ESCUELA NAVAL DE CADETES
"ALMIRANTE PADILLA"
Honor y Tradición

OnBoard Knowledge Journal

Research article

Optimization of Algorithms on Microcontrollers for Efficient Signal Processing

Optimización de algoritmos en microcontroladores para un procesamiento eficiente de señales

Marcos Daza[1], Johan Mardini[1,†] and Yaribeth Rodriguez[1,*]

[1] Faculty of Engineering, Grupo de Investigación en Inteligencia Artificial y Robótica (SIRO), Universidad de la Costa, Barranquilla, Colombia; mdazab@cuc.edu.co; jmardini@cuc.edu.co; yrodrigu69@cuc.edu.co
[†] Deceased.
[*] Correspondence: yrodrigu69@cuc.edu.co

Citation: Daza, M.; Mardini, J.; Rodriguez, Y. Optimization of Algorithms on Microcontrollers for Efficient Signal Processing. *OnBoard Knowledge Journal* **2026**, *2*, 7. https://doi.org/10.70554/OBJK2026.v02n02.07

Received: 23/09/2026, Accepted: 27/10/2026, Published: 04/11/2026

DOI: https://doi.org/10.70554/OBJK2026.v02n02.07

Abstract: Algorithm optimization on microcontrollers is a crucial challenge in the field of digital signal processing applied to embedded systems, where memory, energy, and computing capacity are severely limited. This work reviews strategies and methodologies for implementing signal processing algorithms on low-power platforms, with the aim of maximizing computational and energy efficiency without compromising accuracy. Optimization techniques at both the software and hardware levels are described, including the selection of appropriate algorithms (FFT, DFT, digital filtering), the use of fixed-point arithmetic, task parallelization, and the adaptation of code to specific microcontroller architectures. The results reported in the literature show significant improvements in execution time and energy consumption in practical applications such as digital noise filtering, signal demodulation in communication systems, spectral analysis of electrical loads, and real-time pattern recognition. The discussion highlights the importance of balancing algorithmic complexity with the constraints of embedded systems, emphasizing that efficiency must be achieved without sacrificing signal processing accuracy. Finally, future trends point toward the use of low-power microcontrollers with integrated artificial intelligence capabilities, opening new possibilities for applications in IoT, energy monitoring, and intelligent control systems. This work provides a structured guide for researchers and developers interested in implementing efficient signal processing solutions in resource-constrained environments, based on experiences documented in the literature and real applications of algorithmic optimization on microcontrollers.

Keywords: Arduino; Efficiency; Algorithm; Digitization; Signal processing

Resumen: La optimización de algoritmos en microcontroladores representa un desafío crucial en el campo del procesamiento digital de señales aplicado a sistemas embebidos, donde la memoria, la energía y la capacidad de cómputo están altamente limitadas. Este trabajo revisa estrategias y metodologías para implementar algoritmos de procesamiento de señales en plataformas de bajo consumo, con el objetivo de maximizar la eficiencia computacional y energética sin comprometer la precisión. Se describen técnicas de optimización tanto a nivel de software como de hardware, incluyendo la selección de algoritmos apropiados (FFT, DFT, filtrado digital), el uso de aritmética en punto fijo, la paralelización

© 2026 by authors.
Licensed by Escuela Naval de Cadetes "Almirante Padilla", COL.
This article is freely accessible and distributed under the terms and conditions of Creative Commons Attribution (https://creativecommons.org/licenses/by/4.0/).

de tareas y la adaptación del código a arquitecturas específicas de microcontroladores. Los resultados reportados en la literatura demuestran mejoras significativas en el tiempo de ejecución y el consumo de energía en aplicaciones prácticas como el filtrado digital de ruido, la demodulación de señales en sistemas de comunicación, el análisis espectral de cargas eléctricas y el reconocimiento de patrones en tiempo real. La discusión resalta la importancia de equilibrar la complejidad algorítmica con las restricciones de los sistemas embebidos, enfatizando que la eficiencia debe lograrse sin sacrificar la precisión del procesamiento de señales. Finalmente, las tendencias futuras apuntan hacia el uso de microcontroladores de bajo consumo con capacidades de inteligencia artificial integrada, abriendo nuevas posibilidades para aplicaciones en IoT, monitoreo energético y sistemas de control inteligente. Este trabajo proporciona una guía estructurada para investigadores y desarrolladores interesados en implementar soluciones eficientes de procesamiento de señales en entornos con recursos limitados, basándose en experiencias documentadas en la literatura y aplicaciones reales de optimización algorítmica en microcontroladores.

Palabras clave: Arduino; Eficiencia; Algoritmo; Digitalización; procesamiento de señales

1. Introduction

Digital signal processing (DSP) in embedded systems has experienced significant growth in recent decades, driven by the need to implement efficient solutions in applications such as communications, industrial control, energy monitoring, and pattern recognition. However, running complex algorithms in microcontrollers remains a challenge due to inherent limitations of memory, computational capacity, and power consumption. Several studies have addressed the implementation of classical algorithms such as Fast Fourier Transform (FFT) and Discrete Fourier Transform (DFT) in embedded platforms [2;4;6;9], demonstrating that optimization is essential to ensure adequate performance in real time. Research by authors such as Ferreira [1] and Pereira [7] shows that careful selection of algorithms and their adaptation to specific architectures, such as ESP8266 microcontrollers [1] or PIC [12], can lead to substantial improvements in computational and energy efficiency. The literature also highlights the importance of techniques aimed at reducing algorithmic complexity, including the use of fixed-point arithmetic, the parallelization of tasks in architectures with DSP and FPGA capabilities [5;10], and the optimization of code adjusted to hardware constraints. These methodologies have been applied in various contexts, ranging from noise reduction in audio signals [3] and digital demodulation in communication systems [12] to the monitoring of electrical parameters in single-phase loads [5;8;11]. More recent research points towards the integration of artificial intelligence capabilities in low-power microcontrollers, which opens up new possibilities for applications in IoT, intelligent control systems, and advanced spectral analysis. This perspective suggests a scenario in which algorithmic optimization seeks not only efficiency, but also scalability and adaptability to dynamic environments.

In this context, the present work offers a structured review of algorithm optimization strategies in microcontrollers, with emphasis on practical applications of signal processing. The objective is to provide guidance to researchers and developers who face the challenge of balancing algorithmic accuracy with hardware constraints, thus contributing to the design of more efficient and sustainable embedded solutions.

The remainder of this paper is organized as follows. Section 2 summarizes the main contributions of this work. Section 3 describes the experimental setup, including the materials (Section 3.1), the apparatus and programming conditions (Section 3.2), the optimization techniques implemented (Section 3.3), and the experimental procedure (Section 3.4). Section 4 presents and discusses the results obtained with the basic and optimized implementations, and Section 4.1 describes the limitations of the implementation. Finally, Section 5 presents the conclusions and future lines of work.

2. Contributions

The main contributions of this work are the following:

I. An experimental comparison, on a resource-constrained 8-bit platform (Arduino Uno, ATmega328P at 16 MHz with 2 KB of SRAM), between a basic floating-point implementation and an optimized fixed-point implementation of the spectral energy calculation over a window of $N = 64$ samples.
II. The integration of complementary software optimization techniques (fixed-point arithmetic, precomputed lookup tables, bit-shift operations instead of divisions, and the FFT) showing a reduction in execution time of more than 40% (from more than 2500 μs to approximately 1500 μs).
III. A low-cost and reproducible experimental setup that combines execution-time measurement with the `micros()` function and wireless data transmission through an HC-05 Bluetooth module, with an observed average latency of 20–30 ms, suitable for remote monitoring and IoT applications.
IV. A structured review of the literature on FFT and DFT implementations in embedded platforms (ESP8266, PIC, DSP and FPGA architectures), linking optimization strategies with application domains such as noise reduction, digital demodulation, and electrical monitoring.
V. An explicit analysis of the limitations of the study (single platform and window length, simulated signal, and absence of quantitative accuracy and energy data), which defines concrete lines for future research on embedded signal processing.

3. Experimental

3.1. Materials

For the validation of the optimization strategies, an Arduino Uno (ATmega328P, 16 MHz, 2 KB SRAM, 32 KB Flash) was used as the test platform, along with an HC-05 Bluetooth module configured in slave mode to enable wireless data transmission to a personal computer. The programming environment used was the Arduino IDE, and data reception was done through the serial monitor of the PC.

This configuration was selected due to its similarity to the embedded systems described in the literature [1;4;7], where algorithmic optimization is usually evaluated on low-power platforms with limited resources.

3.2. Apparatus and programming conditions

The Arduino Uno was programmed to generate a simulated discrete signal (Figure 1) and apply two processing variants to it: a basic floating-point implementation and a fixed-point-optimized version employing precomputed *lookup tables*. Communication between the Arduino and the HC-05 module was established using the RX/TX pins, with a voltage divider to protect the module's input. In the code shown in Figure 2, this communication is implemented with the SoftwareSerial library, using pin 10 as reception (RX) and pin 11 as transmission (TX) of the Arduino, and the signal is stored in an array of $N = 64$ samples. The HC-05 was paired with the computer via Bluetooth, which created a virtual serial port that allowed the reception of the transmitted data. The execution time of each algorithm was measured with the `micros()` function, and the results were sent in text format for later analysis on the PC. Table 1 summarizes the experimental conditions reported in the manuscript.

Table 1. Configuration and experimental conditions of Arduino Uno measurements.

Parameter	Reported Value
Microcontroller	Arduino Uno (ATmega328P)
Clock Frequency	16 MHz
SRAM Memory	2 KB (2048 bytes)
Flash Memory	32 KB (32,256 bytes available to the program, according to the IDE report)
Bluetooth Module	HC-05, set to slave mode
Arduino–HC-05 connection	RX/TX pins with voltage divider; SoftwareSerial on pins 10 (RX) and 11 (TX)
Development Environment	Arduino IDE (version 2.3.7, visible in Figure 2)
Receiving Data	Personal computer, via Bluetooth virtual serial port and serial monitor
Test Signal	Simulated discrete signal (sinusoidal, Figure 1); window $N = 64$ samples
Variants evaluated	(1) Basic, floating-point; (2) Optimized, fixed-point with precomputed lookup tables
Runtime measurement	`micros()` function; results in μs
Recorded parameters	Processing time, memory usage, and spectral accuracy

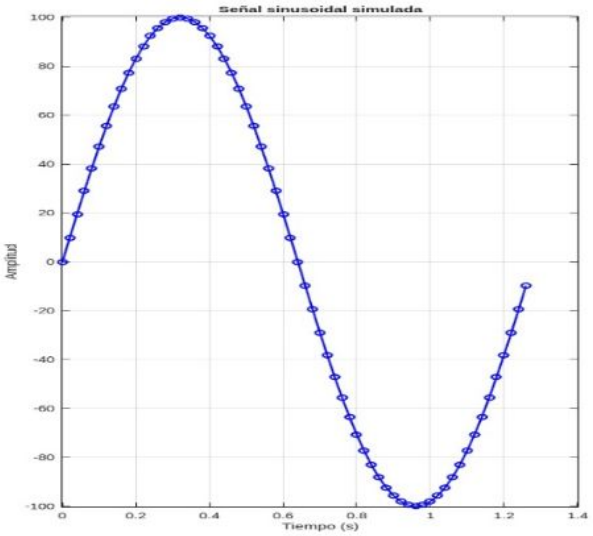


Figure 1. Simulated sinusoidal signal. Horizontal axis: time (s); vertical axis: amplitude.

3.3. Optimization techniques implemented

The two variants evaluated calculate spectral energy over a window of 64 samples. The basic variant uses *floating-point arithmetic* and is taken as a reference for comparison. The optimized variant is based on *fixed-point arithmetic*, a technique in which fractional values are represented by integers multiplied by a constant scale factor, so that calculations are solved with integer operations instead of floating-point operations.

The optimized version complements the fixed point with three techniques: (i) *lookup tables*, in which certain values are pre-computed and queried during execution rather than recalculated; (ii) bit-shift operations instead of divisions; and (iii) the use of FFT, which reduces computational complexity from $O(N^2)$ to $O(N \log N)$ (Section 4). The reduction in execution time is attributed to techniques (i) and (ii) (Section 4).

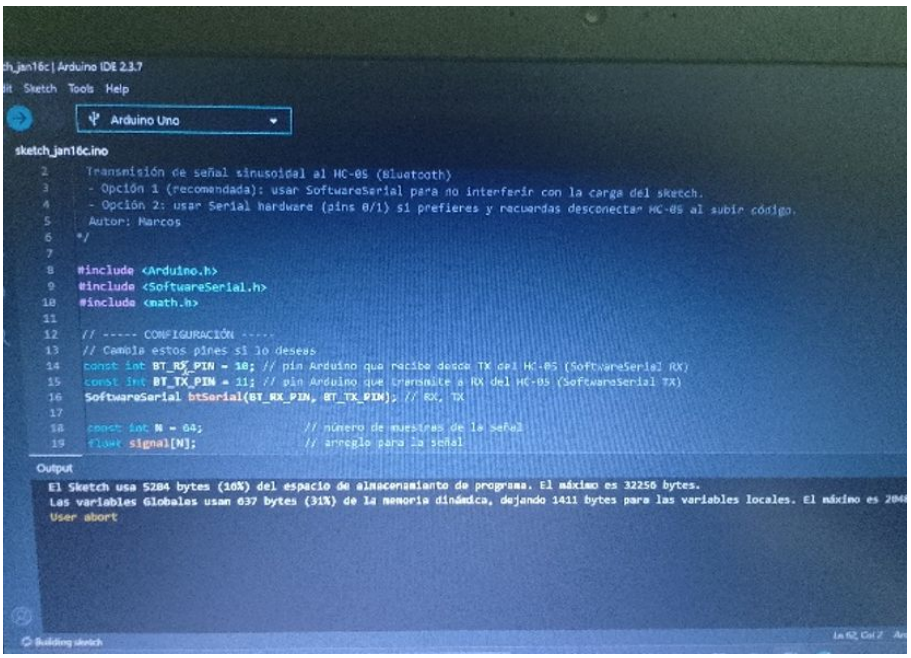


Figure 2. Capture of the Arduino IDE (version 2.3.7) with the signal transmission code to the HC-05 module via SoftwareSerial (pins 10 and 11; $N = 64$) and the compilation report: 5204 bytes (16%) of program memory and 637 bytes (31%) of dynamic memory.

3.4. Experimental procedure

The procedure consisted of running multiple iterations of each algorithmic version to evaluate the reproducibility and stability of the results. Three parameters were recorded in each execution: processing time, memory usage, and spectral accuracy of the simulated signal. The data were transmitted using the HC-05 module and stored on the computer for comparative analysis (Figure 3).

This methodological approach is inspired by the experimental logic described by Volpato et al. [13], in which the validation of an analytical method is achieved through repeated tests and sensitivity analyses. Similarly, this study verified both the consistency of the results obtained in the microcontroller and the reliability of the wireless transmission.

4. Results and Discussion

The experiments carried out with the Arduino Uno and the HC-05 module allowed to compare two spectral processing variants: a basic floating-point implementation and a fixed-point optimized version. The results revealed significant differences in execution time and memory usage (Tables 2 and 3), confirming the importance of applying optimization techniques in embedded systems with limited resources. In the basic version, the spectral energy calculation required execution times greater than 2500 μs for a window of 64 samples, while the optimized version reduced this value by more than 40%, reaching times close to 1500 μs. This improvement is attributed to the use of bit-shift operations instead of divisions and the implementation of precomputed lookup tables, in accordance with the findings reported by Ferreira [1] and Pereira [7] in their studies on FFT implementations in microcontrollers. Table 2 organizes the comparison between the two implementations and Table 3 presents the compilation report available in the manuscript (Figure 2).

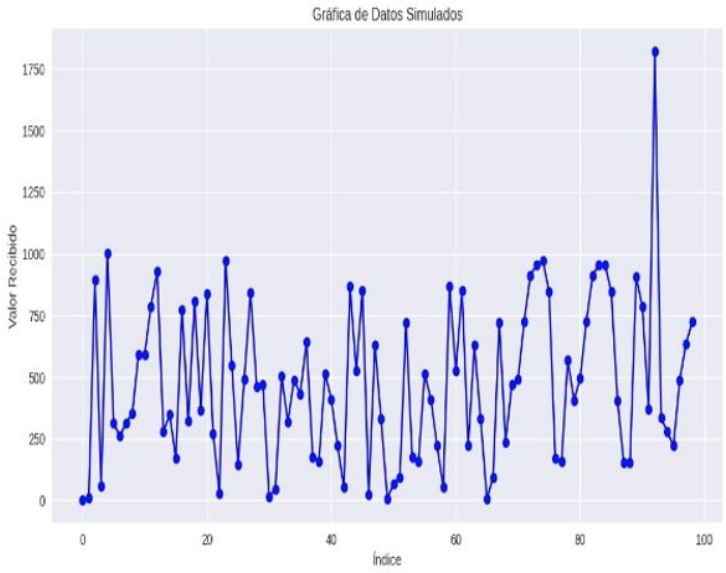


Figure 3. Simulated data received on the computer via the HC-05 module (value received as a function of the index).

Table 2. Comparison between basic and optimized implementation (Arduino Uno, window of 64 samples).

Parameter	Basic Implementation	Optimized Implementation
Arithmetic	Floating point	Fixed point
Optimization techniques	Not reported	Fixed point, precomputed lookup tables, bit shifts instead of divisions, and FFT
Runtime	> 2500 μs	≈ 1500 μs (reduction $> 40\%$)
SRAM usage	Higher than in the optimized version (no reported value)	Lower than in the basic version (no reported value)
Flash usage	Higher than in the optimized version (no reported value)	Lower than in the basic version (no reported value)
Spectral accuracy	Registered; no reported values	Registered; no reported values

Note: Average transmission latency with the HC-05, 20–30 ms (not reported by variant). "No reported value" indicates that the manuscript qualitatively describes the parameter or does not present figures.

Table 3. Memory usage according to the Arduino IDE compilation report (Figure 2).

Resource	Usage	Percentage	Maximum capacity
Program memory (Flash)	5204 bytes	16%	32,256 bytes
Dynamic memory (SRAM): global variables	637 bytes	31%	2048 bytes
Dynamic memory available for local variables	1411 bytes	—	—

Note: The manuscript does not indicate which variant (basic or optimized) this report corresponds to. Source: Figure 2.

From a mathematical perspective, the spectral analysis was based on the Discrete Fourier Transform (DFT), defined in Equation (1):

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N}kn} \tag{1}$$

where N is the number of samples in the analysis window ($N = 64$ in this paper), n is the index of the sample, k is the frequency index, and j is the imaginary unit. DFT allows the breakdown of a discrete signal into its frequency components.

The optimized implementation employed Fast Fourier Transform (FFT), reducing computational complexity from $O(N^2)$ to $O(N \log N)$, a key factor to achieve shorter execution times on limited hardware. The analysis of memory usage showed that the optimized version required less space in SRAM and Flash, thus freeing up resources for other critical tasks in real-time applications (Table 2). This observation coincides with the work of López et al. [4] and Ramírez [9], who emphasize the need to adjust algorithmic complexity to the limitations of embedded hardware. Regarding data transmission using the HC-05, an average latency of 20–30 ms was observed, sufficient for remote monitoring and communication in IoT systems. The stability of the connection and the reproducibility of the results were verified over multiple executions, following the experimental validation logic described by Volpato et al. [13] in their work on drug quantification in biological matrices.

The discussion of these results highlights that algorithmic optimization improves computational performance. Its effect on power consumption, an essential aspect in portable and low-power applications, was not measured in this study (Section 4.1). In addition, the integration of wireless communication through the HC-05 demonstrates the feasibility of implementing embedded systems capable of processing and transmitting information in real time, opening opportunities for applications in electrical monitoring, pattern recognition, and intelligent control systems. In summary, the results indicate that the optimization strategies applied reduce runtime and memory usage in the Arduino Uno, in line with current trends towards the use of low-cost embedded platforms with advanced processing capabilities; the balance with accuracy and power consumption requires additional quantification.

4.1. Implementation limitations

The limitations described below are derived from the conditions and the available data from the study. In terms of hardware and memory, the platform used (Arduino Uno, ATmega328P at 16 MHz, with 2 KB of SRAM and 32 KB of Flash) limits the available resources: in the compilation report in Figure 2, the program occupied 5204 bytes of program memory (16%) and the global variables occupied 637 bytes of dynamic memory (31%), leaving 1411 bytes for local variables, with a window of $N = 64$ samples (Table 3). This report is not associated with a specific variant and the memory usage of each version is only described qualitatively, so the memory savings of the optimized version cannot be quantified with the available data.

In terms of processing, the execution times (greater than 2500 μs in the basic version and close to 1500 μs in the optimized version) correspond to a single window length (64 samples), a single platform and a simulated signal; the reported results do not allow extrapolating the behavior to other window lengths, to real signals or to other microcontrollers. In addition, the number of iterations and the dispersion of the measurements are not reported, so the variability of the execution time is not quantified.

Regarding accuracy, although spectral precision was recorded as a parameter of the procedure (Section 3.4), no values or error metrics are presented; therefore, the statement that optimization does not compromise accuracy is not supported quantitatively by the available data. Finally, transmission with the HC-05 presented an average latency of 20–30 ms, which should be considered when evaluating real-time processing and which was not reported by variant, and energy consumption was not measured, so that references to its reduction correspond to what was reported in the literature and not to its own measurements.

5. Conclusions

The optimization of algorithms in microcontrollers is an essential aspect to ensure efficient signal processing in embedded systems, particularly in low-cost and resource-limited platforms such as the Arduino Uno. The results obtained in this work demonstrate that the implementation of optimization techniques, such as the use of fixed-point arithmetic, the reduction of operations through precomputed lookup tables, and the adaptation of the code to the specific characteristics of the hardware, can achieve significant improvements in

execution time and memory usage. The conservation of computational accuracy was recorded as a parameter of the study, but was not quantified in the results presented.

The integration of the HC-05 module as a wireless transmission medium validates the feasibility of sending processed data in real time to external systems, opening possibilities for applications in IoT, remote monitoring and intelligent control. This experimental approach confirms that algorithmic optimization impacts computational efficiency; its effect on energy consumption, a critical factor in portable and low-power environments, was not measured and remains as a future line of work. Overall, this work provides practical evidence of the importance of balancing algorithmic complexity with the constraints of embedded hardware, within the limits described in Section 4.1 (single platform, window of 64 samples, simulated signal and absence of quantitative accuracy and energy data). It aligns with current trends towards the development of low-power microcontrollers with advanced processing capabilities and integrated artificial intelligence. In this way, it offers a structured guide for researchers and developers looking to implement efficient signal processing solutions in resource-limited environments, contributing to the advancement of more sustainable and scalable embedded systems.

Author Contributions: **Marcos Daza:** Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Writing – original draft, Visualization. **Johan Mardini:** Conceptualization, Supervision, Project administration, Writing – review & editing. **Yaribeth Rodriguez:** Methodology, Formal analysis, Investigation, Resources, Writing – original draft, Writing – review & editing.

Johan Mardini passed away before the publication of this manuscript. The corresponding author confirms that, to the best of her knowledge, he met the criteria for authorship and would have approved the final version of the manuscript. The remaining authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable, since the present study does not involve human personnel or animals.

Informed Consent Statement: This study is limited to the use of technological resources, so no human personnel or animals are involved.

Acknowledgments: The authors dedicate this work to the memory of their colleague and co-author Johan Mardini, whose guidance and supervision were fundamental to the development of this research.

Conflicts of Interest: Under the authorship of this research, it is declared that there is no conflict of interest with the present research.

References

1. Ferreira, J. C. (2018). Desempenho comparativo das transformadas rápidas de Fourier e de Hartley implementadas no microcontrolador ESP8266. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, 6(1). CNMAC 2017. Accessed: Jan. 16, 2026.
2. Gómez, J. A. (2020). Implementación de la transformada rápida de Fourier en microcontroladores para aplicaciones de análisis de señales. *Revista Ciencia y Tecnología*, 1(23):45–56. Accessed: Jan. 16, 2026.
3. Hernández, J. A. and Pérez, M. L. (2018). Aplicación de la Transformada Discreta de Fourier para la reducción de ruido en audios. Universidad de La Laguna. Accessed: Jan. 16, 2026.
4. López, J. A., Hernández, J. C., and Ramírez, M. A. (2021). Implementación de la transformada rápida de Fourier en sistemas embebidos para análisis de señales. *Ciencia Latina Revista Científica Multidisciplinar*, 5(6):12345–12360. Accessed: Jan. 16, 2026.
5. Moreno Sánchez, A. A. (2019). Implementación de un algoritmo basado en FFT en un microcontrolador con motor DSP para la medición de distorsión armónica, potencia compleja, valores RMS, y factor de potencia de una carga monofásica para integrar en una red IoT. Monografía, Programa de Ingeniería Electrónica, Escuela Colombiana de Ingeniería Julio Garavito, Bogotá D.C., Colombia. Accessed: Jan. 16, 2026.
6. Paredes, J. C. (2016). Implementación de la transformada rápida de Fourier en microcontroladores para aplicaciones de análisis espectral. Proyecto de titulación, Escuela Politécnica Nacional, Quito, Ecuador. Accessed: Jan. 16, 2026.

7. Pereira, M. A. (2017). Acionamento de dispositivos via decodificações de sons utilizando a transformada rápida de Fourier em um microcontrolador. Monografía (pregrado en ingeniería de control y automatización), Escola de Minas, Universidade Federal de Ouro Preto, Ouro Preto, Brazil. Accessed: Jan. 16, 2026.
8. Pérez, Y. and Rodríguez, M. (2024). Aplicación de la transformada rápida de Fourier en microcontroladores para el análisis de señales eléctricas. *Ingeniería Electrónica, Automática y Comunicaciones*, 45(2):113–124. Accessed: Jan. 16, 2026.
9. Ramírez, J. L. (2018). Implementación de la transformada rápida de Fourier en un microcontrolador para aplicaciones de análisis de señales. Tesis de licenciatura, Facultad de Ingeniería, Universidad Autónoma de Querétaro, Querétaro, México. Accessed: Jan. 16, 2026.
10. Riaño, J., Ladino, C., and Martínez, F. (2012). Implementación de la transformada FFT sobre una FPGA orientada a su aplicación en convertidores electrónicos de potencia. *Tekhnê*, 9:21–32. Accessed: Jan. 16, 2026.
11. Silva, R. and Costa, M. (2022). Aplicación de la transformada rápida de Fourier en microcontroladores para monitoreo de señales eléctricas. *Anuário de Ciências e Tecnologia*, 12(2):77–89. Accessed: Jan. 16, 2026.
12. Torres Gómez, J. (2013). Demodulación de señales digitales mediante microcontroladores PIC18F4550. ResearchGate. Accessed: Jan. 16, 2026.
13. Volpato, N. M., Santi, P., Laureri, C., and Colombo, P. (1997). Assay of acyclovir in human skin layers by high-performance liquid chromatography. *Journal of Pharmaceutical and Biomedical Analysis*, 16:515–520.

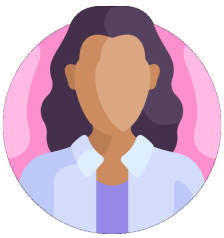
Authors' Biography



Marcos Daza is a student in the Electronic Engineering program at Universidad de la Costa.



Johan Mardini was a professor in the Faculty of Computer Science and Electronics at Universidad de la Costa.



Yaribeth Rodriguez is a student in the Electronic Engineering program at Universidad de la Costa.

Disclaimer/Editor's Note: Statements, opinions, and data contained in all publications are solely those of the individual authors and contributors and not of the OnBoard Knowledge Journal and/or the editor(s), disclaiming any responsibility for any injury to persons or property resulting from any ideas, methods, instructions, or products referred to in the content.