International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 6, 2025
https://www.theaspd.com/ijes.php

Benchmarking Finite Order Filter Algorithms On Low Power Embedded Processors : Computing Efficiency

Ghanshyamkumar Sah[1], Smit Lekhadia[2], Jash Kedia[3], Alpesh M. Patel[4], Kirit V. Patel[5], Anilkumar J. Kshatriya[6] , Chintan Dave[7], Rahul D. Mehta[8]

[1] Electronics and Communication, L. D. College of Engineering, Ahmedabad, Gujarat, India.
[2,3,7] Electronics and Communication, Vishwakarma Government. Engineering College, Ahmedabad, Gujarat, India.
[4,5, 6,8] Electronics and Communication, Government Engineering College, Gandhinagar, Gujarat, India.

[1]Corresponding Author: ghanshyam@ldce.ac.in

Abstract - *Efficient signal processing in embedded systems hinges on selecting the right microcontroller for your Finite Impulse Response (FIR) filter needs, yet engineers often struggle without clear guidelines for making this crucial decision. Our research tackles this practical challenge by systematically studying how different filter complexities affect microcontroller performance across key areas like processing speed, memory usage, CPU load, and power consumption. We discovered that complex, high-order filters need powerful microcontrollers with robust processing capabilities and ample memory, while simpler, low-order filters work perfectly well on energy-efficient, low- power chips that prioritize battery life over raw performance. The results indicate that a 51st-order filter utilizes nearly 76% of the available 2 KB SRAM, beyond which the system exhibits frequent instability stemming from insufficient memory allocation for reliable runtime operations. These findings give embedded system designers a straightforward framework for choosing microcontrollers that strike the right balance between getting the job done and conserving power. By understanding these relationships, engineers can make smarter hardware decisions that deliver effective signal processing while keeping systems efficient and sustainable, ultimately leading to better-performing embedded devices that don't waste precious resources.*

Keywords - *Implementation of FIR Filters, Microcontrollers, Performance Monitoring, Filter Packages, MATLAB, Signal Processing.*

1. INTRODUCTION

Digital filters constitute a foundational element in contemporary signal processing frameworks, underpinning a broad spectrum of applications including, but not limited to, telecommunications, biomedical instrumentation, high-fidelity audio systems, and real-time embedded control environments [1-4]. These filters are predominantly categorized into Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) types, a distinction rooted in the temporal extent of their respective impulse responses. FIR filters, characterized by their inherently finite-duration responses, are renowned for their intrinsic stability and ability to achieve an exact linear phase response [5]. These attributes render FIR filters particularly advantageous in applications demanding stringent phase linearity and low distortion, such as precise sensor signal conditioning, medical diagnostics, and studio-grade audio processing, as discussed by M. Akay [6]. Moreover, the absence of feedback paths in FIR structures simplifies their design and verification processes, as highlighted by E. Ifeachor and B. W. Jervis [7], which is especially valuable in safety-critical systems.

Conversely, IIR filters, which derive their response from recursive formulations involving feedback elements, can achieve steeper frequency roll-offs with significantly reduced filter order, as demonstrated by R. Lyons [8]. This property translates to lower memory requirements and computational efficiency—an

essential consideration in resource-constrained platforms like low-power embedded systems. However, these benefits come with trade-offs: the potential for instability due to feedback, sensitivity to quantization effects in fixed-point arithmetic implementations, and the inherent nonlinearity in phase response, all of which have been discussed in detail by P. K. Meher et al. [9].

This investigation centers on the design and implementation of Finite Impulse Response (FIR) Butterworth filters—an FIR variant distinguished by its maximally flat passband and gradual transition characteristics in the frequency domain, as outlined by A. G. Constantinides [10]. Unlike other FIR filter designs that may exhibit ripples in the passband or stopband, the Butterworth profile ensures a smooth frequency response, particularly advantageous in applications where spectral uniformity and minimal signal distortion are imperative, as detailed by L. Rabiner et al. [11]. Furthermore, the linear phase property intrinsic to FIR filters ensures that all frequency components of a signal are delayed equally, preserving waveform integrity—a critical requirement in real-time control systems and biomedical monitoring, as emphasized by R. Mehboob et al. [12].

In this study, we target the deployment of these FIR Butterworth filters on resource-limited microcontroller plat- forms, specifically the Arduino family, exemplified by the ATmega328P [13]. These widely used microcontrollers, while accessible and cost-effective, impose stringent constraints on computational resources: a mere 2 KB of SRAM, 32 KB of flash memory, and a modest 16 MHz clock frequency, as outlined by J. Lin et al. [14]. Such limitations pose significant challenges for digital signal processing tasks, which often rely on floating-point arithmetic, high-order filter coefficients, and real-time data handling—requirements that can easily exceed the operational capacity of typical 8-bit microcontrollers, as demonstrated by W. Dyason et al. [15] and S. Chikhalikar et al. [16].

The novelty of this work lies in demonstrating the feasibility of executing computationally constrained FIR filtering on such minimal hardware by employing optimized algorithms, fixed-point arithmetic, and memory-efficient coding practices, as shown by A. Omondi et al. [17]. The implementation aims to maintain acceptable signal fidelity and phase characteristics while offering practical insights for real-world embedded applications, where balancing filter performance with hardware limitations is a persistent design dilemma A. Andreou et al. [18]; K. Thyagarajan et al. [19]. This research contributes to the broader discourse on lightweight digital signal processing techniques suitable for low-power, embedded environments, as discussed by R. Nagarajan et al. [20].

2. LITERATURE SURVEY

FIR Butterworth filters emerge as a particularly suitable choice for implementation on resource-constrained embedded platforms, as outlined by S. Kar et al. [21]. H. J. Ko et al. [22] highlighted that their inherent stability and deterministic computational complexity render them particularly well-suited for fixed-point arithmetic operations. This characteristic is critical in environments such as Arduino-based systems, where floating-point computation is either unsupported or prohibitively inefficient. D. Ibrahim [23] emphasized that, unlike IIR filters which frequently encounter numerical instability and quantization-induced degradation on low-precision hardware FIR filters provide robust and predictable behavior under a broad range of conditions.

This study endeavors to identify the optimal filter order that achieves a balance between signal processing efficacy and the hardware limitations of microcontrollers like the ATmega328P. Increasing the filter order generally enhances frequency selectivity, resulting in sharper transition bands and greater stopband attenuation, as noted by M. Mirghani [24]. However, these improvements come at the cost of increased

computational burden and memory consumption, which may exceed the available stack or processing capacity, as demonstrated by V. Varshney et al. [25]. Conversely, filters with lower orders demand fewer system resources but may fail to satisfy the application's spectral constraints.

M. Çelebi [26] observed that determining the appropriate filter order constitutes a multi-objective optimization problem, requiring careful evaluation of trade-offs among spectral performance, execution speed, and memory usage. This investigation systematically explores these trade-offs through empirical testing and performance benchmarking, aiming to establish design guidelines for implementing high-fidelity digital filters on microcontrollers constrained by clock speed, memory footprint, and energy consumption S. Pujari et al. [27].

For implementation, P. C. Joshi et al. [28] precomputed the filter coefficients corresponding to the desired FIR Butterworth response using established digital filter design methodologies, such as the windowing or frequency sampling techniques. These coefficients are then quantized and stored in the flash memory of the Arduino microcontroller to conserve the system's limited SRAM, which is critical for runtime operations. Parhi [29] emphasized that this approach ensures memory-intensive data structures do not occupy volatile memory, thereby preventing overflow and preserving headroom for buffering and control logic.

The filtering operation is performed in the time domain through direct convolution, a method more compatible with the stringent memory constraints and lack of native floating-point hardware on the ATmega328P platform, as analyzed by S. Simona-Daiana [30]. FFT-based filtering, while theoretically more efficient for high-order filters, is impractical on such constrained systems due to the overhead of complex data structures and real-time memory demands, as noted by S. Simona-Daiana et al. [31].

To maintain computational efficiency, the signal processing pipeline is implemented using 16-bit fixed-point arithmetic, as proposed by F. Liang et al. [32]. This choice balances numerical precision and execution speed. Special attention is devoted to the quantization of filter coefficients: they are scaled and rounded in a manner that minimizes quantization noise while preserving essential frequency-domain characteristics, including passband flatness and stopband attenuation, as addressed by J. Goldsmith et al. [33]. The quantization strategy is validated through both simulation and empirical testing to ensure fidelity to the theoretical design, as demonstrated by X. X. Zheng et al. [34].

This implementation framework illustrates how sophisticated filtering algorithms can be adapted to operate reliably within the severe resource constraints of microcontroller-based platforms, as discussed by B. K. Mohanty et al. [35]. G. Ganjikunta et al. [36] described how managing memory, optimizing arithmetic, and selecting appropriate structures enable the design to achieve high-performance signal conditioning suitable for real-time embedded applications.

The implemented FIR Butterworth filters were subjected to comprehensive experimental evaluation on the Arduino platform, with a focus on three critical dimensions: numerical stability, power efficiency, and signal filtering performance, as discussed by J. Vandenbussche et al. [37]. Although FIR filters are theoretically unconditionally stable due to the absence of feedback loops, empirical validation through pole-zero analysis ensured that fixed-point arithmetic did not introduce computational anomalies. A. M. Krukowski et al. [38] also evaluated the behaviour of the filter under various coefficient quantization schemes to identify rounding-induced distortions.

Power efficiency was evaluated by monitoring the microcontroller's current consumption during real-time

filtering operations. N. Sādhanā et al. [39] established the correlation between filter order and energy demand through measurements under varying filter complexities. These insights are essential for battery-powered applications, where operational longevity depends on minimizing power draw without sacrificing fidelity, as emphasized by T. Ingolfsson [40]. G. Kumar et al. [41] further demonstrated that while higher-order filters offer sharper spectral discrimination, they significantly increase energy consumption.

Filtering performance was assessed by passing both synthetic inputs—such as sinusoidal signals—and real-world data streams from MEMS-based accelerometers through the implemented filters. Outputs were analyzed in time and frequency domains and benchmarked against ideal Butterworth characteristics, as examined by U. Güner et al. [42]. Metrics included passband ripple, stopband attenuation, and transition bandwidth. The implemented filters achieved the desired behavior within the constraints imposed by quantization and finite arithmetic. These empirical assessments confirm the viability of deploying FIR Butterworth filters on constrained platforms such as Arduino, provided that design choices are optimized for precision, energy, and computational trade-offs. The results contribute benchmarks and design considerations for signal processing tasks in low-power embedded applications.

This research presents a structured and empirically vali- dated methodology for selecting optimal FIR Butterworth filter orders tailored to resource-constrained embedded platforms, exemplified by the Arduino microcontroller family. The study offers a framework that enables deployment of high-fidelity digital filters in environments with limited power and memory resources. The approach is relevant to IoT devices, wearable health monitoring systems, and autonomous sensor nodes.

The proposed approach not only addresses practical challenges of implementing real-time signal processing on low-end microcontrollers using fixed-point arithmetic but also provides scalable design principles applicable to broader FIR filter architectures. While this work focuses on the Butterworth design due to its flat frequency response and stability, the techniques and insights can be readily adapted to alternative FIR formulations, such as Chebyshev or equiripple filters.

Furthermore, the framework established herein serves as a foundation for future exploration into adaptive filtering, multirate processing, and dynamic reconfiguration in embedded systems, particularly as emerging microcontrollers begin to of- fer enhanced digital signal processing capabilities. This study contributes to the growing body of knowledge that bridges advanced signal processing theory and practical realization in low-power, embedded environments, enabling more intelligent and autonomous edge-computing solutions.

3. IMPLEMENTATION

3.1 Selection of Embedded Processor

The Arduino UNO shown in Fig. 1 was selected as the target embedded processing platform for this investigation following a rigorous comparative assessment of contemporary microcontroller-based systems in terms of computational performance, memory architecture, development accessibility, and cost-efficiency. Among the wide array of available embedded platforms, the Arduino UNO featuring the Atmel ATmega328P microcontroller emerged as a practical and pedagogically valuable choice due to its optimal balance between simplicity, functionality, and widespread adoption in academic and prototyping contexts, as demonstrated by P. García-Tudela [43].

Operating at a clock frequency of 16 MHz, the ATmega328P offers 32 kilobytes of flash memory and 2 kilobytes of SRAM, resources that, while modest by modern standards, are adequate for implementing lightweight digital signal pro- cessing algorithms when optimized carefully. Its architecture supports efficient execution of fixed-point arithmetic, which is crucial for the resource-constrained deployment of FIR filter structures, especially in real-time applications.

Fig. 1 Arduino UNO

The selection of the Arduino UNO was further motivated by its exceptional accessibility. The platform is supported by a large global user base, extensive open-source libraries, and a comprehensive repository of community-generated documentation and instructional content. This ecosystem significantly reduces development time and lowers the entry barrier for rapid prototyping and academic experimentation. In particular, the availability of well-maintained libraries for digital I/O, ADC interfaces, and timing control facilitated seamless integration of signal acquisition and filtering routines within the constraints of the hardware, as highlighted by N. Prabowo et al. [44].

Using the Arduino platform's robust ecosystem and proven track record, FIR Butterworth filter packages originally developed and validated in MATLAB were successfully implemented and tested for real-time execution. Consequently, the Arduino UNO served not only as a cost-effective testbed for experimental validation but also as a representative model for evaluating the viability of real world signal processing applications on ultra-low-power microcontrollers.

3.2 Creating Filter Packages Using MATLAB

The formulation of reliable digital signal processing (DSP) solutions necessitates a methodical design methodology that integrates theoretical precision with practical feasibility. In pursuit of this objective, a comprehensive library of FIR Butterworth filters was systematically developed using MAT- LAB, a high-level computational environment well-suited for prototyping and analysis of DSP algorithms. The filter suite encompasses orders ranging from 1 to 101 and is configured to operate within a frequency span of 100 Hz to 1000 Hz, all synthesized under a uniform sampling frequency of 2000 Hz. This choice ensures adherence to the Nyquist criterion while providing sufficient resolution for accurate spectral shaping across a broad range of applications.

The development process adhered to a structured four-phase workflow, designed to maintain fidelity between theoretical expectations and embedded implementation outcomes. The initial phase involved the definition of key design specifications, including the cut-off frequency (550 Hz), the passband edge frequency (450 Hz), and the global sampling frequency (2000 Hz). These parameters were selected based on

the requirements of typical embedded signal acquisition systems, such as those employed in environmental monitoring or biomedical instrumentation. Additionally, decisions regarding the approximation method (Butterworth) and windowing technique were made to control side-lobe behaviour and optimize transition bandwidth.

In the subsequent design phase, MATLAB's Filter Design and Analysis (FDA) Tool was employed to facilitate precise filter construction. This graphical interface supports intuitive entry of design parameters and offers in-depth analysis of magnitude and phase response characteristics. By enabling real-time visualization of the frequency response, the FDA tool allowed for iterative refinement of filter configurations to meet performance targets related to passband flatness, stopband attenuation, and group delay linearity.

The resulting filter coefficients were exported with high numerical resolution and served as a reference for fixed-point quantization strategies implemented later in the embedded phase. Furthermore, the MATLAB generated models served as a simulation baseline, enabling comparative analysis against real-time performance on constrained hardware platforms. Overall, this MATLAB based filter design framework provided a rigorous and flexible foundation for developing and validating FIR Butterworth filters suitable for deployment in resource-limited embedded systems.

The implementation phase entails the seamless integration of precomputed FIR Butterworth filter coefficients into microcontroller-level scripts, followed by the deployment of efficient digital filtering algorithms directly applied to real- time analog-to-digital converted (ADC) data streams. Given the limited computational and memory resources of embedded platforms such as the Arduino UNO, particular emphasis is placed on optimizing the codebase for fixed-point arithmetic and minimizing execution latency. Algorithmic refinements including loop unrolling, memory-efficient buffer management, and conditional simplification are employed to ensure that filtering operations can be performed within strict timing constraints imposed by real-time systems. Subsequent to code optimization, the filtering routines are subjected to comprehensive empirical validation. This validation process encompasses the deployment of compiled firmware onto the target hardware, real-time signal monitoring via serial interfaces, and the quantitative evaluation of filter behaviour under operational conditions, as demonstrated by Y. S. Shmaliy et al. [45].

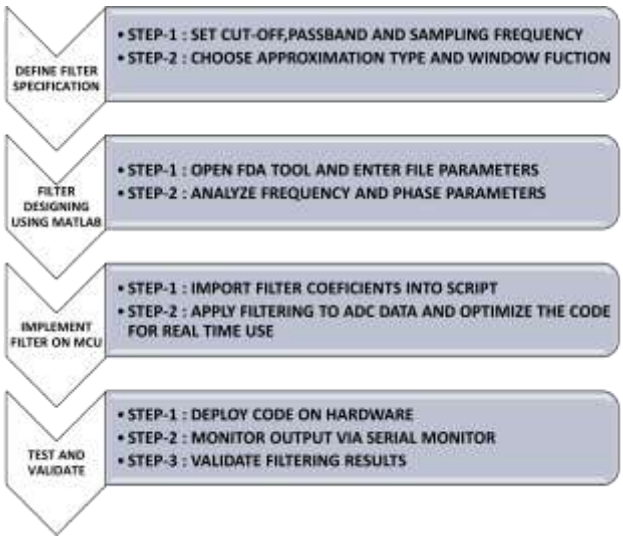

Fig. 2 Flowchart to Create a Digital Filter Using MATLAB Tool

Output signals are analyzed to verify adherence to the design specifications, particularly in terms of frequency selectivity, passband integrity, and attenuation in the stopband. This iterative validation ensures that the implemented filters not only replicate theoretical models but also perform reliably under physical hardware constraints and real-world signal conditions.

The complete end-to-end workflow from initial specification through filter design, embedded implementation, and final validation—is systematically illustrated in Fig. 2. The diagram outlines the four major stages of the development pro- cess: (1) defining filter specifications, (2) designing the filter using MATLAB's Filter Design and Analysis (FDA) Tool, (3) implementing the filter on the microcontroller, and (4) testing and validating results through hardware-based measurements. This structured approach bridges simulation and deployment, facilitating reproducible, high-fidelity digital signal processing in constrained embedded environments.

3.3 Implementing the set of filters on Arduino UNO

To evaluate the processing capacity and energy efficiency of the Arduino UNO platform, a diverse set of FIR Butter- worth filters was designed using MATLAB and subsequently deployed for experimental testing. Filters of orders 13, 15, 17, 21, 26, 34, 51, 53, 61, 62, 63, and 101 were selected to span a wide spectrum of computational complexity and memory demand. Each filter was independently compiled and uploaded to the ATmega328P microcontroller to assess its operational feasibility under constrained resources. This systematic variation in filter order provided critical insights into the scalability of digital signal processing on low-power hardware.

Performance metrics were recorded with respect to both signal processing fidelity and power consumption. As filter order increased, the number of multiplications and additions per sample grew proportionally, imposing greater computational load and memory usage. These conditions allowed for an empirical investigation of the trade-off between spectral selectivity and resource utilization. The outcomes informed thresholds beyond which the Arduino UNO may become unsuitable for real-time filtering tasks, thereby contributing to practical design guidelines for embedded system engineers.

A precision-controlled experimental environment was established to evaluate the real-time filtering performance of the Arduino UNO with high repeatability and accuracy. The setup consisted of three essential components, each selected to ensure consistent and quantifiable measurement across all test conditions:

A regulated 5V DC power source was employed to ensure consistent voltage delivery to the Arduino UNO throughout all experimental runs. Maintaining a stable power input is essential not only for ensuring reproducibility but also for accurate assessment of current consumption, which directly correlates to the system's energy efficiency under varying computational loads. A serial communication interface operating at 9600 baud was used to monitor the microcontroller's real-time response.

The analyzer logged filter output data with time-stamped precision, facilitating the examination of response latency, numerical stability, and signal integrity. The output stream was further utilized to generate visual representations—such as time-domain and frequency-domain plots—for comparing the filter's empirical behaviour against theoretical expectations.

A calibrated function generator delivered a continuous sine wave input signal, set to 1 kHz frequency and 1V peak-to-peak amplitude, to the analog input pins (A0 and A1) of the Arduino UNO. This analog stimulus allowed for the verification of the microcontroller's analog-to-digital conversion performance and the filter's ability to process periodic waveforms without introducing artifacts or phase distortion.

This integrated experimental framework enabled a robust validation of each filter's implementation, highlighting the boundaries of real-time signal processing on a constrained embedded platform. The methodology not only supports re- producibility but also offers a foundation for future comparative studies involving alternative filter structures or hardware configurations.

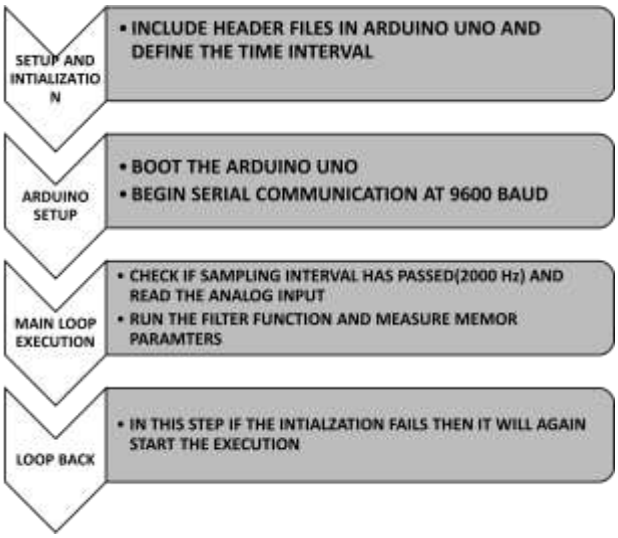

Fig. 3 Procedure for Testing Filter Packages on Arduino UNO

To ensure the reliability and accuracy of signal acquisition during real-time testing, high-integrity connectors and shielding techniques were employed to minimize electromagnetic interference and signal degradation. These hardware provisions were critical for maintaining signal integrity from the function generator to the analog input stage of the microcontroller, thereby enabling high-fidelity filtering and precise analog-to-digital conversion. Such considerations are particularly important in embedded signal processing applications, where noise artifacts can propagate through the system and compromise filter performance evaluations.

The carefully constructed testbench facilitated the granular assessment of key operational metrics, including signal fidelity, computational latency, and overall system efficiency. These parameters were observed and logged under consistent and repeatable experimental conditions, ensuring that performance trends could be directly attributed to variations in filter order or algorithmic configuration rather than external perturbations.

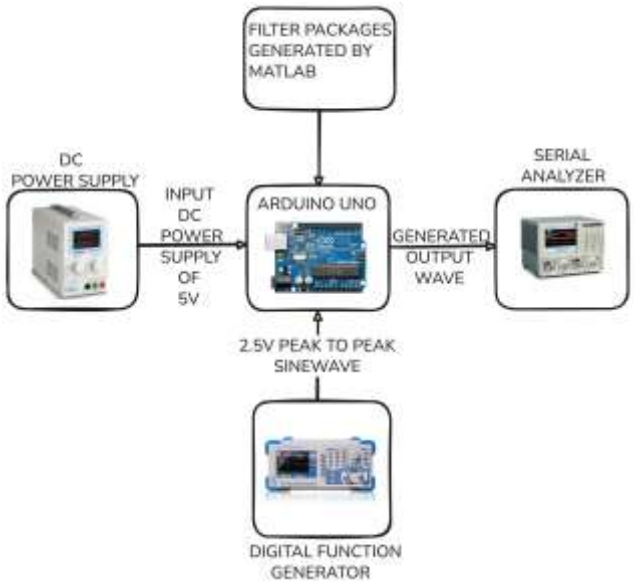

Fig. 4 Setup For Testing

The complete operational workflow implemented on the Arduino UNO—from system initialization and input acquisition to real-time digital filtering and data logging is systematically illustrated in Fig. 3. The diagram delineates each phase of the process, beginning with the inclusion of necessary header files and the definition of timing intervals, followed by power-up and serial communication initialization. The main execution loop monitors the sampling interval, reads ADC inputs, and executes the filtering routine while tracking memory usage. A fail-safe loopback mechanism ensures robust operation by reinitializing the system in case of configuration errors or communication faults. This structured flow guaran- tees a stable and efficient testing environment for real-time filter validation on constrained embedded platforms. The whole setup which is used to test various filter packages on Arduino UNO has been showcased in Fig.4

4. RESULTS AND DISCUSSION

The empirical investigation of Butterworth FIR filter implementations on the Arduino UNO microcontroller yields critical insights into the feasibility and performance boundaries of digital signal processing within constrained embedded environments. A total of 15 distinct filter configurations were rigorously tested over a frequency spectrum ranging from 100 Hz to 1000 Hz. For each configuration, we conducted a detailed assessment of computational overhead, memory utilization, and real-time processing capability.

This systematic analysis revealed the inherent trade-offs between filter complexity and system performance, highlighting the specific constraints posed by the limited computational power and memory architecture of the AVR-based Arduino UNO platform. The findings underscore the importance of aligning filter design specifications with the operational characteristics of the target microcontroller to ensure reliable and efficient execution.

Moreover, the study provides a robust empirical basis to guide engineers and system designers in the judicious selection of embedded processing units tailored for digital filtering ap- plications. Such data-driven selection is particularly pertinent in scenarios where power efficiency, hardware footprint, and cost constraints are critical design considerations. Ultimately, this work contributes to the broader discourse on

optimizing algorithmic deployment in embedded signal processing, offering a template for evaluating other microcontroller architectures under similar constraints.

The analysis of CPU utilization as a function of filter order reveals a pronounced and steadily escalating trend, clearly indicative of the computational burden introduced by increasing filter complexity. Commencing with a utilization of 28.6% for a 13th-order filter, the processing load exhibits a progressive increment, rising to 36% at the 15th order, 39.2% at the 17th order, and further to 43.2% by the 21st order. This ascending trajectory culminates at a substantial 68% CPU usage for the 51st-order configuration.

Such a quasi-linear correlation between filter order and processor load underscores the direct contribution of each additional filter tap to overall computational demand. This proportionality is of paramount importance in the context of real-time embedded systems; wherein deterministic timing behaviour and strict latency thresholds must be upheld. The findings emphasize that, in resource-constrained environments, even marginal increases in filter order can precipitate significant performance degradation, potentially compromising the temporal fidelity required in time-sensitive signal processing tasks. Consequently, these results advocate for a meticulous balance between filter design objectives and system-level processing capabilities, particularly in applications with stringent real-time constraints.

The dynamic memory utilization exhibits a parallel trend to that of CPU load, displaying a marked increase as filter order escalates. Specifically, memory consumption rises from 32% at the 13th order configuration to a significant 76% at the 51st-order implementation. This substantial allocation of memory resources can be attributed to the cumulative storage demands imposed by filter coefficients, temporary computational variables, and input/output data buffers each of which is integral to the stable and accurate functioning of the finite impulse response (FIR) filtering process.

Notably, the rate of memory consumption accelerates dis- proportionately at higher filter orders, with utilization increas- ing from 47% at the 26th order to 76% at the 51st order. This steep gradient suggests a superlinear or near-exponential growth in memory requirements, driven by both the linear increase in coefficient storage and the compounded buffering needs for managing larger convolution windows.

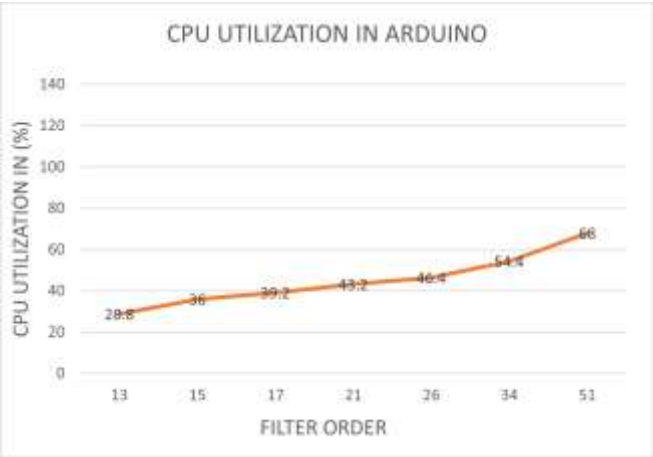

Fig. 5. CPU Utilization of Arduino UNO up till its Saturation Point

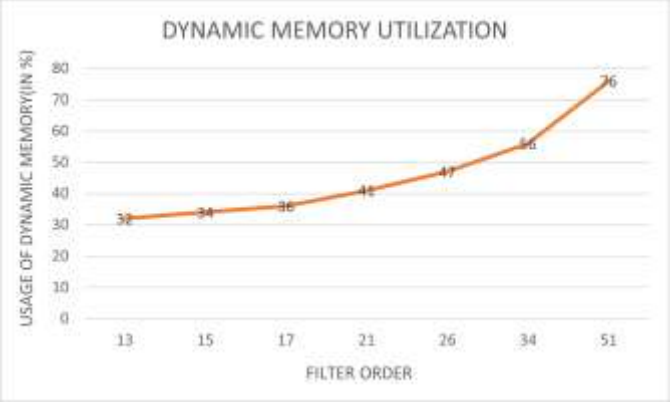

Fig. 6. Dynamic Memory Utilization of Arduino UNO up till its Saturation Point

Such rapid escalation in memory usage imposes critical constraints on platforms like the Arduino UNO, where SRAM is limited to 2 KB. Therefore, the deployment of higher-order filters must be approached with caution to avoid runtime in- stability or memory overflow. The interplay between memory saturation and real-time operability further emphasizes the necessity of optimizing filter order not only for frequency response characteristics but also for hardware-level resource constraints. The corresponding CPU utilization patterns associated with each filter configuration are illustrated in Fig. 5, providing a comprehensive view of the system's processing and memory behavior under varying computational loads.

These empirical findings serve as a critical reference point for engineers and system architects engaged in the development of Arduino-based digital signal processing (DSP) applications. By elucidating the quantitative relationship between filter order and system-level resource consumption specifically in terms of CPU load and dynamic memory usage this study facilitates informed decision-making in the design phase. Engineers can leverage this data to judiciously select filter parameters that align with the computational constraints and memory limitations inherent to the Arduino UNO platform, without compromising real-time performance or functional stability.

Furthermore, the analysis underscores the importance of evaluating the trade-offs between filter precision, execution efficiency, and hardware overhead in embedded DSP implementations. Higher-order filters, while offering enhanced frequency selectivity and noise attenuation, impose escalating demands on processing power and memory, thereby potentially exceeding the thresholds necessary for reliable operation within time-critical tasks.

Accordingly, this research advocates for a design method- ology that integrates algorithmic accuracy with hardware awareness, ensuring an optimal balance between signal fidelity and system sustainability. The detailed profile of dynamic memory utilization corresponding to each filter configuration is presented in Fig. 6, offering a visual representation of the growing memory footprint as filter complexity increases.

5. CONCLUSION

This research presents a detailed evaluation of Butterworth FIR filter implementations on the Arduino UNO, revealing the practical limitations of real-time digital signal processing on resource-constrained microcontrollers. The findings show that a 51st-order filter consumes approximately 76% of the 2 KB

SRAM, beyond which the system frequently becomes unstable due to inadequate memory for runtime operations.

To preserve system stability and ensure deterministic performance in real-time applications, it is imperative to maintain a safe margin in memory utilization. Empirical evidence from this study suggests that limiting dynamic memory usage to approximately 76% thereby reserving at least 20–25% of SRAM provides sufficient buffer space for CPU scheduling, stack operations, and intermediate calculations during filter execution.

A key contribution of this work is the identification of this empirical threshold, which serves as a practical design constraint for embedded system developers. By linking filter complexity to platform-specific resource usage, the study offers a transferable framework for evaluating the feasibility of DSP algorithms across similar microcontroller architectures. These findings are particularly relevant for low-power, real- time applications in IoT and embedded sensing systems, where design choices must balance filtering precision with computational efficiency and system robustness.

REFERENCES

[1] A. V. Oppenheim and R. W. Schafer, Discrete-Time Signal Processing, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2009.
[2] L. Cromwell, F. J. Weibell, and E. A. Pfeiffer, Biomedical Instrumentation and Measurements, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1980.
[3] K. C. Pohlmann, Principles of Digital Audio, 6th ed. New York, NY, USA: McGraw-Hill, 2010.
[4] J. G. Proakis and D. G. Manolakis, Digital Signal Processing: Principles, Algorithms, and Applications, 4th ed. Upper Saddle River, NJ, USA: Pearson, 2007.
[5] S. K. Mitra, Digital Signal Processing: A Computer-Based Approach, 4th ed. New York, NY, USA: McGraw-Hill, 2010.
[6] M. Akay, Biomedical Signal Processing, San Diego, CA, USA: Academic Press, 1994.
[7] E. Ifeachor and B. W. Jervis, Digital Signal Processing: A Practical Approach, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.
[8] R. Lyons, Understanding Digital Signal Processing, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2010.
[9] P. K. Meher, S. Chandrasekaran, and A. Amira, "FPGA realization of FIR filters by efficient and flexible systolization using distributed arithmetic," IEEE transactions on signal processing , vol. 56, no. 7, pp. 3009–3017, 2008.
[10] A. G. Constantinides, "Spectral transformations for digital filters," Proceedings of the Institution of Electrical Engineers. Vol. 117. No. 8. IEE, 1970.
[11] L. R. Rabiner and B. Gold, Theory and Application of Digital Signal Processing, Englewood Cliffs, NJ, USA: Prentice-Hall, 1975.
[12] R. Mehboob, S. A. Khan and R. Qamar, "FIR filter design methodology for hardware optimized implementation," in IEEE Transactions on Consumer Electronics, vol. 55, no. 3, pp. 1669-1673, 2009.
[13] Microchip Technology Inc., ATmega328P Datasheet, San Jose, CA, USA: Microchip, 2022.
[14] J. Lin, W.-M. Chen, Y. Lin, C. Gan, and S. Han, "MCUNet: Tiny deep learning on IoT devices," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 33, pp. 11711–11722, 2020.
[15] W. Dyason, T. I. van Niekerk, R. Phillips, and R. Stopforth, "Performance evaluation and comparison of filters for real-time embedded system applications," in Proc. Pattern Recognit. Assoc. South Africa and Robotics and Mechatronics (PRASA-RobMech), Bloemfontein, South Africa, pp. 242–248, 2017.
[16] S. Chikhalikar, O. Khandekar, and C. Bhattacharya, "Design of real-time acquisition and filtering for MEMS-based accelerometer data in microcontroller," in Proc. IEEE Electron Devices Kolkata Conf. (EDKCON), Kolkata, India, pp. 15–18. 2018.
[17] P. Gaydecki, Foundations of Digital Signal Processing: Theory, algorithms and hardware design, London, U.K.:

IET, 2004.
[18] A. G. Andreou and M. A. Bayoumi, Energy-Efficient Digital Signal Processing, 1st ed., R. M. Osgood, Jr., Ed. Boston, MA, USA: Kluwer Academic Publishers, 2001.
[19] K. S. Thyagarajan, A. E. Briseño, and A. A. Valencia, "Design of Butterworth and Chebyshev digital filters," Int. J. Electr. Eng. Educ., vol. 20, no. 1, pp. 59–70, 1983.
[20] R. Nagarajan, A. Martino, D. A. Morero, L. Patra, C. Lutkemeyer, and M. A. Castrillón, "Recent advances in low-power digital signal processing technologies for data center applications," J. Lightw. Technol., vol. 42, no. 12, pp. 4222–4232, 2024.
[21] S. Kar and M. Ganguly, "Study of effectiveness of FIR and IIR filters in exon identification: A comparative approach," Mater. Today Proc., vol. 58, pp. 437–444, 2022.
[22] H. J. Ko and J. J. P. Tsai, "Robust and computationally efficient digital IIR filter synthesis and stability analysis under finite precision implementations," IEEE Trans. Signal Process., vol. 68, pp. 1807–1822, 2020.
[23] D. Ibrahim, "Low-cost microcontroller-based hardware for introducing digital filter fundamentals to students," Int. J. Eng. Educ., vol. 23, no. 5, pp. 1000, 2007.
[24] M. Mirghani, "Implementation of matched filters using microcontrollers," in Proc. Conf. Basic Sci. Eng. Studies (SGCAC), Khartoum, Sudan, pp. 1–6. 2016.
[25] V. Varshney and M. Tiwari, "Realization of an FIR filter using ATMEGA32 microcontroller," in Proc. Int. Conf. Emerging Trends Comput. Commun. Technol. (ICETCCT), Dehradun, India, pp. 1–5, 2017.
[26] M. Çelebi, "Digital filter design based on Arduino and its applications," in Proc. 2020 Medical Technologies Congress (TIPTEKNO), Antalya, Turkey, pp. 1–4, 2020.
[27] S. Pujari, A. Yeotkar, V. Shingare, S. Momin and B. Kokare, "Performance analysis of microcontroller and FPGA based Signal Processing a case study on FIR filter design and implementation," International Conference on Industrial Instrumentation and Control (ICIC), Pune, India, pp. 252-257, 2015.
[28] P. C. Joshi and D. S. Khurge, "Comparative analysis of embedded computing platforms for FIR filter implementation," in Proc. IEEE Int. Conf. Comput. Commun. Control Autom. (ICCUBEA), Pune, India, pp. 1–4. 2016.
[29] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation, New York, NY, USA: Wiley, 1999.
[30] S. Simona-Daiana, Z. Lendek, and P. Dobra, "Analysis of the quantization effects in the implementation of numerical filters," in IEEE Proc. 2023 27th Int. Conf. Syst. Theory, Control Comput. (ICSTCC), Timisoara, Romania, pp. 227–232. 2023.
[31] S. Simona-Daiana et al., "Improving the Fast Fourier Transform for Space and Edge Computing Applications with an Efficient In-Place Method," Micromachines, vol. 4, no. 2, p. 11, 2024.
[32] F. Liang, Y. Wang, X. Liu, and H. Zhou, "Design of 16-bit fixed-point CNN coprocessor based on FPGA," in Proc. IEEE Int. Conf. DSP Syst. (ICDSP), pp. 1–6. 2018.
[33] 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. 28–38, Mar. 2022.
[34] X. X. Zheng, T. Al-Naffouri, and Y. C. Eldar, "Error feedback approach for quantization noise reduction of distributed graph filters," in Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), Seoul, South Korea, pp. 1–5, 2025.
[35] B. K. Mohanty, P. K. Meher, S. Al-Maadeed, and A. Amira, "Memory footprint reduction for power-efficient realization of 2-D finite impulse response filters," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 61, no. 1, pp. 120–133, 2014.
[36] G. Ganjikunta, M. M. Basha, and S. I. Khan, "Energy efficient FIR filter design using distributed arithmetic," J. Shanghai Jiaotong Univ. (Sci.), vol. 29, no. 6, pp. 1023–1027, 2024.
[37] J. J. Vandenbussche, P. Lee, and J. Peuteman, "Round-off noise of multiplicative FIR filters implemented on an FPGA platform," Appl. Sci., vol. 4, no. 2, pp. 99–127, 2014.
[38] A. M. Krukowski, R. C. S. Morling, and I. Kale, "Quantization Effects in the Polyphase N-Path IIR Structure," IEEE Transactions on Instrumentation and Measurement, vol. 51, no. 6, pp. 1271–1278, 2002.
[39] N. Sādhanā, M. Mohanty, and S. K. Vigneswaran, "An optimization of low-power 4-bit PAL FIR filter using adiabatic techniques," Sādhanā, vol. 48, Art. 84, 2023.

[40] T. M. Ingolfsson, X. Wang, M. Hersche, A. Burrello, L. Cavigelli, and L. Benini, "ECG-TCN: Wearable cardiac arrhythmia detection with a temporal convolutional network," in Proc. IEEE 3rd Int. Conf. Artif. Intell. Circuits Syst. (AICAS), Washington, DC, USA, 2021, pp. 1–4, 2021.

[41] G. K. Kumar, R. Chinnapurapu, and K. Srinivas, "Power and area efficient FIR filter using Radix-2^r multiplier for de-noise the electrooculography (EOG) signal," Sci. Rep., vol. 14, Art. 22599, 2024.

[42] U. Güner, H. Canbolat, and A. Ünlütürk, "Design and implementation of adaptive vibration filter for MEMS based low cost IMU," in Proc. 9th IEEE Int. Conf. Electr. Electron. Eng. (ELECO), Bursa, Turkey, 2015, pp. 130–134, 2015.

[43] P. A. García-Tudela and J.-A. Marín-Marín, "Use of Arduino in primary education: A systematic review," Education Sciences, vol. 13, no. 2, Art. no. 134, 2023.

[44] N. K. Prabowo, M. Paristiowati, and I. Irwanto, "Arduino-based real-time data acquisition systems: Boosting STEM career interest," Int. J. Eval. Res. Educ., vol. 13, no. 4, pp. 2316–2325, 2024.

[45] Y. S. Shmaliy, Y. Xu, J. A. Andrade-Lucio, and O. Ibarra-Manzano, "Predictive tracking under persistent disturbances and data errors using H_2 FIR approach," IEEE Trans. Ind. Electron., vol. 69, no. 6, pp. 6121–6129, 2022.