Estimador de F_0 en tiempo real basado en el algoritmo de YIN

Franco S. Caspe; Adrián H. Laiuppa, Oscar A. Rodriguez, Miguel A. Banchieri, Christian L. Galasso
Departamento de Ing. Electrónica
Facultad Regional Bahía Blanca
Universidad Tecnológica Nacional.
francocaspe@hotmail.com, alaiuppa@gmail.com, arodrig@frbb.utn.edu.ar, mbanch@frbb.utn.edu.ar, christian_galasso81@yahoo.com.ar

***Resumen*— Se describe la implementación de un estimador de F_0, basado en el algoritmo de YIN[1], en un microcontrolador de la familia Cortex M4. Este sistema, orientado al análisis de voz humana, fue pensado para la operación en conjunto con dispositivos de generación y reproducción de audio frecuencias, por lo que se buscó un funcionamiento en tiempo real, sin pérdida de datos y con inmunidad al ruido, todo esto asegurando un consumo computacional acorde a la capacidad del microcontrolador.**
Por último se detallan los ensayos realizados y las directivas tomadas para salvar algunas de las limitaciones del algoritmo y un error sistemático encontrado.

Palabras clave— Detección de F_0; Procesamiento de señales; Microcontrolador; Frecuencia fundamental; Sistema embebido.

I. INTRODUCCIÓN

Hoy en día existe una gran demanda de herramientas vinculadas al procesamiento de voz. Sus aplicaciones, que van desde la producción musical, hasta la fonoaudiología, pasando por la domótica, la clasificación de contenido, y los sistemas de seguridad, requieren la implementación de algoritmos que estimen la frecuencia fundamental y que sean cada vez más eficientes. Usualmente, se busca integrar estas funcionalidades en sistemas embebidos, campo que demostró contener un gran potencial de aprovechamiento por la variedad de usos finales que estos poseen.

Los microcontroladores utilizados en estos sistemas, generalmente cuentan con la capacidad de procesamiento necesaria para la implementación de un estimador. Es por ello que se decidió ensayar uno, sobre una arquitectura ARM Cortex M4 [2], extensamente difundida.

Por otro lado, las aplicaciones de sistemas embebidos centradas en procesamiento de audio, generalmente imponen la necesidad de una ejecución en tiempo real; es natural pensar que se preferirá operar con un sistema más potente y flexible, como una computadora personal, si los resultados son solicitados en forma diferida (sin restricciones de tiempo real).

Bajo este contexto se diseñó una plataforma de experimentación sobre la cual opera un estimador de frecuencia fundamental, que actúa en tiempo real, orientado a una aplicación musical, que se representa en el esquema mostrado en la Figura 1.

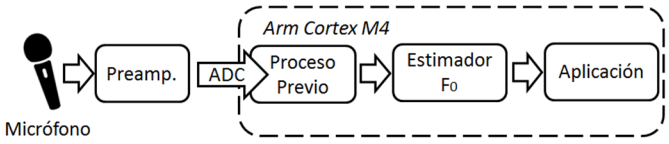

Figura 1. Esquema de la plataforma de experimentación. Se muestra el flujo de la información que será procesada por el bloque de aplicación.

Debido a la variedad de utilizaciones posibles, este trabajo está focalizado sobre los primeros bloques funcionales, explicando la captura y procesamiento de la señal, y finalmente como se detecta su frecuencia fundamental. Se buscó, así mismo, el menor tiempo de respuesta de forma de ponderar las prestaciones para su aplicación.

II. FUNDAMENTOS

Dentro de la gama de recursos musicales que son empleados comúnmente, y de los que interesa extraer la F_0, la voz humana cantada es generalmente la más difícil de procesar correctamente, debido a que presenta una serie de características que complican la determinación de su frecuencia fundamental. Un análisis del espectro revela una gran cantidad de energía inarmónica, cuya magnitud puede lograr que un estimador sea desviado hacia un resultado erróneo, especialmente si éste espera una señal puramente armónica.

Es por eso que cualquier detector de F_0, preparado para trabajar con voz debe incorporar en su modelo estrategias en las que se consideran los factores que apartan a la voz humana de una señal puramente armónica, a saber:

- Respuesta de la glotis a la vibración de las cuerdas vocales: Los inarmónicos que genera son los responsables del timbre de la voz [3].

- Consonantes: Estas fonías son naturalmente inarmónicas, su espectro frecuencial es denso dentro de un rango de frecuencias [4].

- Modulaciones: Llamados “vibratos” son variaciones naturales o intencionales sobre la amplitud y la frecuencia de la voz.

Otra característica, está vinculada a la percepción de tono, es la que genera en los algoritmos el denominado “error de octava”. Durante el canto, los resonadores acústicos del cuerpo (tórax, cabeza, nariz, otros) pueden modificar el nivel de energía de algún armónico, provocando que el algoritmo considere un pico frecuencial que no es la fundamental. Estos armónicos aumentados son conocidos como formantes. Sin embargo, auditivamente, el tono que se percibe es el correspondiente a la nota más grave y no a la de mayor energía [5].

En este campo, el algoritmo de YIN, descripto en 2002 por Alain de Cheveigne y Hideki Kawahara, supera la performance de otros métodos clásicos como Cepstrum o la interpretación de resultados obtenidos por FFT, disminuyendo los errores de octava, presentando mayor precisión y acelerando los tiempos de proceso [6].

Por otro lado, el microcontrolador seleccionado para el desarrollo del estimador, posee la simpleza suficiente para poder codificar la aplicación en bare machine, es decir sin ningún entorno de ejecución más que el propio chip, pudiendo de esta manera controlar finamente los tiempos de proceso, que serán detallados más adelante, para obtener la característica de tiempo real buscada.

Este microcontrolador, cuenta con una unidad de punto flotante de 32 bits, admite una frecuencia de reloj de hasta 168 MHz, aunque se lo operó a 96 MHz, y puede implementar puertos serie de velocidades no estándar, haciéndolo ideal para operar con dispositivos MIDI [7], de uso muy frecuente en producción musical. Por último, posee muy buenas características de gestión de energía, permitiendo administrar el consumo durante períodos de carga computacional y durante los tiempos de inactividad en donde no se detecta señal para procesar.

III. DESARROLLO

A. *Muestreo de la señal*

Considerando que se pretende trabajar con la información contenida en la voz humana y que la misma queda comprendida en la banda de bajas frecuencias, se definió como segmento útil del espectro al intervalo de frecuencias que comprende hasta los 4000 Hz [8]. En este segmento puede encontrarse tanto la frecuencia fundamental de la voz, que será procesada, como así también la inteligencia contenida, que es de interés para aplicaciones de transcripción a texto el cual es otro posible uso de los estimadores de F_0.

Para concentrar la capacidad computacional del microcontrolador en la estimación, se evitó en principio el uso de un filtro digital, por lo que la frecuencia de corte superior fue entonces definida por un filtro activo de orden 2, integrado dentro del preamplificador de entrada, rechazándose así la banda audible que no contiene información, y oficiando también de filtro anti-alias.

La señal es digitalizada entonces a una velocidad de 8 kHz, con una resolución de 12 bits, que es la máxima resolución alcanzable por los ADC del microcontrolador, y establece un piso de ruido que es despreciable frente al ruido de fondo capturado por el micrófono.

La información capturada y codificada, es volcada en un arreglo de 256 valores, ubicado en la memoria RAM, gestionado por el DMA del microcontrolador, automatizando el proceso y liberando al CPU de esta tarea. Este arreglo captura 32 ms de audio.

Teniendo en cuenta las limitaciones del algoritmo de YIN, postuladas en [1], empleando un buffer de 256 muestras, se podrá detectar períodos cuya longitud no supere las 128 muestras. Esto es debido al proceso de autocorrelación que el algoritmo ejecuta, en donde se necesitan capturar al menos dos ciclos completos de una señal para estimar correctamente su período; de manera que teóricamente, se podrá detectar una frecuencia de por lo menos 62,5 Hz (frecuencia correspondiente a un período de 128 valores, digitalizado a 8 kHz), ubicada al comienzo de la banda audible. El algoritmo no especifica una frecuencia de detección superior, por lo que, en principio, se espera una capacidad de reconocimiento algo superior a los 2000 Hz, frecuencia fundamental para la cual se detecta un período de 4 muestras.

B. *Codificación del algoritmo de YIN*

La implementación del estimador, se realizó mediante tres funciones separadas que integran la totalidad de los pasos a través de los cuales se logra obtener el período de la señal.

En primer lugar, se calcula, en formato de punto flotante lo que en el algoritmo se define como función diferencia, de longitud igual a un medio la longitud del buffer circular, como se muestra en la ecuación (1):

$$d_t(\tau) = \sum_{j=1}^{W} (x_j - x_{j+\tau})^2 \qquad (1)$$

Donde *x*, es el arreglo de la señal de entrada, en formato entero, de longitud de 12 bits, acoplado a un valor constante de offset que idealmente equivale a la mitad del rango numérico que admite la resolución de muestreo.

Por otro lado, *W* es la longitud de la ventana sobre la cual se efectúa la diferencia y τ es la variable independiente que controla su deslizamiento.

El cálculo de $d_t(\tau)$, se implementó en una primera función, que admite el arreglo de valores muestreados, y devuelve los 128 valores de la función diferencia.

Posteriormente, debe normalizarse de forma acumulativa la función diferencia, obteniendo una secuencia de la misma longitud que la anterior, pero que admite valores entre 0 y 1. Este segundo paso, fue implementado en otra función separada. El proceso de normalización se muestra en la ecuación (2).

$$d'(\tau)=\begin{cases}1, & \text{si } \tau=0,\\ d_t(\tau)\Big/\left[(1/\tau)\sum_{j=1}^{\tau} d_t(j)\right] & \text{si } \tau\neq 0.\end{cases} \quad (2)$$

Como puede deducirse de las ecuaciones (1) y (2), la señal muestreada no necesita ser normalizada ni desacoplada digitalmente, puesto que las funciones calculadas dependen únicamente de las variaciones entre muestras, y este resultado se normaliza posteriormente. Evitando estos dos procesos, se logra incrementar la eficiencia del cómputo.

Posteriormente, siguiendo los pasos del algoritmo, se codificó una tercera función que incluye en primer lugar la búsqueda de la posición $\tau_{mín}$ dentro de la región de $d'(\tau)$ caracterizada por la presencia de valores menores a un determinado umbral denominado "umbral de YIN". Es recomendable que este valor sea lo suficientemente estricto para evitar que el algoritmo genere errores de octava. Si no se encuentra ningún valor que cumpla con estas características, se buscará el mínimo global de toda la función.

Finalmente, se incluye en la misma función, el proceso de interpolación de una parábola positiva entre ($\tau_{mín} - 1$) y ($\tau_{mín} + 1$), que permite hallar, haciendo ciertas suposiciones sobre sobre el contenido armónico de la señal, el valor final de período de *x*, denominado $\tau_{mín}$'. Por último, la amplitud mínima de la parábola interpolada, es considerada por el autor del algoritmo como una medida de la energía aperiódica de la señal, teniendo en cuenta que, idealmente, este valor debería ser cero cuando la señal a comparar y la contenida en la ventana son perfectamente iguales. Este proceso se muestra en la Figura 2.

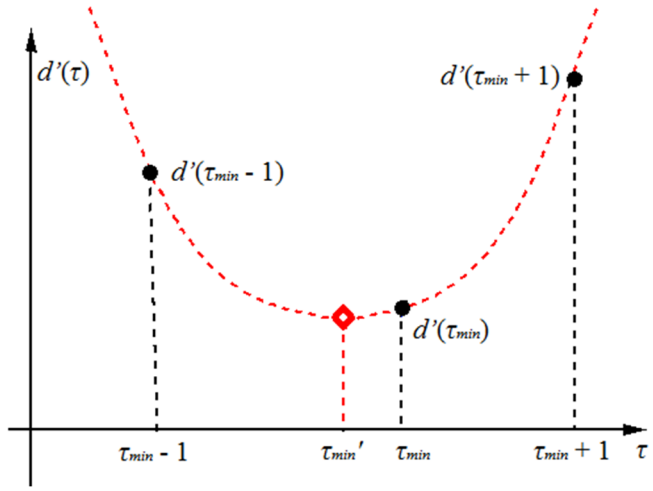

Figura 2. Búsqueda del mínimo e interpolación parabólica

C. Ajuste de las características de tiempo real

Considerando las tres funciones implementadas, y bajo las condiciones de operación anteriormente mencionadas, se midieron los tiempos de procesamiento de cada una de las tres operaciones que juntas permiten arribar al valor de período de la señal. Los tiempos se determinaron primero utilizando un timer del microcontrolador, reseteado y disparado antes de comenzar cada operación, y luego se cotejaron utilizando una salida digital de señalización del microcontrolador, asignable en una operación unitaria, y un osciloscopio. Se creó una señal de prueba grabada como constante en memoria, que obligaba al algoritmo a ejecutar la búsqueda del mínimo más largo posible. Los resultados se muestran en la Tabla 1.

TABLA I. TIEMPO DE PROCESAMIENTO DE LAS OPERACIONES DE YIN

Operación	***Tiempo* [µs]**
1. Cálculo de la función diferencia	7378
2. Normalización de la función diferencia	9494
3. Algoritmo de búsqueda e interpolación	9534
Tiempo total de procesamiento	**26406**

Con el objetivo de no interrumpir el muestreo para no perder información, y teniendo en cuenta que el acceso a memoria se realiza de forma progresiva en el tiempo durante el cálculo de la función diferencia (acorde la ventana se desliza hasta el final del arreglo), es posible realizar este cómputo en paralelo con el refresco del buffer, debido a que el análisis de la información se realiza más velozmente que la sobre escritura. Esto permite realizar una gestión fina del temporizado, evitando la necesidad de utilizar buffers dobles, por ejemplo, ampliando la gama de dispositivos que toleren la implementación. Un diagrama temporal se muestra en la Figura 3.

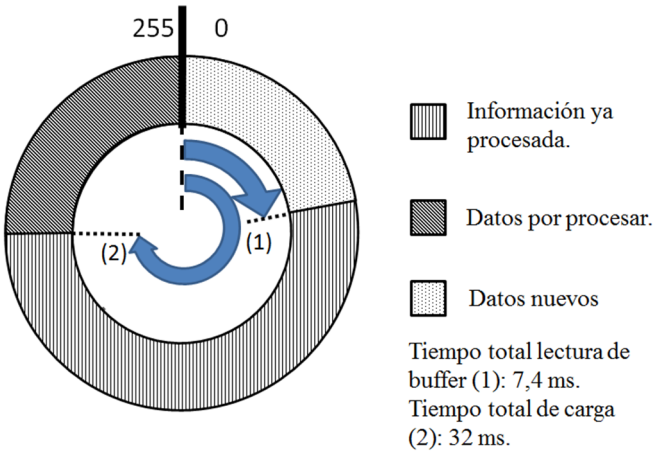

Figura 3. Diagrama de procesos en un instante temporal. La lectura progresiva de la información del buffer (2) se realiza mas rápidamente que su sobreescritura (1).

Por otro lado, dado que para procesar 32 ms de audio, se tarda más de 26 ms, no fue posible aumentar de forma práctica la cadencia de detecciones, efectuándose entonces una estimación de frecuencia cada 32 ms. El microcontrolador tendrá un espacio de 5 ms para realizar las tareas de rutina como refresco del display, control de la interfaz de usuario, otros. El diagrama de la Figura 4 muestra la evolución temporal de los procesos.

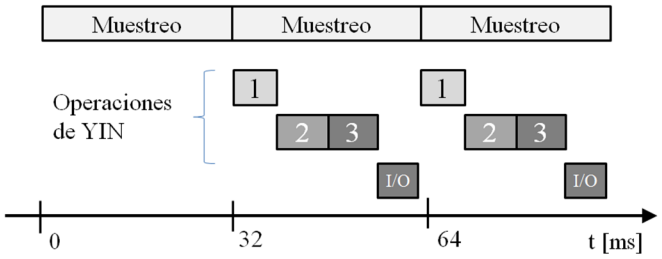

Figura 4. Diagrama temporal de respuesta. Téngase en cuenta que las operaciones anteriores y sucesivas de muestreo y cálculo no se muestran.

Habiendo determinado los tiempos de operación, y con la certeza de que no se perderá información, se puede afirmar que el sistema presenta la característica de tiempo real, por lo que interesa conocer el tiempo de respuesta.

Considérese un evento que se quiere detectar, como un cambio en la frecuencia fundamental de la señal de entrada. El máximo retardo en la detección estará dado por el tiempo de carga del buffer, sumado al tiempo de ejecución del estimador, dando un total de 58,4 ms. El esquema de proceso se muestra en la Figura 5.

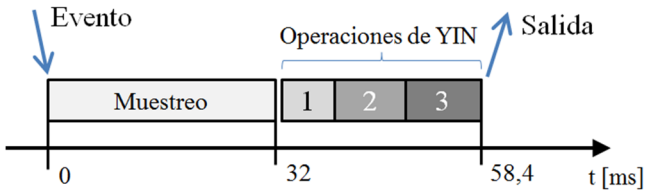

Figura 5. Diagrama temporal de respuesta. Téngase en cuenta que las operaciones anteriores y sucesivas de muestreo y cálculo no se muestran.

Para mejorar el tiempo de respuesta, se tuvieron en cuenta nuevamente los tiempos de acceso y de carga del buffer circular; si se dispara el cálculo de la función diferencia antes de que finalice el llenado, se puede lograr que ambas operaciones concluyan casi simultáneamente, reduciendo el tiempo de respuesta hasta 7 ms, como se muestra en la Figura 6.

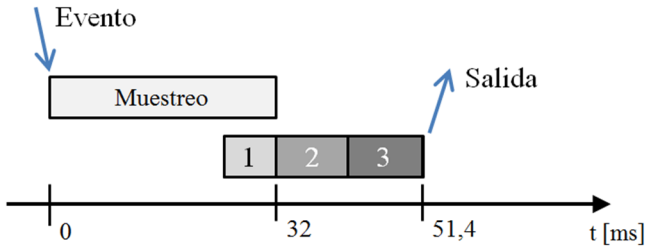

Figura 6. Reducción del retardo por paralelización del proceso de lectura/escritura.

D. Reducción de la sensibilidad al ruido

El algoritmo de YIN tiene la particularidad de asignar un período válido a un amplio rango de señales. El ruido eléctrico o el ruido de fondo de un micrófono, por ejemplo, suelen presentar variaciones que, dentro del cuadro de análisis, se consideran de naturaleza armónica, produciéndose un valor de F_0 erróneo. La ocurrencia de este fenómeno es independiente de la amplitud de la señal de entrada (debido a que la función diferencia se normaliza).

Con el objetivo de evitar la implementación de un discriminador basado en el cálculo del valor RMS de la entrada, puede establecerse un segundo umbral que contempla la aperiodicidad mínima de la función diferencia normalizada.

Recordando que la aplicación del "umbral de YIN" define una región de búsqueda sobre $d'(\tau)$, el umbral de discriminación determinará si esa búsqueda es aplicable, analizando la presencia de al menos un valor menor a este límite, sobre la función mencionada.

Durante las pruebas previas a la implementación, se descubrió que un valor adecuado para el umbral de discriminación fue 0,2, mientras que el valor seleccionado para el umbral de YIN fue de 0,1. En las Figuras 7 y 8, se comparan los resultados logrados al aplicar o no la discriminación, utilizando como entrada una señal capturada por un micrófono electret desde una PC. En la misma, un vocalista ejecuta una escala ascendente nombrando cada una de las notas que canta. Los parámetros de operación se definieron iguales a los del microcontrolador.

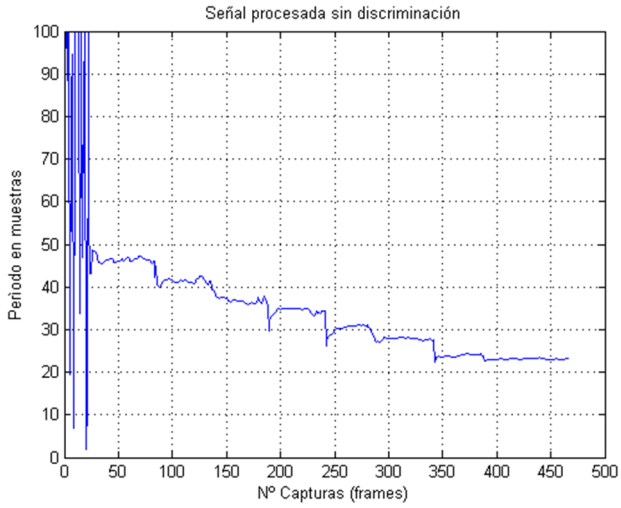

Figura 7. Evolución del período de una señal de voz cantada. El ruido de fondo al inicio de la grabación genera valores incorrectos.

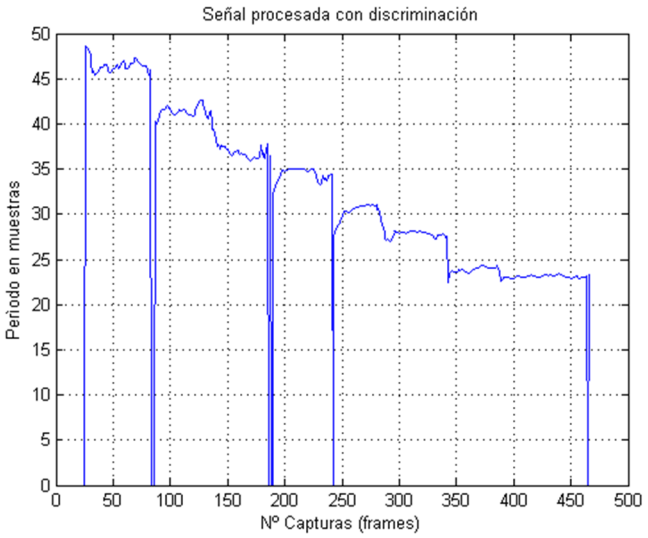

Figura 8. Procesamiento de la señal con discriminador de aperiodicidad. Se observa que no se detectan eventos hasta que la voz no se escucha en la grabación. Adicionalmente se detectan separadas varias de las notas ejecutadas (detección de consonantes).

IV. ENSAYOS REALIZADOS

Considerando el modelo establecido en la Figura 1, para el primer ensayo se diseñó un convertidor de frecuencia a MIDI, con el objetivo de poder observar en una escala musical la evolución de la frecuencia fundamental del audio captado por el micrófono.

El convertidor genera una nota musical cuando se detecta un nuevo evento, es decir, se obtiene un valor de período válido. Adicionalmente, se redondearán los valores de frecuencia fundamental hacia la nota más cercana de la escala.

El primer ensayo consistió en conectar un micrófono, y sin ejecutar nota alguna, elevar la ganancia del preamplificador de entrada, confirmándose la ausencia de disparos por ruido.

Posteriormente se ensayó el dispositivo a ganancia reducida, observándose una detección correcta hasta una determinada nota ejecutada por el vocalista, donde se detecta en la salida, un desplazamiento de un semitono hacia los agudos. Se comprobó en esta instancia la correcta generación de los eventos de cambio y aparición de frecuencia a la entrada, aunque con un error en el valor detectado.

Para caracterizar el error encontrado en el ensayo anterior, se modificó el bloque de aplicación, con la función de mostrar el valor de frecuencia fundamental detectada, en un display instalado para tal fin. Se conectó un generador de funciones a la entrada, buscando realizar el ensayo con una señal armónica pura.

Se efectuaron dos barridos, partiendo desde el mínimo valor detectable (62,5 Hz), hasta la frecuencia considerada de detección factible, correspondiente a un período de 4 muestras. Estos se efectuaron empleando dos valores distintos de umbral de YIN.

En las Figura 9, se muestran los valores de error relativo obtenido, en función de la fracción entre la frecuencia de entrada y la de muestreo. Se comienza a graficar desde f/f_s = 0,01, dado que para valores menores, el error observado no supera el 1 %.

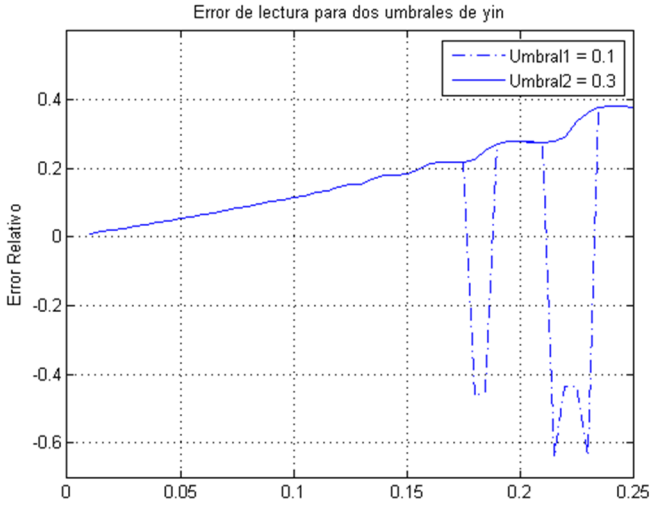

Figura 9. Gráfica del error relativo, para dos valores de umbral de operación, computado entre la frecuencia de entrada, y la lectura del equipo. Las gráficas generalmente coinciden, separándose en dos rangos distintos de frecuencias.

Observando la Figura 9, pueden extraerse dos conclusiones importantes: primero, se verifica la existencia de un error acumulativo con la frecuencia, independiente de las condiciones de operación del algoritmo. Por otro lado se observa que si se opera con un valor de umbral pequeño, aparecen dos regiones de detección en donde no se respeta la correspondencia uno a uno entre la frecuencia de entrada y la leída por el dispositivo. Se supone, que esto último se debe a que al trabajar con valores pequeños de umbral, y períodos de escasas muestras, la región de búsqueda que se abre en la función diferencia normalizada puede llegar a no contener el mínimo representante del período de la señal. Evidentemente, los valores de aperiodicidad aumentan cuando el período se achica. La falta de resolución en muestras puede llegar a ser el factor causante de este problema.

Posteriormente se descubrió que el error acumulativo se debía a una interpretación textual y errónea de la posición $\tau_{mín}$, que indica el valor del período de la señal. Debido a que el arreglo de la función diferencia comienza en la posición cero, una señal con un período de 64 muestras, tiene un mínimo en la posición 63, un período de 8 muestras, en la ubicación 7, y así sucesivamente. Si se toman como válidas las posiciones sobre la función diferencia, se generará un patrón de error relativo que crece con la frecuencia, acorde el período se achica. La solución a este problema fue justamente, sumar una unidad al valor calculado.

Respecto al quiebre de la correspondencia inequívoca entre la frecuencia de entrada y la lectura, se prefirió operar con un valor de umbral de YIN pequeño, debido a que la aplicación de

una restricción más laxa genera errores de octava en toda la banda útil de detección. El valor máximo de frecuencia para el cual no se presentan problemas, es del 17,5 % de la frecuencia de muestreo, lo que corresponde a un 35 % de la banda de Nyquist. Cabe aclarar que teóricamente, solo el 65 % inferior de esta banda presenta condiciones favorables de operación, debido a que la detección de períodos de 3 muestras o menos no es estable si no se sincroniza la frecuencia de muestreo con la señal de entrada.

De esta manera se definió el rango útil de operación del algoritmo, que está comprendido entre los 62,5 Hz y la frecuencia máxima de detección, que resulta ser de 1400 Hz, correspondiente a la nota F6, que escapa del rango de las tesituras vocales clásicas (la voz soprano tiene tabulada al DO6 como nota máxima).

V. Conclusiones

El estimador presentado, cumplió con los requisitos postulados al principio del trabajo. Sin embargo, si se desea asegurar un comportamiento en tiempo real, sin pérdida de información, no será posible la implementación de alguna aplicación de relevancia debido a la falta de tiempo de operación disponible.

El cuello de botella presentado por el sistema quedó determinado por la capacidad y frecuencia de operación del procesador. Sin embargo, el Cortex M4 utilizado ofrece la posibilidad de incrementar en casi un 60% la velocidad de reloj utilizada para el presente desarrollo, pudiéndose así por ejemplo, disparar dos veces el estimador por cada llenado de un buffer más corto, o de ampliar las funcionalidades del sistema embebiendo una aplicación cuyo tiempo de operación dentro del bucle principal sea mayor a 5ms.

Por otro lado, es posible modificar la frecuencia de muestreo y la longitud de la ventana temporal para ajustar el rango de detección. Sin embargo, el incremento de la frecuencia de reloj y el aumento de la velocidad de muestreo dan lugar a nuevas condiciones de operación óptimas, de tiempo de respuesta mínimo. Si se cuenta con velocidades lo suficientemente altas, será la duración de la ventana temporal la que defina el retardo de la respuesta.

Siguiendo con esa idea, el retardo obtenido, de 52 ms, es generalmente demasiado grande para algunas aplicaciones en donde se requiere velocidad de análisis, como los sistemas de efectos digitales, utilizados en conciertos musicales y espectáculos. Deberán aplicarse las estrategias anteriormente mencionadas para lograr retardos adecuados a la aplicación, reservando también el espacio temporal adecuado para su operación.

Respecto al rango limitado de detección para valores pequeños de umbral de YIN, si se debe operar el estimador con otros instrumentos musicales de tesitura más amplia, puede implementarse un filtro IIR, de baja demanda computacional, delimitador de rango, para establecer una comparación entre la energía de la señal de entrada y salida del filtro. Este análisis diferencial puede comandar el proceso de estimación de F_0. Esta estrategia fue implementada posteriormente, incrementando la frecuencia de operación del microcontrolador, con resultados satisfactorios.

La implementación del algoritmo de YIN presentada resulta ideal para el desarrollo de sistemas embebidos con requerimientos de tiempo real, por la flexibilidad de su rango de operación, implementación y consumo computacional.

VI. Trabajos a futuro

Debido a que el tiempo de operación queda determinado en gran medida por la duración temporal del buffer circular, es interesante considerar un arreglo de longitud variable, a frecuencia de muestreo constante, con el que pueden reducirse drásticamente los tiempos de cálculo.

La expansión y compresión del buffer debería darse en torno a una pauta. Por ejemplo, si se están cantando notas agudas, la longitud puede reducirse, debido a que las mismas poseen un período reducido. La implementación de este mecanismo requiere el diseño de un mecanismo de toma de decisiones complejo.

Referencias

[1] “YIN, a fundamental frequency estimator for speech and music” Disponible para acceso libre en: http://iwk.mdw.ac.at/lit_db_iwk/download.php?id=12777

[2] Arm Developer. Detalle de la familia de microcontroladores Cortex M4. https://developer.arm.com/products/processors/cortex-m/cortex-m4

[3] “Measures of the glottal source spectrum”. Jody Kreiman, Bruce R. Gerratt, Norma Antoñanzas-Barroso. http://www.linguistics.ucla.edu/people/keating/Kreiman%20et%20al%202007.pdf

[4] “Speech spectra and Spectograms” de Robert Mannell. Pronunciación de consonantes y la evolución de su espectro en el tiempo. http://clas.mq.edu.au/speech/acoustics/speech_spectra/oral_stops.html

[5] “The Singer’s Formant” Detalle de la distribución frecuencial de la voz, bajo el fenómeno de resonancia. Disponible en: http://www.ncvs.org/ncvs/tutorials/voiceprod/tutorial/singer.html

[6] “Pitch Extraction and Fundamental Frequency: History and Current Techniques” David Gerhard. Disponible para acceso público en: http://www2.cs.uregina.ca/~gerhard/publications/TRdbg-Pitch.pdf

[7] Especificación del protocolo MIDI de interconexión de instrumentos musicales. Enlace a la página del estándar: https://www.midi.org/

[8] “Defining Analog Voice” Nota técnica sobre, canales de comunicación de Cisco. http://www.cisco.com/c/en/us/support/docs/voice/h323/8628-define-analog-voice.html