UNIVERSIDAD PERUANA
CAYETANO HEREDIA

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

UNIVERSIDAD PERUANA CAYETANO HEREDIA

FACULTAD DE CIENCIAS E INGENIERÍA

EVALUACIÓN DE UNA LIBRERÍA DE FILTROS DIGITALES PARA EL PROCESAMIENTO DE BIOSEÑALES EN ARDUINO DUE

Tesis para optar por el título profesional de Ingeniero Biomédico

que presenta:

Sergio Enrique Moreno Elescano

Asesor:

Mg. Moises Stevend Meza Rodriguez

Lima, 2026

Jurado calificador

Presidente: Mg. Umbert Lewis De La Cruz Rodriquez

Vocal: Mg. Miguel Rogger Hoyos Alvitez

Secretario: Mg. Domingo Vladimir Flores Robles

UNIVERSIDAD PERUANA
CAYETANO HEREDIA

DECLARACIÓN DE ORIGINALIDAD

Los egresados:

N°	APELLIDOS Y NOMBRES
1.	ELESCANO MORENO SERGIO ENRIQUE

(Agregar filas adicionales si hay más autores)

Pertenecientes al programa de la **CARRERA PROFESIONAL DE INGENIERÍA BIOMÉDICA** autores del trabajo titulado: **EVALUACIÓN DE UNA LIBRERÍA DE FILTROS DIGITALES PARA EL PROCESAMIENTO DE BIOSEÑALES EN ARDUINO DUE**, el cual ha sido elaborado, sustentado y aprobado, según corresponda, para optar por el **TÍTULO PROFESIONAL DE INGENIERO BIOMÉDICO** bajo la modalidad de **TESIS.**

En calidad de docentes asesores de la Universidad Peruana Cayetano Heredia:

N°	APELLIDOS Y NOMBRES DEL DOCENTE	FACULTAD	NIVEL DE ASESORÍA
1.	MEZA RODRIGUEZ MOISES STEVEND	FACI	ASESOR

Declaramos que el contenido del presente documento es original y que las citas y referencias a otros autores cumplen con las normas académicas establecidas. En ese sentido, hacemos constar que:

- El documento presenta un porcentaje de similitud de **7%**, según el reporte emitido por el software **Turnitin®** (identificador de entrega: **3649097439**; fecha de entrega: **14-09-2026)**
- Tras una revisión detallada del reporte y del contenido del trabajo en cuestión, no se han identificado indicios de plagio.
- Se certifica que el documento respeta los principios de integridad académica y cumple con los requisitos institucionales de originalidad.

Lugar y fecha: **Lima, 14 de septiembre de 2026**

Firma del asesor
N° DNI: 47470482
ORCID: 0000-0002-5806-9014

Dedicatoria

A mi familia, por creer siempre en mí y por su apoyo incondicional para cumplir mis sueños y metas.

A mi asesor, por sus consejos y su guía profesional.

Agradecimientos

A Dios, por darme las fuerzas para cerrar esta etapa de mi formación.

A mi madre Gina Elescano, por darme siempre palabras de aliento y confianza para seguir adelante.

A mi padre Luis Moreno, por leer mis borradores y compartir mi entusiasmo por cada uno de mis proyectos.

A mi asesor Mag. Moisés Meza Rodriguez, por abrirme las puertas de su laboratorio y hacer siempre un espacio en su agenda para atender mis consultas.

ÍNDICE

Resumen 1

Abstract 3

I. Introducción 5

1.1 Marco Teórico 8

1.1.1 Procesamiento Digital de Señales 8

1.1.2 Diseño de Filtros 10

1.1.3 Parámetros importantes en el diseño de filtros 13

1.1.4 Filtros Wavelet 14

1.1.5 Filtros Adaptativos 15

1.1.6 Bioseñales 16

1.1.6.1 Electrocardiografía (ECG) 16

1.1.6.2 Electromiografía (EMG) 18

1.1.6.3 Electroencefalografía (EEG) 19

1.1.7 Hardware Embebido para DSP 20

1.2 Estado del Arte 25

1.2.1 Librerías de DSP en Hardware Embebido 25

1.2.2 Librerías DSP para Arduino en educación 31

1.3 Problemática 33

1.4. Justificación 34

II. Hipótesis y Objetivos 35

2.1. Objetivo general 35

2.2. Objetivos específicos 35

2.3. Hipótesis general 35

III. Metodología 36

3.1. Diseño del proyecto 36

3.1.1 Arquitectura de la librería 39
3.1.2 Diseño de la documentación 43
3.1.3 Consideraciones de rendimiento y optimización 44
3.2. Análisis comparativo 45
3.2.1 Generación y síntesis de las señales 46
3.2.2. Librerías y filtros evaluados 49
3.2.3. Métricas de evaluación 52
3.3. Operacionalización de las variables 55
IV. Resultados 59
4.1 Escenario A – ECG con interferencia de 60 Hz (notch) 59
4.1.1 Formas de onda y espectros de magnitud - Escenario A 60
4.2 Escenario B – EEG con ruido rosa (pasa-altos) 63
4.2.1 Formas de onda y espectros de magnitud – Escenario B 63
4.3 Escenario C — EMG con ruido blanco (pasa-banda) 67
4.3.1 Formas de onda y espectros de magnitud - Escenario C 67
V. Discusión 71
5.1 Análisis de rendimiento computacional y viabilidad técnica 72
5.2 Comparación con librerías existentes para el ecosistema Arduino 73
5.3 Calidad de filtrado y preservación morfológica de bioseñales 74
5.4 Rangos de operación y frecuencias máximas 76
5.4.1 Frecuencia máxima de señal. 76
5.4.2 Frecuencia de muestreo máxima sostenible 76
5.5 Validación de la hipótesis y cumplimiento de objetivos 77
5.6 Implicancias para la enseñanza de DSP 78
5.7 Limitaciones del estudio 79
VI. Conclusiones 81
Bibliografía 83

Índice de Tablas

Tabla 1. Principales bibliotecas de DSP para hardware embebido

Tabla 2. Señales disponibles en la librería BioFilterLib

Tabla 3. Filtros y parámetros de configuración por escenario

Tabla 4. Operacionalización de las variables

Tabla 5. Resultados del Escenario A (ECG + 60 Hz, fs = 960 Hz, SNR de entrada ≈ 5 dB)

Tabla 6. Resultados del Escenario B (EEG + ruido rosa, fs = 960 Hz, SNR de entrada ≈ 5 dB)

Tabla 7. Resultados del Escenario C (EMG + ruido blanco, fs = 960 Hz, SNR de entrada ≈ 5 dB)

Tabla 8. Tiempo de procesamiento, frecuencia máxima en tiempo real y carga de CPU por tipo de filtro (fs = 960 Hz)

Índice de Figuras

Figura 1. Derivaciones de Einthoven. Fuente: (1)

Figura 2. Obtención y descomposición de señal EMG en unidades motoras individuales. Fuente: Khan Academy

Figura 3. Colocación de electrodos según el sistema internacional 10-20 para adquisición de señal EEG. Fuente: (2)

Figura 4. Comparativa de las familias ARM Cortex: Series A, R y M. Fuente: Elaboración propia

Figura 5. Microcontrolador Arduino Due con un CPU Cortex-M3. Fuente: Arduino Store

Figura 6. Experimento de muestreo y generación de señales en tiempo real con Arduino Due. Fuente: Elaboración propia

Figura 7. Componentes del proyecto. Fuente: Elaboración propia

Figura 8. Directorios en GitHub de la librería. Fuente: Elaboración propia

Figura 9. Diagrama UML clases principales de BioFilterLib. Fuente: Elaboración propia

Figura 10. Página principal para filtro FIR en la documentación. Fuente: https://sergiomoreno1060.github.io/BioFilterLib/

Figura 11. Ejemplo de uso de filtro FIR en la documentación. Fuente: https://sergiomoreno1060.github.io/BioFilterLib/

Figura 12. Preprocesamiento de la señal de prueba. Fuente: Elaboración propia

Figura 13. Escenario A — ECG, filtros FIR: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

Figura 14. Escenario A — ECG, filtros FIR: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

Figura 15. Escenario A — ECG, filtros IIR: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

Figura 16. Escenario A — ECG, filtros IIR: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

Figura 17. Escenario A — ECG, filtros Wavelet: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

Figura 18. Escenario A — ECG, filtros Wavelet: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

Figura 19. Escenario A — ECG, filtros NLMS adaptativo: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

Figura 20. Escenario A — ECG, filtros NLMS adaptativo: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

Figura 21. Escenario B — EEG, filtros FIR: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

Figura 22. Escenario B — EEG, filtros FIR: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

Figura 23. Escenario B — EEG, filtros IIR: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

Figura 24. Escenario B — EEG, filtros IIR: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

Figura 25. Escenario B — EEG, filtros Wavelet: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

Figura 26. Escenario B — EEG, filtros Wavelet: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

Figura 27. Escenario B — EEG, filtros NLMS adaptativo: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

Figura 28. Escenario B — EEG, filtros NLMS adaptativo: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

Figura 29. Escenario C — EMG, filtros FIR: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

Figura 30. Escenario C — EMG, filtros FIR: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

Figura 31. Escenario C — EMG, filtros IIR: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

Figura 32. Escenario C — EMG, filtros IIR: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

Figura 33. Escenario C — EMG, filtros Wavelet: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

Figura 34. Escenario C — EMG, filtros Wavelet: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

Figura 35. Escenario C — EMG, filtros NLMS adaptativo: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

Figura 36. Escenario C — EMG, filtros NLMS adaptativo: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

Resumen

La enseñanza del procesamiento digital de señales biomédicas requiere herramientas que equilibren accesibilidad, capacidad técnica y facilidad de uso en contextos educativos. Este proyecto desarrolló y evaluó BioFilterLib, una librería de código abierto para filtrado digital de bioseñales en Arduino Due, optimizada mediante CMSIS-DSP y diseñada para aplicaciones educativas, donde no solo se limita a la simulación, sino que permite la implementación de filtros en hardware como Arduino Due para aplicaciones en tiempo real. Además, BioFilterLib incluye una documentación que facilita la experimentación práctica en hardware.

El objetivo fue desarrollar y evaluar BioFilterLib, analizar su factibilidad técnica en la tarea de procesamiento de bioseñales y comparar su desempeño respecto a otras librerías disponibles en Arduino. Se implementó una arquitectura modular que integra filtros FIR, IIR, adaptativos NLMS y wavelet Daubechies. La evaluación se realizó sobre tres escenarios con bioseñales sintéticas y con una relación señal-ruido de entrada fijada en 5 dB (fs = 960 Hz): ECG con interferencia de 60 Hz, EEG con ruido rosa y EMG con ruido blanco. El desempeño se comparó frente a CMSIS-DSP evaluada directamente, con el objetivo de cuantificar el sobrecosto de la capa de abstracción, y frente a cuatro librerías FIR de terceros del ecosistema Arduino, recopilando métricas de tiempo de procesamiento, uso de CPU, memoria y calidad de filtrado (en función de mejora de SNR, MSE, correlación y RMS).

Los resultados evidenciaron que los filtros IIR alcanzaron la mayor eficiencia computacional (velocidad de procesamiento de 8.40-18.27 µs/muestra y uso de CPU inferior al 2 %), mientras que los FIR resultaron más exigentes computacionalmente (209-416 µs/muestra, 20-40 % de CPU). El sobrecosto de la capa de abstracción de BioFilterLib respecto a CMSIS-DSP fue despreciable, con métricas de calidad idénticas, y su procesamiento por bloques resultó más rápido que las librerías de terceros evaluadas muestra a muestra. La eficacia del filtrado fue variable dependiendo del tipo de ruido: la interferencia de banda estrecha de 60 Hz se atenuó eficazmente (mejoras de SNR de hasta 26,71 dB, correlaciones superiores a 0,98), pero el ruido de banda ancha solapado con la

señal limitó al filtrado lineal (0,07-2,56 dB), siendo el filtro adaptativo NLMS el único que logró mejoras apreciables (8.18 dB en EEG y 5.22 dB en EMG).

Estos resultados confirman un desempeño superior o equivalente a otras librerías para Arduino. El bajo costo de Arduino Due (US$30-50) en comparación con plataformas especializadas (US$150-600) permite una mayor accesibilidad para su uso para entornos educativos.

Palabras clave: Procesamiento digital de señales, filtrado digital, bioseñales, Arduino Due

Abstract

Teaching digital signal processing of biomedical signals requires tools balancing accessibility, technical capability, and ease of use in educational contexts. This project developed and evaluated BioFilterLib, an open-source library for digital filtering of biosignals on Arduino Due, optimized through CMSIS-DSP library and designed for educational applications. The library is not limited to simulation, but enables real-time filter implementation on hardware platforms such as Arduino Due. In addition, BioFilterLib includes documentation that facilitates hands-on experimentation on hardware.

The objective was to develop and evaluate BioFilterLib, assess its technical feasibility for biosignal processing, and compare its performance against other Arduino libraries. A modular architecture was implemented integrating FIR, IIR, NLMS adaptive, and Daubechies wavelet filters. Evaluation was carried out over three synthetic biosignal scenarios with an input signal-to-noise ratio fixed at 5 dB (fs = 960 Hz): ECG with 60 Hz interference, EEG with pink noise, and EMG with white noise. Performance was compared against CMSIS-DSP evaluated directly, in order to quantify the overhead of the abstraction layer, and against four third-party Arduino FIR libraries, collecting processing time, CPU usage, memory, and filtering quality metrics (evaluated through SNR improvement, MSE, correlation with clean signal of reference, and RMS).

Results showed that IIR filters achieved the highest computational efficiency (8.40-18.27 µs/sample, CPU usage below 2 %), whereas FIR filters were more demanding (209-416 µs/sample, 20-40 % CPU). The overhead of BioFilterLib's abstraction layer relative to CMSIS-DSP was negligible, with identical quality metrics, and its block-based processing proved faster than the third-party libraries evaluated sample by sample. Filtering efficacy varied across signals with different types of added noise: the narrowband 60 Hz interference was effectively attenuated (SNR improvements up to 26.71 dB, correlations above 0.98), whereas broadband noise overlapping the signal band limited linear filtering (0.07-2.56 dB), with the NLMS adaptive filter being the only method that achieved appreciable improvements (8.18 dB for EEG and 5.22 dB for EMG).

These results confirm the library's superior or equivalent performance to other Arduino libraries. Arduino Due's low cost (US$30-50) versus specialized platforms (US$150-600) enables greater accessibility for its use in educational environments.

Keywords: Digital signal processing, digital filtering, biosignals, Arduino Due

I. Introducción

En el contexto académico, la inclusión de tecnologías para la enseñanza universitaria y la investigación cobra importancia en la formación de profesionales en las áreas de ciencia y tecnología. La rápida evolución de la tecnología ha traído herramientas y recursos digitales con un potencial significativo para transformar la dinámica tradicional de enseñanza en carreras como ingeniería (3). En particular, la enseñanza del procesamiento de bioseñales en hardware embebido representa un componente de alto valor para estudiantes de ingeniería biomédica, electrónica o mecatrónica (4).

Dentro del procesamiento digital de señales (DSP por sus siglas en inglés), se abordan una amplia gama de temas, desde los fundamentos teóricos de señales y sistemas, principios de muestreo, hasta aplicaciones específicas como algoritmos de filtrado (5). Estos temas aplicados a tratar bioseñales, son de valor para los estudiantes de ingeniería biomédica, ya que proporcionan conocimientos teóricos esenciales para el procesamiento de bioseñales. En este sentido, la interacción práctica con hardware complementa la teoría, aumentando el interés y compromiso de los estudiantes (6).

Las metodologías de enseñanza actuales utilizan herramientas de software como MATLAB y Python para la simulación y experimentación en laboratorios de DSP (7,8); estas metodologías se ven complementadas con hardware para aplicaciones en tiempo real usando sistemas embebidos cuyo microcontrolador es un procesador digital de señales. Estos procesadores, pueden ser complejos en un ambiente educativo, ya que requieren un curva de aprendizaje mayor donde se requiere cursos complementarios previos relacionados a arquitectura de computadores o lenguaje assembler (9). Las tarjetas embebidas de las marcas Arduino, STM Electronics y Texas Instruments ofrecen alternativas accesibles. Particularmente, las tarjetas de la marca Arduino (10) ofrecen ventajas en cuanto a su facilidad de programación, documentación adecuada, comunidad activa, bajo costo y facilidad de uso en entornos educativos, favoreciendo su adopción en proyectos que incluyan DSP.

Sin embargo, persiste una limitación significativa, la falta de una librería que recopile las funciones principales de DSP, que cuente con documentación y tutoriales demostrativos y que permita a estudiantes con conocimientos básicos en programación implementar algoritmos de filtrado biomédico de forma eficiente mediante el uso de software

optimizado para microcontroladores. Los estudiantes principiantes enfrentan dificultades considerables al trabajar con librerías de programación complejas y herramientas técnicas que no fueron diseñadas con propósitos educativos (11). Esta problemática se intensifica en el contexto de sistemas embebidos (12), donde los currículos educativos separan tradicionalmente la teoría de DSP de su implementación en hardware. Esta situación ofrece pocas oportunidades para que los estudiantes integren esta teoría con un conocimiento práctico de diseño de sistemas DSP embebidos (13). Esta falta adquiere particular relevancia para aquellos estudiantes que cursan asignaturas introductorias y se inician en el campo del procesamiento de bioseñales y el prototipado biomédico, disciplina caracterizada por su alta exigencia conceptual (14).

La librería CMSIS (Estándar de Interfaz de Software para Microcontroladores Cortex, por sus siglas en inglés) (15) es un conjunto de funciones altamente optimizadas para su uso en filtrado de señales con microcontroladores como los que son parte de las placas Arduino Due. Esta librería presenta una complejidad de código moderada-alta, gran cantidad de funciones, y una documentación orientada a usuarios avanzados que no se enfoca en ejemplos prácticos de uso, características que representan barreras para estudiantes principiantes en entornos educativos introductorios (11,12). Algunos desarrolladores independientes han creado alternativas de manera aislada, por ejemplo FIR_FILTER_ARDUINO_LIBRARY ha desarrollado algunas funciones en C++ que están listas para descargar y usar en el mismo entorno de desarrollo integrado (IDE) de Arduino (16), sin embargo el código no ha sido actualizado para aprovechar el manejo de memoria que sí ofrece CMSIS-DSP.

Frente a esta necesidad, este trabajo propone BioFilterLib, una librería educativa de código abierto para Arduino Due, diseñada para simplificar la implementación y evaluación de algoritmos de filtrado digital de bioseñales. Esta librería proporciona una colección de funciones accesibles para realizar filtrado, así como herramientas complementarias para generación de señales y ejemplos en la documentación. A diferencia de los frameworks tradicionales, BioFilterLib simplifica el uso de optimizaciones SIMD (Single Instruction Multiple Data (17)) con el objetivo de reducir la complejidad de uso y mejorar la eficiencia en entornos educativos con procesamiento de cierta cantidad de datos en paralelo.

A través de un enfoque centrado en la experiencia educativa, BioFilterLib brinda ejemplos de uso, documentación accesible y funciones optimizadas para trabajar con bioseñales comunes en instrumentación biomédica. En los capítulos siguientes, se procede a detallar la metodología de desarrollo de la librería, se presentan los resultados de su implementación, y se analiza su rendimiento frente a alternativas existentes, con enfoque en su factibilidad técnica y potencial impacto en la enseñanza de DSP para entornos universitarios. Finalmente, se plantean futuras líneas de desarrollo orientadas a ampliar las capacidades de la librería y su adaptación a otras plataformas de hardware.

1.1 Marco Teórico

1.1.1 Procesamiento Digital de Señales

1.1.1.1 Definición y principios básicos

El Procesamiento Digital de Señales es un campo de la ingeniería que se centra en la representación, análisis, transformación y manipulación de señales físicas en formato digital mediante el uso de operaciones matemáticas y algoritmos implementados en sistemas de hardware o software. A diferencia del procesamiento analógico tradicional, el DSP permite realizar operaciones complejas con mayor precisión, reproducibilidad y flexibilidad, al trabajar con representaciones discretas de señales continuas o analógicas (18).

Un sistema DSP se caracteriza por su capacidad para transformar una secuencia de números de entrada en otra secuencia de salida mediante operaciones matemáticas bien definidas. Estas operaciones pueden ser representadas mediante ecuaciones en diferencias, funciones de transferencia en el dominio z, o respuestas al impulso, constituyendo así el corazón teórico del procesamiento digital.

La teoría del DSP se basa en conceptos matemáticos como la transformada Z, que extiende el uso de la transformada de Laplace al dominio discreto, y la transformada discreta de Fourier (DFT), que permite analizar el contenido de frecuencias de señales discretas. La ecuación fundamental de la DFT para una secuencia discreta x[n] de longitud N se expresa como en la Ecuación 1:

$$X[k] = \sum_{n=0}^{N-1} \quad x[n] \cdot x^{-j2\pi\frac{kn}{N}}, k = 0,1,\ldots,N-1 \quad (1)$$

Esta transformación matemática presenta propiedades muy útiles para el análisis espectral, diseño de filtros y numerosas aplicaciones de procesamiento de señal en el dominio de la frecuencia (18).

1.1.1.2 Muestreo, cuantificación y reconstrucción

Para procesar digitalmente una señal analógica, es necesario convertirla a un formato digital mediante un proceso de tres etapas: muestreo, cuantificación y codificación.

El muestreo consiste en tomar valores discretos de una señal continua a intervalos regulares, generando una representación discreta en el tiempo. Este proceso se rige por el teorema Nyquist-Shannon (Ecuación 2), que establece que para reconstruir exactamente una señal limitada en banda, la frecuencia de muestreo debe ser al menos el doble de la frecuencia máxima presente en la señal.

$$f_s \geq 2 \cdot f_{max} \quad (2)$$

Donde f_s es la frecuencia de muestreo y f_{max} es la frecuencia máxima contenida en la señal. Este principio previene el efecto de aliasing, un fenómeno de distorsión que ocurre cuando se incumple el criterio de Nyquist y que resulta en la representación incorrecta de componentes de alta frecuencia (19).

La cuantificación, segunda etapa del proceso, consiste en asignar valores discretos a las amplitudes de la señal muestreada. Este proceso introduce un error inherente conocido como error de cuantificación o ruido de cuantificación, cuya magnitud depende directamente de la resolución del conversor analógico-digital (ADC) expresada en bits. Un ADC de n bits puede representar 2^n niveles diferentes de amplitud, siendo el error máximo de cuantificación igual a $\pm\frac{1}{2}LSB$ (Least Significant Bit) (20).

Finalmente, la reconstrucción permite convertir la señal procesada nuevamente al dominio analógico. Los métodos más comunes incluyen la retención de orden cero (ZOH), que mantiene cada valor digital constante durante un período de muestreo, y la interpolación lineal, que conecta los valores discretos mediante segmentos lineales. En sistemas de mayor precisión, se implementan filtros de reconstrucción basados en la función *sinc* y permiten una reconstrucción perfecta bajo condiciones ideales (21).

1.1.1.3 Aplicaciones generales y aplicaciones biomédicas

El procesamiento digital de señales tiene aplicación en numerosas áreas de la ingeniería. En telecomunicaciones, es fundamental para la modulación, demodulación, ecualización y compresión de datos. En sistemas multimedia, permite la codificación y procesamiento eficiente de audio e imágenes. El control automático, radar, sonar, sismología y geofísica son otros campos donde el DSP ha demostrado su utilidad (22).

En el ámbito biomédico, el DSP ha sido clave tanto en la instrumentación clínica como en la investigación. Rangayyan (23) destaca que el procesamiento digital de bioseñales ha mejorado en gran medida el diagnóstico médico al permitir la detección temprana de anomalías antes imperceptibles mediante técnicas tradicionales de observación.

Las señales electrocardiográficas (ECG) se favorecen del filtrado digital para eliminar interferencias como el ruido de línea eléctrica (50-60 Hz), ruido muscular y variaciones de la línea base, mejorando así la precisión diagnóstica cardiológica (24). El procesamiento de electroencefalografía (EEG) permite identificar patrones específicos asociados con condiciones neurológicas, facilitando el diagnóstico de epilepsia, trastornos del sueño y otras condiciones cerebrales (25).

En electromiografía (EMG), las técnicas de DSP permiten analizar la actividad eléctrica muscular para evaluar neuropatías, miopatías y estudiar la biomecánica del movimiento humano. El procesamiento de señales EMG incluye eliminación de artefactos, análisis tiempo-frecuencia y extracción de características para clasificación (26).

Adicionalmente, el procesamiento de imágenes médicas como radiografías, resonancias magnéticas y tomografías computarizadas constituye un campo de aplicación de gran relevancia, donde algoritmos de filtrado, segmentación y reconocimiento de patrones contribuyen a mejorar la calidad diagnóstica (27).

1.1.2 Diseño de Filtros

Un filtro se define como un circuito o sistema capaz de dejar pasar selectivamente señales eléctricas dentro de un rango de frecuencias específico a partir de una señal de entrada

que consta de una combinación de diferentes frecuencias. Una aplicación común de los filtros se da en sistemas estéreo de alto rendimiento, donde ciertos rangos de frecuencias de audio necesitan ser amplificados o suprimidos para mejorar la calidad del audio o la eficiencia en términos de uso de energía (28).

En el contexto del procesamiento digital de señales, los filtros constituyen herramientas fundamentales para modificar el contenido espectral de señales discretas mediante operaciones matemáticas. Su implementación en hardware embebido permite el acondicionamiento en tiempo real de bioseñales para su posterior análisis o interpretación clínica (29).

Los filtros digitales se clasifican principalmente en dos categorías según su respuesta impulsional: filtros de respuesta impulsional finita (FIR, Finite Impulse Response) y filtros de respuesta impulsional infinita (IIR, Infinite Impulse Response).

1.1.2.1 Filtros FIR

Los filtros FIR se caracterizan por tener una respuesta impulsional de duración finita, es decir, su salida se computa exclusivamente a partir de valores actuales y pasados de la entrada, sin retroalimentación. La ecuación general mostrada en la Ecuación 3 que describe un filtro FIR de orden N es:

$$y[n] = \sum_{k=0}^{N} \quad b_k \cdot x[n-k] \quad (3)$$

Donde $y[n]$ representa la salida actual, $x[n-k]$ son las entradas pasadas, y b_k son los coeficientes del filtro.

Las principales ventajas de los filtros FIR incluyen:

- **Estabilidad garantizada**: Al no tener componentes de retroalimentación, son estables.
- **Fase lineal**: Se pueden diseñar filtros FIR con fase lineal, propiedad crítica en aplicaciones biomédicas donde la preservación de la morfología de la señal es esencial.

- **Simplicidad de implementación**: No requieren estados iniciales y son menos susceptibles a efectos de cuantificación.

Sin embargo, para eliminar completamente frecuencias de ruido específicas de una señal, los filtros FIR suelen requerir órdenes elevados, incrementando así la carga computacional y el uso de memoria (30).

1.1.2.2 Filtros IIR

Los filtros IIR emplean tanto valores de la entrada como valores previos de la salida, incorporando retroalimentación en su estructura. La forma general de un filtro IIR es la que se muestra en la Ecuación 4:

$$y[n] = \sum_{i=0}^{M} \quad b_i \cdot x[n-i] - \sum_{j=1}^{N} \quad a_j \cdot y[n-j] \text{ (4)}$$

Donde b_i y a_j son los coeficientes del filtro, mientras que $y[n-j]$ corresponde a muestras pasadas de la salida.

Las ventajas de los filtros IIR incluyen:

- **Eficiencia computacional**: Requieren órdenes menores que los filtros FIR para lograr características de filtrado similares.
- **Analogía con filtros analógicos**: Permiten adaptar diseños analógicos consolidados mediante técnicas como la transformación bilineal.
- **Menor uso de memoria**: Al necesitar menos coeficientes, requieren menos almacenamiento.

Sus principales desventajas son la posibilidad de inestabilidad y la imposibilidad de conseguir fase lineal exacta, lo que puede resultar en distorsión de la morfología de bioseñales.

En aplicaciones biomédicas, la elección entre FIR e IIR debe considerar requisitos como la preservación morfológica, latencia permisible y recursos computacionales disponibles (31).

1.1.3 Parámetros importantes en el diseño de filtros

El diseño adecuado de filtros digitales para aplicaciones biomédicas requiere la especificación de diversos parámetros que definen su comportamiento en frecuencia:

1.1.3.1 Frecuencia de corte

Define el límite entre la banda de paso y la banda de transición. En filtros paso-bajo, representa la frecuencia a la cual la ganancia se reduce en 3 dB respecto a la ganancia de la banda de paso. Su selección adecuada es crucial en el procesamiento de bioseñales para preservar componentes significativos mientras se atenúan frecuencias no deseadas (32).

1.1.3.2 Ancho de banda de transición

Es el rango de frecuencias entre la banda de paso y la banda de rechazo. Un ancho de transición menor implica una respuesta más aguda del filtro, pero generalmente requiere órdenes más elevados, incrementando la carga computacional. En hardware embebido con recursos limitados, este parámetro adquiere especial relevancia, siendo necesario encontrar un compromiso entre la agudeza de respuesta y la eficiencia (33).

1.1.3.3 Rizado (Ripple)

Se refiere a la variación de amplitud en la banda de paso o de rechazo:

- **Rizado en banda de paso**: Fluctuaciones de amplitud en la banda de paso, expresadas generalmente en decibelios (dB). En aplicaciones como electrocardiografía, un rizado excesivo puede alterar la morfología de ondas críticas para el diagnóstico (24).

- **Rizado en banda de rechazo**: Variaciones de atenuación en la banda de rechazo. Sus efectos son generalmente menos críticos que los del rizado en banda de paso, pero valores insuficientes pueden permitir interferencias significativas en la señal procesada.

1.1.3.4 Atenuación en banda de rechazo

Indica la reducción mínima de amplitud que el filtro proporciona en la banda de rechazo, expresada habitualmente en decibelios (dB). Este parámetro determina la eficacia del filtro para suprimir interferencias y ruido. En aplicaciones biomédicas, donde las interferencias electromagnéticas son comunes, se requieren valores elevados de atenuación para conseguir relaciones señal-ruido adecuadas (34).

1.1.3.5 Retardo de grupo

Representa el retardo que experimenta cada componente de frecuencia al atravesar el filtro. En filtros de fase lineal, este retardo es constante para todas las frecuencias, permitiendo preservar la morfología de la señal. En aplicaciones de monitorización en tiempo real, el retardo de grupo afecta directamente a la latencia del sistema, siendo un factor crítico a considerar .

1.1.4 Filtros Wavelet

La transformada wavelet constituye una poderosa herramienta para el análisis tiempo-frecuencia de señales no estacionarias, como la mayoría de bioseñales (35). A diferencia de la transformada de Fourier, que solo proporciona información espectral, las wavelets permiten localizar simultáneamente características en tiempo y frecuencia.

El filtrado basado en wavelets descompone la señal en diferentes escalas o niveles de resolución mediante la aplicación de bancos de filtros paso-bajo y paso-alto, seguidos de submuestreo. Esta descomposición permite identificar y modificar selectivamente componentes específicos de la señal (36).

Las aplicaciones biomédicas del filtrado wavelet incluyen:

- **Eliminación de ruido en ECG**: La capacidad de las wavelets para adaptarse a discontinuidades las hace particularmente efectivas para eliminar ruido preservando complejos QRS y otras características morfológicas importantes (37).

- **Detección de eventos en EEG**: La descomposición multiresolución facilita la identificación de patrones específicos como puntas epilépticas o complejos K en registros de electroencefalografía (25).

- **Compresión de señales**: La concentración de energía en pocos coeficientes permite implementar esquemas eficientes de compresión, aspecto relevante en sistemas de adquisición prolongada de bioseñales (38).

La selección de la wavelet madre (Daubechies, Symlet, Coiflet, etc.) y el nivel de descomposición son parámetros críticos que dependen de las características específicas de la bioseñal analizada y los objetivos del procesamiento (39).

1.1.5 Filtros Adaptativos

Los filtros adaptativos constituyen una clase especial de sistemas cuya respuesta se ajusta automáticamente conforme a un algoritmo de optimización y a características de las señales procesadas. A diferencia de los filtros convencionales con coeficientes fijos, los adaptativos modifican sus parámetros dinámicamente, permitiendo responder a cambios en las propiedades estadísticas de las señales o interferencias (40).

Su principio de funcionamiento se basa en la minimización iterativa de una función de costo, habitualmente el error cuadrático medio entre una señal deseada y la salida del filtro. Algunos algoritmos usados son:

- **LMS (Least Mean Squares)**: Algoritmo adaptativo de baja complejidad computacional. Su convergencia suele ser lenta y depende de la elección del tamaño de paso.

- **RLS (Recursive Least Squares)**: Presenta una convergencia más rápida y mejor seguimiento de las variaciones de las señales a costa de mayor complejidad computacional y mayor número de operaciones por iteración.

- **Kalman**: Proporciona estimaciones óptimas del estado de un sistema dinámico lineal bajo la suposición de ruido gaussiano.

En el ámbito biomédico los filtros adaptativos encuentran aplicaciones como (41):

- **Cancelación de interferencias**: Eliminación de artefactos como la interferencia de red eléctrica en ECG.
- **Mejora de relación señal-ruido**: Particularmente útil en señales de baja amplitud como las obtenidas en electromiografía de superficie.

Su implementación en hardware embebido presenta retos debido a sus mayores requisitos computacionales, aunque arquitecturas específicas como ARM Cortex-M4F con unidades DSP integradas y algoritmos adaptados a los recursos disponibles en estas plataformas pueden hacer posible su aplicación en instrumentación biomédica portátil (42).

1.1.6 Bioseñales

Las bioseñales son fenómenos de naturaleza eléctrica, química o mecánica generadas por sistemas biológicos que proporcionan información sobre el estado funcional de un organismo. El procesamiento digital de bioseñales es una herramienta fundamental en instrumentación médica moderna, permitiendo la evaluación objetiva de procesos fisiológicos y facilitando el diagnóstico clínico no invasivo (23). A continuación se describen tres tipos principales de bioseñales bioeléctricas de importancia para aplicaciones en instrumentación biomédica.

1.1.6.1 Electrocardiografía (ECG)

El electrocardiograma representa la actividad eléctrica del corazón registrada desde la superficie corporal mediante electrodos colocados en posiciones específicas del tórax y las extremidades (43). Esta actividad eléctrica se origina en el nodo sinoauricular, compuesto por células especializadas que originan y propagan impulsos eléctricos de manera ordenada y periódica a través del miocardio.

El ciclo cardíaco se inicia con la despolarización espontánea del nodo sinoauricular, localizado en la aurícula derecha, generando un impulso eléctrico que se propaga por las aurículas produciendo su contracción. Este impulso se registra como la onda P en el electrocardiograma. La señal alcanza el nodo auriculoventricular, donde experimenta un

retardo fisiológico que permite el llenado ventricular, para posteriormente propagarse por el haz de His y las fibras de Purkinje, despolarizando los ventrículos. Esta despolarización ventricular genera el complejo QRS, de mayor amplitud que la onda P debido a la mayor masa muscular ventricular. Finalmente, la repolarización ventricular produce la onda T (43).

El potencial registrado en un ECG típico presenta amplitudes del orden de 1 mV y contiene información en un rango de frecuencias entre 0.05 y 150 Hz. Willem Einthoven desarrolló el primer electrocardiógrafo de utilidad clínica en 1901, describió las deflexiones P, Q, R, S y T, y estableció el sistema de derivaciones que lleva su nombre (44).

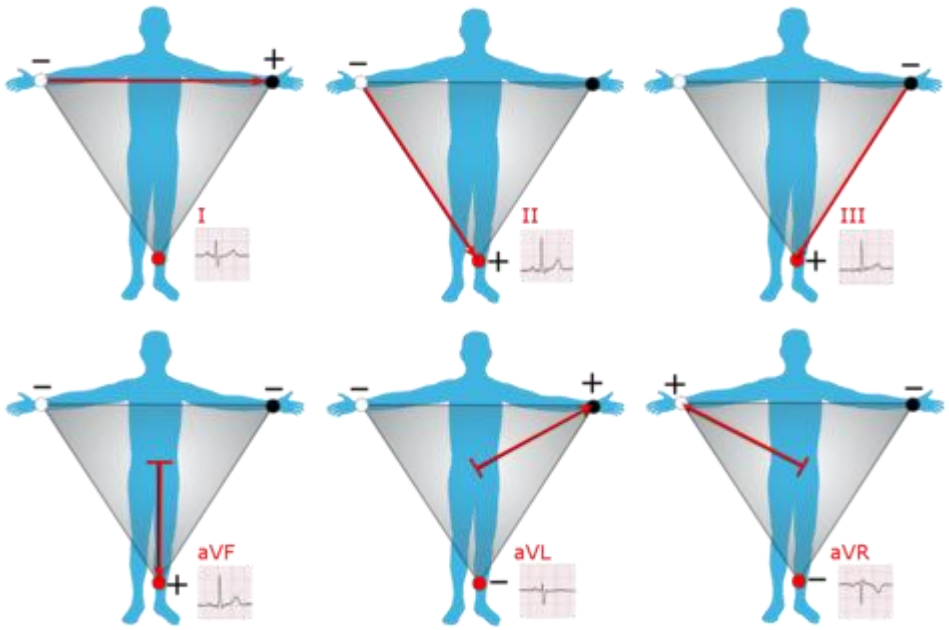

Figura 1. Derivaciones de Einthoven. Fuente: (1)

El ECG constituye una herramienta diagnóstica primaria para la evaluación de arritmias, isquemia miocárdica, infarto de miocardio, hipertrofia ventricular y alteraciones electrolíticas. La correcta preservación de la morfología de la señal durante el procesamiento resulta relevante para la interpretación clínica, particularmente en segmentos como el ST y el intervalo QT.

1.1.6.2 Electromiografía (EMG)

La electromiografía es la técnica de registro de la actividad eléctrica producida por los músculos esqueléticos durante la contracción. Esta actividad eléctrica se origina en el potencial de acción de las fibras musculares, generado cuando un impulso nervioso transmitido a través de la unión neuromuscular despolariza la membrana de la célula muscular.

La unidad funcional básica del sistema neuromuscular es la unidad motora, compuesta por una motoneurona y todas las fibras musculares que inerva. El potencial de acción de la unidad motora representa la suma temporal y espacial de los potenciales de acción de las fibras musculares individuales que la componen. El EMG registrado corresponde a la superposición de los potenciales de múltiples unidades motoras activas durante la contracción muscular (45).

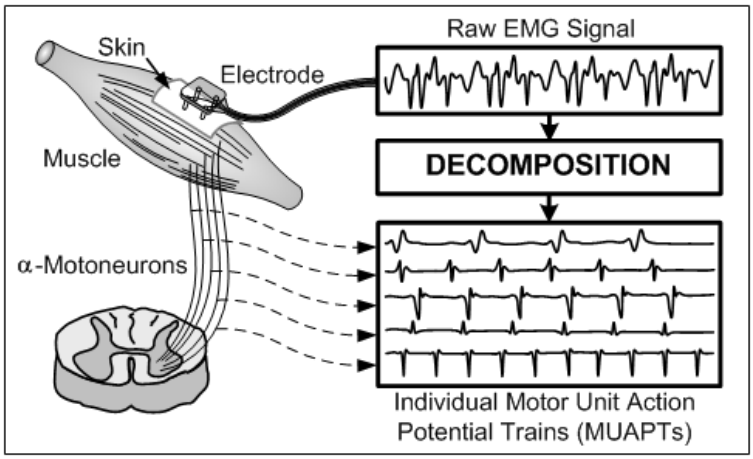

Figura 2. Obtención y descomposición de señal EMG en unidades motoras individuales. Fuente: Khan Academy

El registro electromiográfico puede realizarse mediante electrodos de superficie, colocados sobre la piel que recubre el músculo de interés, o mediante electrodos intramusculares de aguja. Los electrodos de superficie registran la actividad global del músculo y son apropiados para estudios biomecánicos. Los electrodos intramusculares permiten el registro selectivo de unidades motoras individuales y son utilizados en evaluación clínica de patologías neuromusculares.

Las señales EMG de superficie presentan amplitudes que varían típicamente entre 50 µV y 30 mV, dependiendo del músculo evaluado y el nivel de contracción. El contenido espectral de la señal EMG se distribuye principalmente entre 10 y 500 Hz, con la mayor parte de la energía concentrada entre 50 y 150 Hz (46).

La EMG encuentra aplicación en el diagnóstico diferencial de patologías musculares (miopatías) y alteraciones de la inervación (neuropatías), la evaluación de trastornos de la unión neuromuscular como la miastenia gravis, y el estudio de patrones de activación muscular en rehabilitación y análisis del movimiento.

1.1.6.3 Electroencefalografía (EEG)

El electroencefalograma registra la actividad eléctrica cerebral mediante electrodos colocados en el cuero cabelludo según el sistema internacional 10-20 (47). Esta actividad refleja los potenciales postsinápticos generados por la actividad sincronizada de poblaciones neuronales corticales, principalmente en las capas superficiales de la corteza cerebral.

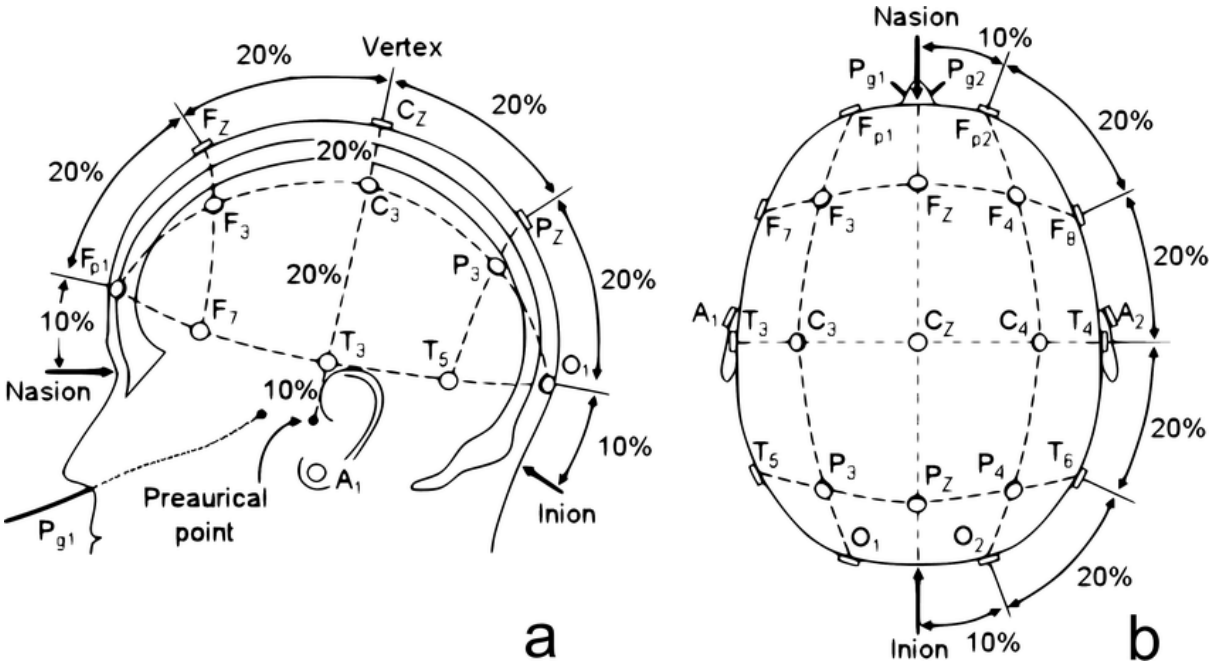

Figura 3. Colocación de electrodos según el sistema internacional 10-20 para adquisición de señal EEG. Fuente: (2)

Los potenciales eléctricos registrados en EEG se originan fundamentalmente en las corrientes transmembrana asociadas a la actividad sináptica. Cuando grupos de neuronas piramidales orientadas perpendicularmente a la superficie cortical se activan de manera sincronizada, sus campos eléctricos se suman espacialmente, generando un potencial suficiente para ser detectado en el cuero cabelludo. Este proceso requiere la activación simultánea de poblaciones neuronales del orden de 10^6 neuronas distribuidas en varios centímetros cuadrados de corteza.

La actividad eléctrica cerebral se manifiesta como oscilaciones rítmicas clasificadas en bandas de frecuencia específicas, cada una asociada a estados funcionales particulares: delta (0.5-4 Hz), presente durante el sueño profundo; theta (4-8 Hz), asociada a estados de somnolencia; alfa (8-13 Hz), característica del estado de vigilia relajada con ojos cerrados; beta (13-30 Hz), relacionada con el estado de alerta y concentración; y gamma (>30 Hz), vinculada a procesos cognitivos complejos. Las amplitudes típicas del EEG varían entre 10 y 100 μV.

El EEG constituye el método diagnóstico primario para epilepsia, permitiendo la identificación de actividad epileptiforme intercrítica y la clasificación de síndromes epilépticos. También encuentra aplicación en la evaluación de alteraciones del nivel de conciencia, encefalopatías, trastornos del sueño, demencias y muerte cerebral. En investigación neurocientífica, el EEG permite estudiar procesos cognitivos, interfaces cerebro-computadora y respuestas cerebrales a estímulos externos.

1.1.7 Hardware Embebido para DSP

El hardware embebido se define como un conjunto de componentes computacionales especializados diseñados para ejecutar funciones específicas, generalmente bajo restricciones de rendimiento en tiempo real, formando parte de un sistema completo que puede incluir software y elementos mecánicos. A diferencia de los computadores de propósito general, el hardware embebido está optimizado para una aplicación particular, lo que permite mejorar su eficiencia, reducir su tamaño, costo y consumo energético.

En el contexto biomédico, el hardware embebido es usado para la implementación de dispositivos de monitorización, diagnóstico y terapia, donde la capacidad de procesamiento en tiempo real de bioseñales resulta fundamental (48). La miniaturización, la eficiencia energética y la fiabilidad que caracterizan a este tipo de hardware lo convierten en un candidato ideal para aplicaciones que van desde la monitorización ambulatoria hasta dispositivos implantables (49).

1.1.7.1 Microcontroladores (ARM Cortex-M)

Los microcontroladores constituyen el núcleo computacional de la mayoría del hardware embebido moderno. Entre las arquitecturas más importantes para aplicaciones de DSP, la familia ARM Cortex-M ofrece equilibrio entre eficiencia energética, rendimiento computacional y funcionalidades específicas para DSP.

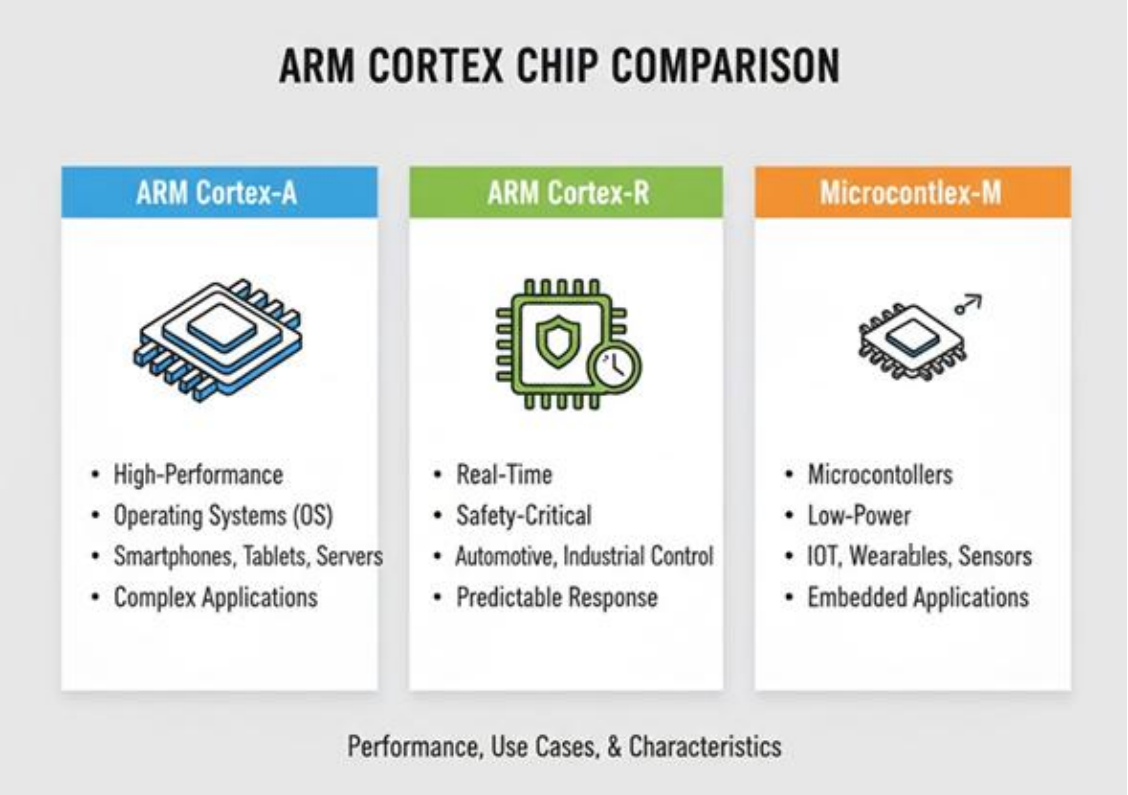

Figura 4. Comparativa de las familias ARM Cortex: Series A, R y M. Fuente: Elaboración propia

La arquitectura ARM Cortex-M, especialmente las variantes M4 y M7, incorpora extensiones SIMD (Single Instruction, Multiple Data) y unidades de punto flotante (FPU), características de gran utilidad para implementar algoritmos de filtrado digital y transformadas (50). Estas características permiten ejecutar operaciones matemáticas complejas de manera eficiente y reducir los ciclos de reloj necesarios y el consumo energético.

Los microcontroladores ARM Cortex-M presentan varias ventajas para aplicaciones de procesamiento de bioseñales:

1. Instrucciones específicas para DSP: Incluyen operaciones optimizadas como MAC (Multiply-Accumulate) esenciales para implementar filtros FIR e IIR de manera eficiente.

2. Pipeline de instrucciones: Su arquitectura permite ejecutar instrucciones en paralelo, mejorando el rendimiento en operaciones repetitivas como las que se ejecutan en los algoritmos de filtrado.

3. Controladores DMA (Direct Memory Access): Facilitan la transferencia eficiente de datos entre periféricos y memoria, minimizando la intervención del procesador.

4. Modos de bajo consumo configurables: Permiten optimizar el consumo energético según los requisitos de la aplicación.

Los sistemas Cortex-M modernos suelen incluir además periféricos de alta precisión como conversores analógico-digitales (ADC) de 12 bits o superiores, cruciales para la adquisición precisa de bioseñales cuyas amplitudes pueden variar desde microvoltios hasta milivoltios (51).

1.1.7.2 Arduino Due para prototipado biomédico

Entre las plataformas disponibles, Arduino Due presenta características particulares que la posicionan como una opción atractiva para el prototipado de aplicaciones biomédicas en entornos educativos y de investigación (52):

1. Microcontrolador SAM3X8E (ARM Cortex-M3): Ofrece suficiente potencia de procesamiento (84 MHz) para implementar algoritmos de filtrado en tiempo real para la mayoría de bioseñales, cuyo ancho de banda típicamente es menor a 500 Hz.

2. Conversores ADC de 12 bits: Proporcionan una resolución adecuada para capturar señales biomédicas con precisión suficiente para aplicaciones educativas y de investigación.

3. Operación a 3.3V: A diferencia de otras placas Arduino que operan a 5V, la tensión de 3.3V del Due es compatible con numerosos sensores biomédicos modernos sin necesidad de adaptación de niveles.

4. Comunidad activa y código abierto: El ecosistema Arduino facilita el acceso a ejemplos, tutoriales y bibliotecas complementarias, esencial en contextos educativos donde el tiempo de aprendizaje es limitado a solamente 1 o 2 semestres académicos (5).

5. Compatibilidad con CMSIS-DSP: Arduino Due puede ejecutar eficientemente la biblioteca CMSIS-DSP, aprovechando optimizaciones específicas como los tipos de datos Q15 (53).

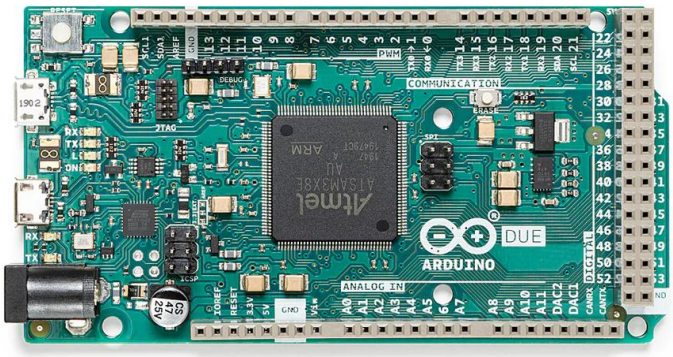

Figura 5. Microcontrolador Arduino Due con un CPU Cortex-M3. Fuente: Arduino Store

Estudios previos han demostrado la capacidad de esta placa para implementar filtros FIR e IIR de orden medio-alto con latencias aceptables para aplicaciones biomédicas no críticas (7), lo que lo posiciona como una solución viable para el contexto educativo.

Estas características, combinadas con su accesibilidad económica y amplia disponibilidad global, convierten al Arduino Due en una plataforma idónea para la implementación de BioFilterLib, especialmente considerando su enfoque educativo y la necesidad de balancear rendimiento con facilidad de uso.

1.2 Estado del Arte

1.2.1 Librerías de DSP en Hardware Embebido

1.2.1.1 CMSIS-DSP

CMSIS-DSP (Cortex Microcontroller Software Interface Standard - Digital Signal Processing) es una librería de alto rendimiento desarrollada por ARM para proporcionar funciones de procesamiento digital de señales optimizadas para arquitecturas ARM Cortex-M (15). Esta librería forma parte del estándar CMSIS, un conjunto de herramientas, APIs, frameworks y flujos de trabajo diseñados para facilitar el desarrollo de software en microcontroladores basados en núcleos ARM.

La biblioteca CMSIS-DSP implementa más de 60 funciones comunes de procesamiento digital de señales, incluyendo operaciones matemáticas básicas, transformadas rápidas de Fourier (FFT), filtros digitales, y funciones estadísticas (15). Estas implementaciones están altamente optimizadas para aprovechar las características específicas de las arquitecturas ARM Cortex-M, utilizando instrucciones SIMD donde están disponibles y minimizando ciclos de reloj y uso de memoria.

Entre las principales ventajas de CMSIS-DSP destacan su rendimiento optimizado, ya que contiene funciones implementadas y optimizadas a nivel de ensamblador para cada arquitectura ARM específica, logrando eficiencias significativamente superiores a implementaciones genéricas en C (54). Por otro lado, presenta una considerable portabilidad y una API que funciona de manera consistente en diferentes núcleos Cortex-M, lo que facilita la migración de código entre plataformas.

Adicionalmente, la librería ofrece precisión controlada mediante implementaciones tanto en punto fijo (q7, q15, q31) como en punto flotante (float32), permitiendo a los desarrolladores seleccionar el equilibrio adecuado entre precisión y rendimiento según los requerimientos específicos de la aplicación.

La verificación exhaustiva es otra ventaja importante, pues las funciones son sometidas a pruebas de validación respaldadas por ARM, garantizando su corrección y rendimiento óptimo en diferentes condiciones operacionales.

Sin embargo, CMSIS-DSP presenta limitaciones desde una perspectiva de usabilidad educativa. Su complejidad de uso representa una barrera considerable, pues la API está diseñada priorizando el rendimiento sobre la facilidad de uso, requiriendo un conocimiento avanzado tanto de DSP como de programación en C y particularidades de las arquitecturas ARM (55). Esta complejidad se ve incrementada por una curva de aprendizaje pronunciada, donde la documentación está orientada a desarrolladores con experiencia y carece de ejemplos didácticos de complejidad progresiva adecuados para estudiantes.

La escasez de ejemplos específicos para bioseñales se presenta como una oportunidad de mejora, ya que la documentación no aborda casos de uso biomédicos, dejando un vacío significativo para estudiantes de ingeniería biomédica. De igual manera, la gestión manual de memoria impone una carga adicional al requerir que el desarrollador gestione explícitamente la asignación y liberación de buffers de estado y coeficientes, aumentando considerablemente la probabilidad de errores en usuarios principiantes.

Por último, la ausencia de abstracciones de alto nivel dificulta el proceso de aprendizaje, pues las funciones exponen directamente los detalles de implementación de los algoritmos, dificultando que los estudiantes se centren en los conceptos fundamentales de DSP durante su proceso formativo.

Estas limitaciones resultan particularmente problemáticas en contextos educativos, donde el tiempo disponible para familiarizarse con detalles técnicos es limitado y el foco debería estar en comprender los principios del procesamiento de señales más que en las complejidades de su implementación.

1.2.1.2 Librerías de filtrado de señales para microcontroladores

El ecosistema de procesamiento de señales digitales de Arduino cuenta con bibliotecas y funciones especializadas, con repositorios mantenidos activamente y con validación académica. Aunque persisten las limitaciones de rendimiento para aplicaciones más profesionales, los recientes avances en las bibliotecas optimizadas para ESP32 y las implementaciones de ARM Cortex-M han ampliado las capacidades prácticas del DSP

basado en microcontroladores. La biblioteca LeemanGeophysicalLLC/FIR_Filter_Arduino_Library, basada en la librería de Nilsson (16), es una librería comúnmente usada para implementar filtros FIR y está presente en el repositorio oficial de librerías de Arduino (56), sin embargo no cuenta con una documentación propia de sus funciones y solo tiene ejemplos de dos tipos de filtrado, con filtro FIR y filtro de media móvil. Además, a septiembre del 2025, el repositorio no ha recibido mantenimiento ni actualizaciones en 8 años lo que podría limitar su compatibilidad con versiones recientes del IDE de Arduino.

También se recopilaron las bibliotecas que mantienen un desarrollo activo con commits regulares entre 2023 y 2024: tttapa/Arduino-Filters (57), espressif/esp-dsp (58), kosme/arduinoFFT (59), pschatzmann/arduino-audio-tools (60) y mozanunal/SimpleDSP (61).

Las implementaciones de filtros FIR han convergido en torno a tres enfoques distintos. La biblioteca daPhoosa/FirFilter optimiza la velocidad con código ensamblador específico para AVR (Arduino Uno, Nano, Mega), logrando una ejecución de 48 µs para filtros de 7 etapas en plataformas Arduino de 16 MHz (62). Moarbue/FIR-Filter ofrece una implementación fácil para principiantes con filtros de media móvil integrados y una especificación de coeficientes simplificada (63).

Los frameworks de DSP integrales son un avance en el procesamiento de señales de Arduino. La biblioteca tttapa/Arduino-Filters destaca por su compatibilidad con filtros IIR, implementaciones BiQuad, filtros Butterworth y una amplia compatibilidad con plataformas en todas las variantes de Arduino, ESP8266 y ESP32 (57). La biblioteca espressif/esp-dsp proporciona implementaciones oficiales optimizadas para hardware con soporte tanto para float32 como para int16, lo que la convierte en la primera biblioteca DSP madura y compatible con el fabricante en el ecosistema Arduino (58).

Adicionalmente se han desarrollado optimizaciones específicas de plataforma con miras a su implementación en aplicaciones profesionales. La biblioteca esp32-i2s-slm implementa filtros IIR en secciones de segundo orden (Second-Order Sections) con optimización en ensamblador para ESP32 para la medición de audio en tiempo real (64). Existen múltiples implementaciones específicas para STM32 que proporcionan filtrado de flujos de audio con baja latencia. El framework arduino-audio-tools integra los

componentes STK, Maximilian y Mozzi, creando una capacidad de procesamiento de audio completa que antes no estaba disponible en los entornos Arduino (60).

La Tabla 1 presenta una comparativa de las principales bibliotecas disponibles para implementar filtros digitales en Arduino y plataformas similares.

Tabla 1. Principales bibliotecas de DSP para hardware embebido

Repositorio	Enfoque principal	Plataforma(s) objetivo	Optimizaciones/Implementación	Capacidades destacadas	Mantenimiento/Último commit	Observaciones
LeemanGeophysicalLLC/FIR_Filter_Arduino_Library	FIR (incluye media móvil)	Arduino clásicos	Implementación sencilla, sin actualizaciones, compatibilidad con IDE 1.0.1	Ejemplos de **implementacion** FIR y media móvil	Sin mantenimiento, último commit en 2017	Implementación independiente de un filtro FIR y media móvil básicos
tttapa/Arduino-Filters	**IIR**, **BiQuad**, **Butterworth**	Arduino, **ESP8266/ESP32**	Estructuras estándar y estables (biquads)	Amplia cobertura de filtros clásicos	Sin mantenimiento, último commit en 2022	Adecuada para cursos (del diseño teórico a implementación robusta)
sebnil/FIR-filter-Arduino-Library	**FIR**	Arduino	Implementación sencilla, sin actualizaciones, compatibilidad con IDE 1.0.1	Ejemplo de **implementación** FIR	Sin mantenimiento, último commit en 2019	Implementación independiente de un filtro FIR básico
espressif/esp-dsp	Framework DSP optimizado	ESP32 (vendor)	**Optimizada** (fabricante); **float32** y **int16**	Conjunto amplio de rutinas DSP	Mantenimiento continuo, último commit en 2025	Ideal para laboratorios con ESP32 y prácticas de rendimiento en tiempo real.
kosme/arduinoFFT	FFT/Espectro	Arduino	C implementación optimizada para MCUs	FFT para laboratorios de frecuencia	Mantenimiento continuo, último commit en 2025	Adecuado para enseñar análisis espectral en hardware limitado.

pschatzmann/arduino-audio-tools	Framework de audio integral	Arduino/ESP	Integración de **STK, Maximilian, Mozzi**	I/O + procesado de audio “end-to-end”	Mantenimiento continuo, último commit en 2025	Útil para prácticas completas de audio (captura-procesado-playback)
mozanunal/SimpleDSP	Operadores DSP básicos	Arduino	Enfoque ligero en operaciones básicas	Conjunto básico de filtros/operaciones	Sin mantenimiento, último commit en 2020	Implementación sencilla y uso adecuado en cursos introductorios
daPhoosa/FirFilter	**FIR** optimizado para AVR	**AVR 16 MHz** (UNO/Nano/Mega)	**ASM AVR**; **48 µs (7 taps, 16 MHz)**	FIR rápido para 8-bit AVR	Sin mantenimiento, último commit en 2021	Ideal para mostrar optimización a bajo nivel y mediciones de latencia.
Moarbue/FIR-Filter	**FIR** (foco educativo)	Arduino	API **simplificada**; incluye **media móvil**	Fácil especificación de coeficientes y testeo rapido	Desarrollo reciente, último commit en 2024	Interfaz simple que facilita comprender la relación entre coeficientes y respuesta del filtro
ikostoski/esp32-i2s-slm	**Aplicación**: medidor de nivel sonoro	ESP32	IIR en SOS + ASM	Pipeline práctico de medición en tiempo real	Sin mantenimiento, último commit en 2019	Caso de estudio de aplicación profesional en ESP32.
etiennedemontalivet arm-dwt-c-library	**1-D Wavelet**	Implementado en stm32h7 (Cortex - M7)	Utiliza CMSIS para aplicar la transformada discreta de wavelet	Optimizado para mejor comprensibilidad de código	Sin mantenimiento, último commit en 2020	Enfocado a tests rápidos, no optimizado para velocidad sino para comprensibilidad de código

1.2.2 Librerías DSP para Arduino en educación

Investigaciones académicas de 2015-2024 demuestran el valor educativo de las plataformas Arduino para la instrucción en DSP, con múltiples publicaciones IEEE documentando implementaciones en programas educativos universitarios (65–67). Los estudios revelan que la simplicidad del IDE de Arduino permite a los estudiantes enfocarse en los fundamentos de DSP en lugar de la complejidad del entorno de desarrollo, con investigaciones mostrando efectos positivos en las actitudes y autoeficacia de los estudiantes en programación (68).

Estudios pedagógicos identifican al Arduino Due como la plataforma mínima viable para educación significativa en DSP en tiempo real. Jaldén et al. (2018) documentaron plataformas educativas basadas en Arduino Due con shields de interfaz de audio analógico dedicados, enfatizando la baja complejidad de la plataforma mientras se mantienen los objetivos educativos (65). Prasad y Rajasri (2019) demostraron la implementación de filtros FIR en tiempo real (orden 10) para frecuencias por debajo de 5 kHz utilizando integración de MATLAB Filter Design Tool con ejecución en Arduino Due (69).

Investigación de análisis de rendimiento establece límites cuantitativos claros para aplicaciones DSP en Arduino. Se han documentado frecuencias de muestreo máximas limitadas en plataformas de 8 bits, con implementaciones típicamente restringidas a frecuencias por debajo de 5kHz para mantener procesamiento en tiempo real con órdenes de filtro significativos (66,67). La investigación indica las restricciones de memoria como la limitación principal, con 2KB SRAM restringiendo tamaños de buffer y almacenamiento de coeficientes en plataformas de 8 bits.

Implementaciones prácticas en entornos universitarios demuestran la viabilidad educativa de Arduino para DSP. El proyecto Lab-In-A-Box de Stanford proporciona un sistema de enseñanza compatible con Arduino para electrónica digital y su implementación y utilidad en programas educativos ha sido documentada y validada (66). Estudios adicionales han validado que Arduino facilita la comprensión de conceptos esenciales de hardware embebido y procesamiento de señales en contextos educativos (70).

Estudios comparativos de efectividad revelan el nicho educativo de las plataformas Arduino. Wickert (2015) demostró que las plataformas de microcontroladores Arduino permiten la exploración de problemas de aliasing y reconstrucción no visibles con convertidores sigma-delta, lo que proporciona un valor educativo adicional para conceptos de DSP. Sin embargo, estudios comparativos de rendimiento muestran limitaciones en cuanto a procesamiento secuencial de ciertos microcontroladores comparado con las capacidades de procesamiento paralelo de FPGAs, definiendo límites claros de aplicación (70,71).

Algunas limitaciones de rendimiento documentadas incluyen la velocidad limitada de conversión ADC y restricciones de memoria que afectan la implementación de filtros de orden alto. La literatura confirma que Arduino Due proporciona capacidades suficientes para educación en DSP y para aplicaciones de alto rendimiento se requieren plataformas más avanzadas. Los estudios enfatizan que la selección de la plataforma a utilizar debe balancear los objetivos educativos con requisitos de rendimiento específicos.

1.3 Problemática

La enseñanza del procesamiento digital de bioseñales en sistemas embebidos enfrenta desafíos relacionados con la falta de componentes de hardware accesibles que tengan acceso a herramientas para el procesamiento de bioseñales. Las plataformas industriales de DSP como el kit de desarrollo TMDSEVM6678 (72), con un costo de 599 USD, o placas con Field-Programmable Gate Array (FPGA) como el Cora Z7-07S de AMD (73), por un precio de 149 USD, son opciones viables, pero presentan una elevada curva de aprendizaje que dificulta su adopción en contextos educativos.

Si bien existen otras propuestas educativas que emplean microcontroladores disponibles en el mercado para la enseñanza de DSP (74,75), las librerías disponibles para Arduino presentan limitaciones adicionales como la escasa orientación específica para bioseñales, la documentación desactualizada y falta de herramientas integradas que faciliten la implementación y la evaluación de algoritmos de filtrado en contextos educativos.

Esta situación pone en evidencia la necesidad de desarrollar una solución que integre accesibilidad económica, documentación completa y funcionalidades diseñadas para el procesamiento de señales biomédicas en plataformas de bajo costo.

1.4. Justificación

Las experiencias en entornos de enseñanza de DSP a nivel universitario se benefician en gran medida de la experimentación práctica con hardware accesible y herramientas bien documentadas. La implementación de ejercicios experimentales con plataformas como Arduino permite a los estudiantes desarrollar una mayor comprensión de los conceptos teóricos fundamentales de DSP, transformando el conocimiento de la teoría en habilidades aplicables a su desarrollo profesional.

BioFilterLib se plantea como una herramienta diseñada que facilita el aprendizaje del procesamiento de bioseñales en entornos educativos. Esta librería proporciona una solución integrada que combina accesibilidad, documentación completa y usabilidad optimizada, lo que brinda a estudiantes de ingeniería un entorno adecuado para la aplicación de técnicas de filtrado digital en niveles principiantes.

La plataforma Arduino Due es una opción prometedora para la enseñanza de DSP, ya que ofrece capacidades de procesamiento adecuadas a un costo accesible. Al desarrollar y validar técnicamente una librería orientada al procesamiento de señales biomédicas para este ecosistema, se puede proponer reforzar su inclusión en planes de aprendizaje mediante la implementación directa de algoritmos de filtrado utilizando hardware real.

Esta propuesta contribuye a fortalecer las propuestas de formación práctica en ingeniería biomédica, proporcionando una herramienta que integra diseño de filtros, implementación en tiempo real, visualización de resultados y métricas de evaluación, elementos esenciales para una experiencia de aprendizaje completa en el procesamiento de bioseñales.

II. Hipótesis y Objetivos

2.1. Objetivo general

Desarrollo y evaluación de la librería de filtros digitales para Arduino Due “BioFilterLib” respecto a otras librerías en términos de velocidad de procesamiento, consumo de recursos y calidad de filtrado.

2.2. Objetivos específicos

- Evaluar la factibilidad técnica de BioFilterLib en la tarea de procesamiento de bioseñales. Esto implica realizar pruebas de filtrado de señales biomédicas y recopilar métricas de calidad, velocidad de procesamiento y recursos de hardware utilizados.
- Comparar el consumo de recursos de hardware en términos de velocidad de procesamiento, calidad de filtrado y consumo de recursos respecto a otras librerías disponibles en lenguaje C++.

2.3. Hipótesis general

La librería de filtros digitales para Arduino “BioFilterLib” es comparable o mejor respecto a otras librerías disponibles en términos de velocidad de procesamiento, consumo de recursos y calidad de filtrado

III. Metodología

3.1. Diseño del proyecto

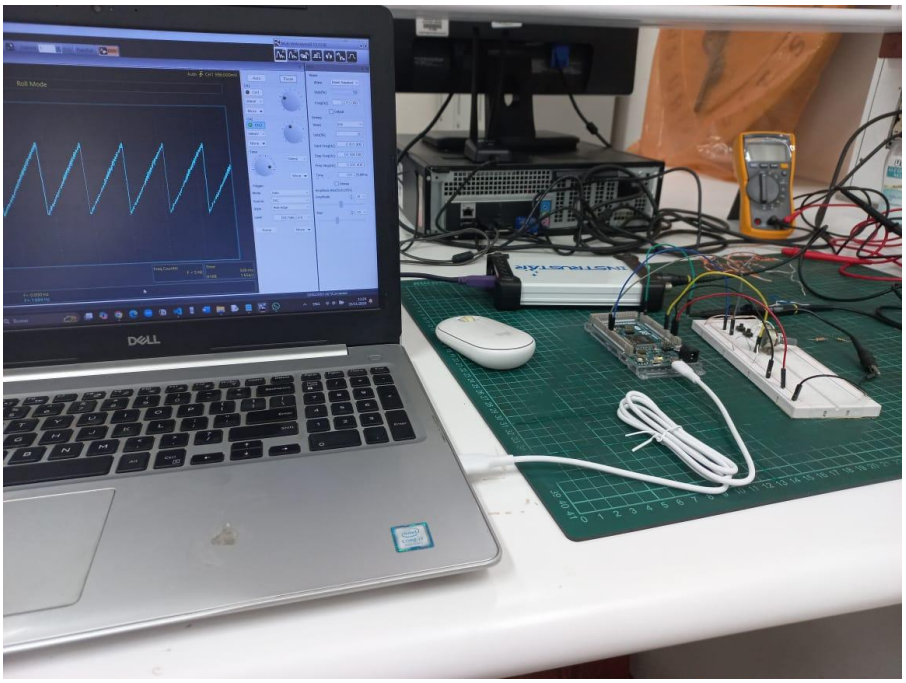

Figura 6. Experimento de muestreo y generación de señales en tiempo real con Arduino Due. Fuente: Elaboración propia

El presente estudio se centró en el desarrollo y evaluación de una librería para procesamiento digital de bioseñales en plataformas de hardware embebido. El diseño metodológico se estructuró en tres componentes principales: (1) desarrollo de la librería BioFilterLib con arquitectura orientada a contextos educativos, (2) implementación de evaluación comparativa, y (3) análisis cuantitativo de métricas de rendimiento y calidad de filtrado.

El flujo de trabajo experimental se estructuró en cinco componentes, como se muestra en la Figura 6:

1. **Generación de señales de prueba:** Utilización de notebooks Jupyter para generar bioseñales sintéticas (ECG, EMG, EEG) con ruido controlado y características morfológicas representativas de señales reales.

2. **Diseño de filtros digitales:** Especificación de parámetros de filtros (orden, frecuencias de corte, tipo de ventana) y generación de coeficientes mediante herramientas computacionales (Python/SciPy).
3. **Implementación en hardware:** Desarrollo de sketches de prueba específicos para cada librería evaluada (BioFilterLib y librerías de referencia).
4. **Ejecución de protocolos de evaluación:** Recopilación automatizada de métricas mediante funciones integradas en cada sketch, lo que incluye tiempo de procesamiento (microsegundos por muestra), uso de CPU (porcentaje), y métricas de calidad de filtrado (SNR, MSE, correlación).
5. **Análisis de resultados:** Procesamiento de datos mediante notebooks Jupyter para generar visualizaciones comparativas en dominios temporal y frecuencial.

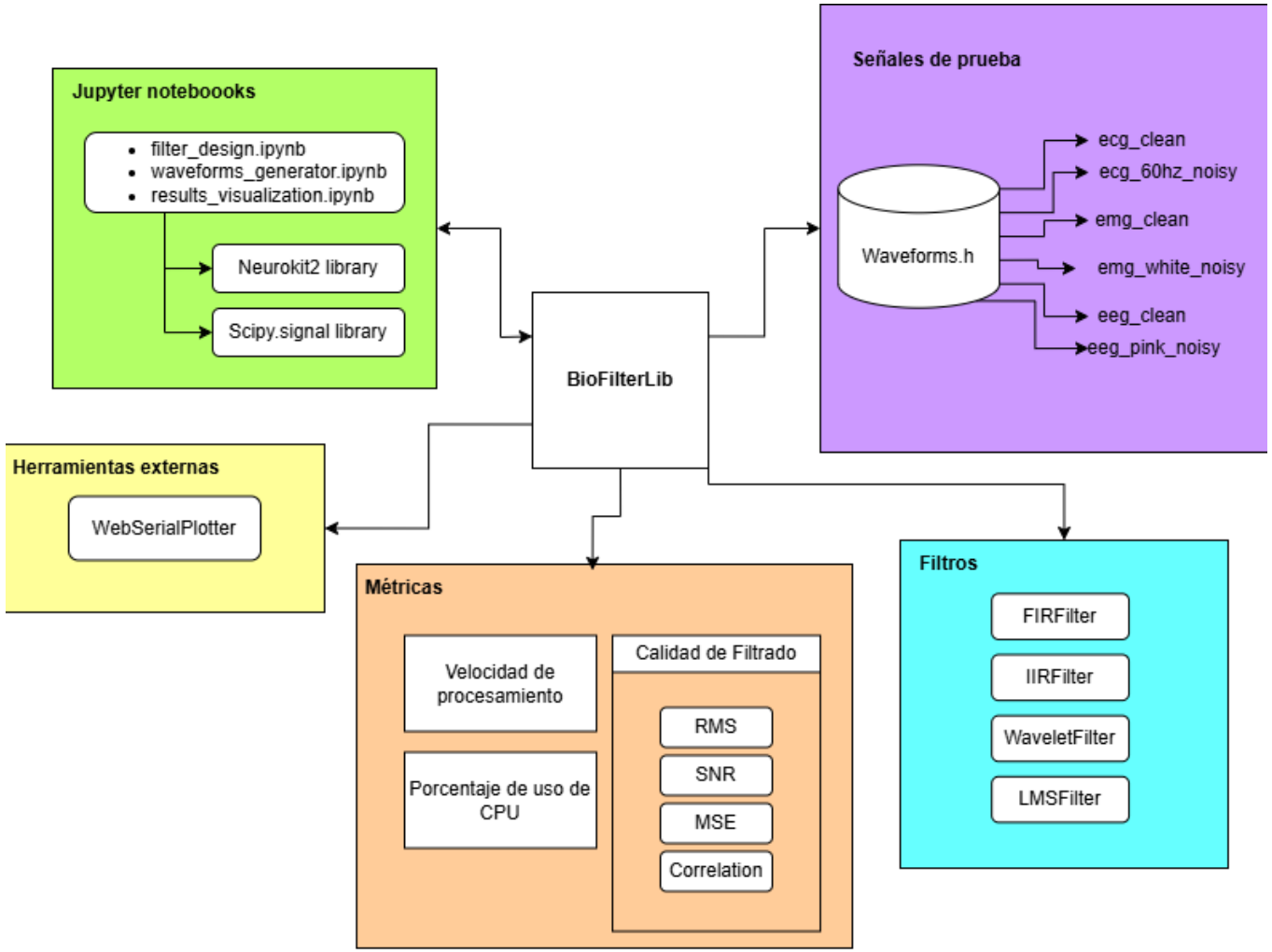

Figura 7. Componentes del proyecto. Fuente: Elaboración propia

3.1.1 Arquitectura de la librería

BioFilterLib ha sido diseñada siguiendo principios de arquitectura de software orientada a objetos, con un enfoque en la modularidad, extensibilidad y usabilidad. La estructura de la librería se ha organizado en múltiples capas de abstracción, permitiendo tanto un uso simplificado para usuarios noveles como acceso a funcionalidades avanzadas para implementaciones especializadas como se muestra en la Figura 7.

El código de la librería está publicado en el siguiente repositorio de GitHub: https://github.com/SergioMoreno1060/BioFilterLib/tree/arduino_ide_version

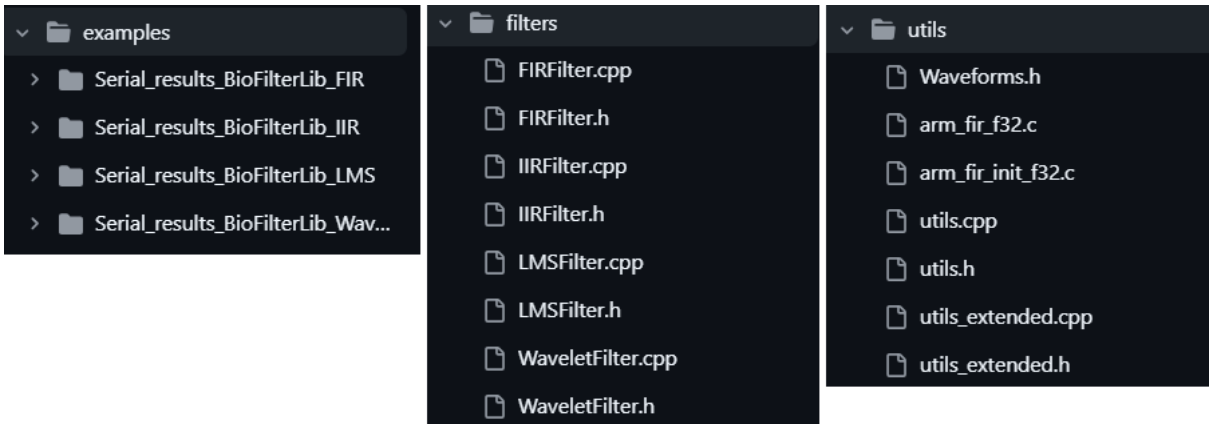

Figura 8. Directorios en GitHub de la librería. Fuente: Elaboración propia

La arquitectura general de BioFilterLib comprende tres directorios principales:

Directorio filters: Contiene las clases principales de filtrado digital implementadas en archivos de cabecera independientes: *FIRFilter.h* (filtros de respuesta finita al impulso), *IIRFilter.h* (filtros de respuesta infinita al impulso con estructura biquad), *LMSFilter.h* (filtros adaptativos basados en algoritmo Normalized Least Mean Squares), y *WaveletFilter.h* (descomposición y reconstrucción wavelet para análisis multiresolución). Esta capa gestiona las interacciones con el hardware específico (Arduino Due), incluyendo configuración de optimizaciones específicas para ARM Cortex-M3 y gestión de memoria, aislando las capas superiores de las particularidades del hardware y facilitando la futura portabilidad a otras plataformas.

Directorio utils: Implementa funciones auxiliares para el cálculo de métricas de evaluación (SNR, MSE, correlación, RMS) y contiene el archivo Waveforms.h, que

almacena bioseñales sintéticas pregeneradas (ECG, EMG, EEG) como arrays constantes en memoria flash. Este archivo permite cargar señales de prueba en cualquier sketch de Arduino mediante la función *loadSignal()*, especificando el nombre de la señal deseada como argumento. Esta capa hace uso de la biblioteca CMSIS-DSP para optimizar el rendimiento, simplificando la complejidad de las llamadas directas a sus funciones de bajo nivel.

Directorio de ejemplos: Directorio donde se pueden encontrar sketches demostrativos para el procesamiento de bioseñales específicas (ECG, EMG, EEG), incluyendo configuraciones preestablecidas, aplicación de filtros y visualización de resultados.

El código de BioFilterLib propone un diseño modular que separa cada tipo de filtro y funcionalidad en clases independientes siguiendo el diagrama UML que se aprecia en la Figura 9. Las clases principales incluyen:

- **FIRFilter:** Implementa filtros de respuesta impulsional finita
- **IIRFilter:** Implementa filtros de respuesta impulsional infinita con estructura biquad
- **WaveletFilter:** Implementa descomposición y reconstrucción wavelet para análisis multiresolución
- **LMSFilter:** Implementa filtros adaptativos basados en algoritmo LMS normalizado

Cada clase expone una API consistente con métodos *init(), process()* y *end()* para gestionar el ciclo de vida del filtro, además de métodos estáticos que permiten operaciones únicas lo que facilita implementaciones con restricciones de memoria.

Por otro lado, el ecosistema de BioFilterLib integra herramientas externas que amplían sus capacidades para visualización de resultados:

Web Serial Plotter: Herramienta de código abierto (https://web-serial-plotter.atomic14.com/) que permite visualizar datos recibidos del puerto serial de Arduino en tiempo real. Cuenta con funcionalidades de acercamiento, captura de pantalla y descarga de datos en múltiples formatos. Esta herramienta se utilizó como alternativa al

Serial Plotter del Arduino IDE, que no ofrece estas capacidades avanzadas de análisis visual.

Notebooks Jupyter: Se desarrollaron tres notebooks de Python para facilitar el flujo de trabajo experimental:

- 00_Generacion_Senales.ipynb: Guía la generación de bioseñales sintéticas con parámetros configurables (amplitud, frecuencia de muestreo, duración, tipo de ruido), facilitando la creación de señales de prueba listas para integrar en el archivo Waveforms.h.
- *01_Diseno_Filtros.ipynb*: Proporciona código paso a paso para la generación de coeficientes de filtros FIR e IIR utilizando las librerías SciPy y NumPy, permitiendo el diseño sistemático de filtros con especificaciones personalizadas.
- *results_visualization.ipynb*: Ofrece análisis en dominios temporal y frecuencial de los resultados de filtrado, utilizando librerías especializadas de Python (SciPy, Matplotlib, NumPy) para generar gráficas comparativas, espectros de magnitud mediante FFT, y cálculo de métricas de calidad.

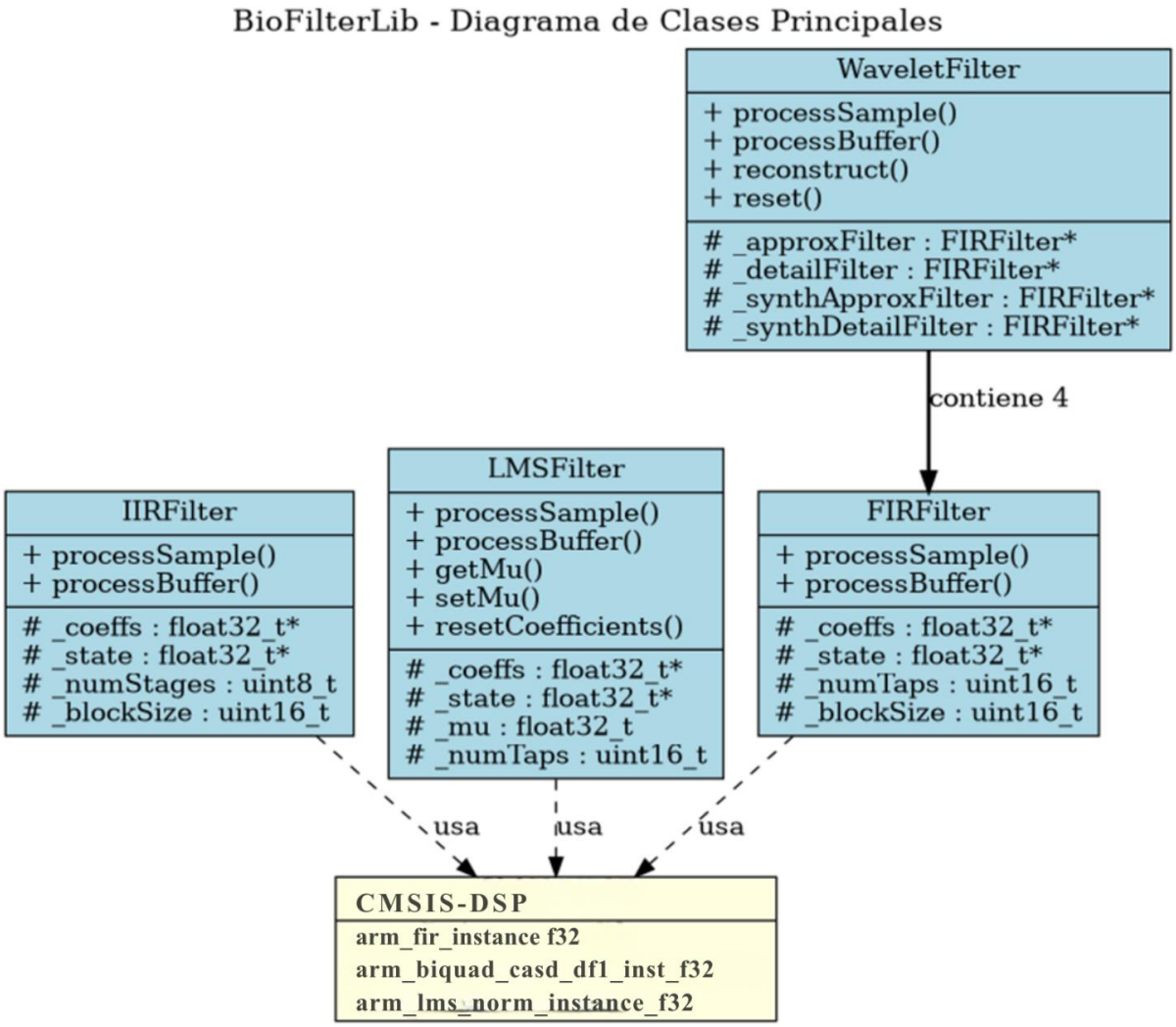

Figura 9. Diagrama UML clases principales de BioFilterLib. Fuente: Elaboración propia

3.1.2 Diseño de la documentación

Un componente fundamental del proyecto BioFilterLib es su enfoque hacia la experiencia educativa, materializado en una estrategia integral de documentación y ejemplos.

Figura 10. Página principal para filtro FIR en la documentación. Fuente: https://sergiomoreno1060.github.io/BioFilterLib/

1. **Ejemplos contextualizados**: Se crearon ejemplos completos para cada tipo principal de bioseñal (ECG, EMG, EEG), que incluyen no solo el código de filtrado sino también la adquisición, visualización y análisis básico, proporcionando un flujo de trabajo completo.

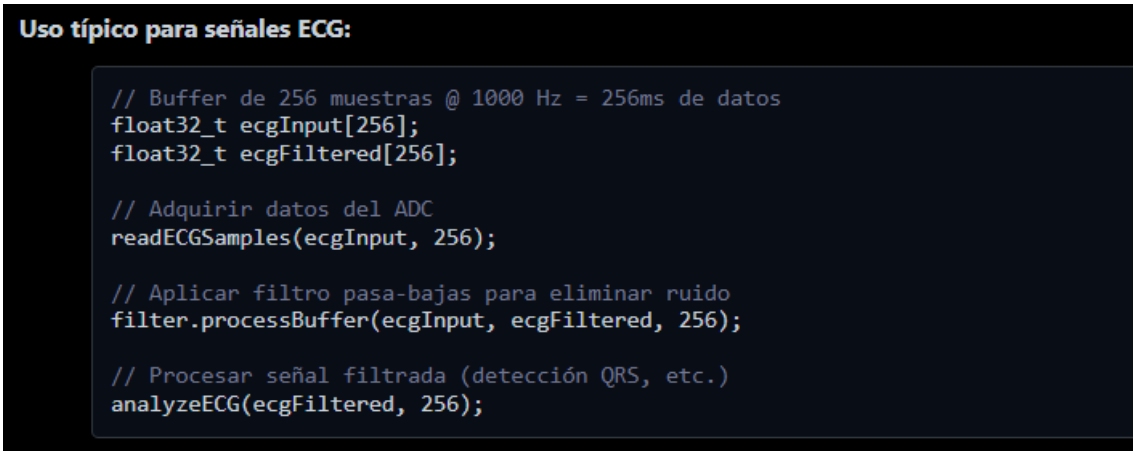

Figura 11. Ejemplo de uso de filtro FIR en la documentación. Fuente: https://sergiomoreno1060.github.io/BioFilterLib/

2. **Tutoriales paso a paso**: Para cada funcionalidad principal se realizaron archivos tutoriales de Arduino que guían al usuario desde configuraciones básicas hasta implementaciones avanzadas, con explicaciones detalladas de los parámetros y su efecto en el resultado.

3. **Uso de herramientas externas para documentación técnica:** La documentación, que involucra explicaciones de las variables de cada instancia, de las variables de entrada y salida y sus tipos de datos, generación de diagramas y referencias cruzadas fue generada con la herramienta Doxygen, una herramienta de documentación de código que genera automáticamente documentación técnica a partir de comentarios especiales dentro del código fuente.

3.1.3 Consideraciones de rendimiento y optimización

El desarrollo de BioFilterLib para Arduino Due se fundamentó en la utilización de la biblioteca CMSIS-DSP, aprovechando sus implementaciones optimizadas para la arquitectura Cortex-M3. La estrategia principal consistió en crear una capa de abstracción que encapsulara las funciones CMSIS manteniendo el acceso directo a sus características de rendimiento.

3.1.3.1 Aprovechamiento de optimizaciones CMSIS-DSP

BioFilterLib actúa como wrapper de las funciones *arm_fir_f32(), arm_biquad_cascade_df1_f32() y arm_lms_norm_f32()* de CMSIS-DSP. Estas funciones están implementadas específicamente para procesadores ARM Cortex-M mediante técnicas como desenrollado de bucles para reducir overhead de iteración, uso eficiente del pipeline del procesador y acceso optimizado a memoria aprovechando el ancho de bus. Al utilizar estas funciones como núcleo computacional, BioFilterLib hereda estas optimizaciones sin necesidad de reimplementarlas.

3.1.3.2 Gestión de memoria y minimización del overhead

El diseño del wrapper se orientó a introducir la mínima latencia posible sobre las funciones CMSIS nativas. Las estructuras de estado (arm_fir_instance_f32, arm_biquad_casd_df1_inst_f32, arm_lms_norm_instance_f32) se inicializan una sola vez y se reutilizan en sucesivas operaciones de procesamiento. Sus buffers de estado se preasignan con el tamaño mínimo que cada función requiere —calculado en función del orden del filtro y del número de etapas biquad— y se accede a ellos mediante punteros directos, admitiendo el procesamiento in-place cuando la operación lo permite, para evitar copias innecesarias de datos. Las validaciones de parámetros y comprobaciones de error se concentran en la fase de inicialización, por lo que la ruta de procesamiento (process) no cuenta con verificaciones adicionales; los métodos de acceso simple se declaran inline y las llamadas a CMSIS se realizan sin capas intermedias adicionales. Finalmente, los métodos destructores liberan de forma explícita los recursos para prevenir fugas de memoria durante el ciclo de vida de las instancias.

3.1.3.3. Portabilidad a otros núcleos ARM Cortex-M

Dado que BioFilterLib se basa en CMSIS-DSP, su portabilidad a otros microcontroladores ARM Cortex-M no requiere modificar el código de la librería, sino únicamente la configuración de compilación de CMSIS para el núcleo objetivo. La selección del núcleo se realiza mediante macros del preprocesador asociadas a arm_math.h (por ejemplo ARM_MATH_CM3, ARM_MATH_CM4 o ARM_MATH_CM7) y, cuando el núcleo dispone de FPU, mediante su activación (__FPU_PRESENT / __FPU_USED y las banderas del compilador -mfpu y -mfloat-abi=hard).

3.2. Análisis comparativo

Para evaluar el desempeño de la librería BioFilterLib en condiciones representativas de la práctica biomédica, se diseñó un experimento de filtrado sobre tres escenarios. Cada

escenario está asociado a una señal fisiológica distinta y a un tipo de perturbación característico de su adquisición:

- **Escenario A: Electrocardiograma (ECG)** contaminado con interferencia de red eléctrica de 60 Hz, se aborda mediante filtrado de rechazo de banda (notch).
- **Escenario B: Electroencefalograma (EEG)** contaminado con ruido rosa (densidad espectral 1/f), abordado mediante filtrado pasa-altos.
- **Escenario C: Electromiograma (EMG)** contaminado con ruido blanco de banda ancha y se aborda mediante filtrado pasa-banda.

La relación señal-ruido (SNR) de entrada se fijó en un valor de 5 dB para los todos los escenarios, de modo que las comparaciones de calidad de filtrado fueran homogéneas entre señales y algoritmos. El desempeño de BioFilterLib se comparó frente al de CMSIS-DSP, evaluada directamente, sin la capa de abstracción que representa la librería BioFilterLib. Esto permite cuantificar el sobrecosto introducido.

Todas las señales comparten una frecuencia de muestreo de 960 Hz y una duración de 3 segundos. La plataforma destino es la placa Arduino Due (microcontrolador ARM Cortex-M3 a 84 MHz, DAC de 12 bits), con procesamiento en aritmética de punto flotante de precisión simple (float32).

3.2.1 Generación y síntesis de las señales

Las señales limpias se sintetizaron en Python (version 3.12.12) mediante la librería Neurokit2 (versión 0.2.10). La señal ECG se simuló con *ecg_simulate* (método ecgsyn, frecuencia cardíaca de 70 lpm), el EEG con *eeg_simulate* (con los parámetros por defecto de neurokit2) y el EMG con *emg_simulate* (3 ráfagas de activación muscular, burst_number = 3). Las perturbaciones se generaron de forma independiente según el escenario:

- Interferencia de 60 Hz (Escenario A): tono sinusoidal determinista a 60 Hz.
- Ruido rosa (Escenario B): ruido de densidad espectral 1/f ($\beta = 1$) generado con la librería Colorednoise (powerlaw_psd_gaussian).
- Ruido blanco (Escenario C): ruido gaussiano de banda ancha (np.random.randn).

Para garantizar reproducibilidad se fijó una semilla aleatoria común y se descartó un tramo inicial de muestras para eliminar el transitorio generado por los modelos matemáticos de síntesis.

3.2.1.1 Control de la relación señal-ruido a 5 dB

La inyección de ruido se realizó con potencia controlada para imponer un SNR de entrada de 5 dB. Tras centrar en media cero tanto la señal limpia como el ruido, se calculó la potencia de cada uno (Ecuacion 5) y se escaló el ruido por un factor que ajusta su potencia al valor objetivo (Ecuacion 6) de forma que la señal ruidosa queda como se describe en la Ecuacion 7:

$$\boldsymbol{P_{objetivo}} = \frac{\boldsymbol{P_{señal}}}{\boldsymbol{10^{SNR_{dB}\cdot 0.1}}} \qquad (5)$$

$$Factor = \sqrt{\frac{P_{objetivo}}{P_{ruido}}} \qquad (6)$$

$$x_{noisy} = x_{clean} + Factor \cdot x_{noise} \qquad (7)$$

De este modo la SNR de entrada queda explícita y verificable para los tres escenarios.

3.2.1.2 Codificación al DAC

Cada señal generada fue procesada mediante los pasos descritos en la figura 12:

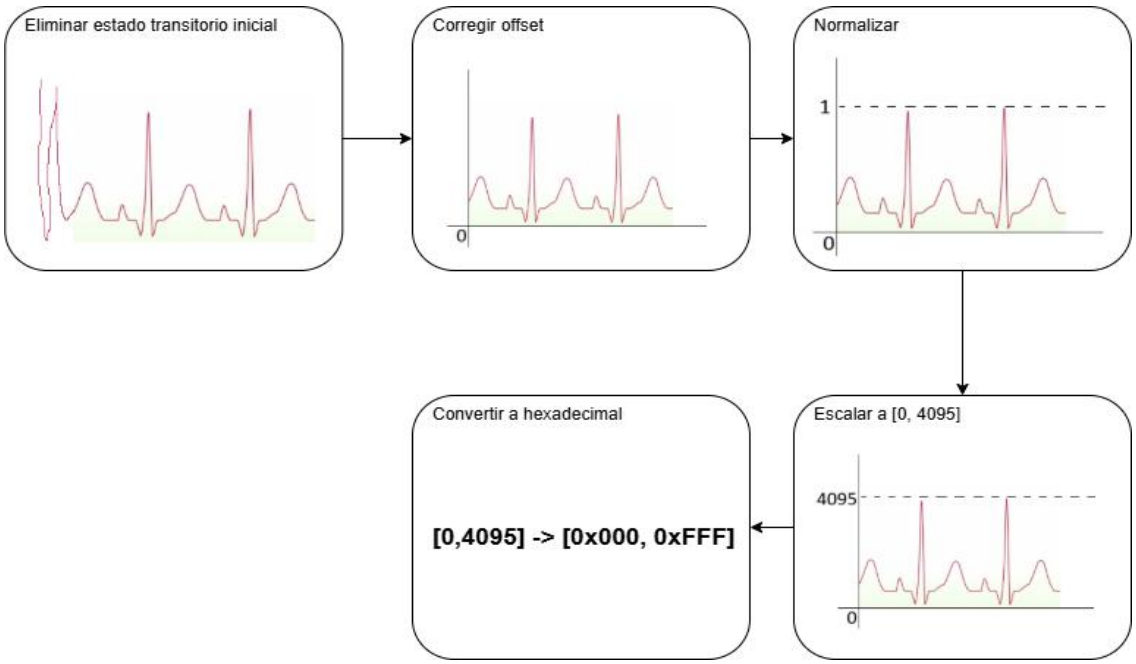

Figura 12. Preprocesamiento de la señal de prueba. Fuente: Elaboración propia

a) **Eliminación de transitorio inicial**: Se descartaron las primeras 100 muestras donde el modelo de simulación presenta inestabilidad numérica.

b) **Corrección de offset**: Se desplazó la señal realizando un mismo desplazamiento offset = 2048 para eliminar componentes negativos.

c) **Normalización**: Se normalizó la amplitud al rango [0, 1]

d) **Escalamiento para DAC**: Se escaló al rango [0, 4095] correspondiente a la resolución de 12 bits del conversor digital-analógico del Arduino Due.

e) **Conversión a formato hexadecimal**: Los valores se convirtieron a representación hexadecimal de 12 bits y se formatearon para su almacenamiento en el archivo *Waveforms.h* como arrays constantes de C++.

Cada escenario produjo así un conjunto de señales tabuladas en flash; en total se exportaron diez vectores (señal limpia, señal ruidosa y ruido para cada escenario, más una

referencia sinusoidal pura de 60 Hz para el filtro adaptativo del Escenario A), según se resume en la Tabla 2.

Tabla 2. Señales disponibles en la librería BioFilterLib

Etiqueta	Contenido	Uso
ecg_clean	ECG limpio	Referencia para métricas
ecg_60hz_noisy	ECG + 60 Hz @ 5 dB	Señal de entrada a filtrar
ecg_60hz_noise	Ruido de 60 Hz exacto	Referencia NLMS
ref_60hz	Seno puro de 60 Hz	Referencia NLMS (Escenario A)
eeg_clean	EEG limpio	Referencia para métricas
eeg_pink_noisy	EEG + ruido rosa @ 5 dB	Señal de entrada
eeg_pink_noise	Ruido rosa exacto	Referencia NLMS
emg_clean	EMG limpio	Referencia para métricas

Las señales se almacenaron en el archivo *Waveforms.h* como arrays constantes en memoria flash del Arduino Due. Este archivo forma parte del repositorio de BioFilterLib. Cada array contiene 2,780 muestras (correspondientes a 3 segundos de señal después del preprocesamiento), con valores representados en formato hexadecimal de 12 bits.

3.2.2. Librerías y filtros evaluados

Se compararon las implementaciones de BioFilterLib (procesamiento por bloques) frente a CMSIS-DSP evaluada directamente. En el Escenario A, por ser el de mayor interés comparativo, se incluyeron además cuatro librerías FIR de terceros de uso común en Arduino: Leeman (LeemanGeophysicalLLC/FIR_Filter_Arduino_Library), Nilsson (FIR-Filter-Arduino-Library), Ttapa (Arduino-Filters) y Moarbue (FIR-Filter).

Estas librerías procesan muestra a muestra, mientras que BioFilterLib y CMSIS-DSP operan por bloques.

Los coeficientes de los filtros FIR e IIR se diseñaron en Python con scipy.signal y se exportaron como cabeceras C. Los filtros IIR se implementaron como cascadas de secciones de segundo orden (biquads) en el formato {b0, b1, b2, −a1, −a2} que exigen CMSIS-DSP y BioFilterLib. Además de los filtros lineales, se evaluaron dos algoritmos propios de BioFilterLib: el filtro adaptativo NLMS y el filtro wavelet por umbralización. La configuración por escenario se resume en la Tabla 3.

Tabla 3. Filtros y parámetros de configuración por escenario.

Escenario	Filtro	Configuración
Escenario A: ECG + 60 Hz	FIR notch	151 coeficientes (banda de rechazo en 60 Hz)
	IIR notch	1 biquad, Q ≈ 30
	NLMS	referencia = seno puro 60 Hz; 64 coeficientes; μ = 0,01
	Wavelet Db4	4 niveles; anulación de detalles cD3 y cD4 (banda de 60 Hz)
Escenario B: EEG + rosa	FIR pasa-altos	201 coeficientes; fc ≈ 4 Hz
	IIR pasa-altos	Butterworth orden 4 (2 biquads)
	NLMS	referencia = ruido rosa; 8 coeficientes; μ = 0,005
	Wavelet Db4	5 niveles; umbralización suave VisuShrink
Escenario C: EMG + blanco	FIR pasa-banda	101 coeficientes; banda 20–450 Hz
	IIR pasa-banda	Butterworth orden 4 (2 biquads)

	NLMS	referencia = ruido blanco; 8 coeficientes; μ = 0,005
	Wavelet Db4 y Db8	4 niveles; umbralización suave VisuShrink

El filtro NLMS se evaluó muestra a muestra. En el Escenario A se empleó la configuración de cancelación clásica, tomando como referencia el tono sinusoidal puro de 60 Hz; la salida limpia se obtuvo restando la estimación del filtro a la señal contaminada. En los Escenarios B y C se utilizó como referencia el ruido exacto exportado (ruido rosa y ruido blanco, respectivamente). En todos los casos se descartó un tramo inicial de muestras para excluir el transitorio de convergencia antes del cálculo de métricas.

Finalmente, el filtro wavelet de BioFilterLib se basa en la familia Daubechies. En el Escenario A se realizó una descomposición de 4 niveles con anulación directa de los coeficientes de detalle que contienen la banda de 60 Hz. En los Escenarios B y C se aplicó umbralización suave (soft-thresholding) con el umbral universal de VisuShrink, calculado como en la Ecuacion 8,

$$thr = \sigma \cdot \sqrt{(2 \cdot ln(N)} \quad (8)$$

, donde σ se estima de forma robusta a partir de la desviación absoluta mediana (MAD) de los coeficientes de detalle de primer nivel (σ = MAD / 0,6745). El Escenario C comparó adicionalmente Db4 (8 coeficientes) frente a Db8 (16 coeficientes) para evaluar el efecto del orden de la wavelet sobre el costo computacional y la calidad.

3.2.2.1 Protocolo de medición

Para cada combinación de escenario, librería y filtro se generaron dos tipos de sketch sobre el Arduino Due:

- Sketches de visualización (Serial_*): Realizan un print por el puerto serie, a 115 200 baudios, pares «entrada salida» con cuatro decimales para su representación

en el Serial Plotter del IDE de Arduino. La señal de salida se alinea temporalmente con la de entrada compensando el retardo de grupo del filtro.

- Sketches de métricas (Test_*): Realizan el cálculo y muestran los resultados en una línea CSV con todas las métricas de desempeño y calidad descritas a continuación.

Cada sketch de métricas se compiló y cargó en el Arduino Due, capturándose su salida por el monitor serie. Las líneas CSV resultantes se consolidaron en tres tablas de resultados, una por escenario, con las siguientes columnas: librería, filtro y modo de procesamiento, tiempo por muestra, uso de CPU, mejora de SNR, MSE y correlación. Las salidas de los sketches de visualización se registraron como capturas del Serial Plotter para el análisis cualitativo de la forma de onda filtrada.

3.2.3. Métricas de evaluación

Las siguientes métricas se calcularon automáticamente en cada sketch de prueba utilizando funciones auxiliares implementadas en BioFilterLib.

3.2.3.1. Tiempo de procesamiento

El tiempo de procesamiento se midió con precisión de microsegundos utilizando la función micros() de Arduino Due antes y después de ejecutar la función principal de procesamiento para el total de las señales de prueba.

Las métricas derivadas incluyen:

- Tiempo por muestra: Esta métrica se calcula midiendo el tiempo total de ejecución requerido para procesar el total de muestras, y dividiendo este valor entre el número de muestras procesadas.

3.2.3.2. Uso de CPU

El porcentaje de uso de CPU se calculó comparando el tiempo de procesamiento real con el tiempo disponible en escenario de tiempo real:

Esta métrica indica qué porcentaje del tiempo disponible entre muestras consecutivas se consume en procesamiento, siendo valores <100% indicativos de viabilidad para procesamiento en tiempo real.

3.2.3.3. Calidad de filtrado

La evaluación de calidad se realizó mediante análisis cuantitativo comparando la señal filtrada con una señal de referencia sin ruido. Las métricas implementadas fueron:

Relación Señal-Ruido (SNR)

Se calculó tanto el SNR de entrada (señal limpia vs señal con ruido) como el SNR de salida (señal limpia vs señal filtrada), reportándose la mejora de SNR como la diferencia entre ambos valores (Ecuación 5).

$$SNR_{dB} = 10 \cdot log_{10}(\frac{P_s}{P_n}) \quad (9)$$

Donde P_s es la potencia de la señal limpia y P_n es la potencia de la señal con ruido.

Error Cuadrático Medio (MSE)

El MSE cuantifica la diferencia promedio entre la señal filtrada y la señal de referencia limpia (Ecuación 6). Valores menores de MSE indican mayor similitud entre las señales y, por tanto, mejor desempeño del filtro.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 \quad (10)$$

Donde n es el número total de muestras Y es la señal sin ruido de referencia y $\hat{Y}$ es la señal filtrada.

Coeficiente de correlación de Pearson

La correlación mide la similitud lineal entre la señal filtrada y la señal de referencia (Ecuación 7). Un valor cercano a 1 indica alta correlación positiva y preservación óptima de la morfología de la señal.

$$r = \frac{n\sum x \cdot y - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2}} \quad (11)$$

Donde n es el número de muestras, x es la señal de referencia y y es la señal filtrada.

Valor RMS (Root Mean Square)

Calculado para caracterizar la amplitud efectiva de cada señal y detectar posibles alteraciones de ganancia introducidas por el filtrado.

$$RMS = \sqrt{\frac{1}{n}\sum_i x_i^2} \quad (12)$$

Donde n es el número de muestras y x corresponde a las muestras de la señal.

3.3. Operacionalización de las variables

Tabla 4. Operacionalización de las variables

Variable	Definición conceptual	Definición Operacional	Unidad de Medida	Tipo y Escala	Rango
Tipo de filtro de Librería (Independiente)	Conjunto de algoritmos de filtro digital provistos por una librería de software para el procesamiento de señales	Identificación del tipo de filtro (FIR/IIR/Butterworth/ Chebyshev/etc.) implementado en las muestras de código extraídas de cada librería	Categorías de filtros (FIR, IIR, etc.)	Categórica Nominal	No aplica
Lenguaje de programación (Independiente)	Sistema formal compuesto por un conjunto de reglas sintácticas y semánticas que especifican instrucciones que una computadora puede ejecutar para realizar diversas tareas	Identificación del tipo de lenguaje de programación utilizado para la librería dependiendo si está diseñada para Arduino Due u otra placa	Categorías de lenguaje de programación (C++ o Python)	Categórica Nominal	No aplica

Frecuencia de reloj (Independiente)	Rapidez con la que un sistema computacional ejecuta una tarea o procesa datos	Medición de la frecuencia de reloj y/o velocidad de ejecución de operaciones de punto flotante en ciclos de reloj para cada muestra de código de filtro	Ciclos de reloj por segundo (Hz)	Cuantitativa Continua De Razón	Según las capacidades de la placa Arduino Due
Tiempo de procesamiento (Dependiente)	Tiempo total de procesamiento digital del software embebido	Medición del tiempo que toma una muestra de señal en entrar y salir de la función de filtrado	Microsegundos por muestra (us/muestra)	Cuantitativa Continua De Razón	Por determinar experimental mente
Error cuadrático medio MSE (Dependiente)	Grado en que el proceso de filtrado logra eliminar el ruido o componentes no deseados de la señal sin distorsionar la información de interés.	Comparación entre la señal filtrada y una señal de referencia libre de ruido usando la métrica del error cuadrático medio (MSE)	Adimensional	Cuantitativa Continua De Razón	≥ 0

Correlación (Dependiente)	Medida estadística que cuantifica el grado de similitud o relación lineal entre la señal filtrada y la señal original de referencia.	Cálculo del coeficiente de correlación de Pearson (r) entre el vector de la señal de salida del filtro y el vector de la señal deseada (limpia).	Adimensional (Coeficiente r)	Cuantitativa Continua De Razón	-1 a 1
Relación Señal-Ruido (SNR) (Dependiente)	Proporción existente entre la potencia de la señal que se quiere transmitir y la potencia del ruido que la corrompe.	Medición del logaritmo de la razón entre la potencia de la señal de referencia y la potencia del ruido residual tras el filtrado.	Decibelios (dB)	Cuantitativa Continua De Razón	> 0 dB
Raíz Cuadrática Media (RMS) (Dependiente)	Valor eficaz de una señal que representa la potencia promedio de la forma de onda en un intervalo determinado.	Cálculo de la raíz cuadrada del promedio de los cuadrados de los valores instantáneos de la señal filtrada en una ventana de tiempo.	Adimensional	Cuantitativa Continua De Razón	Según rango dinámico del ADC (0-3.3V)

Uso de CPU (Dependiente)	Proporción del tiempo empleado por el microcontrolador en ejecutar una tarea específica, en relación al periodo de muestreo de una señal entrante	Medición del porcentaje del intervalo entre muestras que está ocupado por la ejecución del filtro digital implementado en BioFilterLib	Porcentaje (%) de uso de CPU	Cuantitativa Continua De Razón	0 a 100%

IV. Resultados

Esta sección presenta el desempeño medido sobre el Arduino Due para los tres escenarios de filtrado. Para cada combinación de librería y filtro se reportan el tiempo de procesamiento por muestra, el uso de CPU a 960 Hz, la mejora de la relación señal-ruido (SNR), el error cuadrático medio (MSE) y la correlación frente a la señal de referencia, y el valor eficaz (RMS) de la señal filtrada. La SNR de entrada fue de aproximadamente 5 dB en todos los casos. Tras la tabla de cada escenario se incluyen las representaciones en el dominio temporal y en frecuencia de cada filtro.

4.1 Escenario A – ECG con interferencia de 60 Hz (notch)

Tabla 5. Resultados del Escenario A (ECG + 60 Hz, fs = 960 Hz, SNR de entrada ≈ 5 dB).

Librería	Filtro	Tiempo (µs/muestra)	CPU (%)	Mejora SNR (dB)	MSE	Correlación	RMS filtrada
BioFilterLib	FIR notch (por bloques)	294,73	28,29	23,29	0,000046	0,9993	0,1803
BioFilterLib	IIR notch (por bloques)	8,40	0,81	10,87	0,000877	0,9877	0,1890
BioFilterLib	LMS adaptativo (ref. 60 Hz)	290,05	27,84	12,65	0,000583	0,9912	0,1829
BioFilterLib	Wavelet Db4 (4 niveles)	78,08	7,50	17,20	0,000204	0,9968	0,1792
CMSIS-DSP	FIR notch (por bloques)	322,42	30,95	23,29	0,000046	0,9993	0,1803
CMSIS-DSP	IIR notch (por bloques)	8,40	0,81	10,87	0,000877	0,9877	0,1890
Leeman	FIR notch	360,38	34,60	19,29	0,000130	0,9981	0,1835
Moarbue	FIR notch	317,01	30,43	26,71	0,000023	0,9997	0,1835
Nilsson	FIR notch	351,74	33,77	26,71	0,000023	0,9997	0,1835
Ttapa	FIR notch	306,37	29,41	26,71	0,000023	0,9997	0,1835
Ttapa	IIR notch	10,90	1,05	13,88	0,000439	0,9933	0,1816

4.1.1 Formas de onda y espectros de magnitud - Escenario A

Las figuras siguientes presentan, por tipo de filtro, los resultados sobre la señal de ECG. Cada mosaico reúne las librerías evaluadas para ese tipo; en cada panel se superpone la señal ruidosa (gris) sobre la filtrada (azul).

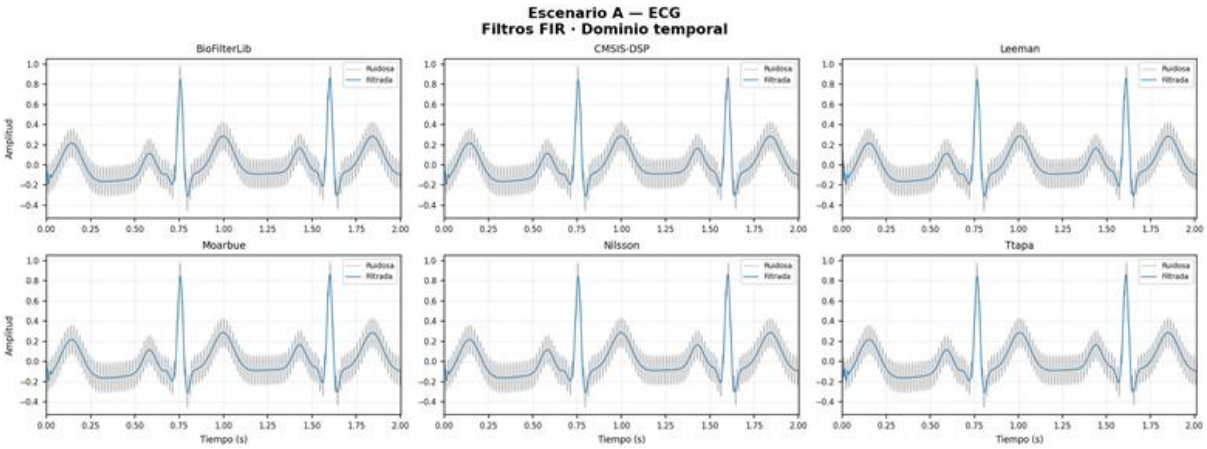

Figura 13. Escenario A — ECG, filtros FIR: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

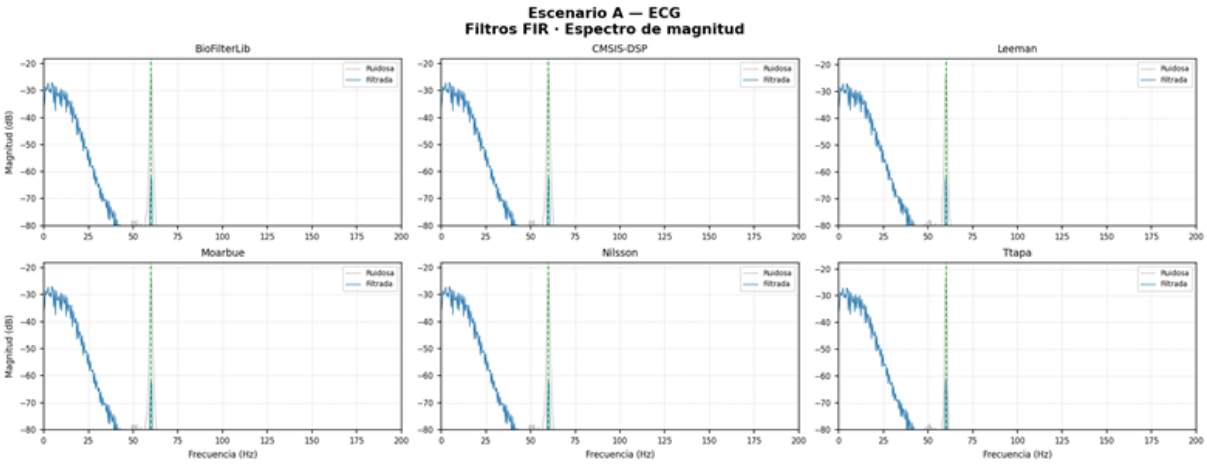

Figura 14. Escenario A — ECG, filtros FIR: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

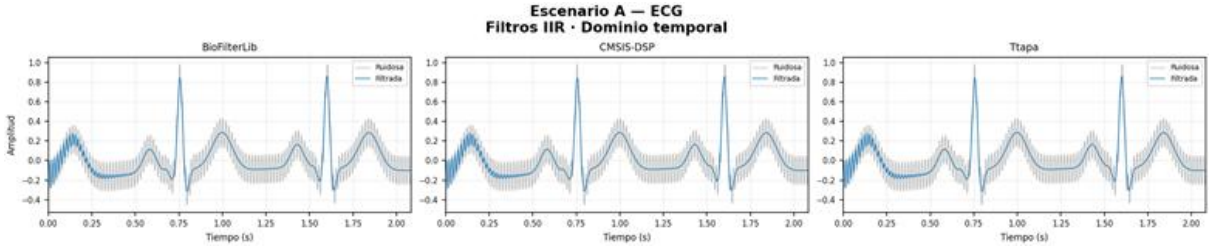

Figura 15. Escenario A — ECG, filtros IIR: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

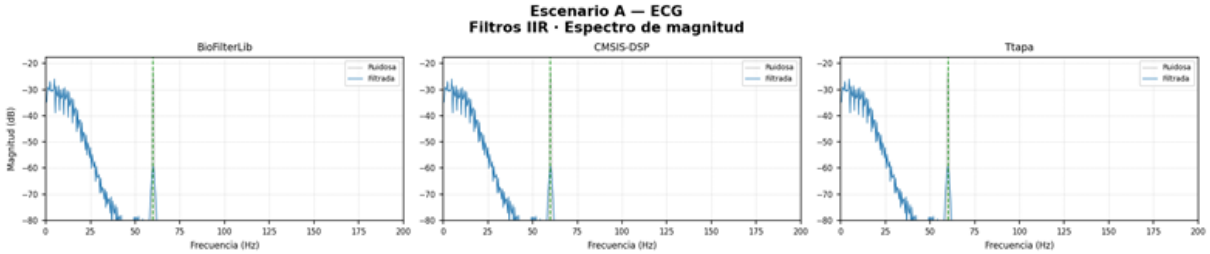

Figura 16. Escenario A — ECG, filtros IIR: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

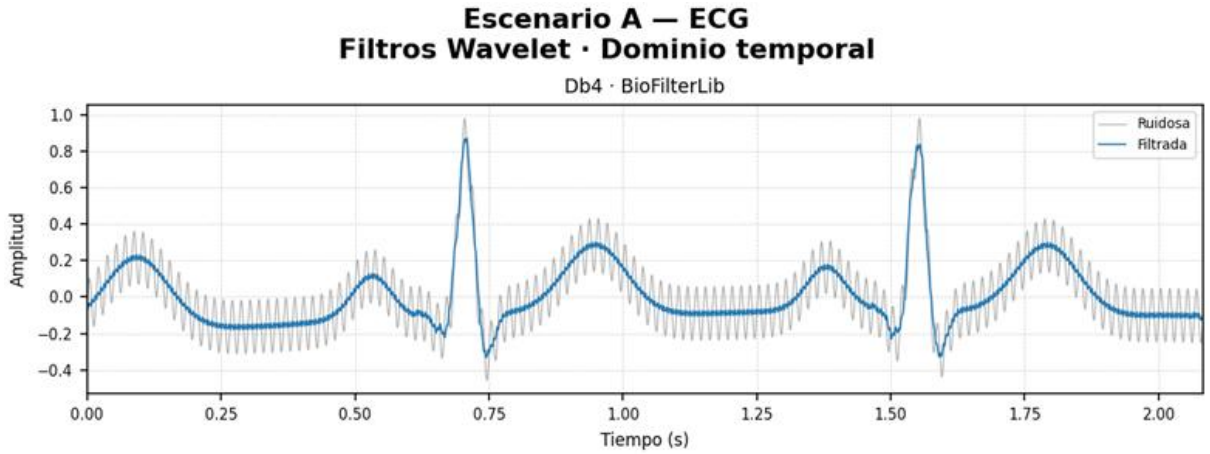

Figura 17. Escenario A — ECG, filtros Wavelet: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

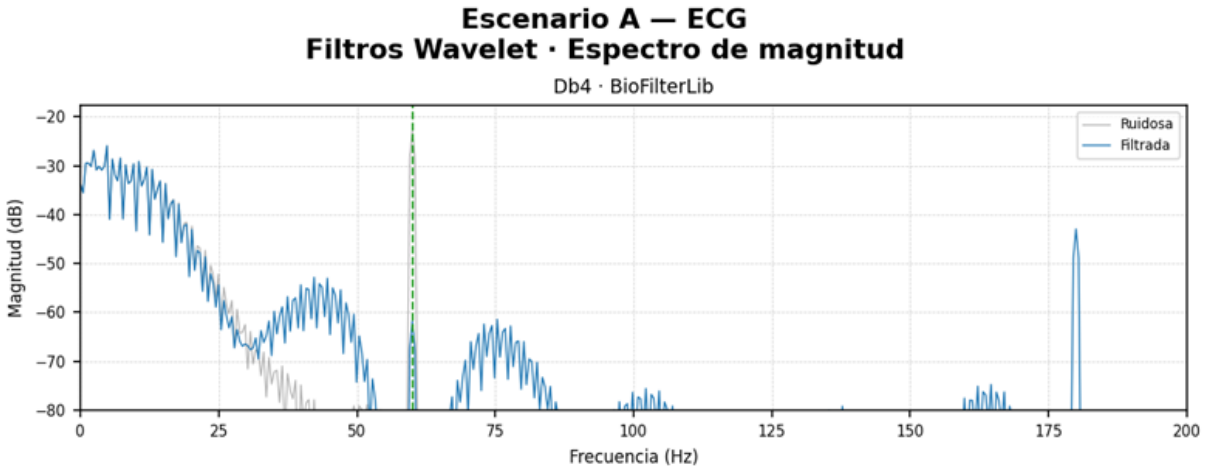

Figura 18. Escenario A — ECG, filtros Wavelet: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

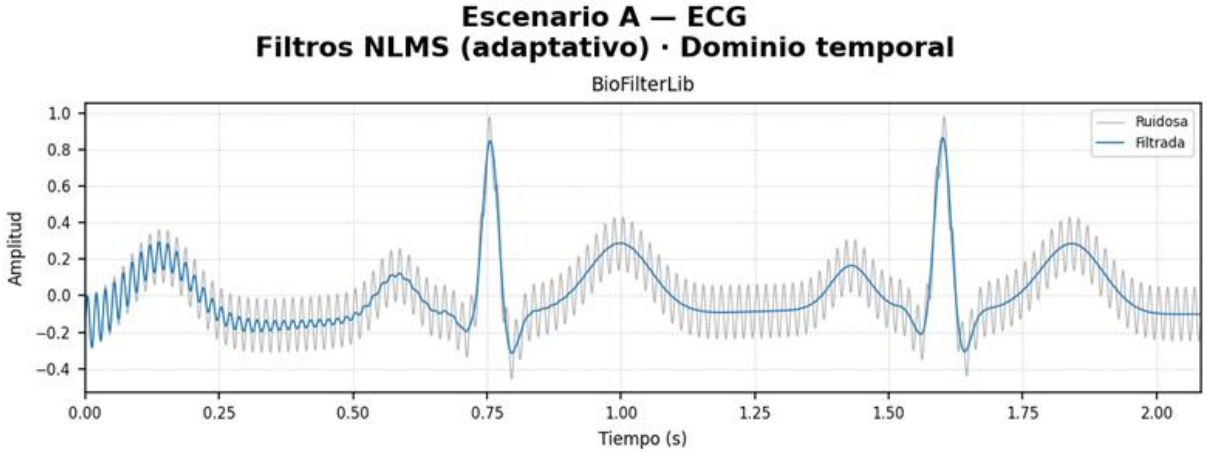

Figura 19. Escenario A — ECG, filtros NLMS adaptativo: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

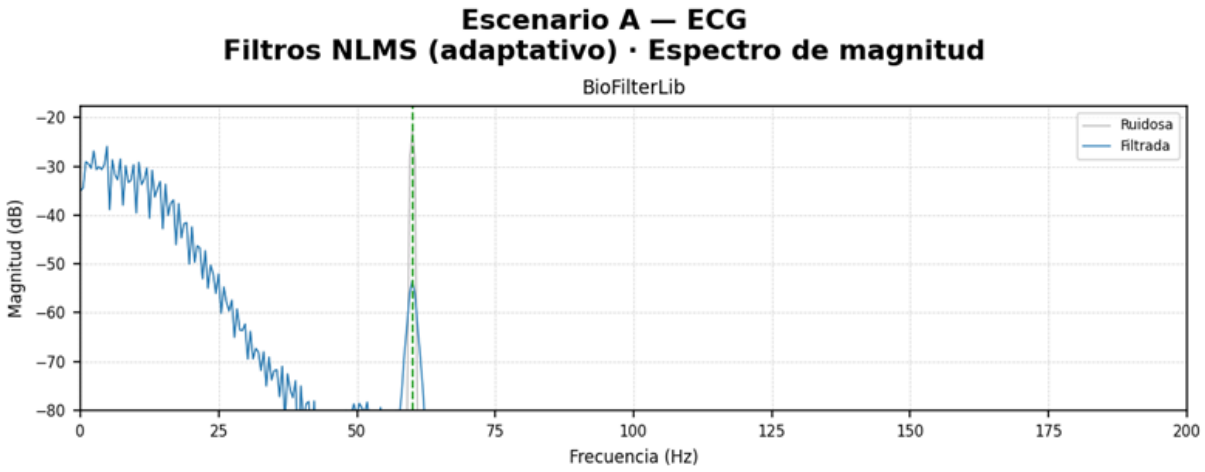

Figura 20. Escenario A — ECG, filtros NLMS adaptativo: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

4.2 Escenario B – EEG con ruido rosa (pasa-altos)

Tabla 6. Resultados del Escenario B (EEG + ruido rosa, fs = 960 Hz, SNR de entrada ≈ 5 dB).

Librería	Filtro	Tiempo (µs/muestra)	CPU (%)	Mejora SNR (dB)	MSE	Correlación	RMS filtrada
BioFilterLib	FIR pasa-altos (por bloques)	415,84	39,92	2,12	0,009281	0,9133	0,2363
BioFilterLib	IIR pasa-altos (por bloques)	18,27	1,75	2,56	0,009068	0,9204	0,2435
BioFilterLib	LMS adaptativo	48,09	4,62	8,18	0,002907	0,9755	0,2450
BioFilterLib	Wavelet Db4 (VisuShrink)	92,76	8,91	−1,33	0,024631	0,7649	0,2188
CMSIS-DSP	FIR pasa-altos (por bloques)	410,62	39,42	2,12	0,009281	0,9133	0,2363
CMSIS-DSP	IIR pasa-altos (por bloques)	16,79	1,61	2,56	0,009068	0,9204	0,2435

4.2.1 Formas de onda y espectros de magnitud – Escenario B

Las figuras siguientes presentan, por tipo de filtro, los resultados sobre la señal de EEG. Cada mosaico reúne las librerías evaluadas para ese tipo; en cada panel se superpone la señal ruidosa (gris) sobre la filtrada (azul).

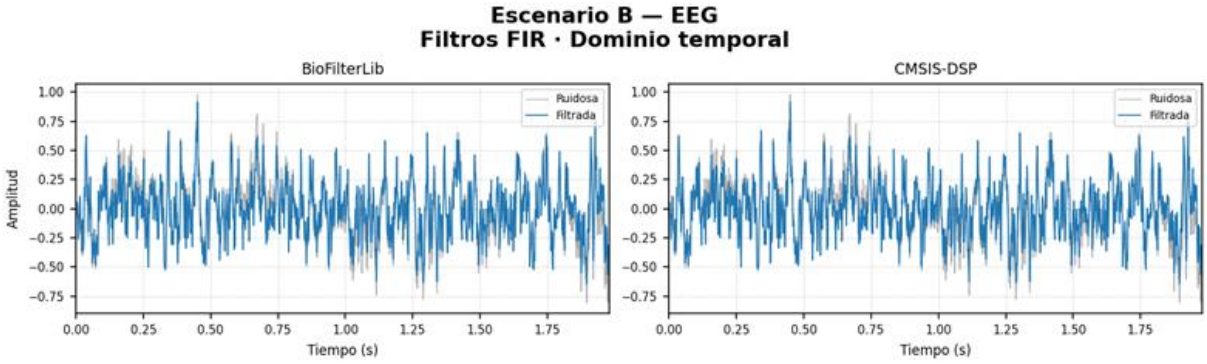

Figura 21. Escenario B — EEG, filtros FIR: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

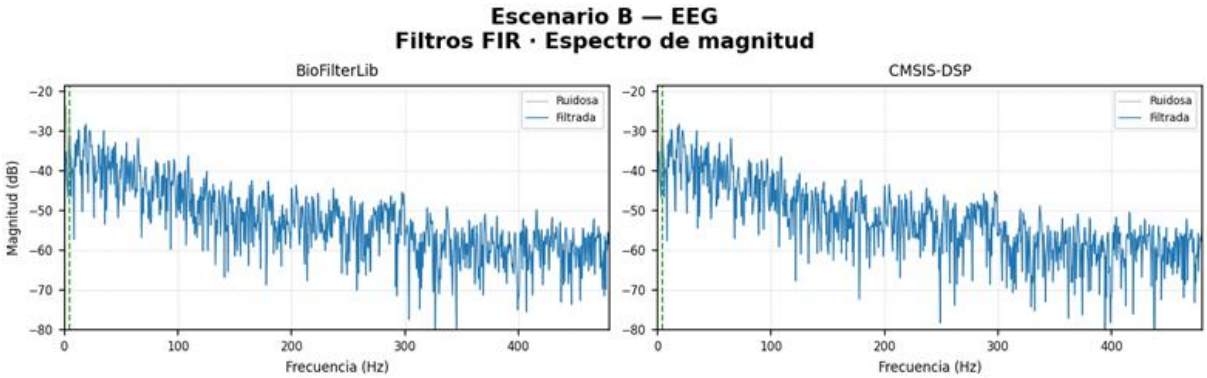

Figura 22. Escenario B — EEG, filtros FIR: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

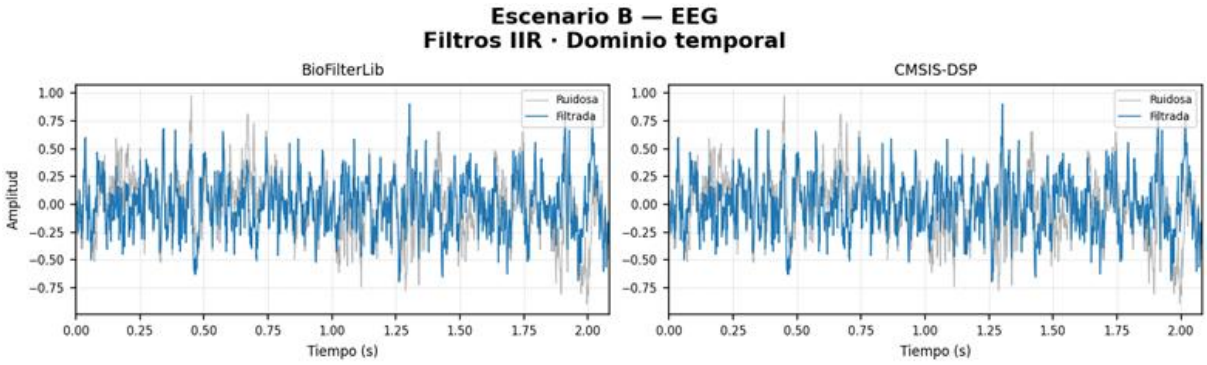

Figura 23. Escenario B — EEG, filtros IIR: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

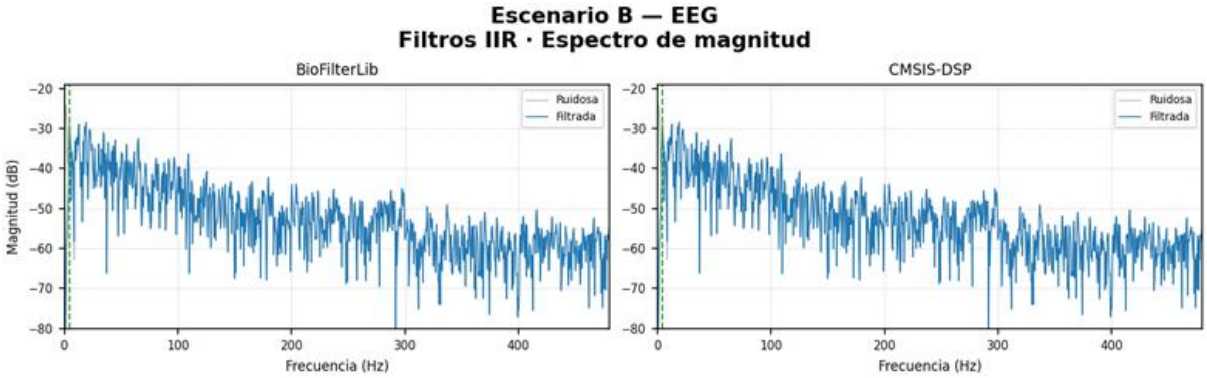

Figura 24. Escenario B — EEG, filtros IIR: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

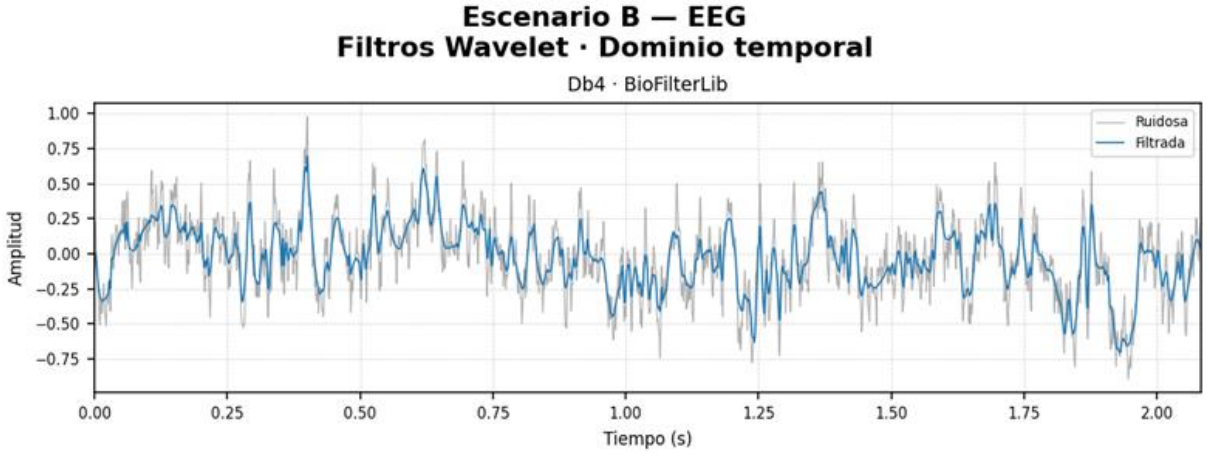

Figura 25. Escenario B — EEG, filtros Wavelet: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

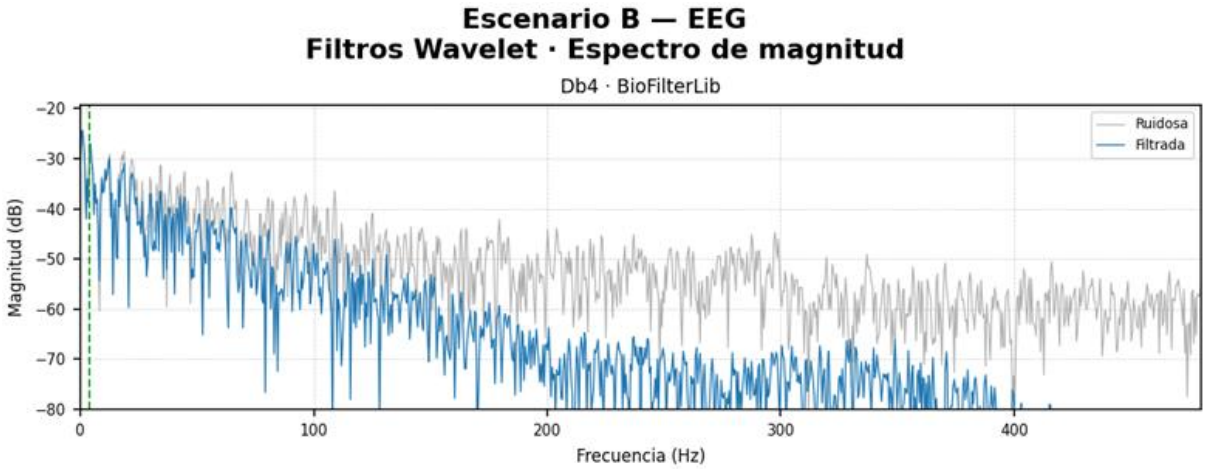

Figura 26. Escenario B — EEG, filtros Wavelet: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

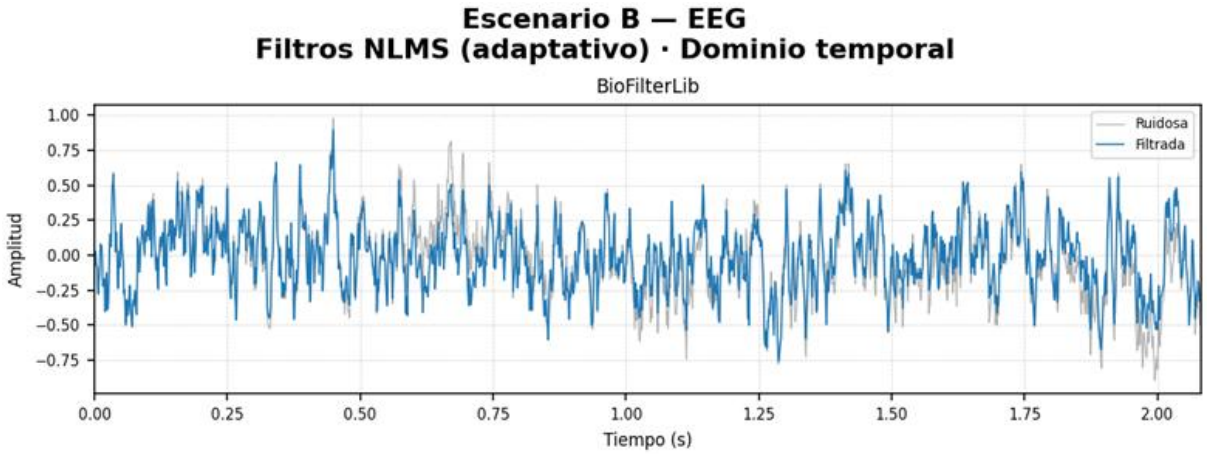

Figura 27. Escenario B — EEG, filtros NLMS adaptativo: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

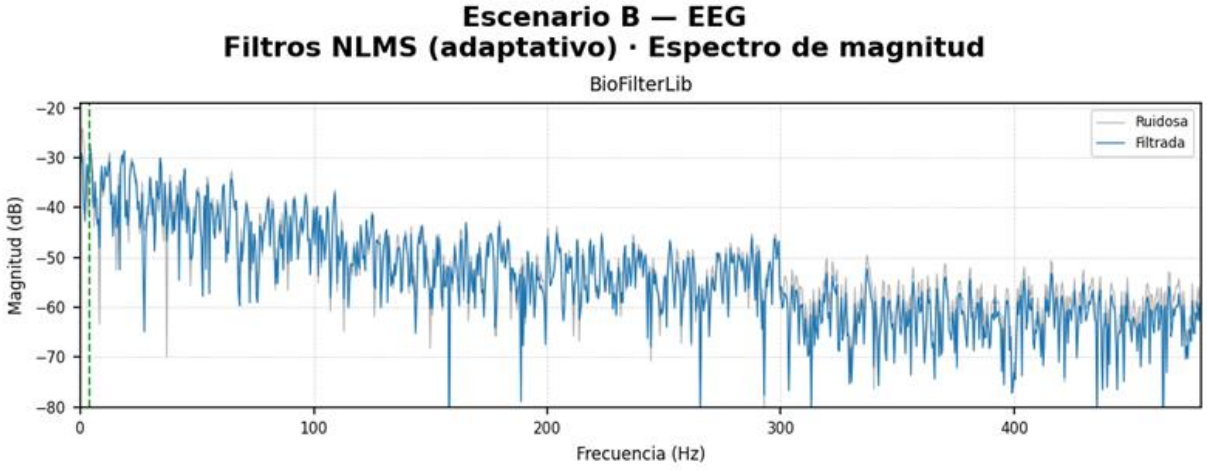

Figura 28. Escenario B — EEG, filtros NLMS adaptativo: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

4.3 Escenario C — EMG con ruido blanco (pasa-banda)

Tabla 7. Resultados del Escenario C (EMG + ruido blanco, fs = 960 Hz, SNR de entrada ≈ 5 dB).

Librería	Filtro	Tiempo (µs/muestra)	CPU (%)	Mejora SNR (dB)	MSE	Correlación	RMS filtrada
BioFilterLib	FIR pasa-banda (por bloques)	209,16	20,08	0,15	0,022555	0,8758	0,3112
BioFilterLib	IIR pasa-banda (por bloques)	18,19	1,75	0,07	0,023046	0,8737	0,3120
BioFilterLib	LMS adaptativo	48,21	4,63	5,22	0,007724	0,9561	0,2997
BioFilterLib	Wavelet Db4 (VisuShrink)	88,82	8,53	−4,73	0,076984	0,2229	0,0863
BioFilterLib	Wavelet Db8 (VisuShrink)	149,75	14,38	−4,73	0,076902	0,2213	0,0825
CMSIS-DSP	FIR pasa-banda (por bloques)	206,15	19,79	0,15	0,022555	0,8758	0,3112
CMSIS-DSP	IIR pasa-banda (por bloques)	16,73	1,61	0,07	0,023046	0,8737	0,3120

4.3.1 Formas de onda y espectros de magnitud - Escenario C

Las figuras siguientes presentan, por tipo de filtro, los resultados sobre la señal de EMG. Cada mosaico reúne las librerías evaluadas para ese tipo; en cada panel se superpone la señal ruidosa (gris) sobre la filtrada (azul).

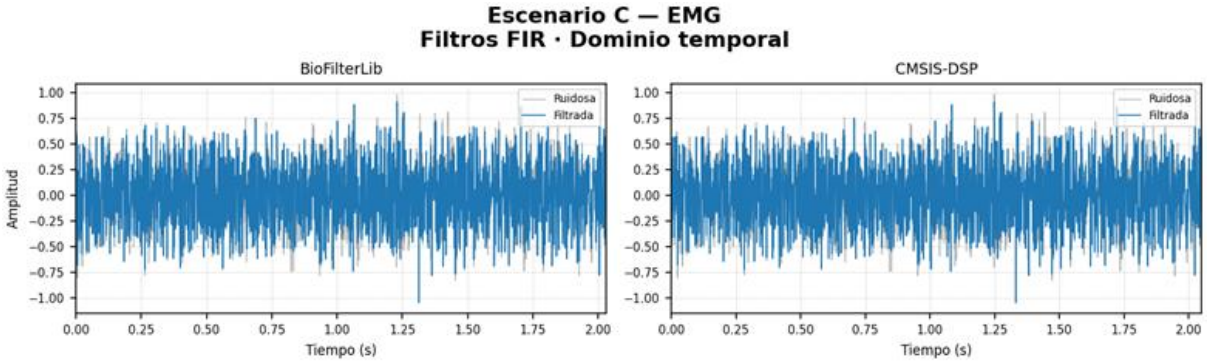

Figura 29. Escenario C — EMG, filtros FIR: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

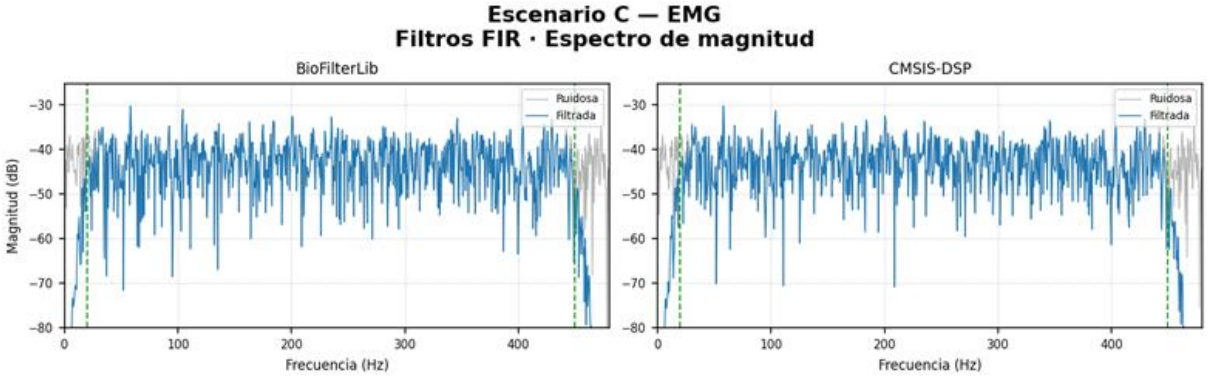

Figura 30. Escenario C — EMG, filtros FIR: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

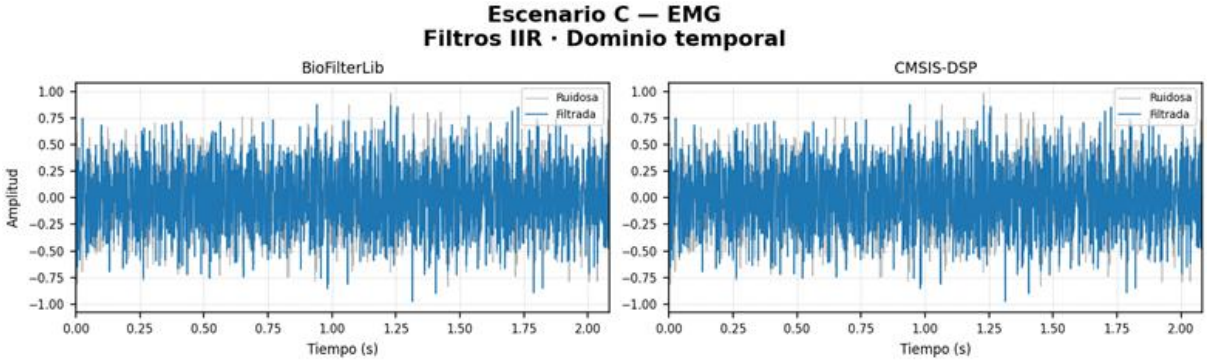

Figura 31. Escenario C — EMG, filtros IIR: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

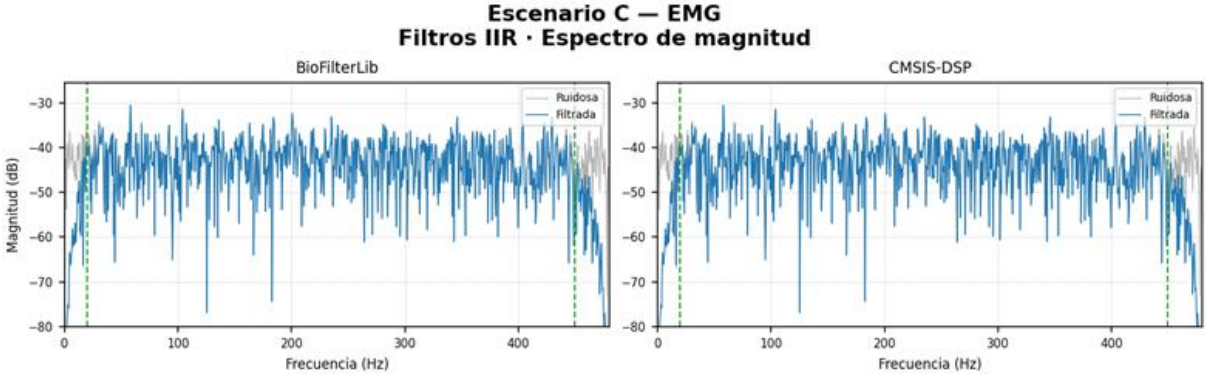

Figura 32. Escenario C — EMG, filtros IIR: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

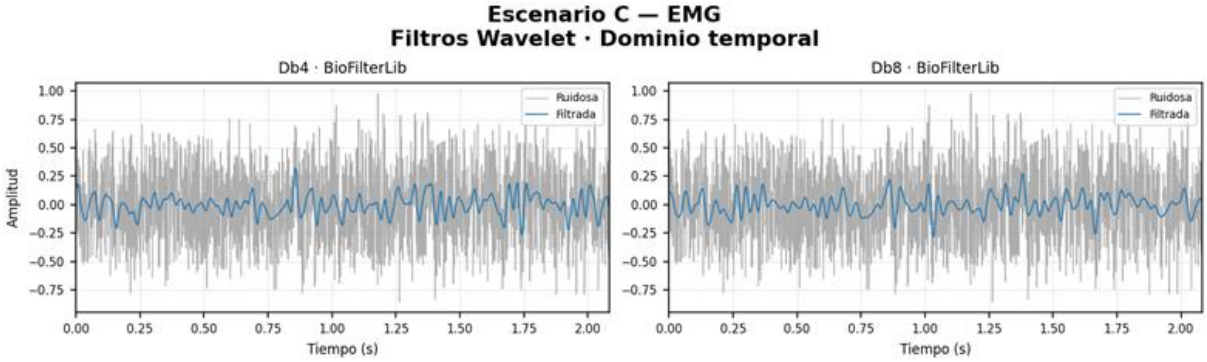

Figura 33. Escenario C — EMG, filtros Wavelet: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

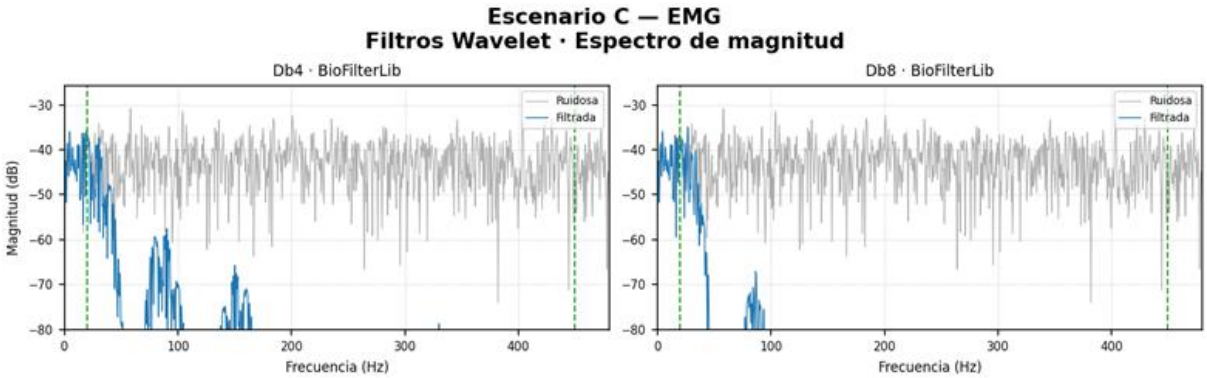

Figura 34. Escenario C — EMG, filtros Wavelet: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

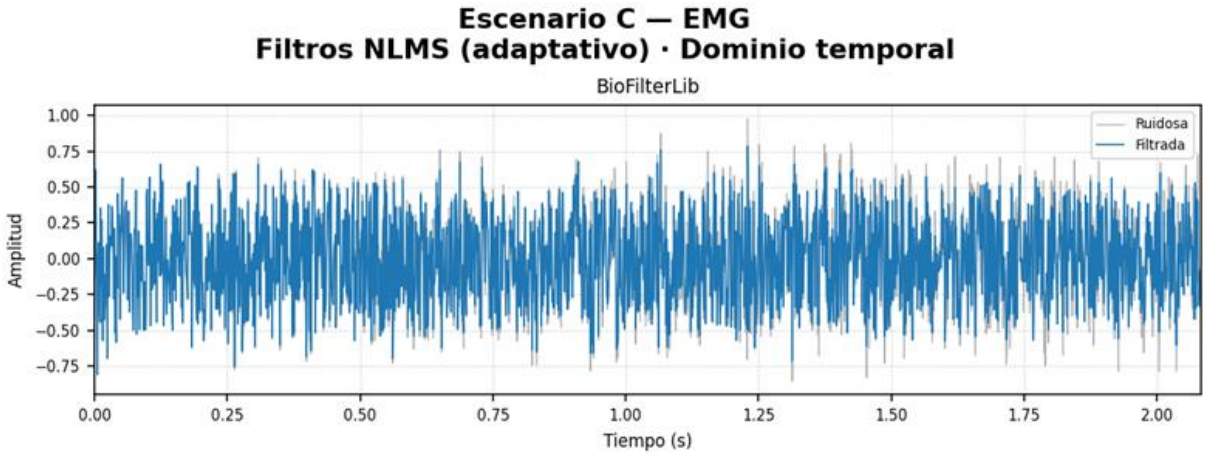

Figura 35. Escenario C — EMG, filtros NLMS adaptativo: señales ruidosa (gris) y filtrada (azul) en el dominio temporal.

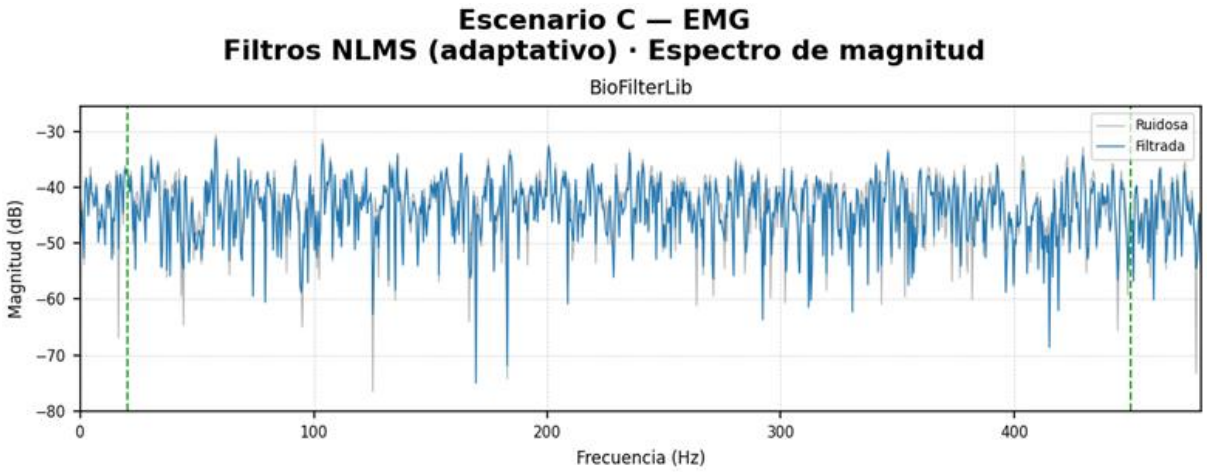

Figura 36. Escenario C — EMG, filtros NLMS adaptativo: espectro de magnitud de las señales ruidosa (gris) y filtrada (azul). Las líneas verticales discontinuas marcan las frecuencias de corte.

V. Discusión

El procesamiento digital de señales biomédicas en plataformas de hardware embebido representa un área de intersección entre la ingeniería biomédica, el desarrollo de software educativo y la implementación de algoritmos de procesamiento en sistemas con recursos limitados. Este estudio desarrolló y evaluó BioFilterLib, una librería de filtrado digital para Arduino Due optimizada mediante CMSIS-DSP, diseñada para contextos educativos y de investigación en el procesamiento de bioseñales.

La validación se realizó sobre tres escenarios de filtrado representativos de la práctica biomédica, cada uno con una señal fisiológica y un tipo de perturbación característico: electrocardiograma (ECG) contaminado con interferencia de red de 60 Hz, cuyo filtrado se realiza mediante rechazo de banda (Escenario A); señal de electroencefalograma (EEG) contaminado con ruido rosa, que se aborda mediante filtrado pasa-altos (Escenario B); y señal de electromiograma (EMG) contaminado con ruido blanco de banda ancha, abordado mediante filtrado pasa-banda (Escenario C). Las tres señales se generaron con un SNR de entrada fijado en 5 dB, una frecuencia de muestreo común de 960 Hz y una duración de 3 s, lo que permitió comparaciones uniformes entre librerías y arquitecturas de filtrado. Adicionalmente, CMSIS-DSP se incorporó directamente en el benchmark, sin la capa de abstracción de BioFilterLib, con el fin de cuantificar el sobrecosto introducido.

La evaluación comparativa mostró que, para los filtros lineales procesados por bloques, BioFilterLib y CMSIS-DSP obtuvieron métricas de calidad idénticas, con tiempos de procesamiento equivalentes dentro del margen de variabilidad de la medición. En el Escenario A, los filtros FIR notch de BioFilterLib (294.73 µs/muestra) y CMSIS-DSP (322.42 µs/muestra) resultaron más veloces en procesamiento que las implementaciones de terceros evaluadas muestra por muestra (de 306.37 a 360.38 µs/muestra). En términos de métricas de calidad estas fueron comparables (Tabla 5). Los filtros IIR exhibieron, en todos los escenarios, una eficiencia computacional muy superior a la de los FIR (de 8.40 a 18.27 µs/muestra frente a 209.16 - 415.84 µs/muestra). Un resultado importante fue la variabilidad de la eficacia del filtrado en función del tipo de ruido: mientras que la interferencia de banda estrecha (60 Hz) se atenuó eficazmente, el ruido de banda ancha y coloreado, solapado con la banda útil de la señal, limitó en gran medida al filtrado lineal y requirió métodos adaptativos. Las siguientes subsecciones analizan en detalle estos

hallazgos, su interpretación en el marco de los objetivos planteados y las implicancias que tienen para el desarrollo de herramientas educativas de DSP.

5.1 Análisis de rendimiento computacional y viabilidad técnica

El análisis del rendimiento computacional constituye un indicador de gran importancia para evaluar la factibilidad de implementación de algoritmos de filtrado digital en sistemas embebidos que presentan restricciones de recursos. Los resultados obtenidos revelan diferencias entre las arquitecturas de filtrado evaluadas, con implicancias directas para la selección de ciertos algoritmos en aplicaciones de procesamiento de bioseñales específicas en tiempo real.

Las estructuras IIR demostraron la mayor eficiencia computacional en los tres escenarios, con tiempos de procesamiento de 8.40 µs/muestra en el Escenario A (un solo biquad, filtro notch) y de aproximadamente 18 µs/muestra en los Escenarios B y C (Butterworth de orden 4, dos secciones biquad), correspondientes a un uso de CPU inferior al 2 % a 960 Hz (Tablas 5, 6 y 7). Esta ventaja se explica por la drástica disminución en el número de operaciones de multiplicación-acumulación por muestra, producto de la retroalimentación que es parte de estos sistemas. Por el contrario, los filtros FIR resultaron los más demandantes, con tiempos de 209.16 µs/muestra (pasa-banda, 101 coeficientes), 294.73 µs/muestra (notch, 151 coeficientes) y 415.84 µs/muestra (pasa-altos, 201 coeficientes), y un uso de CPU de entre 20 % y 40 %; el costo escala directamente cuando incrementa el número de coeficientes del filtro.

El filtro wavelet ocupó una posición intermedia en términos de velocidad de procesamiento (78.08 - 149.75 µs/muestra), mientras que el filtro adaptativo NLMS exhibió un costo que varía fuertemente con el incremento del número de coeficientes: 290.05 µs/muestra en el Escenario A (64 coeficientes, con señal de referencia) frente a 48 µs/muestra en los Escenarios B y C (8 coeficientes). Todas las implementaciones permanecieron holgadamente dentro de los 96 KB de SRAM del Arduino Due, con consumos de memoria de estado que oscilaron entre 36 y 72 bytes (IIR), 5 y 9.6 KB (FIR) y en torno a 12.4 KB (wavelet). BioFilterLib optimiza la gestión de memoria mediante instancias de CMSIS-DSP que preasignan los buffers de estado según las especificaciones de cada función, minimizando la fragmentación y facilitando el análisis de requisitos de recursos, característica valiosa en contextos educativos y experimentales.

Considerando que a 960 Hz el período de muestreo disponible es de aproximadamente 1042 µs, los tiempos observados confirman la viabilidad técnica de BioFilterLib para procesamiento en tiempo real ya que los filtros IIR consumen menos del 2 % del período, dejando amplios márgenes para procesamiento concurrente o cadenas de filtrado multicanal, mientras que incluso el FIR más costoso (pasa-altos, aprox. 40 % de CPU) permite el procesamiento monocanal en tiempo real con margen operativo suficiente.

5.2 Comparación con librerías existentes para el ecosistema Arduino

La evaluación comparativa entre BioFilterLib y otras opciones disponibles evidencia diferencias y similitudes cruciales para tomar decisiones de diseño durante la implementación de algoritmos de procesamiento digital de señales. La comparación con librerías de terceros se concentró en el Escenario A (filtros FIR e IIR notch), por ser el de mayor interés comparativo.

Para los filtros FIR, las implementaciones de Moarbue/FIR-Filter (con velocidad de procesamiento de 317.01 µs/muestra), Nilsson (351.74 µs/muestra), ttapa/Arduino-Filters (306.37 µs/muestra) y LeemanGeophysicalLLC/FIR_Filter_Arduino_Library (360.38 µs/muestra), evaluadas muestra por muestra, resultaron más lentas que BioFilterLib y CMSIS-DSP procesando por bloques (294.73 y 322.42 µs/muestra, respectivamente), como se observa en la Tabla 5. Este resultado se debe a la ventaja que presenta el procesamiento por bloques, ya que minimiza el costo computacional de gestión de buffers a lo largo de múltiples muestras.

Un resultado notable es la equivalencia en las métricas de calidad de filtrado entre las implementaciones evaluadas: varias librerías de terceros (Moarbue, Nilsson y ttapa) alcanzaron incluso una mejora de SNR ligeramente superior (26.71 dB) a la de BioFilterLib y CMSIS-DSP (23.29 dB). Esta diferencia se le puede atribuir al diseño del filtro (elección de coeficientes y de la ventana) y no a la estrategia de implementación. Esta observación respalda el principio de que la calidad de filtrado está determinada principalmente por el diseño del filtro, mientras que la implementación de software influye principalmente en la eficiencia computacional sin comprometer la precisión del procesamiento. Para los filtros IIR notch, BioFilterLib (8.40 µs/muestra, mejora de SNR de 10.87 dB) resultó más rápido que ttapa/Arduino-Filters (10.90 µs/muestra, 13.88 dB), con calidad de filtrado del mismo orden (Tabla 5).

El aporte más relevante de incorporar CMSIS-DSP como referencia directa fue cuantificar el sobrecosto de la capa de abstracción de BioFilterLib. Los resultados muestran que dicho sobrecosto es despreciable ya que en todos los escenarios, BioFilterLib y CMSIS-DSP produjeron métricas de calidad idénticas (misma mejora de SNR, MSE y correlación) y tiempos de procesamiento que difieren solo dentro del margen de variabilidad de la medición, en los filtros IIR de orden 4 el sobrecosto del wrapper fue de apenas 1.5 µs/muestra aproximadamente (18.27 frente a 16.79 µs/muestra en el Escenario B). Esto confirma que la capa de abstracción de BioFilterLib reproduce fielmente el rendimiento de CMSIS-DSP sin penalización apreciable, al tiempo que ofrece una interfaz orientada al contexto educativo. A diferencia de las librerías de terceros, centradas en filtros FIR o IIR, BioFilterLib integra además filtrado adaptativo NLMS y wavelet dentro de una misma interfaz documentada, lo que amplía su valor pedagógico.

5.3 Calidad de filtrado y preservación morfológica de bioseñales

Los indicadores de calidad de filtrado obtenidos mediante las métricas de SNR, MSE y correlación con la señal de referencia proporcionan la evidencia cuantitativa acerca de la efectividad de los algoritmos desarrollados y su capacidad para preservar las características morfológicas clave de las bioseñales. El análisis reveló que dicha efectividad depende en gran medida de la naturaleza del ruido y de su relación espectral de frecuencia con la señal útil.

En el escenario A, en el que la interferencia ocupa una banda estrecha en torno a los 60 Hz que se puede distinguir claramente de la señal, todos los métodos lograron mejoras de SNR importantes, entre 10.87 dB (IIR notch) y 26.71 dB (FIR notch de terceros), con correlaciones superiores a 0.98 (Tabla 5). El análisis en el dominio frecuencial confirmó visualmente la atenuación enfocada en la frecuencia central de 60 Hz, región del espectro de frecuencia que es de gran importancia en la electrocardiografía para preservar el complejo QRS y la onda T. El filtro wavelet Db4, a través de la anulación de los coeficientes de detalle que corresponden a la banda de la interferencia, ofreció el mejor equilibrio entre calidad y costo computacional (17.20 dB de mejora con solo 78.08 µs/muestra).

En contraste, en los Escenarios B y C el ruido (rosa y blanco respectivamente) se superpone su contenido espectral con el de la señal útil, lo que dificultó el filtrado lineal. El filtrado pasa-altos sobre el EEG logró apenas 2.12-2.56 dB de mejora de SNR, y el pasa-banda sobre el EMG prácticamente no tuvo ningún efecto (0.07 - 0.15 dB). En estos escenarios, el filtro adaptativo NLMS fue el único método que mejoró apreciablemente la señal (8.18 dB en EEG y 5.22 dB en EMG, con correlaciones de 0.9755 y 0.9561) gracias a la fuerte correlación que había entre la señal de referencia y el ruido contenido en la observación. Esta diferencia de comportamiento ilustra de manera directa una limitación clásica del filtrado lineal invariante en el tiempo frente a perturbaciones de banda ancha solapadas con la señal.

Por otro lado, los filtros wavelet con umbralización VisuShrink resultaron contraproducentes ante ruido de banda ancha: degradaron la señal en lugar de mejorarla, con mejoras de SNR negativas (-1.33 dB en EEG y -4.73 dB en EMG) y caídas bruscas en el valor de la correlación (hasta 0.2229 en EMG). La comparación entre familias en el Escenario C mostró además que la wavelet Db8 (16 coeficientes) prácticamente duplica el costo computacional de la Db4 (149.75 frente a 88.82 µs/muestra) sin aportar mejora de calidad alguna en este caso, por lo que el mayor orden de la wavelet no es justificable bajo este esquema de umbralización. Los valores de RMS de las señales filtradas fueron consistentes con estas observaciones, las marcadas disminuciones de RMS en los casos wavelet del Escenario C (0.0863 y 0.0825) confirman una sobre-atenuación de la energía de la señal, mientras que en los filtros lineales el RMS se mantuvo acorde con su respuesta en frecuencia.

No obstante, se debe reconocer que la evaluación de calidad se llevó a cabo con señales sintéticas y un modelo de ruido controlado a un SNR fijo de 5 dB, lo que no refleja plenamente patrones de los artefactos presentes en adquisiciones en el mundo real. Asimismo, los parámetros de los algoritmos no lineales (paso de adaptación y longitud del NLMS, umbral y número de niveles del wavelet) se fijaron en valores de referencia y no se optimizaron por cada escenario, de modo que los resultados de estos métodos deben interpretarse como una caracterización inicial y no como su rendimiento óptimo alcanzable.

5.4 Rangos de operación y frecuencias máximas

5.4.1 Frecuencia máxima de señal.

Con fs = 960 Hz, la máxima frecuencia representable es la de Nyquist, 480 Hz. Las bandas de paso de los filtros implementados se encuentran dentro de este límite: el filtro pasa-banda para EMG (20-450 Hz) aprovecha casi todo el ancho disponible, mientras que el filtro notch (60 Hz) y el pasa-altos (corte en 4 Hz) operan en el rango típico clínicamente relevante.

5.4.2 Frecuencia de muestreo máxima sostenible

Basado en el tiempo de procesamiento por muestra medido en el Arduino Due (84 MHz, sin FPU), la Tabla 8 muestra la frecuencia de muestreo máxima a la que cada filtro puede funcionar en tiempo real de forma monocanal (equivalente al throughput observado) y el uso de CPU correspondiente para la frecuencia objetivo de 960 Hz. Los valores corresponden a la implementación de BioFilterLib; el uso directo de CMSIS-DSP arroja cifras equivalentes.

Tabla 8. Tiempo de procesamiento, frecuencia máxima en tiempo real y carga de CPU por tipo de filtro (fs = 960 Hz)

Filtro	t/muestra (µs)	fs máx. tiempo real	CPU @ 960 Hz
FIR notch (151 taps)	294.7	≈ 3.4 kHz	28.3 %
IIR notch (1 biquad)	8.40	≈ 119 kHz	0.81 %
FIR pasa-altos (201 taps)	415.8	≈ 2.4 kHz	39.9 %
IIR pasa-altos (orden 4)	18.27	≈ 55 kHz	1.75 %
FIR pasa-banda (101 taps)	209.2	≈ 4.8 kHz	20.1 %
IIR pasa-banda (orden 2)	18.19	≈ 55 kHz	1.75 %
NLMS (8 taps)	48.1	≈ 21 kHz	4.6 %
NLMS (64 taps)	290.0	≈ 3.4 kHz	27.8 %
Wavelet Db4 (4 niveles)	78.1	≈ 12.8 kHz	7.5 %
Wavelet Db8 (4 niveles)	149.8	≈ 6.7 kHz	14.4 %

Estas cifras suponen un único canal y el filtro como única carga de cómputo; el margen frente a los 960 Hz objetivo es amplio en todos los casos (uso de CPU < 40 %). Los filtros IIR ofrecen el mayor margen (frecuencias máximas de decenas a más de cien kHz) gracias a su bajo número de operaciones por muestra, mientras que los FIR de muchos taps y el filtro adaptativo de orden alto son los más exigentes.

5.5 Validación de la hipótesis y cumplimiento de objetivos

La hipótesis general planteó que "la librería de filtros digitales para Arduino 'BioFilterLib' presenta un desempeño superior o equivalente a otras librerías disponibles en términos de velocidad de procesamiento, consumo de recursos y calidad de filtrado para los estudiantes e investigadores que trabajan con señales biomédicas."

Los resultados proporcionan evidencia favorable a esta hipótesis. En términos de velocidad de procesamiento, BioFilterLib demostró un desempeño competitivo o superior: sus filtros FIR e IIR procesados por bloques resultaron más rápidos que las implementaciones de terceros evaluadas en el Escenario A (por ejemplo, 8.40 µs/muestra en IIR notch frente a 10.90 µs/muestra de ttapa/Arduino-Filters). Respecto a la calidad de filtrado, BioFilterLib reprodujo exactamente las métricas de calidad de CMSIS-DSP y alcanzó valores equivalentes a los de las librerías de referencia, lo que confirma que el uso de optimizaciones de rendimiento no compromete la fidelidad del procesamiento. El consumo de recursos se mantuvo dentro de unos límites aceptables (un uso de CPU inferior al 2 % en los filtros IIR y memoria de estado muy por debajo de la SRAM disponible), lo que permite el uso de arquitecturas multicanal.

El objetivo general de "desarrollar y evaluar una librería de filtros digitales para Arduino BioFilterLib para un contexto educativo" se alcanzó con éxito mediante la implementación de una librería funcional, optimizada y documentada, cuya evaluación comparativa proporcionó evidencia cuantitativa de su desempeño. El primer objetivo específico, "evaluar la factibilidad técnica de BioFilterLib en la tarea de procesamiento de bioseñales", se llevó a cabo mediante pruebas repetibles sobre tres señales biomédicas (ECG, EEG y EMG), tres tipos de artefactos y cuatro arquitecturas de filtrado (FIR, IIR, NLMS y wavelet), donde se recopilaron métricas de calidad de filtrado, velocidad de procesamiento y consumo de recursos. El segundo objetivo específico, "comparar el consumo de recursos de hardware en términos de velocidad de procesamiento, calidad de

filtrado y consumo de recursos respecto a otras librerías disponibles", se cumplió mediante pruebas comparativas frente a CMSIS-DSP y a cuatro librerías de terceros. Una observación adicional de la evaluación multiescenario fue evidenciar que la elección del algoritmo de filtrado debe ajustarse a las características del ruido en cada escenario específico.

5.6 Implicancias para la enseñanza de DSP

El desarrollo y evaluación de BioFilterLib responden a una necesidad reflejada en la literatura sobre enseñanza en ingeniería: la disponibilidad de herramientas que equilibren un costo asequible, facilidad de uso y capacidades técnicas suficientes para un aprendizaje significativo de DSP. La factibilidad técnica demostrada al implementar filtros digitales de adecuadamente en una plataforma de bajo costo como Arduino Due (con un costo de 30-50 USD, frente a 150-600 USD de plataformas especializadas de DSP) respalda su uso como recurso educativo, reduciendo las barreras económicas para instituciones con presupuestos limitados, un aspecto que resulta particularmente relevante en países en desarrollo.

La evaluación sobre tres escenarios con distintos tipos de ruido refuerza el valor pedagógico de la librería, al permitir que el estudiante verifique empíricamente principios fundamentales del DSP como el hecho de que un filtro de rechazo de banda elimina eficazmente una interferencia de banda estrecha como la de 60 Hz; que el filtrado lineal resulta insuficiente cuando el ruido se superpone espectralmente con la señal útil, en cuyo caso el filtrado adaptativo NLMS demuestra ser una mejor opción; y que técnicas como la umbralización wavelet pueden degradar la señal si se aplican fuera de su dominio adecuado. La disponibilidad de múltiples arquitecturas (FIR, IIR, NLMS y wavelet) dentro de una misma librería facilita esta comparación experimental directa, así como la verificación de la relación entre orden de filtro, complejidad computacional y calidad de filtrado.

Adicionalmente, la página de documentación de BioFilterLib, con ejemplos contextualizados para bioseñales específicas y tutoriales paso a paso, aborda una limitación de las librerías existentes sin documentación actualizada u orientadas a usuarios avanzados. Su arquitectura, que encapsula CMSIS-DSP mediante una capa de

abstracción fácil de usar, familiariza gradualmente a los estudiantes a las consideraciones de optimización en sistemas embebidos.

La programación en entornos embebidos suele introducir complejidades adicionales (gestión de memoria, temporización, depuración limitada) que pueden distraer del aprendizaje de conceptos fundamentales en etapas iniciales, por lo que BioFilterLib se presenta como complemento de herramientas de simulación como MATLAB/Python, sin embargo, la evaluación de su eficacia pedagógica en contextos educativos reales sigue siendo una tarea para futuros estudios.

5.7 Limitaciones del estudio

Para la interpretación de los resultados es importante tener en cuenta las limitaciones metodológicas y de alcance del estudio que pueden restringir la generalización de las conclusiones obtenidas.

Limitaciones de la plataforma de evaluación: la validación se realizó de forma exclusiva en Arduino Due (Cortex-M3, 84 MHz), por lo que las conclusiones relativas al rendimiento absoluto no pueden extrapolarse directamente a otras arquitecturas. Plataformas con FPU mejorada (Cortex-M4/M7), doble núcleo (ESP32) o DSP dedicado (STM32) podrían llegar a exhibir características de desempeño considerablemente diferentes y estudios futuros deberían evaluar la portabilidad y el rendimiento en múltiples plataformas.

Limitaciones de las señales de prueba: aunque la evaluación se realizó con tres bioseñales (ECG, EEG y EMG), estas fueron señales sintéticas, generadas computacionalmente usando modelos matemáticos y modelos de ruido controlado a un SNR fijo de 5 dB. Este enfoque no captura completamente la complejidad de los artefactos en el mundo real (interferencias electromagnéticas no gaussianas, artefactos de movimiento, variabilidad morfológica en diferentes individuos, ruido de cuantización del ADC). La ausencia de evaluación con bases de datos biomédicas estandarizadas (por ejemplo, MIT-BIH Arrhythmia Database (76) o PTB Diagnostic ECG Database (77)) limita la validación de la preservación morfológica en contextos clínicos reales.

Limitaciones metodológicas de la evaluación: la comparación empleó configuraciones específicas de filtro (número de coeficientes, órdenes, frecuencias de corte y factor de calidad Q) que no representan el todo el rango completo de aplicaciones posibles. En

particular, los parámetros de los métodos no lineales (paso de adaptación y longitud del NLMS; umbral, familia y número de niveles del wavelet) se fijaron en valores de referencia y no se optimizaron por escenario, lo que no refleja el rendimiento que se puede llegar a tener con parámetros óptimos. Además, la comparación con librerías de terceros se limitó al Escenario A, y la evaluación no incluyó mediciones de consumo energético, métrica relevante en aplicaciones portátiles alimentadas por batería.

Limitaciones de validación educativa: Aunque BioFilterLib se diseñó pensando en un contexto educativo, la evaluación no incluyó estudios de usabilidad con estudiantes ni mediciones de la eficacia pedagógica. Por este motivo, su impacto educativo sigue siendo una hipótesis sujeta a validación empírica.

Limitaciones de alcance funcional: La implementación actual se enfoca en filtrado digital básico (FIR, IIR, NLMS y wavelet), sin incluir técnicas más avanzadas como el filtrado de Kalman o algoritmos de detección automática de características de interés biomédico (detección de complejos QRS, clasificación de arritmias). La extensión hacia aplicaciones más especializadas requeriría un desarrollo y validación adicionales.

Limitaciones de alcance en uso de hardware: Aunque BioFilterLib se diseñó con el objetivo de la experimentación práctica con adquisición de señales, en este trabajo el enfoque fue en la evaluación comparativa y de factibilidad técnica. No se realizaron pruebas de uso en adquisición de señales externas, lo cual supondría el uso del ADC del Arduino Due.

VI. Conclusiones

En este estudio se diseñó, implementó y evaluó BioFilterLib, una librería de filtrado digital para Arduino Due basada en optimizaciones de la librería CMSIS-DSP, con el objetivo de facilitar el procesamiento de bioseñales en contextos educativos y de investigación. La evaluación comparativa se llevó a cabo a través de pruebas experimentales donde se recopilaron tiempo de procesamiento, uso de CPU, consumo de memoria RAM y métricas de calidad de filtrado (mejora de SNR, error cuadrático medio, correlación y valor eficaz) sobre tres escenarios biomédicos sintéticos con SNR de entrada fijo de 5 dB, señal de ECG con interferencia de 60 Hz, señal de EEG con ruido rosa y señal de EMG con ruido blanco. Las arquitecturas evaluadas comprendieron filtros FIR, filtros IIR, filtros adaptativos NLMS y filtros wavelet basados en la transformada de Daubechies.

El análisis mostró que los filtros IIR alcanzaron una eficiencia computacional superior a la de los FIR (velocidad de procesamiento de 8.40 a 18.27 µs/muestra frente a 209.16-415.84 µs/muestra) con un uso de CPU inferior al 2 %, mientras que el costo de los filtros FIR escaló conforme se incrementó el número de coeficientes. La incorporación de CMSIS-DSP como referencia en el benchmark demostró que el sobrecosto de la capa de abstracción de BioFilterLib es despreciable. La librería reprodujo exactamente las métricas de calidad de CMSIS-DSP y mostró tiempos de procesamiento equivalentes dentro del margen de medición. Frente a las librerías de terceros del ecosistema Arduino, BioFilterLib presentó una calidad de filtrado equivalente y velocidades de procesamiento equivalentes o superiores.

Un resultado derivado de la evaluación multiescenario es que la eficacia del filtrado encontró diferencias en función de la naturaleza del ruido a eliminar y de su relación espectral con la señal. La interferencia de banda estrecha de 60 Hz se atenuó eficazmente mediante filtros notch y wavelet (mejoras de SNR de hasta 26.71 dB y correlaciones superiores a 0.98), pero el ruido de banda ancha y coloreado, superpuesto con la banda útil de la señal, limitó severamente al filtrado lineal (mejoras de apenas 0.07-2.56 dB en EEG y EMG). En estos casos, únicamente el filtro adaptativo NLMS logró mejoras apreciables (8.18 dB en EEG y 5.22 dB en EMG), en tanto que la umbralización wavelet VisuShrink resultó contraproducente. Este resultado subraya que la selección y

optimización de parámetros del algoritmo de filtrado debe ajustarse a las características del ruido.

Los resultados obtenidos validan la hipótesis general del estudio, al demostrar que BioFilterLib ofrece un rendimiento igual o superior al de las alternativas evaluadas, sin comprometer la calidad del procesamiento. No obstante, la validación se limitó a una única plataforma de hardware y a señales sintéticas con un modelo de ruido controlado. Además los parámetros de los métodos no lineales no se optimizaron por escenario y la confirmación de la preservación morfológica en aplicaciones clínicas reales requiere estudios complementarios con bases de datos biomédicas estandarizadas y evaluación por especialistas.

Finalmente, este estudio refuerza la importancia de desarrollar herramientas de software específicamente diseñadas para contextos educativos, especialmente en instituciones con recursos limitados, y de explorar estrategias que concilien un adecuado rendimiento computacional con facilidad de uso. BioFilterLib representa una contribución en esta dirección, ya que proporciona una herramienta documentada y optimizada que, además de igualar el rendimiento de CMSIS-DSP, integra múltiples arquitecturas de filtrado y permite contrastar experimentalmente su comportamiento ante distintos tipos de bioseñales y de ruido.

Bibliografía

1. Npatchett. English: Derivation of the limb leads [Internet]. 2015 [citado 6 de diciembre de 2025]. Disponible en: https://commons.wikimedia.org/wiki/File:Limb_leads_of_EKG.png

2. Sepúlveda R, Montiel Ross O, Díaz G, Guetierrez D, Castillo O. Classification of Encephalographic Signals using Artificial Neural Networks. Comput Syst. 1 de marzo de 2015;19:69-88.

3. Mashwama NX, Madubela B. Innovations in pedagogy and technology for engineering education: A systematic review. Interdiscip J Educ Res. 17 de abril de 2025;7(s1):a10-a10.

4. Kalyani R, Nirmala S, Gireesha H, Shet R, Iyer N, Chikkamath S, et al. Problem Based Learning–An Effective way of teaching DSP course. J Eng Educ Transform. 2024;76-81.

5. Hochgraf C. Using Arduino To Teach Digital Signal Processing. En 2013 [citado 2 de mayo de 2024]. Disponible en: https://www.semanticscholar.org/paper/Using-Arduino-To-Teach-Digital-Signal-Processing-Hochgraf/290a8c9aab485d3b8f4ebe8e08584bd300d7666c

6. Tupac-Yupanqui M, Vidal-Silva C, Pavesi-Farriol L, Sánchez Ortiz A, Cardenas-Cobo J, Pereira F. Exploiting Arduino Features to Develop Programming Competencies. IEEE Access. 2022;10:20602-15.

7. Fernández J, Gemin W, Rivera R, Revuelta M, Kuzman M, Hidalgo R. Digital filter design with Arduino DUE and Matlab. En: 2015 XVI Workshop on Information Processing and Control (RPIC) [Internet]. 2015 [citado 2 de mayo de 2024]. p. 1-6. Disponible en: https://ieeexplore.ieee.org/document/7497060

8. Carlos AC, Arturo ZL, Cesar BA, Andrés FR, Miriam AS. Improving the teaching of the digital signal processing course for undergraduate students in electrical engineering. Discov Educ. 4 de marzo de 2025;4(1):49.

9. Bolanakis DE, Evangelakis GA, Glavas E, Kotsis KT. A teaching approach for bridging the gap between low-level and high-level programming using assembly language learning for small microcontrollers. Comput Appl Eng Educ. 2011;19(3):525-37.

10. Arduino - Home [Internet]. [citado 20 de junio de 2024]. Disponible en: https://www.arduino.cc/

11. Kadar R, Wahab NA, Othman J, Shamsuddin M, Mahlan SB. A study of difficulties in teaching and learning programming: a systematic literature review. Int J Acad Res Progress Educ Dev. 2021;10(3):591-605.

12. Nurhayati N, Husain S, Muhammad Y. Student Difficulties and Alternative Solutions in the Embedded System (ES) Subject. Asian J Appl Sci. 2022;10(4).

13. Hall TS, Anderson DV. A framework for teaching real-time digital signal processing with field-programmable gate arrays. IEEE Trans Educ. agosto de 2005;48(3):551-8.

14. Barquero-Pérez Ó, Cámara-Vázquez MÁ, Goya-Esteban R. Hands-On Mastery in Physiological Signal Processing: A Challenge-Based Learning Approach. Biomed Eng Educ. 1 de julio de 2025;5(2):311-28.

15. CMSIS: Introduction [Internet]. [citado 20 de junio de 2024]. Disponible en: https://arm-software.github.io/CMSIS_6/latest/General/index.html

16. Nilsson S. GitHub - sebnil/FIR-filter-Arduino-Library: FIR filter Arduino Library [Internet]. [citado 3 de junio de 2025]. Disponible en: https://github.com/sebnil/FIR-filter-Arduino-Library

17. Zhou J, Ross KA. Implementing database operations using SIMD instructions. En: Proceedings of the 2002 ACM SIGMOD international conference on Management of data [Internet]. New York, NY, USA: Association for Computing Machinery; 2002 [citado 17 de diciembre de 2025]. p. 145-56. (SIGMOD '02). Disponible en: https://doi.org/10.1145/564691.564709

18. Alan V. Oppenheim, Alan S. Willsky, S. Hamid Nawab. Signals and Systems. 2nd ed. Upper Saddle River, NJ: Prentice Hall; 1997. (Prentice Hall Signal Processing Series).

19. Nyquist H. Certain Topics in Telegraph Transmission Theory. Trans Am Inst Electr Eng. abril de 1928;47(2):617-44.

20. Widrow B. Quantization noise: roundoff error in digital computation, signal processing, control, and communications. Cambridge: Cambridge University Press; 2008. 751 p.

21. Antoniou A. Digital signal processing: signals, systems and filters. New York, NY: McGraw-Hill; 2006. 965 p.

22. Orfanidis SJ. Introduction to signal processing. Englewood Cliffs, N.J: Prentice Hall; 1996. 798 p. (Prentice Hall signal processing series).

23. Rangayyan RM. Biomedical signal analysis: a case-study approach. [Piscataway, NJ] : New York, N.Y: IEEE Press ; Wiley-Interscience; 2002. 516 p. (IEEE Press series in biomedical engineering).

24. Sörnmo L, Laguna P. Bioelectrical signal processing in cardiac and neurological applications. Nachdr. Amsterdam Heidelberg: Elsevier Academic Press; 2006. 668 p. (Academic Press series in biomedical engineering).

25. Sanei S, Chambers JA. EEG signal processing. Chichester, England Hoboken, NJ: John Wiley & Sons; 2007..

26. Parker PA, Merletti R, editores. Electromyography: physiology, engineering, and noninvasive applications. Hoboken, N.J: Wiley-Interscience; 2004. 1 p. (IEEE

Press series in biomedical engineering).

27. Prince JL, Links JM. Medical imaging signals and systems. 2.ª ed. Boston: Pearson; 2015. 519 p.

28. Theraja BL. Fundamentals of Electrical Engineering and Electronics. S. Chand Publishing; 2006. 859 p.

29. Semmlow JL. Biosignal and Medical Image Processing. 3rd ed. Milton: Taylor & Francis Group; 2014. 1 p.

30. Lyons RG. Understanding digital signal processing. 3rd ed. Upper Saddle River, NJ: Prentice Hall; 2011. 954 p.

31. Paarmann, Larry D. Design and Analysis of Analog Filters: A Signal Processing Perspective. Boston; New York; Dordrecht; London; Moscow: Springer; 2013. 440 p. (The Springer International Series in Engineering and Computer Science).

32. Madisetti VK, editor. Wireless, networking, radar, sensor array processing, and nonlinear signal processing. Boca Raton, Fl.: CRC Press; 2010. (The digital signal processing handbook).

33. Rabiner LR, Gold B. Theory and application of digital signal processing. 18. [print.]. Englewood Cliffs, N.J: Prentice-Hall; 1975. 762 p.

34. Bianchi G, Sorrentino R. Electronic filter: simulation & design. New York, NY: McGraw-Hill; 2007. 606 p.

35. Mallat S. A wavelet tour of signal processing: the sparse way. 3rd ed. Burlington, MA: Academic Press/Elsevier; 2009. 1 p.

36. Addison PS. The illustrated wavelet transform handbook: introductory theory and applications in science, engineering, medicine and finance. Bristol Philadelphia: Institute of physics publ; 2002.

37. Thakor NV, Zhu YS. Applications of adaptive filtering to ECG analysis: noise cancellation and arrhythmia detection. IEEE Trans Biomed Eng. agosto de 1991;38(8):785-94.

38. Nait-Ali A, Cavaro-Menard C. Compression of Biomedical Images and Signals. Hoboken: John Wiley & Sons, Inc; 2010. 330 p. (ISTE).

39. Unser M, Aldroubi A. A review of wavelets in biomedical applications. Proc IEEE. abril de 1996;84(4):626-38.

40. Haykin SS. Adaptive filter theory. 5. ed. Upper Saddle River, NJ: Pearson; 2014. 889 p. (Always learning).

41. He P, Wilson G, Russell C. Removal of ocular artifacts from electro-encephalogram by adaptive filtering. Med Biol Eng Comput. mayo de 2004;42(3):407-12.

42. Gutierrez-Rivas R, Garcia JJ, Marnane WP, Hernandez A. Novel Real-Time Low-Complexity QRS Complex Detector Based on Adaptive Thresholding. IEEE Sens J. octubre de 2015;15(10):6036-43.

43. Ganz L, Link M. Electrocardiography. En: Goldman L, Cooney K, editores. Goldman-Cecil Medicine. 27th ed. Philadelphia, PA: Elsevier; 2024. p. chap 42.

44. Einthoven W. The different forms of the human electrocardiogram and their signification. Lancet. 1912;179(4622):853-61.

45. Merletti R, Parker P. Electromyography: Physiology, Engineering, and Noninvasive Applications. Hoboken, NJ: John Wiley & Sons; 2004.

46. Rangayyan R. Biomedical Signal Analysis: A Case-Study Approach. 2nd ed. Hoboken, NJ: John Wiley & Sons; 2015.

47. Niedermeyer E, Lopes da Silva F. Electroencephalography: Basic Principles, Clinical Applications, and Related Fields. 5th ed. Philadelphia, PA: Lippincott Williams & Wilkins; 2005.

48. Panneerselvam P. Application of embedded system for a genetic disease, sickle cell anemia. En: 2014 International Conference on Advances in Electrical Engineering (ICAEE) [Internet]. Vellore, India: IEEE; 2014 [citado 30 de junio de 2025]. p. 1-4. Disponible en: http://ieeexplore.ieee.org/document/6838446/

49. Lin R, Brown F, James S, Jones J, Ekinci E. Continuous glucose monitoring: A review of the evidence in type 1 and 2 diabetes mellitus. Diabet Med. mayo de 2021;38(5):e14528.

50. Ünsalan C, Yücel ME, Gürhan HD. Digital signal processing using Arm Cortex-M based microcontrollers: theory and practice. Cambridge: arm Education Media; 2018. 334 p.

51. Toulson R. Fast and effective embedded systems design: applying the ARM mbed. Oxford, U.K: Newnes; 2012. 1 p.

52. Due | Arduino Documentation [Internet]. [citado 4 de julio de 2025]. Disponible en: https://docs.arduino.cc/hardware/due/#features

53. CMSIS DSP Software Library [Internet]. [citado 4 de julio de 2025]. Disponible en: https://arm-software.github.io/CMSIS_5/DSP/html/index.html

54. Grubb S. ARM DSP is the way to go – An intern's perspective - Architectures and Processors blog - Arm Community blogs - Arm Community [Internet]. 2013 [citado 15 de agosto de 2025]. Disponible en: https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/arm-dsp-is-the-way-to-go-an-intern-s-perspective

55. Tran K, Bacher J, Shi Y, Skripchuk J, Price T. Overcoming Barriers in Scaling Computing Education Research Programming Tools: A Developer's Perspective. En: Proceedings of the 2024 ACM Conference on International Computing Education

Research - Volume 1 [Internet]. New York, NY, USA: Association for Computing Machinery; 2024 [citado 15 de agosto de 2025]. p. 312-25. (ICER '24; vol. 1). Disponible en: https://doi.org/10.1145/3632620.3671113

56. LeemanGeophysicalLLC/FIR_Filter_Arduino_Library [Internet]. Leeman Geophysical LLC; 2025 [citado 24 de agosto de 2025]. Disponible en: https://github.com/LeemanGeophysicalLLC/FIR_Filter_Arduino_Library

57. P P. tttapa/Arduino-Filters [Internet]. 2025 [citado 24 de agosto de 2025]. Disponible en: https://github.com/tttapa/Arduino-Filters

58. espressif/esp-dsp [Internet]. Espressif Systems; 2025 [citado 24 de agosto de 2025]. Disponible en: https://github.com/espressif/esp-dsp

59. Condes Brena E. ArduinoFFT [Internet]. 2024 [citado 24 de agosto de 2025]. Disponible en: https://github.com/kosme/arduinoFFT

60. Schatzmann P. pschatzmann/arduino-audio-tools [Internet]. 2025 [citado 24 de agosto de 2025]. Disponible en: https://github.com/pschatzmann/arduino-audio-tools

61. mozanunal. mozanunal/SimpleDSP [Internet]. 2025 [citado 24 de agosto de 2025]. Disponible en: https://github.com/mozanunal/SimpleDSP

62. Schmidt P. daPhoosa/FirFilter [Internet]. 2024 [citado 25 de agosto de 2025]. Disponible en: https://github.com/daPhoosa/FirFilter

63. Moarbue. Moarbue/FIR-Filter [Internet]. 2025 [citado 25 de agosto de 2025]. Disponible en: https://github.com/Moarbue/FIR-Filter

64. ikostoski/esp32-i2s-slm: Sound Level Meter with ESP32 and I2S MEMS microphone [Internet]. [citado 25 de agosto de 2025]. Disponible en: https://github.com/ikostoski/esp32-i2s-slm/tree/master

65. Jaldén J, Moreno XC, Skog I. Using the Arduino Due for Teaching Digital Signal Processing. En: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) [Internet]. 2018 [citado 2 de mayo de 2024]. p. 6468-72. Disponible en: https://ieeexplore.ieee.org/document/8461781

66. Esposito WJ, Mujica FA, Garcia DG, Kovacs GTA. The Lab-In-A-Box project: An Arduino compatible signals and electronics teaching system. En: 2015 IEEE Signal Processing and Signal Processing Education Workshop (SP/SPE) [Internet]. 2015 [citado 20 de agosto de 2025]. p. 301-6. Disponible en: https://ieeexplore.ieee.org/document/7369570

67. Wickert MA. Real-time DSP basics using Arduino and the Analog Shield SAR codec board. En: 2015 IEEE Signal Processing and Signal Processing Education Workshop (SP/SPE) [Internet]. Salt Lake City, UT, USA: IEEE; 2015 [citado 20 de agosto de 2025]. p. 59-64. Disponible en: http://ieeexplore.ieee.org/document/7369528/

68. Arslan K, Tanel Z. Analyzing the effects of Arduino applications on students'

opinions, attitude and self-efficacy in programming class. Educ Inf Technol. 1 de enero de 2021;26(1):1143-63.

69. Ali BSM, Farej ZK, Ibrahim AM. Implementing FIR Filters using Arduino Due Platform for Educational Purposes. En: 2019 2nd International Conference on Engineering Technology and its Applications (IICETA) [Internet]. Al-Najef, Iraq: IEEE; 2019 [citado 14 de febrero de 2024]. p. 49-54. Disponible en: https://ieeexplore.ieee.org/document/9012990/

70. Jamieson P, Herdtner J. More missing the Boat — Arduino, Raspberry Pi, and small prototyping boards and engineering education needs them. En: 2015 IEEE Frontiers in Education Conference (FIE) [Internet]. 2015 [citado 25 de agosto de 2025]. p. 1-6. Disponible en: https://ieeexplore.ieee.org/document/7344259

71. Diouri O, Gaga A, Ouanan H, Senhaji S, Faquir S, Jamil MO. Comparison study of hardware architectures performance between FPGA and DSP processors for implementing digital signal processing algorithms: Application of FIR digital filter. Results Eng. 1 de diciembre de 2022;16:100639.

72. TMDSEVM6678 Development kit | TI.com [Internet]. [citado 2 de mayo de 2024]. Disponible en: https://www.ti.com/tool/TMDSEVM6678

73. Cora Z7-07S: Single Core for ARM/FPGA SoC Development [Internet]. [citado 2 de mayo de 2024]. Disponible en: https://www.xilinx.com/products/boards-and-kits/1-1qlaz7n.html

74. Vostrukhin A, Vakhtina E. STUDYING DIGITAL SIGNAL PROCESSING ON ARDUINO BASED PLATFORM. Eng RURAL Dev.

75. Tan L, Jiang J. Teaching Digital Filter Implementations Using the 68HC12 Microcontroller. En: 2011 ASEE Annual Conference & Exposition Proceedings [Internet]. Vancouver, BC: ASEE Conferences; 2011 [citado 14 de febrero de 2024]. p. 22.1384.1-22.1384.19. Disponible en: http://peer.asee.org/18681

76. Moody GB, Mark RG. MIT-BIH Arrhythmia Database [Internet]. physionet.org; 1992 [citado 16 de enero de 2026]. Disponible en: https://physionet.org/content/mitdb/

77. Bousseljot RD, Kreiseler D, Schnabel A. The PTB Diagnostic ECG Database [Internet]. physionet.org; 2004 [citado 16 de enero de 2026]. Disponible en: https://physionet.org/content/ptbdb/