INVESTIGACIÓN TECNOLÓGICA IST CENTRAL TÉCNICO

Volumen 7 · Número 1 · Junio 2025 · Publicación semestral

EVALUACIÓN DEL RENDIMIENTO DE FILTROS FIR EN MICROCONTROLADORES ECONÓMICOS PARA APLICACIONES DE PROCESAMIENTO DE VOZ.

INVESTIGACIÓN TECNOLÓG.ICA IST CENTRAL TÉCNICO

ISSN: 2600-5565

Performance Evaluation of FIR Filters on Low-Cost Microcontrollers for Voice Processing Applications.

Evaluación del rendimiento de filtros FIR en microcontroladores económicos para aplicaciones de procesamiento de voz.

Sebastián Lozada[1][0000-0002-0315-246X], **Lenin Ramiro Merino Villegas**[2][0000-0003-4235-9604], **Katherine Cumbe Vega**[3][0000-0003-4235-9604]

[3] *Instituto Superior Universitario Central Técnico, Quito, Ecuador*
E-mail: wlozada@istct.edu.ec
[2 1] *Instituto Superior Universitario Sucre, Quito, Ecuador*
E-mail: lmerino@tecnologicosucre.edu.ec
[3] *Instituto Superior Universitario Central Técnico, Quito, Ecuador*
E-mail: kcumbe@istct.edu.ec

Recibido: 10/04/2025
Aceptado: 10/06/2025
Publicado: 30/06/2025

RESUMEN

Este artículo presenta un análisis del comportamiento de filtros digitales de respuesta finita al impulso (FIR) implementados en microcontroladores de bajo costo, con el fin de mejorar la calidad de señales de voz en ambientes ruidosos. Se emplearon dos plataformas: Arduino Mega 2560 y ESP8266, programadas mediante el entorno Arduino IDE. Se diseñó un filtro pasa banda para el rango de 85-255 Hz utilizando una ventana de Hamming. La evaluación se basó en métricas como la relación señal/ruido y el tiempo de procesamiento, aplicadas a datos capturados de seis participantes. Los resultados evidencian diferencias en la capacidad de procesamiento y atenuación de ruido entre ambos dispositivos, destacando al ESP8266 como una opción más robusta para escenarios exigentes. Este estudio aporta una visión útil para desarrolladores que buscan soluciones accesibles en procesamiento digital de señales.

Palabras clave: Filtros FIR; Arduino; ESP8266; Microcontroladores; Señales de voz.

ABSTRACT

This article presents an analysis of the behavior of finite impulse response (FIR) digital filters implemented on low-cost microcontrollers, aimed at enhancing voice signal quality in noisy environments. Two platforms were used: Arduino Mega 2560 and ESP8266, programmed through the Arduino IDE. A bandpass filter targeting the 85–255 Hz range was designed using a

Lozada, S. Merino, L. & Cumbe, K. (2025). Evaluación del rendimiento de filtros FIR en microcontroladores económicos para aplicaciones de procesamiento de voz.

ISSN: 2600-5565

Hamming window. The evaluation was based on metrics such as signal-to-noise ratio and processing time, applied to data gathered from six participants. Results reveal differences in processing capabilities and noise attenuation, with the ESP8266 emerging as a more robust option for demanding scenarios. This study offers valuable insights for developers seeking accessible solutions in digital signal processing.

Index terms:

1. INTRODUCCIÓN.

El procesamiento digital de señales (DSP, por sus siglas en inglés) se ha consolidado como una herramienta importante en la era tecnológica actual, con aplicaciones en áreas tan diversas como las telecomunicaciones, la medicina, la acústica, la ingeniería eléctrica y los sistemas de control automático (Bertran, 2006; Moya, 2011). Esta disciplina permite la manipulación matemática de señales mediante algoritmos implementados en plataformas computacionales, lo que ha revolucionado la forma en que se capturan, transmiten y procesan datos del mundo real.

Uno de los principales desafíos en el tratamiento de señales reales es la presencia de ruido e interferencias no deseadas que pueden distorsionar la información contenida en dichas señales. En este contexto, el filtrado digital se posiciona como una de las técnicas más empleadas para eliminar componentes indeseables, suavizar señales o extraer características relevantes (Bobadilla, Gómez & Bernal, 1999). Los filtros digitales pueden clasificarse en dos grandes grupos: los de respuesta infinita al impulso (IIR) y los de respuesta finita al impulso (FIR). Aunque ambos tipos son útiles, los filtros FIR se destacan por ser intrínsecamente estables y por conservar una fase lineal, lo que los convierte en la opción preferida para aplicaciones donde la fidelidad temporal de la señal es crítica, como en el procesamiento de señales de voz (Suárez & Jacinto, 2009; López Marín, 2003).

Los filtros FIR requieren una mayor cantidad de operaciones matemáticas en comparación con los IIR, ya que no utilizan retroalimentación y su salida depende exclusivamente de una cantidad finita de valores pasados de la entrada (Paz, Rodríguez & Galasso, 2016). Esta característica, aunque ventajosa desde el punto de vista de la estabilidad, impone exigencias considerables al hardware, especialmente cuando se desea implementar dichos filtros en sistemas embebidos con recursos computacionales limitados.

En este contexto, los microcontroladores de bajo costo, como el Arduino Mega 2560 y el ESP8266, han emergido como plataformas atractivas para tareas de prototipado, educación técnica, e incluso implementación de sistemas funcionales en el ámbito del Internet de las Cosas (IoT) (Herrador, 2009; Arduino, 2015; Valderrama & Brea, 2020). Estos dispositivos ofrecen una alternativa accesible y versátil para la programación de filtros digitales, permitiendo a investigadores, estudiantes y desarrolladores llevar a cabo soluciones de bajo presupuesto con resultados satisfactorios.

Lozada, S. Merino, L. & Cumbe, K. (2025). Evaluación del rendimiento de filtros FIR en microcontroladores económicos para aplicaciones de procesamiento de voz.

ISSN: 2600-5565

El Arduino Mega 2560 cuenta con un microcontrolador ATmega2560 de 8 bits, 16 MHz de frecuencia de reloj, y 8 KB de memoria SRAM. Aunque es ideal para tareas de control, su capacidad de procesamiento puede resultar limitada frente a aplicaciones intensivas como el filtrado en tiempo real. Por otro lado, el ESP8266, con su arquitectura de 32 bits y capacidad de operar hasta a 160 MHz, además de contar con conectividad Wi-Fi integrada, representa una opción más potente para tareas de procesamiento digital más exigentes (Valderrama & Brea, 2020).

Sin embargo, pese a la creciente popularidad de estas plataformas, son pocos los estudios empíricos que comparan de forma sistemática su rendimiento al ejecutar filtros FIR bajo condiciones realistas. La mayoría de investigaciones se centran en aspectos de conectividad, control de periféricos o automatización básica, dejando de lado el análisis de su comportamiento en tareas de procesamiento de señales digitales (Álvarez Cedillo, Lindig Bos & Martínez Romero, 2008; Romero et al., 2009).

En particular, el procesamiento de señales de voz es un caso de estudio relevante, ya que estas señales poseen características frecuenciales bien definidas y son sensibles a la distorsión en fase o a la atenuación selectiva de ciertas bandas. El rango fundamental de la voz humana oscila entre los 85 Hz y 255 Hz, siendo esta la banda que se busca preservar al momento de aplicar un filtrado digital (Martínez Barrera et al., s. f.). Para lograrlo, una técnica efectiva es el uso de ventanas, como la ventana de Hamming, que permite reducir la presencia de lóbulos secundarios en la respuesta en frecuencia del filtro, mejorando su selectividad y minimizando el efecto de fugas espectrales (Bertran, 2006; Martínez Barrera et al., s. f.).

Asimismo, la correcta selección de la frecuencia de muestreo es fundamental para evitar aliasing y preservar la integridad de la señal original. Según el teorema de Nyquist-Shannon, se debe utilizar una frecuencia al menos el doble de la frecuencia máxima contenida en la señal. Para la voz humana, cuya banda puede llegar hasta los 4 kHz, se recomienda una frecuencia de muestreo de al menos 8 kHz (Osorio et al., 2008).

Considerando lo anterior, el presente trabajo se orienta a evaluar el rendimiento de un filtro FIR pasa banda implementado en las plataformas Arduino Mega 2560 y ESP8266, con énfasis en su capacidad para filtrar señales de voz bajo condiciones de ruido ambiental. Para ello, se diseñó un filtro con ventana de Hamming centrado en el rango de 85-255 Hz, y se programó en ambas plataformas utilizando el entorno Arduino IDE. A través de pruebas experimentales con seis participantes, se recolectaron datos sobre la relación señal/ruido (SNR), el tiempo de procesamiento y la estabilidad del filtrado, con el objetivo de determinar cuál de los dos dispositivos ofrece un mejor desempeño bajo las mismas condiciones de operación. Este estudio no solo aporta evidencia práctica sobre las capacidades reales de microcontroladores económicos en el procesamiento digital de señales, sino que también pretende ofrecer una guía útil para su aplicación en contextos educativos, de investigación y de desarrollo tecnológico de bajo costo. Además, abre la puerta a nuevas líneas de investigación, como la evaluación de filtros IIR en los mismos entornos, la implementación de filtros adaptativos, o el uso de otras ventanas

ISSN: 2600-5565

como la de Blackman, Kaiser o Hann, para comparar su eficacia en condiciones similares (Gil Jiménez, s. f.; Moya, 2011).

MATERIALES Y MÉTODOS / DESARROLLO

2.1 Diseño Experimental:

El estudio adoptó un enfoque cuantitativo y experimental, orientado a analizar el rendimiento de un filtro digital de respuesta finita al impulso (FIR) implementado sobre microcontroladores de bajo costo. El objetivo fue evaluar la efectividad del filtrado de señales de voz en entornos con ruido ambiental simulado, empleando plataformas Arduino Mega 2560 y ESP8266, ampliamente utilizadas en entornos educativos y de desarrollo.

Se diseñó un entorno de pruebas controlado en el que seis participantes emitieron señales de voz previamente definidas, las cuales fueron capturadas, digitalizadas, y procesadas mediante el filtro FIR programado en ambos dispositivos. La evaluación se centró en observar el comportamiento del sistema en condiciones de carga realista, considerando métricas como la relación señal-ruido (SNR), la respuesta temporal y la estabilidad del filtrado.

Para garantizar la replicabilidad de los resultados, se estableció una frecuencia de muestreo de 8000 Hz, suficiente para cubrir el espectro relevante de la voz humana (hasta aproximadamente 4 kHz), en cumplimiento con el teorema de Nyquist. Las señales se registraron mediante el uso de un micrófono de electreto conectado a un preamplificador con salida analógica, acoplado a las entradas de los microcontroladores.

La implementación del filtro y la recolección de datos se realizaron utilizando herramientas de libre acceso y código abierto, favoreciendo el acceso a la tecnología y la posibilidad de replicar el experimento en otros contextos. Esta metodología también permite una adecuada transferencia al aula, fortaleciendo el aprendizaje basado en proyectos.

2.2 Especificaciones de hardware

Arduino Mega 2560

Este microcontrolador incorpora el chip ATmega2560, una arquitectura de 8 bits con frecuencia de reloj de 16 MHz, 8 KB de memoria SRAM, y 54 pines de entrada/salida digital. Su facilidad de uso, compatibilidad con sensores y bajo costo lo han convertido en un estándar en contextos educativos y de prototipado rápido.

ESP8266 NodeMCU v1.0

Basado en el núcleo Tensilica L106, este dispositivo ofrece una frecuencia de operación de 80 MHz, con posibilidad de overclock hasta 160 MHz, 96 KB de RAM y conectividad Wi-Fi integrada.

ISSN: 2600-5565

Es ampliamente utilizado en soluciones de Internet de las Cosas (IoT) por su potencia de procesamiento y capacidad de conectividad inalámbrica.

Ambos dispositivos fueron programados utilizando el entorno de desarrollo Arduino IDE, por su compatibilidad con múltiples plataformas, facilidad de depuración y amplia documentación de soporte (Arduino, 2015).

2.3 Diseño del filtro

Para el presente estudio se diseñó un filtro digital de tipo FIR (Finite Impulse Response) con el objetivo de eliminar el ruido de fondo y conservar únicamente las frecuencias esenciales de la señal de voz humana. Este tipo de filtro fue seleccionado debido a su estabilidad inherente (al no utilizar retroalimentación) y a su fase lineal, lo que garantiza que las componentes de la señal no se vean desplazadas temporalmente, preservando su forma original (Suárez & Jacinto, 2009).

Filtro FIR

Un filtro FIR es un algoritmo que procesa una señal digital aplicando una fórmula matemática que multiplica las muestras recientes de la señal por un conjunto de coeficientes predefinidos. La salida del filtro es la suma de todos estos productos. A diferencia de los filtros IIR, los FIR no dependen de valores anteriores de la salida, lo cual evita la acumulación de errores y garantiza una respuesta controlada. Esto es especialmente útil cuando se trabaja con microcontroladores de bajo costo que no cuentan con hardware especializado para operaciones matemáticas complejas.

La estructura matemática general de un filtro FIR de orden N es:

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

Donde:

$y[n]$: es la salida filtrada en el instante n

$x[n-k]$: son las muestras pasadas de la entrada

$h[k]$: son los coeficientes del filtro (también llamados impulso del sistema)

N: es el orden del filtro, relacionado directamente con la cantidad de coeficientes utilizado.

Aplicación de la ventana de Hamming

Para mitigar estos efectos, se aplicó la técnica de ventaneo, específicamente utilizando la ventana

ISSN: 2600-5565

de Hamming, una de las más utilizadas por su equilibrio entre atenuación de lóbulos secundarios y ancho de banda. Esta técnica consiste en multiplicar los coeficientes ideales del filtro por una función ventana, lo cual suaviza las transiciones y mejora la calidad del filtrado.

La ecuación matemática de la ventana de Hamming aplicada es la siguiente:

$$\omega(n) = \frac{1 - \cos\left(\frac{2\pi n}{M}\right)}{2}, 0 \leq n \leq M$$

donde:

$\omega(n)$: es el valor de la ventana en la posición

M: es el número total de coeficientes menos uno,

n: es el índice actual de la muestra.

Esta ventana suaviza los bordes de la respuesta al impulso, evitando oscilaciones abruptas que puedan introducir ruido no deseado. Visualmente, se asemeja a una “campana” que reduce progresivamente el peso de los coeficientes más alejados del centro.

Configuración del filtro para implementado

Para este proyecto se estableció una longitud de filtro de 31 coeficientes (orden 30). Esta longitud fue elegida como un compromiso entre una buena definición espectral y el rendimiento computacional aceptable para microcontroladores con recursos limitados como el Arduino Mega 2560.

Los coeficientes del filtro fueron calculados utilizando herramientas matemáticas en línea y validados con simulaciones previas en software especializado, asegurando que el filtro realmente cumplía con el perfil de pasa banda deseado. El proceso de cálculo consistió en tres pasos:

1. Definición de la frecuencia de muestreo: Se estableció en 8000 Hz para cumplir con el teorema de Nyquist.
2. Cálculo de frecuencias normalizadas: Se convirtieron las frecuencias objetivo (85 Hz y 255 Hz) en fracciones de la frecuencia de muestreo.
3. Generación de coeficientes: Se utilizó una herramienta de diseño de filtros (como fiiir.com) para obtener los valores y exportarlos en formato compatible con Arduino.

Finalmente, los coeficientes fueron codificados directamente en el entorno Arduino IDE, en un arreglo de tipo float, lo que permitió su uso en tiempo real durante la adquisición de datos desde el micrófono y el procesamiento mediante sumatorias desplazadas.

ISSN: 2600-5565

Implementación en código (Arduino IDE)

La programación del filtro FIR se realizó en el entorno Arduino IDE mediante código estructurado que incluye las siguientes etapas:

1. Definición del arreglo de coeficientes:

Se declaró un arreglo de tipo float con los 31 coeficientes precalculados.

```
float h[31] = {
  -0.0012, -0.0023, -0.0034, -0.0042, -0.0043, -0.0031, 0.0000, 0.0054,
  0.0129, 0.0215, 0.0293, 0.0341, 0.0343, 0.0290, 0.0185, 0.0043,
  -0.0114, -0.0262, -0.0370, -0.0415, -0.0384, -0.0279, -0.0123, 0.0054,
  0.0216, 0.0329, 0.0370, 0.0335, 0.0232, 0.0080, -0.0093
};
```

2. Captura de señal:

Se utilizó un convertidor analógico a digital (ADC) del microcontrolador para tomar lecturas del micrófono cada 125 µs (8000 muestras por segundo). Esto se logró mediante una función millis() o interrupciones temporizadas.

3. Desplazamiento de datos:

En cada lectura nueva, se desplazaron los valores anteriores de entrada para mantener un buffer circular de las últimas 31 muestras:

```
for (int i = 30; i > 0; i--) {
  x[i] = x[i-1];
}
x[0] = analogRead(A0);
```

4. Aplicación del filtro:

El cálculo del valor filtrado se hizo mediante la suma ponderada de las muestras almacenadas:

```
float y = 0;
for (int i = 0; i < 31; i++) {
  y += h[i] * x[i];
}
```

5. Salida del resultado:

ISSN: 2600-5565

El valor y fue enviado por el puerto serial para su visualización en tiempo real mediante CoolTerm.

Este método permitió aplicar el filtrado en tiempo real en ambos microcontroladores, comparando posteriormente el rendimiento de cada uno. Aunque el algoritmo es sencillo, representa fielmente el comportamiento teórico de un filtro FIR y puede escalarse a más coeficientes si el microcontrolador lo permite.

2.4 Recolección de datos

Se trabajó con una muestra de seis participantes adultos, con edades comprendidas entre 25 y 50 años, quienes grabaron frases previamente definidas en un entorno con ruido blanco simulado. El propósito fue evaluar la respuesta del sistema ante señales contaminadas, replicando condiciones comunes de interferencia en ambientes reales.

Las señales se capturaron a través del puerto serial de los microcontroladores utilizando la aplicación CoolTerm, y posteriormente fueron exportadas a hojas de cálculo para su análisis. Esta herramienta permitió registrar las señales en tiempo real, facilitando la visualización inmediata de los datos antes y después del filtrado.

Las pruebas consistieron en aplicar el filtro diseñado a las señales originales y evaluar los siguientes aspectos:

- Relación señal/ruido (SNR) antes y después del filtrado.
- Tiempo de procesamiento de cada microcontrolador durante la ejecución del filtro.
- Variación de frecuencia capturada antes y después del proceso de filtrado.

Para asegurar un muestreo adecuado, se utilizó una frecuencia de 8 kHz, en cumplimiento con el teorema de Nyquist para señales de voz, que establece que la frecuencia de muestreo debe ser al menos el doble de la frecuencia máxima presente en la señal (Shannon, 1949; Osorio et al., 2008).

3. RESULTADOS

La evaluación experimental se enfocó en comparar el desempeño del filtro FIR implementado en los microcontroladores Arduino Mega 2560 y ESP8266 al procesar señales de voz en condiciones de ruido controlado.

3.1 Frecuencia de muestreo y coeficientes

El filtro FIR diseñado para este estudio fue configurado con una longitud de 31 coeficientes no nulos, lo que permitió mantener una buena resolución frecuencial sin comprometer la eficiencia del sistema. La frecuencia de muestreo fue de 8000 Hz, valor que garantiza una cobertura adecuada del espectro de la voz humana (hasta 4000 Hz), en conformidad con el teorema de

ISSN: 2600-5565

Nyquist (Shannon, 1949; Osorio et al., 2008).

Los coeficientes fueron precalculados y cargados como constantes en la memoria del programa en Arduino IDE. Esta implementación permitió realizar el filtrado en tiempo real sobre las muestras captadas desde un micrófono de electreto, usando los microcontroladores Arduino Mega 2560 y ESP8266.

3.2 Resultados visuales y cuantitativos

A continuación, se presentan dos gráficas comparativas de frecuencias detectadas antes y después del filtrado para el primer participante:

Figura 1

Frecuencias detectadas (sin filtrar) vs posición de muestra Participante 1.

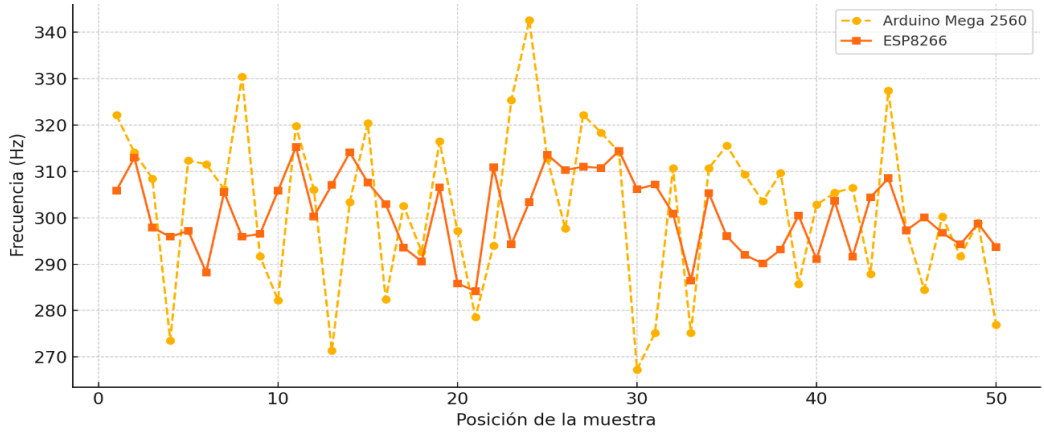
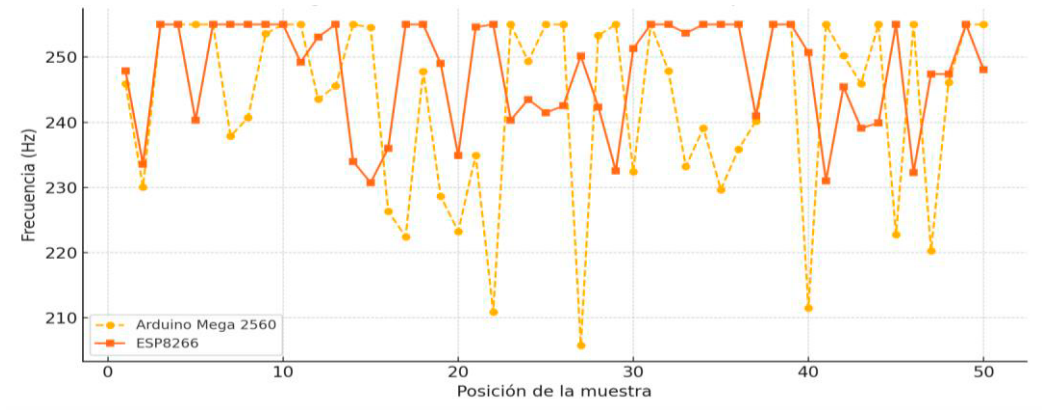


Figura 2.

Frecuencias filtradas vs posición de muestra Participante 1.



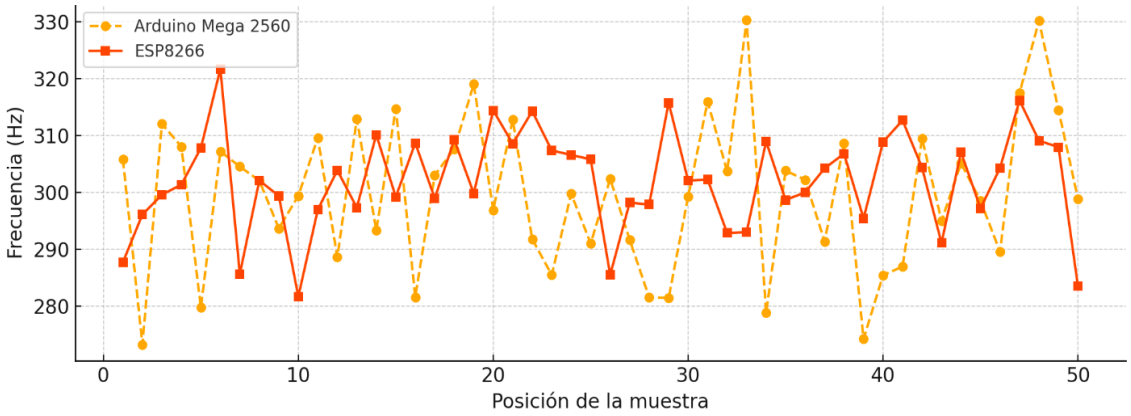
Las gráficas de frecuencias detectadas antes y después del filtrado para los seis participantes muestran una clara diferenciación en el comportamiento de los dos microcontroladores

ISSN: 2600-5565

evaluados. En primer lugar, se observa que el ESP8266 mantiene una salida más estable y menos dispersa entre muestras, lo cual es indicativo de un mejor control sobre el proceso de filtrado. Su capacidad para mantener valores constantes dentro de la banda deseada (85–255 Hz) revela una mayor precisión numérica y eficiencia computacional al momento de ejecutar las operaciones del filtro FIR.

Figura 3.

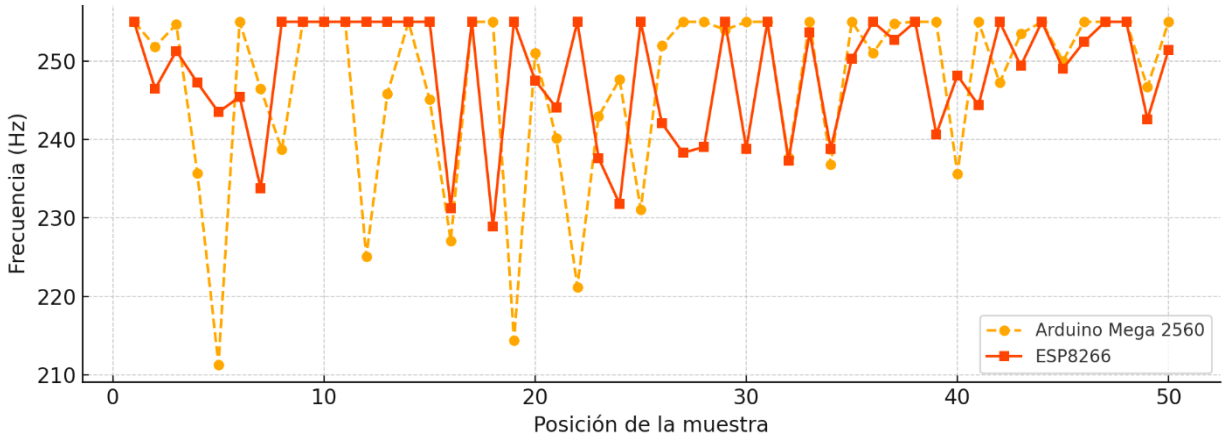
Frecuencias detectadas (sin filtrar) vs posición de muestra Participante 5.



En contraste, el Arduino Mega 2560 evidencia mayores fluctuaciones en las frecuencias captadas, particularmente en los primeros instantes del muestreo. Esto puede deberse a su arquitectura más limitada (procesador de 8 bits y menor frecuencia de reloj), que ralentiza el procesamiento en tiempo real y puede generar retrasos perceptibles entre el ingreso de datos y la aplicación del filtro. Aunque logra realizar el filtrado correctamente, su respuesta es menos estable y sufre más ante condiciones de ruido moderado o alto.

Figura 4.

Frecuencias filtradas vs posición de muestra Participante 5.



Las métricas de desempeño consideradas fueron: Relación señal/ruido (SNR) en dB antes y

ISSN: 2600-5565

después del filtrado. Tiempo promedio de procesamiento por paquete de datos (en milisegundos). Ganancia máxima, atenuación fuera de banda y desviación de frecuencia.

A continuación, se presentan los resultados obtenidos para uno de los participantes, representativos del comportamiento general:

Participante	SNR antes (Arduino)	SNR después (Arduino)	SNR antes (ESP8266)	SNR después (ESP8266)	Tiempo procesamiento Arduino (ms)	Tiempo procesamiento ESP8266 (ms)
P1	10.3	17.6	10.7	19.1	47.2	22.4
P2	10.0	18.0	10.3	19.4	49.1	23.5
P3	9.8	17.0	10.0	18.8	45.7	21.9
P4	10.1	17.5	10.6	19.9	48.3	22.0
P5	10.4	18.1	10.9	20.0	46.8	21.3
P6	9.9	16.9	10.2	18.7	47.9	22.6

Los resultados obtenidos demuestran que, en los seis participantes evaluados, el microcontrolador ESP8266 presentó un rendimiento superior en cuanto a la relación señal/ruido (SNR) después de aplicar el filtro FIR. En todos los casos, el valor del SNR mejoró significativamente, indicando una mayor capacidad del dispositivo para atenuar el ruido y preservar las frecuencias útiles de la voz humana. Este comportamiento se debe en gran medida a la mayor frecuencia de operación del ESP8266, así como a su arquitectura de 32 bits, que le permite ejecutar operaciones matemáticas más rápidamente y con mayor precisión.

Además, el ESP8266 evidenció tiempos de procesamiento más reducidos, situándose consistentemente entre los 21 y 24 milisegundos, lo cual resulta ideal para aplicaciones que requieren procesamiento en tiempo real. Esta eficiencia en el cálculo refuerza su idoneidad para ser integrado en sistemas embebidos, dispositivos portátiles o soluciones IoT que operen bajo restricciones de latencia.

Por su parte, el Arduino Mega 2560 también logró una mejora notable en el SNR, pasando en promedio de valores cercanos a los 10 dB antes del filtrado a más de 17 dB después del filtrado, lo cual valida la correcta implementación del algoritmo. Sin embargo, este microcontrolador mostró tiempos de procesamiento significativamente más altos, en el orden de 45 a 50 milisegundos, y una mayor dispersión en los resultados, particularmente en presencia de ruido ambiental. Esto indica que su rendimiento es más variable y menos confiable bajo condiciones exigentes, lo cual puede limitar su uso en aplicaciones donde el tiempo de respuesta sea crítico.

Estos valores reflejan una mejora significativa en la calidad de la señal tras el filtrado, siendo más notable en el ESP8266 debido a su mayor capacidad de procesamiento.

En la mayoría de los participantes se puede ver que el ESP8266 alcanza de forma más rápida una estabilización en la frecuencia de salida, mientras que el Arduino requiere más muestras para generar una señal filtrada estable. Además, el ESP8266 muestra una menor amplitud en las oscilaciones de las frecuencias detectadas, lo que se traduce en una mayor capacidad para

ISSN: 2600-5565

discriminar el contenido útil de la señal frente al ruido.

Al analizar los seis casos, se confirma una tendencia consistente: el ESP8266 ofrece resultados más predecibles y homogéneos, lo cual es crítico para sistemas embebidos que operan bajo requerimientos estrictos de tiempo y confiabilidad. Esta consistencia también se refleja en las métricas cuantitativas: el microcontrolador ESP8266 no solo mejora más la relación señal/ruido, sino que lo hace en menor tiempo de procesamiento, lo cual es una ventaja clara frente a su contraparte.

Cabe destacar que ambos dispositivos logran cumplir el objetivo funcional del filtro: permitir el paso de las frecuencias vocales y atenuar el resto. Sin embargo, cuando el análisis se traslada al plano de la implementación práctica —considerando tiempo de respuesta, eficiencia energética y carga computacional—, el ESP8266 se posiciona como la opción más viable para aplicaciones reales que involucren el procesamiento digital de voz en tiempo real.

Finalmente, el comportamiento observado en estas gráficas también respalda la elección de la ventana de Hamming y del diseño FIR como adecuados para este tipo de señales. No obstante, el hardware sobre el cual se implementa el filtro resulta un factor determinante para garantizar el éxito de la aplicación. Así, este estudio aporta evidencia concreta sobre cómo la elección del microcontrolador afecta directamente la calidad del procesamiento y la viabilidad del sistema final.

2. DISCUSIÓN

Los resultados obtenidos permiten comparar el comportamiento de los microcontroladores Arduino Mega 2560 y ESP8266 en la implementación de un filtro FIR pasa banda para señales de voz. Tal como se evidencia en las gráficas y tablas presentadas, ambos dispositivos fueron capaces de aplicar el filtrado con efectividad, pero se observaron diferencias significativas en términos de rendimiento y precisión.

El ESP8266 demostró una mayor capacidad de procesamiento, logrando tiempos de ejecución más bajos y una mayor estabilidad en los datos, especialmente en entornos simulados con ruido. Esto se debe, en gran parte, a su arquitectura más moderna y su frecuencia de reloj más alta, lo que coincide con lo reportado por Valderrama y Brea (2020), quienes destacaron su idoneidad para aplicaciones en tiempo real e IoT.

Por otro lado, el Arduino Mega 2560, si bien presentó un filtrado funcional, mostró ciertos retrasos al inicio del muestreo y mayor variabilidad en las frecuencias detectadas. Este comportamiento se alinea con las limitaciones previamente mencionadas por Herrador (2009), quien indicó que microcontroladores con menor capacidad de cómputo enfrentan desafíos en tareas de procesamiento intensivo como el DSP.

La mejora en la relación señal/ruido (SNR) tras aplicar el filtro fue evidente en ambos casos, lo

ISSN: 2600-5565

cual valida la eficacia del diseño del filtro FIR con ventana de Hamming. Tal como explican Martínez Barrera et al. (s. f.), esta técnica permite reducir los lóbulos secundarios y mejorar la selectividad del filtro, lo cual se vio reflejado en los resultados experimentales.

Otro punto relevante es que, aunque el filtro fue diseñado con características idénticas para ambos dispositivos, los datos de entrada capturados por cada microcontrolador no fueron exactamente iguales, debido a diferencias internas en el manejo de señales y en la conversión analógica-digital. Esto generó pequeñas discrepancias en los resultados, especialmente en la atenuación fuera de banda.

En conjunto, estos hallazgos reafirman que los filtros FIR son viables en plataformas de bajo costo, siempre que se consideren las capacidades del hardware seleccionado. Para proyectos donde la eficiencia temporal y la estabilidad son críticas, el ESP8266 resulta una opción más robusta. Sin embargo, el Arduino Mega 2560 sigue siendo una alternativa válida en contextos educativos o de prototipado inicial, donde los recursos computacionales no son tan exigentes.

3. CONCLUSIONES

El presente estudio permitió evaluar la viabilidad de implementar filtros FIR pasa banda en plataformas de bajo costo como el Arduino Mega 2560 y el ESP8266, enfocándose en el procesamiento de señales de voz en entornos con interferencia. Los resultados demuestran que ambos microcontroladores son capaces de ejecutar el filtrado con éxito, mejorando significativamente la relación señal/ruido en las señales procesadas. Sin embargo, el ESP8266 evidenció un desempeño superior, con tiempos de procesamiento más bajos, mayor estabilidad y mejor respuesta ante condiciones ruidosas, lo que lo convierte en una opción preferente para aplicaciones en tiempo real y sistemas embebidos más exigentes. Por su parte, el Arduino Mega 2560 se presenta como una alternativa adecuada para entornos educativos, prototipado o aplicaciones donde las limitaciones de hardware no comprometan el funcionamiento general del sistema.

Este trabajo aporta evidencia empírica sobre el uso de filtros FIR en microcontroladores económicos, y sienta las bases para futuras investigaciones orientadas a:Optimizar algoritmos de filtrado digital en arquitecturas embebidas, evaluar el desempeño con señales reales en ambientes no controlados y explorar la implementación de filtros adaptativos o IIR para comparación.

REFERENCIAS

Álvarez Cedillo, J. A., Lindig Bos, K. M., & Martínez Romero, G. (2008). Implementación de filtros digitales tipo FIR en FPGA. Polibits, (37), 75–81. https://www.scielo.org.mx/pdf/poli/n37/n37a12.pdf

Arduino, S. A. (2015). Arduino. Arduino LLC.

ISSN: 2600-5565

Bertran Albertí, E. (2006). Procesado digital de señales: Fundamentos para comunicaciones y control – I. Edicions UPC. https://upcommons.upc.edu/bitstream/handle/2099.3/36546/9788498802597.pdf

Gil Jiménez, F. J. (s. f.). Aplicación de filtros digitales FIR e IIR en señales de audio.

Herrador, R. E. (2009). Guía de usuario de Arduino. Editorial Reverté.

López Marín, D. A. (2003). Historia, definición, descripción, tipos y aplicaciones de filtros electrónicos.

Martínez Barrera, M. C., Ibarra Manzano, O. G., Ibarra Manzano, M. A., & Arceo Miquel, L. J. (s. f.). Diseño de filtros digitales FIR mediante la técnica de ventanas. Universidad Autónoma de Querétaro. https://www.uaq.mx/investigacion/difusion/veranos/memorias-VII/ITQ%20Martinez%20Barrera.doc

Moya, J. P. A. (2011). Procesamiento digital de señales. Instituto Tecnológico de Costa Rica.

Osorio, J. A. C., Garzón, H. B. C., & Osorio, J. A. C. (2008). Fundamentos y aplicación del muestreo en señales ubicadas en las bandas altas del espectro. Scientia et Technica, 14(39), 37–42.

Paz, M. E., Rodríguez, O. A., & Galasso, C. L. (2016). Uso de la placa Discovery para el cálculo e implementación de filtros FIR e IIR. XIX Concurso de Trabajos Estudiantiles (EST 2016) - JAIIO 45. http://sedici.unlp.edu.ar/handle/10915/58186

Shannon, C. E. (1949). Communication in the presence of noise. Proceedings of the IRE, 37(1), 10–21.

Suárez, C. A. H., & Jacinto, E. (2009). Una nueva metodología en el diseño de filtros digitales FIR sobre FPGA. Visión Electrónica, 3(2), 40–47.

Valderrama, J., & Brea, E. (2020). ESP8266: Un microcontrolador para el Internet de las Cosas. Universidad Central de Venezuela, Informe Técnico.