Implementation and Analysis of a Low-Power Heart Rate Prediction Solution Using Hardware Acceleration on the ESP32-S3 Microcontroller

João P. M. Clevelares[1], Higor D. Oliveira[1], André G. C. Pacheco [1], Luis A. Souza Jr[1], Rodolfo S. Villaça [1]

[1] Departamento de Informática (DI)
Universidade Federal do Espírito Santo (UFES)
Vitória – ES – Brasil

{joao.clevelares, higor.d.oliveira}@edu.ufes.br
{apacheco, la.souza, rodolfo.villaca}@inf.ufes.br

Abstract. *This work details the implementation of an accelerometry-based heart-rate estimation system optimized for the ESP32-S3 microcontroller. It leverages PIE-based Single Instruction, Multiple Data (SIMD) extensions via ESP-DSP library to accelerate the pre-processing, training and inference of a continual-learning model. Under a modular FreeRTOS architecture, the system achieved 4.99 μs training latency, requiring less than 1 kB of RAM for the ML model itself. Results show a 30.6% computational performance gain and 8% energy consumption reduction with acceleration enabled. These efficiencies allow race-to-sleep strategies that extend the estimated battery life to 51 days on COTS hardware.*

1. Introduction

In recent years, there has been a significant increase in the demand for wearable devices, with an expected annual growth rate of 55% until 2029. It is estimated that 80% of global consumers use this type of device for health monitoring [Del-Valle-Soto et al. 2024]. Among the various features offered, heart-rate (HR) monitoring stands out as one of the most relevant vital metrics, serving as the basis for calculating caloric expenditure, sleep quality and intensity levels in physical exercise [Ogata et al. 2024].

Traditionally, HR estimation in these devices is performed through photoplethysmography (PPG), which measures variations in blood volume using LEDs and photodiodes [Allen 2007]. However, this technique suffers from limitations, such as sensitivity to *Motion Artifacts* (MA), a type of noise derived from movements. During intense physical activities, the signal often becomes unstable, which compromises the accuracy of the data provided to the user [Shin and Cho 2019].

As an alternative to PPG, accelerometry-based models can help to mitigate impacts of MA and, in addition, can estimate HR with energy costs up to 5000 times lower than PPG [Elsts et al. 2018], though they face the challenge of extracting patterns from noisy signals without a direct blood volume reference. In this context, the emergence of AI-optimized microcontrollers allows complex mathematical models to be executed locally. This edge computing capability enables modular architectures that achieve a sustainable balance between high-performance signal processing and energy efficiency.

While these models are often validated on high-performance mobile architectures, proprietary commercial hardware, or general-purpose ARM-based microcontrollers, this work differs by presenting the implementation and analysis of an adaptation of the Personalized Exponential Model (PEM) [Pacheco et al. 2023a] tailored for a low-cost commercial off-the-shelf (COTS) platform: the ESP32-S3 SoC[1]. Based on the Xtensa LX7 architecture, this approach exploits chip-specific SIMD instructions to accelerate mathematical and Digital Signal Processing (DSP) operations essential for accelerometry-based prediction. The investigation focuses on the integration of a modular firmware architecture with hardware-vector capabilities, measuring latency, memory, and energy consumption. The subsequent sections detail the related work, algorithm's mathematical basis, system architecture, experimental methodology, results and conclusions.

2. Related Work

Embedded HR estimation involves a trade-off between accuracy and power autonomy. Beyond ECG-based methods, research primarily focuses on two fronts: mitigating MA in optical signals and exploring low-power alternatives, such as inertial sensors.

Classical PPG approaches utilize adaptive filters and decomposition techniques to cancel noise. Algorithms such as TROIKA [Zhang et al. 2015] and NR-OSC-ANF [Shin and Cho 2019] employ Singular Spectrum Analysis (SSA) combined with accelerometry to mitigate the impacts of MA. With the advancement of Edge Computing, Deep Learning models like FitRec [Ni et al. 2019] and Deep PPG [Veluguri 2025] have been proposed to increase prediction robustness. Despite their accuracy, the high memory and processing requirements often make them impractical for low-power Systems-on-a-Chip (SoCs).

To achieve extreme efficiency, inertial sensors are used as the primary data source. Models such as AR-PID [Pacheco et al. 2023b] and PHM [Pacheco et al. 2024] combine linear regression and Proportional-Integral-Derivative (PID) controllers over accelerometer signals to mimic cardiovascular kinetics, while the PEM [Pacheco et al. 2023a] adopts a recursive exponential formulation for real-time updates without history storage.

3. Theoretical Background

The PEM does not rely on PPG data to estimate HR. However, it was not designed to replace PPG. One of the approaches in which the model can be used is to provide support when the PPG signal is affected by MA, which occurs mainly during physical exercise.

An alternative application for the model involves its operation even when the PPG signal quality is not affected by MA. The approach involves using PEM for HR estimation most of the time, while the device periodically verifies the error between the PEM and PPG. If the error remains below a predefined threshold, PEM continues the inference process without using the PPG signals; otherwise, PPG is reinstated as the primary method, and PEM reverts to the personalization phase. This strategy can be used to reduce the device's overall power consumption by avoiding continuous PPG sensing.

Figure 1 shows the general structure of the PEM model, which consists of three main blocks: (1) Preprocess Block, (2) Prediction Block and (3) Manager Block.

[1]`https://www.espressif.com/en/products/socs/esp32-s3`

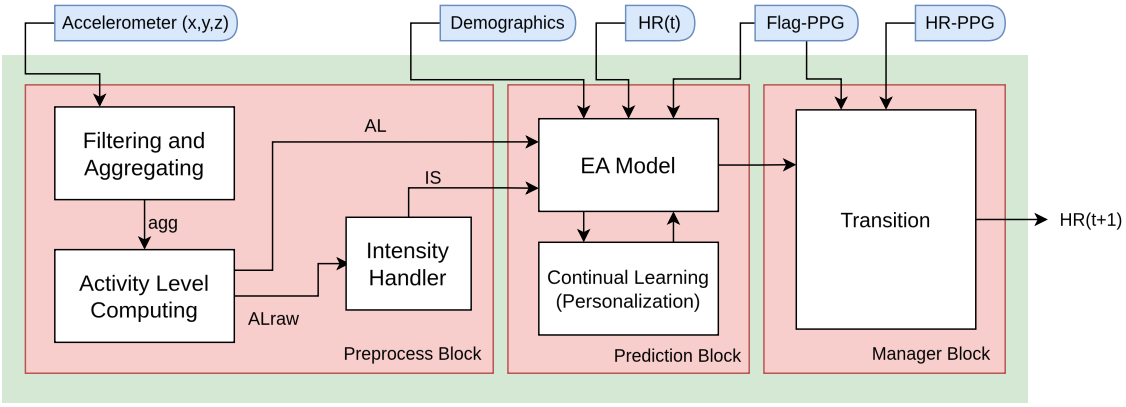

Figure 1. Modular PEM architecture, integrating data preprocessing, personalized prediction via continual learning, and dynamic sensor-to-model management.

The PPG provides the system with the signal-quality flag (`Flag-PPG`). Assuming the approach to mitigate effects of MA, if the signal is reliable (i.e., free from significant MA), the PEM can performs personalization using the `HR-PPG` as ground truth; otherwise, it can infer the next HR using its trained parameters. The `Manager Block` is responsible for transitioning between the two methods.

3.1. Pre-processing

The `Preprocess Block` comprises signal filtering, *Activity Level* (AL) and *Intensity State* (IS) computation. Adapted from the original algorithm's 8-12Hz band-pass design, the accelerometer signals (a_x, a_y, a_z) are processed by a 2^{nd}-order high-pass Finite Impulse Response (FIR) filter with coefficients $\{-0.5, 1, -0.5\}$. This provides a cutoff frequency of approximately 8Hz at a 25Hz sampling rate, removing the gravity component and low-frequency noise while preserving dynamic motion. Next, the signals filtered in the previous step are aggregated using the Euclidean norm (Eq. 1):

$$agg(a_x, a_y, a_z) = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{1}$$

In this context, the raw *Activity Level* (AL_{raw}) is computed by integrating the aggregated signal over a sliding time window using the trapezoidal approximation rule, presented in Eq. 2:

$$AL_{raw} = \int_{t_1}^{t_2} agg(\mathbf{a}) \approx \Delta agg(\mathbf{a}) \sum_{k=1}^{N-1} agg(\mathbf{a}_k) + \frac{agg(\mathbf{a}_n) + agg(\mathbf{a}_0)}{2} \tag{2}$$

where $a = [a_x, a_y, a_z]$, N is the number of data points within the window, which depends on the accelerometer signal sampling rate and $\Delta agg(a) = \frac{agg(a_{t2}) - agg(a_{t1})}{N}$. AL_{raw} is used to compute IS in the `Intensity Handler`. For the Exponential Approximation Model (`EA Model`) input, a normalized version of the *Activity Level* is used, which is obtained by applying a clipping operation over the $agg(a)$ to ensure the value remains within $[0, 1]$. Then, the signal is integrated and the result is divided by the window length $(t_2 - t_1)$.

The IS defines which bias (to be described later in Eqs. 4 and 8) will be selected for training. Thus, it can assume two distinct values: low or high. Technically, a threshold ϕ_i is defined based on the empirical observation that the activity level typically exhibits two well-defined states with a clear transition between them. Therefore, ϕ_i is set at the midpoint of this transition to effectively distinguish between these levels. If $AL < \phi_i$ for a pre-defined time t_ϕ, the state is set to low; otherwise, to high.

3.2. Estimation and Training

The cardiovascular response to effort is dynamic and can be approximated by exponential functions over time [Zakynthinaki 2015]. The PEM exploits this kinetics with a recursive formulation, estimating $HR(t+1)$ from $HR(t)$ and from a physiological target $\tilde{HR}$, in order to track HR variations even when optical signals are unavailable.

The model predicts HR at time $t+1$ through a recursive function (Eq. 3):

$$HR(t+1) = \tilde{HR} - [\tilde{HR} - HR(t)] \times e^{-\frac{1}{\tau}} \tag{3}$$

where:

- $HR(t)$: heart rate at the current time.
- τ: constant that defines the rate of cardiac acceleration or deceleration.
- $\tilde{HR}$: target value to be reached for a given effort level, determined by Eq. 4:

$$\tilde{HR} = D_s \times W^\top + \{b_{low}, b_{high}\} \tag{4}$$

where $\tilde{HR}$ is the result of a linear regression whose input is the vector $D_s = [AL, BMI, age, male, female]$, weighted by the coefficient vector $W = [w_1, \ldots, w_5]$. The BMI is the Body Mass Index, the $male$ and $female$ are represented by binary variables and must assume 1 or 0. Thus, the model has only eight trainable parameters: $[w_1, .., w_5]$, b_{low}, b_{high}, and τ.

In this context, to fine tune the parameters in real time, the model implements the Gradient Descent algorithm using derivatives of the Mean Absolute Error (MAE) over time, presented in Eq. 5. Since MAE is not differentiable at zero, and considering y and $\hat{y}$ as the HR estimated from PPG and from the PEM, respectively, the derivatives are computed for $y_i - \hat{y}_i > 0$ or $y_i - \hat{y}_i < 0$ by simply flipping the sign. Letting $\Omega = \{W, \tau, b_{low}, b_{high}\}$, we have:

$$\frac{\partial MAE}{\partial \Omega} = \frac{\partial}{\partial \Omega}\left(\frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y}_i)\right) \tag{5}$$

Expanding for each parameter, the terms are translated into the weight-update equations (Eqs. 6, 7, 8) used during the model's online training:

$$W_{t+1} = W_t \pm \alpha(D_s e^{-\frac{1}{\tau_t}} - D_s) \tag{6}$$

$$\tau_{t+1} = \tau_t \pm \alpha \frac{-e^{-\frac{1}{\tau_t}}}{\tau_t^2}[D_s \times W_t^T - HR(t)] \tag{7}$$

$$\{b_{t+1_{low}}, b_{t+1_{high}}\} = \{b_{t_{low}}, b_{t_{high}}\} \pm \alpha(e^{-\frac{1}{\tau_t}} - 1) \tag{8}$$

where α is the learning rate.

4. System Architecture and Implementation

The ESP32-S3[1] incorporates a dual-core Xtensa LX7 CPU (up to 240 MHz, 512 kB SRAM and 16 MB flash) featuring the Processor Instruction Extensions (PIE) unit, a SIMD-based architecture designed for AI acceleration [Espressif Systems 2025a]. This unit is centered around a bank of eight 128-bit registers, which can be partitioned into 16x8-bit, 8x16-bit, or 4x32-bit words for parallel processing. To support this high-bandwidth architecture, the Arithmetic Logic Unit (ALU) performs up to 16 multiply-and-accumulate (MAC) operations per instruction on 8-bit data, utilizing two 160-bit accumulators and one 40-bit accumulator to consolidate results without precision loss.

Memory efficiency is achieved through the PIE Addressing Unit, which enables 128-bit data reads in a single cycle, given that the data is aligned to 16-Byte addresses. These hardware capabilities are enabled by the Espressif ecosystem through libraries such as ESP-NN[2] and ESP-DSP[3]. This work leverages the ESP-DSP[3] to optimize the PEM modules, using primarily dot-product and FIR-filter routines.

The firmware architecture was designed to operate in a modular and concurrent manner, using Espressif's ESP-IDF SDK[4] and the FreeRTOS real-time operating system[5] to manage the data flow between modules. As shown in Figure 2, the system consists of four main tasks interconnected by queues, a buffer pool (also implemented as a queue), components from Espressif's ecosystem (in orange), and components created and dedicated to the specific tasks (in purple). The pipeline operates in a circular manner: `RX xTask` captures the pointer to a buffer in the pool, processes it, fills in the required data, and forwards the pointer to the queue destined for the next task. Thus, all tasks repeat the process until the pointer returns to the pool and the cycle restarts.

To validate the real-time performance on the ESP32-S3[1], a Hardware-in-the-Loop (HIL) environment was configured, as described in Figure 3. A Python script acted as a bridge, streaming the pre-recorded accelerometer data and HR ground truth to the SoC via UART. This setup simulated a real-time sensor feed, allowing the system to process the incoming signals and return the telemetry metrics to the host computer for logging and subsequent analysis.

4.1. Buffer Structure and Memory Management

To avoid the overhead of dynamic allocation, a static `buffer-pool` system was implemented. These buffers are used to carry data such as: the accelerometer signal window, *Activity Level*, HR ground truth and $HR(t+1)$. Such data persist throughout the pipeline for communication between tasks and also for the system output. To ensure compatibility with SIMD instructions, all data accessed by optimized ESP-DSP[3] routines are defined using the `__attribute__((aligned(16)))` directive, aligning data to 16-byte boundaries. Regarding memory constraints, the core tasks: `Preprocess xTask` and `Prediction xTask`, operate with a limited stack size of 1024 Bytes each.

[2]`https://components.espressif.com/components/espressif/esp-nn/versions/1.1.2/readme`
[3]`https://components.espressif.com/components/espressif/esp-dsp/versions/1.7.0/readme`
[4]`https://idf.espressif.com/`
[5]`https://www.freertos.org/`

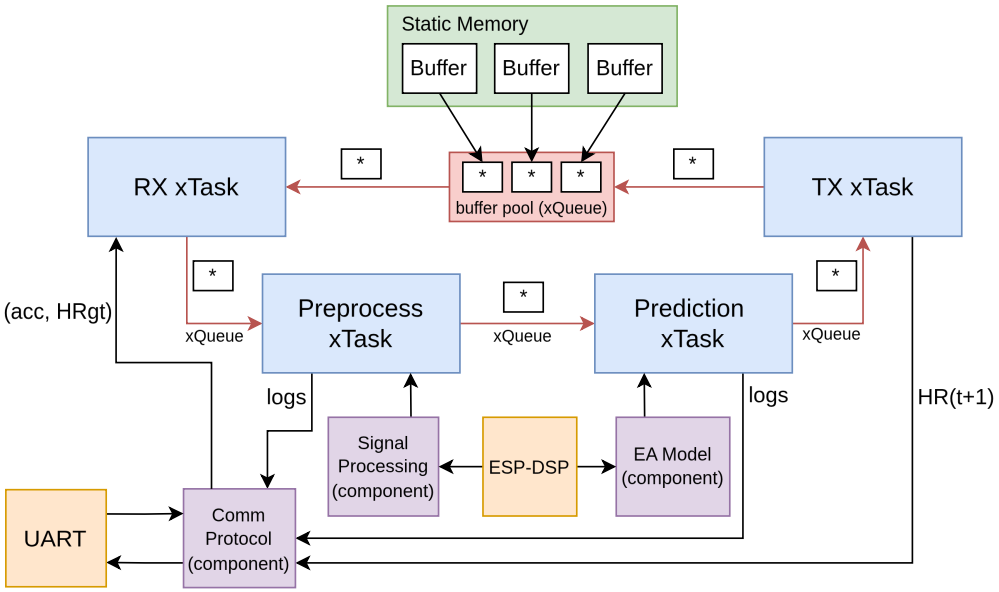

Figure 2. Proposed firmware architecture showing the task-based execution flow. The design features a modular structure using queues for synchronization and a static memory management.

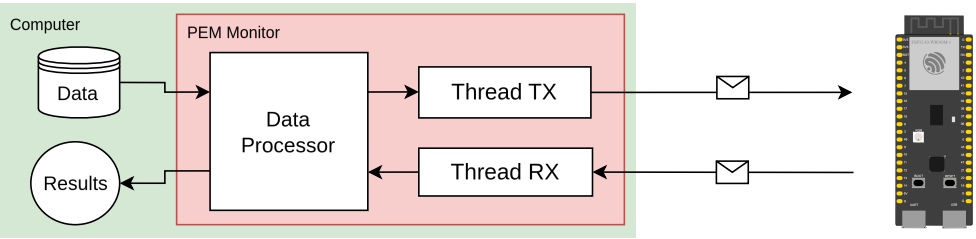

Figure 3. Block diagram of the PEM test environment. The setup consists of a host-based monitoring tool designed to stream input data and log inference results from the embedded device.

4.2. Communication and Synchronization

The communication between computer and SoC is provided by the `Comm Protocol` component (see Figure 2), which bridges the computer's asynchronous threads with the firmware's tasks. Specifically, the host's `Thread TX` transmits serialized data input to `RX xTask`. After a model's execution cycle, the `TX xTask` uses the same protocol component to return the predicted $HR(t+1)$ and execution logs to the computer's `Thread RX`. This structure ensures that the `Comm Protocol` acts as a synchronized abstraction layer, maintaining data integrity across the UART interface while isolating the communication logic from the core processing tasks.

4.3. Signal Processing

The `Preprocess xTask`, which corresponds to the `Preprocess Block` in Figure 1, executes the accelerometer signal processing by managing three FIR filter instances (a_x, a_y, a_z). This task leverages ESP-DSP[3] functions to map convolution operations to the PIE unit's 16/32-bit multipliers and to compute the aggregated magnitude (Eq. 1) of the sliding window. Subsequently, the *Activity Level* (AL) is derived through a sum-and-accumulate implementation of the trapezoidal-rule integration (Eq. 2).

4.4. Inference and Personalization Engine

The `Prediction xTask`, representing the `Prediction Block` in Figure 1, handles both real-time inference (Eqs. 3, 4) and personalization (Eqs. 5–8). The computational core of this stage relies on the vectorized dot-product routine from the ESP-DSP library[3], which calculates the physiological target $\tilde{HR}$ by processing the weight vector W and input data D_s (Eq. 4). This optimized routine is also central to the model-parameter update logic.

5. Experimental Methodology

The system evaluation aimed to quantify the performance of the embedded PEM model, analyzing prediction fidelity and the computational cost of hardware acceleration. Thus, the model was also implemented without acceleration for comparison purposes. Evaluating the non-accelerated version serves to quantify the performance bottlenecks inherent in standard CPU execution. In addition, tests were carried out with the MCU operating at 80 MHz, 160 MHz, and 240 MHz.

5.1. The Dataset and Model Parameters

The publicly available BAMI1 dataset [Lee et al. 2019] was selected for evaluating the proposed implementation because its walking and running sessions mirror the activities in the original study [Pacheco et al. 2023b] and it provides a substantial number of participants. It contains 12-minute records from 24 individuals (average age 26.9; gender distribution of 0.58 male and 0.42 female). Since individual Body Mass Index (BMI) details were unavailable, an average adult value of 22.7 was adopted based on [Jung et al. 2020]. The original signals, sampled at 50 Hz for the accelerometer and 125 Hz for the HR ground truth, were resampled via linear interpolation to 25 Hz and 1 Hz, respectively, to ensure compatibility with the model's requirements.

As the model is designed for individual personalization, training and testing were performed independently for each subject. Given the lack of predefined splits or signal quality flags in the dataset, 60% of the data was allocated for training and 40% for testing, using the ECG-measured reference HR as the ground truth. The model parameters, used in Eqs. 3 to 8, were empirically adapted from the original implementation [Pacheco et al. 2023a] to serve as the initial starting point: $W = [93.75, 0.82, -0.90, -6.44, -0.09]$, $b_{low} = 100$, $b_{high} = 120$, $\tau = 56.25$, $\alpha = 0.0075$, $\phi_i = 0.2$, and $t_\phi = 5$. While the parameters W, b_{low}, b_{high}, and τ are subsequently updated during the online fitting process to achieve fine-tuned personalization, α, ϕ_i, and t_ϕ remain as fixed hyperparameters.

5.2. Performance Metrics Collection

Performance was measured from two perspectives:

- **Physiological**: The system exports the estimated heart rate $HR(t+1)$, the real reference (HR ground truth), and other data. From this information, the MAE is computed for each individual in the dataset. These individual metrics are then used to calculate the overall average MAE and to generate the curves shown in Figure 4.

- **Computational**: Execution latency was quantified using internal hardware counters to record total CPU cycles per operation. Memory utilization for both the preprocessing and prediction tasks was assessed through high-water mark analysis, which tracks peak stack consumption to determine the minimum safety margin. Energy consumption was estimated by correlating measured task duty cycles with the power states defined in the datasheet [Espressif Systems 2025b].

6. Results and Discussion

This section presents the results of the PEM model implementation on the ESP32-S3[1], analyzing the impact of hardware acceleration on latency, energy and memory consumption.

6.1. Physiological Performance

Figure 4 shows the heart-rate tracking estimated by the system compared to the ground truth during the 12-minute session. Personalization occurred between 60 and 480 seconds, while HR prediction was performed in the final running-and-walking session between 480 and 780 seconds. Note that the model can accurately track the transitions between walking and running states, presenting a smooth exponential response that mimics real cardiovascular kinetics.

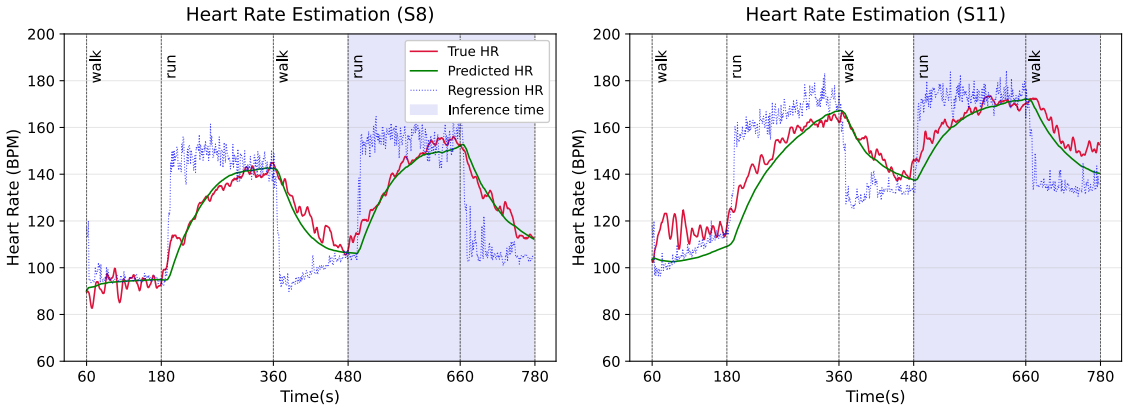

Figure 4. Estimated Heart Rate (HR) using PEM model over a 12-minute session for subjects S8 and S11 from dataset [Lee et al. 2019]. The True HR is the ground truth HR obtained from annotated ECG signals. Regression HR is $\tilde{HR}$. Predicted HR is $HR(t+1)$.

Quantitatively, the model exhibits a MAE of 7.66 ± 3.17 BPM for walking and 10.61 ± 5.86 BPM for running across all dataset subjects. To accommodate non-normal distributions, a paired Wilcoxon test [Wilcoxon 1945] on individual MAEs was performed, confirming this performance gap is significant ($p < 0.05$). This expected difference stems from the greater intensity and movement variability of running segments. Nevertheless, combined sessions achieved an overall MAE of 9.48 ± 4.93 BPM, demonstrating stable estimation across effort regimes (Figure 4).

Although deep models on this same dataset report lower MAEs — 2.17 BPM for BeliefPPG [Bieri et al. 2023] — they require complex CNN/LSTM architectures with more than 100K parameters ($\sim$10,000$\times$ PEM's footprint). This complexity renders them

impractical for low-power wearables, highlighting PEM's superior cost-benefit for edge-embedded processing.

6.2. Latency Analysis and SIMD Acceleration

Table 1 presents the system computational performance. One can observe that acceleration via SIMD instructions (PIE unit) provided a latency reduction of approximately 30.6% at all tested operating frequencies. At 240 MHz, the full execution cycle, in the worst case (filtering + integration + training + inference), is completed in only 44.08 μs, with no cost to model accuracy.

Table 1. Current, latency, and energy per complete pipeline cycle for different operating frequencies, comparing execution without acceleration (No-Acc) and with SIMD acceleration (Acc).

Frequency	Mode	Current (mA)	Latency (μs)	Energy (mJ)
80 MHz	No-Acc	47.30	190.44	0.0297
	Acc	56.30	132.20	0.0246
160 MHz	No-Acc	64.10	95.30	0.0202
	Acc	81.10	66.01	0.0177
240 MHz	No-Acc	81.30	63.54	0.0170
	Acc	107.90	44.08	0.0157

Figure 5 illustrates this time reduction. Notably, pre-processing is the main bottleneck, consuming about 82% of total time, which validates the strategy of optimizing FIR filters using the ESP-DSP library[3]. At the operating point with the lowest latency (240 MHz), pre-processing reached 36.26 μs, while training reached 4.99μs and inference 2.83μs.

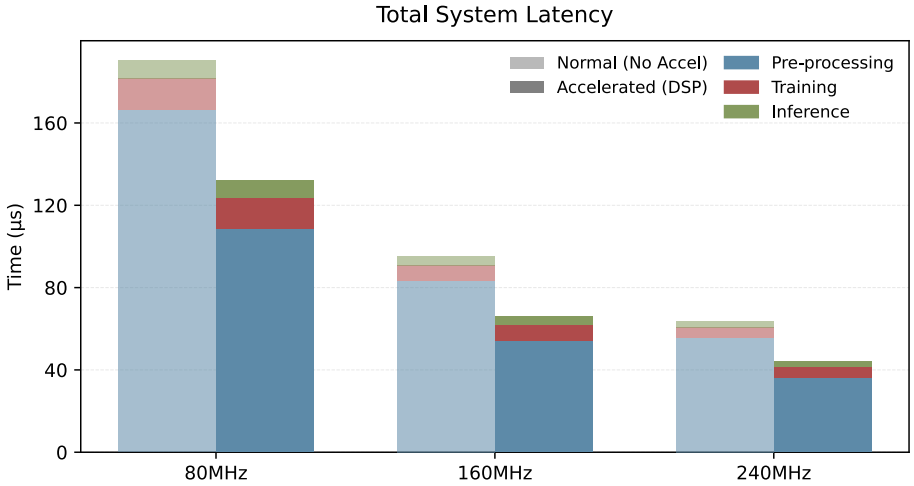

Figure 5. Total pipeline latency (comprising pre-processing, training, and inference times) for No-Acc and Acc modes at 80, 160, and 240 MHz.

6.3. Memory Consumption

Since the system does not use heap allocation, static memory usage was monitored. It was verified that `Preprocess xTask` reached a peak usage of 920 Bytes, while

`Prediction xTask` (responsible for personalization and inference) reached a maximum of 896 Bytes, also considering the allocation of variables and functions used for metrics extraction. Furthermore, considering that two buffers were used in the pool, each buffer occupies 360 bytes (376 bytes including variables for metric extraction). The final system image occupies a total of 231.3 kB of flash memory. These values show that the system operates with low memory consumption, keeping the model's core footprint with approximately 2.5 kB of RAM.

6.4. Energy Efficiency and Battery Life

The current consumption values shown in Table 1 were taken from the MCU datasheet [Espressif Systems 2025b], considering the 2 processor cores and the peripheral clock enabled, but WiFi disabled. The energy consumed per cycle was theoretically estimated using the Eq. 9:

$$E = V \cdot I \cdot \Delta t \tag{9}$$

where I is the tabulated current for a given usage, the operating voltage is 3.3V, Δt is the cycle latency in seconds, and E is the result in mJ. Although the current consumption in accelerated mode at 240 MHz is higher (107.9 mA), the short execution time results in the lowest energy consumption per prediction: 0.0157 mJ. Furthermore, in accelerated mode, an average reduction of 8% in energy consumption per cycle was achieved across all three operating frequencies.

The system's autonomy was evaluated in isolation (without considering I/O and peripheral operations) and under a race-to-sleep perspective, which prioritizes rapid task completion to maximize the duration in light sleep mode ($I_{sleep} = 240\mu A$ [Espressif Systems 2025b]). This approach isolates the algorithm's computational cost from peripheral overhead. Battery life (B_{life}) is estimated via Eq. 10, considering a capacity (B_{cap}) of 300 mAh, the active current (I_{op}), and the duty cycle (D):

$$B_{life} = \frac{B_{cap}}{24 \cdot ((D \cdot I_{op}) + ((1 - D) \cdot I_{sleep}))} \tag{10}$$

Figure 6 illustrates the impact of SIMD acceleration on the device's longevity. The optimized configuration (240 MHz with acceleration, $I_{op} = 107.9\ mA$) achieves a duty cycle of only 0.0044%, resulting in an estimated 51.1 days of autonomy. In contrast, the non-accelerated baseline at 80 MHz draws a lower active current ($I_{op} = 47.3\ mA$) but increases the duty cycle to 0.019%. Despite the lower peak current, the prolonged active time reduces the overall lifespan to 50.2 days, confirming the energy efficiency of the accelerated approach.

7. Conclusion

This work implemented and evaluated the PEM model on the ESP32-S3, showing that accelerometry-based heart-rate estimation can be executed with low latency and low memory consumption on a low-cost microcontroller. With SIMD acceleration, a 30.6% reduction in execution time and a latency of 44.08 μs per complete cycle were observed, along with an 8% reduction in power consumption, while keeping the model-core footprint to aproximatelly 2.5 kB of RAM.

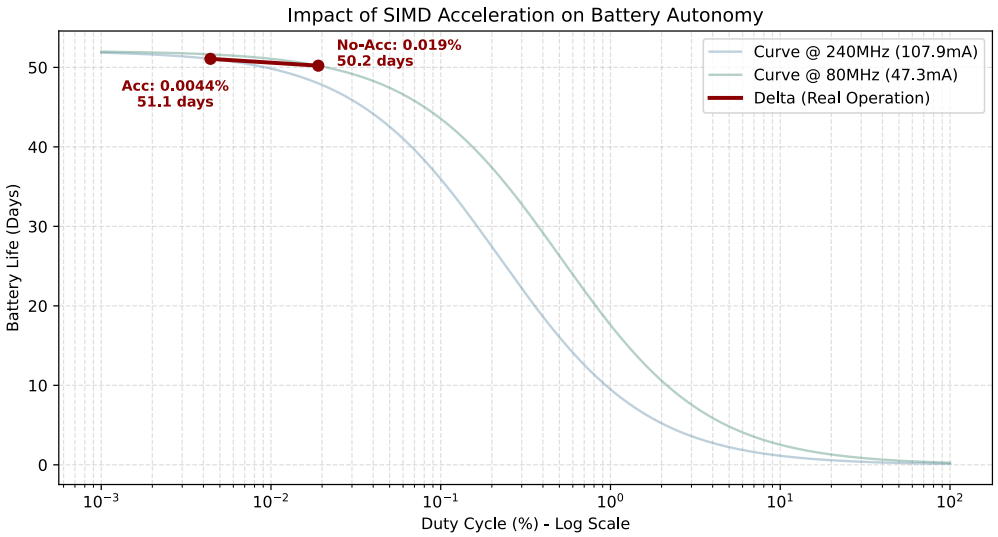

Figure 6. Battery life projection: accelerated 240 MHz achieves higher autonomy than 80 MHz baseline by reducing duty cycle.

The results also indicate that the computational cost is mostly in pre-processing, while personalization and inference remain lightweight, leaving headroom for coexistence with other FreeRTOS tasks (for example, sensor acquisition, logging, BLE, etc.). This headroom enables race-to-sleep strategies and, in the analyzed scenario, a theoretical 51-day battery life, while also supporting higher inference and sampling rates with ample margin.

Future work involves empirical validation of power consumption using physical sensors and precision instrumentation. Proposed hardware-level optimizations include the implementation of deep-sleep mode, with static buffers and filter delay lines allocated to the RTC Fast Memory to preserve system state during low-power cycles. Furthermore, it is intended to investigate task partitioning within the SoC's heterogeneous architecture by offloading specific low-frequency operations to the Ultra-Low-Power (ULP) coprocessors, evaluating the resulting energy efficiency against the associated inter-processor communication overhead.

Acknowledgements

The authors thank Fapes (2026-B74MN, 2023-RWXSZ) and CNPq; the Department of Science and Technology of the Ministry of Health (Decit/SECTICS/MS); and Brazilian National Program of Genomics and Precision Health (Genomas Brasil). Also, Espressif for providing us with hardware and support in the development of this work.

References

Allen, J. (2007). Photoplethysmography and its application in clinical physiological measurement. *Physiological Measurement*, 28(3):R1–R39. PMID: 17322588.

Bieri, V. et al. (2023). Beliefppg: Uncertainty-aware heart rate estimation from ppg signals via belief propagation. In *Uncertainty in Artificial Intelligence*, pages 173–183. PMLR.

Del-Valle-Soto, C. et al. (2024). Unveiling wearables: exploring the global landscape of biometric applications and vital signs and behavioral impact. *BioData Mining*, 17(1):15.

Elsts, A. et al. (2018). On-board feature extraction from acceleration data for activity recognition. In *Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks*, EWSN '18, page 163–168. International Conference on Embedded Wireless Systems and Networks (EWSN).

Espressif Systems (2025a). *ESP32-S3 Technical Reference Manual*. Espressif Systems (Shanghai) Co., Ltd., Shanghai, China. Version 1.7.

Espressif Systems (2025b). *ESP32-S3-WROOM-1 & ESP32-S3-WROOM-1U Datasheet*. Espressif Systems (Shanghai) Co., Ltd., Shanghai, China. Version 1.7.

Lee, H., Chung, H., and Lee, J. (2019). Motion artifact cancellation in wearable photoplethysmography using gyroscope. *IEEE Sensors Journal*, 19(3):1166–1175.

Ni, J., Muhlstein, L., and McAuley, J. (2019). Modeling heart rate and activity data for personalized fitness recommendation. In *WWW '19: The World Wide Web Conference*, pages 1343–1353.

Ogata, H. et al. (2024). Individually optimized estimation of energy expenditure in rescue workers using a tri-axial accelerometer and heart rate monitor. *Frontiers in Physiology*, Volume 15 - 2024.

Pacheco, A. G. C. et al. (2023a). Method and wearable electronic system for predicting the heart rate of a user during a physical activity, and, non-transitory computer readable storage medium. US Patent Application US 2023/0190120 A1. Published: Jun. 22, 2023. Filed: Feb. 24, 2022.

Pacheco, A. G. C. et al. (2023b). Towards low-power heart rate estimation based on user's demographics and activity level for wearables. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

Pacheco, A. G. C. et al. (2024). Learning to estimate heart rate from accelerometer and user's demographics during physical exercises. *IEEE Journal of Biomedical and Health Informatics*, 28(9):5092–5102.

Shin, J. and Cho, J. (2019). Noise-robust heart rate estimation algorithm from photoplethysmography signal with low computational complexity. *Journal of Healthcare Engineering*, 2019:1–7.

Veluguri, S. P. (2025). Deep ppg: Improving heart rate estimates with activity prediction. In *2025 1st International Conference on AIML-Applications for Engineering & Technology (ICAET)*, pages 1–6.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

Zakynthinaki, M. S. (2015). Modelling heart rate kinetics. *PLOS ONE*, 10(4):1–26.

Zhang, Z., Pi, Z., and Liu, B. (2015). Troika: A general framework for heart rate monitoring using wrist-type photoplethysmographic signals during intensive physical exercise. *IEEE Transactions on Biomedical Engineering*, 62(2):522–531.