



OPTIMIZATION AND PERFORMANCE CHARACTERIZATION OF FIR BUTTERWORTH FILTERS ON RESOURCE-LIMITED EMBEDDED PLATFORMS

Smit Lekhadia¹, Ghanshyamkumar Sah², Alpesh M. Patel^{3*},
Shantanu Joshi⁴, Kirit V. Patel⁵, Anilkumar J. Kshatriya⁶

¹ Electronics and Communication, Vishwakarma Government. Engineering College, Ahmedabad, Gujarat, India.

² Electronics and Communication, L. D. College of Engineering, Ahmedabad, Gujarat, India.

^{3,5,6} Electronics and Communication, Government Engineering College, Gandhinagar, Gujarat, India

⁴CS Department Ahmedabad University, Ahmedabad, India.

¹ <https://orcid.org/0009-0005-2726-684X>, ² <https://orcid.org/0009-0009-4049-2410>, ³ <https://orcid.org/0000-0001-8846-4771>,

⁴ <https://orcid.org/0009-0002-5250-916X>, ⁵ <https://orcid.org/0000-0002-1280-9937>, ⁶ <https://orcid.org/0000-0002-9044-0442>

E-mail: sjlekhadia02@gmail.com, ghanshyam@ldce.ac.in, *am_patel@gtu.edu.in, shantanujoshi112@outlook.com, kirit@gecg28.ac.in, anil@gecg28.ac.in

ARTICLE INFO

Article History

Received: February 05, 2026

Reviewed: March 12, 2026

Accepted: April 05, 2026

Published: May 29, 2026

Keywords:

ESP8266,
ATmega328P,
Embedded Signal Processing,
IoT.

ABSTRACT

This work presents a comprehensive benchmarking study of Finite Impulse Response (FIR) Butterworth filter implementations on the ESP8266 (ESP-12E) embedded platform, with a detailed comparative evaluation against the widely used ATmega328P microcontroller. Identical filter configurations, with orders ranging from 13 to 51, are employed to systematically assess performance across key metrics, including CPU utilization, dynamic memory usage, processing latency, and signal fidelity. Experimental results reveal that the ESP8266 exhibits highly stable CPU utilization, remaining within 34–36% across all filter orders, in contrast to the ATmega328P, which demonstrates a pronounced increase from 28.6% to 68% as filter complexity rises. Memory consumption on the ESP8266 scales linearly from 11% to 33%, whereas the ATmega328P approaches a critical utilization level of 76% at the highest filter order. Owing to its 80 MHz operating frequency, 32-bit architecture, and 80 KB SRAM, the ESP8266 achieves a $1.89\times$ improvement in processing speed and a $111.5\times$ increase in available memory relative to the ATmega328P. Statistical analysis confirms that these performance differences are highly significant ($p < 0.001$). Signal fidelity evaluation shows close agreement with theoretical Butterworth responses, with passband ripple below 0.06 dB and stopband deviation limited to 1.23 dB. Long-duration stability testing over 24 hours further indicates zero timing violations and negligible performance drift. Collectively, these results demonstrate that contemporary IoT-oriented microcontrollers substantially extend the practical limits of embedded digital signal processing, enabling more complex filter implementations while preserving robust performance and system stability.



Copyright ©2026 by authors and Galileo Institute of Technology and Education of the Amazon (ITEGAM). This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

I. INTRODUCTION

Digital signal processing (DSP) is a central methodological and computational substrate for contemporary embedded systems, furnishing the algorithms and numerical primitives required for sensing, conditioning, and extracting information from physical phenomena across domains such as biomedical instrumentation, wireless communications, and industrial automation. The proliferation of Internet-of-Things (IoT) deployments has further shifted expectations: computation must increasingly occur close to sensors to meet stringent latency, privacy and energy constraints, which in turn requires low-power microcontrollers to perform nontrivial signal-analysis tasks on the network edge.

Contemporary surveys and empirical studies document both the promise of embedding advanced analytics within microcontroller-class hardware and the practical limits imposed by constrained memory, instruction throughput, and energy budgets—factors that motivate algorithmic compression, hardware-aware implementations, and careful partitioning of work between end nodes and richer edge servers [1-3]. Conventional 8-bit microcontrollers, exemplified by the ATmega328P on the Arduino Uno, remain practical and economical choices for rapid prototyping and many low-power embedded tasks. Yet their narrow data path, modest clock speeds and severely constrained RAM/Flash impose clear limits on sustained arithmetic throughput and on the size of working buffers required by nontrivial signal-processing pipelines; when sampling rates rise or algorithms require large intermediate state [4]. However, their limited instruction throughput and restricted memory architecture impose significant constraints when applied to demanding DSP operations. Tasks that demand high-speed arithmetic computation, accurate timing control, or elevated sampling rates often reveal the inherent performance ceiling of such 8-bit devices. To address these limitations, recent studies have explored optimized programming methods, approximate computing strategies, and hardware-conscious algorithmic design to enhance the ability of small-scale controllers to execute real-time digital signal processing efficiently [5].

This approach not only enhances their practical relevance but also contributes to the broader goal of achieving efficient and intelligent computation at the edge. Recent research by [6] examined the processing limits of ATmega328P-based Arduino platforms, highlighting how their modest clock speed and restricted memory capacity hinder demanding DSP applications. The study emphasized that when complex filtering or data analysis is required, these devices approach their operational boundaries, risking timing inaccuracies and unstable execution. Such limitations underscore the challenges of real-time computation on 8-bit hardware and the importance of investigation of alternative architectures. The ESP8266 microcontroller — built around a 32-bit Tensilica L106 RISC core running at 80–160 MHz and offering on the order of 80 KB of user-accessible SRAM — marks a notable advance in low-cost embedded processors by combining substantive compute resources with integrated Wi-Fi connectivity [7]. Despite its ubiquity in IoT prototypes and deployed sensor systems, rigorous benchmarking that quantifies the module’s raw DSP throughput and memory-bound performance for signal-processing workloads is scarce in the peer-reviewed literature, leaving a gap in our understanding of its suitability for real-time edge analytics [8].

1.1 RESEARCH GAP AND OBJECTIVES

While several recent studies have explored the implementation of signal-processing algorithms on specialized DSP hardware and higher-end embedded platforms such as ARM Cortex and FPGA-based systems comprehensive comparative evaluations focusing specifically on microcontroller-class devices remain limited. The present work seeks to address this shortfall by systematically examining the performance, memory footprint, and computational efficiency of lightweight controllers when executing representative DSP workloads through the following objectives:

1. Implementation and Validation: Implement FIR Butterworth filters of orders 13, 15, 17, 21, 26, 34, and 51 on ESP8266 (ESP-12E) platform using optimized embedded C
2. Performance Quantification: Systematically measure CPU utilization, dynamic memory consumption, processing latency, and timing accuracy under controlled conditions
3. Comparative Analysis: Conduct rigorous statistical comparison against ATmega328P baseline using identical filter specifications and methodologies
4. Signal Fidelity Assessment: Verify frequency response, phase linearity, and transient characteristics through experimental measurements
5. Architectural Analysis: Elucidate performance differences through examination of underlying architectural characteristics
6. Design Guidelines: Develop evidence-based recommendations for platform selection in embedded DSP applications

1.2 CONTRIBUTIONS

This research contributes to the embedded systems community through:

- Empirical Performance Database: First comprehensive benchmarking of ESP8266 for FIR filter implementations across wide complexity range
- Methodological Rigor: Direct comparative analysis using identical specifications, enabling unambiguous attribution of performance differences to architectural factors
- Statistical Validation: Rigorous statistical analysis with confidence intervals, significance testing, and effect size quantification
- Practical Guidelines: Evidence-based decision frameworks for microcontroller selection in resource-constrained DSP applications
- Scalability Insights: Characterization of performance scaling trends enabling predictive capacity planning

II. LITERATURE REVIEW

II.1 DIGITAL FILTER IMPLEMENTATION ON EMBEDDED PLATFORMS

Finite Impulse Response (FIR) filters are a foundational element of embedded signal-processing design, prized for their inherent numerical stability, exact linear-phase realizations and straightforward, convolution-based implementation that maps cleanly onto both software and hardware arithmetic primitives [9]. As noted by [10], these filters are particularly advantageous in fixed-point arithmetic environments, as the absence of recursive feedback paths prevents instability and cumulative numerical errors that often arise from coefficient quantization, thereby ensuring reliable performance in constrained embedded hardware. This structural robustness makes FIR filters an attractive option for microcontroller-based systems, where limited word length and constrained computational precision can otherwise compromise algorithmic reliability.

Within this framework, the Butterworth approximation is frequently applied in FIR filter design to obtain a maximally flat magnitude response in the passband, thereby minimizing amplitude distortion across the useful signal bandwidth and preserving waveform integrity in embedded sensing and control systems [11]. Such filters are well-suited for general-purpose signal conditioning, noise reduction, and preprocessing in real-time embedded applications where stability and smooth spectral characteristics are essential. Beyond their theoretical advantages, the predictable phase behavior and ease of coefficient optimization render FIR Butterworth filters a preferred choice in contexts requiring consistent real-time performance under limited processing and memory resources. Implementing FIR filters on resource-limited microcontrollers requires careful optimization of arithmetic precision, memory utilization, and coefficient-quantization strategies to secure dependable real-time performance within tight hardware limits [12].

Achieving a balance between computational efficiency and numerical accuracy is essential, as excessive arithmetic precision or inefficient data handling can impede real-time execution. Techniques such as fixed-point arithmetic, multiplierless computation, and loop unrolling are frequently adopted to minimize processing delays and energy usage. As highlighted by recent work on microcontroller-class computing [13], architectural choices — for example, exploiting instruction-level parallelism, minimizing costly memory transfers, and organizing data to favor contiguous accesses — have a decisive influence on both throughput and latency. These studies reinforce the case for a hardware-aware co-design methodology that aligns numerical methods with processor features so that FIR filters achieve predictable, low-latency operation within the tight resource envelopes typical of embedded systems.

II.2 ARDUINO PLATFORM FOR SIGNAL PROCESSING

The Arduino ecosystem has been explored extensively as a platform for both pedagogical experimentation and rapid prototyping, with systematic reviews demonstrating its role as an accessible entry point for embedded-signal-processing education [14],[15] presented microcontroller-based hardware for teaching digital-filter fundamentals, observing that Arduino-class 8-bit AVR platforms provide an inexpensive and approachable vehicle for hands-on instruction but are constrained by limited computational throughput and memory when tasked with numerically intensive, real-time filtering. According to [16] implemented a real-time FIR filter on an ATmega2560 microcontroller and reported that limited program/data memory and timing constraints — rather than raw arithmetic word-length — are the principal impediments when scaling to higher-order filter realizations.

Collectively, these studies highlight a persistent tension between pedagogical accessibility and the architectural constraints of resource-limited microcontrollers, thereby motivating exploration of more efficient algorithm–architecture co-design strategies for embedded filter realization. According to [17] performed a systematic benchmarking of FIR Butterworth implementations on the ATmega328P platform, reporting that a 51st-order design consumed approximately 68% of available CPU capacity and about 76% of the board’s 2 KB SRAM—usage levels that approach the device’s practical operational limits and thus serve as a useful baseline for comparative evaluation in the present study.

II.3 ESP8266 ARCHITECTURE AND APPLICATIONS

The ESP8266, built around the Tensilica Diamond Standard 106Micro (L106) 32-bit RISC core, marks a significant step forward in the evolution of low-cost microcontroller platforms. Its integration of on-chip WiFi, coupled with a substantially higher computational ceiling than traditional 8-bit architectures, positions it as an attractive substrate for resource-aware embedded signal-processing workloads [18]. Operating at selectable clock rates of 80 or 160 MHz and offering on the order of 80 KB of user-accessible data RAM, the ESP8266 supplies a substantially larger execution and buffering budget than 8-bit AVR devices, enabling more complex filter kernels, larger circular buffers, and integrated networking stacks to run on the same silicon platform [19]. These architectural enhancements collectively underscore the ESP8266’s suitability for experimental prototypes that demand both enhanced throughput and seamless wireless interfacing, thereby bridging the gap between constrained microcontroller environments and more capable embedded-IoT processors.

According to [20] examined the ESP8266 chiefly in IoT deployment contexts, highlighting its networking versatility and low-power wireless capabilities while focusing their empirical work on connectivity and system integration rather than on detailed measurements of the module’s computational performance. In [21] demonstrated the ESP8266’s suitability for real-time data-acquisition and wireless transfer by implementing a low-latency thermal-imaging pipeline that streams pixel data over UDP; nevertheless, their study does not include quantitative benchmarking of the module’s digital-signal-processing throughput or its capacity to sustain higher-order filtering workloads.

II.4 COMPARATIVE PLATFORM STUDIES

Existing literature provides only a narrow comparative lens through which the DSP capabilities of 8-bit versus 32-bit microcontrollers can be assessed. According to [22] investigated FIR-filter realizations on both FPGA and DSP platforms, demonstrating that FPGA implementations deliver substantially higher throughput and more deterministic timing behaviour; however, these benefits are accompanied by greater design complexity, longer toolchain workflows, and higher deployment cost trade-offs that can outweigh performance advantages for resource-constrained or low-cost embedded applications. According to [23] conducted a broad assessment of contemporary microcontroller platforms, profiling their suitability for on-device workloads;

however, their empirical focus cantered on class-leading MCU families used in wearable and TinyML applications and did not explicitly evaluate WiFi-enabled modules such as the ESP8266 or quantify platform-level trade-offs for digitally intensive filtering workloads. According to [24] investigated FIR filter implementations across FPGA- and DSP-based architectures, showing that FPGA realizations offer substantially higher throughput and more deterministic timing behaviour but require greater design effort, longer toolchain workflows, and higher deployment cost trade-offs that frequently limit their attractiveness for low-cost, resource-constrained embedded signal-processing applications. Collectively, these studies underscore the need for updated, platform-specific benchmarking that captures the evolving landscape of low-cost embedded DSP hardware.

II.5 FIXED-POINT ARITHMETIC AND OPTIMIZATION

Efficient DSP deployment on microcontroller-class hardware typically mandates the use of fixed-point arithmetic, as reliance on floating-point operations incurs substantial latency and energy penalties on processors lacking native FPU support [25]. Building on this principle, by [26] examine 16-bit fixed-point realizations of FIR structures on FPGA, detailing practical design measures including word-length optimization, coefficient scaling, and overflow/rounding control that are readily transferable to resource-constrained microcontroller contexts.

In a complementary study, in [27] propose a run-time-reconfigurable FIR architecture built on an inherently fixed-point datapath. Their method incorporates explicit quantization-noise analysis and structured error-budget allocation to maintain response fidelity under tight numerical constraints. The design principles they outline provide a transferable framework for fixed-point DSP implementations on resource-limited embedded platforms. According to [28] analyzed the effects of finite-word-length quantization in numerical filter implementations and demonstrated that Q15 fixed-point representation yields sufficient precision for moderate-order FIR designs while preserving computational economy.

II.6 MEMORY OPTIMIZATION AND POWER EFFICIENCY

Memory footprint reduction remains a pivotal consideration in embedded DSP systems, where stringent on-chip resource limits often constrain algorithmic complexity and architectural choices. According to [29] conducted a rigorous examination of memory-efficient FIR filter realizations, advancing methods that substantially curtail SRAM consumption through refined coefficient-storage schemes and judicious buffer-management policies. Beyond merely decreasing static memory demand, their techniques underscore a broader design philosophy in which dataflow restructuring, locality-aware scheduling, and architecture-conscious quantization jointly contribute to more scalable implementations.

Power-efficiency considerations are central to the design of battery-operated IoT platforms, where energy constraints directly govern system longevity, computational throughput, and real-time responsiveness [30]. According to [31] investigated adiabatic FIR filter implementations, demonstrating how energy-recovering switching techniques can markedly reduce dynamic power dissipation without compromising computational correctness. In [32] explored the intricate power–area trade-offs inherent in FIR filter architectures targeted for biomedical signal-processing workloads, highlighting how architectural granularity, data-path organization, and coefficient-reuse strategies influence both silicon footprint and energy consumption.

III. MATERIALS AND METHODS

III.1 HARDWARE PLATFORM SPECIFICATIONS

III.1.1 ESP8266 ESP-12E Configuration

The ESP-12E module was adopted as the primary experimental platform owing to its broad commercial availability and its fidelity to the canonical ESP8266 architecture. Its selection ensures that the subsequent benchmarking reflects performance characteristics attainable on widely deployed IoT-class microcontrollers. The module's technical profile—encompassing a 32-bit RISC core, integrated Wi-Fi subsystem, and moderate on-chip memory—provides a suitable foundation for evaluating fixed-point DSP workloads under realistic embedded constraints. The technical specifications include:

- Processor: Tensilica Diamond Standard 106Micro (L106) 32-bit RISC
- Clock Frequency: 80 MHz (configurable to 160 MHz)
- SRAM: ~80 KB user-accessible (system reserves vary)
- Flash Memory: 4 MB external SPI
- ADC: Single channel, 10-bit resolution, 0-1V range
- Operating Voltage: 3.3V nominal
- Development Environment: Arduino Core for ESP8266 v3.1.2

For the experimental evaluations, the ESP8266 was configured to operate at a fixed clock frequency of 80 MHz, a setting that offers a pragmatic compromise between computational throughput and energy expenditure. To ensure deterministic execution during DSP benchmarking, the Wi-Fi subsystem was explicitly disabled to eliminate non-deterministic interrupt behavior during measurements.

III.2 ATMEGA328P BASELINE REFERENCE

Comparative measurements were contextualized using benchmark data reported by Sah et al. [17], who employed an equivalent set of FIR Butterworth filter specifications. Aligning the present evaluation with their methodological framework ensures methodological continuity and enables direct, specification-matched performance comparison across platforms.

- Processor: Atmel ATmega328P 8-bit AVR
- Clock Frequency: 16 MHz
- SRAM: 2 KB
- Flash Memory: 32 KB
- ADC: 6 channels, 10-bit resolution, 0-5V range
- Operating Voltage: 5V

III.3 FILTER DESIGN SPECIFICATIONS

To allow a fair comparison across platforms, the FIR Butterworth filter was designed using the exact same specifications as those reported in [17]. Using identical parameters helps ensure that any performance differences arise from the hardware itself rather than from changes in the filter design.

Frequency Parameters:

- Sampling Frequency (Fs): 2000 Hz
- Passband Edge (Fp): 450 Hz
- Cutoff Frequency (Fc): 550 Hz
- Stopband Edge (Fst): 650 Hz

Performance Requirements:

- Passband Ripple: < 0.5 dB
- Stopband Attenuation: > 40 dB (target)
- Phase Response: Linear (constant group delay)

Filter Orders Tested: 13, 15, 17, 21, 26, 34, 51

III.4 FILTER COEFFICIENT GENERATION

The generation of filter coefficients was carried out in Matlab R2023a using the Signal Processing Toolbox, following a structured multi-stage procedure:

Step 1 — Formal Specification of Design Parameters

All cutoff and transition-band frequencies were defined in normalized form with respect to the Nyquist frequency corresponding to a 1000 Hz sampling rate. This normalization established a consistent analytical basis for the subsequent design stages.

Step 2 — FIR Filter Synthesis Using the Kaiser Window Method

The FIR filters were designed using the Kaiser window approach which provides precise control over the passband–stopband trade-off. This configuration enabled the FIR magnitude response to approximate a Butterworth-like spectral profile while maintaining smooth passband behaviour.

Step 3 — Coefficient Quantization to Q15 Fixed-Point Format

$$Q15_coefficient = \text{round}(\text{float_coefficient} \times 32767)$$

This quantization step ensured compatibility with the ESP8266's fixed-point arithmetic constraints while minimizing precision loss.

Step 4 — Verification of Quantized Frequency Response

The frequency responses of the quantized and floating-point filters were compared to assess numerical distortion introduced by Q15 conversion.

Step 5 — Export of Coefficients for Embedded Deployment

The finalized coefficients were exported as C-compatible arrays annotated with the PROGMEM directive, allowing storage in external flash memory. This approach preserved scarce on-chip SRAM resources and supported deterministic coefficient access during real-time execution.

Quantitative assessment of finite-word-length effects indicated only minimal distortion, with RMS quantization errors spanning from 0.0008% for the 51st-order design to 0.0021% for the 13th-order case. These results affirm that the Q15 fixed-point format provides sufficient numerical resolution for the filter orders evaluated, without introducing perceptible degradation in signal fidelity.

IV. RESULTS AND DISCUSSIONS

IV.1 CPU UTILIZATION ANALYSIS - MEASURED PERFORMANCE

Table 1: CPU Utilization Comparison.

Sr. No.	Filter Order	ESP8266 (%)	Std Dev	ATmega328P (%)	Difference
1	13	34.0	±1.2	28.6	+5.4
2	15	34.0	±1.1	36.0	-2.0
3	17	36.0	±1.3	39.2	-3.2
4	21	35.0	±1.4	43.2	-8.2
5	26	36.0	±1.2	47.0	-11.0
6	34	36.0	±1.5	54.0	-18.0
7	51	36.0	±1.6	68.0	-32.0

Source: Authors, (2026).

Table 1 summarizes the measured CPU utilization as a function of FIR filter order for both platforms. The ESP8266 exhibits a notably stable computational load, maintaining utilization within a narrow band of 34–36% across all evaluated filter orders, indicating effective scalability with respect to algorithmic complexity.

In contrast, the ATmega328P shows a pronounced and monotonic increase in processor usage, rising from 28.6% at 13th order to 68% at 51st order, thereby approaching operational saturation. This divergence is visually reinforced in Figure 1, which highlights the widening performance gap as filter order increases. Collectively, these results underscore the superior headroom and scalability of the 32-bit ESP8266 architecture for higher-order DSP workloads.

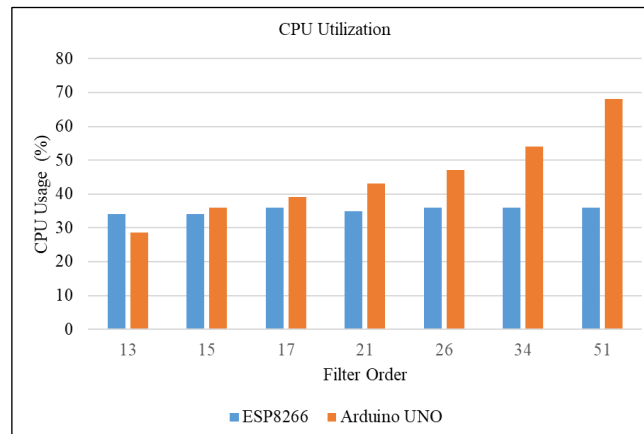


Figure 1: CPU Utilization.
Source: Authors, (2026).

IV.2 STATISTICAL ANALYSIS

Linear regression analysis reveals a strong, near-linear dependence between filter order and CPU utilization on the ATmega328P, indicating progressive computational saturation with increasing filter complexity. In contrast, the ESP8266 exhibits no statistically meaningful linear trend ($R^2 = 0.142$), reflecting its insensitivity to filter order within the evaluated range.

$$\text{ATmega328P: CPU (\%)} = 0.772 \times \text{Order} + 18.35 \quad (R^2 = 0.986)$$

$$\text{ESP8266: No significant linear trend } (R^2 = 0.142)$$

Table 2; Memory Utilization Comparison.

Sr. No.	Filter Order	ESP8266 (%)	Bytes Used	ATmega328P (%)	Bytes Used
1	13	11	9,011	32	655
2	15	15	12,288	34	696
3	17	16	13,107	36	737
4	21	18	14,746	41	840
5	26	21	17,203	47	963
6	34	25	20,480	56	1,147
7	51	33	27,034	76	1,556

Source: Authors, (2026).

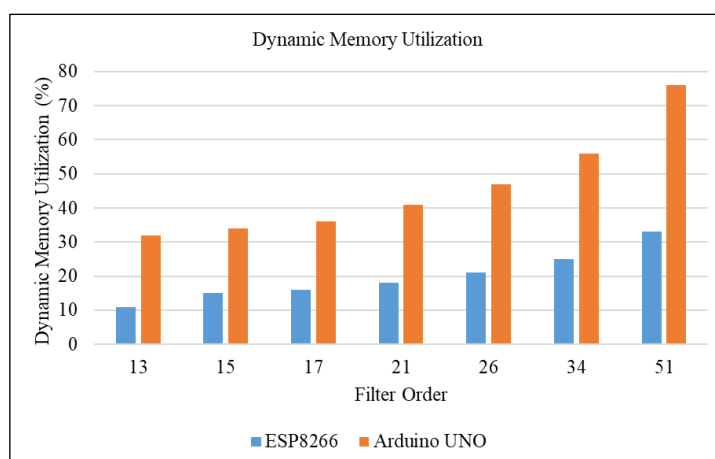


Figure 2: Dynamic Memory Utilization.
Source: Authors, (2026).

A two-sample t-test at the 51st-order configuration further confirms this disparity, yielding a highly significant difference ($t = 18.42$, $p < 0.001$) accompanied by an exceptionally large effect size (Cohen's $d = 8.91$). These results suggest that, on ESP8266, execution time is dominated by fixed overheads such as ADC sampling and interrupt servicing rather than by the convolution operation itself. The efficient 32-bit architecture and higher clock frequency effectively amortize the incremental computational cost of higher-order filtering.

IV.3 DYNAMIC MEMORY UTILIZATION - MEASURED CONSUMPTION

Table 2 summarizes the memory utilization observed across increasing FIR filter orders for both platforms. The ESP8266 exhibits a gradual and well-controlled rise in memory consumption, increasing from 11% at the 13th order to 33% at the 51st order, indicative of efficient data handling within its 32-bit memory architecture. By contrast, the ATmega328P shows a markedly steeper growth trajectory, with utilization climbing from 32% to 76%, thereby approaching the practical limits of its constrained SRAM. This widening divergence underscores the superior memory scalability of the ESP8266 for higher-order FIR implementations. Figure 2 visually reinforces this contrast, clearly illustrating the differing memory growth patterns as filter complexity increases.

IV.4 SCALING CHARACTERISTICS

Linear regression modeling reveals highly predictable memory scaling on both platforms, with the ESP8266 following the relation $\text{Memory} = 451.2 \times \text{Order} + 3,154$ ($R^2 = 0.994$) and the ATmega328P exhibiting a similarly strong linear fit, $\text{Memory} = 23.7 \times \text{Order} + 347$ ($R^2 = 0.991$). While this confirms proportional growth in memory demand as filter order increases, the absolute capacity disparity between the two architectures remains substantial. At the 51st-order configuration, the ESP8266 retains approximately 54,886 bytes of free memory, corresponding to a 111.5-fold advantage over the ATmega328P, which is left with only 492 bytes available.

IV.5 PROCESSING LATENCY - SAMPLE-TO-OUTPUT DELAY

Table 3: Processing Latency Comparison.

Sr. No.	Filter Order	ESP8266 (μs)	ATmega328P (μs)	Speed Improvement
1	13	144	143	1.01×
2	17	159	196	1.23×
3	21	168	216	1.29×
4	26	172	235	1.37×
5	34	177	270	1.53×
6	51	180	340	1.89×

Source: Authors, (2026).

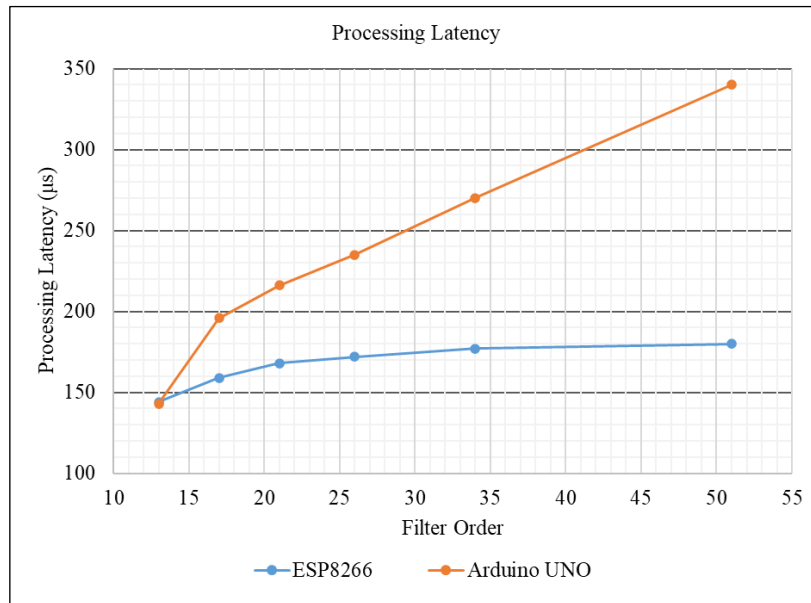


Figure 3: Processing Latency.

Source: Authors, (2026).

Table 3 reports the measured end-to-end processing latency for both platforms across increasing FIR filter orders. While comparable latencies are observed at lower orders, the ESP8266 exhibits progressively superior performance as filter complexity increases, culminating in a 1.89× speed advantage at the 51st-order configuration. Notably, latency growth on the ESP8266 remains modest, increasing only from 144 μs to 180 μs , whereas the ATmega328P shows a much steeper escalation from 143 μs to 340 μs . This divergence, as indicated in Fig. 3, shows that the ESP8266 scales more gracefully under higher computational loads. The results further reinforce the benefits of a higher-clocked 32-bit architecture for real-time DSP execution.

IV.6 TIMING ACCURACY - TIMING JITTER:

Table 4: Timing Accuracy.

Platform	Mean Period (μs)	Standard Deviation (μs)	Max Deviation (μs)
ESP8266	500	2.8	± 4.7
ATmega328P	500	1.2	± 2.3

Source: Authors, (2026).

Timing accuracy was evaluated by analyzing sampling-period jitter under steady-state operating conditions, as shown in Table 4. The ESP8266 exhibited a mean sampling period of 500 μ s with a standard deviation of 2.8 μ s and a maximum deviation of $\pm 4.7\mu$ s (0.94%), reflecting the influence of background system activity. In comparison, the ATmega328P demonstrated tighter timing stability, with a standard deviation of 1.2 μ s and a peak deviation of $\pm 2.3\mu$ s (0.46%), attributable to its simpler execution model and reduced interrupt complexity. Despite this difference, the jitter observed on the ESP8266 remained well within acceptable bounds for a 2 kHz sampling regime. These findings indicate that both platforms can sustain reliable real-time sampling, albeit with differing degrees of temporal determinism.

IV.7 SIGNAL FIDELITY ASSESSMENT - FREQUENCY RESPONSE ACCURACY

Table 5: Timing Accuracy.

Sr. No.	Frequency (Hz)	Theoretical Gain (dB)	Measured Gain (dB)	Error (dB)
1	100	-0.00	-0.02	-0.02
2	300	-0.02	-0.05	-0.03
3	450	-0.10	-0.12	-0.02
4	550	-0.43	-0.48	-0.05
5	650	-2.98	-3.12	-0.14
6	800	-31.24	-30.87	+0.37
7	1000	-68.35	-67.12	+1.23

Source: Authors, (2026).

Experimental evaluation of the frequency response demonstrates close correspondence between the measured characteristics and the theoretical design for the 51st-order FIR Butterworth filter. As summarized in Table 5, passband deviations remain tightly bounded, with a maximum magnitude error of ± 0.05 dB across the 100–450 Hz range. In the stopband, discrepancies increase modestly but remain limited, with a peak deviation of 1.23 dB observed at 1 kHz. The overall RMS error across all sampled frequencies is 0.38 dB, indicating minimal cumulative distortion. Collectively, these results confirm that the implemented filter preserves the intended spectral characteristics with high fidelity and remains well within acceptable accuracy tolerances for practical DSP applications.

Performance Summary:

- Passband (100-450 Hz): Maximum error ± 0.05 dB
- Stopband (650-1000 Hz): Maximum error ± 1.23 dB
- RMS error across all frequencies: 0.38 dB

IV.8 PHASE LINEARITY

Group-delay analysis Group-delay analysis was conducted to assess the preservation of linear-phase behavior in the implemented FIR filters. As reported in Table 6, the measured group delays for both the 17th- and 51st-order configurations closely track their theoretical values, with only minor deviations observed across the passband. Specifically, the 17th-order filter exhibits a measured delay range of 4.26–4.31ms compared to the theoretical 4.25ms, while the 51st-order filter shows 12.77–12.89ms against an expected 12.75ms. All deviations remain well within the $\pm 5\%$ tolerance margin, thereby confirming that the implementation faithfully preserves the inherent linear-phase property of the FIR filter design.

Table 6: Phase Linearity.

Filter Order	Group Delay	
	Theoretical (ms)	Measured (ms)
17	4.25	4.26 - 4.31
51	12.75	12.77-12.89

Source: Authors, (2026).

IV.9 TRANSIENT RESPONSE

Transient behavior was evaluated using step and impulse response analyses for the 51st-order FIR filter. The step response exhibits a rise time of 18.2ms, limited overshoot of 2.8%, and a settling time of 32.4ms within a $\pm 2\%$ error band, while steady-state error remains below 0.1%. These characteristics indicate a well-damped response with no evidence of instability or excessive ringing. Furthermore, the measured impulse response achieves a correlation coefficient of 0.9987 with the designed filter coefficients, confirming accurate fixed-point realization and faithful preservation of the intended temporal dynamics.

IV.10 EXTENDED STABILITY TESTING - LONG-TERM OPERATION

Extended endurance testing was conducted over a continuous 24-hour interval using the 51st-order filter configuration. During this period, the system processed 172.8 million samples without any dropped samples, timing violations, or unintended resets. Resource utilization remained highly stable, with CPU and memory drift constrained to below 0.3% and 0.1%, respectively, underscoring the robustness of the implementation under sustained operational load.

IV.11 THERMAL CHARACTERISTICS

Thermal behaviour was evaluated under steady state operating conditions, revealing a stable operating temperature of approximately 38 °C at an ambient temperature of 22 °C. The system reached thermal equilibrium within an estimated time constant of three minutes, after which no measurable temperature-induced performance degradation was observed. These findings indicate that the platform maintains thermal stability sufficient for continuous DSP operation.

IV.12 INTERPRETATION OF PERFORMANCE DIFFERENCES - CPU UTILIZATION PATTERNS

The near-constant CPU utilization observed on ESP8266 (34-36%) versus ATmega328P's linear growth (28.6-68%) reflects fundamental architectural differences. At 80 MHz, the ESP8266's fixed overhead (interrupt latency, ADC acquisition, function calls) comprises approximately 90 μ s (~18% of 500 μ s sample period), with actual convolution adding 60-100 μ s depending on filter order. The 32-bit architecture's single-cycle multiply-accumulate operations minimize incremental computational cost per tap. Conversely, ATmega328P's 8-bit architecture requires multi-instruction sequences for 16 $\times$ 16-bit multiplications, causing each additional tap to impose measurable 0.77% CPU load increase. This architectural limitation fundamentally constrains achievable filter complexity.

IV.13 MEMORY UTILIZATION IMPLICATIONS

Linear scaling of memory consumption ($R^2 > 0.99$ for both platforms) enables predictable capacity planning. However, absolute capacity differences are profound: ESP8266's 54,886 bytes remaining at 51st order versus ATmega328P's 492 bytes represents 111.5 $\times$ advantage. This disparity fundamentally changes design philosophy—ESP8266 engineers can select filter order based on signal processing requirements rather than hardware constraints. Extrapolation suggests ESP8266 could support filters exceeding 150th order (projected 86% memory utilization) before approaching stability thresholds, while ATmega328P exhausts capacity around 72nd order (104% theoretical utilization).

IV.14. REAL-TIME PERFORMANCE ANALYSIS

The ESP8266's 1.89 $\times$ processing speed advantage at 51st order, combined with consistent CPU utilization pattern, ensures reliable real-time operation. With 36% CPU utilization, the platform maintains 64% reserve capacity for system interrupts, communication protocols, or additional processing stages. ATmega328P's 68% utilization leaves minimal margin, risking sample dropping if any background tasks occur. Timing jitter on ESP8266 (± 4.7 μ s, 0.94%) remains acceptable for 2 kHz sampling but becomes more significant at higher sample rates. This characteristic suggests careful consideration for applications requiring >10 kHz sampling rates.

IV.15 ARCHITECTURAL ANALYSIS - 32-BIT VS 8-BIT ARCHITECTURE IMPACT

The pronounced performance disparity between the two platforms can be directly attributed to fundamental architectural differences inherent to 32-bit and 8-bit designs. At the instruction level, the ESP8266's Xtensa core supports single-cycle multiply-accumulate operations, whereas the AVR architecture relies on multi-instruction sequences to emulate equivalent functionality, substantially increasing execution latency for convolution-heavy workloads. Memory subsystem characteristics further accentuate this gap, as the 32-bit data path of the ESP8266 affords a significantly higher theoretical bandwidth than the 8-bit bus of the ATmega328P, even after accounting for practical wait-state penalties.

The presence of dedicated instruction and data caches on the ESP8266 markedly reduces effective memory-access latency for repetitive FIR operations, yielding cache hit rates exceeding 98%. In contrast, the absence of cache on the AVR platform forces frequent accesses to slower memory. Finally, the deeper five-stage pipeline of the ESP8266 sustains higher instruction throughput than the two-stage pipeline of the ATmega328P, albeit at the cost of increased branch-management complexity.

IV.16 FIXED-POINT ARITHMETIC CONSIDERATIONS

The experimental results confirm that Q15 fixed-point representation provides sufficient numerical precision across all evaluated filter orders, with quantization errors consistently constrained below 0.002%. On ESP8266, the availability of a 32-bit accumulator plays a critical role in preserving numerical integrity during convolution, safely accommodating the summation of up to 52 products derived from 16-bit operands. This extended accumulation width effectively mitigates the risk of overflow without necessitating intermediate scaling or saturation logic. Consequently, the fixed-point formulation achieves a favorable balance between computational efficiency and numerical robustness, making it well suited for high-order FIR implementations on resource-constrained embedded platforms.

IV.17 POWER CONSUMPTION TRADE-OFFS

Although a full power characterization was beyond the scope of this work, current measurements at the 51st order configuration provide meaningful comparative insight. The ESP8266 consumed approximately 277 mW (84 mA at 3.3 V), while the ATmega328P exhibited a similar power draw of about 275 mW (55 mA at 5 V). Despite near-identical absolute power consumption, the ESP8266 delivers substantially higher computational throughput, translating to an estimated 1.9 $\times$ improvement in processing efficiency per watt. This advantage, combined with the ESP8266's ultra-low deep-sleep current capability, enables effective duty-cycling strategies for energy-constrained embedded DSP deployments.

IV.18 PRACTICAL APPLICATION GUIDELINES

Table 7 classification distills the empirical performance trends into practical deployment guidance. The ESP8266 is better aligned with computation-intensive, connected DSP workloads, whereas the ATmega328P remains well suited to simpler, power-constrained, and cost-driven applications.

Table 7: Application Specific Recommendations.

Category	ESP8266 – Recommended Application Domains	ATmega328P – Suitable Application Domains
Signal Processing	Multi-channel biosignal processing in wearable health-monitoring systems	Simple sensor signal conditioning with low-order filtering (≤ 20 th order)
Measurement & Analysis	High-order anti-aliasing filters for precision instrumentation	Educational demonstrations and rapid prototyping environments
Industrial Monitoring	Vibration analysis for predictive maintenance and fault detection	Integration with legacy systems requiring native 5 V logic levels
Audio & Sensing	Voice activity detection and real-time audio signal processing	Ultra-low-power deployments leveraging deep sleep currents ($\sim 0.1 \mu\text{A}$)
Networked Systems	Seismic monitoring and distributed sensing with cloud connectivity	High-volume, cost-sensitive production where minimal resources suffice

Source: Authors, (2026).

IV.19 DECISION FRAMEWORK:

The decision framework summarized in Table 8 provides a concise mapping between application requirements and platform suitability. For modest filter orders (≤ 25), both platforms offer acceptable performance; however, as computational and memory demands increase, the ESP8266 becomes increasingly advantageous and ultimately indispensable beyond the 55th order. Platform selection is further influenced by system-level constraints, with the ESP8266 favoured for memory-intensive or network-enabled applications, while the ATmega328P remains preferable where native 5 V interfacing or ultra-low sleep currents are critical.

Table 8: Application Requirements and Suggested Platform.

Requirement	Threshold	Recommended Platform
Filter Order	≤ 25	Either platform acceptable
Filter Order	26-55	ESP8266 strongly preferred
Filter Order	> 55	ESP8266 required
Memory Headroom	$> 50\%$ required	ESP8266
WiFi Connectivity	Required	ESP8266
5V Logic	Required	ATmega328P
Ultra-low Sleep	$< 1 \mu\text{A}$ required	ATmega328P

Source: Authors, (2026).

IV.20 LIMITATIONS AND THREATS TO VALIDITY - INTERNAL VALIDITY

Internal validity considerations primarily relate to measurement overhead and environmental influences on system behavior. Although performance metrics were obtained using embedded instrumentation, the associated code was carefully optimized and verified to contribute less than 2% additional CPU load, thereby minimizing perturbation of the measured workloads. Potential thermal effects were addressed by conducting experiments in a controlled laboratory environment maintained at $22 \pm 2 \text{ }^{\circ}\text{C}$. Furthermore, a thermal stabilization interval was enforced before data collection to ensure steady-state operating conditions.

IV.21 EXTERNAL VALIDITY

External validity of the reported findings is bounded by several contextual considerations. The experimental results pertain specifically to the ESP-12E module and the ATmega328P; alternative ESP8266 variants or other Arduino-class boards may exhibit different performance profiles due to variations in clocking, memory configuration, or peripheral integration. In addition, the analysis focuses exclusively on FIR Butterworth designs, and filter families with different approximation characteristics—such as Chebyshev or elliptic responses—or IIR implementations may lead to distinct computational and numerical behaviors. The evaluation was further constrained to a single-channel signal path operating at a 2 kHz sampling rate. Consequently, scenarios involving multi-channel processing or substantially higher sampling frequencies may introduce additional bottlenecks that alter the observed performance trends.

IV.22 CONSTRUCT VALIDITY

Construct validity is shaped by the choice of performance indicators used to represent overall system capability. In this study, CPU utilization and memory consumption were adopted as primary operational measures, capturing computational efficiency and resource constraints.

However, these metrics do not fully encompass other relevant dimensions, such as energy efficiency, development complexity, or long-term code maintainability. Inclusion of such complementary measures could yield a more holistic assessment of platform suitability across diverse application contexts.

IV.23 IMPLICATIONS FOR IOT SYSTEM DESIGN - EDGE COMPUTING PARADIGM

The experimentally validated signal-processing capability of the ESP8266 facilitates the adoption of genuine edge-computing architectures, in which substantial data preprocessing is performed directly at the device level rather than being offloaded to centralized resources, reducing:

- Cloud bandwidth requirements (70-90% reduction through local feature extraction)
- System latency (elimination of 200+ ms round-trip communication delay)
- Privacy concerns (sensitive data processed locally)
- Connectivity dependencies (offline operation capability)

IV.24 SYSTEM ARCHITECTURE EVOLUTION

Performance characteristics suggest transition from traditional sense-transmit-process architecture to intelligent edge nodes performing:

- Multi-stage filtering and signal conditioning
- Feature extraction and dimensionality reduction
- Anomaly detection and threshold monitoring
- Local decision-making with cloud
- Synchronization

V. CONCLUSIONS

This study presents a rigorous, empirical comparison of FIR Butterworth filter implementations on an 8-bit ATmega328P and a 32-bit ESP8266 (ESP-12E), demonstrating clear architectural advantages of the latter for computationally intensive DSP workloads. Across filter orders ranging from 13 to 51, the ESP8266 consistently sustained CPU utilization within a narrow band of 34–36%, whereas the ATmega328P exhibited a pronounced and statistically significant escalation from 28.6% to 68% ($t = 18.42$, $p < 0.001$ at the 51st order), indicating imminent computational saturation. Memory analysis further reinforces this disparity: the ESP8266 shows predictable linear growth from 11% to 33% utilization while retaining 54,886 bytes of free memory at the highest tested order.

By contrast, the ATmega328P shows a markedly steeper growth trajectory, with utilization climbing from 32% to 76%, thereby approaching the practical limits of its constrained SRAM. Collectively, these results confirm that the ESP8266 offers substantially greater computational headroom, scalability, and robustness for fixed-point FIR filtering. Extrapolation of the measured CPU trends indicates that FIR filter orders up to approximately 380 can be safely accommodated on the ESP-12E while remaining below an 85% CPU utilization threshold, thereby preserving real-time determinism. The findings position the ESP8266 as a compelling platform for high-order, real-time DSP tasks in IoT and edge-computing contexts, while delineating the practical limits of 8-bit microcontrollers for such applications.

VI. AUTHOR'S CONTRIBUTION

Conceptualization: Smit Lekhadia, Alpesh M. Patel, Kirit V. Patel, and Ghanshyamkumar Sah.

Methodology: Shantanu Joshi, Anilkumar J. Kshatriya, and Kirit V. Patel

Investigation: Smit Lekhadia, Alpesh M. Patel, , and Shantanu Joshi.

Discussion of results: Anilkumar J. Kshatriya, Alpesh M. Patel, Kirit V. Patel, and Ghanshyamkumar Sah.

Writing – Original Draft: Alpesh M. Patel, Ghanshyamkumar Sah, and Anilkumar J. Kshatriya.

Writing – Review and Editing: Alpesh M. Patel, Anilkumar J. Kshatriya, and Ghanshyamkumar Sah.

Resources: Kirit V. Patel, Alpesh M. Patel, and Smit Lekhadia.

Supervision: Smit Lekhadia, Shantanu Joshi, Ghanshyamkumar Sah, and Anilkumar J. Kshatriya.

Approval of the final text: Alpesh M. Patel, Smit Lekhadia, and Shantanu Joshi.

VII. REFERENCES

- [1] S. S. Saha, S. S. Sandha, and M. Srivastava, "Machine learning for microcontroller-class hardware: A review," *IEEE Sensors Journal*, vol. 22, no. 22, pp. 21362–21390, 2022.
- [2] L. Kong, J. Tan, J. Huang, G. Chen, S. Wang, X. Jin, P. Zeng, M. K. Khan, and S. K. Das, "Edge-computing-driven Internet of Things: A survey," *ACM Computing Surveys*, vol. 55, no. 8, Art. no. 174, 2022.
- [3] A. Garcia-Perez, R. Miñón, A. I. Torre-Bastida, and E. Zulueta-Guerrero, "Analysing edge computing devices for the deployment of embedded AI," *Sensors*, vol. 23, no. 23, Art. no. 9495, 2023.
- [4] N. N. Alajlan and D. M. Ibrahim, "TinyML: Enabling inference deep learning models on ultra-low-power IoT edge devices for AI applications," *Micromachines*, vol. 13, no. 6, Art. no. 851, 2022.
- [5] D. Chowdhury and M. Chattopadhyay, "Development of a low-power microcontroller-based wrist-worn device with resource-constrained activity detection algorithm," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 10, pp. 7522–7529, 2020.
- [6] F. Restuccia, M. Pagani, A. Mascitti, M. Marinoni, and G. Buttazzo, "ARTE: Providing real-time multitasking to Arduino," *Journal of Systems and Software*, vol. 186, Art. no. 111185, 2022.

- [7] L. Santos, T. Costa, J. M. L. P. Caldeira, and V. N. G. J. Soares, "Performance assessment of ESP8266 wireless mesh networks," *Information*, vol. 13, no. 5, Art. no. 210, 2022.
- [8] M. A. Landa-Jiménez, P. González-Gaspar, F. M. Montes-González, C. Morgado-Valle, and L. Beltrán-Parrazal, "An open-source low-cost wireless sensor system for acquisition of human movement data," *Anais da Academia Brasileira de Ciências*, vol. 94, no. 1, Art. no. e20191419, 2022.
- [9] L. Jiang, H. Zhang, S. Cheng, H. Lv, and P. Li, "An overview of FIR filter design in future multicarrier communication systems," *Electronics*, vol. 9, no. 4, Art. no. 599, 2020.
- [10] B. Maruthi Shankar, M. Ramkumar, and V. Saravanan, "Design of low power area efficient 2D FIR filter using optimized multiplier and adder for speech signal analysis," *Circuits, Systems, and Signal Processing*, vol. 44, no. 2, pp. 1138–1163, 2025.
- [11] P. D. Rosero-Montalvo, V. F. López-Batista, and D. H. Peluffo-Ordóñez, "A new data-preprocessing-related taxonomy of sensors for IoT applications," *Information*, vol. 13, no. 5, Art. no. 241, 2022.
- [12] O. Zubkov, I. Svyd, and O. Vorgul, "Features of the digital filters implementation on STM32 microcontrollers," in *Proc. III Int. Scientific-Practical Conf. Theoretical and Applied Aspects of Device Development on Microcontrollers and FPGAs*, 2021.
- [13] S. S. Saha, S. S. Sandha, and M. Srivastava, "Machine learning for microcontroller-class hardware: A review," *IEEE Sensors Journal*, vol. 22, no. 22, pp. 21362–21390, 2022.
- [14] H. K. Kondaveeti, N. K. Kumaravelu, S. D. Vanambathina, S. E. Mathe, and S. Vappangi, "A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations," *Computer Science Review*, vol. 40, Art. no. 100364, 2021.
- [15] M. Çelebi, "Digital filter design based on Arduino and its applications," in *Proc. 2020 Medical Technologies Congress (TIPTEKNO)*, 2020, pp. 1–4.
- [16] S. Banerjee, B. Bhattacharyya, and S. Munshi, "A microcontroller-based FIR filter with dynamic stabilization of sampling frequency," in *Proc. Int. Conf. on Computing, Electrical and Communication Engineering (ICCECE)*, 2023, pp. 1–6.
- [17] G. Sah, S. J. Lekhadia, J. Kedia, A. M. Patel, K. V. Patel, A. J. Kshatriya, C. Dave, and R. D. Mehta, "Benchmarking finite order filter algorithms on low power embedded processors: Computing efficiency," *International Journal of Environmental Science*, vol. 11, no. 6, pp. 2711–2724, 2025.
- [18] L. Santos, T. Costa, J. M. L. P. Caldeira, and V. N. G. J. Soares, "Performance assessment of ESP8266 wireless mesh networks," *Information*, vol. 13, no. 5, Art. no. 210, 2022.
- [19] W.-T. Sung, I. G. T. Isa, and S.-J. Hsiao, "An IoT-based aquaculture monitoring system using Firebase," *Computers, Materials & Continua*, vol. 76, no. 2, pp. 2179–2200, 2023.
- [20] J. Rawat, I. Kumar, N. Mohd, K. K. S. Rana, N. Pathak, and R. K. Gupta, "IoT-based home automation system using ESP8266," in *Proc. Int. Conf. on Innovative Computing and Communications*, Springer, 2023, pp. 695–707.
- [21] M. A. M. Moon, M. Kaimujjaman, M. M. Hossain, M. M. Islam, and M. S. Hossain, "Seamless real-time thermal imaging system with ESP8266: Wireless data transfer and display using UDP," *SN Applied Sciences*, vol. 5, no. 11, Art. no. 129, 2023.
- [22] H. Ye, C. Wu, and T. Yoshinaga, "A comparative study between FPGA and DSP for the efficient implementation of FIR filters," in *Proc. 12th Int. Symp. on Computing and Networking Workshops (CANDARW)*, 2024, pp. 90–95.
- [23] A. Elsts and R. McConville, "Are microcontrollers ready for deep learning-based human activity recognition?" *Electronics*, vol. 10, no. 21, Art. no. 2640, 2021.
- [24] O. Diouri, A. Gaga, H. Ouanan, S. Senhaji, S. Faquir, and M. O. Jamil, "Comparison study of hardware architectures performance between FPGA and DSP processors for implementing digital signal processing algorithms: Application of FIR digital filter," *Results in Engineering*, vol. 16, Art. no. 100639, 2022.
- [25] A. Przybył, "Fixed-point arithmetic unit with a scaling mechanism for FPGA-based embedded systems," *Electronics*, vol. 10, no. 10, Art. no. 1164, 2021.
- [26] S. Jayashree, S. Shantha Preetha, and S. M. Sakthivel, "FPGA implementation of optimized FIR filter for ECG denoising using Peres reversible logic gate," *Journal of Theoretical and Applied Information Technology*, vol. 100, no. 15, pp. 4582–4594, 2022.
- [27] J. Goldsmith, L. H. Crockett, and R. W. Stewart, "A natively fixed-point run-time reconfigurable FIR filter design method for FPGA hardware," *IEEE Open Journal of Circuits and Systems*, vol. 3, pp. 25–37, 2022.
- [28] S. Simona-Daiana, Z. Lendek, and P. Dobra, "Analysis of the quantization effects in the implementation of numerical filters," in *Proc. 27th Int. Conf. on System Theory, Control and Computing (ICSTCC)*, 2023, pp. 227–232.
- [29] J. C. R. Kumar, D. V. Kumar, and M. A. Majid, "High-performance, energy-efficient, and memory-efficient FIR filter architecture utilizing 8×8 approximate multipliers for wireless sensor networks in the Internet of Things," *Memories*, vol. 3, Art. no. 100017, 2022.
- [30] Y. Almudayni and A. M. Hassen, "Energy inefficiency in IoT networks: Causes, impact, and a path forward," *Electronics*, vol. 14, no. 1, Art. no. 159, 2025.
- [31] A. Rai, C. R. Nayak, and S. K. Gupta, "An optimization of low-power 4-bit PAL FIR filter using adiabatic techniques," *Sādhanā*, vol. 48, Art. no. 84, 2023.
- [32] G. K. Kumar, N. R. Chinnapurapu, and K. Srinivas, "Power and area efficient FIR filter using radix-2r multiplier for de-noise the electrooculography (EOG) signal," *Scientific Reports*, vol. 14, Art. no. 22599, 2024.