. |
| ARM Core Cortex M3/M4 Support Library | Librería complementaria utilizada en conjunto con CMSIS DSP Library para su acoplamiento con procesadores Cortex M3 y M4. | Habilitación del FPU; acoplamiento de las funciones del CMSIS DSP Library; otros. | Provista por los desarrolladores. |
| STM32F4xx Peripheral Libraries | Librería encargada de inicializar y configurar los dispositivos periféricos de la plataforma de desarrollo STM32F407 Discovery. | Controladores del ADC, DAC, DMA, UART, memoria flash, otros. | Provista por los desarrolladores |
| STM32F4-Discovery Board Support Libraries | Librería encargada de inicializar y configurar los dispositivos de bajo nivel de la plataforma de desarrollo STM32F407 Discovery. | Controladores de LEDs, botones de pulso (push-buttons), salidas de audio, otros. | Provista por los desarrolladores. |
| STM32F4xx_serlcd Library | Librería encargada de comunicar y acoplar la plataforma de desarrollo STM32F407 Discovery con la pantalla Parallax Serial LCD 27977. | Envío de resultados, sentencias y caracteres ASCII a la pantalla; reinicio de la pantalla; otros. | Creada para el presente proyecto. |
| Archivos Complementarios | Archivos .c y .h con definiciones de funciones complementarias para la implementación del presente proyecto. | Ventanas de Hamming y Hann, filtros, zero-padding, otros. | Creados para el presente proyecto. |

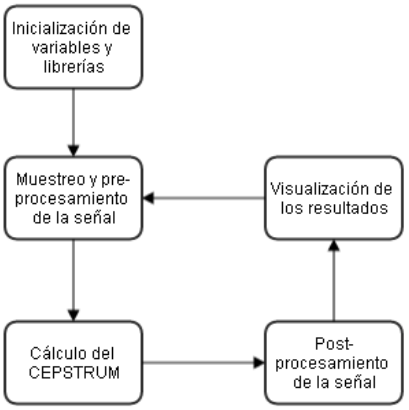


**Figura 4.2:** Flujograma simplificado de las fases generales del programa final diseñado para la estimación de la frecuencia fundamental de la voz en tiempo real.

El algoritmo de CEPSTRUM sin alterar sería suficiente para hacer una primera estimación de la frecuencia fundamental de la voz; sin embargo, fue necesario modificar y agregar ciertas etapas para obtener un resultado más preciso y escribir el código del programa aprovechando las herramientas matemáticas y de procesamiento digital de señales disponibles en forma de librerías en C y C++. Primeramente, se agregó una etapa de eliminación del offset (componente DC) de la señal de entrada muestreada, debido a que la misma no es de importancia en el cálculo de la frecuencia fundamental de la voz. Luego se realizó una rutina de relleno con ceros para rellenar los últimos índices del bloque de datos que contiene la señal muestreada, hasta que el tamaño del mismo coincidiera con el requerido por las funciones de FFT e IFFT de las librerías, en este caso 4096 elementos. Estas funciones están optimizadas para arreglos de datos cuya longitud es potencia de base 2, por lo que al rellenar con ceros la señal muestreada se logra mayor rapidez y eficiencia en el cómputo[28]. Luego se implementó una ventana de Hamming, definida como:

$$W(n) = 0.56 - 0.42 \cdot \cos(\frac{2 \cdot \pi \cdot n}{M - 1}) \tag{4.1}$$

siendo M la longitud de la ventana y n el número de muestras de la señal de entrada.

El uso de una función ventana fue necesario debido a que la FFT asume que la señal que procesa se repite infinitamente, lo cual no es cierto puesto que el muestreo de la señal de entrada genera una señal que no es necesariamente periódica en su totalidad; esto ocasiona que exista una *fuga* (leakage) en el espectro, lo cual consiste en la presencia de picos no deseados a lo largo del dominio frecuencial, i.e., parte de la energía de la señal se dispersa a lo largo de todo el espectro[29]. La utilización de la ventana de Hamming permite reducir esta fuga y el efecto de las discontinuidades en las señales no periódicas, como es el caso de la mayoría de aquellas obtenidas mediante conversores analógico-digitales y almacenadas en bloques de datos; el archivo que contiene la definición de esta función fue creado específicamente para el presente trabajo.

Una vez realizado el pre-procesamiento de la señal (muestreo, eliminación de offset, zero-padding y aplicación de la ventana de Hamming), se implementó el bloque de aplicación del algoritmo de CEPSTRUM. Este bloque se diseñó en concordancia con el algoritmo descrito en la sección 4.2.1, hasta la etapa del cálculo del valor absoluto del dominio cepstral. Las funciones FFT, IFFT, valor absoluto, cuadrado del valor absoluto y logaritmo decimal fueron provistas por la librería CMSIS DSP.

La fase de post-procesamiento consiste en la eliminación de la componente cepstral correspondiente al sistema de tracto vocal (respuesta impulsiva), de modo que sólo quede la componente de excitación, y en el ordenamiento de los índices del arreglo final. La eliminación del componente de la respuesta impulsiva se realizó mediante la implementación de un *lifter* (filtro en el dominio cepstral) pasa altas, i.e., las quefrency bajas fueron multiplicadas por cero, mientras que las componentes de mayor variación fueron dejadas intactas. Luego se aplicó una función de cálculo de máximo, provista por la librería CMSIS DSP, que almacenara el valor del pico de mayor magnitud y su posición en el arreglo. Cabe destacar que los resultados de las funciones de FFT e IFFT provistas por las librerías se presentan en forma de espejo (i.e. la FFT da una representación espectral tanto en el semieje de frecuencia positivo como en el negativo); se diseñó una rutina tal que, en caso

de que el máximo estuviera en los índices que correspondieran al semieje negativo, llevara el valor del índice a su equivalente en el semieje positivo del espejo, de modo que los resultados mostrados en la visualización fueran coherentes con la realidad. Posteriormente, se calculó la frecuencia fundamental de la voz a partir de la siguiente ecuación:

$$f_0 = \frac{fs}{i_{max}} \tag{4.2}$$

donde $f_0$ es la frecuencia fundamental de la voz, $fs$ es la tasa de muestreo utilizada por el ADC de la plataforma STM32F407 Discovery e $i_{max}$ es la posición o índice del pico de mayor magnitud calculado en el arreglo final de resultados.

Finalmente, se diseñó la fase de visualización de resultados a partir de una librería creada específicamente para el presente proyecto y compatible con la unidad gráfica elemental utilizada. Se diseñaron las funciones que permitieran almacenar y mostrar en la pantalla el valor de la frecuencia fundamental de la voz estimado.

Todo el procedimiento, desde el muestreo de la señal hasta la visualización de los resultados, se diseñó dentro de un ciclo *while*, de modo que se repitiera infinitamente o hasta que el usuario le diera un alto cuando se presionara el botón *reset*, para lograr de esta manera la estimación de la frecuencia fundamental de la señal de voz en tiempo real. La figura 4.3 muestra el flujograma del algoritmo del programa final diseñado y descrito anteriormente.

El anexo E muestra el código del programa principal utilizado para la estimación de la frecuencia fundamental de la voz en tiempo real, debidamente comentado.

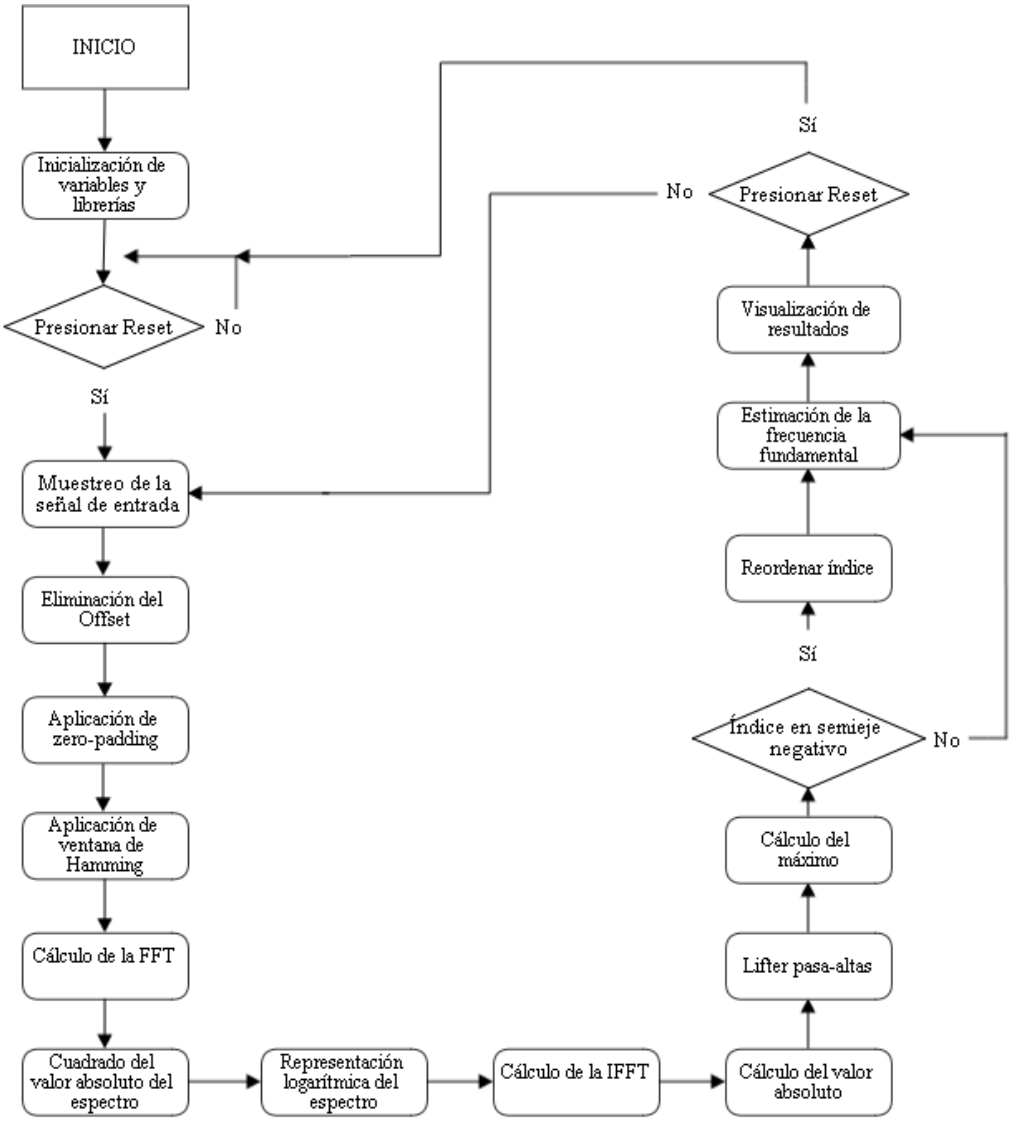


**Figura 4.3:** Flujograma del programa final diseñado para la estimación de la frecuencia fundamental de la voz en tiempo real.

## 4.3. Implementación del algoritmo en el hardware a utilizar

Una vez diseñado el programa final en C, se procede a implementar el mismo en la plataforma de desarrollo. Para ello, se emplea el software IAR Embedded Workbench 6.30 como herramienta para escribir el código en la memoria flash del

módulo por medio del puerto miniUSB del mismo. Este software permite seleccionar entre una amplia variedad de plataformas de desarrollo como objetivo (*target*) para escribir el programa; se seleccionó la plataforma de desarrollo STM32F407VG6 y se procedió a crear el proyecto. En primer lugar, se establecieron las rutas donde se encontraban los archivos a utilizar en el proyecto y se establecieron los símbolos asociados a las librerías utilizados por el preprocesador del software. Se creó el archivo *main.c* donde se escribió el código principal del programa; luego se añadieron cada uno de los archivos *.c* y *.h* correspondientes a las librerías utilizadas y se procedió a compilar el programa final.

La implementación física del detector de frecuencia fundamental se realizó sobre una sección de *15 x 10* cm de baquelita perforada. La misma está dividida en tres módulos:

1. **Adecuación de la señal de entrada** La señal de audio entrante es recibida por un grupo de cuatro amplificadores operacionales independientes entre sí (LM324N de Texas Instruments), configurados en tres etapas conectadas en cascada ordenadas de la siguiente manera:

   - *Amplificación de la señal de entrada*. Este primer operacional está configurado como amplificador inversor de ganancia *10 V/V*, definida por las resistencias R2 = *100 k*$\Omega$ y R1 = *10 k*$\Omega$.
   - *Filtrado de componentes de alta frecuencia*. Consiste filtro activo pasa-bajas de $4^o$ orden, respuesta Chebyshev con 2 dB de rizado, ganancia unitaria, topología Sallen-Key de frecuencia de corte *4* kHz. La primera etapa posee una frecuencia de polo de *1,88* kHz y un factor de calidad *Q* de *0,929*; la segunda etapa tiene una frecuencia de polo de *3,85* kHz y un factor de calidad de *4,594*. Se realizó un ordenamiento de etapas de menor a mayor *Q* para evitar la saturación, ya que de esta manera los anchos de banda son progresivamente más pequeños[30][31]. Esto ayuda a prevenir la pérdida de rango dinámico y de la exactitud del filtro producto de

un posible recortamiento de la señal. La utilización de un filtro pasabajas de frecuencia de corte 4 kHz permite reducir los efectos del aliasing, debido a que la tasa de muestro utilizada es de 8 kHz.

- *Elevación del nivel de offset*. El cuarto amplificador operacional se configura como seguidor de tensión y se utiliza como buffer para eliminar los efectos de carga. Antes del comienzo de esta etapa se agrega a la señal una tensión offset de *+3* V con el fin de acoplar la señal con el rango de entrada del conversor analógico a digital de la plataforma de desarrollo.

2. **Procesamiento de la señal**. Esta etapa es la implementación como tal del programa final en la plataforma de desarrollo STM32F407 Discovery de STMicroelectronics. Se recibe la señal amplificada y filtrada, con el offset de *+3* V, a través del pin PA3, el cual se configuró como entrada analógica del ADC. La misma es procesada utilizando el programa escrito en la memoria flash del kit de desarrollo y se envía el valor detectado a la unidad gráfica utilizada.

3. **Visualización de la frecuencia fundamental detectada**. Se utiliza un display *2x16* serial LCD (27977-RT de Parallax Inc.) operando a *9600* bps. Recibe los datos a través del pin PB6 (USART_TX) de la plataforma de desarrollo STM32F407 Discovery, también configurada para enviar información a *9600* bps en modo USART. La pantalla muestra en Hz el valor de la frecuencia fundamental detectada.

**Detalles adicionales**.

Para energizar el conjunto de amplificadores operacionales (±5V) se conecta el pin 4 (Vcc+) del LM324N al pin de 5 V de la plataforma de desarrollo. Para generar -5 V, se usa un convertidor de voltaje DC/DC (LTC1046CN8 de Linear Technology), que recibe en el pin 8 (V+) +5V del STM32F407 Discovery y suministra -5V a través del pin 5 (VOUT), necesarios en el pin 11 (Vcc-) del LM324N. La energización de la plataforma de desarrollo se realiza mediante el suministro de 5 V a la misma por medio del puerto miniUSB integrado.

La figura 4.4 muestra la implementación final del detector de frecuencia fundamental en operación. En la figura 4.5 se tiene un esquema del circuito de amplificación, filtrado y elevación de offset de la señal de entrada descrito anteriormente.

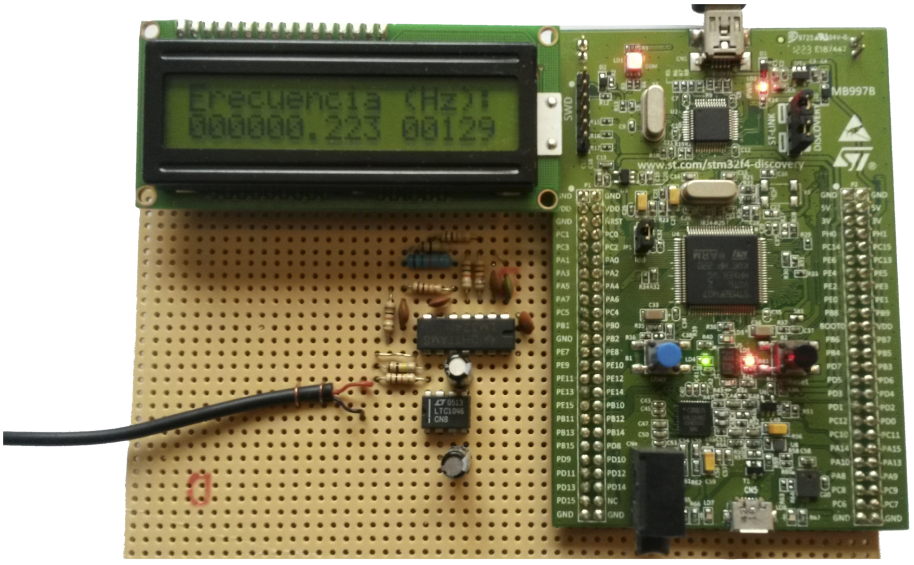


**Figura 4.4:** Implementación final del detector de frecuencia fundamental.

El tiempo de cómputo de cada ciclo de estimación de frecuencia fundamental en la implementación fue de 0,201621 segundos, suficiente para la estimación en tiempo real.

## 4.4. Pruebas con archivos estandarizados. Correcciones necesarias.

Una vez implementado el programa y elaborado el circuito que permite elevar el nivel de la señal de entrada hasta valores utilizables por el ADC de la plataforma de desarrollo, se procedió a realizar las pruebas de estimación de frecuencia

fundamental de la voz utilizando archivos de prueba con parámetros conocidos.

Se utilizaron archivos de audio en formato *.wav* PCM de 16 bits, con frecuencia de muestreo *22050* Hz, tanto con voces femeninas y como con voces masculinas. Cada uno de estos archivos posee valores de frecuencia fundamental y de relación de señal a ruido conocidos, valores que adicionalmente fueron estimados a través de una herramienta de software libre y de una herramienta de software de licencia comercial, de modo que pudieron utilizarse para determinar la confiabilidad de los resultados dados por el detector fundamental de frecuencia fundamental implementado, además de dar una idea del nivel de ruido que puede estar presente en la señal sin que afecte considerablemente el funcionamiento del mismo.

Se utilizaron dos señales con vocales de voz femenina y dos señales con palabras de voz masculina, cada una de las cuales se envió desde la PC hasta el ADC de la plataforma de desarrollo. Se agregó ruido gaussiano a diferentes niveles a cada una de estas señales de voz, para generar así archivos con diferentes niveles de relación señal a ruido. Se generaron 20 archivos con ruido para cada nivel de SNR de cada señal de voz, de modo que se tuviera ruido aleatorio en un mismo nivel de SNR para realizar un análisis estadístico de la frecuencia estimada. Las señales de voz originales fueron grabadas en un ambiente libre de ruido, mientras que los demás archivos fueron generados a partir del original mediante un software especializado de licencia comercial, que agregaba ruido gaussiano en cada archivo hasta lograr valores de SNR de 10 dB, 6 dB, 4dB, 3dB, 2dB, 1 dB y 0 dB.

La tabla 4.2 muestra las estimaciones de frecuencia fundamental obtenidas al enviar el archivo de voz masculina *huts.wav* y los archivos con ruido generados. En primer lugar, se envió el archivo original sin presencia de ruido, el cual posee una frecuencia fundamental de voz conocida de 135 Hz. Se observa que para este archivo libre de ruido el detector de frecuencia fundamental de la voz implementado estimó una media de 134 Hz como frecuencia fundamental, lo cual implica una desviación estándar de 1,0259 Hz respecto a la misma. La moda detectada fue de 135 Hz, correspondiente al valor real de la frecuencia fundamental de la voz. Se observa que a medida que se aumentó el nivel de ruido presente en la señal se

tiene una menor precisión de las estimaciones hechas, así como mayores márgenes de error.

**Tabla 4.2:** Pruebas de estimación de frecuencia fundamental de voz masculina a diferentes niveles de relación señal a ruido. Archivo "huts.wav".

| Archivo | Frecuencia Real | SNR | Media Detectada | Moda | Desviación Estándar |
|---|---|---|---|---|---|
| huts.wav | 135 Hz | Sin Ruido | 134 Hz | 135 Hz | 1,0259 Hz |
| huts.wav | 135 Hz | 10 dB | 133,9 Hz | 133 Hz | 1,3726 Hz |
| huts.wav | 135 Hz | 6 dB | 133,9 Hz | 133 Hz | 1,5183 Hz |
| huts.wav | 135 Hz | 4 dB | 134,6 Hz | 135 Hz | 1,3917 Hz |
| huts.wav | 135 Hz | 3 dB | 133,8 Hz | 133 Hz | 2,1908 Hz |
| huts.wav | 135 Hz | 2 dB | 137,9 Hz | 133 Hz | 39,2802 Hz |
| huts.wav | 135 Hz | 1 dB | 168,85 Hz | 135 Hz | 38,8428 Hz |
| huts.wav | 135 Hz | 0 dB | 170,8 Hz | 135 Hz | 70,6098 Hz |

Como muestra la tabla 4.3, al enviarse el archivo de voz masculina *beds.wav*, de frecuencia fundamental conocida 130 Hz y sin presencia de ruido, el detector de frecuencia implementado estimó una frecuencia fundamental de la voz de media 129,5 Hz con una desviación estándar correspondiente a la misma de 0,8885 Hz y una moda de 129 Hz. Se observa que a 2 dB de SNR la desviación estándar fue significativa, a pesar de que la media de las estimaciones es muy cercana al valor real de la frecuencia fundamental. Al igual que para el caso anterior, con 3 dB o más de SNR se tiene estimaciones cercanas al valor real con desviaciones estándar pequeñas.

El mismo procedimiento que se hizo para los archivos de voces masculinas se repitió para los archivos de voces femeninas. En primer lugar, se envió desde la PC al detector de frecuencia fundamental implementado el archivo sin ruido *a.wav*, de frecuencia fundamental conocida 193 Hz. En la tabla 4.4 se muestran los resultados obtenidos; para el archivo original con voz femenina sin ruido, el detector implementado estimó una media de frecuencia fundamental de la voz de 194 Hz, con una desviación estándar de 2,0519 Hz con respecto a la misma y una moda de 195 Hz. A partir de esta señal de voz femenina se observa que las estimaciones realizadas por el detector de frecuencia empiezan a ser menos precisas con respecto a los archivos de voz masculina para un mismo nivel de SNR.

**Tabla 4.3:** Pruebas de estimación de frecuencia fundamental de voz masculina a diferentes niveles de relación señal a ruido. Archivo "beds.wav".

| Archivo | Frecuencia Real | SNR | Media Detectada | Moda | Desviación Estándar |
|---|---|---|---|---|---|
| beds.wav | 130 Hz | Sin Ruido | 129,5 Hz | 129 Hz | 0,8885 Hz |
| beds.wav | 130 Hz | 10 dB | 129,1 Hz | 129 Hz | 0,4472 Hz |
| beds.wav | 130 Hz | 6 dB | 129,15 Hz | 129 Hz | 2,6808 Hz |
| beds.wav | 130 Hz | 4 dB | 130,25 Hz | 129 Hz | 3,8234 Hz |
| beds.wav | 130 Hz | 3 dB | 129,65 Hz | 129 Hz | 2,7390 Hz |
| beds.wav | 130 Hz | 2 dB | 131,15 Hz | 129 Hz | 20,7466 Hz |
| beds.wav | 130 Hz | 1 dB | 128 Hz | 129 Hz | 35,2972 Hz |
| beds.wav | 130 Hz | 0 dB | 200,4 Hz | 129 Hz | 106,4110 Hz |

**Tabla 4.4:** Pruebas de estimación de frecuencia fundamental de voz femenina a diferentes niveles de relación señal a ruido. Archivo "a.wav".

| Archivo | Frecuencia Real | SNR | Media Detectada | Moda | Desviación Estándar |
|---|---|---|---|---|---|
| a.wav | 193 Hz | Sin Ruido | 194 Hz | 195 Hz | 2,0519 Hz |
| a.wav | 193 Hz | 10 dB | 194,5 Hz | 195 Hz | 1,5389 Hz |
| a.wav | 193 Hz | 6 dB | 194,5 Hz | 195 Hz | 2,2360 Hz |
| a.wav | 193 Hz | 4 dB | 193,55 Hz | 195 Hz | 3,4713 Hz |
| a.wav | 193 Hz | 3 dB | 193,1 Hz | 195 Hz | 4,5986 Hz |
| a.wav | 193 Hz | 2 dB | 178,55 Hz | 195 Hz | 32,3703 Hz |
| a.wav | 193 Hz | 1 dB | 178,9 Hz | 195 Hz | 35,8107 Hz |
| a.wav | 193 Hz | 0 dB | 187,35 Hz | 195 Hz | 45,2051 Hz |

La tabla 4.5 muestra los resultados de las estimaciones para el archivo de voz femenina *i.wav*, el cual tiene una frecuencia fundamental de la voz conocida de 225 Hz. Se envió el archivo con la señal de voz original sin ruido, dando como resultado una estimación media de 224,8 Hz para una desviación estándar correspondiente de 5,8991 Hz y una moda de 228 Hz.

De lo anterior se determina que el detector de frecuencia fundamental de voz implementado trabaja mejor con señales de voz masculina de SNR mayor a 2 dB, y mayores a 3 dB para señales de voz femenina. Para un mismo valor de SNR, las

**Tabla 4.5:** Pruebas de estimación de frecuencia fundamental de voz femenina a diferentes niveles de relación señal a ruido. Archivo "i.wav".

| Archivo | Frecuencia Real | SNR | Media Detectada | Moda | Desviación Estándar |
|---|---|---|---|---|---|
| i.wav | 225 Hz | Sin Ruido | 224,8 Hz | 228 Hz | 5,8991 Hz |
| i.wav | 225 Hz | 10 dB | 226,15 Hz | 222 Hz | 6,8692 Hz |
| i.wav | 225 Hz | 6 dB | 226,15 Hz | 228 Hz | 7,6521 Hz |
| i.wav | 225 Hz | 4 dB | 224,95 Hz | 222 Hz | 11,1613 Hz |
| i.wav | 225 Hz | 3 dB | 233,05 Hz | 235 Hz | 18,2770 Hz |
| i.wav | 225 Hz | 2 dB | 229,85 Hz | 228 Hz | 20,8611 Hz |
| i.wav | 225 Hz | 1 dB | 221,25 Hz | 205 Hz | 12,5985 Hz |
| i.wav | 225 Hz | 0 dB | 231,25 Hz | 242 Hz | 23,2806 Hz |

señales de voz femeninas presentaron una mayor desviación estándar que las masculinas, lo cual implica un mayor grado de error en las mismas. En general, las estimaciones realizadas a señales de voces masculinas resultaron más precisas que aquellas realizadas a señales de voces femeninas; esto se debe al resultado inherente de implementar un programa basado en algoritmo de CEPSTRUM con una FFT e IFFT de 4096 puntos. La estimación de la frecuencia fundamental de la voz es inversamente proporcional a la quefrency donde se encuentra el pico de mayor magnitud, lo cual implica que existe mayor precisión en los resultados para las frecuencias menores (mayor quefrency), i.e., los pasos de frecuencia resultantes entre dos *bins* de quefrency consecutivos son menores a medida que los mismos correspondan a índices más altos del arreglo. Es necesario, entonces, utilizar una tasa de muestreo mayor y funciones FFT e IFFT de mayor tamaño, capaces de procesar los bloques de datos provenientes del muestreo, para lograr una mayor precisión en las estimaciones para una misma señal de información, como muestra la ecuación 4.2.

El anexo F muestra los pasos de frecuencia entre índices de IFFT que se pueden estimar para diferentes tasas de muestreo.

## 4.5. Pruebas en tiempo real con señales de voz de frecuencia fundamental desconocida

Una vez realizadas las pruebas de estimación con archivos de voz con parámetros conocidos se procedió a realizar las pruebas de estimación de frecuencia fundamental de la voz en tiempo real de señales de voz con parámetros desconocidos, utilizando para ello dos sujetos de prueba: uno femenino y uno masculino. El sujeto de voz femenina corresponde a una mujer caucásica de 27 años de edad, y el sujeto de voz masculina corresponde a un hombre mestizo de 23 años de edad.

Cada uno de los sujetos de prueba realizó la lectura de las vocales en español para determinar la frecuencia fundamental de cada uno; se utilizó un software de licencia comercial para estimar la frecuencia fundamental de los sujetos y comparar con la estimada por el detector implementado. En primer lugar se realizaron las pruebas para el sujeto de voz femenina; la tabla 4.6 muestra los valores de frecuencia fundamental estimados por el detector para cada una de las vocales habladas y las estimaciones hechas por el software de licencia comercial. Se observa que los valores estimados entran dentro del rango común de frecuencia fundamental de la voz para una mujer, de 120 a 500 Hz.

**Tabla 4.6:** Pruebas de estimación de frecuencia fundamental en tiempo real de sujeto de voz femenina en un ambiente genérico.

| Elemento | Frecuencia Estimada | Frecuencia Detectada por Software Comercial |
|---|---|---|
| Vocal "a" | 190 Hz | 187 Hz |
| Vocal "e" | 205 Hz | 207 Hz |
| Vocal "i" | 210 Hz | 210 Hz |
| Vocal "o" | 235 Hz | 228 Hz |
| Vocal "u" | 258 Hz | 259 Hz |

Seguidamente se estimaron los valores de frecuencia fundamental para las vocales habladas por el sujeto de voz masculina. La tabla 4.7 muestra los resultados obtenidos. Como sucedió con el sujeto de prueba de voz femenina, los valores de

frecuencia fundamental detectados están dentro del rango de valores típicos, en este caso entre 50 y 250 Hz.

**Tabla 4.7:** Pruebas de estimación de frecuencia fundamental en tiempo real de sujeto de voz masculina en un ambiente genérico.

| Elemento | Frecuencia Estimada | Frecuencia Detectada por Software Comercial |
|---|---|---|
| Vocal "a" | 123 Hz | 122 Hz |
| Vocal "e" | 108 Hz | 106 Hz |
| Vocal "i" | 114 Hz | 114 Hz |
| Vocal "o" | 123 Hz | 124 Hz |
| Vocal "u" | 129 Hz | 129 Hz |

De lo anterior se puede establecer que el detector de frecuencia fundamental de la voz fue capaz de estimar valores para sujetos masculinos y femeninos en un ambiente genérico; además, el mismo estima valores de frecuencia fundamental muy cercanos a los estimados por el software de licencia comercial utilizado, especialmente para señales correspondientes a voces femeninas.

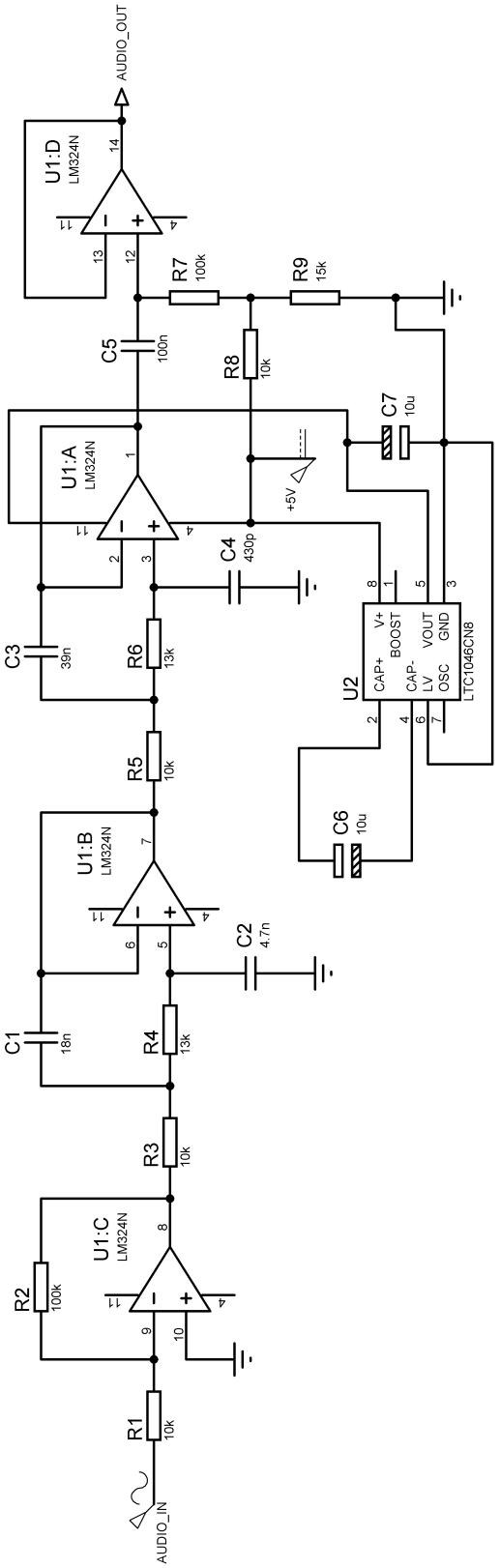


**Figura 4.5:** Esquema circuital del circuito diseñado para la amplificación, filtrado y elevación de offset de la señal de audio de entrada.

# Capítulo V

# Conclusiones y recomendaciones

## 5.1. Conclusiones

La utilización de la plataforma de desarrollo STM32F407 de STMicroelectronics facilitó la integración de herramientas y librerías provistas por los fabricantes de la misma, además de librerías creadas para el presente trabajo. Esto, unido a la facilidad de ordenamiento estructural propia del lenguaje de programación C, permitió el diseño de un código final estructurado, conciso y fácil de entender.

El uso de una tasa de muestreo de 8 kHz para la obtención de la señal de entrada, y utilización de FFT e IFFT de 4096 puntos en la implementación del programa final, permite obtener un compromiso aceptable entre cantidad de información y precisión de cálculos utilizando las librerías disponibles.

Las estimaciones de la frecuencia fundamental de voces masculinas resultaron ser más precisas que las de voces femeninas y con una menor desviación estándar para un mismo nivel de SNR, debido a la característica propia de la voz masculina de ser más grave que la femenina. La técnica de CEPSTRUM ocasiona de forma inherente que exista mayor precisión en la estimación de frecuencias menores.

Se determinó que para voces masculinas fue posible estimar la frecuencia fundamental de voz de manera confiable con niveles de relación señal a ruido superiores a 2 dB, mientras que para voces femeninas fue necesario tener una SNR mayor a 3 dB.

El detector de frecuencia fundamental de la voz implementado en la plataforma STM32F407 fue capaz de estimar valores de frecuencia fundamental de voz en tiempo real para señales de voz grabadas en un ambiente genérico, tanto para hablantes masculinos como femeninos, los cuales son comparables a los estimados por un software de licencia comercial utilizado.

La precisión de las estimaciones del programa final se vio limitada por la falta de FFT e IFFT en las librerías utilizadas capaces de procesar arreglos de datos superiores a 4096 elementos.

## 5.2. Recomendaciones

Incorporar otras librerías, si existen, que permitan optimizar aún más el código final utilizado, disminuyendo así la carga computacional de la plataforma de desarrollo.

Realizar implementaciones utilizando FFT e IFFT capaces de procesar arreglos de 8192 o más elementos, para poder utilizar tasas de muestreo mayores sin inconvenientes y mejorar la precisión de las estimaciones realizadas por el detector.

Promover el estudio de la documentación y contenido bibliográfico, optimizar el código final utilizado en el presente y proseguir con esta línea de investigación para aumentar el conocimiento e información que permita mejorar la implementación de módulos de estimación en tiempo real de la frecuencia fundamental de la voz.

# Referencias Bibliográficas


[1] A. Michael Noll. Cepstrum pitch determination. *The Journal of the Acoustic Society of America*, pages 293–309, Agosto 1966.

[2] John D. Markel. The sift algorithm for fundamental frequency estimation. *IEEE Transactions on Audio and Electroacoustics*, AU-20(5):367–377, Diciembre 1972.

[3] José A. Díaz, Christine Sapienza, Howard B. Rothman, and Yaser Natour. Algoritmo robusto para la detección de la frecuencia fundamental de la voz basado en el espectrograma. *Revista Ingeniería UC*, 10(3):7–16, December 2003.

[4] Sergio Roa, Maren Bennewitz, and Sven Behnke. Fundamental frequency estimation based on pitch-scaled harmonic filtering. In *International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2007.

[5] Eman A. Reda Al-Langawi. Automatic tuning of an ssb receiver. Communications Engineering Project Report. University of Hertfordshire, Abril 2004.

[6] ARM Holdings. Company profile, 2013. URL http://www.arm.com/about/company-profile/.

[7] Leon Couch. *Sistemas de Comunicación Digitales y Analógicos*. pp 312–316. Ed. Pearson, 7 edition, 2008.

[8] Qi Cheng. *Introductory Speech Processing*. pp 1-5. Self Published, 2009.

[9] John Deller, John Hansen, and John Proakis. *Discrete-Time Processing of Speech Signals*. IEEE Press. Ed. Wiley-Interscience, 2000.

[10] David Talkin. A robust algorithm for pitch tracking (rapt). In W.B. Kleijn y K.K. Paliwal, editor, *Speech Coding and Synthesis*, chapter 14, page 497. Elsevier Science B.V., 1995.

[11] Priyabrata Sinha. *Speech Processing in Embedded Systems*. Ed. Springer, 2010.

[12] Mannfred R. Schroeder. Period histogram and product spectrum: New methods for fundamental-frequency measurement. *The Journal of the Acoustic Society of America*, 43(4):829, Enero 1968.

[13] Peter D. Welch. The use of fft for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, AU-15(2):70–73, Junio 1967.

[14] David J. Defatta, Joseph G. Lucas, and Willam S. Hodgkiss. *Digital Signal Processing: A System Design Approach*. Ed. Wiley, 1988.

[15] Fredric J. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):172–204, Enero 1978.

[16] Alan V. Oppenheim and Ronald Schafer. From frequency to quefrency: A history of the cepstrum. *IEEE Signal Processing Magazine*, pages 95–106, Septiembre2004 2004.

[17] L.R. Rabiner and R.W. Schafer. *Theory and Application of Digital Speech Processing*. pp 425-426. Preliminar Edition, 2009.

[18] Wuhan University School of Electronic Information. Speech signal processing. chapter 3: Speech analysis.

[19] Donald G. Childers, David P. Skinner, and Robert C. Kemerait. The cepstrum guide: A guide to processing. *Proceedings of IEEE*, 65(10):1428–1443, Octubre 1977.

[20] Rodrigo Miranda. Sistemas de gobierno para diodos láseres con tecnología arduino y enfriamiento con peltier. Tesis de Grado. Escuela de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de Costa Rica, 2012.

[21] Natalia Ulloa. Control por hardware de sistemas de gobierno para diodos láser con stm32f4 y celdas peltier. Tesis de Grado. Escuela de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de Costa Rica, 2013.

[22] *STM32F4 Discovery Data Brief*. STMicroelectronics, 3 edition, 2013.

[23] ARM Holdings. Arm processor architecture, 2013. URL http://www.arm.com/products/processors/instruction-set-architectures/index.php.

[24] *ARM Cortex-M4 Processor. Technical Reference Manual*. ARM HOldings.

[25] L.R. Rabiner and R.W. Schafer. *Introduction to Digital Speech Processing*. Fundations and Trends in Signal Processing. Ed. NOW, 2007.

[26] L.R. Rabiner and R.W. Schafer. *Theory and Application of Digital Speech Processing*. pp 458. Preliminar Edition, 2009.

[27] C language features. URL http://www.c4learn.com/c-programming/c-features/.

[28] Julius Smith III. *Mathematics of the discrete Fourier transform (DFT) with audio applications*. Online Book, 2 edition. URL http://www.dsprelated.com/dspbooks/mdft//.

[29] Application Note AN014. Understanding fft windows. Technical report, LDS Dactron, 2003. URL http://www.physik.uni-wuerzburg.de/~praktiku/Anleitung/Fremde/AN014.pdf.

[30] Jim Karki. Application Report SLOA049B. Active low-pass filter design. Technical report, Texas Instruments, 2002.

[31] Manos Chaniotakis and David Cory. Introduction to electronics, signals, and measurement. MIT Open Courses. Spring 2006.

# Anexo A

# CMSIS DSP Library Functions Reference

# CMSIS DSP Library Functions Reference

List of all available modules:

## Anexo B

# STM32F4-Discovery Support Library for DSP Reference Guide

# STM32F4-Discovery support library for Digital Signal Processing

This library defines an interface that allows a user to stream input sampled data in real time from one or two ADCs or the Microphone, process the data, and stream the resulting output samples to one or two output DACs.

The user interface is as follows:

1. Required include:

```
#include "ece486.h"
```

2. Optionally, the user calls

```
setblocksize(nsamp);
```

This call is used to change the data block size that will be delivered to the user in later function calls. (If **setblocksize()** is not called, a default value of **DEFAULT_BLOCKSIZE** is used.) Using a larger block size may result in more efficient code, while using a smaller block size will reduce the latency of the system.

3. The user calls

```
initialize(fs, input_mode, output_mode);
```

where:

- *fs* determines the sampling frequency. *fs* should be one of: **FS_500K**, **FS_400K**, **FS_200K**, **FS_100K**, **FS_96K**, **FS_50K**, **FS_48K**, **FS_24K**,**FS_12K**, **FS_10K**, **FS_8K**, **FS_6K**, **FS_5K**
- *input_mode* should be **MONO_IN** or **STEREO_IN** or **MONO_MIC_IN**
- *output_mode* should be **MONO_OUT** or **STEREO_OUT**

The same sample rate is used for the input (ADCs or MIC) and output (DAC(s)) data streams. The **initialize()** call enables clocks and configures the appropriate Microphone, ADC(s), DAC(s), Timers, buttons, DMAs, and interrupts. The function pauses at the beginning of the initialization (with orange and blue LEDs flashing) to wait for a user push-button indicating to continue (allowing the device to be re-programmed without errors).

For *input_mode* = **MONO_IN** or **STEREO_IN**, Analog input waveforms are sampled using ADC1 (MONO) or ADC1 & ADC2 (STEREO_IN). ADC1 is accessed as an input on PA3 (MONO_IN or STEREO_IN). ADC2 is accessed on PA2 (STEREO_IN only).

For *input_mode* = **MONO_MIC_IN**, the one-bit on-board microphone is initialized. In this case the I2S clock controls the sampling frequency, and the timers and DMAs are left unused. PA3 and PA2 are left uninitialized.

Analog output is generated using DAC2 (MONO_OUT) or DAC2 & DAC1 (STEREO_OUT). DAC2 is accessed as an output on PA5 (MONO_OUT or STEREO_OUT). DAC1 is accessed on PA4 (STEREO_OUT only).

LED GPIO pins (PD12, PD13, PD14, PD15) are enabled, and digital outputs are configured for PA1, PC4, and PC5.

4. The user calls

```
nsamp = getblocksize();
```

*nsamp* gives the data block size which will be delivered to the user in later function calls. Input samples (either from the microphone or from the ADC(s) via the DMA) will be accessed in blocks of *nsamp* samples at a time. Likewise, the user programs should generate "nsamp" output samples for delivery to the DAC. Typically, users call **getblocksize()** to learn the required number of samples so that the correct amount of memory for processing the blocks of data may be allocated.

5. Optionally, the user calls

```
fs = getsamplingfrequency();
```

This function returns the best guess (assuming the HFE oscillator is exactly at 8 MHz) at the actual sampling rate being implemented (in samples/second). The actual sample rate may be slightly different from the requested rate, particularly if the Microphone is used as the input device.

6. Users then repeatedly call **getblock()** or **getblockstereo()** to obtain samples from the MIC or ADC(s), and (after processing the samples) **putblock()** or **putblockstereo()** to write the results back to the DAC. Using one of:

```
getblock(input1); // mono input
getblockstereo(input1, input2); // stereo input
```

will fill in the user's input array(s) (type float) with normalized input (ADC or MIC) data. Callers are responsible to make sure that the arrays *input1*and *input2* are allocated with enough memory to hold *nsamp* float values. After processing the input waveforms to create results, the output samples are placed in the DAC output buffers using one of

```
putblock(result1); // Mono Output
putblockstereo(result1, result2); // Stereo Output
```

Typically, after calling the **putblock()** routine, the user calls **getblock()** to wait for the next available block of input data, and the process repeats. If the DMA completes filling a new block of data before the user calls **getblock()**, a **SAMPLE_OVERRUN** error is indicated, and the RED LED is lit. In this case, the data is not being processed fast enough to keep up with the input data stream, and input samples are being lost.

Sample values in the "input" or "result" arrays are of type float, and are normalized to range from -1.0 to +1.0 over the range of the ADCs & DACs. (A sample of -1.0 is near 0V, +1.0 is near 3V, and 0.0 indicates the mid-scale voltage of 1.5V)

## Anexo C

# Archivo header de la librería del serLCD Display

```
/******************************************************************************
* @file stm32f4xx_serlcd.h
* @author Tulio Rafael Loreto and Francisco Rodríguez. Based on an
* example by Jon Masters
* @brief This file contains all the functions prototypes for the SerLCD
* library.
******************************************************************************/

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_PARALLAX_H
#define __STM32F4xx_PARALLAX_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx.h"

/* Exported constants --------------------------------------------------------*/
#define PARALLAX_BufferLength ((uint16_t)0x0040)
#define PARALLAX_Ok ((uint16_t)0x0000)
#define PARALLAX_BufferOverrun ((uint16_t)0x0001)
#define PARALLAX_BufferUnderrun ((uint16_t)0x0002)

/* Display commands ----------------------------------------------------------*/
extern const char PARALLAX_CLRSCRN[]; /* Clear the screen */
extern const char PARALLAX_MVCURSR[]; /* Move the cursor right one */
extern const char PARALLAX_MVCURSL[]; /* Move the cursor left one */
extern const char PARALLAX_DISPON[]; /* Turn visual display on */
extern const char PARALLAX_DISPOFF[]; /* Turn visual display off */

/* Initialization and Configuration functions *********************************/
void PARALLAX_Init();
uint16_t PARALLAX_IsBufferFull();
uint16_t PARALLAX_WriteChar(char c);
uint16_t PARALLAX_WriteMessage(const char* text, uint16_t length);
uint16_t PARALLAX_SendCommand(const char* cmd);
uint16_t PARALLAX_SetCursorPosition(uint16_t row, uint16_t col);

#ifdef __cplusplus
}
#endif
#endif /*__STM32F4xx_PARALLAX_H */
```

# Anexo D

# Archivo header de funciones complementarias

```
/*extras.h*/
/*Funcions and routines required to process data in main file*/


#ifndef _EXTRAS_
#define _EXTRAS_

void addimgceps (float *finalarray, float *inputarray, int length);

void hamming(float *dataOut, float *dataIn, int size2); /*Hamming Window*/

void hanning(float *dataOut2, float *dataIn2, int size3);/*Hanning Window*/

void addimgcepsmean (float *finalarray, float *inputarray, float mean, int length);

void substractmean (float *out, float *inarray, float mean, int length);

#endif
```

## Anexo E

# Código Principal del Detector de Frecuencia Fundamental de la Voz en Tiempo Real

```
/*-------------------------------------------------------
 Fundamental Frequency Detector using STM32F4Discovery
 by Tulio Rafael Loreto and Francisco Alejandro Rodríguez
 -------------------------------------------------------*/
/*-------------  Include Files-------------*/
#include "stm32f4_discovery.h"
#include "stm32f4xx_serlcd.h"
#include "ece486.h"
#include <stdlib.h>
#include "stm32f4xx.h"
#include "arm_math.h"
#include "arm_common_tables.h"
#include "arm_const_structs.h"
#include "extras.h"

#define TEST_LENGTH_SAMPLES 8192

/*Variables to be used*/
static float32_t magOutput1[TEST_LENGTH_SAMPLES/2];
static float32_t logOutput[TEST_LENGTH_SAMPLES/2];
static float32_t arrangedarray[TEST_LENGTH_SAMPLES];
static float32_t sampledarray1[TEST_LENGTH_SAMPLES];
static float32_t output[250];

/* ------------------------------------------------------------------
 Global variables for FFT. ifftFlag=0 means FFT and ifftFlag=1 means IFFT
------------------------------------------------------------------- */
uint32_t fftSize = 4096;
uint32_t ifftFlag = 0, ifftFlag2 = 1;
uint32_t doBitReverse = 1;

int32_t main(void)
{
 /* Array index at which max energy of bin occurs */
 uint32_t testIndex = 0;
 /* Value of max energy bin */
 float32_t maxValue;
 float32_t testValue;
 float32_t meanResult;

 /* Variables used within the main program*/
 arm_status status;
 int nsamp, adder, counter, lo, i, i2, i3, i4, b, time, counter;
 float *input;
 b = TEST_LENGTH_SAMPLES/fftSize;
```

```
time = 120*b; // This will create and index for two minutes of data logging
static float32_t logtime[time]; // array to store two minutes worth of data pitch estimation

/*---------------------------------------------------------------------
  Initialize the Parallax LCD, clear screen and write Frecuencia (Hz):
  on the first line. Move cursor to the second line.
---------------------------------------------------------------------*/
PARALLAX_Init();
PARALLAX_SendCommand(PARALLAX_CLRSCRN);
PARALLAX_SetCursorPosition(0, 0);
PARALLAX_WriteMessage("Frecuencia (Hz):",16);
PARALLAX_SetCursorPosition(1, 0);

/* Initialization for either ADC or Mic input*/
initialize(FS_8K, MONO_IN, MONO_OUT); // Set up the DAC/ADC interface with 8K sampling

/* Varaibles required to allocate memory for outputs and others*/
nsamp = getblocksize(); //Size of data blocks are assigned to nsamp

/*------------------------------   Allocate required memory for input----------------------------------*/
input = (float *)malloc(sizeof(float)*nsamp);

/*Control Flag for Displaying Results in LCD*/
adder = 0;
/*Array and counter used to store pitch data*/
memset(logtime, 0, sizeof(logtime));
counter = 0;

/*---------------------------------------------------------------------
   Infinite Loop to process the data stream, "nsamp" samples at a time
 ---------------------------------------------------------------------*/
while(1){
 arm_cfft_instance_f32 arm_cfft_sR_f32_lenseg = {
    4096, twiddleCoef_4096, armBitRevIndexTable4096,
          ARMBITREVINDEXTABLE4096_TABLE_LENGTH
 };
 arm_cfft_instance_f32 arm_cfft_sR_f32_lenter = {
    4096, twiddleCoef_4096, armBitRevIndexTable4096,
ARMBITREVINDEXTABLE4096_TABLE_LENGTH
  };
   status = ARM_MATH_SUCCESS; // Variable that ensures everything is good.
```

```
/*-----------------------------
 CEPSTRUM Calculation
-----------------------------*/
  /*Creation of 4096-point input array and vector mean calculation to eliminate DC component*/

  for (i=0; i<16; i++)
   {
    getblock(input); // Wait here until the input buffer is filled... Then process
    arm_mean_f32(input, nsamp, &meanResult); //calculate mean
    substractmean (output, input, meanResult, nsamp); /*subtract mean from input array, effectively eliminating DC component of input signal*/
    for (i2=0; i2<nsamp; i2++)
     {
      arrangedarray[i2+i*250] = output[i2];
     }
   }
  for (i3=4000; i3<4096; i3++)
  {
   arrangedarray[i3] = 0.00;
  }

  /*Arrange array for CFFT processing*/
  addimgceps(sampledarray1, arrangedarray, TEST_LENGTH_SAMPLES/2);

  /*Windowing the input data. Choose Hamming or Hanning windowing*/
  hamming(arrangedarray, sampledarray1, TEST_LENGTH_SAMPLES);
  //hanning(arrangedarray, sampledarray1, TEST_LENGTH_SAMPLES);

  /* Process the data through the CFFT module. arrangedarray is both input and output of CFFT*/
  arm_cfft_f32(&arm_cfft_sR_f32_lenter, arrangedarray, ifftFlag, doBitReverse);

  /* Process the data through the Complex Magnitude Squared Module for
  calculating the squared magnitude at each bin */
  arm_cmplx_mag_squared_f32(arrangedarray,magOutput1,fftSize);

  /*Base 10 log calculation for cepstrum*/
  for (lo=0;lo<fftSize;lo++){
   logOutput[lo] = log10(magOutput1[lo]);
  }

  /*Arrange array for CIFFT processing*/
  addimgceps(sampledarray1, logOutput, fftSize);
```

```
  /* Process the data through the CIFFT module. sampledarray is both input and output of CFFT*/
  arm_cfft_f32(&arm_cfft_sR_f32_lenseg, sampledarray1, ifftFlag2, doBitReverse);

  /*Get power magnitude of cepstrum*/
  arm_cmplx_mag_f32(sampledarray1, magOutput1, fftSize);

  for (i4=0; i4<16; i4++)
   {magOutput1[i4] = 0.00;
   magOutput1[4095-i4] = 0.00;}

  /* Calculates maxValue and returns corresponding BIN (position within array) value */
  arm_max_f32(magOutput1, fftSize, &maxValue, &testIndex);

 /*Arrange index. If it is on negative side of mirror, put on its positive mirrored bin*/
  if (testIndex>2047)
  {
   testIndex = 4095 - testIndex;
  }

 /*Calculate Fundamental Frequency*/
  testIndex = 8000/testIndex;

  /*Simple test routine to visualize max energy bin value and position within the array*/
  if (adder==1)
  {
  /*Lifter to eliminate impulsive system response and leave excitation values only. Stay on human pitch range*/
    if (testIndex<70)
    {
     testIndex = 0;
    }
    if (testIndex>499)
    {
     testIndex = 0;
    }

    char bufchar[16];
    char bufchar2[16];
    snprintf(bufchar,11, "%010.3f", maxValue);
    PARALLAX_SetCursorPosition(1, 0);
    PARALLAX_WriteMessage(bufchar,10);
    snprintf(bufchar2,6, "%.5i", testIndex);
    PARALLAX_SetCursorPosition(1, 11);
    PARALLAX_WriteMessage(bufchar2,5);
```

```
      adder=0;
    }

/*Log pitch value in array*/
    if (counter<time)
    {
      logtime[counter] = testIndex;
    }
    if (counter==time)
    {
      memset(logtime, 0, sizeof(logtime));
      counter = 0;
    }
    counter = counter+1;
    adder = adder+1;

    /* Press user button*/
    if (UserButtonPressed==Button_Pressed) {
      memset(logtime, 0, sizeof(logtime));// Reset Data-Logging Array on each button press.
      counter = 0;
      UserButtonPressed = Button_Ready;
    }
  }
}
```

## Anexo F

# Pasos de frecuencia en el rango de voz humana entre índices de IFFT para diversas tasas de muestreo

Valores de Frecuencia para tasa de muestreo de 8 kHz

| Posición en Arreglo | Frecuencia (Hz) |
|---|---|
| 16 | 500 |
| 17 | 470,588235 |
| 18 | 444,444444 |
| 19 | 421,052632 |
| 20 | 400 |
| 21 | 380,952381 |
| 22 | 363,636364 |
| 23 | 347,826087 |
| 24 | 333,333333 |
| 25 | 320 |
| 26 | 307,692308 |
| 27 | 296,296296 |
| 28 | 285,714286 |
| 29 | 275,862069 |
| 30 | 266,666667 |
| 31 | 258,064516 |
| 32 | 250 |
| 33 | 242,424242 |
| 34 | 235,294118 |
| 35 | 228,571429 |
| 36 | 222,222222 |
| 37 | 216,216216 |
| 38 | 210,526316 |
| 39 | 205,128205 |
| 40 | 200 |
| 41 | 195,121951 |
| 42 | 190,47619 |
| 43 | 186,046512 |
| 44 | 181,818182 |
| 45 | 177,777778 |
| 46 | 173,913043 |
| 47 | 170,212766 |
| 48 | 166,666667 |
| 49 | 163,265306 |
| 50 | 160 |
| 51 | 156,862745 |
| 52 | 153,846154 |
| 53 | 150,943396 |
| 54 | 148,148148 |
| 55 | 145,454545 |
| 56 | 142,857143 |
| 57 | 140,350877 |
| 58 | 137,931034 |
| 59 | 135,59322 |
| 60 | 133,333333 |
| 61 | 131,147541 |
| 62 | 129,032258 |
| 63 | 126,984127 |
| 64 | 125 |
| 65 | 123,076923 |
| 66 | 121,212121 |
| 67 | 119,402985 |
| 68 | 117,647059 |
| 69 | 115,942029 |
| 70 | 114,285714 |
| 71 | 112,676056 |
| 72 | 111,111111 |
| 73 | 109,589041 |
| 74 | 108,108108 |
| 75 | 106,666667 |
| 76 | 105,263158 |
| 77 | 103,896104 |
| 78 | 102,564103 |
| 79 | 101,265823 |
| 80 | 100 |
| 81 | 98,7654321 |
| 82 | 97,5609756 |
| 83 | 96,3855422 |
| 84 | 95,2380952 |
| 85 | 94,1176471 |
| 86 | 93,0232558 |
| 87 | 91,954023 |
| 88 | 90,9090909 |
| 89 | 89,8876404 |
| 90 | 88,8888889 |
| 91 | 87,9120879 |
| 92 | 86,9565217 |
| 93 | 86,0215054 |
| 94 | 85,106383 |
| 95 | 84,2105263 |
| Posición en Arreglo | Frecuencia (Hz) |
| 96 | 83,3333333 |
| 97 | 82,4742268 |
| 98 | 81,6326531 |
| 99 | 80,8080808 |
| 100 | 80 |
| 101 | 79,2079208 |
| 102 | 78,4313725 |
| 103 | 77,6699029 |
| 104 | 76,9230769 |
| 105 | 76,1904762 |
| 106 | 75,4716981 |
| 107 | 74,7663551 |
| 108 | 74,0740741 |
| 109 | 73,3944954 |
| 110 | 72,7272727 |
| 111 | 72,0720721 |
| 112 | 71,4285714 |
| 113 | 70,7964602 |
| 114 | 70,1754386 |
| 115 | 69,5652174 |
| 116 | 68,9655172 |
| 117 | 68,3760684 |
| 118 | 67,7966102 |
| 119 | 67,2268908 |
| 120 | 66,6666667 |

Valores de Frecuencia para tasa de muestreo de 12 kHz

| Posición en Arreglo | Frecuencia (Hz) |
|---|---|
| 24 | 500 |
| 25 | 480 |
| 26 | 461,538462 |
| 27 | 444,444444 |
| 28 | 428,571429 |
| 29 | 413,793103 |
| 30 | 400 |
| 31 | 387,096774 |
| 32 | 375 |
| 33 | 363,636364 |
| 34 | 352,941176 |
| 35 | 342,857143 |
| 36 | 333,333333 |
| 37 | 324,324324 |
| 38 | 315,789474 |
| 39 | 307,692308 |
| 40 | 300 |
| 41 | 292,682927 |
| 42 | 285,714286 |
| 43 | 279,069767 |
| 44 | 272,727273 |
| 45 | 266,666667 |
| 46 | 260,869565 |
| 47 | 255,319149 |
| 48 | 250 |
| 49 | 244,897959 |
| 50 | 240 |
| 51 | 235,294118 |
| 52 | 230,769231 |
| 53 | 226,415094 |
| 54 | 222,222222 |
| 55 | 218,181818 |
| 56 | 214,285714 |
| 57 | 210,526316 |
| 58 | 206,896552 |
| 59 | 203,389831 |
| 60 | 200 |
| 61 | 196,721311 |
| 62 | 193,548387 |
| 63 | 190,47619 |
| 64 | 187,5 |
| 65 | 184,615385 |
| 66 | 181,818182 |
| 67 | 179,104478 |
| 68 | 176,470588 |
| 69 | 173,913043 |
| 70 | 171,428571 |
| 71 | 169,014085 |
| 72 | 166,666667 |
| 73 | 164,383562 |
| 74 | 162,162162 |
| 75 | 160 |
| 76 | 157,894737 |
| 77 | 155,844156 |
| 78 | 153,846154 |
| 79 | 151,898734 |
| 80 | 150 |
| 81 | 148,148148 |
| 82 | 146,341463 |
| 83 | 144,578313 |
| 84 | 142,857143 |
| 85 | 141,176471 |
| 86 | 139,534884 |
| 87 | 137,931034 |
| 88 | 136,363636 |
| 89 | 134,831461 |
| 90 | 133,333333 |
| 91 | 131,868132 |
| 92 | 130,434783 |
| 93 | 129,032258 |
| 94 | 127,659574 |
| 95 | 126,315789 |
| 96 | 125 |
| 97 | 123,71134 |
| 98 | 122,44898 |
| 99 | 121,212121 |
| 100 | 120 |
| 101 | 118,811881 |
| 102 | 117,647059 |
| 103 | 116,504854 |
| 104 | 115,384615 |
| 105 | 114,285714 |
| 106 | 113,207547 |
| 107 | 112,149533 |
| 108 | 111,111111 |
| 109 | 110,091743 |
| 110 | 109,090909 |
| 111 | 108,108108 |
| 112 | 107,142857 |
| 113 | 106,19469 |
| 114 | 105,263158 |
| 115 | 104,347826 |
| 116 | 103,448276 |
| 117 | 102,564103 |
| 118 | 101,694915 |
| 119 | 100,840336 |
| 120 | 100 |
| 121 | 99,1735537 |
| 122 | 98,3606557 |
| 123 | 97,5609756 |
| 124 | 96,7741935 |
| 125 | 96 |
| 126 | 95,2380952 |
| 127 | 94,488189 |
| 128 | 93,75 |
| 129 | 93,0232558 |
| 130 | 92,3076923 |
| 131 | 91,6030534 |
| 132 | 90,9090909 |
| 133 | 90,2255639 |
| 134 | 89,5522388 |
| 135 | 88,8888889 |
| 136 | 88,2352941 |
| 137 | 87,5912409 |
| 138 | 86,9565217 |
| 139 | 86,3309353 |
| 140 | 85,7142857 |
| 141 | 85,106383 |
| 142 | 84,5070423 |
| 143 | 83,9160839 |
| 144 | 83,3333333 |
| 145 | 82,7586207 |
| 146 | 82,1917808 |
| 147 | 81,6326531 |
| 148 | 81,0810811 |

Valores de Frecuencia para tasa de muestreo de 16 kHz

| Posición en Arreglo | Frecuencia (Hz) |
|---|---|
| 32 | 500 |
| 33 | 484,848485 |
| 34 | 470,588235 |
| 35 | 457,142857 |
| 36 | 444,444444 |
| 37 | 432,432432 |
| 38 | 421,052632 |
| 39 | 410,25641 |
| 40 | 400 |
| 41 | 390,243902 |
| 42 | 380,952381 |
| 43 | 372,093023 |
| 44 | 363,636364 |
| 45 | 355,555556 |
| 46 | 347,826087 |
| 47 | 340,425532 |
| 48 | 333,333333 |
| 49 | 326,530612 |
| 50 | 320 |
| 51 | 313,72549 |
| 52 | 307,692308 |
| 53 | 301,886792 |
| 54 | 296,296296 |
| 55 | 290,909091 |
| 56 | 285,714286 |
| 57 | 280,701754 |
| 58 | 275,862069 |
| 59 | 271,186441 |
| 60 | 266,666667 |
| 61 | 262,295082 |
| 62 | 258,064516 |
| 63 | 253,968254 |
| 64 | 250 |
| 65 | 246,153846 |
| 66 | 242,424242 |
| 67 | 238,80597 |
| 68 | 235,294118 |
| 69 | 231,884058 |
| 70 | 228,571429 |
| 71 | 225,352113 |
| 72 | 222,222222 |
| 73 | 219,178082 |
| 74 | 216,216216 |
| 75 | 213,333333 |
| 76 | 210,526316 |
| 77 | 207,792208 |
| 78 | 205,128205 |
| 79 | 202,531646 |
| 80 | 200 |
| 81 | 197,530864 |
| 82 | 195,121951 |
| 83 | 192,771084 |
| 84 | 190,47619 |
| 85 | 188,235294 |
| 86 | 186,046512 |
| 87 | 183,908046 |
| 88 | 181,818182 |
| 89 | 179,775281 |
| 90 | 177,777778 |
| 91 | 175,824176 |
| 92 | 173,913043 |
| 93 | 172,043011 |
| 94 | 170,212766 |
| 95 | 168,421053 |
| 96 | 166,666667 |
| 97 | 164,948454 |
| 98 | 163,265306 |
| 99 | 161,616162 |
| 100 | 160 |
| 101 | 158,415842 |
| 102 | 156,862745 |
| 103 | 155,339806 |
| 104 | 153,846154 |
| 105 | 152,380952 |
| 106 | 150,943396 |
| 107 | 149,53271 |
| 108 | 148,148148 |
| 109 | 146,788991 |
| 110 | 145,454545 |
| 111 | 144,144144 |
| 112 | 142,857143 |
| 113 | 141,59292 |
| 114 | 140,350877 |
| 115 | 139,130435 |
| 116 | 137,931034 |
| 117 | 136,752137 |
| 118 | 135,59322 |
| 119 | 134,453782 |
| 120 | 133,333333 |
| 121 | 132,231405 |
| 122 | 131,147541 |
| 123 | 130,081301 |
| 124 | 129,032258 |
| 125 | 128 |
| 126 | 126,984127 |
| 127 | 125,984252 |
| 128 | 125 |
| 129 | 124,031008 |
| 130 | 123,076923 |
| 131 | 122,137405 |
| 132 | 121,212121 |
| 133 | 120,300752 |
| 134 | 119,402985 |
| 135 | 118,518519 |
| 136 | 117,647059 |
| 137 | 116,788321 |
| 138 | 115,942029 |
| 139 | 115,107914 |
| 140 | 114,285714 |
| 141 | 113,475177 |
| 142 | 112,676056 |
| 143 | 111,888112 |
| 144 | 111,111111 |
| 145 | 110,344828 |
| 146 | 109,589041 |
| 147 | 108,843537 |
| 148 | 108,108108 |
| 149 | 107,38255 |
| 150 | 106,666667 |
| 151 | 105,960265 |
| 152 | 105,263158 |
| 153 | 104,575163 |
| 154 | 103,896104 |
| 155 | 103,225806 |
| 156 | 102,564103 |

| Posición en Arreglo | Frecuencia (Hz) |
|---|---|
| 157 | 101,910828 |
| 158 | 101,265823 |
| 159 | 100,628931 |
| 160 | 100 |
| 161 | 99,378882 |
| 162 | 98,7654321 |
| 163 | 98,1595092 |
| 164 | 97,5609756 |
| 165 | 96,969697 |
| 166 | 96,3855422 |
| 167 | 95,8083832 |
| 168 | 95,2380952 |
| 169 | 94,6745562 |
| 170 | 94,1176471 |
| 171 | 93,5672515 |
| 172 | 93,0232558 |
| 173 | 92,4855491 |
| 174 | 91,954023 |
| 175 | 91,4285714 |
| 176 | 90,9090909 |
| 177 | 90,3954802 |
| 178 | 89,8876404 |
| 179 | 89,3854749 |
| 180 | 88,8888889 |
| 181 | 88,3977901 |
| 182 | 87,9120879 |
| 183 | 87,431694 |
| 184 | 86,9565217 |
| 185 | 86,4864865 |
| 186 | 86,0215054 |
| 187 | 85,5614973 |
| 188 | 85,106383 |
| 189 | 84,6560847 |
| 190 | 84,2105263 |
| 191 | 83,7696335 |
| 192 | 83,3333333 |
| 193 | 82,9015544 |
| 194 | 82,4742268 |
| 195 | 82,0512821 |
| 196 | 81,6326531 |
| 197 | 81,2182741 |
| 198 | 80,8080808 |
| 199 | 80,4020101 |