ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

Ingegneria Elettronica, Telecomunicazioni e Tecnologie dell'Informazione

Ciclo XXXVII

Settore Concorsuale: 09/H1 – SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Settore Scientifico Disciplinare: ING-INF/05 - INFORMATICA

Optimized Algorithms and Machine Learning Techniques for Biosignal Processing on Ultra-Low-Power Computing Platforms

Presentata da: ***BENEDETTA MAZZONI***

Coordinatore Dottorato **Supervisore**

Prof. DAVIDE DARDARI **Prof. LUCA BENINI**

Esame finale anno 2025

ALMA MATER STUDIORUM - UNIVERSITY OF BOLOGNA

Optimized Algorithms and Machine Learning Techniques for Biosignal Processing on Ultra-Low-Power Computing Platforms

by

Benedetta Mazzoni

A thesis submitted for the degree of
Doctor of Philosophy

in the
Faculty of Engineering
Department of Electrical, Electronic, and Information Engineering (DEI)

February 2025

I wish to dedicate this thesis to my lovely family.

Acknowledgments

I wish to thank Prof. Luca Benini for the opportunity to pursue my Ph.D. in his research group and for his constant guidance. I specifically appreciated Prof. Giuseppe Tagliavini and Prof. Simone Benatti for their persevering supervision, their teachings, and the motivation they gave me. I thank them for the support and independence they gave me.

I thank Prof. Andrea Cossettini for welcoming me and following my work to his PULP group at ETH for a visiting research period.

I also thank GreenWaves Technologies for the preview access to the GAP SDK.

In closing, I would like to express my gratitude to all my colleagues at the EEES Lab for creating an enjoyable and inspiring environment.

Abstract

In recent years, advancements in electronic systems have driven the development of implantable and wearable devices that facilitate continuous health monitoring through the extraction of physiological parameters from biosignals, such as Electrocardiogram (ECG) and Electroencephalogram (EEG). Biosignal-based applications have become central to various fields, ranging from fitness to medical-grade diagnostics. However, implementing biosignal processing presents significant challenges, notably in achieving a balance between computational power and energy efficiency, which is essential for extended battery life in portable devices.

This thesis contributes to this field by presenting a framework of end-to-end methodologies designed to optimize energy efficiency in executing computationally intensive signal processing tasks on resource-constrained embedded devices. Through a combination of optimized system architectures, low-power processing strategies, and machine learning-based algorithms, the thesis offers novel solutions for achieving high-performance ExG signal analysis within strict energy budgets. Key aspects include the design of Analog Front Ends (AFEs) to ensure high-fidelity signal capture with minimal energy draw, as well as optimizing digital processors to handle complex operations such as filtering, feature extraction, and pattern classification within limited memory and processing power. Additionally, this research explores the adaptation of machine learning algorithms, such as CNNs and TCNs, for edge-based biosignal processing, emphasizing model compression to reduce computational overhead.

The research demonstrates a sustainable solution for real-time biosignal processing on ultra-low-power (ULP) parallel platforms, offering significant advantages over traditional MCUs in both energy efficiency and processing capability. To validate the proposed methodologies, the thesis investigates two primary case studies. The first focuses on ECG signal processing and classification, showcasing how on-device computation minimizes data transmission and latency, thereby improving privacy, energy efficiency, and responsiveness. The second evaluates ear-EEG as a promising alternative to conventional, full-scalp EEG, demonstrating its viability in mobile health applications.

This dissertation advances the design of energy-efficient biosignal processing systems for wearable and implantable devices, emphasizing the synergy between optimized hardware, low-power digital processing, and machine learning algorithms to enable real-time, on-device biomedical analysis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Rapid advances in miniaturized and efficient electronic systems fuel the development of discrete, implantable, and wearable devices, [1]. leading to exciting new applications. Small-sized devices have gained popularity in healthcare and commercial applications due to their portability [2]. These systems facilitate continuous monitoring of health conditions by extracting physiological parameters from biosignal analysis [3]. The body's physiological activity generates biosignals, such as Electrocardiogram, Electroencephalogram, and Electrooculographic (ExG), which provide information on biomedical parameters directly related to a person's health status or enable pattern extraction for direct communication with the external environment.

The growing trend of small form factor devices is pushing the development of wearables, driven by systems such as health patches and trackers [4], [5]. In the fitness and healthcare area, these systems facilitate remote and continuous monitoring of wellness conditions, extracting physiological parameters from the analysis of biosignals [6].

Biosignal-based applications are beneficial in various fields, from consumer electronics (e.g., fitness) [7] to medical-grade equipment (e.g., patient monitoring) [8]. For instance, fitness-focused smartwatches (e.g., Apple Watch Series, KardiaMobile, and QardioCore), pacemakers [9], and Holter monitors [10] use Electrocardiogram (ECG) analysis for heart rate monitoring and detection of general heart disease [11]. Electroencephalogram-based systems are also gaining popularity in wearable applications for the treatment of neurological disorders such as Parkinson's disease [12], [13], epilepsy [3], and hearing dysfunction [14], as well as for detecting drowsiness [15] and other conditions [16]. However, implementing these techniques requires significant computational power while maintaining low energy consumption, a challenge that has only recently become feasible for wearable devices. In fact, the main limitation of wearable sensor nodes is the need to

perform computationally demanding tasks with high energy efficiency to extend battery life

Examples of low-power digital design and efficient computational architectures [17], [18] have enabled the development of real-time systems capable of executing complex algorithms such as filtering [19], augmentation [18], and pattern identification [20].

Designing efficient real-time systems for processing biosignals presents several challenges. The development of Analog Front Ends (AFEs) is crucial, as they must ensure accurate signal capture while minimizing power draw. For example, [21] propose ultra-low-power AFEs designed to reduce energy consumption during continuous ECG monitoring while maintaining signal fidelity. Optimizing digital processors for real-time ExG signal processing is also essential. These processors must perform computationally intensive tasks such as filtering, feature extraction, and classification, often with limited memory and processing power. This condition requires novel approaches in system-level architecture design, including power management strategies that extend battery life during continuous monitoring [22]. Algorithmic design also plays a significant role. Algorithms like the Pan-Tompkins algorithm for ECG signal processing have been optimized for real-time, low-power execution on embedded systems [23]. In recent years, machine learning (ML) is constantly gaining traction in biosignal processing by enhancing the accuracy, efficiency, and reliability of detection and diagnosis because of its ability to identify patterns and relationships within large and noisy datasets, which may not be apparent through traditional methods. However, applying advanced ML techniques (e.g., CNNs or TCNs) for feature extraction and classification introduces additional computational overhead, making it critical to adapt and compress these models for edge devices [24]. Addressing these challenges requires a holistic approach integrating advancements in AFEs, low-power digital processing, system architecture, and optimized algorithms to create efficient real-time systems for ExG signal processing.

Modern multi-core platforms designed for ultra-low power processing can handle computationally intensive algorithms while meeting real-time requirements, with power consumption limited to just a few milliwatts through power management techniques and near-threshold computing (NTC). NTC operates with supply voltages close to the transistor threshold, which can significantly reduce dynamic and static power consumption. This technique enables processors to achieve substantial energy efficiency by sacrificing clock speed in favor of increased parallelism. The target device used for the experimental assessment of this thesis work is the RISC-V Parallel Processing Ultra-Low Power (PULP) many-core platform. Compared to commercial MCUs (e.g., ARM Cortex M4), PULP-based architectures have demonstrated substantial computational power while

staying within an ultra-low power budget, making them suitable for various applications.

1.1 Contributions & Thesis Structure

The contribution of the research presented in this thesis is the development of optimized end-to-end methodologies to achieve high energy efficiency in executing computationally intensive signal processing workloads for biomedical applications on resource-constrained embedded devices. Specifically, this thesis describes novel system-level designs for leveraging architectural features through software and hardware optimizations, including parallel programming, specialized hardware, and memory management strategies. Multicore architectures can distribute tasks across cores, allowing faster data processing and helping meet real-time constraints in applications like ExG signal analysis. When a single processing unit can meet latency constraints, multiple cores running at lower frequencies can significantly reduce energy consumption, as demonstrated in low-power, multicore systems optimized for biomedical signal processing [25], [26], [25]. Additionally, general-purpose architectures benefit from specialized hardware accelerators, such as DSP or AI accelerators, designed to perform specific operations in a few clock cycles. These accelerators can substantially enhance latency and energy efficiency by offloading computationally intensive tasks from the main processor. By combining these methods, parallel and accelerator-based computing architectures enable new performance levels for real-time biomedical applications on constrained devices [27], [28]. Memory management in embedded systems is challenging due to limited resources, multi-level hierarchies, and stringent power and latency requirements, especially in applications with frequent data transfers. Efficient handling of these transfers is crucial to maintain performance and energy efficiency, as excessive data movement can drain power and increase delays, affecting real-time capabilities. Real-time applications of biosignal processing must orchestrate data movements to minimize latency and power consumption. In architectures with memory hierarchies, employing efficient data transfer strategies, such as data prefetching, memory tiling, and buffer management, can help reduce latency and minimize overhead, especially in resource-constrained environments [29], [30], [31], [32]. Techniques like Direct Memory Access (DMA) are also commonly used to offload data transfer tasks from the main processor, enhancing overall system performance and energy efficiency [33], [34],[35].

As a first case study, this thesis addresses the challenge of processing and classifying ECG signals directly on embedded systems, which offers significant advantages over traditional online processing with an external device, as is often required [36], [37].

Localized ECG processing reduces latency and data transmission requirements, which is advantageous for privacy, energy efficiency, and responsiveness in real-time applications [38]. These benefits make on-device processing especially suitable for wearable or implantable devices operating autonomously and maintaining reliable performance [39], [40]. The primary contribution in this context is the creation of a HW/SW co-design that supports inference on the edge. Current wearable devices on the market are reasonably reliable and accurate in providing heart rate data and detecting basic irregularities, such as atrial fibrillation (e.g., Fitbit Charge 5). However, their signal processbing typically relies on remote servers rather than on-device (edge) processing, necessitating a data connection for comprehensive analysis. While some advanced algorithms and ML techniques achieve high accuracy, they often involve memory usage that exceeds the capabilities of these devices and fail to report energy consumption values as in [41], [42], [43], [44].

As a second case study, this thesis considers Electroencephalogram (EEG) signals, particularly highlighting the advantages of ear-EEG over the traditional, full-scalp EEG configuration. While standard EEG systems typically require multiple electrodes positioned across the entire scalp to capture comprehensive brain activity, ear-EEG offers a promising alternative, enabling convenient, unobtrusive monitoring through electrodes embedded in or around the ear canal. This design reduces the discomfort and setup time associated with full-scalp EEG, thus making it especially suited for wearable devices and prolonged monitoring in everyday settings [45]. Kappel and Kidmose [46] conducted a comparative study between a dry-contact ear-EEG electrode and a scalp EEG wet electrode. Additionally, ear-EEG has shown comparable reliability in specific tasks such as sleep monitoring and auditory attention detection [47]–[49], suggesting it is a viable approach for tasks like audiometric characterization in mobile health applications [48], [50]. This study aims to leverage these advantages by evaluating ear-EEG's feasibility for audiometric assessment, a step toward integrating simplified, non-invasive brain-signal monitoring into compact wearable technology.

The contributions outlined above are elaborated in the following chapters as follows:

Chapter 2 presents the background of the research topics. It includes an in-depth description of the architectures explored in this dissertation (ARM Cortex and the PULP), with an insight into the quantization and deployment tools adopted for ML workloads. This chapter also introduces an overview of the current SoA for time-series data elaboration. Finally, the chapter describes the parallel programming methodology, focusing on the main concepts provided by the hardware abstraction layer (HAL).

Chapter 5 exposes the contribution of ear-based Electroencephalogram (EEG). The target is a wireless, parallel ultra-low-power data acquisition platform (BioGAP) paired

with in-ear EEG electrodes (SoftPulseTM, Datwyler Schweiz AG) to explore auditory-based EEG protocols. Validating the signal quality from the ear-EEG electrodes by analyzing alpha-wave responses and performing quantitative comparisons to standard wet-electrodes (positioned behind the ear), this work demonstrates that the in-ear electrodes allow the acquisition of high-quality EEG signals. It also includes the development of an Auditory Steady-State Response (ASSR)-based experimental protocol with a Python-based stimuli generator (sinusoidal tones with a carrier at 4 kHz and AM modulation at 80 Hz and 88 Hz) that delivers sounds to headphones. BioGAP was used to acquire EEG data from the ear-based electrodes, and an offline data analysis extracted the corresponding EEG response in the frequency domain. The analyses successfully detected the ASSR response from ear-EEG electrodes for multiple modulation frequencies (80 Hz and 88 Hz) and multiple stimulation setups (over-the-ear headphones and bone-conducting headphones). Additionally, the experiments proved the response comparable to the one from standard wet electrodes positioned behind the ear.

Finally, Chapter 6 concludes the presented research.

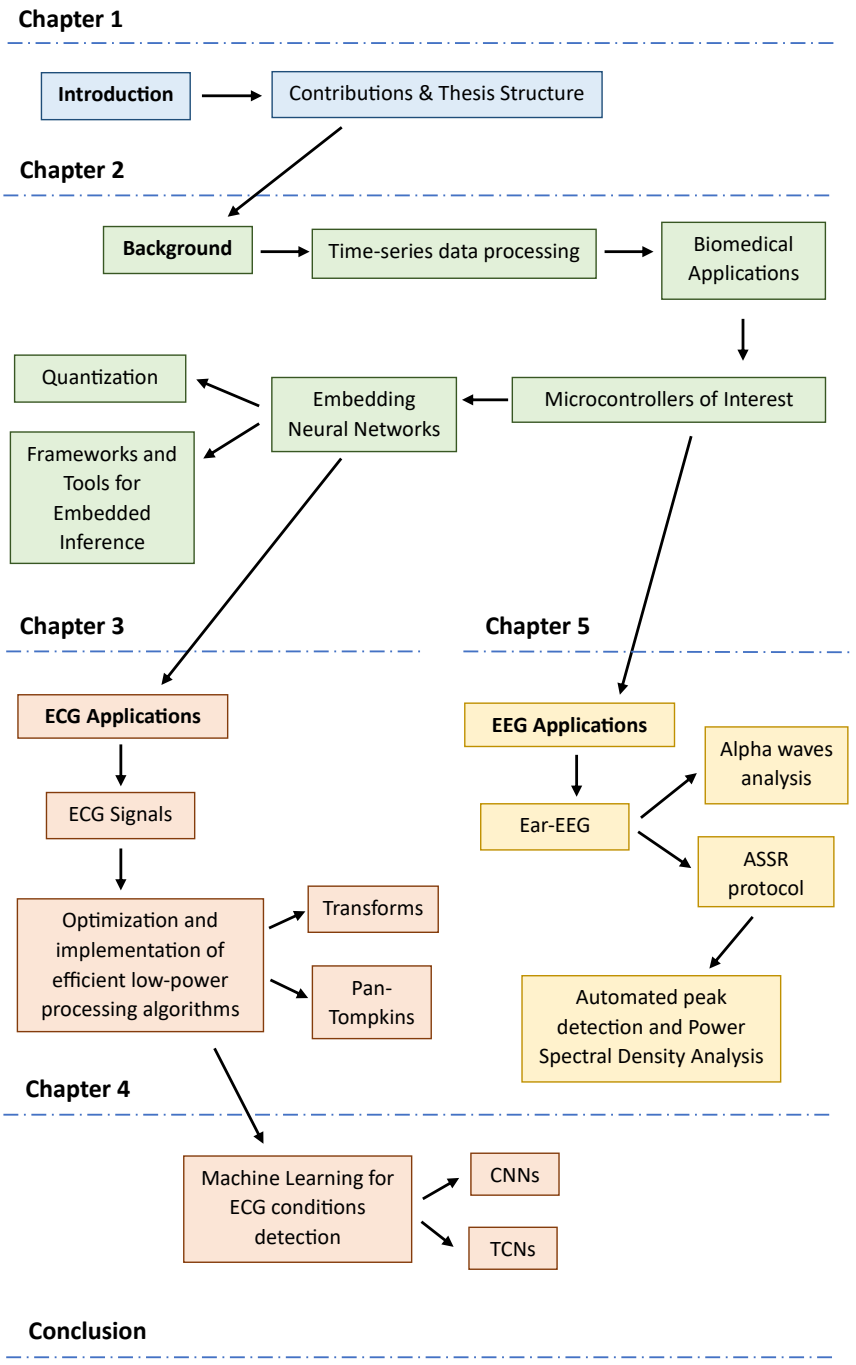


Figure 1.1: Scheme of the contents of this thesis.

Chapter 2

Background

This chapter will present the fundamental concepts that form the basis of the work discussed in this dissertation. Specifically, we will start with an overview of biomedical applications and then explain the fundamental principles of time-series biosignal data processing, which are essential for the biomedical applications discussed in the text. We will explore key techniques such as signal preprocessing, feature extraction, and classification methods. The chapter will highlight the SoA approaches, focusing on the emerging edge computing trend for biosignal analysis and machine learning models in this domain. Finally, the challenges associated with deploying complex biosignal processing systems on resource-constrained edge devices will be addressed, providing a comprehensive background for understanding the contributions made in this dissertation. Next, it will outline the categories of microcontrollers relevant to this thesis. The chapter will overview the tools and techniques for deploying ML models on embedded systems.

2.1 Biomedical Applications based on ECG and EEG Signals

Historically, human body signal monitoring was conducted in laboratories under medical supervision. However, advancements have enabled the development of devices that independently generate reliable data outside clinical settings. These devices allow continuous monitoring, with healthcare providers later analyzing the data to track health status or detect events, providing immediate feedback when necessary. Offline data processing also offers deeper insights into human physiological behaviors. Despite technological progress, there are still scenarios requiring improvement in biomedical devices. This dissertation explores efficient ECG arrhythmia detection and the feasibility of ear-based EEG. Before delving into detailed discussions in subsequent chapters, we

provide an overview of these application scenarios to highlight potential challenges and issues.

2.1.1 Electrocardiogram (ECG)

Electrocardiogram (ECG) monitoring is a widely used method for detecting cardiac arrhythmias, which are abnormal heart rhythms that can lead to severe health conditions, such as stroke, heart failure, or sudden cardiac arrest. The increasing prevalence of cardiovascular diseases and the need for early detection have driven advancements in ECG arrhythmia detection systems [51]. Efficient ECG detection systems are crucial for continuous monitoring and early diagnosis. Traditionally, ECG data has been collected using clinical-grade equipment, which requires patients to visit a healthcare facility and be monitored by a medical professional. However, with the rapid development of wearable technology and mobile health platforms, it is now possible to perform continuous ECG monitoring outside clinical environments. These systems detect irregular heart rhythms in real time, enabling timely intervention and reducing the risk of life-threatening events.

Typical components of an ECG arrhythmia detection system include:

- Wearable ECG sensors: Lightweight, portable devices that can be worn on the body to capture electrical signals from the heart.
- Signal processing algorithms: Techniques used to filter and preprocess ECG signals, removing noise and artifacts that may distort the data.
- Machine learning models: Algorithms designed to classify arrhythmias based on patterns in the ECG signals, enabling automated detection and diagnosis.

In wearable applications, ECG arrhythmia detection systems must balance accuracy, power consumption, and user comfort. Their success relies not only on accurate arrhythmia detection but also on continuous operation in resource-constrained environments [52]. While ECG arrhythmia detection systems have the potential to revolutionize cardiac care, several challenges remain in ensuring their efficiency and reliability. As a result, developers and researchers face various key issues when designing these systems.

ECG signals can be affected by various external factors, including motion artifacts, electromagnetic interference, and body positioning. Wearable sensors are particularly susceptible to noise due to movement and environmental influences. Therefore, one of the primary challenges is maintaining high-quality signal acquisition and preprocessing.

Effective noise-reduction techniques, including advanced filtering methods and adaptive algorithms, are crucial for minimizing distortions and enhancing signal clarity [53], [54].

For ECG arrhythmia detection systems to be effective, they must operate in real-time, continuously monitoring heart activity and providing immediate feedback. Real-time processing is particularly challenging for low-power wearable devices with limited computational resources since detection algorithms must be computationally efficient and accurate enough to avoid false positives or missed arrhythmias [55]. Changing the operational mode, the same algorithms can be applied in an offline processing scenario, which involves the post-acquisition analysis of ECG data over longer periods, providing insights that may not be immediately apparent. Moreover, continuous monitoring generates vast amounts of data, creating challenges for data compression and storage. Advanced machine learning algorithms can be highly beneficial in managing these datasets without losing critical information. Offline processing is valuable for retrospective studies, helping refine detection models by identifying subtle arrhythmias that might not trigger immediate alarms, providing insights for personalized treatments. The challenge is to ensure that offline methods are efficient, minimizing delays while maintaining high accuracy [56].

Machine learning models for arrhythmia detection require extensive datasets to accurately identify a broad range of arrhythmias [57]. However, ECG signals can vary widely due to individual differences such as heart anatomy, age, gender, and medical history. Ensuring models generalize well across diverse patient populations is a challenge, requiring fine-tuning to manage both common and rare arrhythmias while minimizing false alarms, which could lead to unnecessary interventions or patient anxiety.

Energy efficiency is essential for wearable ECG sensors to operate continuously without frequent recharging. This aspect is especially important for long-term monitoring in patients with chronic conditions or those at high risk for cardiac events. Developers must optimize power consumption while maintaining detection accuracy [58]. Since wearable ECG systems are intended to be used over long periods, user comfort is an important consideration. Devices that are too bulky or uncomfortable may lead to poor patient compliance, reducing the effectiveness of the monitoring system. Developers must design discreet, comfortable, and easy-to-use sensors, such as armband devices [59], while ensuring high signal quality.

Efficient ECG arrhythmia detection systems hold great promise for improving cardiac care by enabling continuous, real-time monitoring and early detection of heart abnormalities. However, the development of these systems is not without its challenges. Issues such as signal noise, real-time processing, model accuracy, energy efficiency, and

user comfort must be addressed to create reliable, user-friendly devices that can be deployed in everyday settings. As wearable technology advances, future research should focus on developing robust algorithms, improved signal processing techniques, and enhanced power management solutions. With these improvements, ECG arrhythmia detection systems will be instrumental in reducing mortality and improving the quality of life for millions with heart conditions.

2.1.2 Electroencephalogram (EEG)

Advances in cognitive neuroscience have allowed us to interface directly with the human brain. Thanks to its effectiveness, low cost, and portability, the electroencephalography (EEG) signal is one of the most used techniques for investigating brain function and pathology, both in clinical settings and scientific research [60], [61], [62]. Among diagnostic and screening techniques, EEG analysis and instrumentation is an established standard, since it directly records the electrical field generated by neural activity. The most popular method uses a set of electrodes distributed on the head surface (scalp) [63]. However, other techniques and setups are being explored to enhance comfort and usability, such as ear-EEG systems. Ear-EEG employs sensors embedded in earphones or placed around the ear to capture brain signals in a more discreet and user-friendly manner [49].

This approach offers several advantages over traditional scalp EEG. First, ear-EEG is much more comfortable and non-invasive, allowing for prolonged use without the discomfort associated with scalp electrodes. It is also less obtrusive, making it suitable for real-world monitoring outside clinical environments. The compact and user-friendly design of ear-EEG systems makes them ideal for portable applications, enabling continuous brain activity monitoring during daily activities. Additionally, since ear-EEG systems can be integrated into common earphones, they provide a low-profile solution for neurophysiological monitoring, which may enhance user compliance and make long-term monitoring feasible. Despite these benefits, ear-EEG systems still face challenges in terms of signal quality and spatial resolution, as the signals recorded from the ear are generally weaker than those from the scalp. However, ongoing advancements in sensor technology and signal processing are working towards overcoming these limitations, making ear-EEG a promising option for applications such as auditory processing research, sleep monitoring, and cognitive workload assessment.

Electroencephalography (EEG) is a widely used method for non-invasively measuring electrical activity in the brain, providing insight into neural processes by capturing the voltages generated by synchronized neural activity at the scalp. Originating from the

firing of brain neurons , EEG signals are primarily the result of postsynaptic potentials rather than action potentials, which create measurable voltage changes that propagate through the brain tissue, skull, and scalp to reach the electrodes on the skin's surface [63], [64].

EEG Signal Properties EEG signals are characterized by oscillations across different frequency bands related to specific cognitive and physiological states. These include delta (0.5-4 Hz), theta (4-7 Hz), alpha (7-13 Hz), beta (13-30 Hz), and gamma (30+ Hz) bands, each associated with distinct types of brain activity. For instance, alpha waves are often linked to relaxation, while beta waves indicate active cognitive processing. However, EEG signals are inherently low-amplitude (between 10-100 µV) and highly susceptible to noise, which can arise from various sources such as muscle movements, electrical interference, and impedance fluctuations at electrode sites [65], [66].

EEG Acquisition Techniques EEG acquisition typically employs electrodes placed on the scalp, arranged according to standardized positioning systems such as the 10-20 or 10-10 system, which enables consistent recording locations across individuals and studies [67]. Modern EEG systems may use either wet electrodes, requiring conductive gel for lower impedance, or dry electrodes, which allow for faster setup but with higher contact impedance that can reduce signal quality [68]–[70]. The selection of electrode type depends on the trade-off between signal quality and user comfort, especially in wearable or portable applications [71]. Advanced EEG systems, such as in-ear or ear-EEG devices, have also been developed to address specific application needs. These designs, which house electrodes in the ear canal or outer ear, offer a discrete and convenient form factor for monitoring brain activity, potentially suitable for long-term wearable use [72], [73]. Although they tend to capture a more localized signal than full-head systems, studies indicate that ear-EEG can still provide meaningful data for applications in sleep monitoring [47], [74], drowsiness [75], epilepsy [76]–[80] and auditory perception analysis [48], [49].

Challenges in EEG Acquisition One of the primary challenges in EEG acquisition is maintaining signal quality despite interference. Noise sources can include physiological artifacts (e.g., eye blinks and muscle contractions) and environmental noise, which can mask or distort the brain signals of interest. Strategies to mitigate these issues involve preprocessing techniques like artifact removal, filtering, and adaptive noise cancellation [81], [82]. Additionally, achieving a low-noise signal requires careful electrode placement and skin preparation to reduce impedance, which can be labor-intensive in

traditional systems. EEG acquisition faces additional constraints in embedded and low-power applications, which demand energy efficiency. Processing the signals in real-time with low latency while minimizing power consumption presents significant challenges, often necessitating the integration of specialized signal processing techniques and machine learning algorithms optimized for energy-efficient hardware [83], [84].

EEG Applications A typical use case for EEG analysis is the examination of brain responses to specific stimuli. In particular, EEG has been widely used in Auditory Steady State Response (ASSR), significantly enhancing the understanding of how the brain processes auditory stimuli [85]. ASSR refers to the brain's continuous response to auditory stimuli presented at a fixed frequency, making it a valuable tool for assessing auditory system function. This is especially relevant in clinical contexts, such as diagnosing hearing impairments, auditory pathway dysfunctions, or brainstem anomalies. EEG is particularly well-suited for capturing ASSRs due to its non-invasive nature and high temporal resolution, enabling real-time analysis of auditory information processing. ASSR is elicited using repetitive auditory stimuli, typically pure tones or modulated sounds presented at specific frequencies (e.g., 40 Hz, 80 Hz) [86], [49]. These stimuli evoke periodic electrical responses, captured through EEG, as neural oscillations phase-lock to the stimulus frequency, creating a consistent pattern of brainwave activity. By analyzing these signals, researchers can assess the integrity of the auditory pathways and the brain's sound-processing capabilities.

ASSR is particularly valuable for estimating hearing thresholds in populations where behavioral responses are difficult, such as infants or individuals with cognitive impairments [50]. ASSR provides an objective measurement that can be used to determine hearing sensitivity across different frequencies. Clinicians use EEG-based ASSR to evaluate the integrity of auditory pathways from the cochlea to the auditory cortex, identifying specific dysfunctions and providing detailed insights into auditory processing deficits [87]. It is also useful in assessing hearing aid and cochlear implant effectiveness, enabling clinicians to fine-tune devices based on individual responses. In the research, ASSR studies help explore neural mechanisms underlying auditory perception, speech processing, and sound localization by analyzing brain responses to modulated sounds.

EEG Analysis To analyze EEG signals in the ASSR domain, estimating the Power Spectral Density (PSD) plays a critical role in measuring the brain's response to auditory stimuli. Several methods are commonly used, each with advantages and limitations.

1. The Welch method: a well-established technique in spectral analysis, it is widely used in ASSR studies to quantify EEG responses. It divides the EEG signal into

overlapping segments, applies a window function to each segment, and averages the resulting periodograms. This process reduces variance, providing a smooth and accurate PSD estimate, making it ideal for noisy, non-stationary EEG signals. The Welch method offers a balanced trade-off between spectral resolution and noise reduction, making it particularly effective for steady-state responses, where precise frequency analysis is crucial.

2. Multitaper method: this technique enhances spectral concentration by using multiple orthogonal tapers, improving variance reduction compared to single-taper approaches [88]. However, the fixed design of the orthogonal tapers used in multitapering can be less flexible compared to the wide range of window functions available in the Welch method.

3. Burg method: this method utilizes autoregressive modeling to estimate the PSD, offering high-frequency resolution and better handling of short data segments [89]. However, the estimated autoregressive model makes it less reliable for EEG signals that often contain noise from sources like muscle activity or electrical interference.

4. Thomson's multitaper method: this method applies multiple orthogonal tapers to achieve an unbiased and consistent PSD estimate [90]. The choice of the number of tapers and bandwidth parameters in Thomson's multitaper method can be more challenging to tune compared to the straightforward segment length and overlap settings of the Welch method. Furthermore, the multiple tapers can sometimes lead to over-smoothing of the power spectrum, which may obscure subtle frequency components in the ASSR signal, particularly when precision in distinguishing peaks is necessary.

Welch Method The Welch method is particularly effective in analyzing steady-state responses, such as those elicited by periodic auditory stimuli in ASSR experiments. This approach is ideal for EEG signals, which are typically noisy and non-stationary. By dividing the EEG signal into overlapping segments, applying a window function to each segment, and averaging the resulting periodograms, the Welch method enhances the accuracy and reliability of the spectral estimates, providing a balanced trade-off between spectral resolution and noise reduction. The EEG data recorded during an ASSR experiment are divided into overlapping segments. This segmentation is critical to reduce noise and improve the estimate's stability. A window function, such as a Hamming or Hann window, is applied to each segment to reduce spectral leakage. This smooths the edges of the segment, ensuring that discontinuities do not introduce artifacts into the frequency analysis. For each windowed segment, the power spectrum is calculated using the Fast Fourier Transform (FFT). The periodogram for each segment represents the

distribution of power across different frequencies. The periodograms of all segments are averaged to obtain a final PSD estimate. This averaging reduces the variance of the spectral estimate, providing a more accurate representation of the brain's response to the auditory stimulus. The method provides good frequency resolution, making it well-suited for ASSR studies where precise frequency analysis is essential. Since ASSRs are typically elicited at specific modulation frequencies, the Welch method ensures that the power at these frequencies can be accurately measured. The Welch method, by averaging across multiple segments, helps to account for these changes, providing more stable spectral estimates compared to single-segment approaches. In ASSR studies, the Welch method is typically used to measure the EEG response at the frequency of the auditory stimulus and its harmonics. By analyzing the power at these specific frequencies, researchers can determine whether the brain has successfully phase-locked to the stimulus. This analysis helps assess auditory function and detect abnormalities in neural processing. For instance, in a typical 80 Hz ASSR study, the EEG response is analyzed using the Welch method to estimate the power at 80 Hz. A strong peak in the power spectrum at 80 Hz indicates that the brain is successfully entraining to the auditory stimulus, suggesting normal auditory processing. Conversely, reduced or absent power at this frequency may indicate auditory dysfunction.

While the Welch method is highly effective in analyzing EEG responses to auditory stimuli, there are some challenges to consider. These include the selection of appropriate window length and overlap, which can impact the frequency resolution and the ability to detect the ASSR. Additionally, high noise levels in EEG signals may require advanced artifact removal techniques before applying the Welch method to ensure accurate results.

2.2 Time-Series Biosignal Data Analysis

Time-series data, particularly biosignals such as ECG and EEG, play a crucial role in modern healthcare. The ability to process biosignals effectively has led to the development of advanced diagnostic tools, wearable health monitors, and personalized medicine. Despite these advancements, processing biosignals in real-time and on-device (i.e., edge computing) remains a significant challenge. Most SoA systems rely on cloud-based processing, where raw or minimally processed data is transmitted to a central server for analysis, which introduces latency, data privacy concerns, and high energy consumption. This chapter discusses the SoA approaches in time-series biosignal data processing and highlights the limitations and challenges associated with edge computing for real-time biosignal analysis.

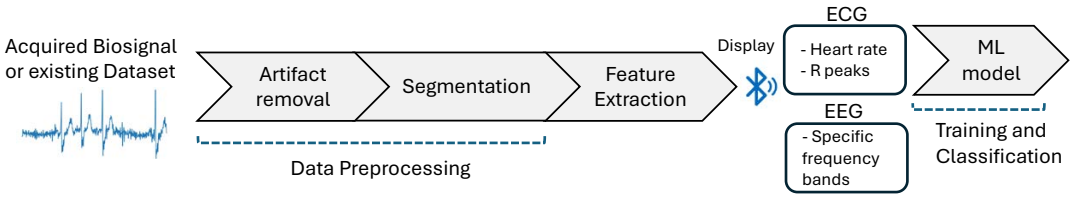


Figure 2.1: Design methodology on wearable devices applied in this dissertation.

2.2.1 Biosignal Processing: Methodology

Time-series biosignal data are dynamic, often non-stationary, and exhibit noise due to physiological and environmental factors. Consequently, processing such data involves various pre-processing, feature extraction, and classification techniques. Historically, biosignal processing has been dominated by techniques focused on analyzing the frequency domain. In recent years, ML methods have gained momentum, with neural network architectures achieving SoA results in classification and prediction.

The process depicted in Figure 2.1, follows a pipeline consisting of:

1. Preprocessing: Removal of noise and artifacts using filters or preprocessing algorithms, such as the Pan-Tompkins algorithm. This step also involves *Segmentation*, which divides the time-series data into manageable windows or segments.

2. Feature Extraction: Deriving useful metrics, such as heart rate, PQRST wave characteristics in ECG, or specific frequency bands in EEG.

3. Training and Classification: Identifying patterns using ML algorithms or statistical methods.

This dissertation explores the benefits of implementing algorithms on edge parallel ultra-low-power (PULP) architectures, leveraging parallel computing inside a general-purpose multi-core accelerator (called *cluster*) along with extensive software and hardware optimizations. The goal is to maximize energy efficiency (i.e., prolong battery life) while ensuring high accuracy and low latency. The target embedded architectures are resource-constrained to enhance compactness and minimize energy consumption. A methodology must be applied to optimize resource utilization to fully exploit these architectures. Key considerations include achieving a high degree of parallelism in many-core implementations, optimizing execution through software or hardware enhancements, and effectively managing the available memory on the SoC. Since the cluster processing elements are general-purpose independent cores and execute separate instruction flows, the programming interface supports the single-program multiple-data (SPMD) paradigm.

The PULP HAL [91] provides two main concepts: *core identifiers* and *barriers*. The core identifier is a fundamental mechanism to split the workload among multiple execution flows (*parallel orchestration*). For instance, programmers can employ loop-level parallelism using core identifiers in the control expressions (i.e., initialization, condition check, and increment). A barrier is a HAL function that stops a core until all other cores arrive at the same execution point. Barriers are synchronization points that guarantee data consistency between adjacent code regions (before and after the barrier). The *event unit* [92] is a dedicated hardware component providing low-overhead support for barriers and enabling power-saving policies when cores are waiting.

To effectively implement an application on an embedded device, the initial step involves conducting a thorough feasibility study on both the application and the potential target architectures. This process can be achieved by breaking the application down into smaller components (kernels), analyzing each kernel individually along with the available resources on the architectures, and identifying which parts can be optimized and any potential challenges that may arise. Another crucial factor to consider is the optimal representation of the data. This analysis helps determine whether an integer implementation is suitable for our processing needs or if floating-point (FP) operations are necessary. In cases where the architecture lacks a Floating Point Unit (FPU), we must assess the impact of software-based FP simulations on performance, particularly for kernels that involve intensive FP computations. With these considerations in mind, we can make an informed decision regarding the architecture to target for our processing chain.

A recommended practice is to first implement the entire processing chain using a scripting tool (such as MATLAB) to create a golden model. This allows for a thorough verification of the theoretical accuracy of our implementation (including numerical results and overall precision) and enables comparison with the outcomes from the embedded implementation. The subsequent step involves transitioning to the actual implementation, where we examine the results of each kernel that makes up the final application in C code. During this phase, we should consider all potential optimizations and structure the code accordingly. This approach facilitates easier adaptation of the code to the targeted architecture. Once we have a stable version, we can begin the porting process to the target architecture, focusing on various optimizations to enhance execution speed.

Furthermore, for initial setup and algorithm tuning, the STM32Cube IDE provides a GUI that allows users to program STM32 MCUs to execute the code, also exploiting ARM CMSIS [93] kernels for the comparison of the results.

Techniques for Biosignal Pre-Processing and Inference This subsection explores the most common techniques in biosignal processing and inference. Traditional methods of biosignal processing have long been the foundation of biosignal analysis, enabling the extraction of meaningful features from raw data. Among these techniques, the Fourier Transform (FT) and Wavelet Transform (WT) are frequently used to analyze the spectral content of biosignals.

In ECG analysis, Fast Fourier Transform (FFT) is typically used to decompose heart rate signals into their constituent frequency components, which can provide insights into heart rhythm abnormalities [53] [94]. Wavelet-based methods have also been extensively applied to ECG signals for arrhythmia detection. Choi [95] highlighted how multi-level wavelet decomposition can isolate ECG signal features indicative of specific heart conditions. Zidelmal [96] applied DWT for ECG signal denoising, achieving superior results in noise suppression compared to traditional frequency-domain methods. Wavelet-based techniques also offer flexibility in handling non-stationary signals like ECGs, where the frequency content changes over time. For instance, Addison [97] showed how DWT can help identify different types of heartbeats by analyzing the ECG signal across multiple frequency scales. Additionally, ECG processing often involves multiple stages, including noise reduction. Techniques like band-pass filtering are applied to remove baseline wander and powerline interference, which can obscure critical ECG features. Arif et al. [98] discuss how filtering enhances the accuracy of subsequent ECG analysis.

To detect a person's alertness/drowsiness, Chen et al. [99] proposed a system that includes the decomposition of EEG data into wavelet frequency sub-bands and FFT-based spectral analysis for comparison. Then, they apply a single-hidden layer of feed-forward neural networks for the recognition. In another study [100], the authors focused on decoding the subjective perception of task difficulty to enhance operator performance by automatically optimizing task difficulty levels. The study used a protocol with two tasks, flying and visual recognition, to induce different difficulty levels and analyzed EEG signals to distinguish between compound cognitive states.

More recently, ML models have emerged as powerful tools for biosignal processing. Machine Learning models, such as CNNs, have demonstrated the ability to automatically learn features from raw time-series data, significantly reducing the need for manual feature engineering. For instance, the authors of [101] applied a 1D-CNN architecture for automatic ECG classification, achieving SoA performance on the MIT-BIH Arrhythmia dataset. Other ML models, such as Recurrent Neural Networks (RNNs), particularly LSTMs, have been effective in modeling the temporal dependencies inherent in biosignal data. Liu et al. [102] leveraged LSTMs for EEG-based emotion recognition, showcasing the potential of ML for decoding complex biosignals in real-time. Zhang et al. [103]

introduced the adaptive exponential smoothing (AES) technique for smoothing ECG and EEG signals in combination with a bounded support vector machine (BSVM) to classify different levels of operator mental workload.

Despite their effectiveness, the ML approaches require significant computational resources, making them unsuitable for real-time analysis on resource-constrained edge devices. Although these models achieve high accuracy, their deployment on edge devices remains limited due to large model sizes, high computational power demands, and substantial memory requirements. As a result, most current approaches still rely on cloud-based servers for model inference, with biosignal data being offloaded from the edge to the cloud.

2.2.2 Challenges in Edge Computing

Edge computing has emerged as a potential solution to address the limitations of cloud-based processing by moving computation closer to the data source. In an edge computing framework, biosignals are processed directly on the device, offering several advantages, including reduced latency, improved data privacy, and lower energy consumption due to reduced transmission of data. However, edge computing for biosignal processing has not yet been widely adopted. There are several technical and computational challenges associated with deploying complex biosignal processing algorithms on edge devices, which are typically constrained by limited processing power, memory, and energy capacity.

Resource Constraints Edge devices, such as wearable systems or portable health monitors, are often battery-operated and have limited computational resources. Running ML models on such devices requires significant model compression or hardware accelerators, which is still an evolving area of research. Sarkar et al. [104] showed that pruning and quantization techniques can reduce the size of neural networks, but there is often a trade-off between accuracy and computational efficiency.

Energy Consumption Most ML models for biosignal processing are computationally intensive. For edge devices, managing the energy consumption of these models is crucial to prolong battery life. The energy-efficient CNN-based approach for real-time ECG classification on edge devices proposed in [105] reduces energy consumption by 3x compared to traditional methods.

Real-Time Processing : Time-series biosignals, particularly in critical health applications, require real-time processing to provide immediate feedback (e.g., detecting abnormal heart rhythms in ECG). Attaran et al. [106] demonstrated a real-time system using edge computing, but the proposed method only detects stress conditions.

2.2.3 ECG and EEG analysis on the edge: an Overview

While the literature on edge computing for biosignal processing is still nascent, some notable works are paving the way.

Tsai et al. [107] developed edge-based techniques for QRS complex detection using a low-power ARM Cortex M4 processor. Their system, while the accuracy is not reported, demonstrated the feasibility of on-device processing for biosignals in resource-constrained environments. De Giovanni et al. [52] proposed a software-based methodology that implements a Bayesian filter, normalization, and a clustering technique to optimize the R peak detection for low-power platforms. However, the authors do not consider the aspects related to real-time signal acquisition due to the use of an existing system (BIOPAC) that requires a 9 V battery and is not energy efficient. Nguyen et al. [108] focused on EEG monitoring from wearable sensors. They applied the Welch method for the power spectral analysis as a preprocessing stage. To detect drowsiness, the authors employed two NN models (Multi-Layer Perceptron and CNN) and apply quantization techniques to reduce the computational complexity at the expense of negligible reduction in accuracy.

A convolutional autoencoder model is proposed in [109] for denoising single-lead ECG signals, optimized for low-power edge devices. It aims to reduce noise and motion artifacts in ECG data, improving the accuracy of detecting conditions like atrial fibrillation. Betti et al. [110] developed a system of wearable physiological sensors, including ECG and EEG, to capture human stress and evaluate if the detected changes in these physiological signals correlate with salivary cortisol levels, a reliable stress biomarker. A support vector machine (SVM) classification algorithm was used for statistical analysis. In [111], the authors propose a robust seizure detection method for wearable platforms, tested on the CHB-MIT Scalp EEG database, achieving a sensitivity of 0.966, specificity of 0.925, and a 34.7% reduction in false alarms. It demonstrates the system's capability to function for up to 40.87 hours on a single battery charge, showing potential for real-time use.

Zanetti et al. [112] presented a methodology based on the Random Forest classifier to implement cognitive workload monitoring (CWM) on resource-constrained wearable devices using four peripheral EEG channels. It includes a data processing strategy such

as artifact removal along with a band-pass filter, as well as feature extraction for processing data in small batches to reduce memory requirements. The challenge of balancing performance and power consumption in wearable EEG applications such as epileptic seizure detection by introducing a novel Knowledge Distillation (KD) methodology is presented in [113]. The goal is to reduce the number of EEG channels, and thus the computational and memory requirements, without compromising performance. The method involves training a high-performing model (teacher) on data from all channels and then training a simpler model (student) on a reduced set of channels using the teacher's soft labels. Risso et al. [57] uses a TCN-based solution to classify the ECG arrhythmia and a Neural Architecture Search (NAS) methodology to optimize the network parameters.

The shift toward processing biosignals on the Edge offers several promising opportunities. The research needs to address the issues such as model compression and optimization. Chapter 3 will delve into the acquisition and preprocessing of the ECG signals, as well as the application of ML, which are central to the focus of this work.

2.3 Microcontrollers of Interest

The increasing demand for portable, small-sized devices has led to significant advancements in healthcare and commercial applications. Microcontroller Units (MCUs) have been pivotal in this shift, enabling compact devices with capabilities such as sensor integration and real-time data processing. However, integrating complex data analysis on embedded platforms poses challenges, requiring a balance between computational efficiency and energy consumption. This dissertation focuses on optimizing biosignal processing pipelines and machine learning (ML) techniques for deployment on MCUs. The primary microcontrollers explored include the STM32F4 and the PULP-based GAP8 and GAP9, which represent two distinct Instruction Set Architectures (ISAs): ARM for the STM32F4 and RISC-V for the GAP series.

The **STM32F4** microcontroller, developed by STMicroelectronics, features an ARM Cortex-M4 core [114]. It is engineered to optimize computational performance while maintaining low power consumption, making it suitable for general-purpose computing. A detailed block diagram is provided in Figure 2.2. The MCU features both data and instruction caches, which significantly improve performance by reducing the time required to fetch instructions and load data into registers. However, this performance gain comes at the cost of increased energy consumption.

The ARM Cortex-M4 processor in this MCU supports frequencies up to 84 MHz with a power consumption of 20 mW. Additionally, the STM32F4 offers various peripherals,

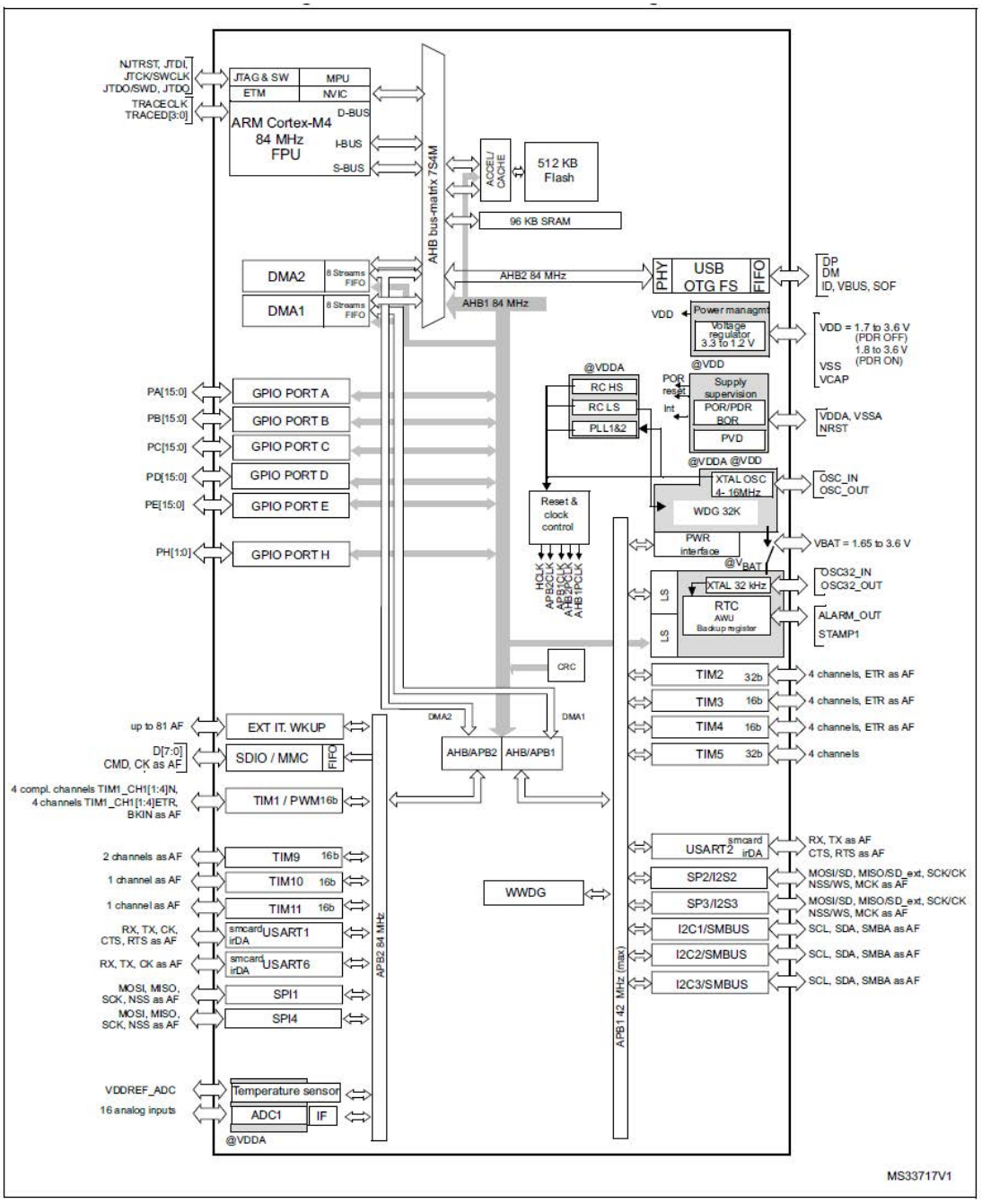


Figure 2.2: Block diagram of the STM32 MCU (sub-family STM32F401xD/xE). Image source: [115].

including a Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I^2C), and Analog-to-Digital Converters (ADCs) for sensor data acquisition.

Unlike the ARM ISA, the RISC-V ISA is open-source, allowing for the development of various public extensions, such as those for Digital Signal Processing (DSP). This flexibility has enabled the creation of new architectural designs based on the RISC-V ISA for general-purpose MCUs, incorporating specialized components to accelerate tasks like deep learning. Advances in this area include the integration of specialized accelerators and hierarchical memory systems that exploit data regularity.

A notable example is the Parallel Ultra-Low Power (**PULP**) computing platform, which leverages near-threshold computing to achieve high energy efficiency and utilizes parallelism to mitigate performance degradation at low voltages. The PULP architecture focuses on optimizing the RISC-V ISA for DSP and Deep Neural Networks (DNNs), heterogeneous parallel acceleration where different compute units are assigned to distinct tasks, and manually controlled memory hierarchies. Key extensions to the ISA include Single Instruction Multiple Data (SIMD) Multiply-and-Accumulate (MAC) operations—central to DNN computations—as well as load/store instructions with post-increment, which streamline memory operations by automatically updating indices. Most current implementations of the PULP paradigm utilize a SoA single-core MCU, known as *fabric controller*, which includes a standard set of peripherals. It offloads computation-intensive tasks to a programmable parallel accelerator referred to as *cluster*, which contains multiple cores and operates within its own voltage and frequency domain.

A commercial embodiment of the PULP architecture is GreenWaves Technologies' (**GWT**) **GAP8** [116], depicted in Figure 2.3. The GAP8 features nine RISC-V cores (one I/O core and an 8-core cluster), making it one of the most advanced MCUs with dedicated optimizations for ML workloads. The GAP8 cluster comprises eight RI5CY cores with a four-stage in-order single-issue pipeline [118], utilizing the RISC-V `RV32IMCXpulpV2` instruction set architecture (ISA). The `XpulpV2` extension is designed specifically for domain-specific applications, optimizing DSP performance through features such as hardware loops, post-modified load/store access, and SIMD instructions supporting vector operands as small as 8 bits. All cores in the cluster share a unified first-level memory, consisting of a 64 kB multi-banked L1 Tightly-Coupled Data Memory (TCDM), which is accessible through a high-bandwidth logarithmic interconnect with single-cycle latency [118]. Data transfers between the L1 TCDM and the second-level 512 kB L2 memory are managed by a cluster DMA [34], providing bandwidth up to 2 GB/s and a latency of 80 ns at peak frequency. The L2 memory acts as a scratchpad and resides in the SoC domain. Additionally, an autonomous I/O subsystem, the I/O

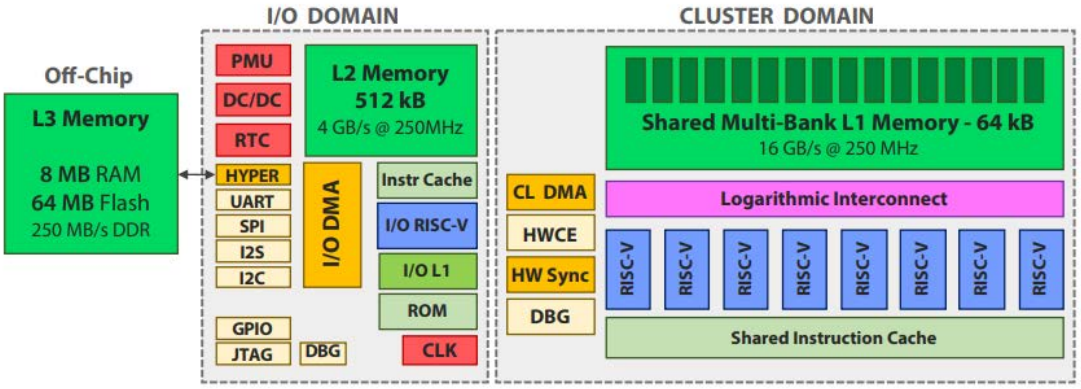


Figure 2.3: Block diagram of the GWT GAP8 microcontroller. Image source: [117].

DMA [119], facilitates data exchange between the L2 memory and external interfaces, including the L3 HyperRAM/HyperFlash module available on the board. Through the HyperBus interface, with a bandwidth of up to 200 MB/s, external L3 memory can be connected, providing up to 64 MB of Flash for read-only data and 8–16 MB of DRAM for volatile data. **GWT GAP9** [120] is the more recent iteration of GWT GAP8, with analogous architectural principles. The chip is equipped with an accelerator (*hardware convolution engine*) designed to optimize both energy efficiency and bandwidth usage. GAP9 sets the standard for low-power processors, having achieved the top ranking in both latency and energy consumption in the MLPerf Tiny v1.0 benchmarks [121].

2.4 Embedding Networks: Compression & Deployment

ML models are increasingly embedded into the digital frameworks of the consumer and healthcare sectors. ML has taken a leading role in addressing various computational challenges for running ML inference locally on edge devices. However, edge platforms are limited by strict constraints on memory and power usage, and handling time-series data adds further complexities, especially concerning computation delays. While some modern edge computing devices are equipped with specialized hardware accelerators to boost performance, efficiently leveraging these resources demands a deep understanding of hardware-specific programming. Research at the algorithmic and software level focuses extensively on compression techniques to reduce the resource demands of training and inference. This thesis examines strategies designed to lower the memory and computational requirements of ML model inference in resource-limited settings.

2.4.1 Quantization

In this dissertation, the most significant network compression technique explored is quantization. Other effective methods to reduce a model's computational burden and memory footprint are **pruning** [122] and **vector compression**. A detailed review of deep network compression techniques falls outside the scope of this thesis; instead, the focus will be specifically on quantization.

Quantization simplifies models by approximating floating-point values (real numbers) with integer values of reduced bit-width, enabling computations with lower precision [122]–[124]. Typically, deep learning models are trained using formats like `float32` or `float16` for parameters and activations. However, utilizing smaller bit-widths can greatly improve memory efficiency and inference speed with only minimal accuracy loss. Model developers can also assign different numerical precision to the parameters and activations of each layer, thus enabling mixed-precision models that strike a balance between computational efficiency and model accuracy. Quantization tends to be most effective when applied during training (Quantization-Aware Training, or QAT). However, Post-Training Quantization (PTQ) is another commonly used method.

In the context of this thesis, the quantization is linear and uniform across layers. This means that the elements t_i of the tensors $\mathbf{t}$ (such as weights $\mathbf{W}$, inputs $\mathbf{x}$, hidden activations $\mathbf{a}$, and outputs $\mathbf{y}$) with dynamic range $[\alpha_\mathbf{t}, \beta_\mathbf{t})$ are mapped to N-bit integers $\widehat{\mathbf{t}}$ using the following transformation:

$$t_i = \alpha_\mathbf{t} + \varepsilon_\mathbf{t}\widehat{t}_i, \qquad \text{with} \qquad \widehat{t}_i \in \mathbb{Z} \tag{2.1}$$

With

$$\varepsilon_\mathbf{t} \triangleq \frac{\beta_\mathbf{t} - \alpha_\mathbf{t}}{2^N - 1} \tag{2.2}$$

Here, $\varepsilon_\mathbf{t}$, referred to as the quantum, represents the smallest difference between values in the quantized tensor. In general, a layer consists of a sequence of three operators: a linear operation, an optional batch normalization (BN), and a non-linear activation function. The latter is merged with a quantization step. This is because the activation function already processes the pre-activation output. Instead of applying activation and then separately quantizing the result, both steps can be combined into a single function to reduce computational overhead [123]–[125].

Without loss of generality, we can assume $\alpha_\mathbf{x} = \alpha_\mathbf{a} = \alpha_\mathbf{y} = 0$ for all the inputs of linear operations and outputs of Quantization/Activations operators, but not for weights. If the original activation function is a Rectified Linear Unit (ReLU), the activations automatically satisfy the assumptions; otherwise, simple transformations can enforce

them. All operators can be mapped in the integer domain by exploiting Equation 2.1. For linear layers:

$$\varphi = \sum_n \mathbf{W}_{mn}\mathbf{x}_n \qquad \longrightarrow \qquad \widehat{\varphi} = \sum_n \widehat{\mathbf{W}}_{mn}\widehat{\mathbf{x}}_n; \tag{2.3}$$

whereas for BNs:

$$\varphi' = \kappa \cdot \varphi + \lambda \qquad \longrightarrow \qquad \widehat{\varphi}' = \widehat{\kappa}\widehat{\varphi} + \widehat{\lambda}. \tag{2.4}$$

For inference, the BN parameters can be merged-pair-wise:

$$\kappa \triangleq \frac{\gamma}{\sigma}, \qquad \lambda \triangleq \beta - \mu\frac{\gamma}{\sigma}. \tag{2.5}$$

The dot-product operation in Equation 2.3 induces a shrinking of the quantum, leading to

$$\varepsilon_\varphi = \varepsilon_\mathbf{W}\varepsilon_\mathbf{x} \ll \varepsilon_\mathbf{W}, \varepsilon_\mathbf{x} \tag{2.6}$$

since, typically, $\varepsilon_\mathbf{W}, \varepsilon_\mathbf{x} \ll 1$. The integer output of the linear operator $\widehat{\varphi}$ (e.g., 32 bits) requires higher precision than its inputs and weights during the accumulation, then it is re-quantized. An analogous effect happens in the BN layers for the output $\widehat{\varphi}'$. The final quantization/activation operator introduces non-linearity and compresses the result into a lower bit-width using the following equation:

$$\widehat{\mathbf{y}} = m\widehat{\varphi}' \gg d, \qquad \text{with} \qquad m \triangleq \left\lfloor \frac{\varepsilon_{\widehat{\varphi}'}}{\varepsilon_\mathbf{y}} 2^d \right\rfloor \tag{2.7}$$

where $\gg$ denotes a right shift. The integer d is set to ensure that $\varepsilon_{\widehat{\varphi}}/\varepsilon_\mathbf{y}$ is represented with enough precision for accurate computations. This process is similarly applied when multiple branches of a network, each with different quantization scales, need to be merged.

2.4.2 Frameworks and tools for embedded neural inference

The tools and frameworks used for deploying and running neural networks on edge computing platforms are essential for SoA ML applications in both research and industry. A major benefit of the leading deployment frameworks is their ability to support and partially automate various compression techniques, which are vital for adapting neural networks to the limitations of edge computing platforms [126].

In the results presented in this thesis, activations and weights are quantized to 8-bit precision, while accumulators and BN parameters retain 32-bit precision. The quantization is performed using **NNTool** [127], a neural network deployment tool within the GAP SDK. This tool plays a crucial role in facilitating the deployment of ML

models on ultra-low-power, RISC-V-based processors optimized for edge computing and embedded applications. NNTool enables developers to import pre-trained models from widely used ML frameworks such as TensorFlow or ONNX, as described as follows. After importing, the tool optimizes the model for efficient execution on the processors, with a focus on minimizing memory usage and enhancing computational performance. NNTool offers support for post-training quantization, enabling the conversion of model weights and activations to lower-precision formats such as int8. This significantly reduces the memory and computational requirements, making it suitable for deployment on resource-constrained devices like the GAP processors. The tool supports techniques like layer fusion, combining multiple layers into a single computational unit to minimize data movement and improve performance. It also handles tiling, which divides large neural network layers into smaller chunks to fit within the limited on-chip memory, such as the L1 cache, while maintaining efficient data processing.

Once the neural network model is optimized, NNTool generates highly efficient C code that is tailored to the GAP processor's architecture. This includes managing memory hierarchies, scheduling operations, and optimizing data transfers between the on-chip and off-chip memory. NNTool is integrated into the GAP SDK, which is part of the broader PULP ecosystem. This allows developers to leverage GAP's parallel processing capabilities and specialized instructions for digital signal processing (DSP) and machine learning tasks, such as SIMD operations and hardware loops.

TensorFlow Lite (**TFLite**)[128] is a streamlined tool designed for running inference on edge devices, built on the widely-used TensorFlow framework[129]–[131]. TFLite supports post-training quantization, including half-precision floating point (`float16`) and `int8` data types. It is also compatible with quantization-aware training and pruning techniques available in TensorFlow and Keras [132], allowing models developed with these upstream tools to be imported into TensorFlow Lite Micro [133], a runtime framework optimized for ML inference on MCUs.

Open Neural Network Exchange (**ONNX**) [134], [135] is an open-source, machine-independent format designed for ML models. Its primary goal is to share models across various frameworks and tools while also accommodating the target hardware, including mobile and edge devices. ONNX facilitates quantization to the `int8` format during both training and inference time for convolutional, fully-connected, and activation layers. This feature enables models to be executed in a framework different from the one used for training, allowing greater flexibility in combining SoA frameworks when implementing an ML pipeline or product.

Neural Minimizer for PyTorch (**NeMO**) [125], [136] is an open-source Python library designed for quantizing neural networks built with PyTorch [137], [138]. NeMO

is specifically aimed at deployment on ultra-low power computing devices that have strict memory limitations, with a particular emphasis on PULP-based MCUs [139]. The library incorporates the Parameterized Clipping Activation (PACT) quantization technique [140] along with other methods, allowing users to configure the quantization bit-width for activations, weights, and BN parameters, as well as facilitate BN folding. NeMO supports mixed-precision quantization and provides a semi-automated approach for precision relaxation.

Deployment Oriented to Memory (**DORy**) [123], [124], [141] is an automated tool designed for deploying deep learning models on resource-constrained embedded platforms, typically those with an on-chip SRAM memory budget of $\leq$ 1 MB. DORy addresses memory limitations by treating tiling as a Constraint Programming (CP) problem, focusing on maximizing L1 memory utilization while adhering to the topological constraints of each model layer. The tool generates C code to manage both off-chip and on-chip data transfers and computation phases; DORy is compatible with the GWT Virtual System-on-Chip (GVSoC) [142], which simulates RISC-V processors for PULP-based platforms such as GWT' GAP8 [116], [120] and GAP9.

STM32 CubeAI [143] is an extension of the STM32 CubeMX [144] code generation tool. It features a graphical user interface (GUI) that enables users to configure STM32 microcontrollers for executing deep learning model inference. The tool supports TFLite and ONNX models and includes functionality for post-training compression. The generated code provides APIs that facilitate the integration of multiple models within a single codebase while optimizing inference performance through the use of ARM CMSIS [93] kernels.

Quantization Library (**QuantLib**) [145] is an open-source library designed for model quantization. It is also an integral part of Quantization Laboratory (QuantLab) [146], [147], which includes additional tools for managing large-scale machine learning tasks. This includes support for multi-GPU acceleration of neural network training in combination with Horovod [148]. Although QuantLib and QuantLab are not used in this work, they are worth mentioning as they have become the primary quantization tools within the PULP Platform project, with applications in image recognition [149] and epilepsy detection [150].

Chapter 3

ECG Applications - Efficient Transforms and Heart Rate Detection Algorithm

Chapter 3 and Chapter 4 introduce the original contributions of this thesis in the area of ECG low-power signal processing and classification. These contributions include optimizing and implementing efficient methods, such as optimized transform algorithms that can be useful in the biosignal analysis pipeline presented in Section 2.2. Additionally, the Pan-Tompkins algorithm, a widely used method for preprocessing and feature extraction, is discussed.

3.1 Signal Description and Acquisition

Electrocardiography is a method used to record the heart's electrical activity over time. The heart's rhythmic contractions generate the ECG signal and provide crucial information about the heart's condition. Typically, an ECG consists of several distinct waves, such as the P-wave, QRS complex, and T-wave, which correspond to different phases of the heart's electrical cycle as depicted in Figure 3.1. Roughly 160 ms after the P wave onset, the right and left ventricles depolarize, resulting in around 80 ms of QRS complex. The end of the QRS complex corresponds to the end of the repolarization of the atria [151]. ECG signals are acquired using electrodes placed on the skin, which capture the small electrical changes produced by the heart's activity. These electrodes are positioned in specific locations on the chest and limbs to provide a complete picture of the heart's function from different angles. The electrical signals are amplified, filtered, and digitized for further analysis. Modern wearable devices and

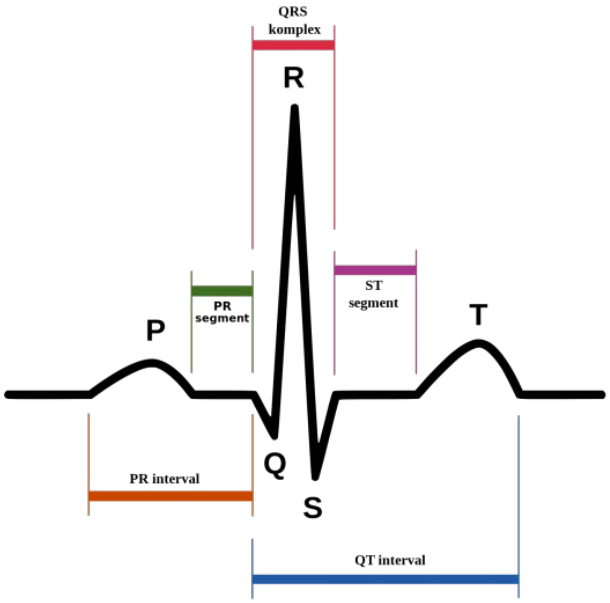


Figure 3.1: The PQRST wave represents the complete electrical cycle of a single heartbeat. Image source: [152].

low-power embedded systems enable continuous ECG monitoring, providing real-time insights into heart health without bulky, hospital-grade equipment. However, noise from muscle activity, motion artifacts, and electromagnetic interference can affect the signal quality, requiring advanced filtering and signal processing techniques to ensure accurate interpretation.

3.2 Efficient Transforms

As described in Section 2.2.1, several techniques exist for the filtering and classification of the ECG signal [153]. This part proposes an efficient parallel design of the widely used short-time Fourier transform (STFT) and discrete wavelet transform (DWT) targeting ultra-low-power devices [154]. The key performance challenges are related to fine-grained synchronization and banking conflicts in shared memory. Modern end-node devices must support computationally intensive workloads at a limited power budget. Parallel ultra-low-power architectures are a promising target for this scenario, and the availability of highly optimized software libraries is crucial to exploit parallelism and reduce software development costs.

3.2.1 Related Work

The ultra-low-power (ULP) parallel computing platforms based on the RISC-V instruction set architecture (ISA) have proved to be an effective solution for Internet of Things (IoT) end nodes as an open alternative to proprietary ISA MCUs (e.g., ARM Cortex-M4) [155], [156]. The parallel ultra-low-power platform (PULP) [157] is an open-source hardware project aiming to provide a RISC-V programmable architecture with the primary goal of meeting the computational requirements of IoT applications within a power envelope of 10 mW. The recent embodiments of this architecture include a control core dedicated to I/O and system management coupled with a cluster of cores sharing a tightly-coupled data memory (TCDM). The PULP approach enables operating the cluster at the energy-optimal operating voltage (i.e., near-threshold [158]) while achieving high computational throughput thanks to parallel execution [159].

The availability of efficient shared-memory parallel software libraries for fundamental algorithmic kernels is a key enabler to fully exploit ULP platforms and reduce software costs. For traditional single-core MCUs, CMSIS-DSP [160] is a hardware abstraction layer (HAL) targeting ARM Cortex-M cores, which provides a set of optimized digital signal processing (DSP) kernels. A key challenge in developing similar libraries for PULP is to achieve a good parallel speed-up, which is essential for obtaining high energy efficiency.

DSP applications make pervasive use of the 1-D floating-point variants of these algorithms: They enable the extraction of relevant features on time and frequency domains serving as preprocessing stages for ML methods. In real-life use cases, DWT is used in [161] to extract features from physiological data in a pattern recognition application that relies on an embeddable support vector machine (SVM). STFT is used in [162] for structural anomaly detection, providing the time-frequency analysis of current consumption, voltage, and vibrations of industrial equipment. Since machine learning models adopted for near-sensor processing, such as multi-layer perceptron (MLP) [163] and SVM [18], are amenable to lightweight designs executing in a few thousand cycles, optimizing the preprocessing stages based on DWT and STFT is crucial to improve performance and energy efficiency. The main block of STFT is the fast Fourier transform (FFT) algorithm. FFTW [164] is the most widespread FFT implementation, and it is widely used in scientific computing. However, this library has a complex design, and embedded system designers do not commonly adopt it for performance reasons. In most cases, lightweight FFT libraries are not portable and do not provide optimized parallel support. For instance, Kiss FFT [165] is parallelized using OpenMP directives, but it is not optimized. The GNU scientific library (GSL) [166] provides a parametric

implementation of the DWT algorithm even though it does not provide any support for code parallelization.

The following section describes an algorithm design for FFT and DWT focused on performance optimization on ULP IoT end nodes. This goal requires a fine-grain analysis to maximize the instructions per cycle (IPC) for each processing thread. Second, this chapter provides an experimental assessment of an 8-core PULP cluster with 4 floating-point units (FPUs), analyzing the impact of the key design optimizations. Finally, a comparison is conducted between the Cortex-M4 platform and alternative libraries, such as GSL and Kiss FFT, to evaluate their relative performance and efficiency.

3.2.2 Algorithms Design

STFT STFT is illustrated in Figure 3.2. Input buffer size, number of data samples, and overlap size are configurable parameters. The most relevant kernel of the STFT is FFT calculation, based on the mixed-radix variant of the decimation-in-frequency Cooley–Tukey algorithm, a solution also used by Kiss FFT and CMSIS. This class of algorithms recursively breaks down a transform on input with size $N = r \times m$ into r smaller transforms of size m. Each recursive call is called a *stage*, and transforms of size r are generally referred to as *butterflies*. Our design adopts a *mixed-2-8* variant that applies a radix-8 FFT when the size of the input is a power of eight; otherwise, it performs one or two preliminary radix-2 stages.

The first FFT stage ($N/2$ butterflies) can be equally split among the available cores. Each of the following stages includes $2^s \times m$ transform step, where s is the zero-based stage index. The butterflies inside a transform step are equally split among the cores if they are enough to guarantee workload balancing – i.e., m/r must be greater or equal to the number of available cores. Otherwise, transform steps are partitioned into disjoint sets that are distributed among the cores. This approach guarantees workload balancing in all cases, and it also minimizes the overhead of parallel orchestration since workload distribution is always associated with a single loop. Each stage requires a single barrier at the end to guarantee data consistency for the next one. Decimation-in-frequency algorithms require output reordering as a final stage. Index remapping is provided by a pre-computed look-up table so that this task can be equally split among the cores. However, access to the look-up table and subsequent swap operations are highly memory-bound and cause TCDM stalls. To hide this latency, we applied *loop unrolling* to the reordering outer loop. Instead of processing one element per iteration, loop unrolling executes multiple reorder operations in a single iteration. This reduces the number of loop overhead instructions (e.g., increment). While one set of swap operations

waits for memory access, the processor can execute other operations in parallel. This overlaps computation with memory accesses, reducing stall time. Unrolling decreases the number of conditional checks and jumps, improving efficiency on deeply pipelined architectures. Accessing multiple contiguous elements in one iteration improves spatial locality, reducing cache misses and minimizing memory stalls.

DWT DWT is a time-frequency analysis technique relying on a pair of recursive convolutions, which decompose the original signals, extracting its low and high-frequency contents, referred to as the time domain [167]. As depicted in Figure 3.2, for a given input signal of length N, DWT applies the two convolutions followed by dyadic downsampling, producing two output vectors that contain namely approximation (i.e., cA) and detail (i.e., cD) coefficients. The first convolution applies a low-pass filter g, and the second a high-pass filter h related to g in a quadrature relationship as they derive from the same mother wavelet. The filter coefficients are pre-computed and passed to the algorithm as input parameters. Initial input data are also provided for the first level, while approximation coefficients represent the input of the next level. The filter size (FS) is an even number equal to or greater than two, and the case of $FS = 2$ is also referred to as Haar wavelet.

In the design, three main optimizations are applied. First, the implementation of a strided convolution routine that performs convolution and downsampling of both filters in a single step, reducing the total number of instructions required to compute cA and cD. Second, a coding variant for the Haar wavelet, which does not require border paddings and fully unrolls the last loop to compensate for the small filter size that induces memory access stalls. Third, the algorithm copies cA values into the input data structure at the end of the second loop, reducing the total memory footprint for data allocation (e.g., GSL requires an additional memory buffer).

The loop-level parallelization is applied on the second level. In the general code variant, this level is further split into three parts, corresponding to the border and inner data. The size of the iteration space for the border computation is equal to $FS - 1$; consequently, an ideal workload balancing of this code is impracticable when there are more numerous cores than iterations. Synchronization barriers are required after applying the filters and after preparing input data for the next level.

3.2.3 Results

The methods are evaluated using a cycle-accurate PULP emulator on a Xilinx UltraScale+ VCU118 FPGA board. The setup included 8 cores and 4 FPUs, chosen for

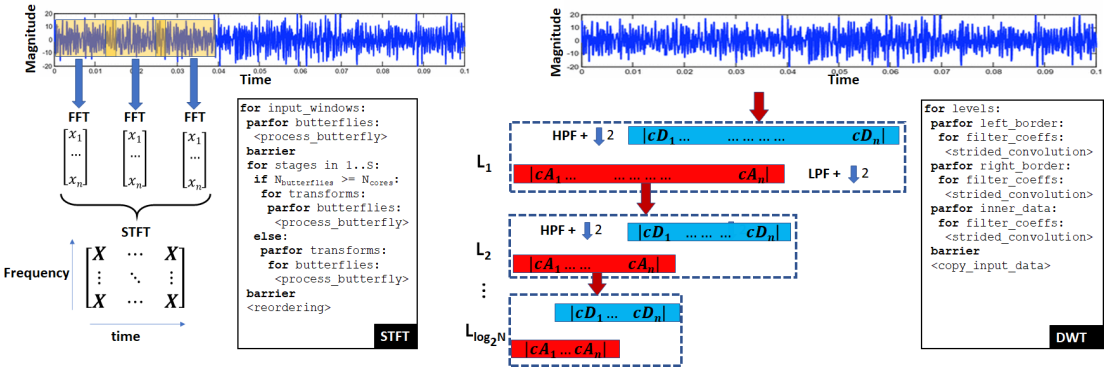


Figure 3.2: Structural and working flow diagrams of STFT and DWT.

Table 3.1: Instructions, hardware stalls, synchronizations occurrences, and throughput (execution on 8 cores, 2048 data samples).

	STFT	**DWT**		
		FS=2	*FS=8*	*FS=16*
Instructions (per core)	20677	3090	12629	23848
TCDM stalls [cycles]	1034	58	610	1539
I-cache stalls [cycles]	504	127	76	189
FPU stalls [cycles]	4992	517	2285	3022
Synchronization occurrences	11	20	20	20
Throughput [samples/μs]	18.81	134.35	32.78	17.86

their energy efficiency in near-sensor applications [168]. The metrics used are execution cycles, instructions, and stalls across different code regions using hardware performance counters. For power analysis, Synopsys PrimeTime 2019.12 is employed, assuming a nominal voltage of 0.65 V and a frequency of 250 MHz. Key metrics for the analysis include:

- Parallel speed-up: sequential versus parallel execution time.
- Throughput: input data samples per total execution cycles.
- Energy Efficiency: operations per second relative to power consumption.

Parallel speed-up and overheads Parallel performance is limited by overheads deriving from two main sources: stalls in the core pipeline during the execution of instructions and time spent in synchronization. Table 3.1 reports stalls and synchronization occurrences considering 8 cores and 2048 data samples. This table also reports the throughput of the algorithms as an absolute performance metric. On PULP, pipeline stalls derive from memory latency (load-use stalls), concurrent accesses to the TCDM banks (memory contention arbitration stalls), concurrent requests to a shared FPU (FPU contention arbitration stalls), and instruction cache misses. Analyzing the cause of stalls

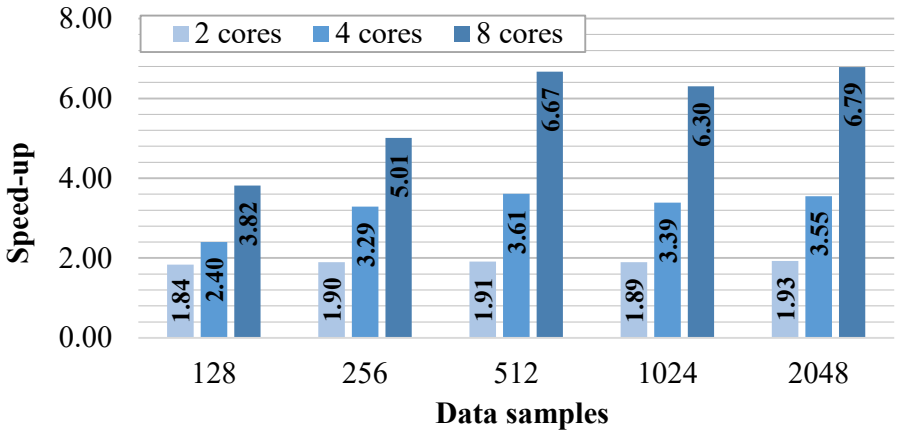


Figure 3.3: Speed-up of STFT varying the number of cores and the input size.

guided fine-grain tuning of the optimization techniques described in the previous section. Synchronization happens on barriers required by the algorithms. As reported in [92], the barrier cost with the event unit is 6 cycles, which implies an average synchronization overhead of around 1.41%. Without the event unit, the barrier overhead is 176 cycles (on 8 cores), and the average synchronization overhead rises to 41.48%.

Figure 3.3 reports the parallel speed-up of STFT. Each value on the x-axis corresponds to a fixed number of input data samples, while the window overlap does not affect speed-up. In general, the speed-up increases with data samples since this trend amortizes the overheads due to loop-level parallelism. The case of 512 data samples is out of trend because a preliminary radix-2 stage is not required since 512 is a power of eight. Table 3.1 shows that the 8-cores configuration is mainly limited by the FPU sharing since the contribution of FPU stalls (4992) over the total instructions (20677) is around 25%.

Figure 3.4 reports the parallel speed-up of DWT. Each bar provides the values of the speed-up for a filter size (FS) equal to 2 (light shade), 8 (intermediate shade), and 16 (dark shade). In general, the speed-up increases with the filter size, but there are some remarkable exceptions. Executing on 2 cores with a workload of size 512 or 1024, the speed-up of $FS = 2$ is higher than $FS = 8$ (label 1). This effect is even more evident when executing a workload of 2048 data samples on 2 or 4 cores, where the case of $FS = 2$ becomes the highest speed-up (label 2). This trend is because the parallel orchestration of the Haar wavelet is more lightweight, as explained in Section 3.2.2. The 8-core configuration implies additional overheads (i.e., TCDM contentions and FPU stalls in Table 3.1) and workload unbalancing (see Section 3.2.2) hiding this beneficial effect.

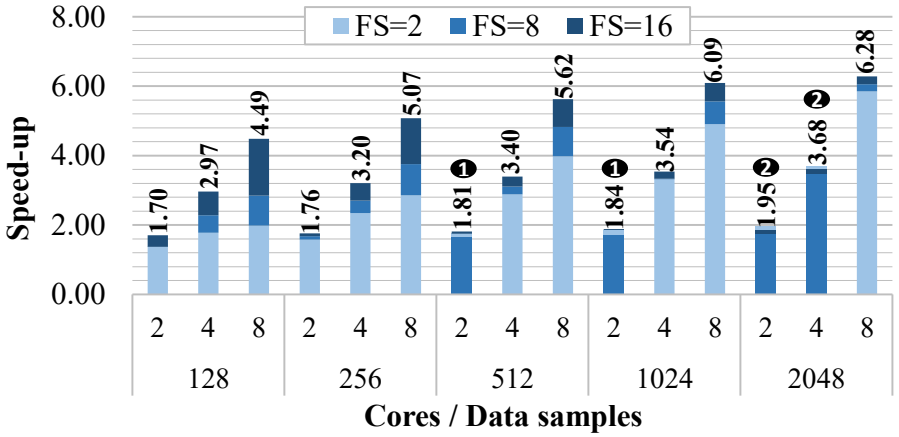


Figure 3.4: Speed-up of DWT varying the number of cores, the input size, and the filter size (FS).

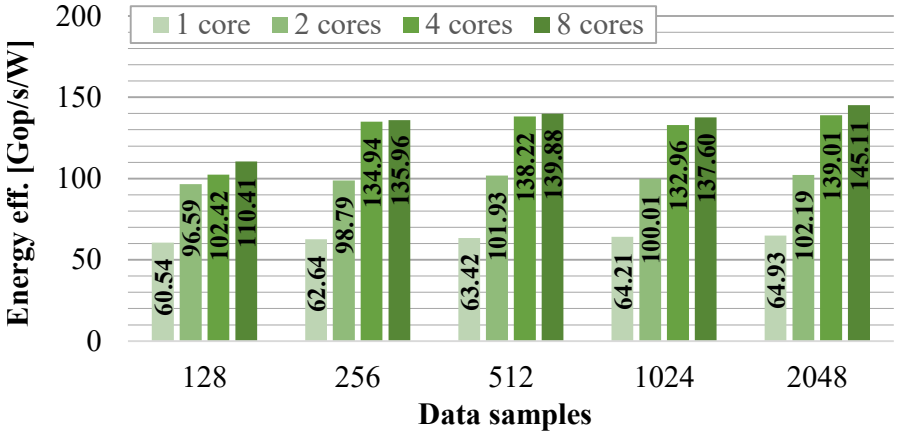


Figure 3.5: Energy efficiency of STFT varying number of cores and input size.

Energy efficiency Figure 3.5 and Figure 3.6 depict the energy efficiency of STFT and DWT, respectively. This metric grows with the input size when considering a fixed number of cores. Moreover, it increases by fixing the input size and changing the number of cores from 1 to 4, but it presents a trend inversion passing from 4 to 8 cores in DWT. Again, this is due to the additional overheads implied by the 8-core configurations, together with the higher power consumption. This effect can be amortized by computing a bigger input set. For instance, the energy efficiency is almost equivalent between the 4-core and 8-core configurations for 2048 samples.

Comparison with other libraries and architectures Table 3.2 compares the execution time of our solution with the ones of the libraries introduced in Section I, namely GSL and Kiss FFT, executing on an 8-core PULP cluster with an input size of 2048 samples. GSL only supports Haar wavelets ($FS = 2$). We applied minimal modifications to these algorithms to use the PULP HAL. Overall, our design outperforms other libraries thanks to our domain-specific code optimizations and parallel design. The design of

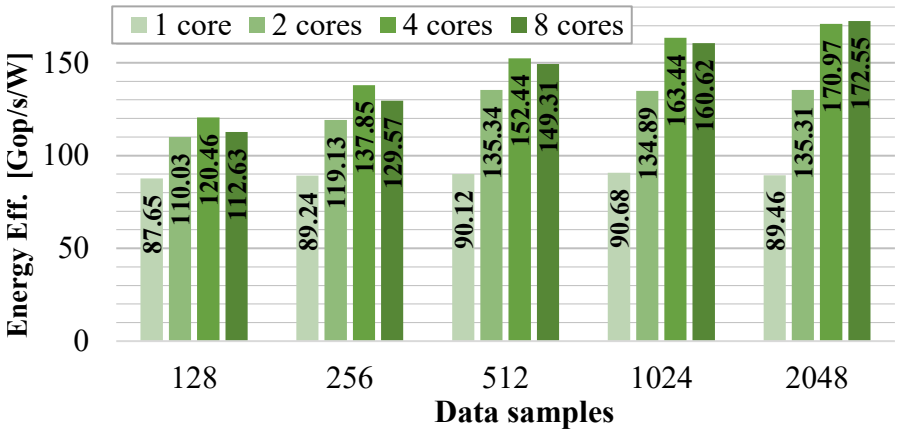


Figure 3.6: Energy efficiency of DWT varying number of cores and input size.

Table 3.2: Comparison with GSL and Kiss FFT on 8-core PULP (cycles).

Our FFT	**Kiss FFT**	**Our DWT** ($FS = 2$)	**GSL DWT** ($FS = 2$)
27218	2293231	3881	67973

Kiss FFT applies parallelization only at the outer loop level (transform steps), which does not guarantee a perfect workload balancing when the number of cores is higher than four, with detrimental effects on performance. Both algorithms do not employ the optimization techniques (e.g., loop unrolling) described in Section 3.2.2. We also performed a comparison between PULP (8-core configuration) and Cortex-M4, using an STM32F401C-DISCO development board running at 1.7 V and 84 MHz, with an average power consumption of 20 mW. The STFT implementation for the Cortex-M4 platform makes use of the `armrfftfastf32` function from CMSIS-DSP, which is partially written using inline assembly and is the most efficient FFT implementation available for this platform. The DWT implementation for Cortex-M4 uses our library since a preliminary analysis highlighted that it is 60% faster than GSL. Figure 3.7 shows that parallel execution on the PULP platform outperforms Cortex-M4 by one order of magnitude. In terms of energy efficiency, PULP achieves 145.11 and 172.55 Gop/s/W for STFT and DWT, respectively, as reported in Figures 3.5 and 3.6. Considering the performance measured on Cortex-M4 (71.4 and 75.6 Mop/s), it reaches 3.42 and 3.63 Gop/s/W, which is about two orders of magnitude worse than PULP.

Experimental results assess that both algorithms achieve high parallel speed-ups, throughput, and energy efficiency on the PULP platform, outperforming a conventional single-core MCU in terms of performance and energy efficiency. These algorithms are pervasive in many applications running on IoT end nodes. For this reason, high optimization is crucial to satisfy the ever-increasing requirements of future applications.

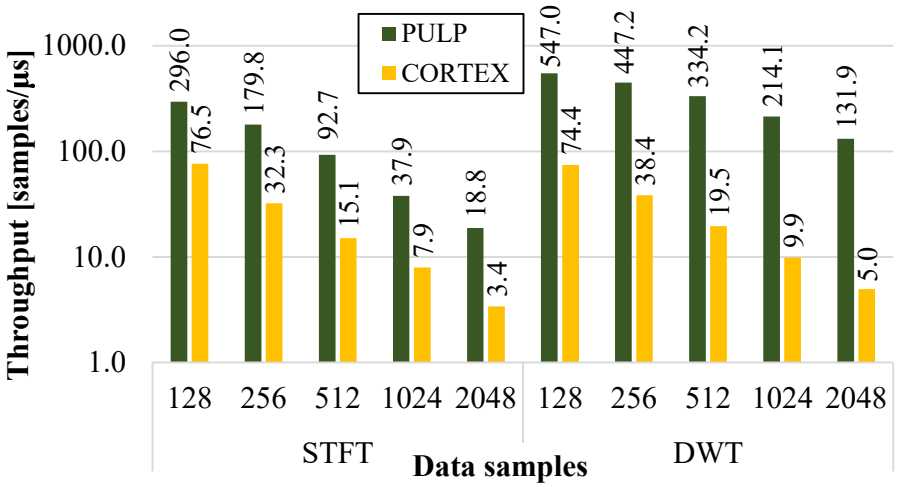


Figure 3.7: STFT and DWT throughput on PULP / Cortex-M4 varying input size.

3.3 Optimized Heart Rate Detection System

This section focuses on another method of the processing of the signal. In real-time, the R peaks detection of the ECG signal is crucial to providing information on cardiac functionality, and several strategies have been presented in the past. The time interval between consecutive R peaks is called the R-R interval, and the heart rate (or heart frequency) is the inverse of this interval. To obtain the heart rate in beats per minute (BPM), divide 60 by the R-R interval (in seconds):

$$\text{Heart rate (BPM)} = \frac{60}{\text{R} - \text{R interval [sec]}} \tag{3.1}$$

So, the R peak distance helps determine the heart frequency.

This work adapts the classical Pan and Tompkins (PT) algorithm [169] for efficient execution on low-power MCU platforms to design a full-fledged heart rate detection system. The target is a commercial MCU based on ARM Cortex-M4 and the ULP solution based on the RISC-V Parallel Ultra-Low-Power Platform (PULP) [170]. This recent SoC is implemented in 22 nm technology, namely Vega [171]. It provides a DSP-oriented instruction set architecture (ISA). Experimental results show that this approach achieves an accuracy above 99.5%, comparable to the SoA solutions, and an energy efficiency that is one order of magnitude better than other software solutions.

The contributions of this work, as discussed in this thesis, are as follows:

- A lightweight design for HR computation based on the PT algorithm.
- An implementation of the PT algorithm optimized for a balanced trade-off between computational complexity and energy efficiency.

- A real-time ECG monitoring application featuring an end-to-end system from data acquisition to signal inference.

The proposed methodology is optimized by simulating the real-time operation in MATLAB and then implementing it with a multi-board setup. Then, the processing is coded in C language and can work in data streaming or with an existing dataset. The proposed system provides a power budget of less than 5 mW for wearable and near-sensor processors. We aim to process ECG signals to carry out the HR, which is a crucial physiological parameter to detect anomaly conditions in heartbeats [172]. Our methodology obtains an acceptable HR detection reliability (higher than 99%) in pathological or sudden changes of the biosignal. The target device that we use for experimental assessment is the Parallel processing Ultra-Low Power (PULP) many-core platform designed for smart ULP embedded devices [170]. For the evaluation, we analyze the performance on the Vega SoC [171], a PULP platform running at 0.8 V at an operating frequency of 170 MHz, and on ARM Cortex-M4, using the STM32 NUCLEO-F401RE development board at 1.8 V and 84 MHz. The PULP provides extreme energy efficiency, and we obtained an energy consumption of 0.2 mJ when considering an average of 25 s of running time. We performed tests on four datasets, three existing ones and one acquired from the proposed system in real-time, taking into account several options: normal conditions, arrhythmia [173], intense physical exercise [174]. Overall, we achieved accuracy above 99.5% that we compared with other SoA solutions.

3.3.1 Related Work

Several works exploit digital platforms capable of executing digital signal processing (DSP) to achieve ULP consumption [175], [176]. In this context, the designers typically adapt optimization strategies to reduce the algorithm complexity and find the best trade-off between reliability and low power consumption. Among the biopotentials that can be acquired with real-time low-power devices, [1], heart activity parameters are the most used to detect and monitor acute severe conditions. Analyzing the QRS complex and detecting R peaks is crucial for providing cardiac functionality information. The scientific literature includes several strategies based on well-established signal-processing techniques. Park et al. [177] propose a technique based on a wavelet transform (WT) coupled with the Shannon energy envelope method in addition to a moving average filter and a squaring operation for the preprocessing step. This method achieves an accuracy of over 99%. However, the algorithm is computationally intensive, and it is not suitable for real-time execution on an ultra-low-power embedded device. Martinez et al. [178] adopt the phasor transform. This approach converts each ECG sample into a complex number, maintaining the phase and the root mean square values to enhance the

wave variations and distinguish them from each other. The overall accuracy is higher than 99% also in this case. However, the analysis excludes five records from the MIT-BIH Database because of the low-quality acquisition of highly variable signals or noise distortion.

A widely explored family of approaches for ECG signal analysis includes *slope-based methods*. In Tekeste et al. [51], the authors optimize peak detection by providing a hardware unit to approximate the computation of the signal derivative. The power consumption of the system, implemented in 65 nm technology, is 3.9 nW at an operating frequency of 3 kHz. Nevertheless, they do not consider the contribution to the power consumption of the additional computations that are strictly required by a real-life scenario. In our work, we use microcontroller-class devices that can perform preprocessing, peak detection, and subsequent computations. De Giovanni et al. [52] propose a software-based methodology that can be considered the current SoA. Their algorithm implements a Bayesian filter, normalization, and a clustering technique to optimize the R peak detection. The authors test the system on a biosignal dataset where sudden event changes occur, such as during intense physical exercise [174]. These physical conditions reduce the robustness of the traditional algorithms, affecting their reliability. Hence, they propose an accurate adaptive design for low-power platforms. However, the authors do not consider the aspects related to real-time signal acquisition. They use an existing system (BIOPAC) that requires a 9 V battery and is not energy efficient. Furthermore, the peak detection algorithm, including all the proposed phases, is very complex and requires a core with native FPU support because the fixed-point representation decreases the accuracy significantly. Overall, we will show that their results in mJ are 7× higher than our method.

An effective and computationally efficient threshold-based approach for QRS extraction and heart rate (HR) calculation is the PT algorithm. It relies on an adaptive dual-threshold technique for R peak detection, leveraging a filtering stage and simple adaptive thresholding methods. PT is a robust technique that uses a preprocessing pipeline that includes standard filtering techniques (pass-band, derivative, squaring, integration). This technique can also be applied to signals with arrhythmia. Furthermore, it can be adapted to process real-time streaming data, which is crucial in the context of wearable systems. PT is a standard approach that was proposed several years ago, but recent works have adopted this methodology yet [179], [180]. We outperformed the accuracy and energy consumption of these works, optimizing the R peak detection on our target architecture.

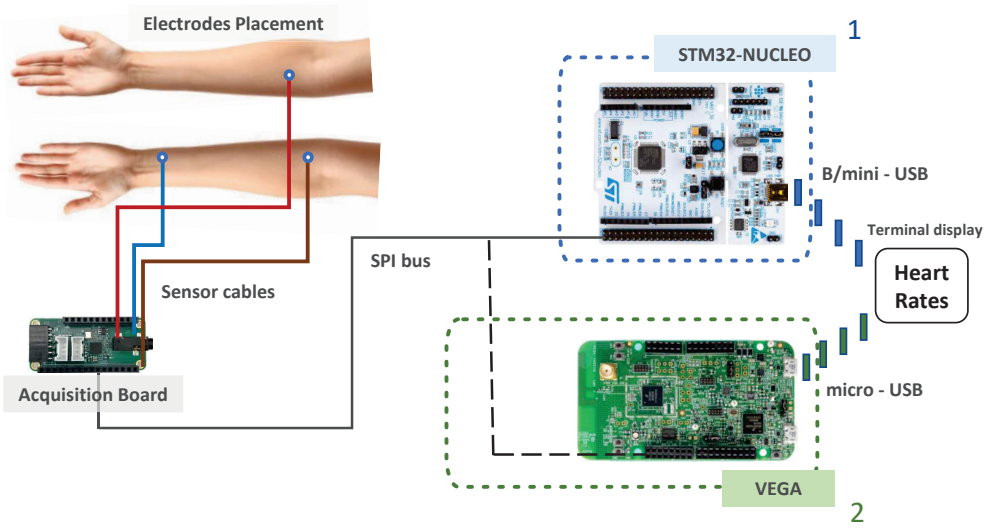


Figure 3.8: Hardware diagram of the proposed system. The active electrodes are located on each forearm and one on a wrist, setting Lead I for the collected data. Single-channel ECG is acquired with a custom AFE board (MAX30003), which sends data via SPI to the platform for processing. We consider two alternative designs: (1) STM32NUCLEO for the initial setup and (2) Vega for ULP optimization. Output and communication are managed via a B/mini-USB and a micro-USB cable, respectively, that leads the platform to a terminal to visualize the HR values.

3.3.2 System Architecture

This work proposes a modular setup for ECG detection. The acquisition board relies on Maxim MAX30003 [181], a chip for ULP acquisition of ECG. MAX30003 is a complete, biopotential analog front-end solution for wearable applications. It offers high performance for clinical and fitness applications at extreme energy efficiency, reaching 85 μW average power consumption. The analog acquisition is based on a 2 leads differential channel providing ECG waveforms and heart rate detection. The biopotential channel has ESD protection, EMI filtering, internal lead biasing, and DC lead-off detection. The biopotential channel also has high input impedance, low noise, high CMRR, and programmable gain, as well as low-pass and high-pass filter options. The digital back end is based on an SPI interface to enable data streaming and communication with an external MCU.

Fig. 3.8 depicts the custom board equipped with MAX30003 and with two alternative test benches: the first one with NUCLEO-F401RE board, used for initial setup and algorithm tuning, and the second one with Vega custom board [171], employed for ULP operation and optimized performance. In both test benches, ECG data are sent from AFE to MCU via SPI. Vega allows a USB device mode interface with a micro USB connector at an operating frequency of 2.4 GHz with a reference oscillator frequency of 32 MHz. Vega receives data from AFE via a 5 MHz SPI channel connection

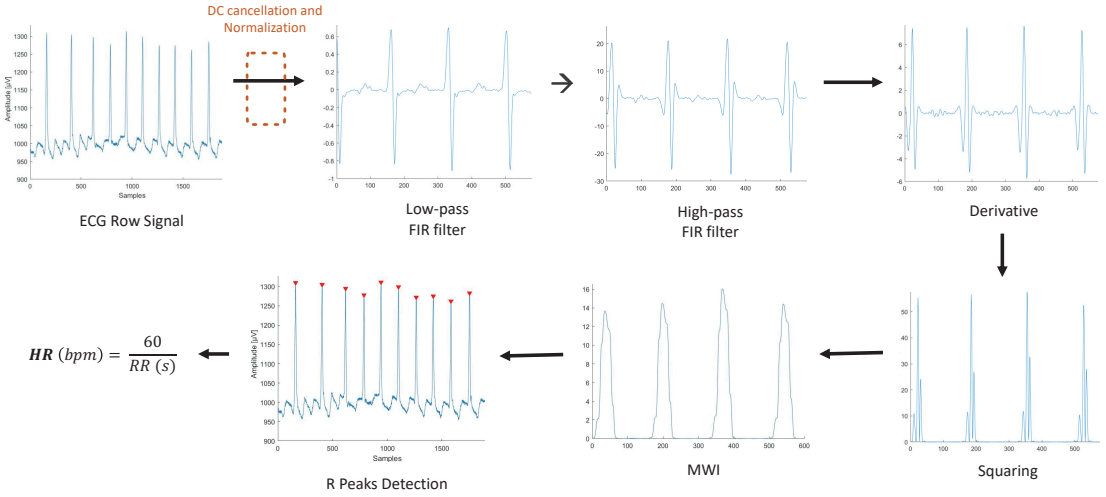


Figure 3.9: Signal processing steps based on PT technique: (1) Cancellation of DC component and addition of Normalization; (2) Band-pass filter that combines the low- and the high-pass filters; (3) Derivative function; (4) Squaring function; (5) Moving window integrator (MWI); (6) R peaks detection. In the last step, we compute the HR in beats per minute.

(Vega acts as master). Data loaded via SPI is stored in the Vega L2 memory as 24-bit signed fixed-point numbers. Acquired ECG samples are used as input of the embedded implementation, described and profiled in Section 3.3.4.

The presented prototype can be integrated into a single PCB with a 20×10 mm form factor, suitable for minimally obtrusive wearable applications. The PULP SoC is equipped with a 2 MB SRAM scratchpad memory (L2), hosting the resident code and application data. A hardware unit called *µ-DMA* performs autonomous data transfers between the L2 memory and the peripherals. The peripherals and the MCU core reside in different clock domains so that the frequency of each domain can be tuned to sustain the application workload with low power consumption (up to 500 MHz for a 22 nm technology node). The peripheral clock can be further divided to match the operating frequency of slower external devices.

3.3.3 Algorithm Description

To compute the HR, we adopt a signal processing pipeline based on the PT technique [169]. This methodology adopts a dual-threshold technique to detect the R peaks and includes multiple preprocessing signal steps required to improve the signal analysis. The block diagram is depicted in Fig. 3.9.

The signal processing pipeline includes a set of preprocessing digital filters followed by the computation of R-peaks. The original implementation considers a sampling rate of 200 Hz.

1. **Band-pass filter**. The low-pass component applies a second-order transfer function to the signal, obtaining a difference equation with a delay of 5 samples and a DC gain of 36. The high-pass design is characterized by a first-order transfer function, with a delay of 16 points and a gain of 1. Overall, the band-pass filter provides a 3 dB pass-band between 5 and 12 Hz and reduces the noise due to the muscle, the baseline wander, and the T-wave interference/frequency content. This filter supplies poles and zeros only on the unit circle, so the system is characterized by a minimum phase, a minimum group delay, and better stability. As a final effect, it increases the signal-to-noise ratio.

2. **Derivative**. The signal is differentiated using a 5-point derivative. The result provides information about the slope of the input waveform. This filter introduces a delay of 2 samples and a gain of 0.1.

3. **Squaring**. The output of the derivative signal is squared to enhance the R peaks, leading the signal to the positive y-axis to emphasize the high frequencies that include the R peaks. This step makes it easier to distinguish R peaks from T-waves.

4. **Integration**. From the output of the squared signal, a moving window integrator extracts the duration of the QRS complex, obtaining a time-averaged signal. Usually, the window length is equivalent to the widest QRS complex (around 150 ms, corresponding to 30 samples at 200 Hz). The time of the rising direction of the window is the duration of the QRS complex.

5. **Computation of R peaks**. The final part of the algorithm finds a set of fiducial marks corresponding to the temporal location of the peaks in the integrated signal. Fiducial marks determined in this area are potential candidates for R peaks. An initial phase of the implementation is necessary for the tuning (2 seconds at 128 Hz). The fiducial mark is compared with a threshold value $threshold_{I1}$ that considers the current estimation and both signal and noise peaks:

$$threshold_{I1} = npk_I + 0.25 * (spk_I - npk_I) \tag{3.2}$$

where npk_I is the estimation for any peak that is not related to an R peak (e.g., the peaks of T waves), and spk_I is the estimated value for the R peak level. When a new peak $peak_I$ is detected, it must be classified as a noise peak or a signal peak. If a sample is greater than the current threshold value $threshold_{I1}$, then it is a peak candidate. In addition, it must have a distance of at least 200 ms from the previously detected peak: this value, referred to as min_rr_width, is the minimum latency time between adjacent R peaks due to physiological constraints.

Otherwise, the fiducial mark is considered a noise peak. spk_I and npk_I parameters are updated accordingly:

$$spk_I = 0.125 * peak_I + 0.875 * spk_I \tag{3.3}$$

$$npk_I = 0.125 * peak_I + 0.875 * npk_I \tag{3.4}$$

If no R peak candidate is found in an interval of duration $1.66 * max\_rr\_width$ starting from the previous peak and ending with the current sample, the algorithm performs a search-back operation on this interval the interval using a lower threshold $threshold_{I2}$ that is empirically computed as:

$$threshold_{I2} = 0.5 * threshold_{I1}; \tag{3.5}$$

The original PT algorithm performs R peak detection also on the output of the band-pass filter, introducing a set of variables with the same meaning (i.e., $peak_F$, spk_F, npk_F, $threshold_{F1}$, and $threshold_{F2}$). We verified experimentally that this step can be skipped without invalidating the detection quality. If a peak candidate occurs after the 200 ms refractory period but within 360 ms of the previous peak, the algorithm makes an additional check to determine if it is an abnormally prominent T wave. This decision is based on the mean slope of the waveform at that position, which must be greater than one-half that of the previous peak. Finally, the average distance between R peaks is computed as the mean of the eight most recent RR intervals. The average HR can be used to refine the duration of the search back interval.

3.3.4 Evaluation

This section provides an experimental evaluation of our system. We use GVSoC [142], an open-source simulator for PULP architectures, to implement and debug the algorithm. GVSoC can simulate a full platform, including multi-memory levels and multi I/O peripherals, and provides a good trade-off between simulation speed, timing accuracy, and completeness. The average energy consumption for the Vega platform has been derived by a post-place-&-route simulation on the RTL. The metrics of interest for our performance analysis are *throughput* (computed as the number of input data samples over the total execution cycles), *energy efficiency* (operations performed in a second over power consumption), *total energy consumption* (in mJ), and *accuracy* of the detection rate (in percentage).

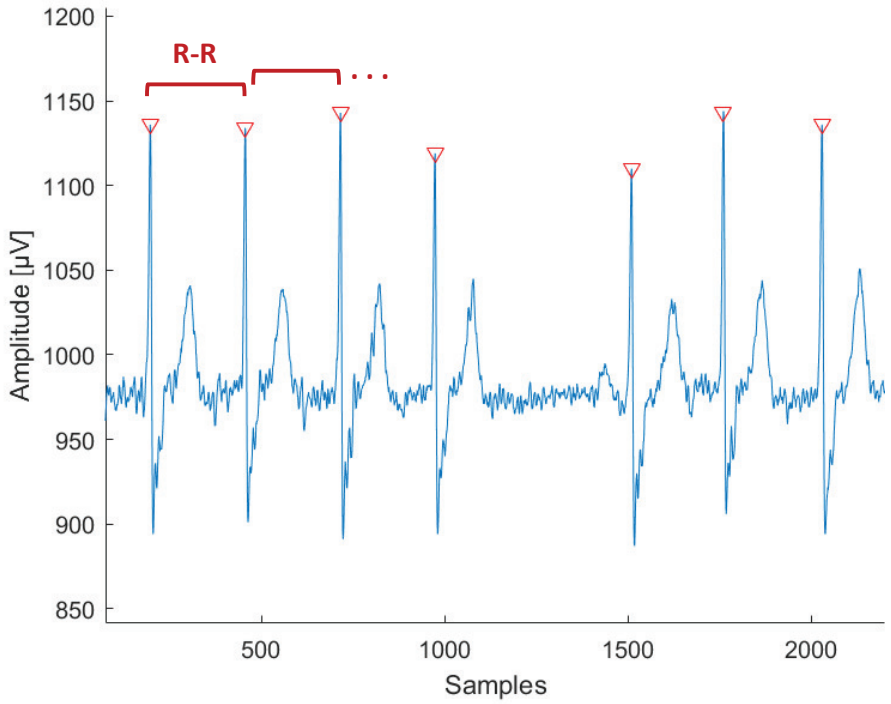


Figure 3.10: Output result of the proposed R peaks detection and R-R intervals method from a segment of record 232 characterized by the supraventricular ectopic beats from the MIT-BIH database.

3.3.5 Implementation on the PULP platform

The AFE IC, described in Section 3.3.2, is connected to the ECG electrodes using 3 ECG surface sensors: two sensors are placed on the wrist, and the other one around the upper forearm of the subject, as a voltage reference. This setup allows sampling with an 18-bit resolution at 128 Hz. In this application, the signal is read 13 samples at a time using a FIFO. We apply the PT algorithm described in Section 3.3.3, implemented in C language, to support different (integer or floating-point) data types. The code supports both buffered and data-streaming simulation with configurable parameters for the sampling frequency. In the case of buffered execution, the input buffer size is selected to contain at least 1.66 times an R-R interval, considering that the maximum physiological beats per minute are 60 or 80 (max 1.66 beats per second). The code includes buffers for the results of the intermediate filters. These buffers have the exact window size for the corresponding filter and are implemented as circular buffers to reduce memory consumption. Buffered execution can be used to execute the algorithm on pre-recorded ECG datasets, while the streaming variant is more efficient for real-time data acquisition.

In addition to the original PT design, we added a preliminary normalization step that removes the DC drift by subtracting the mean value and then dividing by a maximum

absolute value. In the case of buffered execution, this value can be computed as the maximum value in the input buffer; otherwise, we can use the maximum value provided by the sensor as reported in the datasheet. The result is a signal normalized in the range [-1, 1], improving the numerical stability and precision of the next steps.

The filter coefficients are pre-computed using MATLAB and saved into the local memory to maximize the efficiency of the initial steps. To guarantee the minimum latency for streaming execution, we designed a step-by-step convolution function that is invoked for each new available value (i.e., a new input sample or a value computed by the previous filter) and applied a linear convolution filter to the tail on the corresponding data buffer. As introduced in Section 3.3.3, we only consider the integrated signal for R peak detection. Finally, we apply the computation of the HR (beats per minute) from the RR average value.

$$HR = 60/(RR_{avg}/Fs); \tag{3.6}$$

where Fs is the sampling rate.

Parallelization strategies are applied to improve efficiency and performance when processing ECG signals. The BUFFER SIZE-based shifting mechanism is used to maintain a rolling window of past ECG samples. Instead of processing the entire signal at once, this shifting method enables a streaming approach, where different sections of the signal can be processed in parallel; the convolution operations used in filtering (low-pass, high-pass, derivative, and integration) could benefit from loop unrolling to improve instruction-level parallelism. This reduces overhead from loop control instructions and allows multiple computations to be performed per iteration; the searchback algorithm, which scans the ECG signal to detect R-peaks, is split into searchback start and end regions, enabling parallel execution of peak detection across different segments; the algorithm maintains separate calculations for threshold-based peak detection and adaptive threshold adjustment, which could be independently parallelized across available cores; the HR calculation iterates over detected peaks to compute an average. The use of a fixed-size circular buffer allows efficient parallel reductions to compute the mean RR interval, enabling multi-core processing.

Figure 3.10 shows an example of the output result of the R peaks detection and the R-R intervals assessment extracted from a segment of the record 232 of the MIT-BIH Arrhythmia Dataset. Even though some fiducial points can be drifted forward or backward by one sample w.r.t. the exact peak positions, this effect does not affect the correct computation of the R-R distance.

Table 3.3: Cycles for each sample, Instructions, Energy Efficiency, Throughput, and Time executing on the target platforms (average values on a 25 s time window).

	Pulp VEGA	**Cortex-M4 (processing pipeline)**
Cycles	2771	3154
Instructions	2204	3148
Energy efficiency [Gop/s/W]	34.7	3.8
Throughput [samples/ms]	61.35	26.63

Table 3.4: Energy consumption of different SoA solutions for R peak detection (average values on a 25 s time window).

	Platform Architecture	ISA	Algorithm	Technology [nm]	Operating frequency [MHz]	Energy consumption [mJ]
De Giovanni et al. [52]	Pulp (Mr.Wolf)	RV32ICMF + Spec. Ext.	Adaptive slope	40	170	1.553
This work	Cortex-M4	ARMv7-M	PT	90	84	2.652
	Pulp (Vega)	RV32ICMX + Spec. Ext.	PT	22	170	**0.203**

3.3.6 Performance analysis and energy consumption

Table 3.3 reports the performance parameters executing the program (in streaming mode) on PULP (VEGA SoC) and Cortex-M4 (STM32NUCLEO-F401RE development board). We deployed an alternative setup for these experiments where an additional STM32NUCLEO board is used in place of VEGA for the signal processing pipeline. In both cases, the energy consumption of the Nucleo board used for system initialization and debugging is not considered. The resulting values show that execution on the PULP platform is 2.3× faster than Cortex-M4.

Table 3.4 depicts the energy consumption (in mJ) of our algorithm executed on NUCLEOF401RE and Vega platforms compared to the state-of-the-art solution described by De Giovanni et al. [52], which executes on a PULP platform based on the Mr.Wolf architecture [159]. The operating frequency reported for Cortex-M4 is its maximum frequency. For VEGA, we are using an operating frequency lower than the maximum to make a more fair comparison with state-of-the-art solutions. The energy consumption has been estimated using an average power consumption reported by the datasheet. Considering an execution time of 25 s, the average energy consumption of our system is almost 7× lower.

3.3.7 Algorithm Accuracy

Table 3.5 reports an accuracy comparison between our solution and other works. In the worst case, our algorithm reaches 99.53% on the high-intensity physical exercise

Table 3.5: Comparison of R peaks accuracy.

	Acc [%]
Moreira et al. [180]	93.26
De Giovanni et al. [52]	97.90
Tekeste et al. [51]	99.37
Lu et al. [179]	99.41
This work	**99.53**

dataset [174]. To evaluate the accuracy, we used the MATLAB *findpeaks* function as a golden reference, which returns the local maxima. It is extremely accurate, but it has two main flaws. First, it is computed intensive, which is highly detrimental to its adoption in the ultra-low-power embedded domain. Second, it cannot be adapted to a streaming context, so its adoption would increase the latency of the results. We computed the *accuracy* as follows:

$$RMSD = \sqrt{\frac{\sum_{i=1}^{n} (x_{G,i} - x_{PT,i})^2}{n}}; \tag{3.7}$$

$$Acc = 100 - (\frac{RMSD}{x_{max} - x_{min}}); \tag{3.8}$$

where x_G and x_{PT} are the RR intervals (in samples) computed using the golden model or the proposed method, respectively. The parameter n is the number of detected RR intervals, and x_{max} and x_{min} are the maximum and minimum in the set of RR interval values.

Figure 3.11 depicts the accuracy of the code tested on four different datasets. The datasets we consider are Normal Sinus Rhythm (NSR) and Atrial Flutter (AFL). They are both from the MIT-BIH Arrhythmia database, sampled at 360 Hz [173]. The third is the ECG signal acquired in real-time (RT) from our signal acquisition system (described in Section 3.3.2). Finally, the signal on high-intensity exercise (HIE), sampled at 250 Hz [174]. The figure shows the higher value in NSR, for which we achieve 99.95%. In the case of tachyarrhythmia, called atrial flutter (AFL), the accuracy is 99.62%. In the RT, we obtain an accuracy of 99.61%. In HIE, where the beats change suddenly, we assess the lower value of 99.53%.

3.3.8 Discussion

This part of the work presents the design and implementation of a heart-rate detection system leveraging the PT algorithm on low-power MCUs. This approach considers

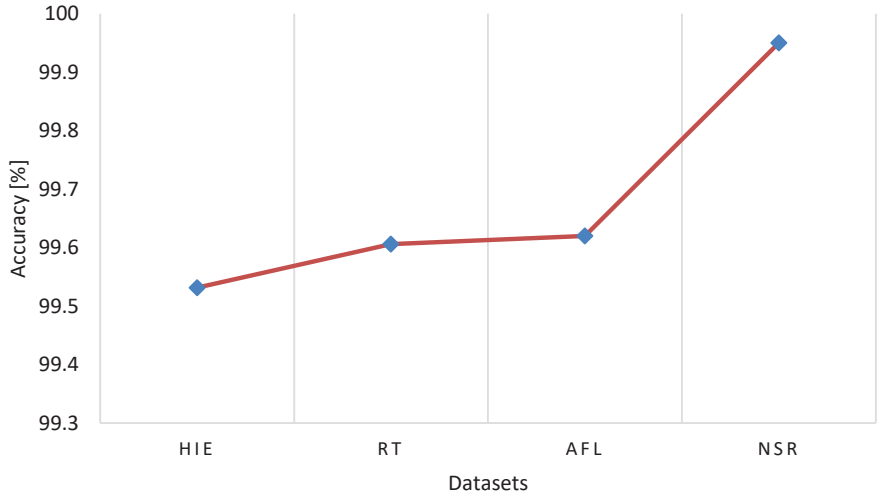


Figure 3.11: Accuracy evaluation on four different datasets: High-Intensity Exercise (HIE) [174], acquired ECG signal in real-time (RT) with the proposed system design, Atrial Flutter (AFL) [173], and Normal Synus Rhythm (NSR) [173].

two alternative platforms: a commercial MCU based on ARM Cortex-M4 and an ultra-low-power solution based on RISC-V, namely the Vega SoC. Experimental results show that this approach implies a lightweight design, with execution times of a few thousand cycles. This system provides a lifetime battery of 81 hours with a 100 mAh battery, achieving an accuracy comparable to the SoA solutions and a better energy efficiency of one order of magnitude.

This chapter refers to publications [154], and [182].

This work does not aim to classify specific health problems but rather to apply preprocessing techniques and to detect HR in real-time with high reliability and energy efficiency. Chapter 4 focuses on machine learning algorithms (e.g., TCN and CNN) to the system pipeline to detect anomalies in HR variability, such as arrhythmia or stress conditions. Moreover, it proposes to design a parallel version of the code using the programmable parallel accelerator available on PULP platforms to improve performance and energy efficiency further compared to commercial alternatives.

Chapter 4

ECG Applications - ML approach

This chapter presents a machine learning (ML) approach to detect pathological conditions in the ECG signal as the final step in the processing pipeline. The primary objective is to design a highly energy-efficient convolutional neural network (CNN) optimized for arrhythmia detection on the PULP platform [157]. This approach also enables a comparison with current state-of-the-art (SOA) solutions on accuracy and energy consumption.

4.1 ML approach for ECG inference

As a first step, we experimented with implementing a temporal convolutional network (TCN) and analyzed its performance based on accuracy, inference operations, and energy efficiency. However, due to the inherent trade-off between computational cost and accuracy, we ultimately found CNNs more suitable for our objectives. This chapter details the evaluation results of both models, concluding that CNNs offer a balanced and scalable solution for reliable ECG-based arrhythmia detection on low-power platforms as explained in Section4.7.

Hence, the CNN solution uses the PT algorithm to perform real-time identification of the heartbeats on the ECG data stream, as described in Section 3.3.3. Next, the processing pipeline includes a CNN that can recognize five arrhythmia classes. The PT workload is negligible compared to the CNN inference, and the heartbeat localization prevents us from running the network continuously. This approach achieves a 95% accuracy on the best energy-efficient configuration, which is 3% lower than the best solu tion available in the current SoA; at the same time, this solution is 3× more energy-efficient in the performance of network inference. Considering the full processing pipeline

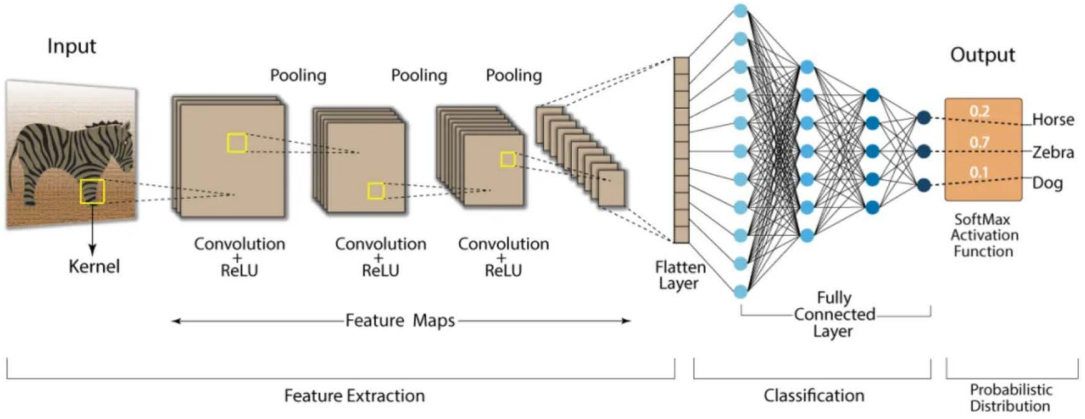


Figure 4.1: Complete CNN representation. Image source: [183].

that takes advantage of the power savings states available on the target platform, the total reduction in energy consumption is approximately 28%. This design improves the battery lifetime compared to the current SoA solution based on TCNs and transformers.

CNN has been adapted to process one-dimensional (1D) data, such as ECG signals. In this case, the input to the network is structured as a 1D channel, which is conceptually equivalent to an image with just one pixel in either the height or width dimension. This design choice enables CNN to extract temporal features from time-series data, similar to how it would detect spatial patterns in a two-dimensional (2D) image. Figure 4.1 illustrates an example of the overall structure of a CNN, using an image as the input to help visualize the process.

To provide a clear context, we first introduce the concepts of CNN and TCN.

Convolutional Neural Networks CNNs are a type of ML model designed primarily for processing data with grid-like topology, such as images, but it can also be adapted for 1D signals like ECG data. CNNs are highly effective at identifying patterns through their ability to automatically learn spatial hierarchies of features from input data using a structure composed of convolutional layers, pooling layers, and fully connected layers.

The key components of a CNN are:

1. Convolutional Layers: The core building blocks of CNNs are convolutional layers, where a set of filters (or kernels) slides over the input data and computes feature maps. These filters detect local patterns like edges or textures in images or specific waveforms in time-series data like ECG signals.

2. Activation Functions: After each convolution operation, an activation function (typically ReLU) is applied to introduce nonlinearity into the model, enabling it to learn more complex patterns.

3. Pooling Layers: Pooling layers, such as max pooling, reduce the spatial dimensions of the feature maps, helping to downsample the data and reduce computational complexity while preserving important information.

4. Fully Connected Layers: After a series of convolutional and pooling layers, the extracted features are flattened and passed to fully connected layers, which combine the features to make predictions based on the learned patterns.

5. Output Layer: In classification tasks, the final fully connected layer outputs class probabilities through a softmax activation function, enabling the model to classify the input into predefined categories (e.g., different heart conditions in the case of ECG signals).

Advantages of CNNs:

- Efficient Feature Extraction: CNNs excel at learning and extracting features automatically from input data without requiring extensive manual feature engineering.
- Translation Invariance: Convolutions provide spatial invariance, meaning the network can detect features regardless of their position in the input.
- Scalability: CNN can handle large and complex datasets, adapting to different input sizes by adjusting the number and depth of convolutional layers.

Temporal Convolutional Networks TCNs are a variant of CNNs designed to catch temporal dependencies within a bio-signal by utilizing 1D-convolutional layers operating along the time dimension. [184], [185]. Compared to Recurrent Neural Networks (RNNs), TCNs are less computationally demanding since they only perform 1D convolutions along the time dimension. Moreover, TCNs overcome common issues associated with recurrent topologies, such as the vanishing gradient problem and memory retention limitations [186]. At the same time, they have been demonstrated to provide comparable accuracy for bio-signal applications. TCNs utilize dilated convolutions and enforce causal connections, ensuring that predictions at any point depend only on past data. This approach helps to capture long-range dependencies but introduces some complexity due to the dilation and causality requirements described below. Dilation increases the receptive field exponentially, but it also leads to sparse connections, which can affect efficient memory access and computation. Causal convolutions enforce a strict ordering

of inputs, which require additional padding or masking, increasing computational overhead. While dilated convolutions reduce the number of layers needed to achieve a large receptive field, the overall number of parameters may still be significant, especially if large kernel sizes or multiple channels are used.

The concepts of causality and dilation hold special significance when dealing with time-dependent inputs and activations in TCNs. **Causality** ensures that no output y_t at time t is influenced by future input values, maintaining temporal consistency. This means the output y_t is derived solely from past input samples, specifically from the set $x_{t-K+1} \ldots, x_t$, weighted by the convolutional kernel of length K. **Dilation** is a technique used in TCNs to expand the receptive field of the convolution without increasing the model size or computational complexity. It introduces a fixed interval d between the input samples considered by the convolutional kernel, allowing the model to capture wider temporal patterns with fewer parameters.

Thus, a general temporal convolution operation incorporating both causality and dilation can be represented by the following formula:

$$y_t^{c_{\text{out}}} = \sum_{c_{\text{in}}}^{C_{\text{in}}-1} \sum_{k=0}^{K-1} W_k^{c_{\text{out}} c_{\text{in}}} x_{st-dk}^{c_{\text{in}}} \tag{4.1}$$

for $c_{\text{out}} = 0, \ldots, C_{\text{out}} - 1$ and $t = 0, \ldots, T-1$; where x and y are the convolutions' input and output, respectively, t is the index of the time sample, T is the sequence length, W is the tensor of kernel parameters, c_{in} and c_{out} are the indices of the input and output channels, respectively, C_{in} and C_{out} are the total numbers of input and output channels, respectively, K is the temporal size of the filter, s is the stride, and d is the dilation. The extension of the receptive field is $F = (K-1)d + 1$ [187].

It is important to note that while causality and dilation are potential features of TCNs, they are not strictly necessary for effectively processing time-series data. In the embedded TCN applications developed in this thesis, the entire input time window is available at the time of inference, and the inference process begins as soon as all the data within the window have been acquired. This allows for the use of a symmetrical neighborhood around each sample without causing any leakage of future data.

Regarding dilation, the emphasis on it derives from the successful heuristics in original temporal deep models, such as the WaveNet architecure [188], which proved to benefit from a modular structure where the i-th convolutional layer of each model had dilation factor $d_i = 2^i$. On the contrary, in the embedded applications of this thesis, the goal is to pursue accuracy with the lightest possible models. Moreover, some works on TCNs emphasize the presence of residual connections [184] in a strong relationship with

ResNet [189]. However, residual connections only sometimes prove beneficial in practice; in particular, they are not used in the work presented in this thesis. Additionally, several studies on TCNs highlight the role of residual connections [184], drawing a strong parallel to ResNet [189]. However, in practice, the advantages of residual connections are not always evident, and they are not implemented in the work discussed in this thesis.

4.1.1 Related Work

In recent years, the remarkable success of transformer models in Natural Language Processing (NLP) has raised interest in applying them to other application domains, including time series processing. One of the major advantages of Transformers is their ability to learn long-range dependencies and model complex relationships between different parts of the time series data. In the context of ECG signal analysis, Busia *et al.* [190] presented a tiny transformer model able to recognize five arrhythmia classes. The accuracy of this model is high (98.97%), and the limited number of parameters (6k) enables its deployments on memory-constrained MCU-class devices. However, the energy consumption of these solutions is significantly higher than that of other ML approaches, and this may severely affect the lifetime of battery-powered devices. Typically ML approaches involve simpler architectures like feed-forward layers or convolutions with local connections and lower parameter counts. Transformers use complex self-attention, multi-head mechanisms, and deep stacks of layers, resulting in higher FLOPs and memory bandwidth requirements and increase power consumption due to dense computation and large model sizes.

Li et al. [191] experiment with different ML techniques, achieving the maximum accuracy (81.02%) with an approach that combines the *filter bank common spatial pattern* (FBCSP) algorithm with multiple binary classifiers. Ismail et al. [192] implement a TCN model, achieving an accuracy of 93.4%. However, they do not report values for energy consumption. In the following work, Ismail et al. [193] adopt an approach based on reinforcement learning to optimize the training and selection of hyperparameters for multiple DL models, including MLP, CNN, LSTM, and GRU, in their evaluation. They achieve the best accuracies with CNN and LSTM (95.82% and 96.41%, respectively) but require a memory usage not suitable for our target devices (8.8 GB and 30.8 GB)

ECG-TCN [58] is a TCN-based solution that classifies the same arrhythmia classes considered in this thesis; processing one heartbeat at a time, they achieve an accuracy of 93.8% with an energy per inference of 0.10 mJ. Risso et al. [57] uses ECG-TCN with dilation set to 1 and a Neural Architecture Search (NAS) methodology to optimize the

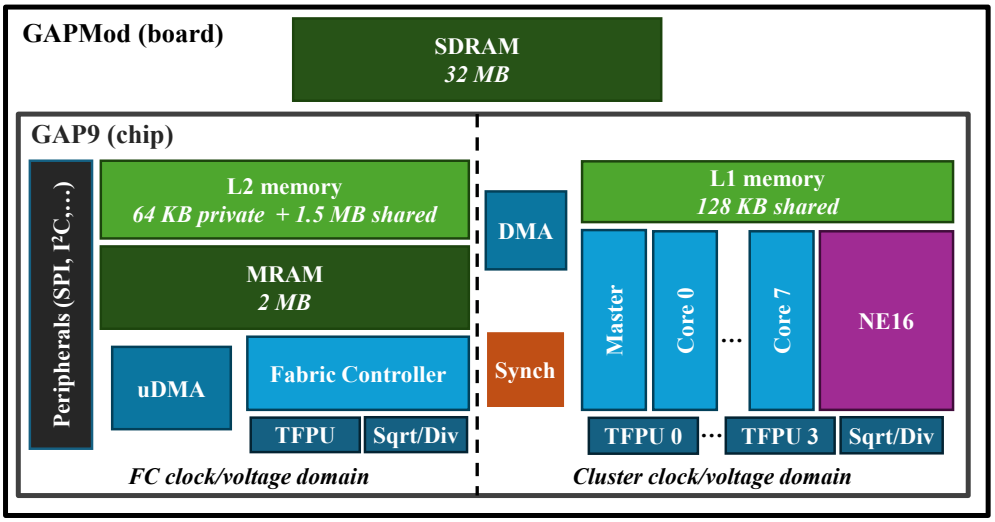


Figure 4.2: GAP9 architecture diagram.

network parameters. They obtain an energy consumption similar to ours (0.04 mJ) but with an accuracy of 93.16%.

In recent years, Transformer-based models have demonstrated remarkable success in different fields. In [194] and [195], the authors adopt an approach based on Transformers and Autoencoders. They report high accuracy (99% and 87.7%, respectively), but their results are not comparable with our setup because they group the pathological classes into a single one, performing a binary classification. Additionally, the size and parameters of the model, along with the inference time and energy consumption, are not reported. Hu et al. [196] and Yan et al. [197] use the multiple classes MIT-BIH Arrhythmia dataset, achieving very high accuracy (>90%) but requiring high complexity (5M parameters) and not suitable for limited computational resources and power constraints, such as MCUs. In [190], the authors tackle this challenge by training a *Tiny* Transformer with fewer parameters to make the deployment feasible on MCUs, and we consider this solution the current SoA.

Another approach is provided by neuromorphic processing, as presented in [198]. This method is based on delay-based reservoir computing, classified as an RNN, but it differs in that it uses an artificial neuron and a delay line. The accuracy values are comparable to the state-of-the-art, but only one class is detected, using only three records from the entire dataset, and energy consumption values are not reported.

4.2 CNN Methodology

4.2.1 Hardware platform

This thesis targets GAP9 [199], introduced in Section 2.3, a PULP commercial platform based on the RISC-V Vega architecture [200]. Figure 4.2 provides a high-level overview of the system architecture. The fabric controller (FC) is a conventional single-core MCU serving as the central orchestrator and controlling a 9-core programmable compute cluster for workload acceleration. FC and cluster cores reside in dedicated clock and voltage domains, supporting three power modes: active, deep sleep, and retentive sleep. The FC domain includes a wide set of peripherals and a micro direct memory access (uDMA) unit that can be programmed to handle data transfers between peripherals and memory.

The cluster cores share four transprecision floating-point units (TFPUs) and a Sqrt/-Div unit. TFPUs support three different formats (binary32, binary16, and bfloat16). The cluster also contains the *Neural Engine 16-channels* (NE16) [201] [202], a convolutional accelerator supporting multiple filters (1x1, 3x3, depthwise, linear), a programmable number of bits for the parameters, and an arbitrary number of input/output channels. This design allows programmers to offload convolutional kernels to a dedicated hardware unit, which significantly improves performance and energy efficiency.

The GAP9 memory hierarchy includes a 128 KB L1 memory local to the cluster and a 1.5 MB L2 memory local to the fabric controller that can operate at frequencies up to 360 MHz. Common synchronization primitives (e.g., barriers and mutual exclusion) are accelerated by a dedicated hardware unit (Synch). In addition, the FC domain includes 2 MB of Magnetoresistive Random Access Memory (MRAM), a non-volatile memory technology that offers several advantages over other more common technologies (e.g., Flash memory), mainly faster access times, lower power consumption, and greater durability [203]. All data transfers between memory levels are software-managed through a direct memory access (DMA) unit that can perform efficient data transfers without keeping the cores busy. The GAPMod development board also includes 32 MB of low-power Synchronous Dynamic Random Access Memory (SDRAM), which can be used as an L3 memory. The access latency and energy consumption are higher since this memory level is off-chip, but its adoption is mandatory when the network parameters do not fit the available L2 memory.

The GAP9 software ecosystem [204] includes *NNTool* [127], a framework to deploy NN graphs starting from the ONNX representation that ML frameworks can export, as

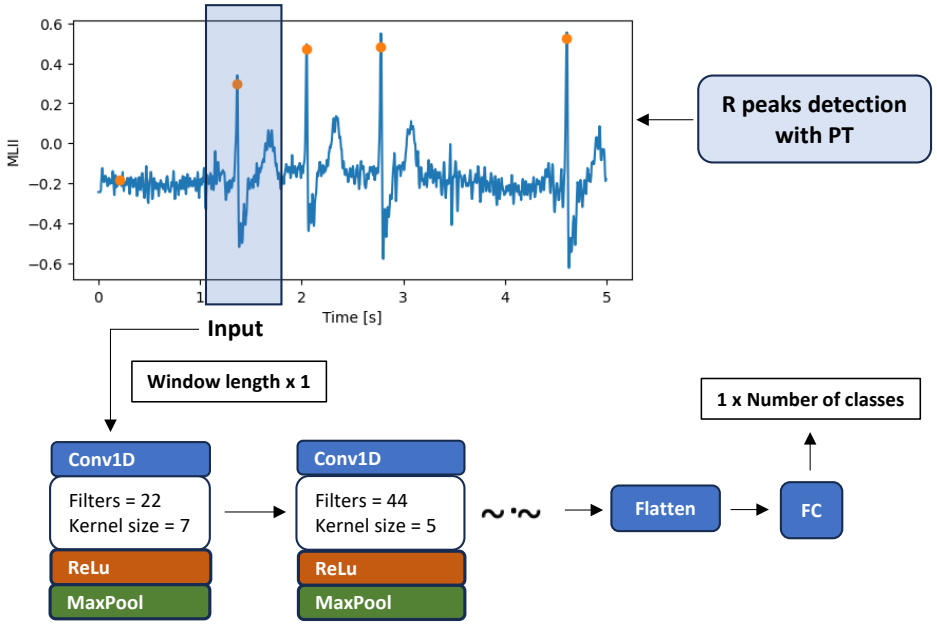


Figure 4.3: Processing pipeline for ECG signal classification. The input is a window of 256 samples (one heartbeat) centered on the R peak. The CNN design includes multiple sub-blocks, progressively increasing the number of filters and reducing the kernel size, followed by a dimensional flattening and a fully connected (FC) layer.

discussed in Section 2.4.2. NNTool includes different options for network deployments, including different quantizations and the NE16 usage.

4.2.2 CNN Preprocessing

We utilize the version of the PT algorithm [169], [182] optimized as discussed in Section 3.3, for the target platform. PT is a signal processing technique used to detect the PQRST complex in the feature extraction step, namely the sequence of waveforms seen on an ECG representing the electrical activity of the heart during a cardiac cycle. The points depicted on the ECG signal in Figure 4.3 are the R peaks detected by the PT algorithm. After this, the processing pipeline feeds a window of 256 samples centered around the heartbeat to a 1D-CNN.

4.2.3 1-D CNN architecture

Our CNN architecture is composed of multiple 1D convolutions used to extract the features of the ECG signal. The network categorizes the heartbeat into five classes: N (non-ectopic), S (supra-ventricular ectopic), V (ventricular ectopic), F (fusion), and Q (unclassifiable).

Table 4.1: Comparison of the performance across data types.

CONFIG (ϕ)	MACs [M]	FLOAT32		FLOAT16			INT8					
		Acc. [%]	Param. [MB]	Acc. [%]	Param. [MB]	Lat. [ms]	Acc. [%]	Param. [MB]	Cluster Energy/ Inference [mJ]	Cluster Lat. [ms]	NE16 Energy/ Inference [mJ]	NE16 Lat. [ms]
0	0.923	91.9	1.062	91.6	0.531	1.262	91.8	0.265	0.013	0.606	0.010	0.532
1	1.822	93.7	1.398	93.6	0.699	2.707	94.0	0.349	0.033	1.371	0.021	1.014
2	3.714	94.7	2.246	94.7	1.123	4.389	95.0	0.561	0.053	2.071	0.033	1.565
3	7.162	95.4	4.828	95.4	2.414	12.413	95.7	1.207	0.294	4.967	0.204	4.399
4	13.472	97.5	7.078	97.5	3.539	23.567	97.6	1.769	0.486	9.018	0.546	8.021

As introduced in Section 1, AI-enabled embedded devices must support computationally demanding tasks at high energy efficiency. Our CNN design is parametric to find the best trade-off between power consumption and accuracy. We vary our architecture by changing the depth (d), i.e., the number of layers, and the width (w), i.e., the number of filters in each layer, following the compound scaling schema proposed in [205]. Unlike them, we do not scale the input size between configurations since the heartbeat duration constrains it.

The compound scaling proposed in [205] introduces the scaling factor ϕ and uses it, in turn, to define the scaling of depth $d = \alpha^{\phi}$ and width $w = \beta^{\phi}$ where α and β are constants following

$$\alpha * \beta^2 \approx 2 \tag{4.2}$$

Given our starting number of filters, we have selected $\beta = 1.33$ for a more straightforward scaling, and $\alpha = 1.15$ was selected accordingly based on the Formula 4.2. The number of MAC operations approximately doubles at each step, as depicted in Table 4.1.

Figure 4.3 depicts the full processing pipeline. As our baseline, we use a network with five layers. Each of the sub-blocks of the network structure comprises a 1D Convolutional kernel (detailed in Figure 4.4), Batch Normalization, ReLU activation, and Max Pooling. In the final step, the network includes a flattening followed by a Fully Connected (FC) layer. The first two convolutions have a kernel size of seven and five elements with a stride of two, while all the others have a kernel size of three and a stride of one. At each increase in the scaling factor, we introduce two new 1D convolutions; considering that the input size remains fixed, we do not introduce new pooling layers and set the stride of each convolution to one.

Due to the limited memory available on the target platform, we employ quantization techniques to reduce the memory footprint. We evaluated two quantization schemes based on 8-bit integer (INT8) or IEEE 754 16-bit floating-point (FP16) formats. Our results, shown in Table 4.1, indicate that the INT8 quantization outperforms the FP16 one in terms of accuracy, memory footprint, and latency. Considering both the NE16 support for INT8 and the higher energy demand for FP16, we focus on INT8 quantization in the remainder of this chapter. For the sake of completeness, Table 4.1 also reports

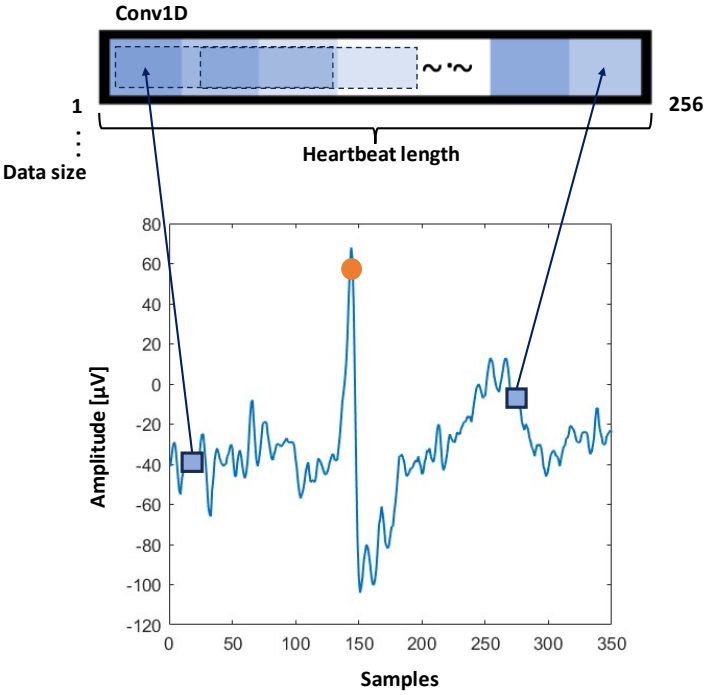


Figure 4.4: The details of the Conv1D kernel processing a single heartbeat segment (256 samples), with the R peak of the heartbeat highlighted. The dashed squares illustrate the sliding windows used in the convolution process.

the accuracy and memory footprint for the IEEE single-precision floating-point type (FLOAT32). As this data type is not supported by the deployment tool, we cannot provide its latency values.

4.2.4 Dataset

For ECG, we use the MIT-BIH Arrhythmia Database [173] for the experimental assessment. This dataset contains a set of two-channel ECG half-hour-long signals from 47 subjects sampled at 360 Hz. The MLII channel represents the signal recorded with the positive electrode placed on the left leg and the negative electrode on the right arm, while the V2 channel records a signal of a chest lead positioned on the fourth intercostal space at the right sternal border. We analyze the MLII channel because normal beats in the V2 channel may be difficult to distinguish [206]. Two or more cardiologists independently annotated each record. Following established literature practices, we adhered to the AAMI standard [207] to categorize arrhythmias into the five classification groups (N, S, V, F, and Q). Consistent with AAMI guidelines, records containing paced beats, specifically those labeled as "102", "104", "107", and "217", were excluded from both the training and test datasets.

Figure 4.5 displays heartbeats from three subjects classified into five distinct classes. The shapes of classes N and S are quite similar, often leading to confusion between them. Following the approach utilized in [208], we employ an input size of 256 samples centered on the R peak position within the QRS complex, as depicted in Figure 4.4. Considering the sampling frequency of the MIT-BIH Arrhythmia Database, the heartbeat corresponds to a duration of 0.71 s.

The ECG5000 dataset [209] is used for the experimental assessment of the TCN design, as discussed in Sections 4.5, to compare the performance with the primary reference in SoA [58]. This dataset is part of the BIDMC Congestive Heart Failure Database (chfdb) [210] [211]. It includes severe congestive heart failures labeled in five classes: Normal Signal (N), R-on-T Premature Ventricular Contraction (r), Supraventricular Premature or Ectopic beat (S), Premature Ventricular Contraction (V), and Unclassifiable Q). The data have been pre-processed to extract heartbeats and make each one equal in length using interpolation. We determined an input size of 140 samples for the TCN network based on the data characterization.

The dataset includes 5000 samples, split into 4500 for the test set and 500 for the training. Considering a time window including one heartbeat at a time, the TCN can effectively learn the temporal dependencies and patterns within the ECG signal. However, this dataset is imbalanced since some classes are under-represented. This can induce misclassification errors, and data techniques such as data augmentation and oversampling address this issue. In the subsection 4.5.2, we explain our mitigation based on oversampling.

4.2.5 Training Process

Networks are generated and trained using TensorFlow and Keras [212]. We divided the dataset into three subsets: training, validation, and test, with a split ratio of 7:1:2, as reported in [197]. We trained the network for 250 epochs using a batch size of 128, the Adam optimizer with a learning rate of $4 \cdot 10^{-4}$, and categorical cross-entropy loss. If the loss does not decrease for 10 epochs, we decrease the learning rate by halving it. Additionally, we employed early stopping based on the F1 score of the validation set, stopping training if it does not improve for 25 epochs. Then, we conduct fine-tuning for 10 epochs on the validation set using the Adam optimizer with a learning rate of $5 \cdot 10^{-6}$. As a data augmentation technique, we generated training data by randomly selecting 256 samples from windows of 300 samples centered on the R peaks.

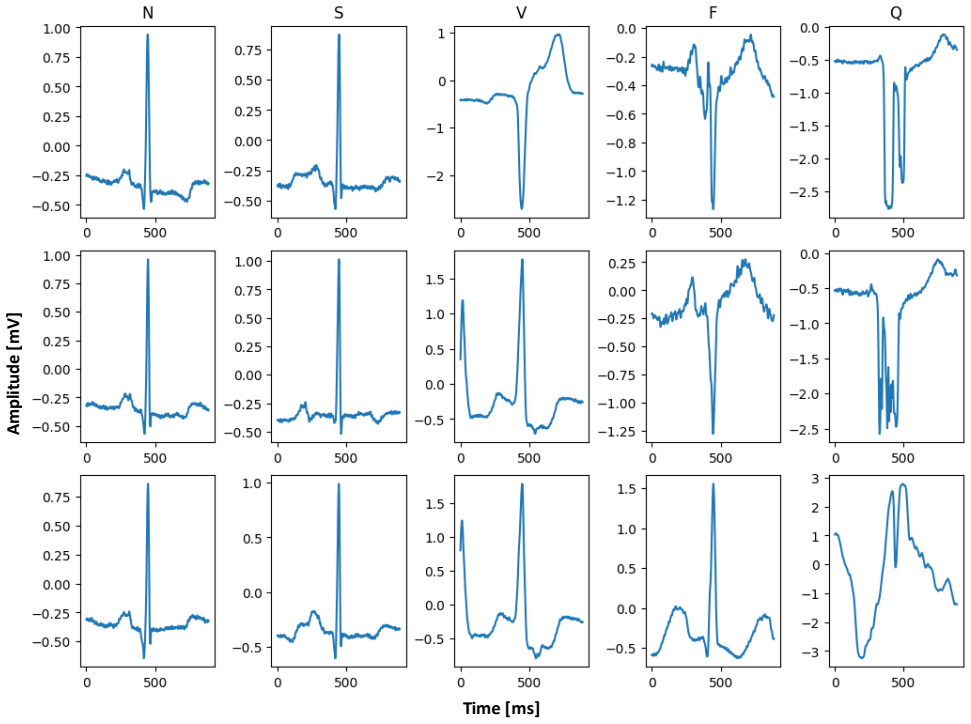
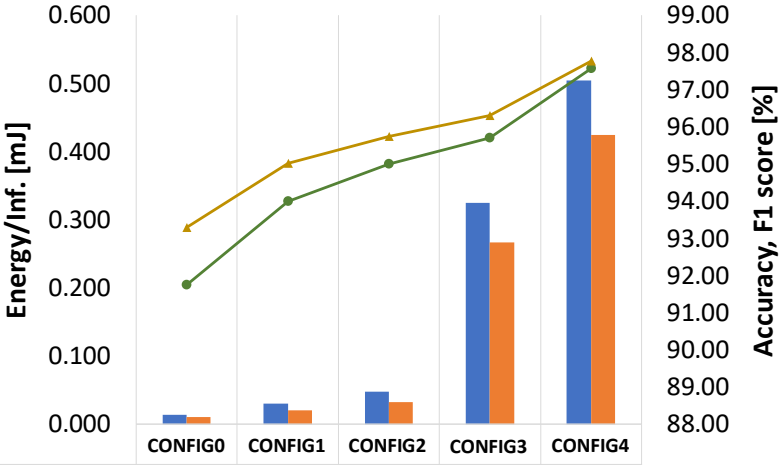


Figure 4.5: Variations in heartbeat patterns across five classes in three subjects taken randomly from the MIT-BIH Arrhythmia Database.

4.3 CNN Experimental setup

The system is configured to duty cycle between deep sleep and active power states (see Section 4.2.1). This design choice requires retrieving network weights before each inference phase. Adopting retentive sleep would also preserve data in the L2 memory; however, considering this application's timings, using deep sleep represents the most energy-efficient solution. Compared to other application domains (e.g., drone control), the time spent in a sleeping mode is higher. Consequently, the application must fetch the network weights from the MRAM to the L2 memory (network generation phase) before performing the inference (inference phase). Our preliminary experiments show that using retentive sleep would result in a $\sim$10$\times$ increase in power consumption, as indicated in the GAP9 datasheet [213], specifically 41 μW compared to 468 μW. These correspond to daily energy consumption values of 3.5424 J and 40.4352 J, respectively.

To report the energy consumption in Section 4.3.2, we consider three phases of the end-to-end application: R peaks detection, network generation, and inference. Power measurements were conducted using a Power Profiler Kit II (PPK2) connected to the GAP9 board, while execution time was measured using the performance counters available on the chip. Measurements have been repeated, setting the frequency at two alternative values (240 MHz and 370 MHz).



	CONFIG0	CONFIG1	CONFIG2	CONFIG3	CONFIG4
Energy/Inf. @370 MHz	0.014	0.030	0.049	0.325	0.504
Energy/Inf. @240 MHz	0.010	0.021	**0.033**	0.267	0.425
Accuracy	91.76	93.99	**95.01**	95.70	97.57
F1 score	93.30	95.03	**95.76**	96.30	97.77

Figure 4.6: Energy consumption across five configurations with *INT8-NE16*. At 240 MHz, CONFIG2 is the best trade-off between energy and accuracy.

In this study, we examine five architecture configurations (CONFIG0-1-2-3-4 based on the network size, from smallest to largest), described in Section 4.2.3.

4.3.1 Accuracy and energy consumption

This part aims to find the configuration that provides the best trade-off between accuracy and energy consumption [214]. Figure 4.6 shows that the accuracy and F1 score increase with the network complexity. As expected from intuition, this trend follows the increase in the scaling factor described in Section 4.2.3. Starting from CONFIG3, the total memory amount required to store weights and activations does not fit the L2 memory. To solve this issue, weights are stored in the MRAM and activations in the SDRAM, and both are tiled to L2 memory to perform inference. Indeed, Figure 4.6 shows a significant increase for CONFIG3 and CONFIG4 due to these effects.

Consistently with findings in [190], 240 MHz is the optimal operating frequency for minimizing energy consumption, and this choice does not affect the timing constraints. After these considerations, we designated CONFIG2 with *INT8-NE16* as the best trade-off between accuracy and energy efficiency. Figure 4.7 depicts the confusion matrix for this configuration.

4.3.2 Comparison with the state of the art

We compared the performance of our optimal solution (CONFIG2) with the current SoA, taking into account [190] as the primary reference; to the best of our knowledge, this is the most efficient alternative targeting the same platform.

The power consumption of PT, estimated at 0.013 mJ from the optimized PULP version [182], is constant across all variants. Based on the number of parameters reported in [190], we analyzed the network generation cost, assessing it to be two orders of magnitude lower than our solution (0.0005 mJ vs. 0.027 mJ). Conversely, our inference phase demonstrates superior efficiency compared to their (0.03 mJ vs. 0.09 mJ).

We observed that the inference phase is the major contributor to energy consumption. Specifically, based on 70 beats per minute requiring 100800 duty cycles between deep sleep and active states in 24 hours, our solution consumes 7413 mJ daily, while daily energy consumption for the Tiny Transformer [190] is 10433 mJ. In general, single-core MCU platforms (e.g., STM32 MCUs) are too constrained for complex neural architectures ([215]); [171] shows that PULP platforms are more computationally powerful and energy efficient. We achieve higher efficiency even in comparison with the study by [216], where they tested their porting to STM32.

Table 4.2 compares our solution with other SoA systems. Our solution achieves the best trade-off between Energy/Inference and accuracy. Our network is ∼3× more computationally demanding (in terms of MACs) than the Tiny Transformer [190]; however, we achieve ∼3× faster inference speed with only a 3.17 degradation in accuracy. We also compare with the ECG-TCN solution proposed by [58], showcasing the higher efficiency of our approach (∼3.33× in terms of Energy/Inference). Furthermore, we overcome [57] in accuracy (95.01% vs. 93.16%) with similar energy consumption (0.03 mJ vs. 0.04 mJ). Both [58] and [57] use a different dataset (i.e., ECG5000), which could introduce a bias in the comparison. Lastly, our solution achieves higher accuracy than an LSTM-based approach with an equivalent computational load proposed by [216] (95.0% vs. 90.2%).

4.4 Conclusion and Future work

This chapter introduces a processing pipeline to detect pathological conditions in the ECG signal using a methodology based on a preprocessing stage followed by a parametric CNN. Our goal was not merely to surpass the accuracy of the SoA solutions but to find an optimal trade-off between accuracy and energy consumption. To this extent, the experimental assessment was performed on the GAP9 PULP platform to

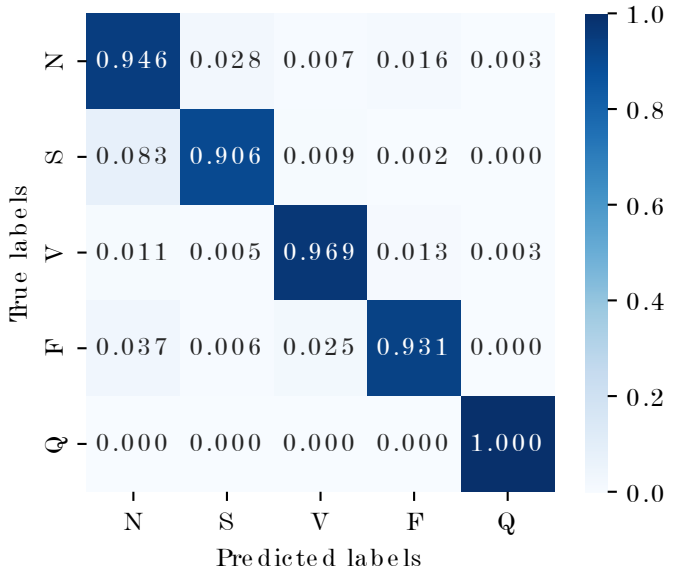


Figure 4.7: Confusion matrix that displays the probability of true positives within the test set for CONFIG2 with INT8-NE16.

understand the interdependencies between software parameters (e.g., number of CNN layers) and hardware constraints (e.g., limited memory). As a result, we found a solution whose accuracy is marginally lower than Transformer-based solutions but improves energy efficiency by 3×. Future work will focus on extending the design knobs to platform architectural parameters (e.g., external memory size, FPU support, number of cores), promoting a hardware-software co-design methodology. Finally, we will also explore Transformer models with adjustable parameters.

Table 4.2: Performance comparison with the SoA.

	Platform	**ISA**	**Technology [nm]**	**Freq. [MHz]**	**MACs**	**Latency [ms]**	**MACs/ Cycle**	**Energy/ Inf. [mJ]**	**Power [mW]**	**Accuracy [%]**
Tiny Transformer [190]	GAP9	RV32ICMX + Spec. Ext.	22	240	1109606	4.28	0.97	0.09	**20.33**	**98.97**
RNN [216]	Cortex-M4	ARMv7-M	90	84	**761200**	148.54	N/A	N/A	N/A	87.0
LSTM [216]					3058000	665.86				90.2
ECG-TCN [a] [58]	GAP8	RV32ICMF + Spec. Ext.	40	100	1030260	2.7	3.795	0.10	38.52	93.8
NAS-based TCN [57]	GAP8	RV32ICMF + Spec. Ext.	40	100	146750[b]	**0.78**	1.9[b]	0.04	51[c]	93.16
This work	GAP9	RV32ICMX + Spec. Ext.	22	240	3698688	1.57	**9.85**	**0.03**	21.28	95.0

[a]Deployment framework: NEMO/DORY. Dataset: ECG5000, 5 classes.
[b]MACs computed dividing by 2 the number of reported operations (OPs).
[c]Estimated using the energy consumption and latency.

4.5 TCN Methodology

4.5.1 TCN architecture

As regards the TCN, this section proposes the simplified network topology w.r.t. the recent work on TCNs of Bai et al. [184].

Each of the seven sub-blocks of the network structure contains a Convolutional kernel, Batch Normalization, ReLu activation, and Max Pooling. Finally, the network applies the inference based on the whole temporal information, using a Fully Connected (FC) layer followed by a flattening and a Softmax. To improve the performance of the TCN model, we apply a variable dilation to increase the receptive field. Also, to ensure causal convolution and to maintain the output sequence with the same length as the input sequence, we apply zero-padding only on the left side of the input feature map/tensor. This configuration is called *causal padding*. Considering stride = 1, we compute the padding size for each block as:

$$p_i = (KernelSize_i - 1) * d_i; \tag{4.3}$$

where i is the i-th block, and d is the variable dilation factor.

Due to the compact size of the TCN model, we do not need to employ heavy quantization techniques to save memory space at the cost of reduced accuracy. We adopt the IEEE 754 16-bit format (FP16) that reduces the memory footprint and computation requirements while preserving the model accuracy [217]. This approach allows to efficiently deploy and run the TCN on the hardware platform, ensuring good performance and resource utilization, as described in Section 4.6.

4.5.2 Training

Data imbalance is prevalent in biomedical datasets where certain events or conditions of interest are underrepresented with a few (or even just a single) occurrences. This scarcity of data for specific classes can severely impact the performance of machine learning models, particularly in classification tasks, as the model may become biased towards the majority class. Oversampling and data augmentation are two common approaches to mitigate the issue of data imbalance.

Data augmentation techniques create new training examples by applying various transformations to the existing data, such as rotations, flips, or translations. As a result,

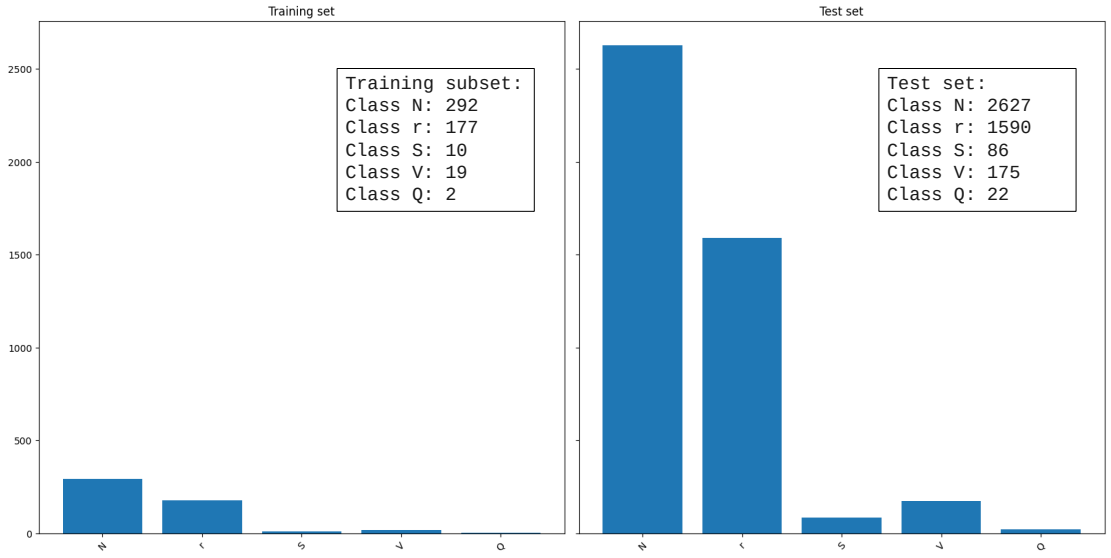


Figure 4.8: Oversampling and mapping of dataset labels. Train data and test data.

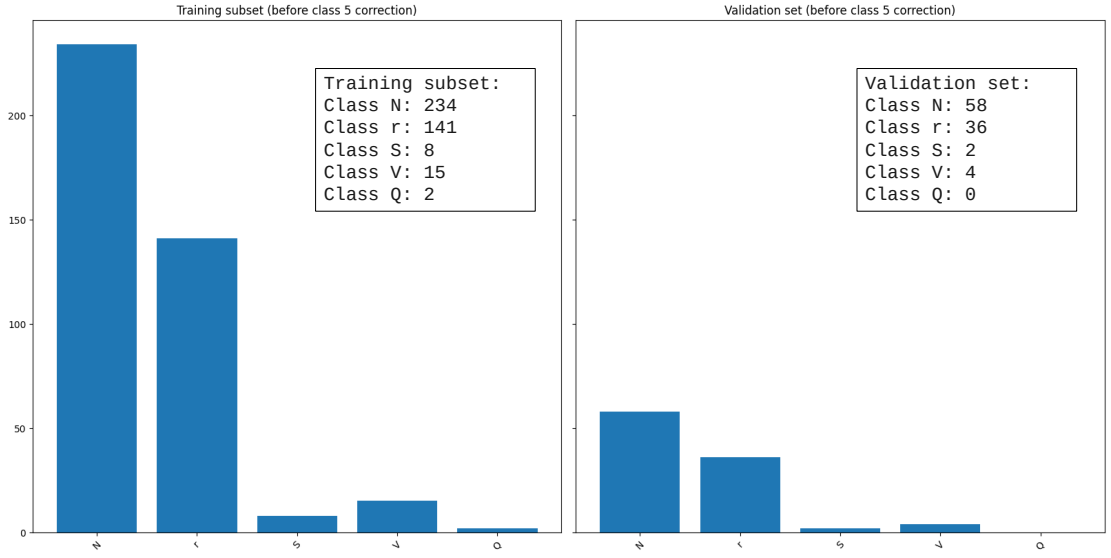


Figure 4.9: Oversampling and mapping of dataset labels. Training data is split into training subset and validation data.

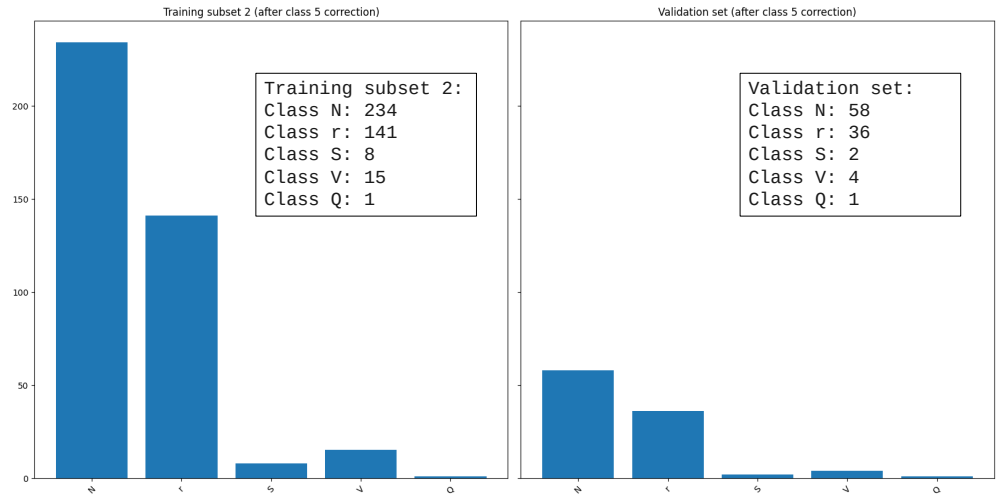


Figure 4.10: Oversampling and mapping of dataset labels. Class 5 (Q) correction. Class 5 is in the training subset and validation data.

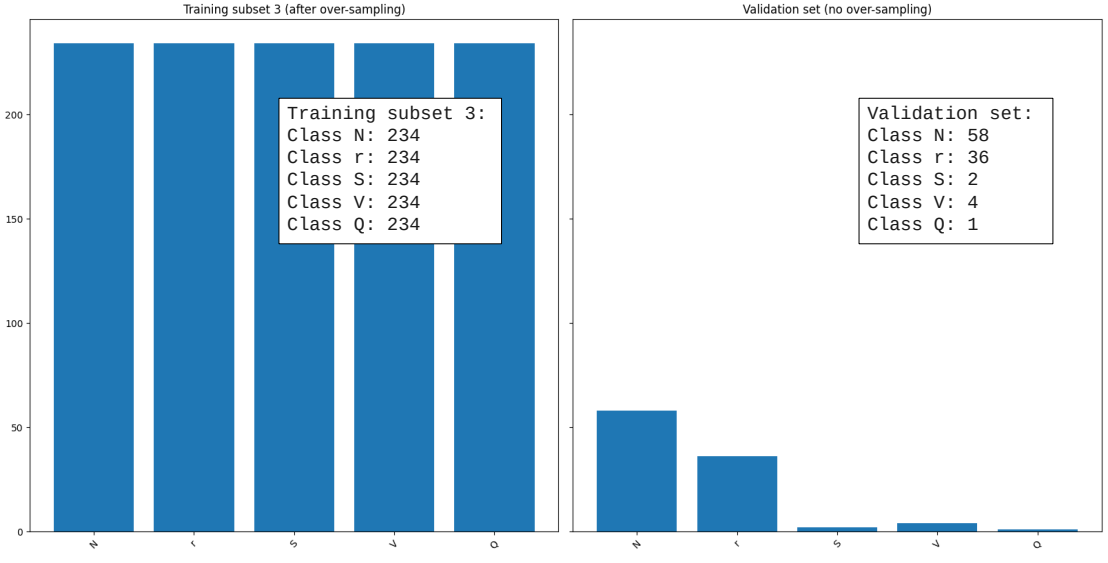


Figure 4.11: Oversampling and mapping of dataset labels. Oversampling on the training subset.

more data are available for training, and the models become more robust and less prone to overfitting. However, since the data in this application are highly dependent on the shape of the ECG trace, techniques that add signal distortion are prone to degrading performance in real use cases.

On the other hand, oversampling involves increasing the number of instances in the minority class. One standard method is to duplicate existing examples in the minority class. As illustrated in Figure 4.8, a noticeable data imbalance emerges in our dataset, particularly concerning the last class, which contains only two occurrences. To address this imbalance, we partitioned the dataset into a training subset (80%) and a validation set (20%), as depicted in Figure 4.9. This partition resulted in 400 samples for training and 100 samples for validation.

In Figure 4.10, our focus turns to class Q (class 5), which experiences significant underrepresentation. We employed an oversampling technique to mitigate this issue, as shown in Figure 4.11. This method involves artificially augmenting the number of samples in the minority class by duplicating existing data points. This approach effectively balances the class distribution.

In this experiment, the ML framework is based on TensorFlow and Keras [212]. For the TCN training, we applied a batch size of 32 samples for 100 epochs with an Adam optimizer and categorical cross-entropy loss. Instead of performing a single run or training session of the model, we run the model training ten times, providing a more stable and robust average performance estimate.

Furthermore, we compute the *validation sample weight* to improve the validation accuracy. The *validation sample weight* refers to a parameter we can use when training a model to assign different weights to individual samples in our validation set. These weights affect the contribution of each sample to the overall validation loss during training. It is a way to control the importance of different validation samples when computing the validation loss.

In summary, we follow these processing steps

1. Data Preparation
    - Train/validation split.
    - Handling of minority class Q.
    - Oversampling of the training subset.
2. TCN model training with repetitions (runs = 10).
    - Calculation of the *validation sample weight*.

Table 4.3: Accuracy, Balanced Accuracy, and Number of Parameters on SoA.

	Ingolfsson et al. [58] (NEMO/DORY)	**This work**
Accuracy [%]	92.8	91.2
Balanced Accuracy [%]	94.8	93.8
#Parameters	10K	4K

3. Estimation of the best validation accuracy between the models.

The GAP9 software ecosystem comprises *NNTool* [127], a framework to deploy NN graphs on embedded platforms starting from its ONNX representation that can be exported by ML frameworks. The current version of NNTool only supports 2D convolutions, while TCNs require 1D convolutions; we overcome this limitation by considering a second EGG channel as the depth dimension.

4.6 TCN Results

Table 4.3 reports the computation of the accuracy and parameters, compared with the current SoA for ECG anomaly detection on embedded devices [58]. The *accuracy* metric is computed as the ratio between true positives (TP) for that class and the total number of instances (support). It indicates how well the model correctly predicts instances of a given class. Since the ECG5000 dataset is highly imbalanced, we also consider a *balanced accuracy* metric, calculated as the weighted average between the Sensitivity and Specificity. Weighted average considers the class distribution, giving higher weights to the minority class. Hence, it provides a more stable and reliable estimate of the model performance when dealing with class imbalance.

For a fair comparison, we execute the ECG-TCN, replicating the experiment described in [218] to compute their *balanced accuracy* using the NEMO/DORY configuration. Table 4.3 shows that our model achieves a balanced accuracy of 93.8%, comparable with [218], and an accuracy of 91.2%, even though it reduces the number of parameters, with a beneficial impact on energy consumption.

Table 4.4 reports the energy consumption compared with [218], considering the same input size of 140 samples. The time per inference of our solution is 7.3× faster than the methodology proposed by Ingolfsson et al. [58]. Considering MACs as a performance metric, which does not depend on the technology node, our implementation results are 9.1× faster. This effect is primarily due to the complexity reduction of the net topology and directly impacts battery lifetime.

Moreover, implementation on the GAP9 processor features only 0.01 mJ per inference, maintaining a floating point data representation for input data and intermediate layers. Deploying the algorithm on an end-to-end platform with a commercial analog front-end (AFE) for ECG (e.g., MAX 30003, which consumes 85 μW), the battery lifetime of the whole system (AFE + GAP9) reaches 6316 hours, compared to 5310 hours in [58], corresponding to a 19% improvement.

As mentioned at the beginning of Section 4.1, the implementation of the CNN overcomes the TCN performance in terms of MACs/cycles and power consumption and achieves an accuracy of 95% (Table 4.2 in Section 4.3.2)) vs. 91.2%, respectively.

4.7 CNN vs TCN discussion

As outlined in Section 4.1, we selected CNN over TCN as the optimal solution for this application. CNNs, which leverage convolutional layers to extract spatial features, can process entire data sequences without enforcing temporal causality. This capability simplifies the model, resulting in enhanced classification performance for our purposes since the CNN does not depend on time constraints inherent to TCN models.

In this study, there's no need to enforce TCN's *causality* property, as all temporal dependencies are already met by the pre-processing (PT) stage, which pinpoints the relevant interval in the ECG signal for the CNN. The PT algorithm requires data windows that capture information after each heartbeat so the network can operate on complete data without maintaining causality constraints at inference time.

Although CNNs incur a slightly higher energy cost per inference (0.049 mJ) compared to TCNs (0.01 mJ) at an operational frequency of 370 MHz, they achieve higher accuracy (95% vs. 91.2%) and lower computational complexity, making them more advantageous for energy-efficient embedded edge applications.

Chapter 4 refers to publication [219].

Table 4.4: Energy consumption compared with SoA solution considering one heartbeat as input of the network.

	Platform Architecture	**ISA**	**Technology [nm]**	**Operating frequency [MHz]**	**MACs**	**Time/inference [ms]**	**MACs/Cycle**	**Energy/inference [mJ]**	**Power [mW]**
Ingolfsson et al. [58] (NEMO-DORY)	GAP8	RV32ICM + Spec. Ext. (Xpulpv2)	40	100	1030260	2.7	3.795	0.10	38.52
Our work	GAP9	RV32ICMF + Spec. Ext. (Xpulpv3)	22	370	**112873**	**0.37**	**0.899**	**0.01**	**36.8**

Chapter 5

EEG Applications

The last category of biosignals explored in this thesis is the electroencephalogram (EEG) as introduced in Section 2.1.2. This dissertation prioritizes the evaluation of ear-EEG sensors in terms of feasibility and signal quality rather than focusing solely on optimizing offline processing. Specifically, the study develops a system that adheres to the Auditory Steady-State Response (ASSR) protocol and includes an analysis of alpha waves (AW).

For the validation measurements, data is collected and processed offline using two distinct approaches: (i) AW analysis via spectrogram computation and (ii) automated peak detection of the response at the 80 Hz amplitude-modulated frequency, which follows a Power Spectral Density (PSD) analysis. This methodology allows for a comprehensive assessment of the ear-EEG system's performance in capturing relevant neural activities related to auditory stimuli.

5.1 Ear-EEG Description

As reported in Chapter 1, wearable devices are particularly valuable in healthcare and wellness, as they enable monitoring of health metrics in everyday environments beyond the confines of clinical settings [220]. Ear-worn wearable devices are particularly appealing for physiological sensing due to their unique positioning on the head, enabling the unobtrusive monitoring of various physiological signals. Designed to resemble common audio devices, such as earphones or earbuds, earables are discreet and convenient. The ear's anatomical structure also provides a stable and accessible location to house the necessary electronics for a wearable sensor, making earables an ideal platform for continuous, comfortable physiological monitoring [221]. EEG is traditionally captured

by placing electrodes on the scalp, a procedure usually done in clinical settings since electrode positioning and preparation are critical for data accuracy and usability. In 2011, ear-centered EEG systems were introduced as a more convenient alternative, allowing for EEG measurement from around the ear region, which simplifies setup and enhances user comfort without compromising data quality [72]. Ear-centered EEG devices are user-friendly and require significantly less setup time than traditional EEG, which involves placing multiple electrodes on the scalp by trained personnel.

In this chapter, I present the characterization and testing of a custom sensor developed by Dätwyler Schweiz AG [222] in collaboration with ETH Zürich during my internship. The focus of this work involves applying the Auditory Steady-State Response (ASSR) protocol, particularly targeting frequencies of 80 Hz and 88 Hz, along with alpha-band modulation (7-13 Hz).

5.2 ASSR Related Work

The wide use of the ASSR as an objective method for measuring hearing in clinical settings is due to its high-frequency specificity. It can accurately assess hearing ability at specific sound frequencies without relying on the patient's subjective responses. The recruitment phenomenon (a condition associated with hearing loss where soft sounds are not heard, but loud sounds are perceived as disproportionately more audible than they are) is typically detected through subjective evaluations, which require direct communication with the patient. If the recruitment phenomenon could be detected using ASSR, it would simplify diagnosing patients who cannot communicate effectively, such as those with developmental disorders or infants. This is because ASSR provides objective measurements that do not depend on active responses from the patient [223].

Kidmose et al. [49] conducted a study testing four different Evoked-Related Potentials (ERP) paradigms—ASSR, steady-state visual evoked response, an auditory-evoked P1–N1–P2 complex, and a visual-evoked onset response—using ear-EEG on a group of healthy subjects. The findings revealed that the ear-EEG signal amplitude was typically around 20 dB lower than the scalp EEG recorded over the temporal cortex. To assess ear-EEG performance in real-life settings, Kappel and Kidmose [46] compared a dry-contact ear-EEG electrode with a traditional wet electrode scalp EEG across four paradigms (ASSR, SSVEP, auditory onset response, and alpha-band modulation) and in both controlled and real-life environments. They found a high level of similarity between ear-EEG and scalp EEG results, particularly when referencing the Cz electrode on the scalp. Although statistically significant responses were only noted for the ASSR in both environments when using within-ear-referenced electrodes, the results support

Figure 5.1: Ear-EEG electrode fabricated by Dätwyler Schweiz AG and connected to an active buffering PCB

ear-EEG's reliability in everyday contexts. To further explore this application, Paul et al. [224] introduced a fully integrated in-ear EEG prototype with Ag/AgCl dry electrodes designed to both elicit and detect ASSR responses, highlighting the potential of ear-EEG for convenient auditory monitoring.

Christensen et al. [225] further validated the potential of ear-EEG-based ASSR as a hearing assessment tool. In their study, the researchers found that, in response to broadband chirp stimuli with repetition rates ranging from 20 to 95 Hz, the ASSR amplitude recorded with ear-EEG was generally lower across most frequencies than conventional scalp-EEG recordings.

Finally, Fiedler et al. [226] also identified ear-EEG as a promising approach for capturing neural responses during a range of auditory stimuli, including discrete, dichotic, and continuous biotic sounds. When comparing their in-ear EEG device with signals from a 64-electrode scalp EEG system, they observed that event-related potentials (ERP) obtained through ear-EEG highly depended on the reference configuration. Their findings support the feasibility of integrating ear-EEG technology into hearing aids for effective neural activity monitoring.

5.3 Methodology

5.3.1 Setup Description

The electrodes used in this setup are ear-EEG electrodes as depicted in Figure 5.1, fabricated by Dätwyler Schweiz AG [222]. These specialized electrodes are designed to be placed in the ear to capture EEG signals with minimal interference from ambient noise. The ear-EEG electrodes offer a more comfortable and less intrusive alternative to traditional scalp electrodes, making them suitable for long-term monitoring and testing

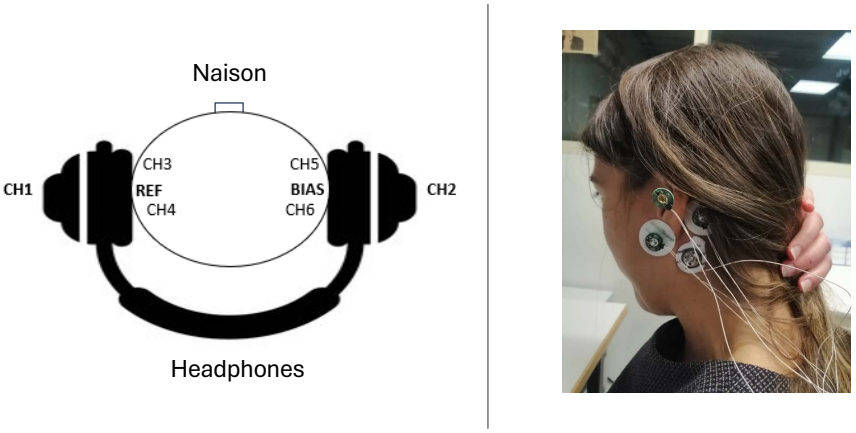


Figure 5.2: Channels placement and electrodes configuration on a test subject.

in both clinical and real-life settings. The use of these electrodes enhances the ability to detect neural responses to auditory stimuli while maintaining the participant's comfort during the procedure.

In these ASSR experiments, stimuli are delivered using either over-the-head or bone-conduction headphones. This choice of headphones is significant, as both options offer distinct advantages for auditory testing. Over-the-head headphones are typically used for their comfort and ability to deliver clear sound, making them suitable for prolonged listening sessions. On the other hand, bone-conduction headphones transmit sound vibrations directly through the skull to the inner ear, which is particularly beneficial in scenarios where traditional headphones may not be effective, such as for individuals with conductive hearing loss [227].

To compare the signal quality, we also used traditional wet (gel-based) electrodes around the ear, as benchmarks. Figure 5.2 and Table 5.1 show the placement and the type of the electrodes and their corresponding channels.

The data is stored in the following format: *'N_yyyy_mm_dd_PROTOCOL'*, where *N* represents the subject number, *PROTOCOL* is the protocol name AW or ASSR for alpha waves or auditory steady-state response, respectively, and the date of the data acquisition is formatted as *year_month_day*.

5.3.2 Data Acquisition

The acquisition platform used in this setup is BioGAP [228] (see Figure 5.3) a PULP-based architecture, specifically the GAP9 microcontroller discussed in Section 2.3, which features powerful TOPS (tera operations per second) capabilities and includes 10

Table 5.1: Electrodes allocation and types.

Channel	Location	Type
CH1	In-ear (left)	Ear-EEG electrode + active buffering
CH2	In-ear (right)	Ear-EEG electrode + active buffering
CH3	Front of the ear (left)	Wet electrode + active buffering
CH4	Back of the ear (left)	Wet electrode + active buffering
CH5	Front of the ear (right)	Wet electrode + active buffering
CH6	Back of the ear (right)	Wet electrode + active buffering
REF	Mastoid (left)	Wet electrode + active buffering
BIAS	Mastoid (right)	Wet electrode

RISC-V cores optimized for energy-efficient, high-performance computation tailored for tinyML applications. This architecture allows for robust, parallel processing suitable for low-power embedded applications such as biosignal processing.

The platform is equipped with a Nordic nRF52 module for connectivity, enabling efficient Bluetooth Low Energy (BLE) communication, essential for wireless data transfer in wearable applications. Additionally, the platform offers versatile interfacing with a range of sensor inputs, making it adaptable to different biosignal acquisition setups.

The platform integrates an ADC (Analog-to-Digital Converter) module to capture biopotential signals, specifically the ADS1298. This module provides 8 ExG channels with 24-bit resolution, allowing for precise, high-quality signal acquisition across multiple channels, which is ideal for applications requiring detailed and accurate biosignal data.

The EEG data acquired by the system is continuously transmitted to a PC for real-time visualization and is saved in a log file for further analysis. After initialization, BioGAP establishes a Bluetooth Low-Energy (BLE) connection with the receiver dongle. It transitions into a low-power BLE-connected state, awaiting further instructions. Real-time measurements are displayed using a Java GUI, which also sends commands (e.g., switch between streaming and sleep mode; start or stop measurement) and parameters (e.g., number of active EEG channels, sampling frequency, EEG gain). When the measurement is complete, it returns to the connected state. If inactive, BioGAP can enter sleep mode through a BLE command to conserve power and can be reactivated with a double-tap gesture.

5.4 Alpha Waves Validation Measurements

This validation procedure is designed to assess the presence and power of AW (7–13 Hz) in EEG data using a controlled protocol in which subjects alternate between eyes-open and eyes-closed states every 10 seconds. This alternating pattern aids in identifying

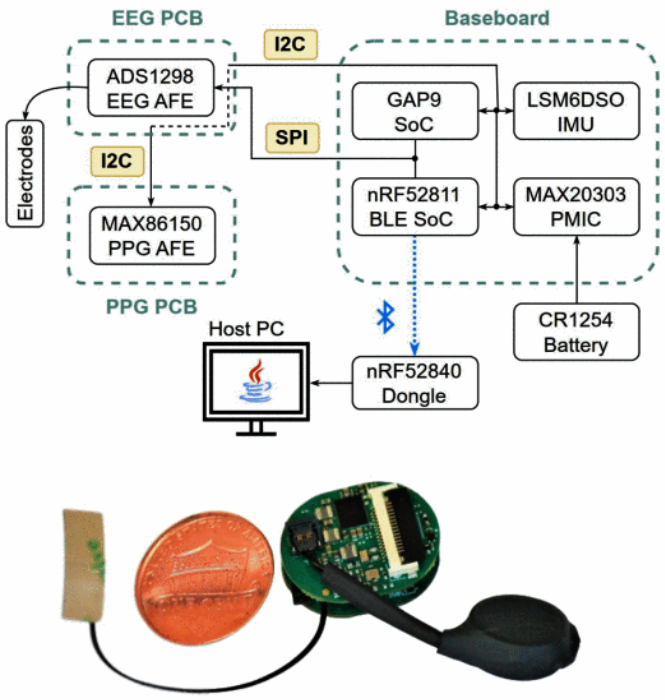


Figure 5.3: System diagram (top) and photo (bottom) of the BioGAP platform next to a one-cent coin. Image source: [228]

changes in alpha wave power associated with eye closure, typically observed as a natural increase in the 7–13 Hz frequency range when the eyes are closed.

5.4.1 Alpha Waves Processing

The raw EEG signals are preprocessed through a series of filters followed by spectral analysis, with the processing code developed in MATLAB to optimize data handling and analysis:

- Band-Pass Filtering: A 10th-order Butterworth filter with cutoff frequencies set at 0.5 Hz and 40 Hz isolates the relevant EEG frequency range, removing very low and high-frequency noise while retaining most of the alpha-band activity.
- Notch Filtering: To suppress 50 Hz powerline noise, a notch filter of order 2 is applied. This step is crucial in minimizing electrical interference that could distort the analysis of alpha frequencies.
- Spectrogram Computation: a sliding-window spectrogram is generated to visualize the temporal evolution of signal power in different frequency bands. Each window contains 1024 samples (2 seconds), with a 768-sample (1.5 seconds) overlap, providing a near-continuous display of frequency power over time. This approach enables a dynamic analysis of the alpha wave intensity, particularly in response

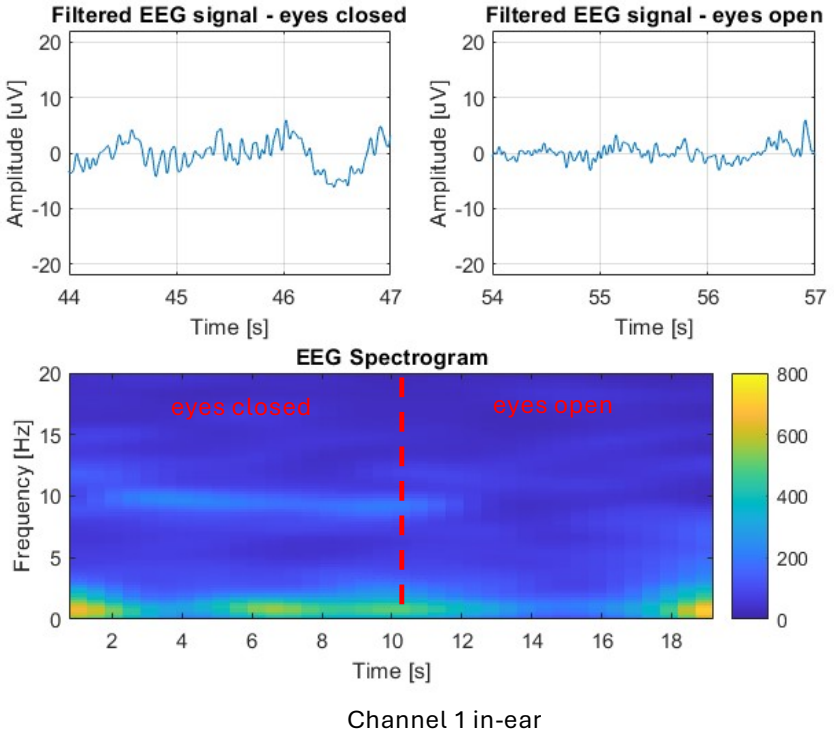


Figure 5.4: Measurement of the alpha wave activity in eyes closed vs eyes open condition of CH1 in-ear. Filtered signal in the time domain (TOP) and spectrogram (BOTTOM).

to the open- and closed-eye conditions, supporting insights into how alpha wave power fluctuates in a practical, non-invasive setup.

This methodology provides a reliable system for AW validation, which is crucial for applications involving EEG analysis in wearable or non-traditional electrode placements.

5.4.2 Results

Figures 5.4 and 5.5 (TOP) illustrate the EEG response from one subject of both in-ear channels (CH1 and CH2, respectively) in the time domain, showing distinct amplitude variations between the open- and closed-eye conditions. Meanwhile, the same figures at the bottom present a spectrogram computed for a signal segment during an eye transition, where a high-energy component in the alpha band is noticeable during the closed-eye phase and declines significantly as the subject opens their eyes.

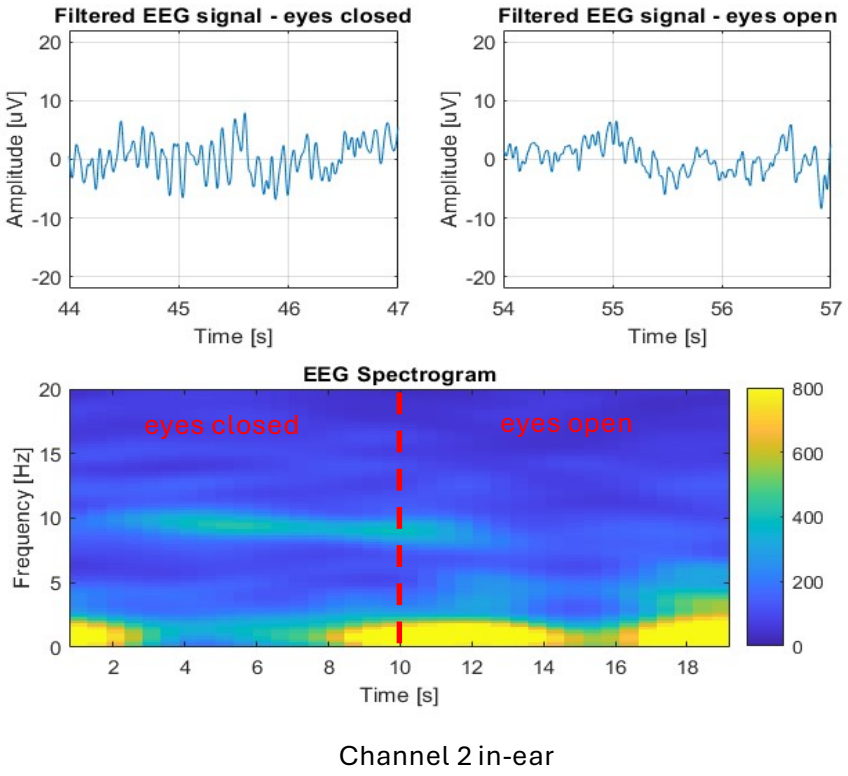


Figure 5.5: Measurement of the alpha wave activity in eyes closed vs eyes open condition of CH2 in-ear. Filtered signal in the time domain (TOP) and spectrogram (BOTTOM).

5.5 ASSR Validation Measurements

5.5.1 ASSR Protocol

This study uses an ASSR protocol to assess the feasibility of ear-EEG measurements for auditory stimulus detection. The protocol was conducted with five healthy volunteers, each outfitted with in-ear electrodes and headphones. During each session, subjects were allowed to read to create a natural setting, which minimizes artificial conditions while maintaining consistent auditory stimulation.

The auditory stimulus was presented through over-the-ear headphones, consisting of a sinusoidal tone with a 4 kHz carrier frequency [14], [223] modulated at two different frequencies, 80 Hz and 88 Hz, based on common testing standards in auditory research studies. The software we developed using PsychoPy [229] allows the user to customize the duration of the tone, carrier frequency, and modulation frequency. The intensity of the auditory stimulus was set at 65 dB SPL (Sound Pressure Level), with each acquisition session lasting 15 minutes, consistent with standard EEG protocols for auditory testing

(stimuli). In addition, a silent 15-minute session was conducted to observe and compare responses in the absence of stimuli (rest), following similar protocol setups [230].

EEG data was acquired using the BioGAP platform at a sampling frequency of 500 Hz, sufficient to capture the ASSR frequencies of interest accurately. This protocol aimed to detect the brain's steady-state response to these modulated auditory stimuli, providing insights into the performance of ear-EEG for real-world auditory monitoring applications.

5.5.2 Signal Processing Pipeline for ASSR Analysis

The signal processing pipeline for analyzing the ASSR in EEG data includes several key steps designed to filter, analyze, and quantify the auditory response. I implemented the processing for this analysis in MATLAB. Below is an outline of each stage in the pipeline:

1. **Filtering**
    - Band-pass Filtering: A 10th-order Butterworth filter is applied to retain frequencies between 30 and 180 Hz, isolating the desired range that captures relevant ASSR frequencies while excluding lower and higher frequency noise.
    - Notch Filtering: To reduce interference from power line noise and its harmonics, two notch filters are applied at 50 Hz (for Power Line Interference, PLI) with an order of 2 and a quality factor of 100, and at 100 Hz (second harmonic of PLI) with the same parameters.
2. **Power Spectral Density (PSD) Analysis using Welch's Method**
    - Windowing: Segments the signal into 5-second windows.
    - Discrete Fourier Transform (DFT): Computes the periodogram for each window.
    - Power Spectrum Estimation: The squared magnitude of the DFT yields power estimates for each segment.
    - Averaging: The individual power spectra are averaged across segments to minimize noise, producing an array of power measurements versus frequency bins, resulting in a robust estimate of the power distribution over frequencies.
3. **Peak Detection and SNR Estimation**

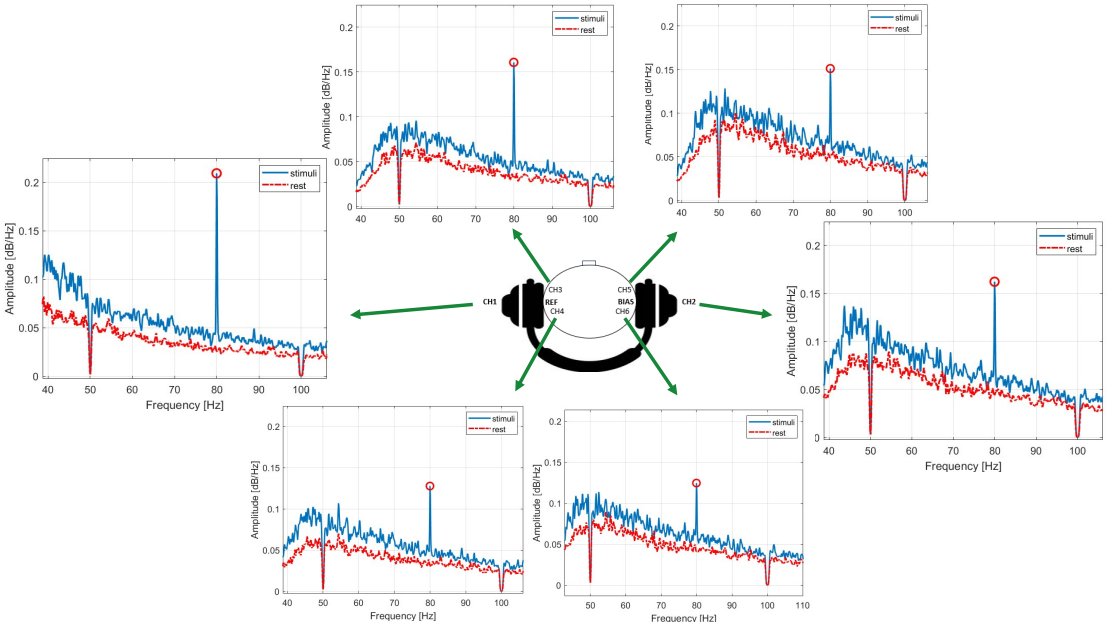


Figure 5.6: Power spectra at 80 Hz AM stimuli across all channels. CH1 and CH2 represent the in-ear channels of primary interest.

- Automatic Peak Detection: Detects the peak in the power spectrum at the amplitude-modulated (AM) frequency (80 Hz and 88 Hz) corresponding to the ASSR.
- Signal-to-Noise Ratio (SNR) Estimation: An automatic estimation of SNR is performed to assess the clarity and significance of the response signal relative to background noise.

This structured approach facilitates the extraction of clean and interpretable ASSR signals from the raw EEG data, which is essential for evaluating auditory responses efficiently and reliably.

5.5.3 ASSR Results

Figure 5.6 presents the power spectra for a single subject across all eight channels, highlighting a prominent peak at the AM frequency of 80 Hz. This peak is visible in each channel, with higher amplitude observed in the two ear-EEG channels, emphasizing their effectiveness in capturing the ASSR signal.

In Figure 5.7, we also compared the power spectrum at 80 Hz with a secondary frequency at 88 Hz. The peak was distinctly visible at both frequencies, particularly in the two in-ear channels, indicating a consistent and strong response for the ear-EEG channels across different AM frequencies. This result supports the effectiveness of

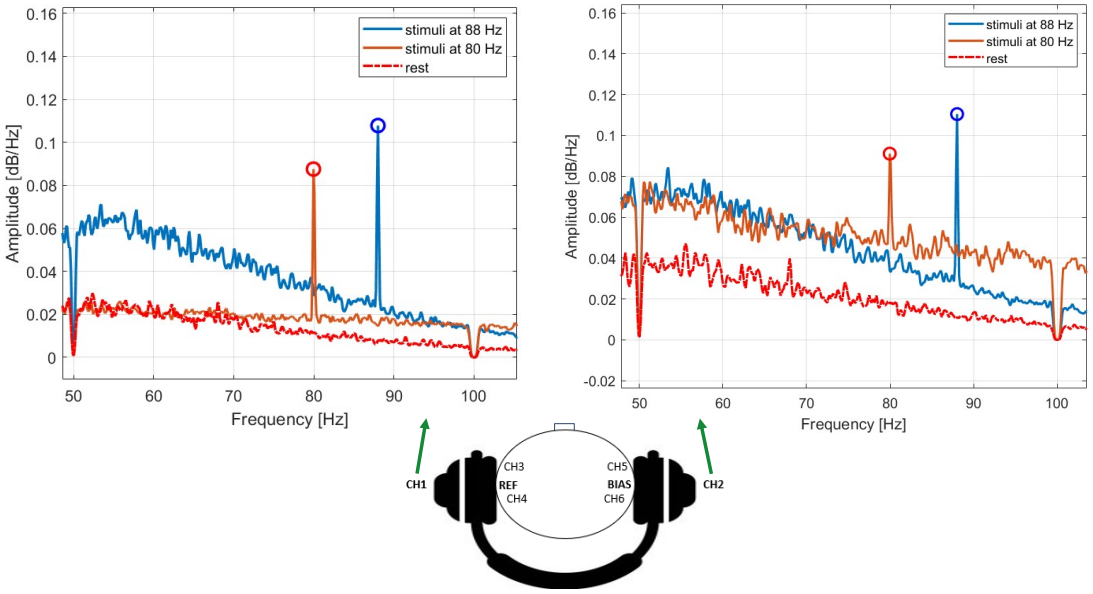


Figure 5.7: In-ear CH1 and CH2 power spectra comparison at 80 and 88 Hz AM stimuli.

the in-ear setup for capturing auditory steady-state responses across multiple stimulus frequencies.

Finally, we compared signal quality using bone-conduction headphones, as illustrated in Figure 5.8. As discussed in Section 5.3.1, bone-conduction headphones transmit sound vibrations directly through the skull to the inner ear, which can be especially useful in situations where conventional headphones are less effective, such as for individuals with conductive hearing loss. This method bypasses the outer and middle ear, allowing the user to perceive sound even if those areas are impaired. Furthermore, bone conduction is a promising alternative for minimizing overlap and reducing discomfort caused by the combination of an in-ear sensor and over-the-ear headphones.

5.6 Future Work

In future work, a significant focus could be placed on optimizing signal processing techniques specifically for the ASSR and perform AW analysis on low-power, embedded systems. The current study prioritized setup verification and assessed the feasibility and signal quality of ear-EEG, leaving considerable potential for computational improvements. One of the primary computational elements, the Welch method for Power Spectral Density (PSD) estimation, is particularly demanding in terms of processing power due to its requirement for extensive Fourier transformations and averaging across

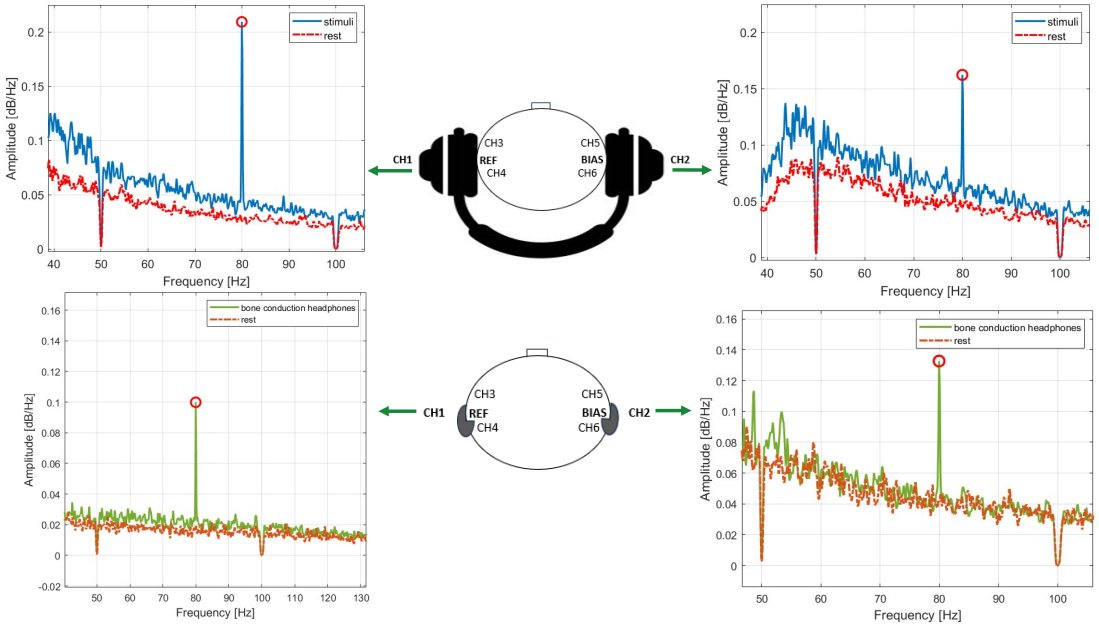


Figure 5.8: In-ear CH1 and CH2 signal power spectra comparison between over-the-ear (TOP) and bone-conduction headphones (BOTTOM) at 80 Hz AM stimuli and rest condition.

multiple windows. Optimizing this method at the software level, such as with a low-complexity Welch method, could substantially reduce computational load and energy consumption, making real-time analysis more feasible on embedded platforms such as the BioGAP system used in this study [231] .

Furthermore, future work could explore the integration of machine learning techniques optimized for edge computing to facilitate real-time classification of alpha wave activity or response to ASSR stimuli. This could enable more sophisticated monitoring applications in wearable form factors, such as continuous monitoring for auditory or neurological assessments, without the need for high-power computational resources. Lastly, evaluating the potential of bone-conduction headphones as an alternative stimulus delivery method would address any discomfort arising from overlapping ear sensors and traditional headphones, potentially broadening the use case for individuals with conductive hearing impairments. This line of inquiry would contribute to establishing a more accessible, adaptable, and power-efficient ear-EEG-based platform.

Chapter 6

Conclusion

This thesis explores the design and optimization of ultra-low-power systems for real-time biosignal processing, focusing on enabling efficient on-device computation for wearable and implantable applications. By addressing key challenges in signal acquisition, digital processing, and algorithmic optimization, this research contributes to the field by developing comprehensive methodologies for energy-efficient and high-performance biosignal analysis.

Through a multi-faceted approach encompassing optimized Analog Front Ends (AFEs), digital processing techniques, and adapted machine learning algorithms, the work demonstrates that achieving real-time processing on constrained devices is feasible without sacrificing accuracy or reliability. The RISC-V Parallel Processing Ultra-Low Power (PULP) platform, with its near-threshold computing (NTC) capabilities, proved essential in achieving these goals by balancing processing power and energy efficiency. This system's ability to handle complex tasks within a stringent energy budget showcases its applicability across various biomedical applications.

The two case studies presented highlight the real-world implications and adaptability of the developed methodologies. The first case study, focused on ECG processing, illustrates the advantages of on-device signal analysis, such as reduced latency, enhanced privacy, and higher responsiveness in real-time monitoring applications. The second case study examines the potential of ear-EEG, demonstrating the viability of this more convenient alternative to full-scalp EEG for monitoring brain activity in wearable devices. This study confirms that ear-EEG can achieve comparable signal quality and reliability for specific tasks, such as sleep and auditory response monitoring, with the added benefits of user comfort and a simpler setup.

In conclusion, this thesis provides valuable insights and practical solutions for energy-efficient biosignal processing on resource-constrained platforms, demonstrating the potential to advance wearable healthcare technology. Future work may explore additional signal types, refine machine learning models for further optimization on embedded systems, and extend these methodologies to other application areas within mobile health. The continued development of low-power, real-time processing solutions holds promise for a new generation of autonomous, intelligent biomedical devices capable of improving patient outcomes and accessibility to healthcare monitoring.

Bibliography

[1] P. Schönle, F. Glaser, T. Burger, G. Rovere, L. Benini, and Q. Huang, "A Multi-Sensor and Parallel Processing SoC for Miniaturized Medical Instrumentation," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 7, pp. 2076–2087, 2018.

[2] J. Dunn, R. Runge, and M. Snyder, "Wearables and the medical revolution," *Personalized medicine*, vol. 15, no. 5, pp. 429–448, 2018.

[3] F. Forooghifar, A. Aminifar, L. Cammoun, I. Wisniewski, C. Ciumas, P. Ryvlin, and D. Atienza, "A self-aware epilepsy monitoring system for real-time epileptic seizure detection," *Mobile Networks and Applications*, pp. 1–14, 2022.

[4] P. Rai, S. Oh, P. Shyamkumar, M. Ramasamy, R. E. Harbaugh, and V. K. Varadan, "Nano- bio- textile sensors with mobile wireless platform for wearable health monitoring of neurological and cardiovascular disorders," *Journal of The Electrochemical Society*, vol. 161, no. 2, pp. 3116–3150, 2013.

[5] S. Lee, S.-W. Kim, M. Ghidelli, H. S. An, J. Jang, A. L. Bassi, S.-Y. Lee, and J.-U. Park, "Integration of transparent supercapacitors and electrodes using nanostructured metallic glass films for wirelessly rechargeable, skin heat patches," *Nano letters*, vol. 20, no. 7, pp. 4872–4881, 2020.

[6] F. Forooghifar, A. Aminifar, L. Cammoun, I. Wisniewski, C. Ciumas, P. Ryvlin, and D. Atienza, "A self-aware epilepsy monitoring system for real-time epileptic seizure detection," *Mobile Networks and Applications*, pp. 1–14, 2019.

[7] T. Hilbel and N. Frey, "Review of current ecg consumer electronics (pros and cons)," *Journal of Electrocardiology*, vol. 77, pp. 23–28, 2023.

[8] S. D. Baker and D. H. Hoglund, "Medical-grade, mission-critical wireless networks [designing an enterprise mobility solution in the healthcare environment]," *IEEE Engineering in Medicine and Biology Magazine*, vol. 27, no. 2, pp. 86–95, 2008.

[9] P. S. G. V. Parsonnet, "Implantable cardiac pacemakers: Status report and resource guideline," *American Journal of Cardiology*, vol. 34, no. 4, pp. 487–500, 1974.

[10] J. P. DiMarco and J. T. Philbrick, "Use of ambulatory electrocardiographic (Holter) monitoring," *Annals of internal medicine*, vol. 113, no. 1, pp. 53–68, 1990.

[11] R. Čihák, L. Haman, and M. Táborskỳ, "2016 ESC Guidelines for the management of atrial fibrillation developed in collaboration with EACTS," *Cor et vasa*, vol. 6, no. 58, e636–e683, 2016.

[12] R. Bhidayasiri and P. Martinez-Martin, "Clinical assessments in Parkinson's disease: scales and monitoring," *International review of neurobiology*, vol. 132, pp. 129–182, 2017.

[13] M. Capecci, L. Pepa, F. Verdini, and M. G. Ceravolo, "A smartphone-based architecture to detect and quantify freezing of gait in parkinson's disease," *Gait & Posture*, vol. 50, pp. 28–33, 2016.

[14] A. T. Herdman, O. Lins, P. Van Roon, D. R. Stapells, M. Scherg, and T. W. Picton, "Intracerebral sources of human auditory steady-state responses," *Brain topography*, vol. 15, pp. 69–86, 2002.

[15] V. Kartsch, S. Benatti, D. Rossi, and L. Benini, "A wearable EEG-based drowsiness detection system with blink duration and alpha waves analysis," in *2017 8th International IEEE/EMBS Conference on Neural Engineering (NER)*, IEEE, 2017, pp. 251–254.

[16] A. B. Brandwein, J. J. Foxe, J. S. Butler, H.-P. Frey, J. C. Bates, L. H. Shulman, and S. Molholm, "Neurophysiological indices of atypical auditory processing and multisensory integration are associated with symptom severity in autism," *Journal of autism and developmental disorders*, vol. 45, pp. 230–244, 2015.

[17] J. Constantin, A. Dogan, O. Andersson, P. Meinerzhagen, J. Rodrigues, D. Atienza, and A. T.-C. Burg, "An ultra-low-power application-specific processor for compressed sensing," in *Proceedings of IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), Santa Cruz, CA, USA*, vol. 710, 2012.

[18] F. Montagna, S. Benatti, and D. Rossi, "Flexible, Scalable and Energy Efficient Bio-Signals Processing on the PULP Platform: A Case Study on Seizure Detection," *Journal of Low Power Electronics and Applications*, vol. 7, no. 2, p. 16, 2017.

[19] M. Tomasini, S. Benatti, B. Milosevic, E. Farella, and L. Benini, "Power Line Interference Removal for High-Quality Continuous Biosignal Monitoring With Low-Power Wearable Devices," *IEEE Sensors Journal*, vol. 16, no. 10, pp. 3887–3895, 2016.

[20] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 144–152.

[21] M. Vafaei, A. Parhizgar, E. Abiri, and M. R. Salehi, "A low power and ultra-high input impedance analog front end based on fully differential difference inverter-based amplifier for biomedical applications," *AEU-International Journal of Electronics and Communications*, vol. 142, p. 154 005, 2021.

[22] M. S. Diab and E. Rodriguez-Villegas, "Embedded Machine Learning Using Microcontrollers in Wearable and Ambulatory Systems for Health and Care Applications: A Review," *IEEE Access*, vol. 10, pp. 98 450–98 474, 2022.

[23] X. Wu, Z. Wang, B. Xu, and X. Ma, "Optimized Pan-Tompkins Based Heartbeat Detection Algorithms," in *2020 Chinese Control And Decision Conference (CCDC)*, IEEE, 2020, pp. 892–897.

[24] M. Merenda, C. Porcaro, and D. Iero, "Edge Machine Learning for AI-Enabled IoT Devices: A Review," *Sensors*, vol. 20, no. 9, p. 2533, 2020.

[25] I. A. Khatib, F. Poletti, D. Bertozzi, L. Benini, M. Bechara, H. Khalifeh, A. Jantsch, and R. Nabiev, "A multiprocessor system-on-chip for real-time biomedical monitoring and analysis: ECG prototype architectural design space exploration," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 13, no. 2, pp. 1–21, 2008.

[26] T. Hussain, A. Haider, and A. Taleb-Ahmed, "A heterogeneous multi-core based biomedical application processing system and programming toolkit," *Journal of Signal Processing Systems*, vol. 91, pp. 963–978, 2019.

[27] M. A. Wickert, "Using the ARM Cortex-M4 and the CMSIS-DSP library for teaching real-time DSP," in *2015 IEEE Signal Processing and Signal Processing Education Workshop (SP/SPE)*, IEEE, 2015, pp. 283–288.

[28] W. Li and M. Liewig, "A survey of AI accelerators for edge environment," in *Trends and Innovations in Information Systems and Technologies: Volume 2 8*, Springer, 2020, pp. 35–44.

[29] E. De Giovanni, F. Montagna, B. W. Denkinger, S. Machetti, M. Peón-Quirós, S. Benatti, D. Rossi, L. Benini, and D. Atienza, "Modular design and optimization of biomedical applications for ultralow power heterogeneous platforms," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3821–3832, 2020.

[30] S. Rheindt, S. Maier, F. Schmaus, T. Wild, W. Schröder-Preikschat, and A. Herkersdorf, "SHARQ: Software-defined hardware-managed queues for tile-based manycore architectures," in *Embedded Computer Systems: Architectures, Modeling, and Simulation: 19th International Conference, SAMOS 2019, Samos, Greece, July 7–11, 2019, Proceedings 19*, Springer, 2019, pp. 212–225.

[31] C. Giannoula, N. Vijaykumar, N. Papadopoulou, V. Karakostas, I. Fernandez, J. Gómez-Luna, L. Orosa, N. Koziris, G. Goumas, and O. Mutlu, "SynCron: Efficient Synchronization Support for Near-Data-Processing Architectures," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 263–276.

[32] L. Zhao, L. N. Bhuyan, R. Iyer, S. Makineni, and D. Newell, "Hardware support for accelerating data movement in server platform," *IEEE Transactions on Computers*, vol. 56, no. 6, pp. 740–753, 2007.

[33] T. Benz, M. Rogenmoser, P. Scheffler, S. Riedel, A. Ottaviano, A. Kurth, T. Hoefler, and L. Benini, "A high-performance, energy-efficient modular DMA engine architecture," *IEEE Transactions on Computers*, 2023.

[34] D. Rossi, I. Loi, G. Haugou, and L. Benini, "Ultra-low-latency lightweight dma for tightly coupled multi-core clusters," in *Proceedings of the 11th ACM Conference on Computing Frontiers*, 2014, pp. 1–10.

[35] T. Ziegler, V. Leis, and C. Binnig, "RDMA Communication Patterns: A Systematic Evaluation," *Datenbank-Spektrum*, vol. 20, pp. 199–210, 2020.

[36] G. Xu, "IoT-assisted ECG monitoring framework with secure data transmission for health care applications," *IEEE Access*, vol. 8, pp. 74 586–74 594, 2020.

[37] N. Y. Philip, J. J. Rodrigues, H. Wang, S. J. Fong, and J. Chen, "Internet of Things for in-home health monitoring systems: Current advances, challenges and future directions," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 2, pp. 300–310, 2021.

[38] M. Hartmann, U. S. Hashmi, and A. Imran, "Edge computing in smart health care systems: Review, challenges, and research directions," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 3, e3710, 2022.

[39] Y. Xu and T. T. Qiu, "Human Activity Recognition and Embedded Application Based on Convolutional Neural Networkk," *Journal of Artificial Intelligence and Technology*, vol. 1, no. 1, pp. 51–60, 2021.

[40] E. De Giovanni, A. A. ValdÉs, M. PeÓn-QuirÓs, A. Aminifar, and D. Atienza, "Real-Time Personalized Atrial Fibrillation Prediction on Multi-Core Wearable Sensors," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 1654–1666, 2021.

[41] J. Matthews, J. Kim, and W.-H. Yeo, "Advances in biosignal sensing and signal processing methods with wearable devices," *Analysis & Sensing*, vol. 3, no. 2, e202200062, 2023.

[42] M. O. Santos, J. Costa, T. R. Fernandes, C. Silva, and S. M. M. Faria, "Wearable Inertial and Bio-signal Device for Real-time Swimmer's Monitoring," in *2021 Telecoms Conference (ConfTELE)*, 2021, pp. 1–6.

[43] S. Eom, S. Eom, and P. Washington, "SIM-CNN: self-supervised individualized multimodal learning for stress prediction on nurses using biosignals," in *Workshop on Machine Learning for Multimodal Healthcare Data*, Springer, 2023, pp. 155–171.

[44] A. K. Kumar, M. Ritam, L. Han, S. Guo, and R. Chandra, "Deep learning for predicting respiratory rate from biosignals," *Computers in Biology and Medicine*, vol. 144, p. 105 338, 2022.

[45] Y. Gu, E. Cleeren, J. Dan, K. Claes, W. Van Paesschen, S. Van Huffel, and B. Hunyadi, "Comparison between Scalp EEG and Behind-the-Ear EEG for Development of a Wearable Seizure Detection System for Patients with Focal Epilepsy," *Sensors*, vol. 18, no. 1, 2018.

[46] S. L. Kappel and P. Kidmose, "Real-life dry-contact ear-EEG," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2018, pp. 5470–5474.

[47] K. B. Mikkelsen, Y. R. Tabar, S. L. Kappel, C. B. Christensen, H. O. Toft, M. C. Hemmsen, M. L. Rank, M. Otto, and P. Kidmose, "Accurate whole-night sleep monitoring with dry-contact ear-EEG," *Scientific reports*, vol. 9, no. 1, p. 16 824, 2019.

[48] K. B. Mikkelsen, S. L. Kappel, D. P. Mandic, and P. Kidmose, "EEG recorded from the ear: characterizing the ear-EEG method," *Frontiers in neuroscience*, vol. 9, p. 438, 2015.

[49] P. Kidmose, D. Looney, M. Ungstrup, M. L. Rank, and D. P. Mandic, "A study of evoked potentials from ear-EEG," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 10, pp. 2824–2830, 2013.

[50] C. B. Christensen, J. M. Harte, T. Lunner, and P. Kidmose, "Ear-EEG-based objective hearing threshold estimation evaluated on normal hearing subjects," *IEEE Transactions on Biomedical Engineering*, vol. 65, no. 5, pp. 1026–1034, 2017.

[51] T. Tekeste, H. Saleh, B. Mohammad, and M. Ismail, "Ultra-low power qrs detection and ecg compression architecture for iot healthcare devices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 2, pp. 669–679, 2019.

[52] E. De Giovanni, T. Teijeiro, G. P. Millet, and D. Atienza, *Adaptive R-peak detection on wearable ECG sensors for high-intensity exercise*, 2022.

[53] V. Gupta and M. Mittal, "A Comparison of ECG Signal Pre-processing Using FrFT, FrWT and IPCA for Improved Analysis," *IRBM*, vol. 40, no. 3, pp. 145–156, 2019.

[54] C. Venkatesan, P. Karthigaikumar, and R. Varatharajan, "A novel LMS algorithm for ECG signal preprocessing and KNN classifier based abnormality detection," *Multimedia Tools and Applications*, vol. 77, pp. 10 365–10 374, 2018.

[55] K. Guk, G. Han, J. Lim, K. Jeong, T. Kang, E.-K. Lim, and J. Jung, "Evolution of Wearable Devices with Real-Time Disease Monitoring for Personalized Healthcare," *Nanomaterials*, vol. 9, no. 6, 2019.

[56] E. De Giovanni, F. Montagna, B. W. Denkinger, S. Machetti, M. Peón-Quirós, S. Benatti, D. Rossi, L. Benini, and D. Atienza, "Modular Design and Optimization of Biomedical Applications for Ultralow Power Heterogeneous Platforms," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3821–3832, 2020. DOI: 10.1109/TCAD.2020.3012652.

[57] M. Risso, A. Burrello, F. Conti, L. Lamberti, Y. Chen, L. Benini, E. Macii, M. Poncino, and D. J. Pagliari, "Lightweight neural architecture search for temporal convolutional networks at the edge," *IEEE Transactions on Computers*, vol. 72, no. 3, pp. 744–758, 2022.

[58] 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 *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, IEEE, 2021, pp. 1–4.

[59] J. Lázaro, N. Reljin, M.-B. Hossain, Y. Noh, P. Laguna, and K. H. Chon, "Wearable armband device for daily life electrocardiogram monitoring," *IEEE Transactions on Biomedical Engineering*, vol. 67, no. 12, pp. 3464–3473, 2020.

[60] D. P. Subha, P. K. Joseph, R. Acharya U, and C. M. Lim, "EEG signal analysis: a survey," *Journal of medical systems*, vol. 34, pp. 195–212, 2010.

[61] T. Thompson, T. Steffert, T. Ros, J. Leach, and J. Gruzelier, "EEG applications for sport and performance," *Methods*, vol. 45, no. 4, pp. 279–288, 2008.

[62] N. Srinivasan, "Cognitive neuroscience of creativity: EEG based approaches," *Methods*, vol. 42, no. 1, pp. 109–116, 2007.

[63] E. Niedermeyer, *Niedermeyer's electroencephalography: basic principles, clinical applications, and related fields*. Lippincott Williams & Wilkins, 2011.

[64] F. L. da Silva, "EEG and MEG: relevance to neuroscience," *Neuron*, vol. 80, no. 5, pp. 1112–1128, 2013.

[65] A. Cimenser, P. L. Purdon, E. T. Pierce, J. L. Walsh, A. F. Salazar-Gomez, P. G. Harrell, C. Tavares-Stoeckel, K. Habeeb, and E. N. Brown, "Tracking brain states under general anesthesia by using global coherence analysis," *Proceedings of the National Academy of Sciences*, vol. 108, no. 21, pp. 8832–8837, 2011.

[66] W. Klimesch, "EEG alpha and theta oscillations reflect cognitive and memory performance: a review and analysis," *Brain research reviews*, vol. 29, no. 2-3, pp. 169–195, 1999.

[67] G. H. Klem, "The ten-twenty electrode system of the International Federation. The International Federation of Clinical Neurophysiology," *Electroencephalogr. Clin. Neurophysiol. Suppl.*, vol. 52, pp. 3–6, 1999.

[68] D. Looney, P. Kidmose, C. Park, M. Ungstrup, M. L. Rank, K. Rosenkranz, and D. P. Mandic, "The in-the-ear recording concept: User-centered and wearable brain monitoring," *IEEE pulse*, vol. 3, no. 6, pp. 32–42, 2012.

[69] Y. M. Chi, Y.-T. Wang, Y. Wang, C. Maier, T.-P. Jung, and G. Cauwenberghs, "Dry and noncontact EEG sensors for mobile brain–computer interfaces," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 20, no. 2, pp. 228–235, 2011.

[70] M. A. Lopez-Gordo, D. Sanchez-Morillo, and F. P. Valle, "Dry EEG Electrodes," *Sensors*, vol. 14, no. 7, pp. 12 847–12 870, 2014.

[71] G.-L. Li, J.-T. Wu, Y.-H. Xia, Q.-G. He, and H.-G. Jin, "Review of semi-dry electrodes for EEG recording," *Journal of Neural Engineering*, vol. 17, no. 5, p. 051 004, 2020.

[72] D. Looney, C. Park, P. Kidmose, M. L. Rank, M. Ungstrup, K. Rosenkranz, and D. P. Mandic, "An in-the-ear platform for recording electroencephalogram," in *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, IEEE, 2011, pp. 6882–6885.

[73] P. Kidmose, D. Looney, L. Jochumsen, and D. P. Mandic, "Ear-EEG from generic earpieces: A feasibility study," in *2013 35th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, IEEE, 2013, pp. 543–546.

[74] T. Nakamura, V. Goverdovsky, M. J. Morrell, and D. P. Mandic, "Automatic sleep monitoring using ear-EEG," *IEEE journal of translational engineering in health and medicine*, vol. 5, pp. 1–8, 2017.

[75] E. Kuatsjah, X. Zhang, M. Khoshnam, and C. Menon, "Two-channel in-ear EEG system for detection of visuomotor tracking state: A preliminary study," *Medical Engineering & Physics*, vol. 68, pp. 25–34, 2019.

[76] I. Zibrandtsen, P. Kidmose, and T. Kjaer, "Detection of generalized tonic-clonic seizures from ear-EEG based on EMG analysis," *Seizure*, vol. 59, pp. 54–59, 2018.

[77] C. S. Musaeus, K. S. Frederiksen, B. B. Andersen, P. Høgh, P. Kidmose, M. Fabricius, M. C. Hribljan, M. C. Hemmsen, M. L. Rank, G. Waldemar, *et al.*, "Detection of subclinical epileptiform discharges in Alzheimer's disease using long-term outpatient EEG monitoring," *Neurobiology of Disease*, vol. 183, p. 106 149, 2023.

[78] C. S. Musaeus, T. W. Kjær, M. Cacic Hribljan, B. B. Andersen, P. Høgh, P. Kidmose, M. Fabricius, M. C. Hemmsen, M. L. Rank, G. Waldemar, *et al.*, "Subclinical epileptiform activity in dementia with Lewy bodies," *Movement Disorders*, vol. 38, no. 10, pp. 1861–1870, 2023.

[79] A. D. Lam, R. A. Sarkis, K. R. Pellerin, J. Jing, B. A. Dworetzky, D. B. Hoch, C. S. Jacobs, J. W. Lee, D. S. Weisholtz, R. Zepeda, *et al.*, "Association of epileptiform abnormalities and seizures in Alzheimer disease," *Neurology*, vol. 95, no. 16, e2259–e2270, 2020.

[80] I. Zibrandtsen, P. Kidmose, C. B. Christensen, and T. Kjaer, "Ear-EEG detects ictal and interictal abnormalities in focal and generalized epilepsy–A comparison with scalp EEG monitoring," *Clinical Neurophysiology*, vol. 128, no. 12, pp. 2454–2461, 2017.

[81] S. M. M. Islam and M. S. U. Farid, "Denoising EEG signal using Different Adaptive Filter Algorithms," *Int. Journal of Enhanced Research in Science, Technology & Engineering*, vol. 4, 2015.

[82] M. K. Ahirwal, A. Kumar, and G. K. Singh, "EEG/ERP Adaptive Noise Canceller Design with Controlled Search Space (CSS) approach in Cuckoo and other Optimization Algorithms," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 10, no. 6, pp. 1491–1504, 2013.

[83] M.-P. Hosseini, A. Hosseini, and K. Ahi, "A review on machine learning for EEG signal processing in bioengineering," *IEEE reviews in biomedical engineering*, vol. 14, pp. 204–218, 2020.

[84] F. J. Ramírez-Arias, E. E. García-Guerrero, E. Tlelo-Cuautle, J. M. Colores-Vargas, E. García-Canseco, O. R. López-Bonilla, G. M. Galindo-Aldana, and E. Inzunza-González, "Evaluation of machine learning algorithms for classification of EEG signals," *Technologies*, vol. 10, no. 4, p. 79, 2022.

[85] T. W. Picton, M. S. John, A. Dimitrijevic, and D. Purcell, "Human auditory steady-state responses: Respuestas auditivas de estado estable en humanos," *International journal of audiology*, vol. 42, no. 4, pp. 177–219, 2003.

[86] T. Kubota, T. Ito, Y. Abe, H. Chiba, Y. Suzuki, S. Kakehata, and M. Aoyagi, "Detecting the recruitment phenomenon in adults using 80-Hz auditory steady-state response," *Auris Nasus Larynx*, vol. 46, no. 5, pp. 696–702, 2019.

[87] G. Concina, A. Renna, A. Grosso, and B. Sacchetti, "The auditory cortex and the emotional valence of sounds," *Neuroscience & Biobehavioral Reviews*, vol. 98, pp. 256–264, 2019.

[88] E. McCoy, A. Walden, and D. Percival, "Multitaper spectral estimation of power law processes," *IEEE Transactions on Signal Processing*, vol. 46, no. 3, pp. 655–668, 1998.

[89] S. M. Kay, *Modern spectral estimation*. Pearson Education India, 1988.

[90] D. Thomson, "Spectrum estimation and harmonic analysis," *Proceedings of the IEEE*, vol. 70, no. 9, pp. 1055–1096, 1982.

[91] "PULP Software Development Kit." Accessed: 2020-08-20. ().

[92] F. Glaser, G. Tagliavini, D. Rossi, G. Haugou, Q. Huang, and L. Benini, "Energy-Efficient Hardware-Accelerated Synchronization for Shared-L1-Memory Multiprocessor Clusters," *IEEE Trans. on Parallel and Distributed Systems*, vol. 32, no. 03, pp. 633–648, 2021.

[93] Arm Holdings, *Common Microcontroller Software Interface Standard (CMSIS)*, Website: https://www.arm.com/technologies/cmsis, Accessed: 2024-10-20.

[94] I. Fathail and V. D. Bhagile, "ECG Paper Digitization and R Peaks Detection Using FFT," *Applied Computational Intelligence and Soft Computing*, vol. 2022, no. 1, p. 1 238 864, 2022.

[95] C.-H. Choi, J.-H. Park, H.-N. Lee, and J.-R. Yang, "Heartbeat detection using a doppler radar sensor based on the scaling function of wavelet transform," *Microwave and Optical Technology Letters*, vol. 61, no. 7, pp. 1792–1796, 2019.

[96] Z. Zidelmal, A. Amirou, M. Adnane, and A. Belouchrani, "QRS detection based on wavelet coefficients," *Computer methods and programs in biomedicine*, vol. 107, no. 3, pp. 490–496, 2012.

[97] P. S. Addison, "Wavelet transforms and the ECG: a review," *Physiological measurement*, vol. 26, no. 5, R155, 2005.

[98] M. A. Arafat and M. K. Hasan, "Automatic detection of ECG wave boundaries using empirical mode decomposition," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, pp. 461–464.

[99] L.-l. Chen, Y. Zhao, J. Zhang, and J.-z. Zou, "Automatic detection of alertness/drowsiness from physiological signals using wavelet-based nonlinear features and machine learning," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7344–7355, 2015. DOI: https://doi.org/10.1016/j.eswa.2015.05.028.

[100] P.-K. Jao, R. Chavarriaga, F. Dell'Agnola, A. Arza, D. Atienza, and J. d. R. Millán, "EEG Correlates of Difficulty Levels in Dynamical Transitions of Simulated Flying and Mapping Tasks," *IEEE Transactions on Human-Machine Systems*, vol. 51, no. 2, pp. 99–108, 2021.

[101] U. R. Acharya, H. Fujita, O. S. Lih, Y. Hagiwara, J. H. Tan, and M. Adam, "Automated detection of arrhythmias using different intervals of tachycardia ECG segments with convolutional neural network," *Information Sciences*, vol. 405, pp. 81–90, 2017.

[102] S. Liu, W. Zheng, T. Song, and Y. Zong, "Sparse graphic attention LSTM for EEG emotion recognition," in *International Conference on Neural Information Processing*, Springer, 2019, pp. 690–697.

[103] J. Zhang, Z. Yin, and R. Wang, "Recognition of Mental Workload Levels Under Complex Human–Machine Collaboration by Using Physiological Features and Adaptive Support Vector Machines," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 2, pp. 200–214, 2015.

[104] S. Sarkar, M. Agarwalla, S. Agarwal, and M. P. Sarma, "An Incremental Pruning Strategy for Fast Training of CNN Models," in *2020 International Conference on Computational Performance Evaluation (ComPE)*, 2020, pp. 371–375.

[105] N. Alamatsaz, L. Tabatabaei, M. Yazdchi, H. Payan, N. Alamatsaz, and F. Nasimi, "A lightweight hybrid CNN-LSTM explainable model for ECG-based arrhythmia detection," *Biomedical Signal Processing and Control*, vol. 90, p. 105 884, 2024.

[106] N. Attaran, A. Puranik, J. Brooks, and T. Mohsenin, "Embedded Low-Power Processor for Personalized Stress Detection," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. PP, pp. 1–1, Jan. 2018.

[107] T.-H. Tsai and W.-T. Kuo, "An Efficient ECG Lossless Compression System for Embedded Platforms With Telemedicine Applications," *IEEE Access*, vol. 6, pp. 42 207–42 215, 2018.

[108] H.-T. Nguyen, N.-D. Mai, B. G. Lee, and W.-Y. Chung, "Behind-the-ear eeg-based wearable driver drowsiness detection system using embedded tiny neural networks," *IEEE Sensors Journal*, vol. 23, no. 19, pp. 23 875–23 892, 2023.

[109] R. Banerjee, A. Mukherjee, and A. Ghose, "Noise Cleaning of ECG on Edge Device Using Convolutional Sparse Contractive Autoencoder," in *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 2022, pp. 491–496.

[110] S. Betti, R. M. Lova, E. Rovini, G. Acerbi, L. Santarelli, M. Cabiati, S. D. Ry, and F. Cavallo, "Evaluation of an integrated system of wearable physiological sensors for stress monitoring in working environments by using biological markers," *IEEE Transactions on Biomedical Engineering*, vol. 65, no. 8, pp. 1748–1758, 2018.

[111] R. Zanetti, A. Aminifar, and D. Atienza, "Robust epileptic seizure detection on wearable systems with reduced false-alarm rate," in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, vol. 2020, 2020, pp. 4248–4251.

[112] R. Zanetti, A. Arza, A. Aminifar, and D. Atienza, "Real-time EEG-based cognitive workload monitoring on wearable devices," *IEEE transactions on biomedical engineering*, vol. 69, no. 1, pp. 265–277, 2021.

[113] V. P. Kumaravel, U. Pale, T. Teijeiro, E. Farella, and D. A. Alonso, "Knowledge distillation-based channel reduction for wearable eeg applications," *Authorea Preprints*, 2023.

[114] Arm Holdings, *Cortex-M7*, Website: https://developer.arm.com/Processors/Cortex-M4, Accessed: 2024-10-14.

[115] STMicroelectronics, *STM32H742xI/G STM32H743xI/G*, available at https://www.st.com/resource/en/datasheet/stm32f401re.pdf (14/10/2024), 2023.

[116] E. Flamand, D. Rossi, F. Conti, I. Loi, A. Pullini, F. Rotenberg, and L. Benini, "GAP-8: A RISC-V SoC for AI at the edge of the IoT," in *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2018, pp. 1–4.

[117] A. Burrello, "Optimizing AI at the Edge: from network topology design to MCU deployment," Ph.D. dissertation, University of Bologna, 2023.

[118] A. Rahimi, I. Loi, M. R. Kakoee, and L. Benini, "A fully-synthesizable single-cycle interconnection network for Shared-L1 processor clusters," in *2011 Design, Automation & Test in Europe*, 2011, pp. 1–6.

[119] A. Pullini, D. Rossi, G. Haugou, and L. Benini, "µDMA: An autonomous I/O subsystem for IoT end-nodes," in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2017, pp. 1–8.

[120] GreenWaves Technologies, *Low-Power Processors*, Website: https://greenwaves-technologies.com/low-power-processor/, Accessed: 2024-10-14.

[121] MLCommons, *MLPerf inference: Tiny benchmark suite results*, Website: https://mlcommons.org/benchmarks/inference-tiny/, Accessed: 2024-09-10.

[122] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, 2021.

[123] A. Burrello, F. Conti, A. Garofalo, D. Rossi, and L. Benini, "DORY: Lightweight memory hierarchy management for deep NN inference on IoT endnodes: Work-in-progress," in *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis Companion*, ser. CODES/ISSS '19, Association for Computing Machinery, 2019.

[124] A. Burrello, A. Garofalo, N. Bruschi, G. Tagliavini, D. Rossi, and F. Conti, "DORY: Automatic end-to-end deployment of real-world DNNs on low-cost IoT MCUs," *IEEE Transactions on Computers*, vol. 70, no. 8, pp. 1253–1268, 2021.

[125] F. Conti, *Technical report: NEMO DNN quantization for deployment model*, 2020.

[126] M. G. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *ACM Comput. Surv.*, vol. 54, no. 8, Oct. 2021. DOI: 10.1145/3469029.

[127] G. Technology, *GAP SDK NNTool*, Website: https://github.com/GreenWaves-Technologies/gap_sdk/blob/master/tools/nntool/README.md, Accessed: 2024-08-28.

[128] Google DeepMind, *TensorFlow Lite*, Website: https://www.tensorflow.org/lite/guide, Accessed: 2024-22-10.

[129] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-scale machine learning on heterogeneous distributed systems*, 2016.

[130] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'16, USENIX Association, 2016, pp. 265–283.

[131] Google DeepMind, *TensorFlow*, Website: https://www.tensorflow.org/, Accessed: 2024-10-10.

[132] Google DeepMind, *Keras: The high-level API for TensorFlow*, Website: https://www.tensorflow.org/guide/keras, Accessed: 2024-10-25.

[133] Google DeepMind, *TensorFlow Lite for microcontrollers*, Website: https://www.tensorflow.org/lite/microcontrollers, Accessed: 2024-10-25.

[134] *Open Neural Network Exchange*, Website: https://github.com/onnx, Accessed: 2024-10-25.

[135] T. Jin, G.-T. Bercea, T. D. Le, T. Chen, G. Su, H. Imai, Y. Negishi, A. Leu, K. O'Brien, K. Kawachiya, and A. E. Eichenberger, *Compiling ONNX neural network models using MLIR*, 2020.

[136] F. Conti and A. Di Mauro, *NEMO (NEural Minimizer for pytOrch)*, Website: https://github.com/pulp-platform/nemo, Accessed: 2024-25-10.

[137] A. P. *et al.*, "Automatic differentiation in PyTorch," in *2017 Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.

[138] P. *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2019.

[139] F. Conti, D. Rossi, A. Pullini, I. Loi, and L. Benini, "PULP: A ultra-low power parallel accelerator for energy-efficient and flexible embedded vision," *J. Signal Process. Syst.*, vol. 84, no. 3, pp. 339–354, Sep. 2016.

[140] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, *PACT: Parameterized clipping activation for quantized neural networks*, 2018.

[141] A. Burrello, F. Conti, L. Macan, G. Rutishauer, T. M. Ingolfsson, A. Garofalo, N. Bruschi, G. Tagliavini, D. Rossi, and L. Benini, *DORY: Deployment ORiented to memorY*, Website: https://github.com/pulp-platform/dory, Accessed: 2024-25-09.

[142] N. Bruschi, G. Haugou, G. Tagliavini, F. Conti, L. Benini, and D. Rossi, "GV-SoC: a highly configurable, fast and accurate full-platform simulator for RISC-V based IoT processors," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*, IEEE, 2021, pp. 409–416.

[143] STMicroelectronics, *X-Cube-AI*, Website: https://wiki.stmicroelectronics.cn/stm32mcu/wiki/AI:X-CUBE-AI_documentation/, Accessed: 2025-02-01.

[144] STMicroelectronics, *STM32CubeMX*, Website: https://www.st.com/content/st_com/en/stm32cubemx.html/, Accessed: 2025-02-01.

[145] M. Spallanzani, G. Rutishauser, M. Scherer, P. Wiese, and F. Conti, *QuantLib*, available at https://github.com/pulp-platform/quantlib (25/01/2024).

[146] M. Spallanzani, G. Rutishauser, M. Scherer, P. Wiese, and F. Conti, *QuantLab*, available at https://github.com/pulp-platform/quantlab (25/01/2024).

[147] M. Spallanzani, G. Rutishauser, M. Scherer, A. Burrello, F. Conti, and L. Benini, *QuantLab: A modular framework for training and deploying mixed-precision NNs*, available at https://cms.tinyml.org/wp-content/uploads/talks2022/Spallanzani-Matteo-Hardware.pdf (25/01/2024).

[148] *Horovod on GPU*, available at https://horovod.readthedocs.io/en/stable/gpus_include.html (25/01/2024).

[149] M. Spallanzani, G. P. Leonardi, and L. Benini, "Training quantised neural networks with STE variants: The additive noise annealing algorithm," in *2022 IEEE / CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 470–479. DOI: 10.1109/CVPR52688.2022.00056.

[150] P. Busia, "Optimizing neural networks for embedded edge-processing platforms," Ph.D. dissertation, University of Cagliari, Cagliari, Italy, available at https://iris.unica.it/handle/11584/357302 (25/01/2024), 2023.

[151] A. Dupre, S. Vincent, and P. A. Iaizzo, "Basic ECG theory, recordings, and interpretation," *Handbook of cardiac anatomy, physiology, and devices*, pp. 191–201, 2005.

[152] John Furst, *A Basic Guide to ECG/EKG Interpretation*, Website: https://www.firstaidforfree.com/a-basic-guide-to-ecgekg-interpretation/, Accessed: 2024-10-14.

[153] R. Braojos, G. Ansaloni, D. Atienza, and F. J. Rincón, "Embedded real-time ECG delineation methods: A comparative evaluation," in *2012 IEEE 12th International Conference on Bioinformatics & Bioengineering (BIBE)*, IEEE, 2012, pp. 99–104.

[154] B. Mazzoni, S. Benatti, L. Benini, and G. Tagliavini, "Efficient Transform Algorithms for Parallel Ultra-Low-Power IoT End Nodes," *IEEE Embedded Systems Letters*, vol. 13, no. 4, pp. 210–213, 2021.

[155] E. Flamand, D. Rossi, F. Conti, I. Loi, A. Pullini, F. Rotenberg, and L. Benini, "GAP-8: A RISC-V SoC for AI at the Edge of the IoT," in *IEEE 29th Int. Conf. on Application-specific Systems, Architectures and Processors (ASAP)*, IEEE, 2018, pp. 1–4.

[156] C. Chen, X. Xiang, C. Liu, Y. Shang, R. Guo, D. Liu, Y. Lu, Z. Hao, J. Luo, Z. Chen, *et al.*, "Xuantie-910: A Commercial Multi-Core 12-Stage Pipeline Out-of-Order 64-bit High Performance RISC-V Processor with Vector Extension: Industrial Product," in *ACM/IEEE 47th Annual Int. Symp. on Comp. Arch. (ISCA)*, 2020, pp. 52–64.

[157] D. Rossi, F. Conti, A. Marongiu, A. Pullini, I. Loi, M. Gautschi, G. Tagliavini, A. Capotondi, P. Flatresse, and L. Benini, "PULP: A parallel ultra low power platform for next generation IoT applications," in *2015 IEEE Hot Chips 27 Symposium (HCS)*, IEEE, 2015, pp. 1–39.

[158] M. Alioto, E. Consoli, and G. Palumbo, *Flip-Flop Design in Nanometer CMOS*. Springer, 2016.

[159] A. Pullini, D. Rossi, I. Loi, G. Tagliavini, and L. Benini, "Mr. Wolf: An energy-precision scalable parallel ultra low power SoC for IoT edge processing," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 7, pp. 1970–1981, 2019.

[160] *CMSIS Source Code*, Website: https://github.com/ARM-software/?q=cmsis&type=all&language=&sort=, Accessed: 2024-10-16.

[161] S. Benatti, E. Farella, and L. Benini, "Towards EMG control interface for smart garments," in *Proc. of the 2014 ACM Int. Symposium on Wearable Computers: Adjunct Program*, 2014, pp. 163–170.

[162] E. Cabal-Yepez, A. G. Garcia-Ramirez, R. J. Romero-Troncoso, A. Garcia-Perez, and R. A. Osornio-Rios, "Reconfigurable monitoring system for time-frequency analysis on industrial equipment through STFT and DWT," *IEEE Trans. on Industrial Informatics*, vol. 9, no. 2, pp. 760–771, 2012.

[163] X. Wang, M. Magno, L. Cavigelli, and L. Benini, "FANN-on-MCU: An open-source toolkit for energy-efficient neural network inference at the edge of the internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4403–4417, 2020.

[164] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005.

[165] *Kiss FFT*, Website: https://github.com/mborgerding/kissfft, Accessed: 2024-10-08.

[166] *GSL library*, https://https://github.com/ampl/gsl, Accessed: 2024-10-02.

[167] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[168] F. Montagna, S. Mach, S. Benatti, A. Garofalo, G. Ottavi, L. Benini, D. Rossi, and G. Tagliavini, "A low-power transprecision floating-point cluster for efficient near-sensor data analytics," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 5, pp. 1038–1053, 2021.

[169] J. Pan and W. J. Tompkins, "A Real-Time QRS Detection Algorithm," *IEEE Transactions on Biomedical Engineering*, vol. BME-32, no. 3, pp. 230–236, 1985.

[170] D. Rossi, F. Conti, A. Marongiu, A. Pullini, I. Loi, M. Gautschi, G. Tagliavini, A. Capotondi, P. Flatresse, and L. Benini, "PULP: A parallel ultra low power platform for next generation IoT applications," in *2015 IEEE Hot Chips 27 Symposium (HCS)*, 2015, pp. 1–39.

[171] D. Rossi, F. Conti, M. Eggiman, S. Mach, A. D. Mauro, M. Guermandi, G. Tagliavini, A. Pullini, I. Loi, J. Chen, E. Flamand, and L. Benini, "4.4 a 1.3tops/w @ 32gops fully integrated 10-core soc for iot end-nodes with 1.7μw cognitive wake-up from mram-based state-retentive sleep mode," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 60–62.

[172] P. Kirchhof, S. Benussi, D. Kotecha, A. Ahlsson, D. Atar, B. Casadei, M. Castella, H.-C. Diener, H. Heidbuchel, J. Hendriks, G. Hindricks, A. S. Manolis, J. Oldgren, B. A. Popescu, U. Schotten, B. Van Putte, P. Vardas, and E. S. D. Group, "2016 ESC Guidelines for the management of atrial fibrillation developed in collaboration with EACTS," *European Heart Journal*, vol. 37, no. 38, pp. 2893–2962, Aug. 2016.

[173] G. Moody and R. Mark, "The impact of the mit-bih arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[174] E. De Giovanni, T. Teijeiro, D. Meier, G. Millet, and D. Atienza, *Ecg in high intensity exercise dataset*, Zenodo, Nov. 2021.

[175] S. Benatti, F. Casamassima, B. Milosevic, E. Farella, P. Schönle, S. Fateh, T. Burger, Q. Huang, and L. Benini, "A versatile embedded platform for emg acquisition and gesture recognition," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 9, no. 5, pp. 620–630, 2015.

[176] M. Magno, G. A. Salvatore, S. Mutter, W. Farrukh, G. Troester, and L. Benini, "Autonomous smartwatch with flexible sensors for accurate and continuous mapping of skin temperature," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2016, pp. 337–340.

[177] J. S. Park, S. W. Lee, and U. Park, "R peak detection method using wavelet transform and modified shannon energy envelope," *Journal of Healthcare Engineering*, vol. 2017, pp. 1–14, Jul. 2017.

[178] A. Martinez Rodrigo, R. Alcaraz, and J. Rieta, "Application of the phasor transform for automatic delineation of single-lead ECG fiducial points," *Physiological measurement*, vol. 31, pp. 1467–1485, Nov. 2010.

[179] X. Lu, M. Pan, and Y. Yu, "QRS Detection Based on Improved Adaptive Threshold," *Journal of Healthcare Engineering*, vol. 2018, pp. 1–8, Mar. 2018.

[180] R. Moreira, J. Leite, T. C. Pimenta, and R. Moreno, "Online heartbeat classification using low cost algorithms," in *2019 31st International Conference on Microelectronics (ICM)*, IEEE, 2019, pp. 150–153.

[181] *Maxim datasheet*, Website: https://datasheets.maximintegrated.com/en/ds/MAX30003.pdf, Accessed: 2024-05-10.

[182] B. Mazzoni, G. Tagliavini, L. Benini, and S. Benatti, "An optimized heart rate detection system based on low-power microcontroller platforms for biosignal processing," in *Advances in System-Integrated Intelligence: Proceedings of the 6th International Conference on System-Integrated Intelligence (SysInt 2022), September 7-9, 2022, Genova, Italy*, Springer, 2022, pp. 160–170.

[183] *Cnn representation*, Website: https://developersbreach.com/convolution-neural-network-deep-learning/, Accessed: 2024-05-10.

[184] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint: 1803.01271*, 2018.

[185] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1003–1012.

[186] S. Gopali, F. Abri, S. Siami-Namini, and A. S. Namin, "A comparison of TCN and LSTM models in detecting anomalies in time series data," in *2021 IEEE International Conference on Big Data (Big Data)*, IEEE, 2021, pp. 2415–2420.

[187] A. Burrello, M. Zanghieri, C. Sarti, L. Ravaglia, S. Benatti, and L. Benini, "Tackling time-variability in sEMG-based gesture recognition with on-device incremental learning and temporal convolutional networks," in *2021 IEEE Sensors Applications Symposium (SAS)*, 2021, pp. 1–6.

[188] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, *WaveNet: A generative model for raw audio*, 2016.

[189] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[190] P. Busia, M. A. Scrugli, V. J.-B. Jung, L. Benini, and P. Meloni, "A Noisy Beat is Worth 16 Words: a Tiny Transformer for Low-Power Arrhythmia Classification on Microcontrollers," *arXiv preprint arXiv:2402.10748*, 2024.

[191] Y. Li, Y. Qi, and Y. Wang, "Avoiding subject-specific model selection via highway networks in EEG signals," in *BIBE 2019; The Third International Conference on Biological Information and Biomedical Engineering*, VDE, 2019, pp. 1–5.

[192] A. R. Ismail, S. Jovanovic, N. Ramzan, and H. Rabah, "ECG Signal Classification Using Temporal Convolutional Network," in *2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, IEEE, 2022, pp. 1–4.

[193] H. Ismail, M. A. Serhani, N. M. Hussein, and M. Elhadef, "RL-ECGNet: resource-aware multi-class detection of arrhythmia through reinforcement learning," *Applied Intelligence*, vol. 53, no. 24, pp. 30 927–30 939, 2023.

[194] A. Alamr and A. Artoli, "Unsupervised Transformer-Based Anomaly Detection in ECG Signals," *Algorithms*, vol. 16, no. 3, 2023.

[195] P. Matias, D. Folgado, H. Gamboa, and A. V. Carreiro, "Robust Anomaly Detection in Time Series through Variational AutoEncoders and a Local Similarity Score," in *Biosignals*, 2021, pp. 91–102.

[196] R. Hu, J. Chen, and L. Zhou, "A transformer-based deep neural network for arrhythmia detection using continuous ECG signals," *Computers in Biology and Medicine*, vol. 144, p. 105 325, 2022.

[197] G. Yan, S. Liang, Y. Zhang, and F. Liu, "Fusing Transformer Model with Temporal Features for ECG Heartbeat Classification," in *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2019, pp. 898–905.

[198] X. Liang, H. Fan, J. Mercer, and H. Heidari, "A delay-based neuromorphic processor for arrhythmias detection," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[199] GreenWaves Technology, *GAP9 processor*, Website: https://greenwaves-technologies.com/gap9_iot_application_processor/, Accessed: 2024-08-22.

[200] D. Rossi, F. Conti, M. Eggiman, A. D. Mauro, G. Tagliavini, S. Mach, M. Guermandi, A. Pullini, I. Loi, J. Chen, E. Flamand, and L. Benini, "Vega: A Ten-Core SoC for IoT Endnodes With DNN Acceleration and Cognitive Wake-Up From MRAM-Based State-Retentive Sleep Mode," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 1, pp. 127–139, 2022.

[201] F. Conti, P. Schiavone, and L. Benini, "XNOR Neural Engine: a Hardware Accelerator IP for 21.6 fJ/op Binary Neural Network Inference," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, pp. 1–1, Jul. 2018.

[202] Francesco Conti, *Neural Engine 16*, Website: https://github.com/pulp-platform/ne16, Accessed: 2024-09-01.

[203] H. Yoda, E. Kitagawa, N. Shimomura, S. Fujita, and M. Amano, "The progresses of MRAM as a memory to save energy consumption and its potential for further reduction," in *2015 Symposium on VLSI Circuits (VLSI Circuits)*, IEEE, 2015, T104–T105.

[204] GreenWaves Technologies, *GAP SDK*, Website: https://greenwaves-technologies.com/setting-up-sdk/, Accessed: 2024-10-02.

[205] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *ArXiv*, vol. abs/1905.11946, 2019.

[206] Harvard-MIT Division of Health Sciences and Technology Biomedical Engineering Center, *Dataset intro*, Website: https://physionet.org/physiobank/database/html/mitdbdir/intro.htm, Accessed: 2024-09-30.

[207] R. Mark, "AAMI-recommended practice: Testing and reporting performance results of ventricular arrhythmia detection algorithms," *Association for the Advancement of Medical Instrumentation, Arrhythmia Monitoring Subcommittee, AAMI ECAR, 1987*, 1987.

[208] M. A. Scrugli, D. Loi, L. Raffo, and P. Meloni, "An adaptive cognitive sensor node for ECG monitoring in the Internet of Medical Things," *IEEE Access*, vol. 10, pp. 1688–1705, 2021.

[209] Y. Chen, Y. Hao, T. Rakthanmanon, J. Zakaria, B. Hu, and E. Keogh, "A general framework for never-ending learning from time series streams," *Data mining and knowledge discovery*, vol. 29, pp. 1622–1664, 2015.

[210] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," *circulation*, vol. 101, no. 23, e215–e220, 2000.

[211] D. S. Baim, W. S. Colucci, E. S. Monrad, H. S. Smith, R. F. Wright, A. Lanoue, D. F. Gauthier, B. J. Ransil, W. Grossman, and E. Braunwald, "Survival of patients with severe congestive heart failure treated with oral milrinone," *Journal of the American College of Cardiology*, vol. 7, no. 3, pp. 661–670, 1986.

[212] François Chollet, *KerasAPI*, Website: https://keras.io/api/, Accessed: 2024-09-30.

[213] G. Technology, *GAP9 Product Brief Greenwaves*, Website: https://greenwaves-technologies.com/wp-content/uploads/2022/06/Product-Brief-GAP9-Sensors-General-V1_14.pdf, Accessed: 2024-08-30.

[214] J. Getzner, B. Charpentier, and S. Günnemann, *Accuracy is not the only Metric that matters: Estimating the Energy Consumption of Deep Learning Models*, 2023.

[215] M. Konopik, T. Korten, E. Lutz, and H. Linke, "Fundamental energy cost of finite-time parallelizable computing," *Nature Communications*, vol. 14, no. 1, p. 447, 2023.

[216] L. Falaschetti, M. Alessandrini, G. Biagetti, P. Crippa, and C. Turchetti, "ECG-based arrhythmia classification using recurrent neural networks in embedded systems," *Procedia Computer Science*, vol. 207, pp. 3479–3487, 2022.

[217] J. Johnson, "Rethinking floating point for deep learning," *arXiv preprint: 1811.01721*, 2018.

[218] T. M. Ingolfsson, M. Hersche, X. Wang, N. Kobayashi, L. Cavigelli, and L. Benini, "EEG-TCNet: An Accurate Temporal Convolutional Network for Embedded Motor-Imagery Brain–Machine Interfaces," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2020, pp. 2958–2965.

[219] B. Mazzoni, L. Bompani, M. Orlandi, S. Benatti, and G. Tagliavini, "Balancing Accuracy and Energy Efficiency on Ultra-Low-Power Platforms for ECG Analysis," in *2024 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, IEEE, 2024, pp. 1–6.

[220] S. Majumder, T. Mondal, and M. J. Deen, "Wearable Sensors for Remote Health Monitoring," *Sensors*, vol. 17, no. 1, p. 130, 2017.

[221] M. G. Bleichner, P. Kidmose, and J. Voix, *Ear-centered sensing: from sensing principles to research and clinical devices*, 2020.

[222] Datwyler Group, *Ear-EEG electrodes*, Website: https://datwyler.com/, Accessed: 2024-09-12.

[223] T. Kubota, T. Ito, Y. Abe, H. Chiba, Y. Suzuki, S. Kakehata, and M. Aoyagi, "Detecting the recruitment phenomenon in adults using 80-hz auditory steady-state response," *Auris Nasus Larynx*, vol. 46, no. 5, pp. 696–702, 2019.

[224] A. Paul, A. Akinin, M. S. Lee, M. Kleffner, S. R. Deiss, and G. Cauwenberghs, "Integrated in-ear device for auditory health assessment," in *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2019, pp. 56–59.

[225] C. B. Christensen, S. L. Kappel, and P. Kidmose, "Auditory steady-state responses across chirp repetition rates for ear-EEG and scalp EEG," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2018, pp. 1376–1379.

[226] L. Fiedler, J. Obleser, T. Lunner, and C. Graversen, "Ear-EEG allows extraction of neural responses in challenging listening scenarios—a future technology for hearing aids?" In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2016, pp. 5697–5700.

[227] S. Surendran, S. Prodanovic, and S. Stenfelt, "Hearing through bone conduction headsets," *Trends in Hearing*, vol. 27, p. 23 312 165 231 168 741, 2023.

[228] S. Frey, M. Guermandi, S. Benatti, V. Kartsch, A. Cossettini, and L. Benini, "BioGAP: A 10-core FP-capable ultra-low power IoT processor, with medical-grade AFE and BLE connectivity for wearable biosignal processing," in *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, IEEE, 2023, pp. 1–7.

[229] J. W. Peirce, "PsychoPy—Psychophysics software in Python," *Journal of Neuroscience Methods*, vol. 162, pp. 8–13, 2007.

[230] C. B. Christensen, T. Lunner, J. M. Harte, M. L. Rank, and P. Kidmose, "Chirp-evoked auditory steady-state response: The effect of repetition rate," *IEEE Transactions on Biomedical Engineering*, vol. 69, no. 2, pp. 689–699, 2021.

[231] K. K. Parhi and M. Ayinala, "Low-complexity Welch power spectral density computation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 1, pp. 172–182, 2013.