



Politecnico di Bari

Repository Istituzionale dei Prodotti della Ricerca del Politecnico di Bari

Design and implementation of embedded subsystems for monitoring and managing vital parameters

This is a PhD Thesis

Original Citation:

Design and implementation of embedded subsystems for monitoring and managing vital parameters / Mascellaro, Grazia. - ELETTRONICO. - (2026).

Availability:

This version is available at <http://hdl.handle.net/11589/296340> since: 2026-01-26

Published version

DOI:

Publisher: Politecnico di Bari

Terms of use:

(Article begins on next page)

22 April 2026



Department of Electrical and Information Engineering

ELECTRICAL AND INFORMATION ENGINEERING

Ph.D. Program

SSD: IINF-05/A

Final Dissertation

**Design and implementation of
embedded subsystems for monitoring
and managing vital parameters**

by

Grazia Mascellaro

Supervisors:

Prof. Eng. Michele Ruta

Prof. Eng. Floriano Scioscia

Coordinator of Ph.D. Program:

Prof. Mario Carpentieri

LIBERATORIA PER L'ARCHIVIAZIONE DELLA TESI DI DOTTORATO

Al Magnifico Rettore
del Politecnico di Bari

Il/la sottoscritto/a GRAZIA MASCELLARO nato/a a ALTAMURA il 11/03/1993

residente a GRAVINA IN PUGLIA in via GIARDINI, 33 e-mail grazia.mascellaro@outlook.com

iscritto al 3° anno di Corso di Dottorato di Ricerca in INGEGNERIA ELETTRICA E DELL'INFORMAZIONE ciclo 38°

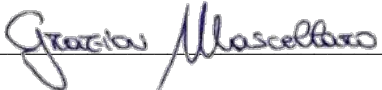
ed essendo stato ammesso a sostenere l'esame finale con la prevista discussione della tesi dal titolo:

DESIGN AND IMPLEMENTATION OF EMBEDDED SUBSYSTEMS FOR MONITORING AND MANAGING VITAL PARAMETERS

DICHIARA

- 1) di essere consapevole che, ai sensi del D.P.R. n. 445 del 28.12.2000, le dichiarazioni mendaci, la falsità negli atti e l'uso di atti falsi sono puniti ai sensi del codice penale e delle Leggi speciali in materia, e che nel caso ricorressero dette ipotesi, decade fin dall'inizio e senza necessità di nessuna formalità dai benefici conseguenti al provvedimento emanato sulla base di tali dichiarazioni;
- 2) di essere iscritto al Corso di Dottorato di ricerca INGEGNERIA ELETTRICA E DELL'INFORMAZIONE ciclo 38°, corso attivato ai sensi del "Regolamento dei Corsi di Dottorato di ricerca del Politecnico di Bari", emanato con D.R. n.286 del 01.07.2013;
- 3) di essere pienamente a conoscenza delle disposizioni contenute nel predetto Regolamento in merito alla procedura di deposito, pubblicazione e autoarchiviazione della tesi di dottorato nell'Archivio Istituzionale ad accesso aperto alla letteratura scientifica;
- 4) di essere consapevole che attraverso l'autoarchiviazione delle tesi nell'Archivio Istituzionale ad accesso aperto alla letteratura scientifica del Politecnico di Bari (IRIS-POLIBA), l'Ateneo archiverà e renderà consultabile in rete (nel rispetto della Policy di Ateneo di cui al D.R. 642 del 13.11.2015) il testo completo della tesi di dottorato, fatta salva la possibilità di sottoscrizione di apposite licenze per le relative condizioni di utilizzo (di cui al sito <http://www.creativecommons.it/Licenze>), e fatte salve, altresì, le eventuali esigenze di "embargo", legate a strette considerazioni sulla tutelabilità e sfruttamento industriale/commerciale dei contenuti della tesi, da rappresentarsi mediante compilazione e sottoscrizione del modulo in calce (Richiesta di embargo);
- 5) che la tesi da depositare in IRIS-POLIBA, in formato digitale (PDF/A) sarà del tutto identica a quelle **consegnate**/inviata/di inviarsi ai componenti della commissione per l'esame finale e a qualsiasi altra copia depositata presso gli Uffici del Politecnico di Bari in forma cartacea o digitale, ovvero a quella da discutere in sede di esame finale, a quella da depositare, a cura dell'Ateneo, presso le Biblioteche Nazionali Centrali di Roma e Firenze e presso tutti gli Uffici competenti per legge al momento del deposito stesso, e che di conseguenza va esclusa qualsiasi responsabilità del Politecnico di Bari per quanto riguarda eventuali errori, imprecisioni o omissioni nei contenuti della tesi;
- 6) che il contenuto e l'organizzazione della tesi è opera originale realizzata dal sottoscritto e non compromette in alcun modo i diritti di terzi, ivi compresi quelli relativi alla sicurezza dei dati personali; che pertanto il Politecnico di Bari ed i suoi funzionari sono in ogni caso esenti da responsabilità di qualsivoglia natura: civile, amministrativa e penale e saranno dal sottoscritto tenuti indenni da qualsiasi richiesta o rivendicazione da parte di terzi;
- 7) che il contenuto della tesi non infrange in alcun modo il diritto d'Autore né gli obblighi connessi alla salvaguardia di diritti morali od economici di altri autori o di altri aventi diritto, sia per testi, immagini, foto, tabelle, o altre parti di cui la tesi è composta.

Luogo e data Bari, 22.01.2026

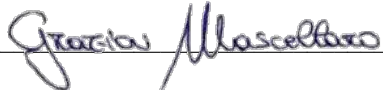
Firma 

Il/La sottoscritto, con l'autoarchiviazione della propria tesi di dottorato nell'Archivio Istituzionale ad accesso aperto del Politecnico di Bari (POLIBA-IRIS), pur mantenendo su di essa tutti i diritti d'autore, morali ed economici, ai sensi della normativa vigente (Legge 633/1941 e ss.mm.ii.),

CONCEDE

- al Politecnico di Bari il permesso di trasferire l'opera su qualsiasi supporto e di convertirla in qualsiasi formato al fine di una corretta conservazione nel tempo. Il Politecnico di Bari garantisce che non verrà effettuata alcuna modifica al contenuto e alla struttura dell'opera.
- al Politecnico di Bari la possibilità di riprodurre l'opera in più di una copia per fini di sicurezza, back-up e conservazione.

Luogo e data Bari, 22.01.2026

Firma 



Department of Electrical and Information Engineering

ELECTRICAL AND INFORMATION ENGINEERING

Ph.D. Program

SSD: IINF-05/A

Final Dissertation

**Design and implementation of
embedded subsystems for monitoring
and managing vital parameters**

by

Grazia Mascellaro

Referees:

Dr. Eng. Angelo Cardellicchio

Prof. Eng. Gaetano Patti

Supervisors:

Prof. Eng. Michele Ruta

Prof. Eng. Floriano Scioscia

Coordinator of Ph.D. Program:

Prof. Eng. Mario Carpentieri

Course n° 38, 01/11/2022–31/10/2025

Abstract

This dissertation presents an interdisciplinary research pathway that integrates functional safety engineering, biosensing technologies, and eXplainable Artificial Intelligence (XAI) for embedded and resource-constrained systems. The common objective across these research lines is the development of intelligent and autonomous systems that are not only functional and efficient, but also interpretable, verifiable, explainable and safe. In the current landscape of pervasive computing and embedded intelligence, ensuring reliability and transparency has become essential, particularly for monitoring vital parameters of – natural or artificial – complex systems in domains such as transportation, healthcare, and human–machine interaction, where incorrect or opaque behavior can have severe consequences.

The first research line investigates functional safety within railway automation systems, focusing on the validation of autonomous train operation (ATO) over the European Train Control System (ETCS). As automation in railway control advances, it becomes increasingly necessary to guarantee system integrity, redundancy, and predictability under all operating conditions. Through systematic hazard analysis and its implementation in a Vital Control (VC) module, the study shows how formal safety requirements can be met on embedded platforms, ensuring deterministic behavior and compliance with IEC 61508 principles. Indeed, the module is demonstrated to ensure deterministic safety responses, such as autonomous emergency braking and system decay management, when remote supervision fails or subsystem faults occur. The implemented architecture supports a hierarchical safety logic that preserves both functional continuity and fail-safe behavior. The experimental validation confirms that formal safety engineering principles can be effectively realized on embedded platforms, achieving real-time operation with limited computational resources.

The second research line investigates biosensing technologies aimed at continuous and non-invasive physiological monitoring. In particular, the study explores chloride ion de-

tection as a proxy for sweat analysis, due to its relevance in thermoregulation, hydration assessment, and early detection of clinical conditions. A complete wearable biosensing platform is designed and implemented, integrating the electrochemical sensor, conditioning electronics, microcontroller-based acquisition unit, and embedded signal processing chain. The system's design emphasizes modularity, low power consumption, and robustness against environmental variations. Experimental results demonstrate the system's capability to perform stable and repeatable measurements of chloride concentration. Furthermore, the platform provides a solid basis for the future integration of intelligent data analysis modules, enabling context-aware health monitoring and adaptive physiological feedback. This contribution positions wearable biosensing not only as a biomedical tool, but as a component of a broader human-centered cyber-physical ecosystem, where sensors, intelligence, and explainability are interconnected.

The third and central contribution concerns the work performed on MAFALDA 2.0, a framework for XAI on Micro-Controller Units (MCUs). Unlike most state-of-the-art Machine Learning (ML) systems that rely on opaque models and cloud-based computation, MAFALDA 2.0 enables on-device learning and reasoning while preserving transparency and interpretability. Implemented entirely in C language, the framework supports Incremental Learning (IL) and Federated Learning (FL) paradigms, allowing models to evolve dynamically over time and across distributed devices. Its core innovation lies in the integration of symbolic reasoning and semantic matchmaking, which transforms the classification process into a reasoning task grounded in ontology-based knowledge representations. This approach enables both global and local explainability, as each decision can be expressed in terms of symbolic relationships between input features and learned concepts. Moreover, MAFALDA 2.0 includes a configurable cutoff parameter that regulates the depth and granularity of the user-readable explanations. Extensive experimental evaluation, conducted on standard datasets for monitoring vital parameters, shows that the proposed framework achieves accuracy comparable to state-of-the-art Tiny Machine Learning (TinyML) models while maintaining a small memory footprint, short inference times, and low energy consumption. The system supports on-device training and prediction without relying on external computational infrastructures, thus enabling fully autonomous intelligent behavior even on resource-constrained hardware.

Overall, the dissertation advances the state of the art toward trustworthy and self-explanatory embedded intelligence, where functional safety, biosensing, and explainability converge to enable reliable human-machine symbiosis in safety-critical, wearable and ambient

intelligence contexts.

Contents

Abstract

1	Introduction	4
2	Monitoring and Managing Vital Parameters	9
2.1	Monitoring vital parameters	9
2.1.1	Motivating scenarios	10
2.1.1.1	Safety-critical autonomous systems	10
2.1.1.2	Healthcare	14
2.1.1.3	Affective Computing	15
2.1.2	Embedded monitoring architectures	17
2.1.2.1	Sensors and microdevices	17
2.1.2.2	Processing boards	22
2.1.2.3	Operating Systems	25
2.2	Artificial Intelligence for managing vital parameters in embedded systems	28
2.2.1	Machine Learning	29
2.2.1.1	Decision Trees	31
2.2.1.2	Random Forests	34
2.2.1.3	Support Vector Machines	35
2.2.1.4	Artificial Neural Networks	39
2.2.1.5	Deep Learning	42
2.2.2	Incremental Machine Learning	44
2.2.3	Federated Machine Learning	47
2.2.4	Knowledge Representation and Reasoning	49
2.2.4.1	Ontology	50
2.2.4.2	Description Logics	51
2.2.4.3	Web Ontology Language and Semantic Web	52

2.2.4.4	Reasoning	53
2.2.5	Optimizations for embedded systems	55
2.3	Open issues	58
2.3.1	Integration on embedded systems	58
2.3.2	Trustworthiness	62
2.3.2.1	Explainability and Interpretability	62
2.3.2.2	Explainability dimensions and evaluation metrics	64
2.3.3	A five-star rating of eXplainable AI for pervasive CPS	67
3	Embedded Subsystem Design for monitoring vital parameters	71
3.1	Hazard Analysis in railway systems	71
3.1.1	HAZOP	71
3.1.2	Extracted Hazards	76
3.1.3	Hazard analysis on subsystem design	76
3.2	Biosensing Design for Cystic Fibrosis	77
3.2.1	Related work and design requirements	78
3.2.2	Sensing principle and potentiometric configuration	79
3.2.3	Electrode materials and geometrical design	82
3.2.4	Electronics design for potentiometric biosensing	87
4	EXplainable Artificial Intelligence for managing vital parameters in embedded platforms	90
4.1	Related Work	91
4.2	Software architecture	99
4.2.1	Framework	99
4.2.2	Ontology schema and data annotation	102
4.2.3	Inference and explainability	103
4.3	Explainability properties	106
5	Experiments	112
5.1	Hazard analysis in autonomous safety-critical systems	112
5.1.1	General use case and scope of the experimental campaign.	112
5.1.2	Interaction-driven cases	113
5.1.3	Experimental evidence: signal-level interpretation of scenario execution	114

5.1.4	Redundancy, scenario coverage, and implications for verification effort	115
5.2	Biosensing	117
5.2.1	Electrodes	117
5.2.2	Electronics	118
5.2.3	Case Study	122
5.3	On-device XAI	126
5.3.1	Materials and methods	126
5.3.2	Experimental outcomes	129
5.3.3	Explainability analysis	135
6	Conclusion and perspectives	140
6.1	Hazard-driven design	140
6.2	Sensing	142
6.3	On-device XAI	144
	Bibliography	147
	List of publications	172

Chapter 1

Introduction

The evolution of intelligent systems in both industrial and biomedical domains has increasingly relied on the ability to integrate heterogeneous components into a unified and reliable architecture. From autonomous transportation to wearable sensing devices, the convergence between system engineering, multimodal sensing technologies, and Artificial Intelligence (AI) has made it possible to design systems capable of perceiving, learning, and adapting in the management and monitoring of vital parameters within complex operating environments. However, this technological advancement has also raised new challenges related to safety, trustworthiness, and explainability, especially when intelligence must be embedded in hardware characterized by limited computational resources and strict operational constraints. In this context, the motivation for the research arises from the need to design intelligent embedded systems that are not only functional and efficient, but also interpretable, verifiable, and safe, in order to ensure both operational and technical reliability across different critical domains. In particular, a focus is needed on functional safety, transparent behavior, and the ability to explain outcomes and decisions. Yet, achieving these goals concurrently remains an open challenge. State-of-the-art ML models are often seen as black-boxes, highly accurate but opaque, while embedded systems are constrained by limited computational, memory, and energy resources. Bridging the gap between trustworthy AI and resource-constrained hardware requires a new paradigm: one that integrates formal reasoning, human-centered sensing, and symbolic explainability within the same embedded framework.

The work presented in this dissertation originates from interdisciplinary investigations that combine:

- **functional safety and autonomous control** for safety-critical applications such as railway automation;
- **wearable biosensing** for physiological signal acquisition and human-centered monitoring;
- **XAI on embedded systems**, enabled in the development of MAFALDA 2.0, a fully C-based, semantic, and IL and FL framework for MCUs.

Together, these research lines converge toward a unified vision: enabling trustworthy and self-explanatory embedded intelligence capable of acting autonomously and justifying its reasoning.

The increasing automation of transport systems, especially in railways, raises the question of how to guarantee safe operation when human supervision is reduced or absent. Standards such as IEC 61508 and EN 50128 establish the principles of functional safety, defining methods for ensuring that electrical, electronic, and programmable systems respond predictably to faults. Within this context, the European rail network is transitioning from traditional signaling systems to Automatic Train Operation (ATO) over the European Train Control System (ETCS). This shift promises higher efficiency and autonomy but introduces new safety challenges: the system must detect, assess, and mitigate faults autonomously. The first research line proposes and validates a Vital Control (VC) module for *ATO over ETCS*, designed to maintain safe train operation. The approach demonstrates how safety-critical automation can be realized on embedded platforms through structured hazard analysis (HAZOP) and modular architecture compliant with functional safety standards. Through the application of the HAZOP the analysis identifies preliminary hazardous scenarios within an ATO system and formalizes the corresponding safety requirements. These requirements have been then implemented in VC module capable of performing emergency braking, demonstrating how a rigorous requirements extraction and validation process can lead to the design of an effective embedded safety system. This approach highlights the importance of combining hazard analysis techniques with electronic system design to ensure the dependability of safety-critical architectures.

Parallel to the evolution of autonomous systems, the field of biosensing has transformed how physiological parameters are measured and interpreted. Modern biosensors integrate electrochemical interfaces, microcontrollers, and wireless communication to enable continuous, non-invasive monitoring. The second research direction focuses on biosens-

ing technologies for the monitoring of vital parameters. Within this perspective, the dissertation explores chloride ion detection as a biomarker for sweat composition, thermoregulation, and electrolyte balance. Chloride is particularly significant for assessing hydration status and metabolic activity. The developed wearable platform comprises:

- an electrochemical sensing layer for chloride detection,
- signal conditioning circuits and a low-power acquisition module,
- embedded signal processing and wireless data transmission.

The work explores the design and characterization of a chloride-based electrochemical sensor, capable of detecting physiological variations associated with metabolic processes. The sensor's fabrication, calibration, and response analysis illustrate the integration between material science, electronics, and signal processing, ultimately supporting the development of a low-cost and efficient sensing platform for wearable monitoring. Experimental validation confirms the reliability and repeatability of the measurements, establishing the platform as a basis for personalized health monitoring. Beyond its biomedical function, this research line emphasizes trustworthy signal acquisition, where accuracy, calibration stability, and interpretability of sensor behavior are as critical as the measurement itself. The third fundamental research line concerns the development of a XAI framework for MCUs-based platforms, referred to as MAFALDA 2.0. This component of the thesis bridges the gap between data-driven learning and symbolic reasoning, proposing a unified approach for training and inference directly on embedded devices. Recent advances in TinyML have made it possible to deploy ML models directly on embedded platforms. However, most TinyML approaches prioritize Neural Architecture Search for predictive performance and inference efficiency optimization, often neglecting capabilities such as on-device training, IL, FL and eXplainability. Existing solutions are typically designed as static, pre-trained models, with limited adaptability to new data and environments, and without the ability to formally represent or explain the decision logic. Additionally, the few available XAI approaches for edge computing and embedded devices rely on post-hoc interpretability methods that require external computational resources and only provide approximate explanations of model behavior, making them unsuitable for real-time embedded deployment. MAFALDA 2.0 addresses these limitations by integrating symbolic reasoning: the system performs classification tasks through a semantic matchmaking process, using Web Ontology Language (OWL) 2 ontologies to represent domain knowledge

and to produce explicit, logically interpretable reasoning structures. Unlike conventional TinyML models, MAFALDA 2.0 enables training and inference directly on-device while preserving formal interpretability of its outcomes. The framework supports IL, allowing models to adapt to new data without retraining from scratch, and FL, enabling distributed updates across multiple devices while preserving data privacy. Implemented entirely in C, Mafalda 2.0 runs seamlessly on PCs, single-board computers, and MCUs, providing a unique combination of on-device training, explainability, and compliant semantic representation.

The contribution of this dissertation therefore lies in the integration of engineering, sensing, and AI paradigms within a coherent methodological framework. By extending safety analysis methods to embedded design, demonstrating biosensing applications for vital parameter monitoring, and developing an explainable and adaptive AI architecture deployable on MCUs, this research advances the current state of knowledge in embedded intelligent systems. It shows that safety, reliability, and explainability can be approached as complementary aspects of a unified design process. In this sense, the work contributes to the evolution of embedded intelligence toward trustworthy, interpretable, and self-adaptive systems, capable of operating transparently across both biological and artificial domains. The remainder of this dissertation is organized as follows. Chapter 2 provides the theoretical and technological foundations of the research. It introduces the concept of vital parameter monitoring in both artificial and biological domains, discussing the design principles underlying safety-critical systems, healthcare devices, and affective computing. This chapter first explores the application of hazard analysis methodologies, particularly HAZOP, within autonomous railway systems, showing how safety requirements can be systematically extracted and implemented in embedded control boards. It then extends the notion of vital monitoring to the biomedical field, through the design and characterization of an electrochemical chloride-based sensor for continuous detection of physiological parameters. Finally, it addresses the role of AI in managing and interpreting sensory data within embedded architectures, providing an overview of classical ML algorithms, IL, and FL paradigms, and the use of Knowledge Representation and Reasoning (KRR) for enabling semantic interpretability. Chapter 3 provides the proposed design choices performed for both safety-critical systems requirements and sensing interface for wearable electrochemical sensor. Chapter 4 presents the proposed MAFALDA 2.0 framework, which integrates symbolic reasoning and statistical learning in a unified architecture optimized for embedded devices. The chapter describes the internal representation model, the incremen-

tal and federated learning mechanisms, and the process of transforming numerical input data into semantic descriptions suitable for reasoning and explainability on MCUs. Chapter 5 reports the experimental evaluations carried out on the hazard-driven derivation, the wearable platform for chloride detection and MAFALDA 2.0. The first one provides a basis for evaluating the temporal behavior of safety-related functions applied to an autonomous system enabling emergency brake. The second consists in a performance assessment of sensing subsystems, electronic interface and a case study evaluation on a pediatric patient. The experiments on MAFALDA 2.0 include performance evaluations across heterogeneous hardware platforms, a case study in comparison with similar state-of-the-art on-device learning approaches, and qualitative explainability assessments through symbolic reasoning visualization. The results demonstrate the feasibility of achieving both training and inference on-device, maintaining satisfying levels of accuracy and efficiency along with interpretability. Finally, Chapter 6 summarizes contributions and future directions.

Chapter 2

Monitoring and Managing Vital Parameters

2.1 Monitoring vital parameters

Monitoring of vital parameters refers to the acquisition and processing of physiological or operational signals that reflect the health status of natural systems, *e.g.*, human, animals, or artificial systems, *e.g.*, robots, industrial machines, autonomous vehicles. Therefore, vital parameters can be broadly defined as measurable quantities that provide fundamental information on the state of a system, across an expansive range of natural and artificial systems, serving to assess health, optimize performance, ensure safety, and predict states. In the biological domain, vital signal monitoring encompasses everything from tracking physiological markers up to measurements of heart rate (HR), respiration rate (RR), blood oxygen saturation, blood pressure (BP), body temperature, and other biosignals [115], to observe macrolevel environmental indicators, for example, atmospheric CO_2 levels, ocean temperature, biodiversity indices. Vital signs play an instrumental role in healthcare by acting as crucial indicators of health status. They facilitate early diagnosis of pathological conditions, the continuous monitoring of chronic disease, and the proactive prevention of critical events, including cardiac arrest and sepsis [114, 115].

In this context, the role of wearable devices and embedded architectures has become increasingly important. These devices enable continuous non-invasive monitoring, providing real-time data, and improving clinical efficiency. As demonstrated in [114], the development of low-power devices integrated with biosensors has allowed the acquisition

and processing of physiological signals for extended monitoring periods, thereby facilitating early diagnosis and improving quality of life. Concurrent studies in hospital settings have demonstrated that wearable sensors can facilitate the detection of clinical deterioration compared to conventional observations made by healthcare personnel. This enhanced capability enables more timely interventions, thus reducing the risk of mortality [115].

In artificial systems, the concept of vital parameters can be extended to operational signals that reflect critical indicators for the safety and operational reliability of engineered devices, such as temperature, pressure, electrical voltage, current, vibration, memory integrity, processor performance, and status signals from mechanical and electronic subsystems. When these parameters are subject to continuous monitoring, they function as indicators of the ‘health’ of the system, enabling prognostics, fault detection, and predictive maintenance [126]. In [2] the pivotal role of embedded systems is emphasized for safety-critical domains, such as the automotive, aerospace, healthcare, and industrial sectors, where failures can lead to disastrous consequences. Consequently, continuous monitoring of these operational metrics is imperative to ensure reliability, avert unexpected disruptions, and ensure the adherence to stringent regulatory standards [2].

From this dual perspective, the integration of heterogeneous sensors, processing boards, and communication modules is required in the design of embedded monitoring architectures for vital parameters. In addition, they must confront rigorous design limitations, including, but not limited to, minimal energy consumption, dependability, adherence to diverse regulatory frameworks, and, particularly in safety-critical domains, congruence with hazard and risk assessment methodologies [74, 232].

2.1.1 Motivating scenarios

2.1.1.1 Safety-critical autonomous systems

The European railway sector is undergoing a paradigm shift towards increased automation and enhanced operational safety. Within this framework, the Shift2Rail [196] initiative has promoted the deployment of Automatic Train Operation (ATO) technologies, progressively advancing toward Grade of Automation 4 (GoA4), also known as Unattended Train Operation (UTO). The proposed solution is the use of an ATO over the European Train Control System (ETCS) architecture, that is, an Automatic Train Operation system capable of driving a train and autonomously managing all train operations in full integra-

tion with the European Rail Traffic Management System (ERTMS) traffic protection and management system, as represented in Figure 2.1.

Unlike lower levels of automation (GoA1–2), where drivers remain on board to provide supervision, GoA4 requires complete elimination of human presence in the cabin. This scenario underscores the imperative for uninterrupted monitoring of vital parameters, encompassing both physical quantities (*e.g.*, current, voltage, vibration, temperature) and logical signals (*e.g.*, subsystem status, operator vigilance, odometry). These parameters must be systematically supervised to ensure that hazardous conditions are detected in real time and mitigated through immediate safety actions, such as emergency braking.

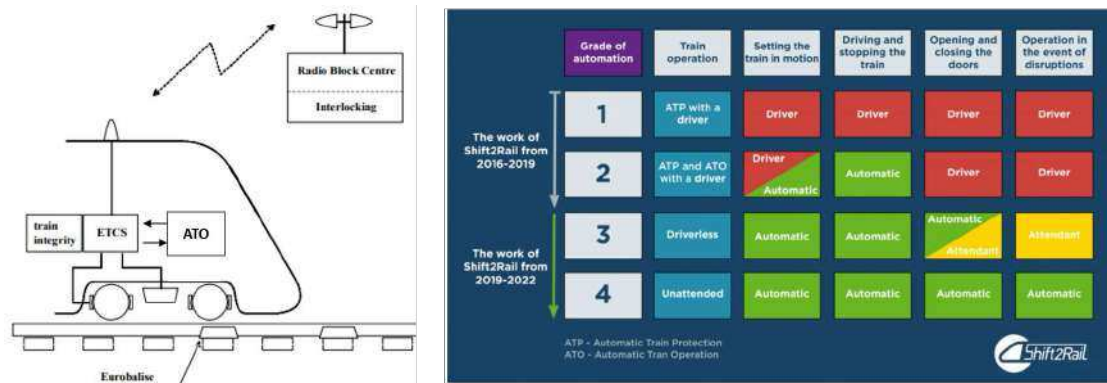


Figure 2.1: The ATO system interfaces with ground infrastructures and all onboard systems, including and especially the ETCS, to issue traction commands in the most efficient way possible. This means that the ATO system is responsible for processing traction commands and travel profiles, optimizing them by considering energy consumption, timing requirements, and route punctuality. With Level 4 automation, there is no human presence onboard; safety supervision is handled by the ETCS, and all procedures and operations are fully unattended and automated, both between stations and within stations.

It is imperative to acknowledge that the design of safety-critical subsystems in railway applications cannot be reduced to the optimization of hardware and software performance alone. Instead, it is subject to a rigorous lifecycle regulated by hazard analysis (HA) methodologies and international standards. The overarching framework is delineated by IEC 61508 [111], which specifies the *functional safety requirements for electrical, electronic, and programmable electronic (E/E/PE) systems*. Within the context of the railway domain, certain CENELEC standards serve to complement IEC 61508. EN 50126 [44] establishes the framework for the RAMS (Reliability, Availability, Maintainability, Safety) process. EN 50128 [42] delineates the requirements for safety-related software in control

and protection systems. EN 50129 [43] addresses safety-related electronic systems. Additional standards, such as EN 50155 (electronic equipment for rolling stock) and EN 50124 (insulation coordination), ensure compliance at the hardware level. Collectively, these regulations establish Safety Integrity Levels (SIL) that apply throughout the entire lifecycle of a system, from concept definition to verification and validation. Hazard Analysis, therefore, becomes a structural phase of the engineering process, informing architectural choices such as redundancy levels, sensor allocation, and fault-tolerant I/O design, as well as the selection of scheduling policies in Real-Time Operating Systems (RTOS) and exception handling in on-board logic.

In the domain of safety-critical development, the quality and dependability of a product are of paramount importance and must be addressed in all design phases, from concept definition to verification and validation (V&V) of safety and security features [174]. According to IEC 61508, the most critical systems require the highest safety integrity requirement, SIL 4, which requires that the hazard analysis must be embedded into the system design from its earliest stages [143]. The Preliminary Hazard Analysis (PHA) is a pivotal phase in the risk management process. It involves determining the acceptable residual risk level, identifying potential hazards, and establishing additional safety requirements to mitigate them. This phase is crucial in determining the reliability and safety of a system or process, as outlined in the research [158, 209]. This analysis must encompass not only single-component failures but also the interaction of subsystems, which, in complex railway architectures, frequently engender unanticipated hazardous conditions.

A variety of techniques have been employed throughout history for the purpose of hazard analysis. Failure Modes and Effects Analysis (FMEA) and Fault Tree Analysis (FTA) remain foundational approaches; however, they are inadequate for large-scale, highly integrated systems. The Hazard and Operability (HAZOP) technique has demonstrated notable efficacy in the railway sector, and it is formally recognized in the standard CENELEC 50126 for complex safety-critical domains [121].

Recent advancements have sought to augment the scalability of HAZOP by integrating it with contemporary tools and methodologies. Examples of such methods include simulation-based hazard assessment [50], *Stateflow*¹-driven hazard identification [51], and ontology-based classification of hazardous events [233]. These model-based strate-

¹Stateflow home page: <https://mathworks.com/products/stateflow.html>

gies facilitate more exhaustive coverage and enhance efficiency by reducing redundancy in hazard identification. The shared objective of these entities is to ensure that all safety-critical requirements are extracted systematically and consistently, thereby fortifying the correlation between hazards and design choices.

The increasing autonomy of GoA4 architectures introduces new challenges in terms of supervision, reaction times, and system-level accountability. In the absence of an onboard driver, safety can no longer rely on human intervention, but must be ensured through the systematic identification of hazardous scenarios and their translation into explicit, enforceable safety requirements. Although norm-compliant hazard analysis techniques provide the methodological foundation for this process, the transition toward highly integrated architectures, *e.g.*, ATO over ETCS, increases the proportion of hazards that are interaction-driven and mode-dependent. Unsafe behavior may emerge from inconsistent assumptions across subsystems, *e.g.*, status observability, degraded states and isolation procedures, even when individual components are operating within their local specifications.

A practical consequence of this analysis is that deviation-based analyses, such as HAZOP, which are effective and formally recognized for complex domains, [121] may produce large sets of partially overlapping scenarios when applied to systems characterized by multiple interacting subsystems and operational modes. This issue is not merely methodological; it is also an engineering problem. The use of large and redundant scenario sets leads to an increase in verification effort and a reduction in maintainability. This is due to the fact that it becomes more difficult to distinguish requirements that are strictly necessary for safety coverage from requirements that are refinements introduced to improve robustness, diagnosability, or operational completeness. Recent model-based strategies, such as simulation-supported assessment [50], Stateflow-driven identification [51], and ontology-based classification [233], have been developed to address the scalability challenge. These strategies aim to strengthen traceability and reduce redundancy in hazard identification. In this thesis, the resulting need is addressed by adopting a hazard analysis process that is not only norm-compliant but also oriented toward (i) explicit mapping between hazardous scenarios, mitigations, and requirements, and (ii) quantitative assessment of system behavior under hazardous conditions. This need represents a key challenge in the design of safety-critical embedded railway subsystems and motivates the hazard analysis approach presented in Chapter 3.1. Its experimental validation is detailed in Chapter 5.1, wherein representative scenarios are complemented with a coverage-oriented perspective

on hazard/requirement completeness.

2.1.1.2 Healthcare

In the healthcare domain, the monitoring of natural vital parameters often involves biochemical markers that provide direct insight onto physiological state of human body. For example, the chloride ions concentration in sweat is a well-established key diagnostic indicator for cystic fibrosis (CF), a genetic disorder that affects ion transport mechanisms and requires early and continuous monitoring to improve patient outcomes. Traditional diagnostic approaches are typically laboratory-based and episodic, limiting their suitability for long-term or continuous observation [65, 63].

Translating laboratory-based methods into wearable, embedded subsystems has therefore emerged as a key research challenge. Wearable biosensing solutions enable non-invasive, continuous acquisition of biochemical parameters, supporting early diagnosis, disease management, and personalized healthcare. In this context, embedded systems play a central role by providing signal acquisition, processing, and communication capabilities while operating under several constraints in terms of power consumption, size, and reliability [140, 64].

Biosensing devices are subject to regulatory requirements that span the entire system lifecycle. Standards such as ISO 13485 for quality management and ISO 14971 for risk management impose structured design processes aimed at ensuring safety, reliability, and traceability. Consequently, the development of embedded biosensors for healthcare applications cannot be decoupled from considerations related to process quality, risk mitigation, and system validation [72, 73]. Electrochemical biosensors have gained significant attention in this context due to their high sensitivity, compatibility with miniaturized architectures, and suitability for integration into low-power wearable devices. Their ability to operate with simple analog front-end circuitry makes them particularly attractive for embedded implementations targeting point-of-care and continuous monitoring scenarios [140, 227]. However, achieving clinically meaningful performance requires careful co-design of sensing elements, electronics, and calibration strategies within a framework that is compliant with medical device regulations.

From an architectural point of view, miniaturization has favored the adoption of planar electrodes, which are fabricated using deposition or printing techniques. These techniques reduce bulk and simplify integration into wearable devices. Recent studies have demon-

strated that electrode geometry exerts a substantial influence on the sensitivity, temporal response, and long-term stability of the sensor. Specifically, the surface-to-volume ratio and the uniform distribution of the active material have been shown to enhance linearity performance and reduce background noise [140, 227].

These considerations motivate the design of embedded biosensing architectures specifically tailored for screening of diseases related to biological fluids, where sensing accuracy, stability, and energy efficiency must be jointly addressed. Section 3.2 focuses on the design of an electrochemical biosensor for chloride monitoring in sweat, detailing the technological choices and design rationale that underpin the experimental validation presented later in Section 5.2.

2.1.1.3 Affective Computing

Starting from the motivating scenario provided by the development of electrochemical monitoring applications for the detection of physiological biomarkers, such as potentiometric electrodes for measuring chloride concentration in sweat, the scope of the investigation has been expanded to encompass other scenarios in which vital parameters assume a pivotal role. In addition to the diagnosis of specific organic conditions, biosignals, including heart rate, heart rate variability, electrodermal activity, and respiration can provide fundamental information about the emotional and cognitive state of individuals. This scenario belongs to the field of affective computing, a multidisciplinary domain that combines biosensing, wearable systems, and AI to support both human-machine interaction and clinical applications in psychiatry, psychology, economy and more.

The integration of AI into human environments has prompted a paradigm shift in the design of intelligent systems, moving from purely functional to emotionally responsive agents. At the core of this transformation lies affective computing, a multidisciplinary research area initially introduced by Rosalind Picard in 1997 [165] with the aim of creating computing systems able to detect, interpret, simulate, and respond to human emotions, thus creating a natural human-computer interaction [166, 164].

In the context of daily human communication, emotions are predominantly expressed through cues such as facial expressions, voice, and language. However. It must be noted that these data may not always accurately reflect true emotional states, as they are contingent on social environment and cultural background, and can be deliberately masked. Physiological signals are not as easily controlled, offering a more objective and

reliable means of detecting, recognizing, and interpreting emotions in real time [222]. The autonomic nervous system is responsible for the regulation of various physiological functions, including heart rate, respiration, perspiration, and pupil dilation. Measurement of these reactions offers insights into emotional states. The main physiological signals include electroencephalogram (EEG), electrocardiogram (ECG), electrodermal activity (EDA, a.k.a. skin conductance – SC – or galvanic skin response – GSR), respiration, and electromyogram (EMG) [142, 93].

Evidence of the feasibility of using physiological sensors for affect recognition in real-world settings was provided by Healey & Picard (2005) [104], who monitored drivers during actual road conditions using ECG, EMG, skin conductance, and respiration signals. Their findings indicated a robust correlation between skin conductance and heart rate measures and stress levels, thereby substantiating the efficacy of biosignals in discerning emotional states in dynamic and uncontrolled environments. This pioneering work demonstrated the potential of affective computing to transition from laboratory experiments to naturalistic contexts, thereby laying the foundation for the utilization of embedded systems in everyday life.

Advancements in the field of wearable technology have enabled the development of devices capable of continuous acquisition of multimodal biosignals. A prominent example is the *Empatica E3* wristband, which was described by Garbarino *et al.* (2014) [91]. This device integrates sensors for photoplethysmography (PPG), electrodermal activity (EDA), skin temperature, and accelerometry. The device was engineered to be lightweight, portable, and unobtrusive, with the capacity to facilitate both Bluetooth Low Energy streaming and local data logging. Its capacity to measure heart rate variability (HRV) and EDA in real-world settings has led to its widespread adoption as a research instrument in the domains of stress and emotion recognition studies [91].

The Body Area Network (BAN) architecture is instrumental in the design of emotion-aware and health-monitoring systems. A BAN is a network of miniaturized, intelligent sensors that can be implanted in the body, worn on the body surface, or positioned in close proximity to the body. These sensors are designed to continuously collect, process, and communicate physiological and contextual data [132]. The IEEE 802.15.6 standard [110] establishes the technical foundations for BANs, supporting narrowband, ultra-wideband, and human body communication, and provides multiple security levels, addressing privacy concerns in medical and affective applications. A typical BAN comprises three layers: sensing, communication, and application. The sensing layer involves biosensors, such as

ECG, EDA, EEG, or smart textiles. The communication layer relies on BLE, ZigBee, or Wi-Fi. The application layer is responsible for decision-making and may integrate edge computing devices to reduce latency and preserve privacy [82, 129, 159].

The integration of physiological biosensors with wearable computing devices facilitates the continuous monitoring of internal affective states that are otherwise challenging to assess. These systems have the capacity to provide psychiatrists and clinicians with objective data to monitoring anxiety, depression, stress, and other mood disorders in daily life contexts [104, 91]. The capacity to assess long-term variations in HRV, respiration, and skin conductance makes affective computing a promising instrument for precision psychiatry. Furthermore, the implementation of privacy-preserving and FL methodologies can ensure that the processing of sensitive emotional data occurs at the local level on embedded devices. This approach helps mitigate ethical concerns while facilitating personalization [133].

In summary, affective computing exemplifies the potential for embedded monitoring of physiological signals reflecting emotions to fulfill a dual function: facilitating adaptive human-machine interaction and supporting the clinical management of psychiatric disorders. Consequently, the integration of multi-sensor acquisition, robust edge processing, energy efficiency, and compliance with data protection and medical regulations is imperative in the design of embedded systems intended for affective computing. This integration is crucial to ensure the provision of trustworthy and clinically relevant monitoring solutions.

2.1.2 Embedded monitoring architectures

After outlining the main application scenarios, it is necessary to analyze the embedded monitoring architectures that translate these requirements into concrete solutions based on sensors, processing boards, and dedicated operating systems.

2.1.2.1 Sensors and microdevices

Sensors are regarded as the primary interface between the physical world and embedded intelligent systems. These devices facilitate the measurement of physical, chemical, or physiological quantities and their conversion into electrical signals suitable for digital processing. The sensor, as the initial stage of the signal chain, provides the raw data necessary for monitoring, control, and decision-making processes. In the context of

embedded applications, sensors are integrated within compact nodes that also include conditioning electronics, microcontrollers, communication interfaces, and power sources. The components form a *sensor node*, that is capable of acquiring, processing, and transmitting data within distributed or networked systems [6].

Recent advances in low-cost and low-power electronics have enabled the use of wearable sensors capable of continuously recording physiological signals [222]. Compared with remote sensing approaches such as camera-based or radar-based monitoring, wearable systems offer higher robustness in clinical, laboratory and pervasive applications [109].

Sensing technologies and physiological parameters. Physiological monitoring relies on transduction mechanisms that convert variations in electrical, optical, mechanical, or thermal energy into voltage or current signals. Typical healthcare measurable parameters include ECG, EMG, electroencephalogram (EEG), skin temperature, blood oxygen saturation (SpO_2), blood pressure, respiration rate, and body motion. Each of these quantities is associated with specific sensing modalities [6].

Electrical transducers detect bioelectric potentials using electrodes and high-impedance amplifiers. They are essential for ECG, EMG, and EEG acquisition [6]. Electrocardiography (ECG) sensors measure cardiac electrical activity, providing heart rate (HR) and heart-rate variability (HRV) features that are strong correlates of stress and arousal. Typical acquisition are made using wearable devices like chest straps, wristbands and smartwatches [93].

Electroencephalography (EEG) captures brainwave oscillations across multiple frequency bands, *i.e.*, delta, theta, alpha, beta, gamma, offering direct access to cognitive and emotional states [85]. Portable EEG headsets (*e.g.*, Emotiv Insight [85]) have made neuro-affective monitoring feasible outside laboratories, and on-chip implementations such as ear-EEG have been demonstrated for real-time emotion detection through embedded classification algorithms [141].

Electromyography (EMG) measures the electrical activity of skeletal muscles through surface or needle electrodes, reflecting tension, expression, and movement. It measures the degree of muscle activation by collecting the voltage difference generated during muscle contraction [85]. Recently, [122] used a biomedical signal amplifier to recognize and classify human emotions. Height electrodes, embedded on the amplifier, have been positioned on user faces and the signals have been recorded when users acted out joy, sadness, surprise, disgust, anger, fear, and neutral emotions. All participants were asked to express specific emotions or relax or do nothing in 30 seconds; all electrodes recorded

the signal power bond to muscle contractions.

Optical transducers employ light emission and detection to monitor pulsatile blood flow and oxygenation levels, forming the basis of photoplethysmographic (PPG) and pulse oximetry sensors. Optical transducers are highlighted for their sensitivity and simplicity but also for susceptibility to motion and ambient-light artifacts, which must be mitigated through adaptive filtering. [6]. PPG sensors use optical non-invasive pulses generated by a red or near-infrared light source (light emitting diode) and receive the reflected light with a photodetector, to measure light absorption changes caused by blood-volume variations [93]. This technology has been first proposed in 1930. Nowadays, an increasingly used technique is remote photoplethysmography (rPPG). It consists of a multiwavelength RGB camera that identifies skin color variations on human face caused by changes in blood volume during heartbeat. Thus, PPG provides HR and HRV non-invasively [38].

Mechanical transducers, such as piezoelectric and strain-based sensors, measure pressure, vibration, and motion, often integrated into wearable patches or smart textiles [6]. Respiration (RSP) biosensor is a chest elastic belt worn in the thorax or the abdominal area and the stretch measurements are expressed as change of voltage level due to respiration compressing the attached sensor (inhalation and exhalation cycles) [207]. Breath sensors estimate the amount of air exchange in the lungs, which should be performed talking or moving [93]. They are employed to measure the respiration pattern, *i.e.*, how fast and deep a person is breathing. The way a person breathes is associated with a variety of emotional states, both negative, like anger and stress, and positive, like joy [85]. [126] proposed a model for detection of mental stress levels using sensors embedded on smartwatches. Once the respiration rate was predicted, the origin and predicted biomarkers were used to classify the stress level.

Thermal transducers use thermistors or semiconductor sensors to capture variations in body temperature or environmental conditions [6]. EDA reflects sympathetic nervous-system activation through changes in skin conductivity due to variation of sweat gland activity. Typically, electrodes are placed across the palm or between the index and middle fingers. However, EDA readings can be distorted by ambient temperature, humidity, or motion, necessitating the use of additional environmental sensors to filter artifacts [222].

Eye-tracking systems are commonly adopted in desktop setups, mobile devices, as well as in virtual reality headsets and they are classified into three categories: eye-attached tracking, optical tracking and electric potential measurements. The relevant features related to eye tracking are pupil diameter, electro-oculographic (EOG) signals,

pupil position, fixation duration, distance between sclera and iris, motion speed of the eye and pupillary responses [134]. A recently-proposed method for emotion recognition acquires the pupillary signal using a mobile eye-tracker mounted onto a pair of glasses. A matching pairs card game framework was developed, and an optical sensor-based triggering system was used to monitor start and end time for game trials. The authors determined an alteration of the emotional status of participants [39]. EOG is also used to measure the resting potential of the eye and fluctuations caused by horizontal and vertical motions, using electrodes placed above and below or on the sides of the eyes [85]. Decreased blink rates typically signify focused attention, while irregular movement can indicate stress or fatigue [222].

Multimodal and hybrid configurations. Embedded nodes can combine multiple transducers and share conditioning, conversion, and wireless interfaces [6]. Several studies have adopted the use of multimodal biosignals for classification tasks. For instance, [142] proposed a low-invasive experimental setup by combining autobiographical emotion memory task with a wristband device measuring blood volume pressure, electrodermal activity, skin temperature and acceleration. At each defined temporal step, each participant alternates between a moment of neutral topic writing and one in which a random task distracts the patient. The aim is to elicit four different emotions: sadness, excitement, fear and relaxation [142]. In [101] a negative-emotion recognition system integrates EEG, heart rate, galvanic skin response, and skin temperature through a wearable Internet of Things (IoT) band and smartphone application. The multimodal data were transmitted via Bluetooth to an Android app and processed on a server achieving 93% recognition accuracy.

In embedded platforms and hybrid nodes, sensor selection must balance measurement accuracy, response time, and energy cost and manage synchronization, data-fusion, and power-allocation challenges. The reference values for sampling frequency, data rate, and quantization for each physiological parameter must be followed, illustrating how design constraints directly influence power consumption and communication bandwidth [6].

Sensor node architecture. General sensor node architecture is composed of five main subsystems [6]:

1. the *transducer*, responsible for converting a physical phenomenon into an electrical signal;
2. the *signal-conditioning stage*, which amplifies, filters, and linearized the output;

3. the *analog-to-digital converter (ADC)*, where the signal is digitized;
4. the *microcontroller* or *processing unit*, which executes local computation and control algorithms;
5. the *communication* and *power subsystems*, enabling data transmission and energy management.

At the hardware level, **microdevices** are presented as miniaturized transducers fabricated through Micro-Electro-Mechanical System (MEMS) or thin-film technologies. These processes enable the integration of mechanical, optical, and electrical elements on the same substrate, achieving compactness, low mass, and high sensitivity. Common fabrication steps include deposition, lithography, etching, and packaging, which together define the geometry and performance of the device. Piezoelectric, capacitive, and resistive transduction methods are used for mechanical or pressure sensing, while bioelectrochemical microsensors integrate conductive polymers and flexible substrates to achieve conformal contact with the skin. In biomedical applications, the same is applied to microfluidic and bioelectrochemical sensors, which are designed for compatibility with soft substrates and biological interfaces. Packaging and protection layers are essential design aspects, as well. They ensure mechanical stability, electrical insulation, and, when necessary, biocompatibility. The choice of materials, metals, polymers, or ceramics, depends on the operating environment and expected lifetime of the device [6].

These design principles have been recently extended toward microdevices that simultaneously perform sensing and actuation functions within the same MEMS structure. In [11] a resonant micro-sensor and micro-actuator unifies both capabilities in a single chip. The device consists of a clamped-guided curved microbeam connected to straight and V-shaped actuator beams, fabricated using a specific process. Its operation relies on the interaction between electrothermal excitation and Lorentz-force coupling, allowing dynamic tuning of resonance modes. The microdevice is capable of modulating the applied thermal power, thereby enabling the switch between a high-sensitivity magnetic-sensing mode and a magnetic-field-insensitive actuation mode which generates 4 μm displacement with low power consumption of approximately 43 mW. This duality exemplifies the capacity of MEMS platforms to deliver multifunctional, reconfigurable smart nodes that adapt their behavior through internal multiphysics coupling, rather than structural modification.

Energy Management and Integration Energy consumption is a key constraint in

embedded sensing; power is primarily dissipated in data acquisition, computation, and wireless transmission. Energy efficiency is addressed through low-power analog front-ends, voltage regulation, dynamic control of the operating modes, and energy-aware communication protocols. Several techniques such as voltage scaling, power gating, and adaptive sampling adjust the activity of the sensor node to extend the operational lifetime without sacrificing data quality [6].

Integration with wireless networks, like Bluetooth, Zigbee, or proprietary links, completes the architecture, transforming individual sensors into cooperative nodes within Wireless Sensor Networks (WSNs) or Body Area Networks (BANs). Each node can perform partial processing locally and forward processed data to higher-level units, reducing latency and bandwidth requirements. For embedded nodes capable of executing on-device ML training, **federated learning** extends this WSN concept; it is analyzed in greater detail in Section 2.2.3.

Safety, reliability, and calibration. Reliability of embedded sensors is related to calibration and environmental robustness. There are several mechanisms to ensure measurement consistency, including periodic calibration routines, redundant sensing, and fault detection at the node level. Temperature compensation, shielding from electromagnetic interference, and filtering of motion artifacts are identified as essential techniques to maintain data quality over time. Fault-detection and self-diagnostic techniques allowing each node to assess its operational status and report degradation or failure to supervisory layers, as well as periodic recalibration to compensate for drift or environmental interference. These mechanisms can ensure long-term stability, enabling sensors to operate reliably in continuous monitoring scenarios [6].

2.1.2.2 Processing boards

Processing boards are the physical and computational substrate of intelligent embedded systems. Their design must balance computational performance [221, 185], electrical and electronic characteristics [19, 189], and system integration [21, 156] and determines how efficiently data are sensed, processed, and transmitted in environments where latency, energy, and memory are strictly constrained.

Processing Requirements. Typical MCU devices used in TinyML applications integrate ARM Cortex-M cores operating between 48 and 240 MHz, with SRAM capacities of 64–512 kB and flash memory up to 4 MB. Such boards are designed to execute mod-

els with a small number of parameters, often limited to a few hundred kilobytes, where inference time must remain within tens of milliseconds to satisfy control or monitoring tasks. These constraints push research toward lightweight architectures (*e.g.*, MobileNet, SqueezeNet, MicroNets) and application quantization or pruning methods [185].

Balfour *et al.* [19] demonstrated that in embedded processors, 70% of the total energy is consumed by data and instruction transfers rather than arithmetic operations. Their *Ensemble Processor* architecture introduced distributed register files physically located near computation units, reducing memory-access energy by 15× for data and 58× for instructions compared with a standard RISC baseline. The data locality is the principal factor that determines performance and energy efficiency in embedded systems.

Edge computing platforms are based on heterogeneous System-on-Chip (SoC) architectures integrating Central Processing Unit (CPU), Graphics Processing Unit (GPU), Neural Processing Unit (NPU), Digital Signal Processor (DSP) and/or other accelerators within the same chip to improve computational and power efficiency, and these components share internal memory and bus communication on-chip [221].

A review of over 30 TinyML-compatible platforms is in [189], reporting detailed characteristics such as RAM (128 kB – 1 MB), clock frequencies (48–480 MHz), and power consumption typically below 10 mW per active inference cycle. Boards like the Arduino Portenta H7 and SparkFun Edge combine low-power MCUs with on-board sensors and wireless modules, supporting end-to-end pipelines from sensing to inference. Wang *et al.* [221] discuss the scalability of such devices within the broader edge-AI hierarchy, where computation is distributed between MCUs, heterogeneous System-on-a-Chips (SoCs) (*e.g.*, NVIDIA Jetson, Rockchip, Movidius), and high-performance edge servers. They emphasize that the heterogeneity of AI hardware, from 8-bit MCUs to multi-core NPUs, requires flexible scheduling and runtime adaptation to maintain deterministic response across neural nodes.

Electrical and Electronic Characteristics. At the circuit level, the energy behavior of processing boards is largely governed by their electrical organization. Reducing interconnect length and pipeline depth minimizes dynamic switching power and leakage currents. In [19] the processor pipeline was exposed to the compiler, eliminating unnecessary latches and registers; an approach that later influenced dataflow-based embedded processors and Field-Programmable Gate Array (FPGA) AI accelerators.

At the electronic level, processing boards for on-device AI must integrate low-power design, efficient power-management subsystems, and optimized interconnects to support

continuous operations. SoC platforms typically employ dynamic voltage and frequency scaling (DVFS) and clock-gating mechanisms to minimize leakage and switching power, while hardware accelerators such as NPUs or DSPs are included to offload computationally intensive tasks. Energy consumption management represents a primary design concern: voltage regulators, efficient DC-DC converters, and on-chip power domains enable adaptive scaling according to workload intensity [221]. Furthermore, signal integrity and noise isolation must be ensured in mixed-signal layouts where analog sensor interfaces coexist with high-frequency digital buses. These aspects directly impact inference stability and long-term reliability of edge nodes [221].

Saha *et al.* [185] note that dynamic voltage and frequency scaling (DVFS), clock gating, and low-leakage CMOS transistors are now standard mechanisms to achieve sub-milliwatt consumption levels during inference. The authors also discuss memory compression and on-chip buffering, which limit the need for high-energy off-chip accesses. These techniques are critical for wearable or battery-powered nodes, where power density and heat dissipation directly affect system longevity.

The electrical parameters of common TinyML boards are operating voltages between 1.8 and 3.3 V, current draws of 1–30 mA during inference, and thermal envelopes compatible with passive cooling. The electrical design of TinyML platforms is optimized for ultra-low-power operation, often below 1 mW per inference, and integration of sensors and wireless modules must occur under the same energy and noise constraints [189].

From an electronic-design perspective, signal synchronization and clock-domain management are fundamental to ensuring reliable operation of complex embedded systems [21]. Finite-state machine (FSM) controllers coordinate communication between modules, prevent metastability, and ensure deterministic timing across data paths. Such design methodologies are essential when integrating multiple functional units, such as sensor interfaces, memory controllers, and AI accelerators, on a single printed circuit board (PCB).

Integration challenges. The integration of processing boards into intelligent embedded systems remains one of the most complex design stages. According to [221], effective system integration requires co-optimization of hardware, runtime, and communication layers. Differences in data formats, driver architectures, and memory management models between frameworks can introduce incompatibilities during deployment. Several open-source frameworks have been developed by AI companies with community involvement

and are widely used, including TensorFlow Lite², ExecuTorch³, ONNX Runtime⁴ and OpenVINO⁵. Incompatibilities require middleware capable of adapting model representations to heterogeneous processors and instruction sets. address the same issue at the digital-logic level, describing modular FSM coordination as a strategy to decouple hardware functions and reduce timing interference. This modular approach also simplifies testing and fault isolation, key properties for safety-critical embedded systems.

At runtime, synchronization between the ML and the sensing–actuation pipeline must be guaranteed to avoid timing violations. The importance of hardware–software co-design, where algorithmic simplifications, *e.g.*, operator fusion, mixed precision, are aligned with the board’s architectural features [221]. Integration complexity also increases when multiple nodes cooperate in distributed edge networks: consistency of local models, communication latency, and adaptive offloading mechanisms become critical factors.

In [156] the perspective further expands to system-level integration, discussing the interaction between embedded nodes, edge gateways, and cloud layers. The authors identify latency, bandwidth, and energy allocation as key aspect for large-scale deployments. Putting emphasis on adaptive partitioning of AI workloads and communication, data processing can shift dynamically between the device and the edge infrastructure according to resource availability and application criticality.

2.1.2.3 Operating Systems

Embedded systems are characterized by a cooperation between hardware and software components, designed to perform a specific set of functions under constraints in terms of time, resources and energy [145]. In this context, Operating System (OS) is the software component that ensures deterministic task execution, efficient resource management, and reliable interaction with the surrounding physical environment [69]. When the system behavior depends not only on the logical correctness of computations but also on the timing of responses, the OS must operate under real-time constraints. Such systems are commonly known as Real-Time Operating System (RTOS) [145, 194]. Thus, RTOS provides deterministic task scheduling, resource management, and communication services

²TensorFlow Lite project: <https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite>

³ExecuTorch (formerly PyTorch Mobile) home page: <https://docs.pytorch.org/executorch/stable/index.html>

⁴Open Neural Network Exchange Runtime: <https://onnxruntime.ai/>

⁵OpenVINO project: <https://github.com/openvinotoolkit/openvino>

under strict temporal constraints for embedded computing [37, cap. 1].

While, a General-Purpose Operating System (GPOS), *e.g.*, Linux or Windows, focuses on maximizing system throughput and minimizing the average response time, a RTOS is designed to ensure predictable temporal behavior and the ability to guarantee the execution of the operation into defined deadlines, even if under complex conditions [145] [37, cap. 1]. According to Laplante, a real-time system is defined by two factors. Firstly, the "correctness of its output" is determined by the logical result. Secondly, the time at which that result is produced is also a contributing factor to the definition [128, cap. 1]. RTOS for constrained devices must provide low interrupt latency, efficient task synchronization, and predictable context switching, while maintaining a small memory footprint [100]. These characteristics are essential for resource-constrained devices like sensors, actuators, or IoT nodes [194].

The literature distinguishes two fundamental categories: hard real-time systems, where missing a deadline can lead to catastrophic failure (*e.g.*, avionics, medical control systems), and soft real-time systems, where a delay only results in acceptable degradation of performance (*e.g.*, multimedia streaming) [102]. The essential difference between GPOS and RTOS lies not in raw speed but in predictability [206, cap. 13]. An RTOS must therefore be deterministic, preemptive, and priority-based, ensuring that each task receives CPU control according to its temporal importance.

From an architectural perspective, a RTOS must facilitate the concurrent execution of multiple processes or threads, allocate priorities to each task, and ensure the preemption of lower-priority processes when a higher-priority task becomes ready for execution. Additionally, it offers synchronization functions to securely and predictably coordinate access to shared resources [145]. These characteristics enable the system to ensure both the logical correctness of operations and the timeliness of responses.

The kernel of a RTOS is the component that must ensure process concurrency, task prioritization, and interrupt management. The internal structure of the kernel directly influences its predictability and responsiveness. Laplante classifies kernel architectures into three main categories: monolithic, microkernel, and layered [128, cap. 3.2].

In *monolithic kernels*, all OS functionalities, *e.g.*, process management, I/O, interprocess communication (IPC), are integrated into a single block of code. This provides low context-switch latency but limits modularity and fault isolation [102].

Microkernels move most services into user space, keeping only essential functions such as scheduling, interrupt handling, and IPC inside the kernel. This approach enhances

modularity and security, albeit with a small performance overhead [102].

Layered kernels organize system modules hierarchically, reducing complexity and facilitating verification and maintenance [128, cap. 3].

Buttazzo describes the core components of a *real-time kernel*: the scheduling module (which determines the next task to run), the synchronization module (which manages semaphores and mutexes), and the interrupt handler (which ensures bounded reaction times to asynchronous events) [37, cap. 10].

Desai and Patoliya illustrate, through a FreeRTOS-based application, how a *multitasking kernel* can manage multiple sensors as independent tasks with distinct priorities and dynamic states: ready, running, blocked, suspended [69].

Predictability is the defining property that distinguishes real-time systems from conventional ones. It is the ability to determine a priori whether all deadlines will be met [37, cap. 1.3].

A RTOS must ensure bounded execution times for every system operation. Laplante notes that kernel functions such as task creation, queue management, and synchronization must have constant and analyzable time complexity [128, cap. 3.3].

Buttazzo formalizes two fundamental scheduling algorithms for real-time systems: Rate Monotonic Scheduling (RMS) assigns fixed priorities inversely proportional to task periods; Earliest Deadline First (EDF) dynamically assigns higher priority to tasks with the closest deadlines [37, cap. 4.3–4.4].

With the rise of the IoT, RTOS must now operate on devices with extremely constrained computational and energy resources. Shammar and Zahary survey lightweight IoT operating systems, TinyOS⁶, Contiki⁷, FreeRTOS⁸, and RIOT⁹, highlighting differences in architecture, multitasking models, and memory footprint [194]. TinyOS implements an event-driven, cooperative multitasking model optimized for wireless sensor networks. Contiki offers a modular kernel with optional preemption and a compact IPv6 networking stack. FreeRTOS provides a fully preemptive kernel with high portability and low footprint [69]. RIOT is based on a multithreaded microkernel design, combining flexibility and low energy consumption. Koulamas and Lazarescu describe these systems as the “neural infrastructure” of cyber-physical environments, integrating sensing, computing,

⁶TinyOS project: <https://github.com/tinyos/tinyos-main>

⁷Contiki Next Generation: <https://www.contiki-ng.org/>

⁸FreeRTOS home page: <https://www.freertos.org/>

⁹RIOT home page: <https://www.riot-os.org/>

and control under real-time constraints [124]. However, increasing complexity introduces new challenges, including energy efficiency, security, and support for virtualization.

In distributed embedded systems, ensuring end-to-end temporal guarantees requires tight integration between the RTOS scheduler and the networking subsystem. Behnke shows that packet-switched networks can introduce priority inversion and break determinism due to asynchronous interrupts [24, cap. 2.1–2.2]. To address this, he proposes hardware-assisted Network Interface Controllers (NICs) that can classify packets by task priority and moderate interrupts to maintain predictable communication latency [24, cap. 6.2]. Buttazzo had previously highlighted the importance of deterministic communication at the kernel level, proposing formal methods to analyze inter-task message latency [37, cap. 10.6]. Koulamas and Lazarescu note that Edge Computing and Time-Sensitive Networking (TSN) technologies are driving the evolution of RTOS toward deterministic networking, enabling concurrent execution of control and communication tasks within the same node [124].

2.2 Artificial Intelligence for managing vital parameters in embedded systems

The increasing availability of biosensors and wearable devices generates a large amount of heterogeneous data, whose interpretation requires AI methodologies capable of supporting predictive analysis, automatic classification, and decision-making processes.

Historically, AI has relied on rule-based systems, in which expert knowledge is explicitly encoded into logical rules and symbolic reasoning procedures [179]. Although these methodologies established robust foundational principles for early expert systems, they exhibited deficiencies in attaining scalability in dynamic environments, such as healthcare, where variability in patient signals and conditions is prevalent. In recent decades, the field has undergone a progressive shift towards data-driven methodologies, particularly ML and Deep Learning (DL). These approaches enable systems to automatically extract patterns and predictive models from raw data without the need for handcrafted rules [151, 30, 29]. This evolution is of particular relevance for monitoring vital parameters, where biosignals such as ECG, respiration, and blood oxygenation vary widely between individuals and contexts. ML methods have the ability to detect anomalies, model temporal dependencies, and provide predictive insights that are not possible with rule-based systems. These

techniques are already being incorporated into wearable health monitoring solutions, providing real-time feedback and diagnostic support [91, 193].

At the same time, the integration of AI into embedded architectures poses distinctive challenges. In contrast to cloud-based systems, which possess ample computational resources, embedded devices are constrained by limitations related to processing power, memory, and energy consumption. Consequently, there is an increasing demand for lightweight and efficient AI paradigms, such as TinyML and self-supervised learning. These paradigms aim to deliver accurate models that can be deployed on resource-constrained devices [199, 41]. In light of this background, the following sections will introduce the fundamental methodologies of ML. The sections will open with a discussion of conventional algorithms, including Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM). Subsequently, the section will progress to Artificial Neural Networks (ANNs) and DL models. These approaches represent the predictive core of AI for embedded monitoring systems. However, prediction alone is not sufficient: to ensure interpretability, interoperability, and knowledge sharing, the role of symbolic methods provided by *Knowledge Representation and Reasoning* becomes essential. For this reason, the chapter will conclude with a discussion on representational vocabularies and Description Logics, which provide the formal backbone for structuring and reasoning over the knowledge extracted from biosignals.

2.2.1 Machine Learning

ML is defined as the scientific discipline that involves designing algorithms that learn patterns from data and make predictions or decisions without explicit programming. According to Murphy [151], ML is a set of methods that *"automatically detect patterns in data and then use the uncovered patterns to predict future data or to make other kinds of decision under uncertainty"* [151]. Bishop [30] underscores that ML methodologies prioritize generalization, defined as the capacity of a model to accurately categorize or predict unseen instances beyond the confines of the training dataset. In a similar way, Rebala *et al.* in 2019 [176] define ML as *"a field of computer science that studies algorithms and techniques for automating solutions to complex problems that are hard to program using conventional methods"*.

ML paradigms are generally classified into the following categories: supervised, unsupervised, semi-supervised and Reinforcement Learning (RL) [30, 151, 175]. In the context

of supervised learning, models are trained on labeled examples to predict outcomes for unseen data. The objective of unsupervised learning is to uncover latent patterns or clusters in unlabeled data. Semi-supervised learning employs a combination of labeled and unlabeled data to enhance generalization [199]. RL involves optimizing actions through feedback from interactions with an environment. These paradigms are applied differently in vital parameters monitoring and management. Supervised methods dominate in diagnostic tasks, such as arrhythmia detection, while unsupervised learning supports clustering of patients by physiological profiles, and RL enables adaptive closed-loop systems, for instance, in personalized therapy. In biomedical contexts where labeled data are scarce, semi-supervised approaches are particularly relevant. Recently, self-supervised learning (SSL) has emerged as a compelling paradigm that utilizes raw signals to automatically generate supervisory signals. As a result, it reduces dependence on labeled data and enables efficient training for resource-constrained embedded devices [199].

ML plays a central role in embedded architectures for vital parameter monitoring. The physiological data continuously generated by biosensors and wearable devices are characterized by significant heterogeneity, noise, and individual variability, which makes traditional rule-based approaches ineffective. ML algorithms have the ability to filter noise, detect anomalies, and model temporal dependencies, enabling predictive healthcare applications and real-time decision support [199].

A growing body of research underscores the significance of ML in the realm of IoT-empowered healthcare, where data emanating from wearable devices undergoes processing either in a local capacity or within hybrid cloud-edge systems. The integration of ML models directly into devices can mitigate reliance on remote servers, thereby reducing latency and enhancing privacy. Such integration is imperative for tasks such as continuous ECG monitoring or personalized stress detection [170]. However, the computational and energy limitations inherent to embedded platforms require the implementation of efficient algorithms. This challenge has prompted the adoption of lightweight paradigms such as TinyML and SSL, which balance accuracy with resource constraints [199].

Traditional ML algorithms, such as Decision Trees (DTs), SVM, and ensemble methods, rely on handcrafted features that are engineered by experts. The advantages of these methods include transparency, robustness with smaller datasets, and relatively low computational requirements, making them suitable for embedded healthcare applications [30]. However, the efficacy of these systems strongly depends on the quality of feature extraction, a process that can be labor-intensive and domain-specific.

DL is an automated technique that learns hierarchical representations from raw data using multilayer Neural Networks (NNs), thereby bypassing the necessity of manual feature engineering [29]. Convolutional Neural Networks (CNNs) have been shown to be particularly effective for detecting spatial and temporal structures in physiological signals. Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) have been demonstrated to be particularly well-suited for sequential biosignals, such as ECG and respiratory signals. However, it should be noted that DL models require larger datasets and greater computational power. As Hsieh (2022) observes, DL demonstrates particular proficiency in the analysis of unstructured data, including images, audio, and text. In contrast, traditional ML methods often exhibit superior effectiveness and efficiency in processing structured tabular data, which is commonly observed in vital sign monitoring contexts [108].

2.2.1.1 Decision Trees

Decision trees allow making choices that depend on several variables and representing the outcome in a graphical and intuitive way, understandable even to an inexperienced human. DT consists of a tree-shaped graph in which the root node is the only node without incoming edges, the leaves are all nodes without outgoing edges, and the nodes with a single incoming edge and multiple outgoing edges are called internal nodes or test nodes. A DT takes as input an object or an instance, characterized by several attributes and provides as output a choice or a class. At root node and at each internal nodes correspond a test or question mark on an attribute, and from each node branch out as many edges as the admissible values of the tested attribute. At the end, each leaf represents the final decision for all instances that reach that node [123, 137]. DTs are among the most widely used ML methods adopted for classification and regression tasks in supervised learning. The concept was initially introduced by Belson in 1959 and then elaborated by Morgan and Sonquist in 1963 [183, 137]. In 1986, Quinlan proposed the ID3 algorithm, formalizing concepts of *entropy* and *information gain* as splitting criteria; the tree is built selecting for each node the attribute with maximum information gain, based on the entropy. Later, Quinlan extended this approach through C4.5, which introduced the *gain ratio* that allowed the handling of continuous attributes and missing data [171]. Breiman *et al.* developed the Classification and Regression Tree (CART) methodology, which introduced the *Gini index*, handled the continuous and discrete attributes, and extended the methodology to

regression analysis [34].

A DT is a data structure that recursively partitions the input space into subsets. At each node, a splitting criterion is applied to select the attribute that maximizes the homogeneity of the resulting subsets. The nodes that are located internally in the model represent the decision rules, whilst the branches correspond to the potential outcomes. Finally, the leaves assign class labels or regression values [123]. The evaluation of splits is conducted through the use of impurity measures:

- **Entropy (ID3, Information Theory):**

$$H(S) = - \sum_{i=1}^c p_i \log_2 p_i \quad (2.1)$$

Entropy defines the amount of disorder in a set S . In the formula, p_i is the probability that an element of S belongs to class i [171]. Among the classical criteria, entropy, derived from information theory, measures the uncertainty of a dataset and reaches its maximum when classes i are equally distributed. Entropy is zero when all instances belong to the same class (pure node) [151, 30].

- **Information Gain (IG):**

$$IG(S, A) = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v) \quad (2.2)$$

The information gain of an attribute A is calculated by dividing the set S into the subset S_v based on the attribute A . Information Gain is defined as the reduction in entropy after splitting on attribute and is used by ID3 and C4.5 to identify the most informative features [171]. Murphy in [151] showed that IG is equivalent of the mutual information between attribute A and the target Y , *i.e.*, $IG(S, A) = I(Y; A) = H(Y) - H(Y|A)$.

- **Gini Index [34, 60]:**

$$Gini(S) = 1 - \sum_{i=1}^c p_i^2 \quad (2.3)$$

The Gini index is a statistical measure of impurity. It calculates the probability that an example would be misclassified if it is randomly labeled according to the distribution of classes in the node. CART employs the Gini index, which measures the probability of

misclassification within a node. These impurity measures, although distinct, converge on the same objective: to maximize the separation of classes at each step of the tree-building process [151, 30]. Breiman *et al.* [34] advocated Gini for its computational simplicity compared to entropy. Daniya *et al.* (2020) [60] demonstrated with toy datasets that Gini frequently generates partitions analogous to entropy but at a reduced cost.

DTs offer several advantages in the context of managing incomplete information. In contrast to other classifiers that necessitate complete records, DTs have the capacity to manage missing data through surrogate splits or probabilistic routing, a process that allocates cases with missing values across multiple branches based on observed frequencies [66]. This feature is of particular importance in healthcare applications, where incomplete or noisy signals are prevalent. However, DTs exhibit a high propensity for overfitting when grown without constraints. In order to mitigate this issue, pruning techniques are applied. Pre-pruning interrupts tree growth prematurely by imposing constraints such as maximum depth or minimum node size. In contrast, post-pruning involves the removal of branches that do not contribute significantly to predictive performance after the tree has been fully constructed. Shamrat *et al.* [195] demonstrated that both of these strategies result in a reduction of model complexity and an enhancement of generalization, especially in noisy datasets, making it essential in embedded contexts with resource constraints [120].

Several variations of DTs have been introduced. The CHAID method utilizes chi-square tests for the analysis of categorical data. QUEST prioritizes the optimization of processing speed and the implementation of unbiased splitting. CTREE employs statistical inference procedures with the objective of mitigating the occurrence of overfitting. GUIDE enhances the statistical rigor in the process of feature selection. MARS integrates regression splines to model non-linear relationships [47]. These extensions exemplify the adaptability of DT methodology across domains and data types.

Recent research has explored hybrid approaches. Pagliarini *et al.* [157] proposed neural-symbolic temporal DTs for multivariate time-series classification, combining symbolic logic with neural representations. Wang *et al.* [218] introduced secure DT classification with decentralized authorization, a relevant approach for privacy-preserving embedded systems. Hou *et al.* [107] demonstrated the extraction of DTs as rule sets for clinical decision support, thereby bridging ML with KRR.

The advantages of DTs are well known: they are inherently interpretable, computationally efficient, and capable of handling both categorical and numerical attributes. Their disadvantages are equally evident: they are unstable, with small variations in the

training data often resulting in different structures, and they are frequently outperformed by ensemble approaches in terms of accuracy [151, 30, 47].

From an implementation perspective, in the context of embedded systems, the deployment of these components in the programming language C can be realized through the utilization of nested if-else statements. In contrast, in high-level languages, such as Python, these components are available through optimized libraries. In particular, scikit-learn provides implementations of CART with entropy and Gini impurity, and has been used in several studies on splitting criteria [1], CART testing with Gini index [60], and pruning methods [195].

Despite their lasting relevance, the limitations of DTs in terms of instability and overfitting paved the way for ensemble techniques. It has been demonstrated that methods such as bagging, boosting, and Random Forest take advantage of the simplicity of DTs while reducing their weaknesses by combining multiple trees to decrease variance and enhance predictive performance [146, 175, 66]. Ensembles outperform single trees in bioinformatics tasks [146]; their importance has been highlighted in modern ML pipelines [175]. Unfortunately, accuracy and generalization gains of these architectures w.r.t. DTs are associated with lower interpretability of models.

2.2.1.2 Random Forests

The RF method was introduced by Breiman in 2001. It represents one of the more widely spread techniques of supervised automated learning adopted for classification and regression tasks. The algorithm is the evolution of the bagging methods [33]. RF combine a group of DTs each trained on a subgroup randomly extracted from dataset and variables; the output is the result of the majority voting or the mean of the predictions.

RF are widely used because of their good accuracy, robustness, easy-to-use and good ability to handle complex data, sometimes without requiring a pre-elaboration of data themselves. Compared with a single DT, Random Forests (RFs) significantly reduce the model variance, resulting in a generalized output and reducing the overfitting [33, 26].

The construction of RF is based on two important rules [33]: *Bootstrap Aggregating (bagging)* where each tree is trained on a sample of the dataset obtained random extraction and replacement; and *Random Feature Selection* which introduces further diversity between trees, limiting the correlation between models and increasing the effectiveness of the ensemble. At each node in the forest, not all available variables are considered, but

rather only a randomly selected subset. The size of this subset is defined as $mtry$, and it is a key parameter of the model. Lower values increase the diversity between trees, reducing internal correlation and improving generalization ability. Conversely, higher values make individual trees stronger but can reduce the benefits of the ensemble. The final predictions are then aggregated by majority vote in classification problems or by averaging in regression problems, while the samples excluded from the bootstraps (out-of-bag) serve as internal validation [33].

Great performance in a high-dimensionality dataset with mixed variables [234], a good handling of missing and noisy data, great robustness to overfitting compared to a single DT and provision of variable importance metrics [33], are the most advantages that RFs offer. However, the literature also highlights some limitations. [234] highlights the reduced interpretability of over all trees compared to a single tree; the strongly dependency on the hyperparameters as numbers of trees, maximum depth and number of variables per split; and the phenomenon called diminishing returns, which underscores marginal benefits beyond a certain forest size. [26] also underscore the difficulty of handling high-cardinality categorical variables and highly unbalanced datasets and in a survey of 2018 [192] it is outlined the poor effectiveness in time-series predictions.

In conclusion, the primary advantages of RFs include their resilience to noise and outliers, their capacity to manage mixed variables and high-dimensional datasets, their high accuracy even with default settings, and the availability of diagnostic metrics such as OOB error and feature importance. The main limitations concern lower interpretability, theoretical difficulty in analyzing the method, sensitivity to key parameters such as $mtry$ and tree depth, and inefficiency resulting from unpredictable traversal on resource-limited devices [234, 26, 205]. The implementation aspects and embedded/IoT applications are discussed in Section 2.3.1.

2.2.1.3 Support Vector Machines

Support Vector Machines (SVMs) are predictive ML models based on data learning algorithms that analyze, train and search for patterns within data itself. These models were introduced in 1990s by Vapnik in his publication on the *statistical learning theory* [151]. The main use of SMVs lies in solving classification problems, *i.e.*, obtaining a model that returns a membership class as output. Given a set of data as input, defined as training set, with categorical variable characterized by classes $(-1,1)$, in its simplest form, the Support

Vector Machines (SVMs) learning algorithm returns as result the membership in one of the two classes. The model maps the observations within a multidimensional space, and search for the hyperplane or hyperplanes within it so that the two classes of observations are as far apart as possible. Thus, the whole idea behind the SVMs is to find and hyperplane that separates the data of two classes maximizing the margin, which is the distance between the decision boundary and the closest training point belonging to each class. A wider margin means a better generalization, reducing the risk of errors on not observed data [30]. The points closest to the hyperplanes are those used to identify the optimal separation hyperplane and are therefore defined as *support vectors*. In a bi-dimensional space, this boundary corresponds to a straight line; in three dimensions, it corresponds to a plane; in general, to a hyperplane. Only the subset of data corresponding to support vectors contributes to defining the separation boundary, giving the model a sparse structure that is efficient in the predictive phase [30, 151, 211].

Originally, SVMs assumed that data were perfectly separable. However, in real-world problems this is almost never the case, due to the presence of noise, outliers and overlap between classes. To make the model more flexible, Vapnik introduced Soft Margin SVMs. In this variant, slack variables $\xi_i \geq 0$ are introduced, allowing certain points to lie within the margin or on the wrong side of the hyperplane. The goal then becomes to maximize the margin while allowing a controlled number of errors. The optimization problem is:

$$\min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i \quad \text{subject to} \quad t_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0. \quad (2.4)$$

where:

- $w \in \mathbb{R}^d$ is the weight vector defining the hyperplane;
- $b \in \mathbb{R}$ is the bias (intercept) term;
- $\xi_i \geq 0$ are the slack variables, measuring the violation of the margin for each sample;
- $C > 0$ is the regularization parameter, controlling the trade-off between margin maximization and error penalization;
- $x_i \in \mathbb{R}^d$ is the feature vector of the i -th training example;
- $t_i \in \{-1, +1\}$ is the class label associated with the i -th sample;

- N is the total number of training examples.

C is the cost parameter that determines the importance of erroneous observations. It adjust the trade-off between maximizing margin and penalizing the classification errors: high values of C favor the correct classification of training samples, reducing the margin; small values allow for wider margins but tolerate errors on some points [30, 151].

Unlike more traditional regression models, the SVMs algorithm can also identify non-linear patterns using the *kernel trick*. The model uses a *kernel function*. When linear separation of classes is not possible, using kernel function, the data are transformed from its original space, called input space, and are mapped in a new multidimensional space, called feature space. To do so, instead of calculating the coordinates of the data in the transformed space, the inner product between two points in that space is calculated directly using the kernel function. In other words, the kernel function $K(x_i, x_j)$ provides a measure of similarity that is equivalent to the scalar product between the sample x_i and the sample x_j after an implicit mapping in a high-dimensional space, without the need to map it explicitly. The problem can be reduced to an optimization problem that follows the same approach as in the linear case [30]. The kernel functions, therefore, serve two main purposes: on the one hand, they allow problems not linearly separable in the original space to be addressed; on the other hand, they keep computational costs manageable, replacing the high-dimensional calculation with simple functions defined in the original space. Some examples include the linear kernel, suitable for problems that are already separable in the original space or when an interpretable solution is required. The polynomial kernel considers nonlinear interactions up to a certain order and the RBF (Radial Basis Function) kernel allows for the modeling boundaries, making the SVMs suitable in complex scenarios [151, 211].

The theoretical foundation of SVMs lies in the philosophy of *Structural Risk Minimization (SRM)* developed by Vapnik within statistical learning theory [30]. The idea is that the goal of a classifier is not only to reduce the error on the training set, as in *Empirical Risk Minimization (ERM)* adopted by other models, *e.g.*, NN trained by gradient descent, but also to ensure that the error on unobserved data, *i.e.*, the generalization error, remains low. According to SRM, generalization error can be expressed as the sum of two components: Empirical error, *i.e.*, the percentage of errors made in the training data; Model complexity term, which increases with the ability of the classifier to adapt to complex pattern and can be quantified using the VC-dimension (Vapnik-Chervonenkis dimension).

A model with too low a capacity (reduced VC dimension) risks underfitting the data, while a model with too high a capacity risks overfitting, adapting excessively to the noise in the dataset. SVMs address this trade-off by choosing the hyperplane that maximizes the margin: in this way, they reduce the effective VC-dimension of the classifier and thus the overall structural risk [224].

[17] shows this concept in the construction of a large margin allowing the model to keep the capacity low, ensuring robustness even when the data come from heterogeneous sources or contain noise. Maximizing the margin is a geometric choice and a direct mechanism for controlling the model complexity and improving generalization.

The SVMs in their original formulation were designed exclusively for binary classification problems. Over the years, several extensions have been adapted in order to expand their application. One application is the Support Vector Regression (SVR) that applies SVM to logic regression. The idea is approximating the continuous function, instead of dividing two classes. To do this, the function called ϵ -insensitive loss was introduced. The errors that lie within the ϵ interval around the predicted function are not penalized, while only deviations greater than this threshold are considered. In other words, the model searches for a function that passes “close” to all points, ignoring small discrepancies and focusing on the most significant deviations ([30], §7.1.4). This feature makes SVR robust in the presence of noisy data, and has been used, for example, in chemometrics for calibration problems [35].

Another extension concerns multi-class problems. Since standard SVMs have been defined for binary classification, in which there are only two labels, strategies must be adopted to handle more than two classes. The two most common are one-vs-rest, in which a classifier is trained to distinguish each class from all the others, and one-vs-one, in which a classifier is trained for each pair of classes. In the latter case, final decisions are made by combining the results of all classifiers. An application is that of DAG-SVMs (Directed Acyclic Graph SVMs), which organize classifiers into a hierarchical tree structure, reducing computational costs compared to the classic one-vs-one approach [35]. At last, there are One-Class SVMs (or Support Vector Data Description, SVDD), which are not used to distinguish between two or more classes, but to delimit the region occupied by data considered “normal”. In other words, the model constructs a boundary that encloses most of the samples in the training set, and any point outside this region is flagged as an anomaly or “novelty” [190]. This is the case when detecting incoming faults in industrial machines, where the available datasets mainly contain data related to the correct

functioning of the machines and only rarely examples of faults [224]. In these scenarios, One-Class SVMs construct a boundary around the distribution of normal data and classify observations that fall outside this boundary as “novelty” or “anomaly”. [224] point out that this approach is advantageous in condition monitoring for the detection of defects in critical industrial equipment. In this context, it is difficult to collect a balanced dataset of failures; therefore, the One-Class SVMs offer effective solutions because the model can be trained only with good data, maintaining good performance in recognizing deviations from normal patterns. Furthermore, [224] highlight that the ability of One-Class SVMs to handle noisy and non-linear data makes them suitable for real industrial signals, which often contain vibrations, interference, or variations due to the operating environment.

In the field of computer vision, SVM have been state-of-the-art for years in key tasks such as Optical Character Recognition (OCR) and facial recognition. Before the spread of DL, SVMs were the reference classifier for the handwritten MNIST data set, achieving accuracies of more than 98% when combined with RBF kernels and data normalization techniques. In the case of facial recognition, facial images were typically preprocessed with dimensionality reduction techniques such as PCA (Principal Component Analysis), obtaining so-called eigenfaces. SVMs were then trained on these reduced vectors to distinguish between different individuals ([151], §14.5). Representative SVM deployments in healthcare and industrial monitoring are summarized in Section 2.3.1.

Thanks to margin maximization theory and SRM, they guarantee robust generalization capabilities; they allow nonlinear problems to be addressed with kernels; they offer sparse models that are efficient in prediction and adapt to complex and noisy domains. However, they also have limitations: parameter and kernel selection requires careful tuning, scalability is limited on very large datasets, outputs are not probabilistic in nature, and interpretability is reduced, which hinders their use in sensitive sectors such as healthcare [35, 151, 211, 119].

2.2.1.4 Artificial Neural Networks

ANN are one of the most important models of ML. The first invention was in 1943 from McCulloch, who developed the idea of artificial neuron which was able to represent simple logical operations through a binary threshold neuron [36]. Subsequently, in 1949 Donald Hebb clarified that learning does not affect the functioning of individual neurons, but rather the strengthening or weakening of synaptic connections. This forms the basis of

ANN learning rules. Then in 1957 Rosenblatt developed the idea of Perceptron that was able to learn from labeled examples thanks to a rule for updating weights proportional to the error made [36]. However, in 1969 Minsky and Papert demonstrated the incapacity of perceptron to learn non-linearly separable functions. In the 1980s, thanks to the invention of *backpropagation algorithm*, it became possible to train multilayer network and to handle complex classification and prediction complex tasks [30, 151, 213]. With contributions from Hopfield and Kohonen, associative and self-organizing architectures were developed, while in the 2000s, the advent of GPUs and datasets such as ImageNet made Deep Neural Networks (DNNs) possible, revolutionizing domains such as computer vision, NLP, and biomedicine [36, 29].

The artificial neuron is modeled as a function that combines linearly the input x_i with a weight w_i and a bias b , and applies a non-linear *activation function* $f(\cdot)$:

$$y = f\left(\sum_i w_i x_i + b\right) \quad (2.5)$$

The Perceptron is the base model of a linear classifier in which the output is determined by the sign of the weighted sum of the inputs. The Perceptron Learning Rule is the learning algorithm that updates the weights based on the error made, ensuring convergence in a finite number of steps if the classes are linearly separable (Perceptron convergence Theorem). However, as said before, the Perceptron cannot solve non linearly separable problems, so this limitation led to the rise of multilayer networks [213]. The (2.5) allows non-linear functions to be approximated and, thanks to the universal approximation theory, networks with one or more hidden layers can represent any continuous function with an arbitrarily small error [106]. Typical architectures include *Feedforward network* and *Multilayer Perceptrons* (MLPs). The first consists of one input layer that receives the raw data; one or more hidden layers that transform progressively the data; and one output layer that gives the final prediction of the model. The information flow from input to output and the connections between neurons are weighted and represent the knowledge that the network acquires during the learning phase. MLPs extend the perceptron with hidden layers and nonlinear activation functions [30, 151]. Activation functions define the behavior of neurons and determine the ability of networks to learn complex representations. The step function is the first one used to model the original theory of perceptron. Then the hyperbolic tangent and the logistic sigmoid functions were used in order to model the

continuous output for a binary classification, but both suffer from the vanishing gradient problem: when values saturate, gradients tend toward zero, slowing down or blocking learning. To overcome this limitation, ReLU (Rectified Linear Unit) and its variants have become widespread, defining $f(x) = \max(0, x)$, allowing stable gradients to be maintained for positive values and simplifying the calculation [29, 213].

The learning process consists of the minimization of a cost function, *e.g.*, Mean Squared Error (MSE) for regression problems or cross-entropy for classification tasks, by adjusting weights using backpropagation and gradient-based optimization algorithms, such as Stochastic Gradient Descent (SGD) or Adam. The backpropagation algorithm is used for parameter optimization, and it combines the forward propagation with the backward propagation. First step is performing the forward propagation to compute predictions through all layers; then the final error is calculated; lastly, the error is backpropagated to estimate the contribution of each parameters. Specifically, the cost function gradient is calculated with respect to the weights, propagating the error from the output layer to the previous layers. This allows the weights to be updated iteratively using gradient descent methods [151, 29].

The ANN training depends on the choice of training parameters. The *learning rate* controls the step size taken during gradient descent: high values can lead to divergence or instability, instead low values slows down the learning and risk limiting the network in local minima [30, 151]. The *momentum* is used to add memory to updates and reduce oscillations [29]. The *number of hidden neurons* is another significant element that influences performance: too small networks fail to capture the complexity of the data causing underfitting, while too large networks can cause overfitting and lose generalization [213]. This is linked to the *dataset size*; if the number of samples is larger than model complexity, the risk of overfitting increases, while the underfitting is performed if the data are not enough.

To reduce overfitting and improve generalization, regularization techniques such as weight decay, early stopping, and dropout are employed [29]: the weight decay is used to penalize large weights using the weights square; early stopping interrupts training when performance on the validation set begins to worsen; the dropout deactivates randomly some neurons during the training phase, reducing the dependency on individual nodes.

Several applications show advantages in different scientific and industrial context of adopting ANNs. First of all, the capability of modeling non-linear complex relations that allow ANNs to be applicable to several application domains from biomedical signals to satel-

lite images. Also the ANNs remarked their versatility in processing heterogeneous data and adapting to very different platform. However, the ANNs also have significant disadvantages. The most obvious is higher computational cost that limited their use in resource-constrained devices. Other limitation is the need of larger datasets; without sufficient data the network risks to overfit or not learn adequately. The optimization of network topology and other hyperparameters are a complex process that influences performance and stability. Selected ANN use cases across biomedical, energy and IoT domains are consolidated in Section 2.3.1.

Another critical issue is interpretability. Except elementary topologies, the ANNs inner functioning is still described as a black-box and the nature of hidden layers. This limits adoption in fields where trustworthiness and accountability are essential, like clinical and financial decision-making contexts.

2.2.1.5 Deep Learning

DL is a paradigm of ML capable to automatically learn complex features starting from raw data. A deep network is composed of several neurons that apply linear combination of input followed by a non-linear activation function, *e.g.*, sigmoid, ReLU, tanh. The learning is based on minimization of a cost function using optimization algorithms that use the gradient descent methods, *i.e.*, SGD, momentum and Adam [29].

One critical aspect studied with DL is the *vanishing gradient* that occurs during training phase of deep networks. During backpropagation, the gradient tends approximately to very small numbers, especially when sigmoid functions or hyperbolic tangent are adopted as activation functions. This means that gradient of initial layers tends to zero and the weights are not significantly updated, so the learning ends. Otherwise, when *gradient explosion* occurs, the values of the gradient tend to grow exponentially, generating instability, and the optimization tends to diverge [29]. The literature offers several solutions, such as: the introduction of new activation functions like ReLU and its variant; the *residual connections*, introduced in ResNet and enable the direct link between input and output layers; the gradient clipping used for limiting that the values of gradient became too high; and the *batch normalization* (BN) used for stabilization of internal input reducing the *internal covariate shift* by normalizing the distribution of inputs in each layer [112, 31, 92]. BN accelerates convergence, allows the use of higher learning rates, and improves numerical stability [112]. It acts as a re-parameterization that smooths the loss function and

facilitates its optimization [92]. However, recent studies have shown some limitations: models with BN, while performing better under ideal conditions, are less robust to analog noise in edge/IoT scenarios. In [86] the authors have demonstrated that BN reduces noise-resistance, with worse performance than networks without BN when subjected to analog Gaussian noise.

Another critical aspect is *overfitting*. DL models learn the general structures of data, but also memorize noises and non significant information contained in the training sets. This phenomenon result in high accuracy on training data, but lower performance on validation and test set [176]. Several regularization techniques have been developed to address overfitting. One of them is the *dropout*, which consists of randomly deactivation of neurons during the training phase in order to reduce the dependency and improving the generalization [187].

Other strategies are the weight decay which penalizes too large weights; the early stopping that interrupts training when the performance on validation set stops improving; and the data augmentation that generates new data to increase the variability of the dataset. In specific fields such as healthcare, the risk of overfitting is particularly high due to data scarcity and imbalance. For example, models for chest image analysis have shown that, although deep networks can achieve performance comparable to that of radiologists, generalization to external datasets remains a significant challenge [13].

The principal architecture of DL are CNNs, which give DL the capacity of extract hierarchic features from images and video and recognize local pattern. Convolutional layers learn filter that firstly extract simple local patterns, *i.e.*, edges, then more complex structures, *i.e.*, parts of objects and whole shapes. The goal of CNNs, therefore, is to reduce the hand-crafted feature and provide abstraction in order to enable classification, detection and segmentation [176, 29].

RNN and its variant Long Short-Term Memory Networks (LSTMs) and Gated Recurrent Units (GRU) are used to handle sequential data, maintaining a memory of past states. The goal is to extract temporal dependencies in the data and used them for time series modeling [176].

Transformers, a more recent architecture, are exploited for modeling sequences in parallel, using the *self-attention* mechanism. The goal of Transformers is to overcome the parallelization and memory limitations of RNNs, making it possible to train much larger and more accurate models [29].

Generative models have opened up new frontiers in the creation of synthetic data.

Autoencoders and Variational Autoencoders (VAEs) aim to compress data into a meaningful latent space and then reconstruct it, facilitating applications in compression and anomaly detection. Generative Adversarial Networks (GANs) introduce a competitive approach between the generator and the discriminator to produce realistic images, while the Diffusion Models are applied in photorealistic image generation, medicine (*e.g.*, for the creation of synthetic medical data), and digital art [29]. DL applications in healthcare and perception for Cyber-Physical Systems (CPS) are presented in 2.3.1.

2.2.2 Incremental Machine Learning

Traditional DL models are trained in a *static learning paradigm*, where the network is exposed to a fixed dataset and optimized to minimize the loss function through multiple epochs. Once trained, these models usually exhibit limited flexibility when new data or tasks are introduced [161]. This limitation hinders their deployment in dynamic environments, where data distributions evolve over time or new classes appear. To overcome this rigidity, the research community has developed the paradigm of *IL* [169], also known as *Continual Learning* [130] or *Lifelong Learning* [161]. [219] have provided a theoretical and methodological view of this challenge, emphasizing that IL seeks to overcome the static nature of deep models by enabling continuous adaptation to evolving data streams. IL enables a model to continuously acquire new knowledge from incoming data while retaining previously learned information [161]. This process requires maintaining a balance between *plasticity*, the ability to learn new tasks, and *stability*, the ability to preserve prior knowledge, commonly known as the *stability–plasticity dilemma* [161]. A challenge in IL is *catastrophic forgetting*, a phenomenon where the model rapidly loses previously acquired knowledge when updated with new data [161], as occurring to NNs that are trained sequentially on multiple tasks [89].

In order to define *what to learn*, [212] proposed a taxonomy to distinguish three main types of IL based on the relationship between tasks, domain, and classes. The three types of IL vary in terms of the amount of information on the task that is accessible during testing, as well as the complexity of the learning problem:

- Task-Incremental Learning (Task-IL): The model is trained on a sequence of distinct tasks, and the identity of the current task is known at test time. A common approach is to use separate output heads for each task, so the model classify among

classes belonging to current task. The main challenge lies in sharing representations between tasks while maintaining low computational complexity.

Awareness of the task ID during the test time mitigates complexity, as the model is not required to deduce which context to utilize.

- **Domain-Incremental Learning (Domain-IL):** The model faces the same task, but the data distribution changes across domains, *e.g.*, the variations in lighting, imaging sensors, or environmental conditions. The identity of the task is unknown during testing, which makes it necessary for the network to adapt to domain changes while retaining generalization capabilities. Methods for handling and mitigating concept drift or domain shift are required.

The difficulty increases as the model must recognize the data without any prior knowledge of its domain.

- **Class-Incremental Learning (Class-IL):** The most difficult setting, where new classes appear over time, and the model must recognize both old and new categories. The model is unaware of the current task and must learn to discriminate all classes seen so far. This setup is particularly sensitive to catastrophic forgetting.

The model must be able to differentiate between classes that have never been observed together, which poses the greatest risk of forgetting. This situation is considered the only "complete" scenario, as it simulates the conditions of lifelong learning.

From the literature methods can be classified based on *how* IL is accomplished. [130] identified three main methodological categories that consider the most state-of-the-art solutions: Forgetting-Prevention Methods; Network-Correction Methods; Externalized-Structure Methods. Similarly, [219] proposed a taxonomy of continual learning methods that expands these categories into five complementary paradigms: regularization-based, replay-based, optimization-based, representation-based, and architecture-based approaches. This framework emphasizes that the most effective continual learning strategies integrate parameter regularization, rehearsal mechanisms, and architectural adaptation to preserve previously acquired knowledge across tasks.

The *Forgetting-Prevention methods* are the most used to prevent catastrophic forgetting in Class-IL, identifying the causes of forgetting and addressing the problem through other methods: regularization, rehearsal and knowledge distillation.

- *Regularization* is the most used to mitigate forgetting, also classified as weight regularization and data regularization approaches. When a NN is used to train a new task, the optimization updates all weights in order to adapt to new data. This is called catastrophic forgetting: the weights, which were optimum for previously tasks, are changed, and the network has forgotten the previous knowledge. Regularization introduces an additional penalty term in the loss function that discourages large changes in weights that were important for previous tasks.
- *Rehearsal* is a method used to prevent forgetting and maintain a memory of older data. During the learning of a new task, the network is exposed to older data to consolidate the previous knowledge during the acquisition of new knowledge.
- *Knowledge Distillation* typically used for class-IL, is a method of forgetting prevention that helps the model avoid misclassification between new and old data, maintaining discrimination and reducing forgetting of acquired knowledge.

IL has been successfully applied across a wide range of domains. [68] proposed the *Life-Longer* benchmark for continual disease classification, using the MedMNIST dataset under task-IL, class-IL, and cross-domain IL settings. The results showed that rehearsal-based methods such as iCaRL outperform regularization in complex medical environments. Similarly, [220] proposed an *incremental multi-task shared classifier* for the diagnosis of fault of bearing, combining shared feature extractors with knowledge distillation to adapt to changing operating conditions while preserving learned diagnostic features. In another study, [215] developed an *incremental cascaded model* for tuberculosis detection and localization of chest X-rays, where a three-stage DL pipeline continuously refines its predictions as new data become available.

IL thus represents an evolution of traditional DL, enabling continuous adaptation and knowledge retention through mechanisms of parameter regularization, architectural expansion, and external memory integration. These strategies collectively address the stability–plasticity dilemma and form the foundation for modern lifelong learning systems. [219] consolidates these directions, demonstrating how regularization, replay, optimization, and architectural adaptation together form the core of Continual Learning research.

2.2.3 Federated Machine Learning

Federated Learning (FL) is a distributed learning paradigm created as a solution to the growing need for protection of privacy, reduction of latency, and enhancement of distributed data. Unlike the traditional approach, which involves collecting and analyzing data in a single computing node, FL allows a model to be trained while keeping the data on local devices and sharing only model updates with a central coordinator. This approach, initially introduced by Google with the Federated Averaging (FedAvg) algorithm [144], reduces the risk of exposure to sensitive data and enables the integration of information from heterogeneous and distributed sources. In the FedAvg scheme, local clients perform several training epochs on their own data and periodically send the updated model weights to a central server. The server then aggregates them and calculates the weighted average based on the size of the local datasets.

FL represents a shift from the traditional paradigm of centralized data collection to a decentralized intelligence model based on the idea of bringing the code to the data rather than data to the code. In other words, analytical and learning processes are allowed to occur directly on the devices where data is generated, *e.g.*, smartphones, IoT nodes, or sensors, thus improving latency, efficiency, and compliance with privacy regulations. In this view, FL enables large-scale collaborative learning among distributed entities without violating data locality.

Three modes of FL can be distinguished [231]. In *horizontal FL*, the data share the same feature space but belong to different users, as in the case of millions of smartphones that generate homogeneous but independent interactions that contribute to the improvement of a text prediction model. In *vertical FL*, on the other hand, multiple organizations possess data on the same individuals but with different characteristics, as is the case, for example, between a bank and an e-commerce platform. Finally, *federated transfer learning* applies to contexts where datasets differ in both samples and features, thus requiring the integration of transfer techniques to generalize models across different domains. Beyond these categories, FL is based on three fundamental principles: *data decentralization* [231] which means that each node will preserve local data across multiple devices; *collaboration between multiple autonomous entities* [231] establishing a secure data and knowledge sharing between nodes and data owners; and *privacy preservation* [231] limiting the leakage of sensitive information during the training. According to [116], preservation of privacy in FL relies on a combination of cryptographic and statistical techniques such as

differential privacy, homomorphic encryption, and multiparty secure computation, which limit the possibility of inferring sensitive information from shared parameters.

Privacy preservation is a structural features of FL. Since raw data are implemented in local devices, the model aligns with modern data protection frameworks, reducing the risks associated with data breaches or unauthorized access. FL, also, reduces communication overhead by transmitting only model updates instead of full datasets, achieving bandwidth efficiency while maintaining strong security guarantees.

In general, FL architecture is based on a client–server model. Local devices train the model with their own data and periodically send parameter updates to the server, which are aggregated to form a global model. Although this is a widely used paradigm, based on FedAVg due to its conceptual simplicity, practical implementation requires cryptographic protocols to protect the exchange of information between devices. Several techniques have been adopted to strengthen information protection without excessively compromising model accuracy [231].

The overall FL workflow can be summarized in six main steps:

1. local data collection,
2. on-device preprocessing,
3. local model training,
4. secure parameter transmission,
5. server-side aggregation (typically through FedAvg) and
6. global model redistribution to clients.

This cycle repeats until convergence is reached. Such architecture enables asynchronous participation of clients and is adaptable to both cross-device and cross-silo configurations. FL still presents numerous challenges. One of the main ones concerns the non-independent and non-identically distributed (non-IID) nature of local data, which can compromise the convergence and quality of the global model. Furthermore, the devices involved are often heterogeneous in terms of computational capacity, memory, and energy, and are not always available synchronously. Another critical issue concerns security: the federated model may be vulnerable to attacks that aim to insert a backdoor or degrade its performance,

as well as to inference techniques that seek to extract sensitive information from shared gradients [116].

FL has been widely adopted in domains such as healthcare, mobile systems, and industrial IoT, demonstrating its use for privacy-preserving collaboration across distributed environments

2.2.4 Knowledge Representation and Reasoning

KRR is an important branch of AI, dealing with modeling knowledge related to a domain and using these models to enable automatic systems to perform inference *i.e.*, derive new knowledge from that already defined [201, 77]. The core idea is that an intelligent agent should not only memorize data, but also organize them in semantic (*i.e.*, meaningful) structures to answer questions, solve problems, and make decisions. A KRR formalism must handle several requirements [77]:

- *Expressiveness*: the ability to model complex concepts and constraints;
- *Correctness and Completeness*: the ability to guarantee that the inferences produced are logically valid and that all possible consequences can be derived;
- *Computational efficiency*: the ability to perform reasoning in times compatible with real application constraints.

As Donini *et al.* highlighted [78], '*the greater the expressiveness of a language, the higher the computational complexity of reasoning services tends to be*', meaning that there is an inherent trade-off between the expressive power of a representation language and the difficulty of reasoning about it. This trade-off represents one of the central challenges in the development of KRR systems and has motivated the investigation of logical formalisms and the complexity of reasoning problems [18].

A *Knowledge Representation System (KRS)* is a software system designed to model, manage, and use knowledge related to an application domain. Unlike conceptual models used in software engineering or databases, such as E-R diagrams or UML models, which are design support tools but not an integral part of the final product, in KRSs conceptual models are part of the system itself and are used at run-time [77]. A KRS typically consists of five main elements:

1. A terminological component ($\mathcal{T}$), known as *ontology*, which describes the general concepts of the domain and their hierarchical relationships;
2. An assertion component ($\mathcal{A}$), which represents the concrete objects of the domain and their characteristics;
3. A set of reasoning services, which allow new information to be deduced from that already defined in $\mathcal{T}$ and $\mathcal{A}$;
4. An access interface, which allows external software systems to query and use the available knowledge contained in $\mathcal{T}$ and $\mathcal{A}$;
5. An editing interface, which allows human users to manage, update, and maintain information contained in $\mathcal{T}$ and $\mathcal{A}$.

2.2.4.1 Ontology

The term *conceptualization* refers to an abstraction of relevant phenomena in a domain, isolating the essential aspects and omitting the irrelevant ones. In this context, *ontologies* are the specification of this process: they are structured and shared models that explain the key concepts of a domain, the relationships that link these concepts, and the constraints that define the use of these concepts [95]. As highlighted by Gruber [95], a Knowledge Representation (KR) is an *explicit specification of a conceptualization*, in other words, it is a formal model that describes a set of entities, properties, and relationships considered relevant in a domain. This model is not only descriptive, but also has computational significance, since it must support automatic reasoning processes. Guarino [96] expands this definition by introducing the concept of shared conceptualization, emphasizing how ontologies have the primary purpose of enabling knowledge sharing and interoperability between heterogeneous systems. In this respect, ontologies provide a common vocabulary and a unified formalism with which to describe and share knowledge between heterogeneous systems [77] due to making knowledge machine-readable and supporting semantic communication between various applications [9].

An ontology typically consists of fundamental components that define its conceptual structure and functioning: *concepts*, representing sets of objects of the domain; *individuals*, instances of concepts, *i.e.*, actual objects in the domain; *roles*, defining relationships between pairs of individuals.

As said, a KRS is composed of a $\mathcal{T}$ and a $\mathcal{A}$. The $\mathcal{T}$ essentially contains terminological knowledge, *i.e.*, the definitions of concepts and relationships in the domain and defines a terminology or *ontology*. The $\mathcal{A}$, on the other hand, contains factual knowledge, *i.e.*, assertions about concrete individuals. Together, $\mathcal{T}$ and $\mathcal{A}$ form a Knowledge Base (KB), which can be used by reasoning systems to deduce new knowledge.

2.2.4.2 Description Logics

Description Logics (DLs) are a family of logical languages designed to provide well-defined semantics to concepts, roles, and individuals [18, 78]. The basic elements of Description Logics (DLs) are:

- Concepts (C, D), which represent sets of individuals;
- Roles (R), which represent binary relations between individuals;
- Individuals (a, b), which denote concrete objects of the domain.

These elements can be combined using logical constructors. Each DLs language is characterized by the set of constructors it admits, which determine the level of expressiveness and complexity of reasoning.

Attributive Language with unqualified Number restrictions ($\mathcal{ALN}$) [18] is one of the formalisms that can be used to model ontologies. $\mathcal{ALN}$ allows a few fundamental constructs for modeling complex concepts. each with its semantic meaning, but excludes more powerful operators that would make reasoning less efficient. The definition of the concept used the following constructs:

- The *conjunction* of concepts ($C \sqcap D$) allows us to describe the intersection of two sets. For example, the expression *Student* $\sqcap$ *Worker* represents all individuals who are both students and workers.
- *Atomic negation* ($\neg A$) allows us to express the set of individuals who do not belong to a primitive concept. For example, $\neg$ *Graduate* indicates all people who are not graduates. In $\mathcal{ALN}$, however, negation is limited to atomic concepts and is not allowed on complex expressions.
- The *universal restriction* ($\forall R.C$) specifies that all values of a certain role must belong to a concept. For example, $\forall$ *hasChild.Student* identifies individuals such

that, if they have children, all of them must be students. It is important to note that the universal restriction does not necessarily imply the existence of children, but establishes a membership constraint if they exist.

- *Unqualified numerical restrictions* ($\leq n R$) allow cardinality constraints to be expressed on the number of relationships allowed for a role. For example, $\geq 2 \text{ hasSon}$ indicates the set of individuals who have at least two children, while $\leq 1 \text{ hasCar}$ represents those who own at most one car.
- *Inclusions* ($A \sqsubseteq C$) allow conceptual hierarchies to be defined. For example, $\text{Professor} \sqsubseteq \text{Lecturer}$ establishes that every professor is also a lecturer. Inclusions form the basis of an ontology's conceptual taxonomy.
- *Equivalences* ($A \equiv C$) allow complex concepts or synonyms to be formally defined. For example, $\text{Single} \equiv \text{Man} \sqcap \neg \text{Married}$ states that the concept of "Single" is semantically equivalent to being a man who is not married.

2.2.4.3 Web Ontology Language and Semantic Web

Resource Description Framework (RDF) is the standard W3C used for KR on Web according the triple subject - predicate - object [191]. RDF enable modeling data as graphs, namely Knowledge Graphs (KG), because when more than one triple refers to the same thing, the triples can be viewed as a directed graph in which each triple is an edge from its subject to its object, with the predicate as the label on the edge [9].

Web Ontology Language (OWL), the W3C standard for defining ontologies on the Web [216], is based on RDF. OWL allows one to specify classes, properties, and individuals, and offers advanced logical constructs such as equivalences, disjunctions, and cardinality restrictions. OWL 2 introduces three main profiles:

- OWL 2 EL, suitable for large conceptual hierarchies and polynomial reasoning. It is optimized for ontologies with many properties or classes;
- OWL 2 QL, designed for integration with relational databases;
- OWL 2 RL, useful where reasoning can be implemented in rule-based systems and efficient on very large datasets.

Table 2.1: Summary of $\mathcal{ALN}$ constructs

Name	DLs syntax	OWL RDF/XML element
Top	$\top$	<code><owl:Thing></code>
Bottom	$\perp$	<code><owl:Nothing></code>
Concept	C	<code><owl:Class></code>
Role	R	<code><owl:ObjectProperty></code>
Conjunction	$C \sqcap D$	<code><owl:intersectionOf></code>
Atomic negation	$\neg A$	<code><owl:disjointWith></code>
Unqualified existential restriction	$\exists R$	<code><owl:someValuesFrom></code>
Universal restriction	$\forall R.C$	<code><owl:allValuesFrom></code>
Unqualified number restrictions	$\geq n R$	<code><owl:minCardinality></code>
	$\leq n R$	<code><owl:maxCardinality></code>
Definition	$A \equiv C$	<code><owl:equivalentClass></code>
Inclusion	$A \sqsubseteq C$	<code><owl:subClassOf></code>

Table 2.1 summarizes constructors in *Attributive Language with unqualified Number restrictions* ($\mathcal{ALN}$), which is the main reference DLs, and corresponding elements in OWL RDF/XML syntax [216].

2.2.4.4 Reasoning

The *Reasoning* is the process that allows to move from explicit to implicit knowledge, and it is what separates a KRS from a simple dataset. As reported in [201], the reasoning can take several three forms:

1. *Deductive* which allows logically necessary conclusions to be derived from general premises;
2. *Inductive* which generalizes rules from partial observations;
3. *Abductive* which formulates plausible hypotheses to explain incomplete or unexpected observations.

In KRR systems, standard reasoning is mainly deductive and is accomplished through a series of tasks [18, 78]:

- *Subsumption* that checks whether one concept is a specialization of another ($\mathcal{T} \models A \sqsubseteq B$, meaning that in the ontology $\mathcal{T}$ every instance of A is also an instance of B);

- *Satisfiability*: checks whether a concept can have at least one possible instance, *i.e.*, is not self-contradicting ($\mathcal{T} \models A \not\sqsubseteq \perp$, meaning that the concept A is not a subclass of the empty concept $\perp$ a.k.a. Bottom in the ontology $\mathcal{T}$, and therefore can have at least one possible instance);
- *Coherency*, ensuring all named concepts in an ontology are satisfiable;
- *Consistency*: checks for the absence of contradictions within a KB, meaning that one class has an instance to prevent contradictions;
- *Classification*: automatically constructs the taxonomy of concepts induced subsumption relations.

These tasks require optimization techniques to efficiently handle large ontologies.

Reasoning tasks, often implemented via DLs and KR frameworks, enable inferring new knowledge from explicitly defined facts and relationships. Automated reasoning algorithms fall into three main categories: *structural*, *tableaux-based* and *rule-based*. (i) *structural algorithms* transform logical expressions into simpler forms for efficient processing, typically in polynomial time, though they are incomplete for highly expressive DLs; (ii) *tableaux-based algorithms* construct a tree-like *tableau* to check ontology consistency and concept satisfiability, supporting more expressive DLs but with exponential time complexity. (iii) *rule-based algorithms* use predefined rules applied iteratively to inferred facts, operating efficiently in polynomial time but lacking support for non-monotonic reasoning [71].

This dissertation leverages the *Tiny-ME* [181] Semantic Web of Everything (SWoE)-oriented matchmaking and reasoning engine, which implements both standard and non-standard reasoning through structural algorithms working in polynomial time for $\mathcal{ALN}$. Given an ontology $\mathcal{T}$ in $\mathcal{ALN}$, and two satisfiable concepts R and S representing a request and a resource, respectively. In order to perform *semantic matchmaking* of S w.r.t. R initially a *Consistency Check* is executed to determine whether the conjunction $R \sqcap S$ is satisfiable w.r.t. $\mathcal{T}$. If this check does not succeed, the resource is a *partial match* for the request, and *Concept Contraction* is performed. The result of Contraction is a pair of concepts $\langle G, K \rangle$ such that $\mathcal{T} \models R \equiv G \sqcap K$, *i.e.*, the original request R is split into two parts: G (*Give Up*) will be discarded due to inconsistency with the resource S , while K (*Keep*) will be kept since $\mathcal{T} \models K \sqcap S \not\sqsubseteq \perp$, *i.e.*, K is consistent with S , as their conjunction is satisfiable. Basically, Contraction identifies the portion of R that clashes

with S . By discarding only the clashing requirements G from R , the remaining concept K represents a revised version of the initial request. G acts as an explanation of why $R \sqcap S$ is unsatisfiable, thus enabling a transition from a partial match to a *potential match*.

Concept Abduction formulates hypotheses about what is missing in a resource to satisfy a request. it is computed in case of potential match, which occurs if S does not clash with R but is not subsumed by it. The result of Abduction is finding a concept H (*Hypothesis*) such that $\mathcal{T} \models S \sqcap H \sqsubseteq R$, *i.e.*, by conjoining H with the resource S , the combined concept is more specific than the request R , meaning it fulfills all the requirements in R and $S \sqcap H$ is satisfiable in $\mathcal{T}$. It is important to highlight that Concept Abduction (CA) along with other reasoning services for semantic matchmaking in Tiny-ME are based on the Open World Assumption (OWA), meaning that missing information in a concept expression does not imply negation, but represent an unspecified constraint, possibly due to being unknown or deemed negligible. Therefore H represents the part of R that is required and not found in S , and that should be hypothesized in the OWA to make the Subsumption relation true: this provides a formal explanation for a missed Subsumption and a way to transition from a potential to a *full match*.

Concept Bonus identifies additional attributes present in a resource that are not explicitly requested. Concept Bonus is useful in matchmaking scenarios, as a resource S might include attributes not specified in R either because the requester was unaware or did not care, which can be exploited in a query refinement process. This reasoning task identifies a *Bonus* concept B from S , indicating what the resource offers even if it was not requested. Determining the Bonus B of S w.r.t. R is identical to the Concept Abduction problem where R and S roles are reversed, *i.e.*, R is considered as a resource and S as a request.

2.2.5 Optimizations for embedded systems

To deploy ML algorithms on MCUs, the model must be compressed to a size that fits the chosen MCU's memory limitation, processing power and energy [74]. The techniques used to enable model deployment are pruning, quantization, knowledge distillation and Hyper-Parameter Optimization (HPO).

Pruning process reduces the number of model parameters by eliminating redundant weights or neurons within a model architecture without affecting the overall model performance, but enabling faster inference [74]. Pruning can be applied during the training

process to mitigate the risk of overfitting, or after the model has been trained to eliminate redundant connections from model making it more efficient and faster [41].

Quantization compresses models by converting 32-bit floating point values of weights and activations functions to 8-bit or lower fixed-point formats, reducing model size and computational costs maintaining inference accuracy [74]. Quantization can be applied after training without retraining the model for optimum performance or can involve re-training of the model. *QAT (Quantization-Aware Training)* consists of two processes: pre-training and fine-tuning. The pre-training process learns data patterns and develops feature representations using full-precision floating point format (32-bit). The conversion of the network to a quantized representation, preserving the accuracy, is executed in fine-tuning stage. *The Post-Training Quantization (PTQ)* reduces the memory usage and computational costs by converting the model weights and activations from high-precision floating point, representation of model during training process, to lower-precision numbers using k-means clustering or vector quantization techniques [41].

Quantization and pruning can provide an MCU-compatible model that reduces both power consumption and latency by reducing memory consumption, but at some cost in model accuracy. For the embedded ML model, the challenge is to find an acceptable trade-off between performance and power/latency. Deploying ML algorithms on an MCU requires a degree of embedding optimization in addition to quantization and pruning. Several software tools, libraries and frameworks are used for this purpose, some of them including quantizing in their optimization process [74].

Knowledge distillation allows compact and simpler models, also called students, to inherit the performance characteristics of complex and large models, also called teachers, by transferring knowledge. In general, Knowledge Distillation is used to achieve a good trade-off between computational reduction and model performance and for this reason it is adopted for models that cannot be deployed “as they are” to fit the hardware requirements. There are three categories response-based, feature-based and relation-based knowledge extraction from teachers and its use in training student networks. Specifically, by capturing the neural response in the last output layer, *response-based knowledge distillation* mimics the final predictions of the teacher model. *Feature-based knowledge* extends this approach by using both the output of the last layer and intermediate layers. In addition to the outputs of specific layers in the teacher model, *relation-based knowledge* explores the relationships between different layers or data samples [41].

The distillation schemes depend on the training strategy for student learning process, and

also here, there are three categories: offline distillation, online distillation, self-distillation. *Offline distillation* involves a two-step strategy, first training the teacher model on a set of training samples and then using the trained teacher model to guide the student model through the extraction of intermediate features or logits. On the other hand, *online distillation* is suitable when the teacher model is not significantly larger or more powerful, as it is an end-to-end approach where both the teacher and student models are updated simultaneously. Finally, *self-distillation* is a special case of online distillation in which the network of the teacher and the network of the students have the same architecture [41].

Hyper-parameter optimization (HPO) automates the research for the optimal hyper-parameter values of a model, enhancing its performance and efficiency. In embedded ML, this process is critical because the optimal configuration must account for both model accuracy and hardware constraints such as memory, latency, and energy consumption [74, 41]. Algorithms used for optimizing hyper-parameters are Grid Search, Random Search and Bayesian Optimization [88].

Grid Search involves the definition of a finite set of possible values for each hyperparameter and the subsequent systematic evaluation of the model for all possible combinations. This search guarantees that the global optimum within the grid will be found; however, as the number of hyperparameters or the resolution of the grid increases, the search becomes computationally prohibitive. In practice, grid search is only effective for a small number of hyperparameters or when each parameter has a limited range of discrete values. Despite its simplicity, it provides a baseline for assessing the influence of individual hyperparameters on model performance.

Random Search represents a simpler and efficient alternative to grid search. Instead of evaluating all possible combinations, it samples random configurations from the hyperparameter space according to predefined probability distributions. Random Search tends to outperform grid search when only a subset of hyperparameters strongly influences the model's performance, because it explores a wider variety of values rather than wasting evaluations on unimportant parameters. Additionally, it facilitates enhanced scalability, simplified parallelization, and more adaptable time allocation, as the number of trials can be adjusted to suit the available computational resources. *Grid Search* and *Random Search* provide simple but computationally expensive exploration strategies, unsuitable for large design spaces.

Bayesian Optimization exploits probabilistic models to iteratively refine the search around promising configurations, thus reducing computational cost while maintaining accuracy.

The objective function, which may include metrics such as validation accuracy or loss, is modeled as a black-box function that is costly to evaluate. A probabilistic surrogate model is trained on the results of previously tested hyperparameter configurations. The purpose of this training is to approximate the function. Subsequently, an acquisition function utilizes the uncertainty of the surrogate model to select the next hyperparameter configuration that is most promising. This process balances exploration, defined as *searching unexplored areas*, and exploitation, defined as *refining known good regions*. This iterative process is repeated until convergence or until a computational budget is reached. In comparison with random or grid search, Bayesian optimization generally attains superior outcomes with a significantly reduced number of evaluations. This characteristic renders it especially well-suited for scenarios where model training entails substantial computational expense. Modern HPO approaches are evolving toward *multi-objective* and *hardware-aware* optimization, where algorithmic parameters are tuned with hardware-related configurations to achieve efficient and reliable on-device learning. When combined with model compression methods such as pruning, quantization, and knowledge distillation, hyperparameter optimization completes the toolkit of strategies that enable the deployment of intelligent models to resource-constrained environments [88, 41, 74].

2.3 Open issues

AI is increasingly being applied in various sectors of society, including healthcare, industrial control systems, and public transportation. Application like anomaly detection and system monitoring utilize ML models and techniques to deliver adaptable and precise solutions. AI is raising serious ethical and safety concerns with regard to its extensive use. Software-driven decision-making must be supported by a reliable and certifiable process. Despite significant progress in ML techniques and optimizations for embedded systems, several open challenges remain, particularly concerning the integration of diverse components and the assurance of model reliability and robustness.

2.3.1 Integration on embedded systems

Designing embedded intelligent systems requires an approach that integrates sensing, computation, and modeling. Thus, ML for embedded applications can be viewed as a process of hardware-software system-level co-design, where sensors, processor and learning

models are jointly optimized. The characteristics of the sensing stack, such as bandwidth, sampling rate, signal resolution, and noise level, directly impact the stability of the features and the quality of the extracted features from signals [41]. Conversely, the computational substrate, such as microcontrollers (MCUs), low-power SoCs, or RISC-V architectures, imposes strict constraints on memory footprint, throughput, and energy consumption. Therefore, the choice of model complexity, numerical precision, and inference strategy must be adjusted according to these limitations to ensure real-time operation within the available resource envelope. The model design must be adapted to the processing and memory constraints of the target microcontroller or SoC, integrating the data acquisition, pre-processing, and inference processes into a single design scheme [41].

The TinyML paradigm is defined in [41] and [199] as a pipeline that integrates data acquisition, feature extraction, inference, and communication into a single low-power architecture: in each level, sensors, microcontroller and model must be co-designed to minimize resource usage. This is validated by several use cases, *e.g.*, motor activity recognition and ECG monitoring, where local inference replaces continuous data transmission to the cloud. Furthermore, the stability of the extracted features is related to sensor performance; [91] shows that the variation in the sampling frequency and quantization noise introduced distortion in the biosignal features, compromising the accuracy of final classification. Also, hardware introduces variability in memory throughput, and sampling limits the window size of the process, requiring adaptive representations for on-device learning [199].

To handle these constraints, embedded pipelines typically used preprocessing strategies, such as normalization, to minimize memory usage and latency during inference. These concepts are behind frameworks like TensorFlow Lite Micro and are used on tree and ensemble models. [205] proposes an implementation of RFs for RISC-V architectures, using data structures that improve cache locally and node traversal speed; [59] introduces an energy-efficient approach based on early stopping and incremental evaluation of DTs. In [23] complex data structures are used for more efficient representation of large ensembles on microcontrollers. Overall, these contributions demonstrate that embedded ML requires end-to-end integration between sensory acquisition, signal processing, and model execution, where algorithmic design is intrinsically driven by hardware constraints and sensor characteristics.

The research line of TinyML/TinyDL focuses on the integration of deep learning models into MCU- and SoC-based systems. It is essential that both the model and the infer-

ence pipeline are co-designed with the target hardware and software stack during this process. Overviews of this embedded integration [173, 41] systematize the TinyML stack and highlight that deployability is contingent on the following factors: (i) model architectures (including lightweight CNN); (ii) compression and optimization techniques (*e.g.*, quantization, pruning, knowledge distillation); (iii) automated design methods such as hardware-aware Neural Architecture Search (NAS); and (iv) runtime/toolchain support, from conversion pipelines to device-level execution and operators. Consequently, integration on constrained devices should be regarded as a multi-objective problem, necessitating explicit trade-offs between predictive performance and system-level constraints (flash/RAM usage, execution time, and power consumption) [41]. In addition to the selection of architecture, the feasibility of MCUs frequently depends on the concurrent optimization of three factors: (i) the network structure, (ii) numerical precision, and (iii) the execution pipeline. This is due to the fact that SRAM/Flash limitations and operator-level memory reuse can exert a significant influence on the design space [173].

In this perspective, TinyDL is the deployment of deep neural networks (including emerging transformer-based variants) on MCU-class devices under stringent power and memory constraints [200]. TinyDL is starting to become practical due to several concurrent advancements in the development of lightweight architectural building blocks (*e.g.*, depthwise separable convolutions and inverted residuals), post-training quantization and quantization-aware training, structured pruning and distillation, and hardware-aware automated design methods (*e.g.*, NAS) that explicitly prioritize performance over deployment constraints. TinyDL deployment depends not only on model compression, but also on compilation/runtime support (including memory optimization/planning), operator/kernel support, and the maturity of the surrounding toolchain and ecosystem [200].

Beyond inference, a mounting interest concerns on-device adaptation, also known as continual and decentralized learning, for privacy-preserving and low-latency applications. It acknowledges that numerous continual learning approaches continue to be challenging when compared with MCU constraints due to their computationally intensive regularization. It is imperative to emphasize the necessity of holistic benchmarking, encompassing metrics such as accuracy, latency, energy consumption, model size, and memory footprint. This approach points out open issues within the TinyDL software ecosystem, highlighting the constrained portability of toolchains utilized by advanced operators, including attention mechanisms and mixed-precision arithmetic, across a heterogeneous array of embedded targets. These gaps motivate research toward runtime stacks and automated

workflows that could be extended from deployment-oriented optimization to training on-device under MCU-class constraints. In this direction, [105] propose an automated framework that combines an iterative optimization stage (structured pruning, quantization, and SRAM/Flash-driven memory scheduling) with automatic code generation exploiting hardware-aware acceleration to produce efficient MCU-level inference code. The effectiveness of the framework has been demonstrated through validation on representative CNN models, such as ResNet-8 and MobileNetV2, as well as STM32-based platforms. This validation demonstrates that compiler-oriented deployment and scheduling can effectively reduce runtime memory requirements. In particular, MobileNetV2 has been shown to achieve a reduction in SRAM usage of 51–57% and in flash usage of 17–62%, when compared with TensorFlow Lite Micro. Furthermore, the inference latency has been demonstrated to improve by up to 55%, depending on the sparsity and quantization configuration. These results underscore that, even when focusing on inference, the deployment of embedded DL is a system-level problem in which optimization decisions must be taken across the model, the memory plan, and the generated execution code. Consequently, extending analogous automation to on-device training remains a relevant open challenge for MCU-class hardware.

Federated Learning Integration in Embedded Systems

In recent years, FL has also begun to be adopted in resource-constrained systems, such as embedded systems and Industrial IoT applications. In these contexts, the federated paradigm is particularly promising, as it reduces dependence on the cloud and allows data to be processed directly near its source. As noted by [172], the integration of FL and embedded systems requires lightweight models and optimization techniques such as quantization and pruning to comply with the computational and energy limitations of the devices.

Recently, [3] showed how FL, combined with TinyML, can be used for predictive maintenance and anomaly detection in industrial settings. Using low-cost microcontrollers and boards, data collected from distributed sensors is processed locally with models such as Support Vector Machines, Random Forests, and lightweight neural networks. The local results are then aggregated in a federated manner, obtaining a global model capable of monitoring the status of plants in real time, preventing failures and reducing operating costs.

In addition to industrial settings, FL has been successfully applied in various domains, including mobile devices, such as predictive text and voice recognition, healthcare, where it allows for the analysis of biometric data without centralizing patient information, and network security, where it supports the detection of cyber threats without exposing sensitive logs. Other emerging applications include smart grids and urban IoT systems, which benefit from decentralized data processing for energy forecasting, traffic management, and environmental monitoring.

2.3.2 Trustworthiness

ML systems influence critical decisions that have consequences in human daily lives and for this purpose society need trust in them. AI and ML are critical in enterprise workflows because of a lack of transparency into the inner working and a potential lack of reliability [214]. The *High-Level Expert Group on Artificial Intelligence (AI HLEG)*, under the direction of the European Commission, has developed guidelines with key specifications for trustworthy AI systems [22]. The document consists of three layers of abstraction, from the most abstract to the most concrete. At the conceptual level, according to EU guidelines, trustworthiness of an AI system should align three main components at the top level throughout its evolution.

- (1) Lawful: the AI system must comply with all applicable laws and regulations, *e.g.*, the General Data Protection Regulation (GDPR).
- (2) Ethical: the development of AI systems is in accordance with the ethical values of human society.
- (3) Robust: the AI system is technically and socially reliable so that it does not generate harmful risks that AI systems can cause.

2.3.2.1 Explainability and Interpretability

A critical challenge in modern ML is the tension between predictive performance and interpretability. Conversely, traditional statistical models are characterized by their transparency and ease of interpretation. In contrast, many ML and DL models, particularly deep networks, are often regarded as "black boxes," due to their limited explainability [33, 108]. This limitation is particularly problematic in the healthcare or the safety-critical sectors,

where operators must trust and understand the decisions made. Bridging this gap requires the integration of ML with knowledge representation and reasoning (KR&R) methods. The integration of these methods is crucial to formalizing and structuring the knowledge extracted from signals and high-dimensional data sources.

Among ML models, Decision Trees (DTs) are widely recognized for their intrinsic interpretability, as their hierarchical structure naturally encodes a sequence of if-then rules that can be directly inspected by human users. [66] emphasize that each decision path in a DT corresponds to a transparent sequence of feature-based conditions, making the reasoning process explicit and verifiable. [47] highlight that the readability and traceability of the DT rules are key advantages, particularly in safety-critical applications, because understanding the logic of the model is essential for accountability and regulatory compliance. Furthermore, DTs have inherent transparency that facilitates debugging and validation by allowing errors to be traced back to specific branches of the tree. In bioinformatics, [120] reports the interpretable representation of DTs within biological relationships, because hypotheses can be deriving from a classification pattern such as gene-function associations or protein-interaction mappings.

In addition, healthcare applications are able to take advantage of the explicit rule sets generated by DTs, such as [170] demonstrated how these rules sets help clinicians understand how variations in biosignals contribute to predictions of medical conditions. Likewise, [58] employ DTs to classify the outputs of dynamic system models, showing that the resulting decision rules are not only accurate but also provide insight into underlying behaviors such as growth, oscillation, or equilibrium.

Within the healthcare domain, interpretability can also be externally induced by extracting decision rules from more complex models. [107] proposed an interpretable framework for Clinical Decision Support Systems (CDSS), where Decision Trees are automatically extracted from textual or model-based representations and expressed as if-else pseudocode. Each branch of the tree represents a clinical reasoning step that links observations, plans, and actions through a custom strategy. This approach enables clinicians to visualize the decision logic underlying the system's recommendations, allowing the resulting rules to be transparent and easily interpretable by medical professionals.

In contrast, interpretability is one of main limitations of RFs. Mean Decrease in Impurity (MDI) and Mean Permutation Importance (MPI) metrics give good information about variables [33], but these measurements are often insufficient for critical applications such as healthcare and security systems. Ensemble pruning approaches reduce the number of

trees while maintaining performances, thereby improving readability and computational capacity [118]. Beyond a certain number of trees, the gains in accuracy are negligible, while computational costs (training time, memory, energy) increase.

SVMs offer a good balance between accuracy and generalization, but they are hard to interpret because they rely on high-dimensional feature spaces and kernel transformations that obscure the relationship between inputs and outputs.[97] explains that autoregressive parameters and residual error have a direct relationship with the physical property of the structure, as well with structural anomalies. This means that the decision of the SVMs and the state of damage of the structure have an interpretable binding. [224] highlights that while SVMs perform well in fault diagnosis tasks, their decisions are not inherently transparent, since classification depends on support vectors and kernel functions that lack intuitive meaning. Nevertheless, it is argued that using feature engineering techniques such as Principal Component Analysis (PCA) can improve interpretability by identifying which features contribute most to the separation of classes.

One of the most debated limitations of DL is its *black-box* nature. Deep networks base their predictions on the composition of many distributed parameters, making it difficult to interpret the reasons behind a decision [175]. This problem is critically important in safety-critical sectors, such as healthcare, autonomous driving and finance, where transparency and explainability are fundamental to the acceptance of AI-based solutions [13, 175].

To fill this gap, with the growing interest in XAI, *post-hoc explainability* methods are applied for analysis of model decisions. Among the most frequently adopted there are *SHapley Additive exPlanations* (SHAP) [138], Partial Dependence Plot (PDP) [32] and Local Interpretable Model-Agnostic Explanations (LIME) [177].

In [103] a relationship of interpretability with robustness and security is highlighted. Specifically, causal learning, invariant feature extraction, and noise-robust representation learning can enhance the transparency of model reasoning by correlating learned features with meaningful causal factors.

2.3.2.2 Explainability dimensions and evaluation metrics

Explanations in AI systems traditionally address *What; Why; and How* questions focusing on interpreting the functioning of black-box ML models, *i.e.*, identifying which input features most influence the outputs. Beyond these questions, there is a need for explanations that address user-specific goals, focusing on *what the main factors influencing a decision*

were; whether changing a certain factor could have altered the decision; and why two seemingly similar cases received different decisions, or vice versa. To address these questions, an AI system modeled in a domain KB should incorporate machine-readable and semantic modeling that takes into account the user's context, providing comprehensible, context-aware, and provenance-enabled explanations of its reasoning, also suitable for user requirements.

In line with the competency questions defined by Chari *et al.* [49] for selecting appropriate explanation types in user-centered AI design and generation, the *Explanation Ontology* framework identifies nine major categories of explanations: *case-based*, *contextual*, *contrastive*, *counterfactual*, *everyday*, *scientific*, *simulation-based*, *statistical*, and *trace-based*. Each type of explanation:

1. serves a different purpose, depending on the user's point of view;
2. is suited to different types of situations, contexts, and levels of knowledge;
3. is generated by different AI tasks (*e.g.*, reasoning modes) and implemented through different methods;
4. requires distinct types of information to be constructed.

These aspects define what [49] refer to as the *sufficiency conditions* of each explanation type, that is, the minimal logical and informational requirements that must be satisfied for a system to generate that kind of explanation. Within the ontology, these conditions are formally represented as OWL restrictions linking classes such as *Explanation*, *AITask*, *Knowledge*, and *SystemRecommendation*.

For example, (1) user-centered but different explanation indicate that, while a *contrastive explanation* (why choose option A over option B) is useful for users that wish to compare alternatives, a *scientific explanation* benefits those seeking empiric evidences, whereas a *trace-based explanation* supports users interested in understanding the reasoning process. Fitting particular situations, contexts, and user knowledge levels (2) refers to effectiveness of each explanation type, which changes according to the application domain, *e.g.*, health-care or industrial, and the user's expertise level. In other words, not all explanations are equally suitable for everyone.

Keeping track of the reasoning type used (*i.e.*, inductive, deductive or abductive) and implemented through diverse methods (3) means that some explanation are better derived from

symbolic models, *e.g.*, *trace-bases* or *scientific*, while others from data-driven approaches, *e.g.*, *statistical explanation*.

The last requirement (4) concerns the types of information needed to generate each explanation. For instance, a *case-based explanation* requires a similar past case; a *scientific explanation* requires related scientific works, and a *contextual explanation* requires information about the user or environment that influenced the computation.

According to Naveed *et al.* [152], the quality of an explanation can be defined and assessed through two complementary approaches. Formally, explanation quality refers to the structural and methodological attributes of an explanation, such as its form, content, and internal consistency.

Functionality-grounded evaluations focus on the internal validity of explanations, assessing how accurately, reliably, and compactly the explanation represents the underlying AI model. These metrics include:

- *fidelity*, which measures how accurately the explanation reproduces model behavior;
- *stability*, which evaluates whether similar inputs lead to similar explanations;
- *consistency*, which assesses whether multiple explanations for the same data remain coherent across different model instances;
- and *sparsity*, which quantifies the compactness of explanations and their ability to minimize cognitive load without sacrificing completeness.

Human-centered evaluations, on the other hand, emphasize the user's perspective and cognitive interaction with the explanation based on the quality of assessed explanation. Here, quality is determined by how well the explanation resonates with users' cognitive abilities, goals, and context of use, focusing on *complexity*, *completeness*, *consistency*, and the clarity of *input–output relationships*. Additional constructs include perceived *why-understanding*, *information sufficiency*, and *explanation correctness*, typically measured through questionnaires, self-report scales, or observational studies. These qualitative dimensions are typically evaluated through user studies, surveys, and task-based experiments, reflecting the perceived transparency and usability of AI systems in real-world scenarios.

2.3.3 A five-star rating of eXplainable AI for pervasive CPS

The integration of on-device AI within the Internet of Everything (IoE) paradigm necessitates the capacity to execute learning and inference processes directly on embedded hardware. This development eliminates the need to rely on cloud or edge infrastructure for computationally intensive operations. However, executing training and inference tasks for the ML framework on embedded devices alone is insufficient to fully actualize the IoE vision. In the specific context of IoE, the consideration of unique data management characteristics, Machine-to-Machine (M2M) interactions in smart object networks, and trustworthiness is paramount. Indeed, these factors are closely linked to the challenges posed by big data, including its velocity, variety, and veracity [208]. In light of the growing architectural and functional diversity of available AI-oriented IoE approaches, the development of a simple structured categorization framework for assessing the feasibility of on-device AI solutions in relation to use case requirements is recommended. Inspired by Berners-Lee's 'five-star' model for Linked Open Data (LOD) [25], a similar rating scheme is proposed here to evaluate the progressive integration of AI and XAI capabilities into pervasive CPS. The five stars represent ascending levels of intelligence and transparency, from basic on-device learning to semantically grounded explainability, and is also resumed in Table 2.2

★ **On-device training and inference.** The first level concerns the possibility of performing both the *training* and the *inference* tasks locally on embedded nodes. This is a fundamental requirement for giving autonomous intelligence to IoE devices. Since MCUs impose severe constraints in terms of memory and computing power, models must be simplified and optimized, even though AI-oriented hardware designs are increasingly available leveraging FPGAs or domain-specific accelerators [178]. *TinyML* techniques [189] allow to scale down model size and computational effort: reducing model *depth* and/or *width*, *i.e.*, the number of layers and/or the amount of functional units per layer; model *compression* by exploiting *pruning* and *quantization* to reduce unit density and precision, respectively, without sacrificing predictive performance excessively [178].

★★ **Incremental learning.** The second star reflects the ability to adapt continuously to the dynamics of the data, on-device. In IoE environments, where sensor data are characterized by high *velocity* and contextual volatility, models must handle both *data drift* and *concept drift* [217]. Data drift represents changes in the relationships among the independent features, *i.e.*, to variations in the distribution of input features; while concept

drift describes changes occurring in the relationships between inputs and outputs of ML models, *i.e.*, the relationship between features and target variables. *Incremental* or *online learning* techniques address these challenges by updating model parameters periodically or continuously as new data are acquired, without requiring complete retraining.

★★★ **Federated learning.** The third star highlights the integration of FL as a decentralized approach to training across heterogeneous and distributed devices. Given the *variety* of sensors, contexts, and data modalities in large-scale IoE deployments, *locality* effects play a significant role: the set of measurable independent features can be different, as well as their relationships both among themselves and with dependent variables. This suggests that smaller local models may be more suitable for capturing specific patterns than larger aggregated ones. However, when evaluated on the entire observed environment, larger aggregated models can still demonstrate higher overall accuracy. For this reason, FL is a crucial capability of advanced IoE solutions, where only trained models or intermediate data structures are propagated in the network and aggregated at higher hierarchical levels. Federated paradigms allow these local models to collaborate through aggregation of parameters or intermediate representations, rather than raw data, thus preserving privacy and allowing IoE networks to learn collectively.

★★★★ **Explainability.** The fourth star concerns the *transparency* and *interpretability* of AI-based decisions. As embedded intelligence becomes more pervasive, trustworthiness becomes an indispensable requirement. Data *veracity* issues affect IoE scenarios due to accuracy and robustness limitations of low-cost sensing equipment, compromised by environmental noise, sensor bias, or hardware degradation. Therefore, XAI techniques are needed to ensure that the reasoning behind model outputs is both *interpretable* and *comprehensible*: the former concerns understanding the inner workings of trained models, the latter regards the meaning of prediction outcomes. *Decision trees* have inherently good transparency among classic ML methods, while NN and DL architectures have opaque behaviors in both their models and outcomes. XAI includes model-agnostic and model-specific methods. Model-agnostic techniques are typically based on *surrogate models* providing models that approximate complex black-box behaviors through locally linear explanations, *e.g.*, Local Interpretable Model-agnostic Explanations (LIME) [177], SHapley Additive exPlanations (SHAP) [139] and their variants. Conversely model-specific methods are integrated within DL architecture and application to visualize internal activations or to trace the causal contribution of individual features [81]. Explainable on-device AI techniques are needed in order to make autonomous networks of IoE devices trust-

worthy; however, despite these advances, the human cognitive effort required to interpret such outputs remains significant, highlighting the need for machine-driven explanation mechanisms that go beyond visualization.

★★★★ **Semantic XAI.** The fifth and most advanced level corresponds to the convergence between explainability and symbolic reasoning. In order for a system to be truly comprehensible, it has to “*emit symbols enabling user-driven explanations of how a conclusion is reached*” [79]. This implies that the explanation process should be expressed through formal and machine-understandable languages. *Neurosymbolic* approaches combine symbolic formalisms with DNN architectures to achieve interpretable and comprehensible systems. KRR is often exploited for symbolic XAI, particularly Semantic Web standards [81] such as Resource Description Framework (RDF) [191] and OWL [162]. Benefits include: machine-understandable semantics, enabling self-evaluating autonomous agents; longstanding adoption by software tools and by the community; support of formal frameworks for data provenance like PROV [150] and for constructing several types of explanations [48] via automated reasoning. Due to these reasons, the integration of semantics-based XAI appears as a compelling aspect of the evolution from the IoE to the Semantic Web of Everything (SWoE). Nevertheless, due to language verbosity and algorithmic complexity of reasoning, conventional KRR approaches conceived for the Web require careful adaptation and optimization in order to be computationally sustainable in SWoE platforms.

Table 2.2: Five-star rating model for ML explainability and intelligence.

Rating	Feature	Description
★	On-Device Training & Inference	ML models are trained and executed directly on embedded devices.
★★	Incremental Learning	Models are continuously updated from new data without full retraining.
★★★	Federated Learning	Enables devices to collaboratively update models by sharing learned knowledge, without exposing raw data.
★★★★	Explainability	Model content and prediction outcomes are accompanied by human-understandable explanations.
★★★★★	Semantic Explainability	Explanations are based on formal semantics and symbolic reasoning for greater M2M interoperability.

Chapter 3

Embedded Subsystem Design for monitoring vital parameters

3.1 Hazard Analysis in railway systems

This Section presents the hazard analysis performed for safety-critical railway systems operating under GoA4 conditions. The objective is to systematically extract safety-relevant hazards through a norm-compliant HAZOP methodology and to translate them into explicit safety requirements. The focus of the analysis is not on hardware design, but on establishing traceability between hazardous scenarios, safety-related functions, and the behavioral constraints that must be satisfied by onboard embedded subsystems. The extracted hazards and derived requirements constitute the basis for the experimental evaluation of system intervention times discussed in Section 5.1.

3.1.1 HAZOP

Hazard analysis background. The main objective of hazard analysis is to develop a scenario-based understanding of the *safety gap* of a system, *i.e.*, the set of conditions under which system behavior may become unsafe with respect to its operational context. In safety-critical applications, hazards cannot be inferred only from isolated component failures: interaction effects and coupled behaviors across subsystems must also be considered. Therefore, to identify potential hazards, all relevant combinations of equipment functionalities and subsystem interactions must be taken into account [84].

Preliminary Hazard Analysis (PHA) addresses this need in the early design stages of

electrical/electronic/programmable electronic (E/E/PE) safety-related systems. The goal is to identify hazards proactively, define initial safety measures, and support compliance with CEI EN 61508 [45]. Under CEI EN 61508, the scope of PHA includes the system components and their interactions, with the purpose of addressing hazards across the entire system lifecycle. To achieve the required safety integrity for the system functions, CEI EN 61508 adopts an overall safety lifecycle as a technical framework. For each phase of the lifecycle, objectives, scope, required inputs, and expected outputs are defined.

In this perspective, the early phases are particularly relevant.

- The ***Concept*** phase outlines project objectives and studies the application domain, context, and intended purpose of the system, aiming at a sufficient understanding of physical constraints, legislative aspects, and interfaces.
- The ***System definition and context of operation*** phase defines mission profile and system boundaries, and establishes the security policy and security plan. At this stage, subsystems and their functions are identified: for each i -th equipment, the set of functions is listed, and, where applicable, each j -th function is further decomposed into k -th subfunctions.
- ***Risk analysis and assessment*** determines hazards, hazardous events, and hazardous situations under reasonably foreseeable circumstances, and provides guidance on interfaces and interactions among subsystems or external systems.

In practice, it is not always possible to assess risk mathematically based on reliable data; therefore, the analysis is often initially qualitative, identifying determinants and their combinations before quantification becomes feasible [44].

HAZOP method and guidewords. To support systematic deviation elicitation in complex systems, this work adopts HAZOP (Hazard and Operability Study), a structured, deviation-based hazard analysis technique widely applied to identify deviations from intended design and operation that may lead to safety risks or operability issues [46]. HAZOP investigates departures from the design intent by combining *guidewords* with the functions and interfaces under analysis. In control-centric and software-intensive architectures, the analysis is typically performed on functional blocks and information flows, and guidewords are interpreted with respect to both *logical correctness* and *timing constraints*. In

this work, for each safety-relevant function, the design intent and the operational context in which the function is active are defined, together with the interface assumptions (inputs/outputs and expected timing).

The guidewords adopted in the analysis are reported in Table 3.1. They include: **NO/NOT** complete negation of the intention from the design; **MORE** quantitative increase in whatever is being identified; **LESS** quantitative decrease in whatever is being identified; **PART OF** qualitative modification or decrease; **LATE** something occurred later than intended clock time, **EARLY** something occurred earlier than intended clock time, and **OTHER THAN** complete substitution. The guidewords are applied to each j -th function to generate deviations; when a function is decomposed into k subfunctions, **PART OF** is used to capture partial fulfillment of the intended behavior.

Guideword	Description
NO/NOT	Function not executed or not defined
MORE	Excess / increase of the relevant variable(s)
LESS	Shortage / decrease of the relevant variable(s)
PART OF	Function performed or considered only partially
LATE	Function performed later than the defined time
EARLY	Function performed earlier than the defined time
OTHER THAN	Abnormal execution (wrong value / wrong mode / wrong behavior)

Table 3.1: HAZOP guidewords adopted in this work (EN 61882) within the railway RAMS process (EN 50126) [46, 44].

Procedure: deviations, grouping, and scenario synthesis. Once the system is divided into sections (equipment and functions), standard guidewords are applied to each function to explore potential deviations from nominal operation. For each j -th function, the applicability of guidewords (**NO/NOT**, **MORE**, **LESS**, **EARLY**, **LATE**, **OTHER THAN**) is assessed; if the function is expressed as a combination of several k subfunctions, the **PART OF** guideword is applied.

After deviation generation, functions are grouped according to the guideword that characterizes their deviation. For example, all j -th functions across the equipment that can be affected by **NO/NOT** are grouped together; the same procedure is repeated for the other guidewords. This grouping step enables the analysis of whether multiple deviations can co-occur and jointly lead to hazardous conditions.

Within each group, deviations are combined to obtain hazardous scenarios associated with the system specification and operational context. These scenarios can be interpreted as logic paths triggered by faults or malfunctions across subsystems, potentially requiring mitigation actions. Achieving exhaustive hazard detection requires a logical understanding of the application under analysis in which system parts can interact; in this way, each combination of subsystem functions represents a possible nominal or hazardous event. In the early phases, PHA leverages design information and known hazard knowledge to identify hazards, causal factors, effects, and risk levels, and to derive procedural or technological mitigations that can inform architectural decisions when information is still limited [84].

Outputs: mitigations and requirements (verification-ready specification). The output of the HAZOP sessions is a set of *hazardous scenarios* expressed as context-conditioned deviations, together with their associated mitigations. These scenarios are then translated into standardized requirements to obtain a verification-ready specification and to preserve traceability from hazard identification to enforceable behavioral constraints for the onboard embedded subsystem. Although HAZOP is systematic and particularly suitable for interaction between safety-critical architectures (*e.g.*, traction, signaling, and communication subsystems that must operate coherently), it is also resource-intensive: it requires expert knowledge and significant manual effort, and its cost grows with the number of functions and operational modes, which hinders scalability in highly integrated systems [4, 99]. Figure 3.1 shows the workflow adopted in this thesis.

Railway GoA4 use case. In the context of GoA4 railways, the hazard analysis has been applied to the Vital Control Module (VCM), a safety subsystem realized in collaboration with *Rete Ferroviaria Italiana (RFI)* [70]. The VCM was conceptualized as a hardware/software module integrated into the ATO over ETCS framework to supervise vital parameters, guarantee compliance with safety requirements in the absence of a driver, and enable faster hazard mitigation compared to conventional Automatic Train Protection (ATP) systems [148]. The VCM was assigned a series of functions, as determined by a thorough hazard analysis. The following tasks are to be carried out: (i) supervision of subsystem operativity, (ii) validation of remote operator vigilance, (iii) continuous monitoring of odometry and train speed profiles, (iv) verification of ETCS status (isolation/insertion), and (v) triggering of the Emergency Braking (EB) system under hazardous

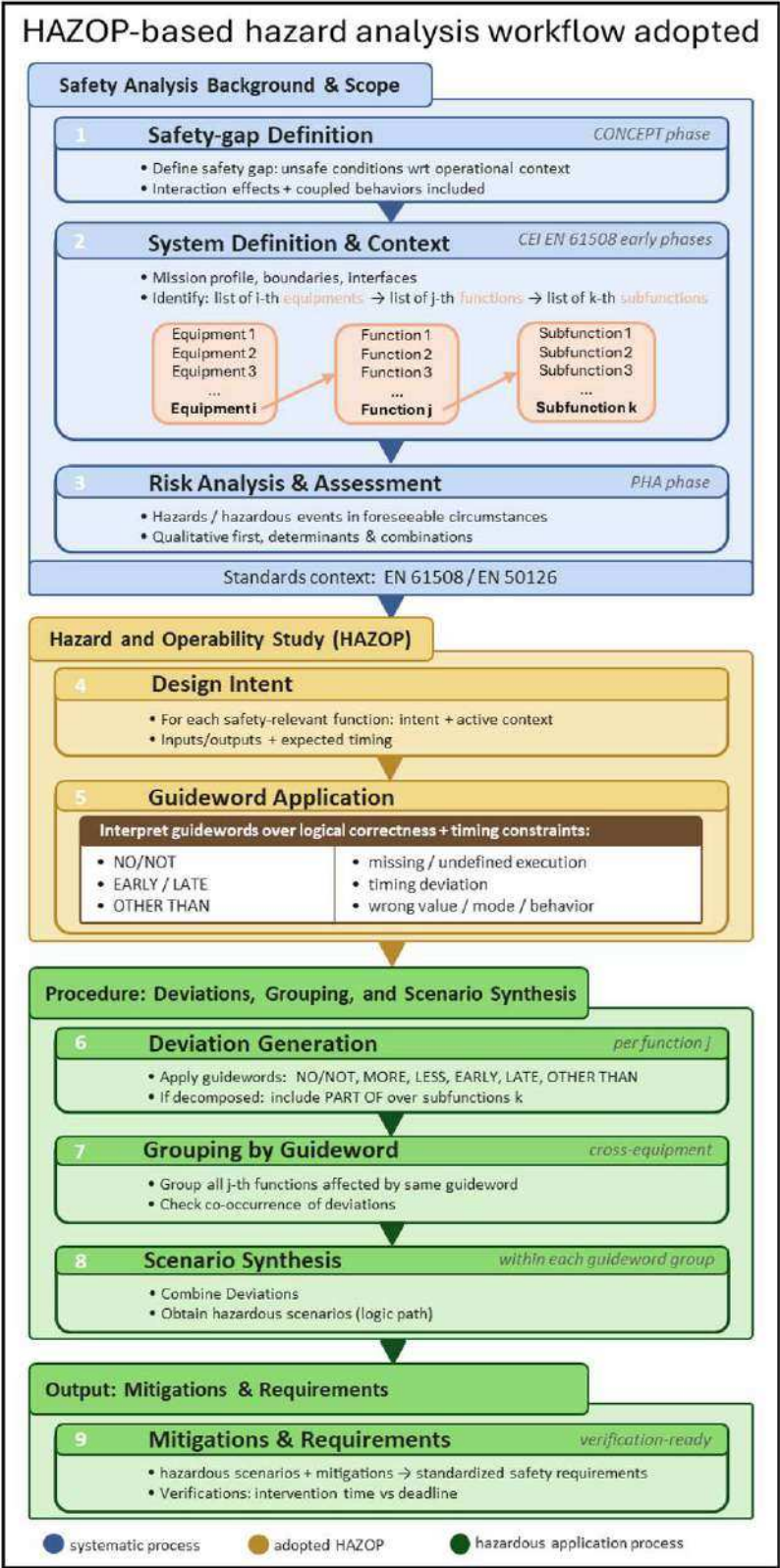


Figure 3.1: HAZOP workflow used

conditions [147, 12]. In particular, the HAZOP-driven elicitation is reflected in the VCM application logic, linking hazardous scenarios to specific safety-related functions and to their implementation in the module [12].

3.1.2 Extracted Hazards

The hazard analysis conducted for the VCM has identified various classes of hazardous events. In certain scenarios, emergency braking is initiated, subsequently followed by safe recovery. This occurs, for example, in the presence of an explicit EB request from a remote or autonomous operator, or when overspeed is detected through odometry while ETCS is isolated. In such circumstances, the braking system can be reinitialized once the train comes to a halt, enabling operations to resume safely. However, in other scenarios, system decay ensues, and recovery is deemed to be unsafe. This phenomenon transpires in instances such as the presence of invalid responses from the ATO subsystem, the failure to confirm operator vigilance within stipulated deadlines, or the malfunction of ETCS isolation/insertion commands. In such conditions, the system enforces a permanent stop until a manual reset is performed, ensuring a safe restart [147]. Table 3.2 reports the standardized requirements extracted from the hazard analysis. Unsafe control actions have been identified and translated into corresponding safety requirements. Refined safety constraints have then been converted into safety-critical and system-functional requirements, consistently aligned with the hazard analysis outcomes and project constraints.

The hardware and software implementation of the Vital Control Module does not represent an original contribution of this work. The VCM architecture is reported here solely to contextualize how the hazards extracted through a norm-compliant HAZOP process are translated into concrete design constraints and safety-related functions.

3.1.3 Hazard analysis on subsystem design

From an architectural standpoint, the VCM has been implemented as a Eurocard-sized PCB that incorporated an *UltraZed-EG* System-on-Module (Xilinx Ultrascale+ MPSoC). The board integrates multiple interfaces (SPI, CAN, RS-422/485, Ethernet) to facilitate connectivity with on-board equipment, and it complies with EN 50155 and EN 50124 for ensuring hardware robustness. The software stack is built on a hard real-time operating system (RTOS) compliant with EN 50128. It is designed with a micro-kernel architecture

SAFETY REQUIREMENTS

ATO fail mode - ODO available - train on moving
ATO not vitality check - ODO available - train on moving
ATO unpowered - ODO available - train on moving
Operation mode incorrect - ODO available - train on moving
Remote operator not vigilant - ODO available - train on moving
ETCS status not known - ODO available - train on moving
ATO fail mode - ODO out of service
ATO not vitality check - ODO out of service
ATO unpowered - ODO out of service
Operation mode incorrect - ODO out of service
Remote operator not vigilant - ODO out of service
ETCS isolated - ATO mode correct - ODO out of service
ETCS status known - ETCS isolated - ATO mode OK - ODO available - maximum speed exceeded (>30 km/h)
CHOSEN SAFETY REQUIREMENT
ETCS status not known - ODO out of service

Table 3.2: System statement

to ensure deterministic behavior under real-time constraints. The application logic, modeled in Simulink/Stateflow and subsequently compiled into MISRA C-compliant code, is divided into three primary tasks: Task A is responsible for vitality and vigilance queries; Task B is dedicated to odometry supervision, ETCS/EB monitoring, and hazard detection; Task C manages ETCS procedures and executes EB commands [147, 12].

The hazard-driven derivation of safety requirements described in this Section provides a structured basis for evaluating the temporal behavior of safety-related functions. In the following experimental analysis, the response times of the VC module are quantitatively assessed under representative hazardous scenarios derived from the identified hazards.

3.2 Biosensing Design for Cystic Fibrosis

The need for continuous and non-invasive monitoring of chloride concentration in sweat, discussed in Section 2.1.1.2, motivates the development of embedded biosensing solutions specifically tailored for cystic fibrosis screening. While the sensing principle itself is well established at the laboratory level, its translation into a wearable and low-power device requires a careful balance between sensing performance, technological feasibility,

and regulatory constraints.

In this work, the design of the biosensing system is approached from a system-level perspective, in which the sensing element, the embedded electronics, and the calibration strategy are conceived as interdependent components. Particular attention is given to the selection of the potentiometric sensing configuration, the geometry and materials of the electrodes, and the characteristics of the embedded platform used for signal acquisition and processing. These elements are critical to achieving stable and repeatable measurements under the constraints imposed by wearable operation.

Beyond sensing accuracy, additional factors such as energy consumption, miniaturization, and compliance with medical device standards play a decisive role in shaping the overall architecture. As a result, the design choices discussed in this section are driven not only by electrochemical considerations, but also by the practical requirements of embedded healthcare applications, where reliability and process quality are essential throughout the system lifecycle.

The design described in the following sections defines the experimental conditions under which the biosensing platform is evaluated. In particular, the preparation of reference chloride solutions and the adopted calibration approach establish a controlled framework for the sensing experiments presented in Section 5.2. This separation between design rationale and experimental validation allows the performance of the proposed biosensing system to be assessed in a structured and reproducible way in cystic fibrosis screening applications.

3.2.1 Related work and design requirements

The development of embedded biosensors for chloride monitoring in CF screening has been widely investigated in literature, motivated by the need to overcome the limitations of conventional laboratory-based sweat tests. In this context, the concentration of chloride ions in sweat is recognized as a reliable biochemical marker for cystic fibrosis diagnosis and follow-up, and electrochemical sensing approaches have been identified as suitable candidates for continuous and non-invasive monitoring.

Potentiometric sensing has emerged as the most widely adopted approach for wearable chloride monitoring, due to its intrinsic low-power operation, simplicity of the readout circuitry, and suitability for miniaturized embedded implementations. Early works demonstrated the feasibility of potentiometric chloride quantification in sweat using Ag/AgCl-

based electrodes, highlighting the relevance of this technique for CF screening applications [94]. Subsequent studies introduced improvements in electrode stabilization and system robustness, for instance through the use of salt bridges or polymeric coatings, achieving near-Nernstian sensitivity and acceptable long-term drift [52, 54, 53]. More recent contributions have focused on fully wearable platforms integrating electrodes and electronics on flexible substrates. In particular, flexible and wearable potentiometric systems have been proposed for real-time chloride monitoring, combining planar Ag/AgCl electrodes with embedded readout electronics and wireless data transmission [83, 230]. These solutions demonstrate the potential of wearable biosensing for CF screening, while also revealing critical challenges related to electrode geometry, mechanical deformation, power consumption, and system integration.

Within this research landscape, works in [65, 63, 64] address chloride biosensing from a system-level perspective, encompassing electrode design, flexible and replaceable sensing elements, and embedded mixed-signal architectures with power-aware operation. These studies show that achieving reliable and repeatable sensing performance in a wearable context requires the joint optimization of sensing materials, electrode geometry, and electronic readout, rather than treating these aspects independently.

Based on the analysis of the related work, the following design requirements have been identified for the proposed biosensing system: (i) non-invasive chloride monitoring suitable for wearable use; (ii) potentiometric sensing to enable low-power and continuous operation; (iii) planar and flexible electrode geometries compatible with miniaturized fabrication techniques; (iv) stable and repeatable measurements over the concentration range relevant for cystic fibrosis screening; and (v) an embedded electronic architecture aligned with quality and risk management principles for medical devices. These requirements guide the design choices discussed in the following sections.

3.2.2 Sensing principle and potentiometric configuration

Potentiometric sensing principle The biosensing platform presented in this work relies on a potentiometric sensing principle for the quantification of chloride ions in sweat. Potentiometry measures the open-circuit potential (OCP) established between a selective electrode and a reference electrode under equilibrium conditions, where negligible current flows through the electrochemical cell. In this configuration, the measured potential directly reflects the activity of the target ions in the electrolyte, enabling continuous and

real-time monitoring without perturbing the sample [223, 140].

When ion-selective electrodes are employed, the relationship between the measured potential and the ionic concentration is governed by the Nernst equation, which predicts a logarithmic dependence of the electrode potential on ion activity. In the specific case of chloride sensing using Ag/AgCl electrodes, the ideal sensitivity approaches a theoretical slope of approximately 59 mV per decade at room temperature. This property makes potentiometric sensing particularly attractive for wearable applications, as it allows quantitative measurements with simple readout circuitry and minimal energy consumption [223, 63]. Different electrochemical configurations can be adopted for potentiometric sensing, including three-electrode and two-electrode setups. The two-electrode configuration is less complex than three-electrode systems. In this configuration, a single electrode functions as both the reference and counter electrodes, while the other serves as the selective or working electrode. This architecture has been shown to reduce circuit complexity, number of connections, and energy consumption, making it particularly suitable for embedded and wearable systems. Despite the potential limitations in absolute measurement accuracy compared to the three-electrode configuration, several studies have demonstrated that meticulous geometric design and the utilization of stable materials enable accuracy and repeatability to be maintained even in reduced configurations [140, 227]. For these reasons, the proposed biosensing architecture adopts a two-electrode potentiometric configuration, in which the reference electrode (RE) and the working electrode (WE) are immersed in the same electrolyte and directly establish the measured potential difference. This configuration significantly reduces system complexity, number of interconnections, and power requirements, while maintaining sufficient accuracy and repeatability for CF screening, as demonstrated in several wearable chloride sensing studies [52, 54, 83]. The adoption of a two-electrode configuration introduces additional design challenges related to reference stability and drift, particularly in variable ionic environments such as sweat. Consequently, the sensing principle must be supported by appropriate electrode materials, geometrical design, and stabilization strategies to ensure a reliable and reproducible potential over time [53, 155, 63]. These aspects are addressed through the electrode design choices.

Overall, the potentiometric sensing principle and the selected two-electrode configuration provide a favorable trade-off between sensing performance, system simplicity, and energy efficiency. This balance is essential for the realization of an embedded biosensing platform suitable for wearable CF screening, and it establishes the foundation for the electrode design

and embedded implementation presented in the next sections [64].

Nernst equation and theoretical response Potentiometric sensing relies on the measurement of the open-circuit potential (OCP) established at equilibrium between a working electrode and a reference electrode, in the absence of significant current flow. Under these conditions, the measured potential is directly related to the activity of the target ionic species in the electrolyte. For ion-selective electrodes, this relationship is described by the Nernst equation, which defines the theoretical dependence of the electrode potential on ionic concentration. In potentiometric measurements of cystic fibrosis (CF), the activity of chloride ions (Cl^-) is derived from the potential difference between two electrodes—namely, a Working Electrode (WE) and a Reference Electrode (RE). The measurement principle follows the Nernst equation:

$$E = E^0 + \frac{RT}{zF} \log(a_{\text{ion}}) \quad (3.1)$$

where E is the measured potential, E^0 is the standard electrode potential, R the gas constant, T the absolute temperature, F the Faraday constant, and z the ionic charge number. For chloride detection ($z = 1$), the theoretical sensitivity is approximately 59.2 mV/decade at room temperature. Deviations from this ideal slope are primarily related to electrode surface morphology, reference stability, and material interfaces.

This logarithmic relationship provides the theoretical reference for the interpretation of potentiometric measurements and represents a key design constraint for both the sensing electrodes and the associated electronic readout. In practical implementations, deviations from the ideal Nernstian behavior may arise due to non-ideal electrode interfaces, variations in reference electrode stability, surface morphology effects, or changes in the ionic environment. Nevertheless, the Nernst equation establishes the expected trend and slope against which the experimental calibration curves are evaluated.

In the context of wearable chloride sensing for CF screening, the Nernstian response serves as a benchmark for assessing sensor performance, enabling quantitative comparison between different electrode designs and guiding the optimization of materials, geometries, and fabrication processes.

3.2.3 Electrode materials and geometrical design

The performance of a potentiometric biosensor critically depends on the properties of the electrodes, both in terms of material composition and geometrical configuration. In wearable chloride sensing applications, electrode design plays a central role in determining sensitivity, temporal stability, repeatability, and robustness under mechanical stress. Consequently, the selection of electrode materials and geometries must be carefully aligned with the sensing principle discussed in Section 3.2.2 and with the constraints imposed by embedded and wearable operation.

Silver and silver/silver chloride (Ag/AgCl) electrodes are widely adopted in potentiometric chloride sensing due to their well-established electrochemical behavior, stable reference potential, and compatibility with miniaturized fabrication processes. The electrochemical stability of Ag/AgCl electrodes is strongly influenced by factors such as microstructure, thickness and uniformity of the AgCl layer, and resistance to ionic interference. Previous studies have shown that controlled chlorination processes improve long-term stability and reproducibility, making Ag/AgCl electrodes suitable in complex environments [155].

From a technological perspective, the miniaturization required for wearable devices has favored the adoption of planar electrode configurations fabricated using deposition, stencil printing, or other additive techniques. Planar electrodes enable compact layouts, simplified integration with flexible substrates, and reproducible manufacturing. Moreover, planar geometries facilitate direct interfacing with embedded electronics, reducing parasitic effects and improving signal integrity [140, 227].

Electrode geometry has been demonstrated to have a significant influence on sensing performance. Parameters such as electrode shape, surface area, and relative spacing between reference and working electrodes affect sensitivity, response time, and noise levels. In particular, an increased surface-to-volume ratio and a uniform distribution of the active material contribute to enhanced linearity and reduced background noise, which are critical for accurate chloride quantification in wearable applications [140, 227].

Within this framework, several planar Ag/AgCl electrode geometries have been investigated in the literature, including linear and concentric configurations. These geometries offer different trade-offs between sensitivity, mechanical robustness, and ease of integration. In particular, flexible planar electrodes have been shown to maintain stable electrochemical behavior even under repeated bending cycles, highlighting their suitability for wearable platforms [63, 65]. The development of the electrodes followed an incremental

process through three main design iterations, each improving manufacturability, flexibility, and sensitivity.

First-generation electrodes. The earliest prototype employed flexible planar electrodes fabricated on a polyethylene terephthalate (PET) substrate using stencil printing of silver ink [65]. Both the working (WE) and reference electrodes (RE) were defined through stencil-printing silver ink tracks. The reference electrode has been chlorinated by dropping a small quantity (5–10 μL) of bleach (NaClO) to obtain a thin layer of AgCl . An insulating layer was subsequently applied to isolate the conductive paths from the electrolyte, establishing the basic architecture for potentiometric chloride sensing on flexible substrates. The Figure 3.2 illustrate the process to obtain the first generation electrodes.

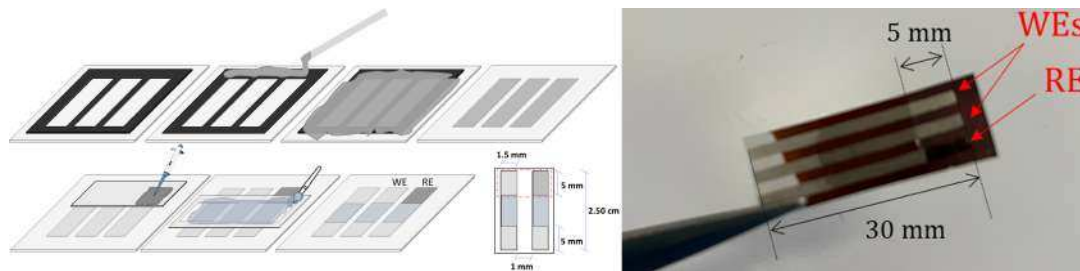


Figure 3.2: LEFT - Electrodes layout and implementation. Top starting from the left: Bonding vinyl stencil to PET, applying silver ink, drying ink and removal vinyl. On the bottom from the left: Bleach application and drying, application and drying of the isolation layer. RIGHT - Photo of the three electrodes: one RE and two WEs.

Second-generation electrodes. The second design iteration improved both the mechanical robustness and chemical uniformity of the sensing area [63]. Electrodes have been fabricated on a Mylar substrate, using a resin stencil created with Fusion 360. Silver ink has been deposited and cured with a controlled airflow (100 $^{\circ}\text{C}$, 50 LPM) to enhance adhesion and reduce microcracks. The Ag/AgCl interface has been obtained by precisely dropping bleach solution on the RE, while the WE remained unchlorinated to act as the ion-sensitive surface, as shown in Figure 3.3. Scanning Electron Microscope (SEM) and Energy Dispersive X-ray Spectroscopy (EDS) analyses revealed 99% Ag purity for the WE and confirmed the formation of an AgCl layer with approximately 23% Cl content, as illustrated in Figure 3.4.

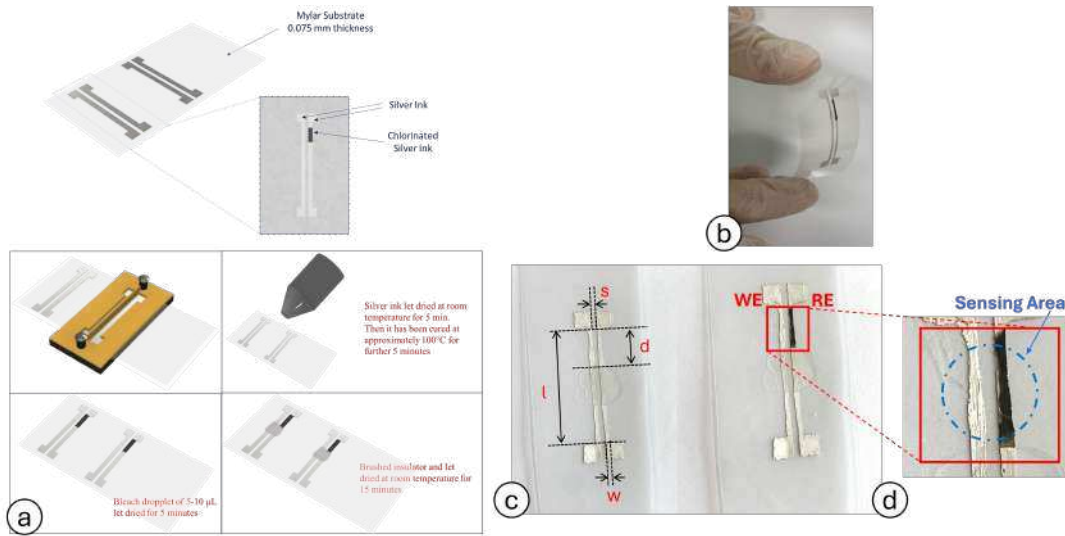


Figure 3.3: *a - up*. Mylar is a clear, pretreated base film with high gloss, low haze. It has also a good thermal stability at higher temperature, it has higher strength and is less permeable to gasses. Ag/AgCl electrodes determine biocompatible sensing area. *a - down*. Once printed a resin stencil, inside of it silver ink has been deposited. Second the ink has been cured at 100°C for 5 minutes. Third 10 μ l have been dropped on one electrode (the right one) to create the RE. Finally, it has been deposited an insulator in the middle to separate the sensing part from the one that will be lead to electronic read-out. *b*. Bending operation on electrodes. *c*. dimensions of electrodes: $d = 10\text{mm}$, $s = 1\text{mm}$, $w = 1\text{mm}$, $l = 30\text{mm}$. *d*. electrode with Ag WE and chlorinated Ag/AgCl RE with highlighted sensing area. 50 μ l of aqueous NaCl solution was dropped onto the electrodes every 10 minutes. The concentrations were increased from 1 mM to 10 mM, 25 mM, 50 mM, 75 mM and 100 mM [63].

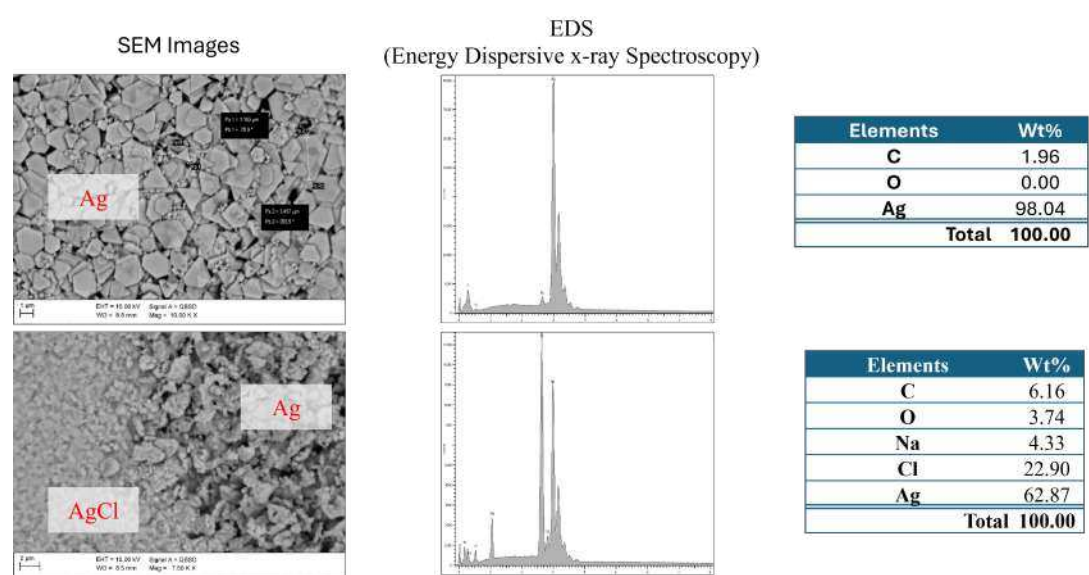


Figure 3.4: Scanning electron microscope (SEM) images have been used to investigate the microstructure, morphology and chemical composition of silver ink. Upper left images show crystal morphology of silver ink before curing process and on the right it is possible appreciate its purity, near 98% of pure silver. Below, instead, on the left it is shown the silver chloride ink after curing ink process. The chemical composition is confirmed by the image on the right where we can find the chlorination of silver.

Third Generation The most recent design iteration introduced significant geometrical optimization and a new passivation method for the reference electrode [64]. The technique used to make the electrodes for this application are based on previous work [63]: as shown in Figure 3.5 two silver tracks have been patterned to define the working electrode (WE) and the reference electrode (RE); and electrodes have been printed on a $75\ \mu\text{m}$ Mylar film, featuring an extended sensing area and enlarged contact pads for improved connectivity. The silver ink has been deposited through stencil printing, followed by a controlled curing process in constant airflow conditions ($50\text{L}/\text{min}$) at 100°C for 10 minutes. This procedure ensured strong adhesion of the conductive layer to the polymeric substrate, preventing delamination during handling or characterization. A major advancement has been introduced in the RE passivation. Unlike previous versions, where a droplet of bleach has been applied to form AgCl by surface chlorination, the new process relied on a short-time immersion technique. The RE has been dipped in a sodium hypochlorite solution for approximately 10 seconds, followed by a resting phase in still air for 15–20 seconds to complete the reaction. This treatment yielded a homogeneous passivating film consisting of silver chloride (AgCl) and silver oxide (Ag_2O), effectively stabilizing the electrode potential and minimizing drift. To stop the reaction, the electrode has been rinsed in deionized water and gently dried at room temperature. The resulting $\text{Ag}_2\text{O}/\text{AgCl}$ composite layer, darker in color, acted as a stable redox buffer, providing high potential reproducibility even under fluctuating chloride concentrations.

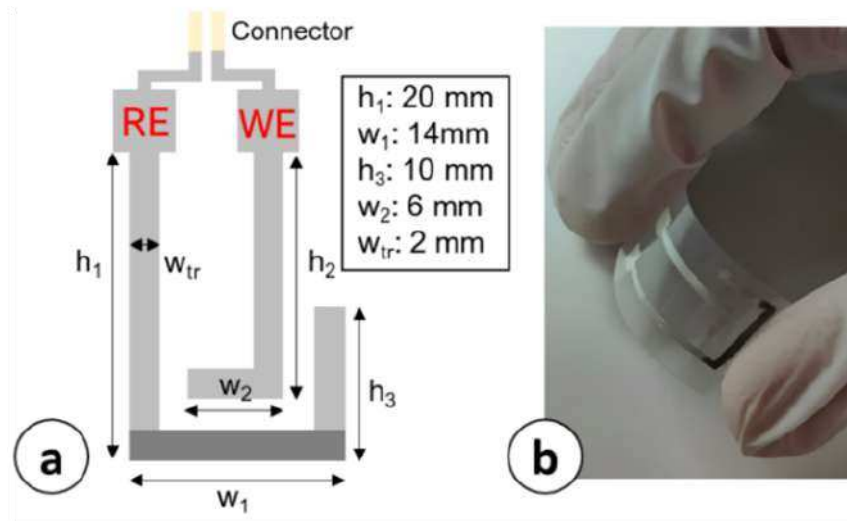


Figure 3.5: Proposed Electrodes. (a) Geometry and (b) electrodes snapshot [64]

In addition to material and geometrical considerations, electrode stabilization strategies are essential to mitigate drift and ensure reproducible measurements over time. Approaches such as optimized AgCl layer formation, surface treatments, and encapsulation techniques have been proposed to enhance durability and resistance to environmental variations. These design aspects are especially relevant in two-electrode potentiometric configurations, where reference stability directly impacts measurement reliability [53, 155]. The electrode materials and geometrical design adopted in this work are chosen to balance sensing performance, mechanical flexibility, and compatibility with embedded electronics.

3.2.4 Electronics design for potentiometric biosensing

The electronic architecture is reported to provide contextual completeness and to illustrate how the proposed sensing principle and electrode designs are interfaced with an existing embedded platform used for experimental validation.

Principle of Potentiometric Measurement

The role of the electronic subsystem is to support potentiometric chloride measurements by acquiring the open-circuit potential (OCP) generated between the working electrode (WE) and the reference electrode (RE) without perturbing the electrochemical equilibrium. Preserving this condition requires an electronic interface characterized by extremely high input impedance, ensuring negligible current flow and avoiding electrode polarization. The measurable quantity of the open circuit potential (E_{cell}) is defined as the potential difference between the working electrode (WE) and the reference electrode (RE) under zero-current conditions. The overall potential can be expressed as:

$$E_{\text{cell}} = E_{\text{WE}} - E_{\text{RE}} + E_J \quad (3.2)$$

where E_J represents the junction potential arising at the electrode–electrolyte interface. This configuration requires negligible current flow to maintain electrochemical equilibrium, thus the use of a **high-impedance readout stage** is essential to avoid charge leakage and to preserve the intrinsic ion–electrode equilibrium. Among electrochemical methods, potentiometry is particularly advantageous for wearable systems because it enables *direct quantification of ionic concentration* with low power consumption and without the need

for reagents or post-processing.

Readout Circuit Design

The analog front-end (AFE) of the proposed sensing platform evolved through several iterations, progressively improving signal quality, noise rejection, and integration on flexible supports. The Electronic interface is represented by three main blocks: the **High Impedance and Differential Stage**, the **Analog Filtering Stage** and the **Microcontroller**, the embedded processing and communication unit. This modular organization allows the sensing front-end to be decoupled from digital control and data transmission functionalities, facilitating scalability and reuse across different sensing configurations.

The *High Impedance and Differential Stage* readout stage buffers the electrode potentials V_{WE} and V_{RE} ensuring an input impedance in the order of hundreds of $G\Omega$. This stage plays a critical role in preserving measurement stability and repeatability, particularly in wearable scenarios where motion artifacts and environmental interference may be present. The differential configuration also contributes to improved common-mode noise rejection. Following signal buffering, an *Analog Low-Pass Filtering Stage* is employed to limit the signal bandwidth prior to analog-to-digital conversion. The filter characteristics are selected to preserve the slow temporal dynamics associated with ion diffusion and electrode equilibration, while attenuating high-frequency noise components that are irrelevant for potentiometric measurements. A *fourth-order Sallen–Key low-pass filter (LPF)* with a cut-off frequency of 200 Hz provided anti-aliasing and noise suppression before digitization has been applied.

The *Digital Stage* of the platform is implemented using an embedded microcontroller, which manages data acquisition, calibration routines, and communication tasks. In addition, the microcontroller coordinates the operating modes of the system, enabling transitions between active measurement and low-power states to support long-term wearable operation. Power-aware control strategies are therefore an integral part of the electronic support platform.

The electronic platform evolved through multiple validated implementations, progressively improving integration and compatibility with flexible substrates and sensing interface.

Within this framework, the electronic subsystem acts as an enabling layer that bridges the electrochemical sensing domain and the digital processing aspect. The original contributions of this work are focused on the sensing principle, electrode materials and geo-

metrical design, and experimental characterization, while the electronic implementation is employed as a validated support platform for experimental evaluation. The quantitative assessment of the electronic interface, including noise performance, stability, and power consumption, is presented in Section 5.2.

Chapter 4

EXplainable Artificial Intelligence for managing vital parameters in embedded platforms

The rapid evolution of intelligent systems has progressively redefined the boundaries between data acquisition, knowledge representation, and autonomous decision-making. The growing of interconnected devices has shaped what is known as the *IoT*, an ecosystem in which sensors, actuators, and embedded platforms continuously collect, exchange, and process information. Despite its success in enabling large-scale data availability, the *IoT* still suffers from the absence of a common semantic layer, which limits the interpretability and interoperability of the information generated by heterogeneous nodes.

To overcome these limitations, the *Semantic Web of Things (SWoT)* merges the pervasive connectivity of *IoT* systems with the semantic technologies of the Web [181]. By annotating data streams with ontological concepts and enabling reasoning over these descriptions, *SWoT* allows devices not only to communicate but also to understand and contextualize the exchanged information. This paradigm shift transforms data networks into knowledge networks, where meaning and reasoning become integral to communication.

In parallel, the evolution from the *IoT* toward the *IoE* has broadened the scope of interconnectivity from devices alone to people, data, and processes. *IoE* environments demand not only syntactic but also semantic interoperability, ensuring that exchanged information retains its meaning across heterogeneous systems and domains [168]. Semantic technologies, such as ontologies and linked data, are therefore essential to enable

context-aware interaction and reasoning among IoE components, particularly in large-scale scenarios such as smart cities. This transition highlights the need for a unified semantic infrastructure that can integrate physical, digital, and human elements into coherent and machine-interpretable ecosystems.

However, the continuous expansion of smart environments has revealed the need for a broader framework. The *SWoE* extends the semantic paradigm beyond “things” to encompass people, processes, and virtual entities within a unified ecosystem [181]. In the *SWoE* vision, every component—human or artificial—can expose machine-understandable descriptions, participate in reasoning activities, and contribute to collective intelligence. This transition moves from simple connectivity to semantic interoperability, enabling systems capable of autonomous adaptation and explainable behavior even under resource constraints.

This evolution calls for lightweight reasoning engines capable of operating directly on embedded nodes, performing inference locally while maintaining formal correctness and semantic coherence. The *SWoE* thus establishes the foundation for a new generation of explainable, distributed, and knowledge-driven AI systems, where intelligence is not centralized but diffused throughout the network.

In this perspective, the following sections explore how ML paradigms, initially confined to centralized cloud architectures, are progressively shifting toward edge and micro-scale platforms, and how semantic reasoning and explainability can enhance their interpretability and trustworthiness in critical embedded scenarios.

4.1 Related Work

Cloud computing has been the de facto paradigm for large-scale ML due to its elasticity and high computational power [156]. In this paradigm, data produced by distributed sensors or IoT devices are centralized for processing, training, and inference. Complex models are trained on high-performance servers and later deployed to lightweight nodes for inference [189]. Although this approach ensures scalability and global data integration, it also introduces several limitations: high communication latency, dependency on continuous Internet connectivity, vulnerability to data privacy breaches, and limited autonomy at the edge. Several studies have demonstrated the feasibility of cloud-based ML for healthcare and industrial monitoring tasks. In [7], the authors propose a framework for predicting

pregnancy risk that uses wearables enabled by IoT for collection of physiological data. The dataset is transmitted to a cloud platform, where a DT classifier performs risk analysis. The model achieves good predictive accuracy, but the system heavily relies on remote connectivity, lacks on-device reasoning, and does not support adaptivity. Similarly, cloud-assisted architectures in [156] show that offloading computational workloads to the cloud simplifies device design but undermines real-time responsiveness. Despite their wide adoption, cloud models are incompatible with many embedded and CPS applications where low latency and privacy preservation are crucial.

These limitations have driven the research community toward **edge and fog computing**, paradigms that relocate learning and inference closer to the data source to achieve distributed autonomy. Edge and fog computing extend the capabilities of IoT architectures by introducing intermediate processing layers between sensors and cloud servers [156]. These paradigms allow data to be preprocessed, analyzed, and sometimes used for model training directly at or near the device level, reducing latency and bandwidth consumption. Moreover, they mitigate privacy risks by limiting the exposure of raw data to remote servers. A representative example of edge-based inference is found in [149], where smart footwear equipped with inertial sensors and pressure transducers monitors foot orientation to detect abnormal postures and prevent falls. A Raspberry Pi processes sensor data locally using a heuristic algorithm, issuing alerts in real time. While the system demonstrates responsiveness and robustness in movement detection, it lacks adaptability and learning capability to new user behaviors. [160] extend this approach by comparing several ML models for fall detection, including SVM, RF, k-Nearest-Neighbors (kNN), and deep architectures. Their experiments indicate that the CNN-LSTM with an attention mechanism achieves superior accuracy and latency performance. Models are trained in the cloud and deployed on a Raspberry Pi 3, showing that DL can be executed on edge devices with moderate computational resources. However, the system still relies on pre-trained models and lacks continuous learning capabilities and exhibits computational limitations. [125] explore a DL-based fall detection approach validated on multiple datasets (MUMA, Sim-Fall, Notch, Smartwatch, SmartFall, UP-Fall). A CNN model runs inference on a Nordic nRF9160 MCU, achieving good accuracy, but the complexity of the model exceeds typical embedded constraints, and the explainability of the results is not addressed.

Overall, edge-based systems have shown that moving computation toward the data source improves latency and autonomy but still depends heavily on offline model training and heuristic tuning. Most TinyML applications adopt offline training followed by pruning

or quantization to fit hardware constraints [189, 173, 154].

To fill this gap, **on-device training** aims to achieve complete autonomy by executing both learning and inference locally on MCUs or low-power single-board computers [156]. This approach improves privacy, reduces latency, and enables real-time adaptation in environments with intermittent connectivity. The *AI/ES* framework [228] represents one of the first general-purpose open-source AI libraries for embedded devices, supporting both training and inference entirely on MCUs. Implemented in ANSI C, it provides a lightweight back-end for feedforward networks and CNN. Experiments on the nRF52840 and several open datasets demonstrate accuracy comparable to TensorFlow Lite while drastically reducing memory footprint. [203] extend this concept through the ML-MCU framework, which provides algorithms for both training and inference directly on MCUs. The framework supports gradient descent and stochastic gradient descent for binary and multi-class classification and is evaluated across multiple MCUs using four datasets. Results show that even with tight resource constraints, on-device learning remains feasible when lightweight optimizers and fixed-point arithmetic are applied. [15] propose a domain-specific solution for industrial anomaly detection that integrates edge computing, TinyML, and blockchain technologies. The system uses an ESP32 platform capable of training and inference in real time, while data are securely managed via blockchain at the edge/cloud layer. The anomaly detection algorithm is based on Isolation Forest and demonstrates strong performance in industrial scenarios, though it requires additional validation on diverse datasets. Similarly, [55] present an on-sensor activity detection framework using an STM Inertial Sensor Processing Unit (ISPU). The system performs real-time segmentation and template matching directly on the sensor, achieving 96.7% accuracy on six classes while consuming only 8 kB of memory. However, the approach is tailored to a specific domain and does not generalize to broader learning tasks.

These works collectively confirm the feasibility of on-device training and inference but also highlight a common limitation: once trained, models remain static. This motivates the exploration of incremental and lifelong learning paradigms that allow continuous updates and adaptability.

In this direction, **Incremental Learning** (IL) enables models to adapt over time by integrating new data without retraining from scratch, as explained in Section 2.2.2. This paradigm is particularly suited for embedded and IoT systems, where memory, energy, and data storage are limited. [67] implement an incremental RNN-based algorithm for real-time anomaly detection on a STM32 device using the X-Cube-AI SDK (<https://>

www.st.com/en/embedded-software/x-cube-ai.html). The model continuously updates its weights as new sensor data arrive, showing feasibility of online incremental learning on constrained hardware maintaining low inference latency. [197] apply incremental learning to heart rhythm classification. Their framework collects ECG signals through IoT sensors and transmits them to a Raspberry Pi that runs a RF model updated incrementally as new patterns are observed. The study confirms that incremental updates enhance classification accuracy by adapting to subject-specific physiological changes. [62] propose an on-device face recognition framework using a genetic algorithm to train a feed-forward NN directly on an Arduino Nano 33 BLE Sense with a camera module. Despite achieving 89.2% accuracy on the dataset, full on-device training remains computationally expensive, and memory limitations restrict scalability.

Edge2Train [202] implements on-device training, inference, and evaluation of SVMs across multiple MCUs. The framework achieves high accuracy in binary classification tasks, demonstrating energy-efficient local learning. The same authors introduced self-learning for streaming data, Train++ [204], reducing energy consumption per training iteration. A performance evaluation in [90] compares Train++ across GD32 (Cortex-M4/M23) and STM32 devices, revealing that while GD32-M4 offers faster training, STM32 achieves shorter inference times. The other MCU, GD32 featuring a Cortex-M23 processor, exhibits slower training and inference time compared to the STM32.

[76] develop an incremental learning approach combining pre-trained CNN feature extractors with kNN classifiers. Tested on Raspberry Pi 3 and STM32 boards for image and audio classification, the method allows continuous updates while preserving efficiency, though it depends on pre-trained features.

[8] introduces an example of Lifelong Incremental Learning (LIL), focusing on enabling models to learn continuously, adapting to new data without forgetting previously acquired knowledge: the *Poly-Instance Lifelong Learning (PILL)* algorithm for on-device training and inference demonstrates strong performance on single-board computers, balancing stability and plasticity, key properties for LIL (Section 2.2.2).

Federated Learning (FL) decentralizes model training by allowing multiple devices to collaboratively train a global model while retaining data locally. This approach preserves privacy, reduces bandwidth requirements, and supports heterogeneous edge environments (Section 2.2.3). However, most FL implementations still depend on central servers for model aggregation and communication [203, 229]. *TinyFL* [229] represents a fully decentralized on-device FL. It introduces a leader/follower protocol for MCU networks,

allowing devices to exchange model updates and aggregate results without relying on external servers. The framework demonstrates good accuracy but faces challenges related to energy consumption, data collection, and network stability. Other frameworks, including Fed4Edge [20], OpenFL [61], and FedAir [80], provide reproducible and scalable FL pipelines but primarily target single-board computers or PC-class devices. They typically assume sufficient memory for model aggregation and offer limited support for MCU-level deployment. Consequently, integrating full FL training and aggregation directly on MCU-class hardware remains a key open research issue.

In parallel to these learning paradigms, deployment on resource-limited devices is enabled by **model optimization** and **Neural Architecture Search (NAS)**. Model optimization for resource-limited devices often relies on pruning and quantization. However, NAS for MCUs is increasingly investigated, aiming to select the best type of ML model from a catalog and optimize its hyperparameters w.r.t. MCU memory constraints and on-board inference latency. *TinyNS* [184] introduces a neurosymbolic platform-aware NAS framework for TinyML. By means of a Bayesian black-box optimizer with platform-in-the-loop feedback, it can synthesize five different types of TensorFlow Lite architectures, combining symbolic code and NNs to fully exploit MCU resources, as well as to incorporate problem-specific rules, physical laws and heuristics. Fine-tuning is supported after deployment via on-device training and FL. Nevertheless, in *TinyNS* and other NAS frameworks the initial training and optimization tasks must be carried out offline, since search requires very large computational resources. *TinyNS* highlights the potential of neurosymbolic optimization but underscores the need for fully autonomous model evolution in the field.

Despite progress in TinyML, **eXplainability** has received limited attention and is often discussed only as a future direction [173, 154]. XAI aims to make ML models transparent and interpretable, which is especially critical for safety-critical domains like healthcare and industrial monitoring. Although FL allows decentralized learning, integrating XAI remains a largely open challenge, as global model aggregation makes individual predictions less interpretable. Due to resource constraints, most on-device FL systems prioritize efficiency over explainability. Nevertheless, integrating XAI with on-device IL and FL is crucial for real-world applications.

The ability to explain and interpret the results of a ML model is a significant advantage across various domains. For example, in healthcare, this capability allows practitioners to uncover critical patterns in patient data, leading to improved diagnosis and treatment

strategies [188].

The TinyML framework for healthcare monitoring in [16] includes explainability features. *Arduino Uno R3* gathers data from heterogeneous sensors and transfers them in a compressed encoding to a Raspberry Pi 3 at the fog level, which performs model training and inference with TensorFlow Lite. Post-hoc explainability is supported via Matthews correlation coefficient, SHAP [139] and Wasserstein distance between the features. Similarly, three TinyML models are compared in [98] for anomaly detection in vehicular networks, leveraging cloud datacenters for training and SHAP and LIME for post-hoc explanation.

On MCU/edge targets, deployment feasibility is largely dictated by memory footprint and inference time, and by whether the software stack provides optimized implementations for the required operators. For instance, TensorFlow Lite for MCUs is designed to minimize memory usage and inference latency, and it integrates MCU-optimized operator implementations (*e.g.*, CMSIS-NN) together with full 8-bit quantization [75]. From the perspective of the underlying predictive model, embedded DL deployments commonly rely on compact *CNN* architectures (also in temporal/1D form), lightweight recurrent models, or constrained Transformer variants. These exhibit different compute/memory profiles and different robustness to reduced precision; hence, model optimization techniques such as quantization and pruning are routinely used to make them deployable on resource-limited devices [5]. Recent edge benchmarks that evaluate post-hoc methods on lightweight models (*e.g.*, MobileNetV2, TinyBERT, and tree ensembles) show a cost-benefit trade-off: SHAP yields stronger explanations but incurs the largest latency and memory increase, whereas LIME typically provides a more favorable operating point for real-time settings; saliency maps remain relatively cheap, but are mainly applicable to vision models [10]. Consequently, selecting a suitable XAI-model combination is crucial for edge deployments, as practicality depends on both model architecture characteristics and hardware/toolchain limitations [10, 5]

Conversely, probabilistic Bayesian Networks are adopted in [131] as an explainable neurosymbolic approach for TinyML on a FPGA and on an ESP32 MCU, but unfortunately models are trained off-line before downscaling them to the target devices.

Moreover, integrating symbolic approaches in XAI is an active research field currently explored to improve the interpretability of *gray box* models [40]. This work is an evolution of *MAFALDA* (MAchmaking Features for mAchine Learning Data Analysis) [182], an IoT-oriented semantics-enhanced ML framework designed to integrate symbolic reasoning

into the typical ML pipeline to grant logic-based explainability. Although the original MAFALDA does not include IL and FL, a subsequent analysis of the core algorithms has suggested the possibility to add these features and has leveraged the framework in a serverless cloud-to-thing architecture for FL [136].

At the state of the art, however, explainable on-device AI is prevalently based on DT models, which combine efficient computation and relatively easy interpretability. The self-supervised system implemented in [186] on a NUCLEO-H743Z12 MCU uses an *Autonomous Edge Pipeline (AEP)* that periodically applies k-means clustering for pseudo-labeling, followed by either kNN or DT for training. Results indicate that DT outperforms kNN on one dataset, though an accuracy drop is observed on another dataset, and latency remains a challenge.

The decentralized FL approach in [198] tracking mental health conditions in psychiatric treatment exploits an on-device DT to predict the local model, while a global RF model selects the best-performing prediction model. Furthermore, blockchain-based encoding rules ensure model security. The *TinyDecisionTreeClassifier* C++ library has been developed in [117] for both training and deployment to the edge. The authors have evaluated the C++ library on three commercial MCUs, tested it on two open-source datasets, and demonstrated its use in two domain-specific applications.

In [28] a port of the *Tiny-ME* reasoner for the Apple Watch platform enables on-board reasoning for explainable and context-aware incremental analysis of physiological data within a Personal Health Knowledge Graph. MAFALDA 2.0 is a generalization of that approach, integrating sub-symbolic statistical techniques for ML model training and support for FL.

The comparative analysis clearly reveals that, although many proposals achieve 1 to 3 stars in the rating proposed in Section 2.3.3, very few integrate IL, FL, and XAI simultaneously, as summarized in Table 4.1. Notably, none of the analyzed works implement on-device symbolic XAI methods, except previous MAFALDA-based proposals.

The fact that no other work achieves full marks in the five-star rating model for on-device AI highlights a gap in the existing literature that needs to be filled: supporting both IL and FL enables dynamic adaptation to new data at various scales, while semantic explanation capabilities grant transparency in models and outcomes, for an overall greater trustworthiness. The proposed MAFALDA 2.0 framework addresses all identified requirements in the five-star rating:

Table 4.1: Related work comparison. ✓ Supported, ✗ Not Supported.

Ref.	Year	★ On-Device Training	★★ Incremental Learning	★★★ Federated Learning	★★★★ XAI	★★★★★ Symbolic XAI
<i>On-Device Training and Inference</i>						
[149]	2017	✗	✗	✗	✗	✗
[125]	2020	✗	✗	✗	✗	✗
[160]	2024	✗	✗	✗	✗	✗
[228]	2024	✓	✗	✗	✗	✗
<i>Incremental and Federated Learning</i>						
[76]	2020	✓	✓	✗	✗	✗
[202]	2020	✓	✓	✗	✗	✗
[204]	2021	✓	✓	✗	✗	✗
[203]	2021	✓	✓	✓	✗	✗
[67]	2022	✓	✓	✗	✗	✗
[197]	2022	✗	✓	✗	✗	✗
[90]	2023	✓	✓	✗	✗	✗
[15]	2023	✓	✓	✗	✗	✗
[229]	2023	✓	✗	✓	✗	✗
[62]	2024	✗	✓	✗	✗	✗
[8]	2024	✗	✓	✗	✗	✗
[184]	2024	✓	✓	✓	✗	✗
<i>eXplainable Artificial Intelligence</i>						
[182]	2019	✓	✗	✗	✓	✓
[186]	2021	✓	✓	✗	✓	✗
[20]	2022	✗	✗	✓	✓	✗
[61]	2023	✗	✗	✓	✓	✗
[198]	2023	✗	✗	✓	✓	✗
[80]	2024	✗	✗	✓	✓	✗
[117]	2024	✓	✗	✗	✓	✗
[136]	2024	✓	✗	✓	✓	✓
[16]	2024	✗	✗	✗	✓	✗
[98]	2025	✗	✗	✗	✓	✗
[87]	2025	✗	✗	✓	✓	✗
[131]	2025	✗	✗	✗	✓	✓
[27]	2025	✗	✓	✗	✓	✓
MAFALDA 2.0	2025	✓	✓	✓	✓	✓

- its efficient C-based implementation allows both training and inference to run entirely on-device, even with limited memory and computational resources;
- incremental learning is enabled by a compact internal data structure that supports iterative model updates from data streams, without the need to store the whole dataset;
- FL is supported by design, as the intermediate data structures can be exchanged and merged across devices and interpreted using a shared and common OWL ontology;
- logic-based inference underpinning both training and prediction supports explainability, providing interpretable characterizations of both the models and their results;
- the adoption of OWL for feature and label encoding enables these explanations to be aligned with standard dictionaries that are portable and understandable across heterogeneous nodes, thereby achieving semantic XAI within decentralized collaborative IoE systems.

4.2 Software architecture

Following the approach introduced in [182], MAFALDA 2.0 (*MA*tchmaking *F*eatures from *m*Achine *L*earning *D*ata *A*nalysis), combines both theoretical foundations and technological aspects of Pervasive Knowledge-Based Systems (PKBS). It relies on KRR techniques based on non-standard and non-monotonic inference services within DLs. The system performs automated reasoning to build an ML classifier optimized for resource-constrained environments, where data acquired from the physical world are enriched with semantic descriptions. Compared to [182], MAFALDA 2.0 improves both prediction accuracy and computational efficiency, enabling full on-device deployment. Moreover, it integrates logic-based explainability to make models and results interpretable to human users.

4.2.1 Framework

Figure 4.1 depicts the overall framework of the system. At its core, the architecture relies on an OWL-based Knowledge Graph (KG) employed for both domain modeling and data annotation. Ontology serialization, deserialization, and management are handled through *Cowl* [27], a lightweight C implementation of the OWL standard specifically optimized for

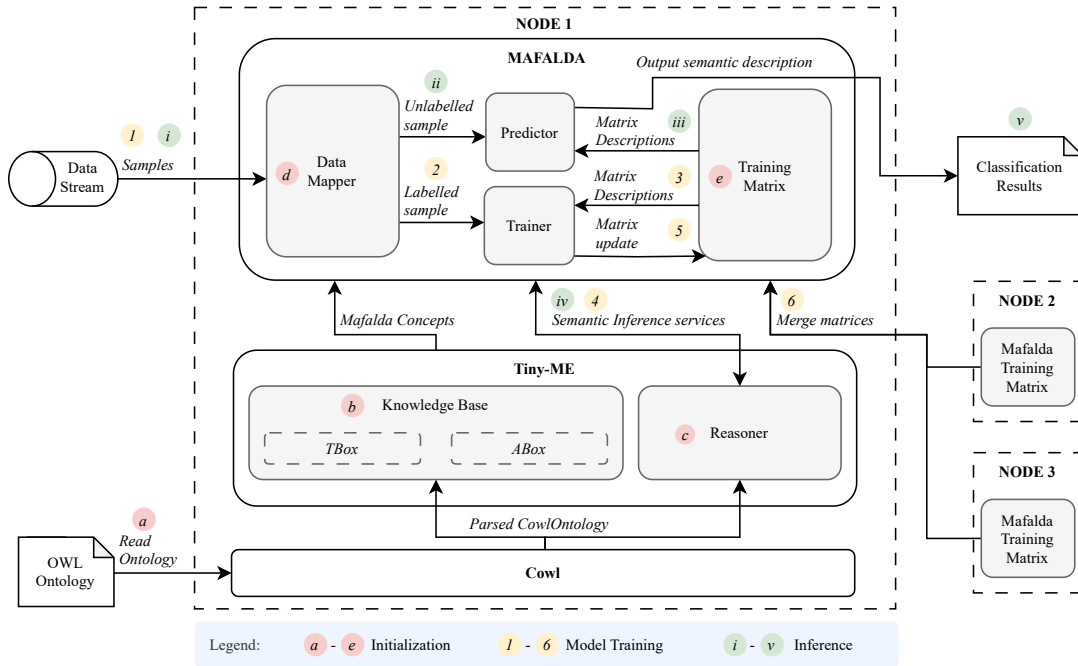


Figure 4.1: Proposed Framework Architecture. Markers indicate the steps of each process: (a-e) red markers for ontology-based knowledge preparation, where the domain knowledge is defined and the Tiny-ME reasoner is instantiated; (1-6) yellow markers for model training, where samples are annotated, the Training Matrix is updated and semantic descriptions of output classes are generated; (i-v) green markers for inference, which classifies test samples by means of semantic matchmaking and outputs the classification result along with its explanation. [135]

constrained platforms with minimal memory footprint. Reasoning services are provided by Tiny-ME [181] a lightweight engine supporting both standard and non-standard inference tasks.

MAFALDA 2.0 is implemented as a modular ML library that can be easily embedded into external systems. It exposes a standardized connector interface for integration with data streams, network sources, or file-based inputs, ensuring interoperability across heterogeneous environments. Thanks to this flexibility, the library can operate as a self-contained inference unit or as a component within larger distributed architectures.

The key concept underlying MAFALDA is the transformation of numerical samples into symbolic representations, reformulating the classification task as a problem of semantic matchmaking. The architecture includes four main components:

- **Training Matrix:** an n -dimensional data structure organized as a matrix, storing the

training dataset in a compact and aggregated representation. This structure maintains only the most representative descriptions due to practical limitations associated with storing all potential semantic descriptions of the reference dataset.

- **Data Mapper:** manages the one-way mapping between numerical samples and their symbolic counterparts, *i.e.*, semantic descriptions built with respect to the reference ontology.
- **Trainer:** during training, processes the semantic descriptions provided by the Data Mapper and updates the Training Matrix incrementally.
- **Predictor:** during inference, analyzes the semantic descriptions of new samples and assigns the most suitable output class.

The overall workflow of MAFALDA 2.0 follows a standard ML pipeline enhanced with ontology-based reasoning. As illustrated in Figure 4.1, the process unfolds in three main phases: ontology parsing and model initialization (red markers a–e), training (yellow markers 1–6), and inference (green markers i–v). The steps are detailed as follows:

- Initially, (a) the domain ontology is parsed using Cowl, producing a `CowlOntology` object; this object initializes (b) the KB and (c) the Tiny Matchmaking Engine (Tiny-ME) reasoner. Together, these components allow the instantiation of (d) the Data Mapper and (e) the Training Matrix (TM), whose dimensions are inferred from the KB. The ontology structure is described in Section 4.2.2.
- When a labeled sample is received from a data stream (1), the Data Mapper generates its semantic description (2). The Trainer integrates this description into the Training Matrix (3), using reasoning mechanisms (4) to determine how the matrix should be updated (5). MAFALDA 2.0 also supports FL (6), enabling the aggregation of multiple Training Matrices. This workflow supports incremental learning by processing one sample at a time.
- During inference, a new sample (i) is converted into a semantic description (ii) and passed to the Predictor module (iii). The Predictor exploits reasoning services (iv) over class descriptions to determine the final classification output (v), which includes both the assigned label and a symbolic explanation of the decision. This phase is explained in Section 4.2.3.

4.2.2 Ontology schema and data annotation

Unlike conventional ML techniques that rely on numerical feature vectors, MAFALDA employs ontology-based Knowledge Representation (KR) to encode raw data into formal concept descriptions. This integration enhances the interpretability of the resulting model, as discussed in Section 4.2.3. The ontology $\mathcal{T}$ defines the conceptual hierarchy and semantic relationships of the target domain, ensuring that each feature extracted from the dataset corresponds to a well-defined concept within the ontology.

To ensure compatibility with the framework, MAFALDA 2.0 introduces a set of ontology modeling patterns that must be followed when designing new ontologies or adapting existing ones. Reserved OWL entities are identified through the namespace:

`mafalda:` <http://swot.sisinflab.poliba.it/mafalda#>

For generality, the notation C^x is used to denote any named `owl:Class`, either categorical (D^x) or numerical (N^x). After mapping, both are handled uniformly within MAFALDA 2.0. The $\mathcal{T}$ can be derived from expert knowledge, standardized vocabularies, specification documents, external Knowledge Graphs (KGs) or Linked Data sources [226], or obtained via data-driven approaches such as clustering and feature selection [208, 167], especially in cases where domain ontologies are unavailable or costly to engineer. Regardless of its origin, an ontology that adheres to the MAFALDA 2.0 modeling guidelines ensures correct parsing and reasoning, enabling the mapping of both labeled and unlabeled samples into $\mathcal{ALN}$ concept descriptions. This process resembles feature discretization through binning, but with the added benefit of formally defined semantics provided by standard OWL constructs.

A labeled sample (training instance) is defined as a pair $\langle \mathbf{x}, \mathbf{y} \rangle$, where $\mathbf{x} = [x_1, \dots, x_n]$ represents the feature vector and $\mathbf{y} = [y_1, \dots, y_m]$ the output label vector. Conversely, an unlabeled sample (used for prediction) is represented as $\langle \mathbf{x} \rangle$. During training, the Trainer module incorporates each received labeled sample into its internal data structure, incrementally constructing domain knowledge from observed instances. During inference, a.k.a. when an unlabeled sample is received, the Predictor exploits this learned knowledge to infer the most appropriate label assignment.

Each feature $x_i \in \mathbf{x}$ may be numerical or categorical. A numerical feature consists of a name, corresponding to the ontology class representing that feature, and a numeric value, which is mapped to the most specific subclass whose interval includes it, based on the annotation properties `mafalda:minValue` and `mafalda:maxValue`. If either boundary

is undefined, the value is treated as unbounded. A categorical feature includes a feature name and category name, mapped directly to an ontology class D^x matching the dataset label. Each output label $y_i \in \mathbf{y}$ is categorical, since MAFALDA 2.0 performs classification; it is mapped to the corresponding subclass C^y under the `mafalda:Output` hierarchy. In accordance with the Open World Assumption (OWA), missing feature values are simply left unmapped, meaning they can assume any possible value within the ontology.

Through this process, each sample is annotated as a semantic description expressed in $\mathcal{ALN}$, represented as a conjunction of ontology concepts. Formally:

$$S \equiv S^x \sqcap S^y \equiv C_1^x \sqcap \dots \sqcap C_f^x \sqcap C_1^y \sqcap \dots \sqcap C_l^y$$

where S^x corresponds to the semantic description derived from the feature vector $\mathbf{x}$ and S^y to that derived from the label vector $\mathbf{y}$. The mapped sample is then used for model training if labeled, or for inference if unlabeled, in which case $S^y \equiv \top$.

4.2.3 Inference and explainability

The classification process in MAFALDA 2.0 relies on the TM generated during the training phase. Given a new unlabeled input, its semantic description is computed according to the ontology-driven mapping procedure detailed in Section 4.2.2. The resulting description is then compared with those stored in the TM, which serve as reference profiles for classification. Explainability in MAFALDA 2.0 arises from two complementary aspects: (i) the symbolic representation of the trained model, *i.e.*, the semantic descriptions associated with the output classes within the TM, and (ii) the reasoning outcomes produced during the semantic matchmaking process, recalled in Section 2.2.4.

Semantic descriptions are generated from the TM by constructing a symbolic profile for each output class. Let $\mathbf{M}$ denote the TM; the semantic description corresponding to its k -th row, R_k , is computed through Algorithm 1. The procedure is governed by a threshold parameter $\theta \in (0, 1]$, which specifies the minimum normalized contribution $\hat{M}_{k,j}$ required for a feature concept to be included in R_k . Higher θ values produce more compact and selective description, while lower values allows more concepts to be included. The optimal value of θ depends on the specific classification task and typically requires optimization. Algorithm 1 initializes an empty description R_k and iterates over all feature columns j , computing the normalized matrix value $\hat{M}_{k,j}$. Whenever $\hat{M}_{k,j} > \theta$, the corresponding

Algorithm 1: Generate a semantic description for the k -th TM row

Input: k (index of the row to map to a semantic description)
Data: M (Training Matrix k -th row), θ (threshold)
Result: R_k (semantic description of the k -th row)
 $R_k \leftarrow \top$
for $j = 1$ **to** F **do**
 $\hat{M}_{k,j} \leftarrow \frac{M_{k,j}}{\sum_{j' \in C_j^{\mathcal{F}}} M_{k,j'}}$
 if $\hat{M}_{k,j} > \theta$ **then**
 $C^x \leftarrow \text{getConcept}(j)$
 // Store the concept in a temporary semantic description

 $R_{tmp} \leftarrow C^x$
 $\langle G, K \rangle \leftarrow \text{conceptContraction}(R_{tmp}, R_k)$
 if $G \equiv \top$ **then**
 $R_k \leftarrow R_k \sqcap R_{tmp}$
 else if $\text{weight}(k, G) \leq \text{weight}(k, R_{tmp})$ **then**
 $R_k \leftarrow K \sqcap R_{tmp}$
 return R_k

ontology concept C^x is retrieved and stored temporarily in description R_{tmp} . A Concept Contraction operation is then executed between R_{tmp} (acting as the *resource*) and the current description R_k (the *request*), producing the pair components $\langle G, K \rangle$ as defined in Section 2.2.4. The algorithm selectively updates R_k according to the following rules:

- If $G \equiv \top$, the temporary description R_{tmp} is fully compatible with R_k and can be directly appended;
- Otherwise, if the weight of the incompatible part G is lower than that being added (R_{tmp}), R_k is updated as $K \sqcap R_{tmp}$, discarding the less relevant incompatible portion G .

The weight of a semantic description R in the k -th row of TM M is defined as the maximum normalized matrix value among the concepts in R :

$$\text{weight}(k, R_k) = \max_{C^x \in R_k} \hat{M}_{k,j} \quad (4.1)$$

where:

- C^x represents a concept in the semantic description R_k .
- j is the column index corresponding to C^x in row k of the training matrix M .
- $\hat{M}_{k,j}$ is the normalized training matrix value for row k and column j .

If a concept does not appear in the row, its contribution is set to zero. After processing all columns, R_k represents the final semantic description of that row.

To classify a new sample, a semantic similarity measure is required between its description and those generated from the TM. The adopted metric combines the reasoning services recalled in Section 2.2.4 with the Conjunctive Normal Form (CNF) Norm, where for a semantic description D , $\|D\|$ denotes a real-valued length proportional to the number of concepts and restrictions [180]. The overall penalty score is defined as:

$$s(S^x, R_k) = \left(\frac{\|G'\| + \|H'\|}{2 \|S^x\|} \right) + \left(\frac{\|G''\| + \|H''\|}{2 \|R_k\|} \right) \quad (4.2)$$

The first term, the descriptions $\langle G', K' \rangle$ and $\langle H' \rangle$, corresponds to the application of Concept Contraction and Concept Abduction (Section 2.2.4) with S^x as the request and R^k as the resource. The numerator $\|G'\| + \|H'\|$ quantifies the missing or incompatible information between the two descriptions, normalized by $2\|S^x\|$. If the descriptions are completely disjoint, $\|G'\| + \|H'\| = \|S^x\|$ and the term equals 0.5; if fully compatible, it becomes zero. The second term is defined analogously, but with the arguments S^x and R^k reversed to capture the non-commutativity of the reasoning inference operators, CC and CA. The resulting score $s(\cdot, \cdot) \in [0, 1]$, where 0 indicates a perfect match and higher values correspond to increasing semantic dissimilarity. During the training process, this score is used to determine which label description in the TM should be updated.

When a new unlabeled sample $S \equiv S^x$ arrives, the classification workflow proceeds as follows:

1. **Iterate over each label superclass $\mathcal{L}$** , corresponding to the subclasses of **MAFALDA:Output** defined in Section 4.2.2;
2. **Retrieve all k -th rows of the TM** associated with all the subclasses C^y of $\mathcal{L}$;
3. **For each candidate description R_k :**
 - (a) Generate the semantic description R_k using Algorithm 1;

- (b) Compute the semantic score between the feature S^x and the generated row R_k ;
- 4. **Select the predicted subclass** C^y corresponding to the row with the lowest penalty score, and add it to S^y .

After the procedure, S^y contains the predicted label(s) for S^x . This workflow generalizes the approach proposed in [182], extending it to multi-class and multi-label classification problems.

The algorithm ensures both global and local explainability. The semantic description R_k captures the knowledge structure derived from the dataset, where the relative importance of each concept can be inferred from its normalized value within the TM. During prediction, the intermediate descriptions G' , K' , G'' , and K'' provide explanations of the differences between the input sample S^x and the internal model representation R_k : G' and G'' identify incompatible components, while H' and H'' highlight missing concepts required for a full semantic match.

Since this reasoning process is repeated for all R_k , it supports both model interpretability and debugging, offering not only justifications of decisions but also contrastive and counterfactual explanations [48]. The former explain why a sample is not assigned to a given class, while the latter indicate which feature changes would alter the classification outcome.

4.3 Explainability properties

MAFALDA 2.0 achieves explainability by encoding domain knowledge through symbolic representations based on OWL, and by expressing classification as a semantic matchmaking process (see Section 4.2.3). Every inference result is accompanied by a symbolic justification derived from the activation of ontological assertions and logical rules, allowing the reasoning process to be both transparent and reproducible. This approach supports interpretability at multiple levels: globally, by exposing the learned semantic structure, and locally, by explaining individual predictions according to the specific features involved.

Since MAFALDA 2.0 operates on structured ontological knowledge, the resulting descriptions can become complex when multiple features and relationships are involved. To maintain readability, the system introduces a *cutoff parameter* that filters concepts whose TM value is below a selected threshold before rendering them as natural language

strings. This mechanism enables the user to control the balance between interpretability and cognitive accessibility: a lower cutoff preserves the full reasoning trace and thus yields exhaustive, highly faithful explanations, while a higher cutoff produces more compact and easier-to-read explanations. The cutoff parameter also provides a practical way to evaluate qualitative XAI dimensions, *e.g.*, fidelity, stability, consistency, sparsity, and user comprehensibility [152], by observing how variations in cutoff affect the resulting explanations. A higher cutoff value favor sparsity, *i.e.*, more compact, stability, *i.e.*, whether similar explanations are generated for similar inputs, and more cognitively accessible explanations, *i.e.*, improving user comprehensibility by reducing conceptual complexity in more compact explanations. However, fidelity, *i.e.*, how accurately the method approximates the ground truth, and consistency, *i.e.*, whether multiple trained models provide similar explanations given the same data, may be reduced, as some relevant concepts might be excluded. Conversely, a lower cutoff that captures all concepts and produce longer semantic descriptions involved in inference emphasize fidelity, as the mirror of the internal symbolic inference process, and logical completeness.

Based on this theoretical framework, the explainability mechanisms implemented in MAFALDA 2.0 can be mapped onto the nine explanation types defined by Chari *et al.* [49]. Table 4.2 summarizes how each explanation category is supported within the system.

Table 4.2: Alignment between the explanation types from [49] and the explainability mechanisms implemented in MAFALDA 2.0.

Explanation Type	Support in MAFALDA 2.0	Description and Motivation
Case-based	<i>Potential</i>	It could be implemented by storing previously analyzed samples. MAFALDA 2.0 alone does not provide case-base explanation; however, the Knowledge Base of the trained model could be enriched with additional individuals representing past cases.

Continued on next page

Table 4.2 (continued)

Explanation Type	Support in MAFALDA 2.0	Description and Motivation
Contextual	<i>Supported</i>	The ontology includes contextual knowledge (<i>e.g.</i> , measurement conditions, environmental parameters, or user-specific constraints) that directly influences reasoning and can be reflected in explanations.
Contrastive	<i>Supported</i>	Implemented through the comparison of alternative outputs: why a certain classification (<i>e.g.</i> , <i>A</i>) was chosen instead of another (<i>B</i>) based on activated features and semantic conditions. Contrastive explanations in MAFALDA 2.0 basically correspond to the output of <i>concept contraction</i> : the <i>Give up G</i> concept pinpoints the conflicting requirements, explaining why outcome <i>A</i> was obtained instead of <i>B</i> .
Counterfactual	<i>Supported</i>	Generated by identifying which input features would need to change to produce a different outcome. In MAFALDA 2.0, counterfactual explanations are realized through <i>concept abduction</i> . In fact, it yields which missing features should be added or changed for a sample to reach a <i>full match</i> .
Everyday	Not supported	Explanations are symbolic and formal rather than narrative; however, simplified linguistic template-matching or Large Language Models (LLMs) could be used to verbalize explanations in natural, common-sense language [56].

Continued on next page

Table4.2 (continued)

Explanation Type	Support in MAFALDA 2.0	Description and Motivation
Scientific	<i>Potential</i>	Could be extended by linking explanations to external domain knowledge, such as scientific standards, ontologies, or provenance metadata that justify the reasoning process. It justifies a result by explicitly referencing scientific relationships, evidence, or methodological principles. At present, MAFALDA 2.0 does not include such references; however, OWL annotation properties could be used to link concepts to scientific sources, domain standards, or references retrieved from input ontology, thereby enabling provenance-based scientific justification.
Simulation-based	<i>Not applicable</i>	Uses an imagined or implemented simulation of a system or process and the results that emerge from similar inputs. MAFALDA 2.0 performs logical reasoning based on ontological models rather than process simulations; thus, this explanation type cannot be applied.

Continued on next page

Table 4.2 (continued)

Explanation Type	Support in MAFALDA 2.0	Description and Motivation
Statistical	<i>Not Supported</i>	Statistical reasoning can complement symbolic explanations (<i>e.g.</i> , frequency of concepts, similarity metrics). Although MAFALDA 2.0 internally uses numerical thresholds (<i>e.g.</i> , TM) to guide semantic matching, these scores are not exposed in the explanation at the moment. Since no numerical evidence (probabilities, frequencies, confidence) is communicated to the user, the system does not provide <i>statistical explanations</i> in the sense of Chari <i>et al.</i> This could become possible if similarity/confidence scores per class were surfaced and integrated into the justification.
Trace-based	<i>Not supported</i>	Every explanation reflects the logical reasoning trace of activated rules and ontological assertions, providing full transparency of the inference process. MAFALDA 2.0 can list the symbolic justification (<i>e.g.</i> , concepts H' , H'') but does not yet expose a full step-by-step reasoning trace nor provenance metadata. By integrating a provenance layer (<i>e.g.</i> , PROV-O) and logging activated axioms/rules and their order, the system could provide full <i>trace-based</i> explanations.

As shown in Table 4.2, MAFALDA 2.0 natively supports explanation types grounded in semantic reasoning and logical transparency, namely *contextual*, *contrastive*, and *counterfactual* explanations. Other categories, such as *case-based*, *everyday*, *scientific*, *statistical*, and *trace-based*, could be incorporated through additional components which have not been developed yet. In contrast, *simulation-based* explanations are not directly applicable due to the declarative, ontology-driven nature of the system. In MAFALDA 2.0, numerical quantities (*e.g.*, TM, thresholds) are used internally in *semantic matchmaking* process but

are not communicated to the end user; therefore, they do not constitute *statistical explanations* as defined by Chari *et al.* [49]. These directions could inspire future research aimed at extending the framework with modules for case-based reasoning, provenance tracking, dynamic simulation, natural language generation, or external scientific sources, thus expanding the range of explainability dimensions addressed.

Chapter 5

Experiments

5.1 Hazard analysis in autonomous safety-critical systems

This section evaluates the temporal response of the Vital Control Module under a set of representative hazardous scenarios derived from the hazard analysis presented in Section 3.1, with particular focus on intervention times and safety-related deadlines.

The VCM has been successfully validated in hardware-in-the-loop testbeds that have been engineered to simulate 36 distinct real-life scenarios, encompassing both nominal and hazardous operational states. Of these, 23 scenarios triggered hazardous events requiring intervention. To provide a complete overview of the VCM intervention capabilities, 6 out of 23 hazardous scenarios are selected and presented here. The selected scenarios are representative of the main hazard classes identified in Section 3.1.2 and are used to assess the temporal behavior of the system with respect to safety-critical intervention deadlines.

5.1.1 General use case and scope of the experimental campaign.

The VCM is a safety-related railway module conceived for operation within a RAMS-driven development process. In the railway domain, the preliminary safety assessment and the definition of verification activities are framed by EN 50126, which defines the RAMS lifecycle and the rationale for risk-based engineering decisions, including acceptability criteria and traceability requirements across the product lifecycle [44]. In this context, the goal of the experimental campaign is not to re-derive hazards, but to validate that the VCM implementation satisfies the intervention-time constraints implied by the hazard-to-requirement mapping produced by the norm-compliant analysis.

From a system-interaction perspective, the scenarios exercised in the testbed are centered on the supervision perimeter of the VCM within the ATO over ETCS architecture. In line with the supervisory role described in Section 3.1.1, the most relevant interacting entities include: (i) the onboard ATO traction/control subsystem (ATO-OB), (ii) the remote operator decision chain (trackside control room), (iii) the onboard odometry/speed-position subsystem (ATO-ODO), and (iv) the onboard ETCS status and procedure layer. In fully automated conditions, hazardous situations are frequently associated with the inability to validate the operational status of one or more of these subsystems, inconsistencies in their operating modes, or violations of speed/supervision constraints when ETCS is isolated. Across these situations, the safety action enforced by the VCM is the activation of Emergency Braking (EB), possibly followed by a non-recoverable safe state (System Decay), depending on the hazard class [147, 12].

5.1.2 Interaction-driven cases

A key purpose of the experimental design is to ensure that the selected verification stimuli cover not only single-fault conditions but also interaction-driven cases, *i.e.*, combinations of subsystem states that are individually meaningful but jointly hazardous. For instance, intervention is required when traction control is not in a valid/vital state while odometry indicates train motion, as well as when odometry supervision is degraded/out-of-service and ETCS status cannot be confirmed in time. Similarly, when ETCS is isolated, the VCM must enforce a conservative supervision profile (including the 30 km/h limit adopted in the validation setup), and must trigger EB if the limit is violated. These interaction patterns are explicitly reflected in the test scenarios used to measure intervention times, and they constitute representative “logic paths” that operationalize the hazardous conditions identified during hazard analysis into concrete, testable situations [147, 12].

Specifically, the best and the worst scenario of each above analyzed category is reported in Figure 5.1LEFT and briefly described in what follows. **Scenario A** involves a direct EB request sent by the remote operator through the ATO on-board system. In **Scenario B**, the train’s speed profile is supervised by the odometer algorithm of the VCM, since the ETCS is isolated. In this configuration, the VCM authorizes traction up to a maximum speed of 30 km/h. If the train exceeds this limit, the system automatically triggers the EB command. **Scenario C** refers to the supervision of the ATO on-board operating mode by the VCM. When a malfunction of the ATO on-board is detected, the system activates the EB and

transitions into the System Decay state, in which operation recovery is not permitted. In **Scenario D**, the VCM does not receive an active vigilance signal from the remote operator within the predefined time window. Consequently, the VC initiates an emergency braking followed by the transition to the System Decay state. When the VCM detects that the train is stationary, it queries the remote operator regarding the isolation or activation of the ETCS. This is carried out through a dedicated radio relay command. However, a relay failure may occur, preventing the state from toggling – this defines **Scenario E**. Since the train is already in standstill, EB activation is unnecessary, but the System Decay state is still applied. Finally, once the train has stopped after an emergency braking and the VCM has verified all safety conditions, the EB rearm procedure can begin according to the recovery routine. This process requires a confirmation from the remote operator via periodic data exchanges. If a reply is received but the message does not conform to the required format (*i.e.*, it is invalid), the VCM sets the System Decay state, completing **Scenario F**.

5.1.3 Experimental evidence: signal-level interpretation of scenario execution

Beyond aggregating end-to-end intervention times, the validation has also been interpreted at signal level to relate the measured timing to the activation of the underlying safety logic. In the testbed traces, the mitigation output corresponds to the EB triggering signal, while the remaining traces represent the status of the monitored inputs that drive the intervention logic (*e.g.*, speed-limit violation, standstill condition, odometry availability, traction subsystem vitality/failure modes and operating mode, remote vigilance signal, ETCS status availability and isolation state). In this interpretation, a hazardous scenario is observed when the required combination of input conditions is satisfied; conversely, isolated changes of one or few inputs do not necessarily imply hazard activation.

This view is useful for two reasons. First, it makes explicit that hazardous behavior is often caused by coupled conditions rather than by a single abnormal signal: EB is triggered only when the complete set of predicates describing a hazard scenario becomes simultaneously true within a given operational context. Second, it reveals that different hazardous scenarios may overlap in time (or share a large portion of their enabling conditions). In practice, during the same observation window, multiple scenario definitions can become true concurrently, all requiring the same mitigation action. This overlap does not weaken the safety argument, but it has consequences for verification planning: when two scenar-

ios are always co-activated under the same enabling conditions, their separate verification may be partially redundant, and evidence collected for one scenario may already cover the other.

5.1.4 Redundancy, scenario coverage, and implications for verification effort

The signal-level analysis and the scenario-based execution provide evidence that, within the considered supervision perimeter, a subset of interaction patterns is sufficient to guarantee complete coverage of the mitigation-triggering behavior. In other words, while hazard analysis intentionally enumerates hazardous situations conservatively, experimental observation confirms that some scenarios are structurally subsumed by others because they share the same necessary enabling conditions and they always occur together in the same operational windows. This motivates a distinction between (i) *essential* hazard scenarios/requirements that are necessary to ensure full safety coverage (*i.e.*, they alone are sufficient to trigger all required mitigations under all hazardous conditions), and (ii) *secondary* scenarios that refine or restate already covered conditions and can be considered redundant from the strict coverage standpoint.

A practical benefit of this distinction is the possibility to reduce the number of test objectives that must be treated as strictly independent for coverage completeness. In a V-model verification workflow, this can simplify the construction of a minimal yet complete verification set: essential scenarios define the non-negotiable safety coverage targets that must be preserved across the entire system lifetime, whereas secondary scenarios can be retained as robustness checks, diagnosability refinements, or maintainability aids, but do not necessarily increase safety coverage proportionally to the effort they require.

The analysis also highlights that some combinations of subsystem states, even if they appear in traces within the same observation window, may be logically non-compliant (*e.g.*, contradictory mode declarations or mutually exclusive status predicates). Detecting these inconsistencies is valuable because it can expose hidden assumptions in the interface contracts among subsystems and can motivate the introduction of explicit priority rules or additional constraints in subsequent design iterations. From the standpoint of the VCM validation, these observations reinforce the importance of treating the verification scenarios not only as timing benchmarks, but also as vehicles for checking the internal consistency of the supervision logic and of its assumptions on upstream subsystem behavior [147, 12].

The experimental findings have demonstrated that the VCM exhibits the capacity to initiate emergency braking within 540 milliseconds in the most unfavorable scenario, which is nearly half the 1-second maximum limit stipulated by the ERTMS/ETCS SUBSET-041 performance criteria [210, 147, 12]. Figure 5.1 shows the architecture intervention times in the analyzed scenarios. Specifically, Figure 5.1a shows the intervention time as the sum of maximum delay in receiving the response (Data Exchange Time), VCM elaboration time and the time needed to depressurize the pipe (ATO-EB Intervention Time). Figure 5.1b expands the data exchange and elaboration time, regardless from the EB intervention. Moreover, empirical evidence substantiates the VCM's ability to attain mitigation approximately two times faster than conventional ATP systems, while maintaining congruence with prevailing ETCS and national signaling infrastructures [12].

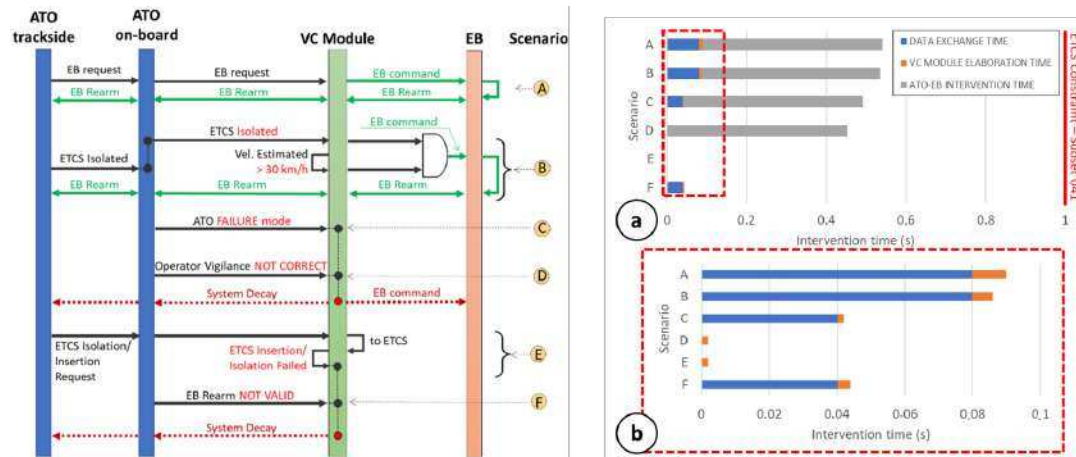


Figure 5.1: LEFT: Hazardous scenarios intended to lead VC module in EB and/or Decay status. RIGHT: VC module intervention times assessed on the selected scenarios.

In conclusion, the application of hazard analysis according to IEC 61508 and EN 50126 has been instrumental in shaping the VCM design. The integration of HAZOP with experimental validation has been demonstrated to be a comprehensive approach for the identification of all critical hazards.

This methodical process involves the mapping of these hazards to system requirements and the subsequent verification of those requirements through test scenarios. The findings indicate that the VCM not only complies with existing safety standards, but also meets them and in the tested scenarios performs within tight margins, facilitating a timely response to hazards and ensuring safe operation under GoA4 conditions [147, 12]. This case study demonstrates the indispensability of monitoring vital parameters and integrating hazard

analysis throughout the lifecycle for ensuring the safety and reliability of fully automated railway systems.

5.2 Biosensing

5.2.1 Electrodes

The earliest prototype demonstrated a pseudo-Nernstian behavior with a calibration slope of approximately 53 mV/dec. The measured open circuit potential (OCP) increased logarithmically with chloride concentration in the range 5–100 mM, confirming the expected theoretical trend. This prototype validated the feasibility of chloride detection through a flexible electrode configuration, although stability over extended measurements has been still limited. The electrodes of second generation achieved a sensitivity of 57.4 mV/dec, remarkably close to the ideal Nernstian slope (59 mV/dec), within the 1–75 mM range. The measured response time averaged 2.7 ± 0.6 min, ensuring suitability for on-body sweat analysis.

The calibration procedure of the last realized electrodes has been performed by sequentially drop-casting *NaCl* aqueous solutions at concentrations of 1 mM, 5 mM, 10 mM, 50 mM, 75 mM, and 100 mM over the electrode surface. Each droplet ($\approx 50\mu\text{L}$) has been allowed to stabilize, and the OCP has been recorded once the potential variation fell below ± 2 mV for at least 5 minutes. The resulting dose–response curve exhibited the expected logarithmic trend with a sensitivity of 56.9 mV/dec, consistent with the theoretical Nernstian slope. The complete calibration curve is reported in Figure 5.2, where each plateau corresponds to a stabilized chloride concentration. To assess stability and reproducibility, the potential difference has been monitored continuously for 10 minutes at constant concentration. The recorded deviation has been below ± 2 mV, confirming minimal drift. Additionally, statistical validation using the Shapiro–Wilk test demonstrated a normal distribution of the OCP data, supporting measurement repeatability and ruling out transient artifacts due to electrode polarization or environmental noise. The measurement accuracy has been concentration-dependent: ± 0.6 mM around 10 mM, ± 2.1 mM around 50 mM, and ± 3.4 mM near 100 mM NaCl. These values represent a significant improvement over the 2023 and 2024 prototypes, which exhibited both higher noise and wider potential drift. The reduced variability is attributed to the improved passivation layer and to the reproducibility of the controlled immersion process. Overall, the third-generation elec-

trodes combined mechanical flexibility, electrochemical stability, and biocompatibility, ensuring compatibility with wearable sweat analysis applications. Their near-Nernstian sensitivity, low noise, and drift-free operation make them ideally suited for integration into autonomous CF diagnostic platforms.

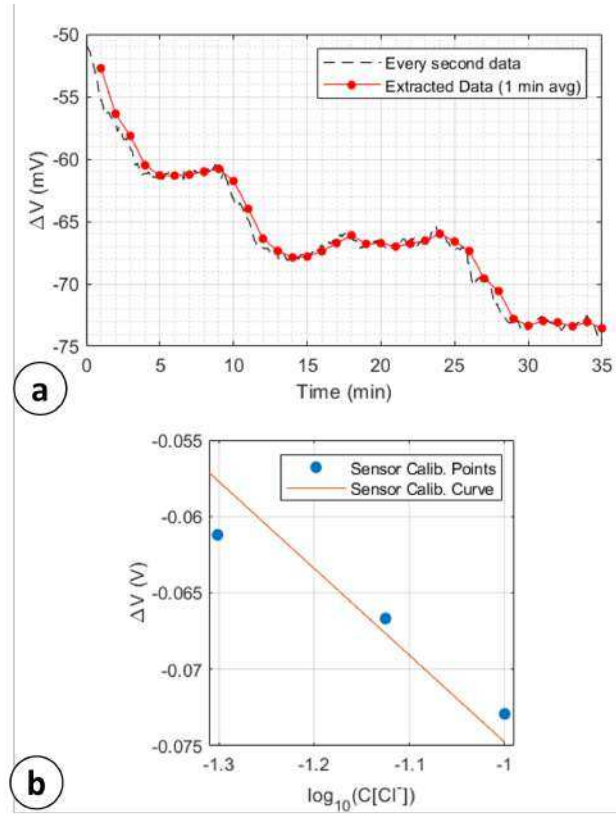


Figure 5.2: Electrodes Calibration. (a) Dose response curve and (b) Extracted Calibration Curve limited to the pathological range 50 mM -100 mM. Data extracted with the custom proposed electronics.

5.2.2 Electronics

Readout Circuit Design

As described in the Section 3.2.4, the Electronic Interface is represented by three main blocks: the **High Impedance and Differential Stage**, the **Filtering Stage** and the **Microcontroller**. The first and the second stage have remain the same in the develop of the device. The first version of the circuit [65] integrated the ATXmega32E5 microcontroller

embedding a 12-bit ADC (300 kps) and a DAC for offset control, enabling dynamic centering of the acquired signal within the range of ADC. The equivalent input-referred noise has been measured at $51.2 \text{ nV}/\sqrt{\text{Hz}}$, and the linearity analysis demonstrated an integral non-linearity (INL) below $400 \text{ } \mu\text{V}$, as illustrated in Figure 5.3.

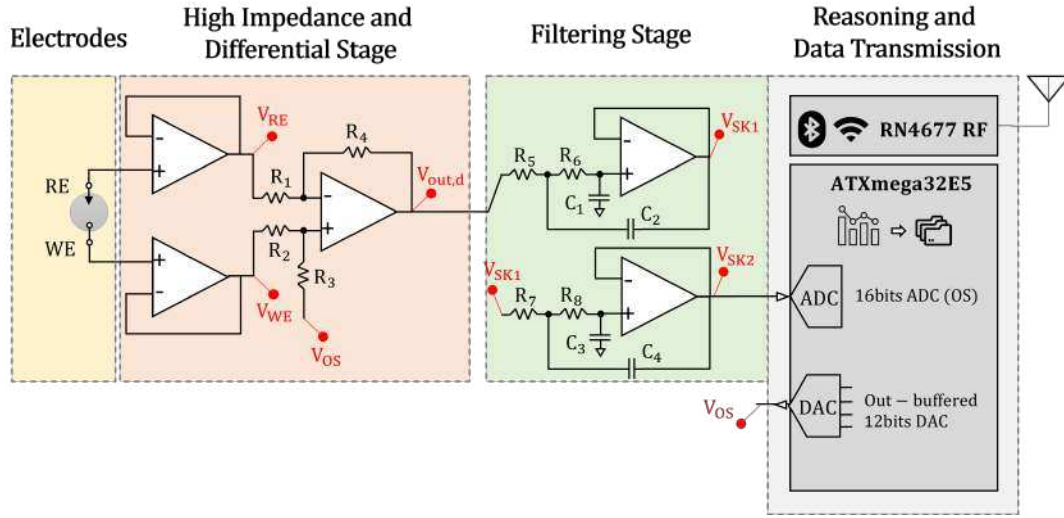


Figure 5.3: Chloride Sensitive Potentiometer Overview. (a) Block diagram of proposed architecture; (b) Simplified architecture of the first-generation potentiometric readout circuit with HI&D stage, differential amplifier, and 4th-order LPF [65].

In the second version of the device [63], the readout circuit has been redesigned on a **miniaturized flexible PCB** ($49.4 \text{ mm} \times 39.5 \text{ mm}$, polyimide substrate). The AFE employed low-noise operational amplifiers (MAX44242 and MAX4477 by Analog Devices) characterized by input bias current below 0.5 pA and input offset voltage below $70 \text{ } \mu\text{V}$. The HI&D stage buffered the two electrode potentials (V_{WE} and V_{RE}), while the differential amplifier extracted the potential difference with a controllable offset V_{OS} generated by the internal DAC of MCU. This offset allowed centering the output waveform within the full input swing of the ADC, ensuring high dynamic resolution even at low voltages. The subsequent LPF consisted of two cascaded 2nd-order Sallen–Key cells forming a 4th-order filter with a -3 dB cut-off at 200 Hz . The gain and phase response of the complete analog chain, shown in Figure 5.4, has been validated both through *Cadence OrCAD* simulation and laboratory measurements, showing excellent agreement.

The final version of the analog Section, described in [64], introduced a fully integrated **mixed-signal architecture** combining analog readout and digital control on a single flexible substrate. The analog interface employed the same high-impedance topology

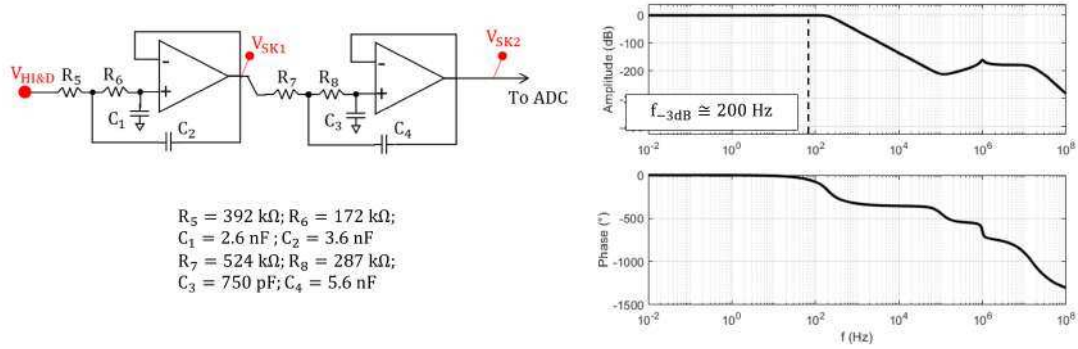


Figure 5.4: Measured and simulated Bode plots of the 4th-order Sallen–Key filter implemented on the second-generation flexible PCB [63].

(OCP block) but has been coupled to a *nRF52840* microcontroller within the NINA-B306 BLE module. This MCU integrates a 12-bit SAR ADC with oversampling capability up to 14 bits, operating at 1 ksps, and manages the system via an embedded *sweat test controller* firmware. The DAC (LTC2611) provided a low-noise, rail-to-rail output for real-time offset control, enabling precise waveform centering at 1.65 V. The analog noise budget of the overall front-end, including the OCP stage and LPF, has been estimated at $2.28 \mu\text{V}_{\text{RMS}}$, allowing the detection of variations in chloride activity as small as $\pm 0.16 \text{ mM}$. A simplified schematic of the implemented analog front-end and its simulated Bode diagram are reported in Figure 5.5.

Power Management and Consumption

A low-power strategy has been progressively integrated into the design to ensure the long-term operation of the wearable system. In the first prototype [65], the total consumption has been approximately 73 mW during active operation, mainly due to continuous BLE communication. In the final version [64], the introduction of hardware-level **power gating** through a TPS22975 load switch and the use of a BU33SD5WG low-dropout regulator reduced the active current to 1.78 mA, with standby currents below 100 μA . The power supply section employed a rechargeable Li-ion coin battery (LIR2477, 3.7 V, 200 mAh), whose output has been regulated to 3.3 V. In standard operation (20 min of measurement per hour), the average consumption reached 350 μW , ensuring an estimated autonomy of 82 days. In standard time operation, measurement per hour, during the first mode, *i.e.*, active measurement, the analog chain is activated for approximately 20 min (INA333

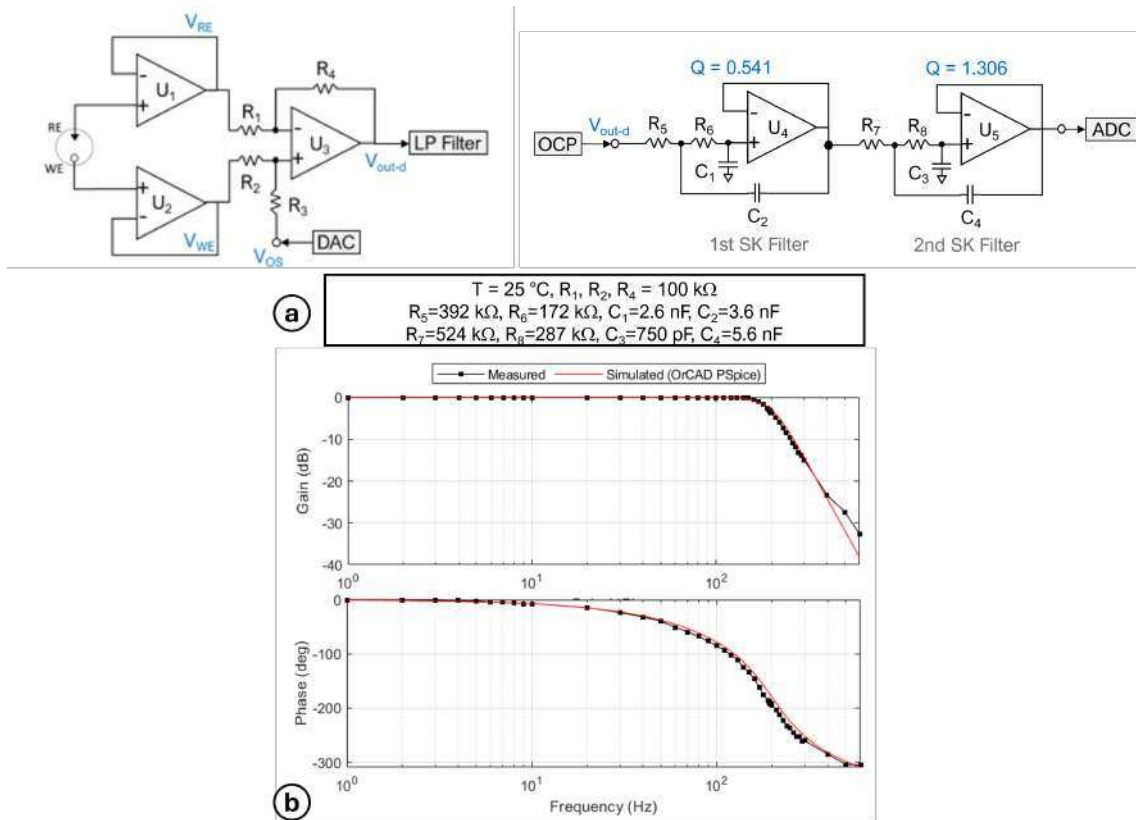


Figure 5.5: Simplified schematic of the third-generation analog front-end with OCP, LPF and programmable offset, adapted from [64].

+ MAX4477), the NINA-B3 is active for 20 min, followed by data transmission to the portable gateway, then goes in standby mode for the remaining 40 min. The DAC is active for 20 min and then turns off for the remaining 40 min. The LDO and load switch, however, remain active. The average current consumption in this configuration is approximately 1.78 mA. The standby mode (second operational mode) involves the NINA-B3 module and the DAC. The average power consumption in this mode is 100 μ A. Considering a 3.7 V LIR 2477 battery with a capacity of 200 mAh, as described in Section II-A5, and considering the described profiles, the estimated battery life for the proposed device is 82 days, with average consumption of the device of 350 μ W.

The fully flexible four-layer PCB layout integrating the sensing front-end, microcontroller, and BLE communication is shown in Figure 5.6.

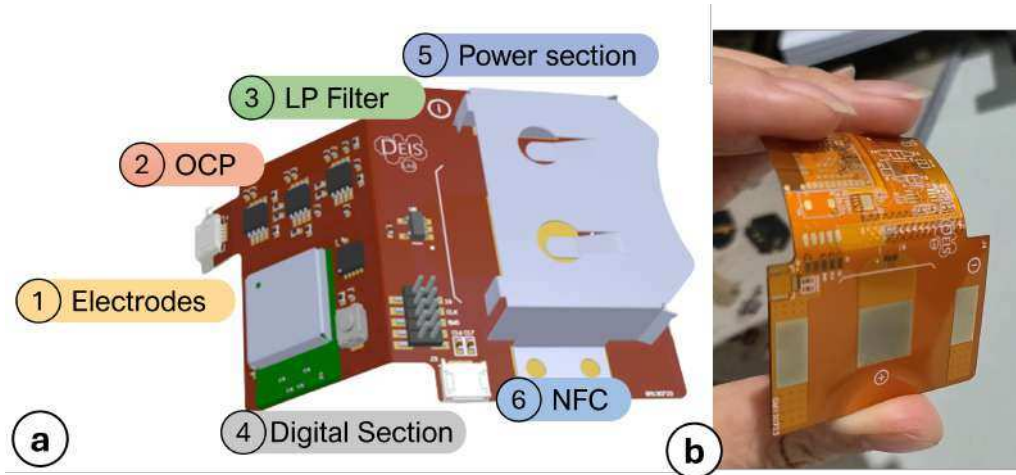


Figure 5.6: Four-layer flexible PCB (49.4 mm $\times$ 39.5 mm) integrating sensing front-end, microcontroller, BLE module, and power management unit [64].

Overall, the evolution of the replaceable electrodes and electronic interface demonstrates a transition from a bench-top prototype to a fully integrated, ultra-low-power system capable of supporting continuous and autonomous sweat analysis in wearable form factors.

5.2.3 Case Study

To validate the proposed wearable platform, a use-case has been designed to simulate a realistic cystic fibrosis (CF) monitoring scenario on pediatric patient. The device operates as an autonomous sweat analyzer, integrating electrodes, analog front-end, digital control,

and wireless communication within a compact and flexible form factor [64].

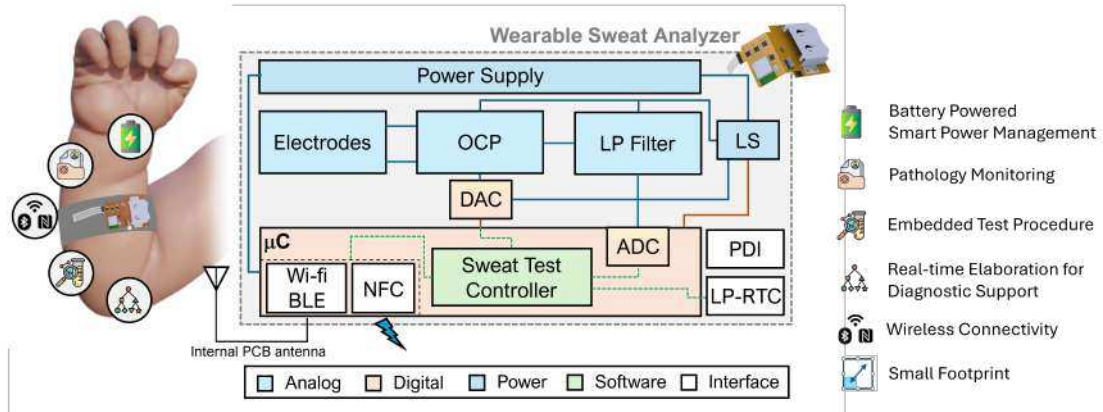


Figure 5.7: Block diagram representation of the proposed wearable sweat analyzer

Figure 5.7 provides a schematic representation of the overall architecture of the wearable chloride analyzer. The system consists of five main functional blocks: (1) the **replaceable electrodes**, responsible for the potentiometric sensing of chloride ions; (2) the **open-circuit potential (OCP)** readout stage; (3) the **low-pass filtering (LPF)** stage; (4) the **microcontroller and transmission module**, which manages data acquisition, signal processing, and BLE communication; and (5) the **power delivery network (PDN)** including voltage regulation and power gating.

The flexible configuration, with dimensions of 49.4 x 39.5 mm, enables the entire system to be worn on the forearm, ensuring continuous and non-invasive monitoring of sweat chloride concentration.

Firmware Workflow and Data Transmission

The firmware executed on the embedded microcontroller, referred to as the *sweat test controller*, supervises the acquisition, calibration, data analysis, and communication phases. Figure 5.8 reports the sequence diagram summarizing the data flow and the interactions between the analyzer, the portable gateway, *i.e.*, smartphone, and the remote *FHIR*-based medical server.

The process starts with the initialization of the system, followed by calibration or measurement selection. Once the sweat test is completed, the measured open-circuit potential is converted into chloride concentration using the stored calibration curve, according to

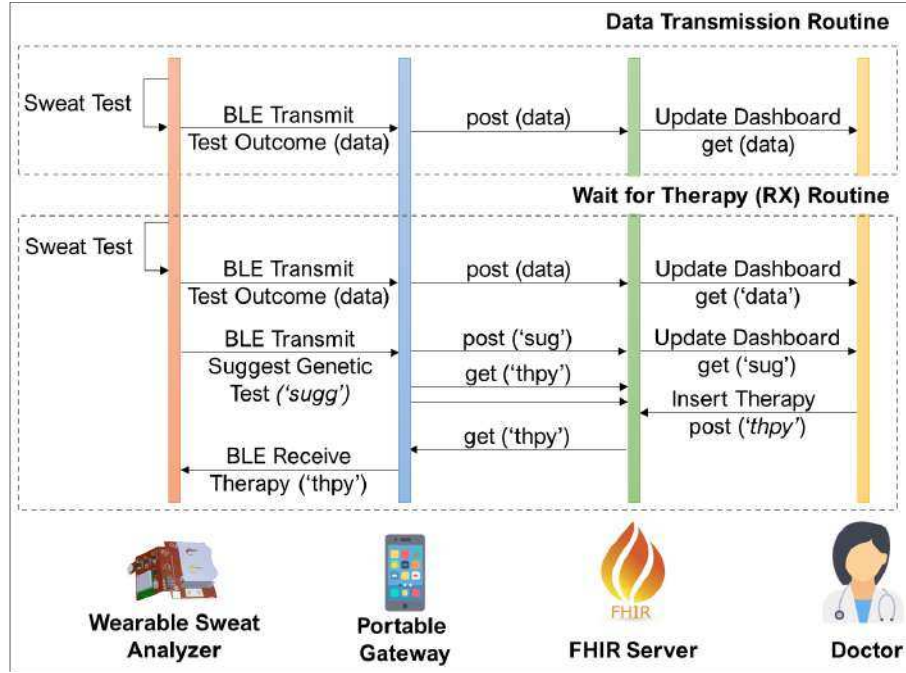


Figure 5.8: Sequence diagram of the communication workflow between the wearable analyzer, portable gateway, and medical FHIR server [64].

the Nernstian relation:

$$C[Cl^-] = 10^{\frac{E-q}{m}} \quad (5.1)$$

The obtained value is then transmitted to the healthcare database, where it is classified as:

- $C[Cl^-] < 30 \text{ mM}$ – *Healthy subject*;
- $30 \leq C[Cl^-] < 60 \text{ mM}$ – *Borderline case*;
- $C[Cl^-] \geq 60 \text{ mM}$ – *Potential CF pathology*.

The same routine manages the scheduling of follow-up tests based on the therapy plan and supports feedback to the physician. As depicted in Figure 5.8, each acquisition is followed by BLE transmission to the portable gateway, which then posts the results to the medical FHIR server. The system subsequently enters a low-power standby mode until the next RTC-triggered event.

Experimental Validation

Figure 5.9(a) shows the experimentally acquired response curve (voltage versus time) during a sweat test. The open circuit potential stabilizes within ± 2 mV after approximately 14 minutes, corresponding to a chloride concentration of 70.9 mM, which falls in the pathological range. The measured value has been compared to the calibration curve reported in Figure 5.9(b), obtained using NaCl solutions between 1 mM and 120 mM. Each plateau corresponds to a stabilized concentration level, and the extracted slope of 56.9 mV/dec demonstrates a near-Nernstian behavior consistent with previous electrode characterizations.

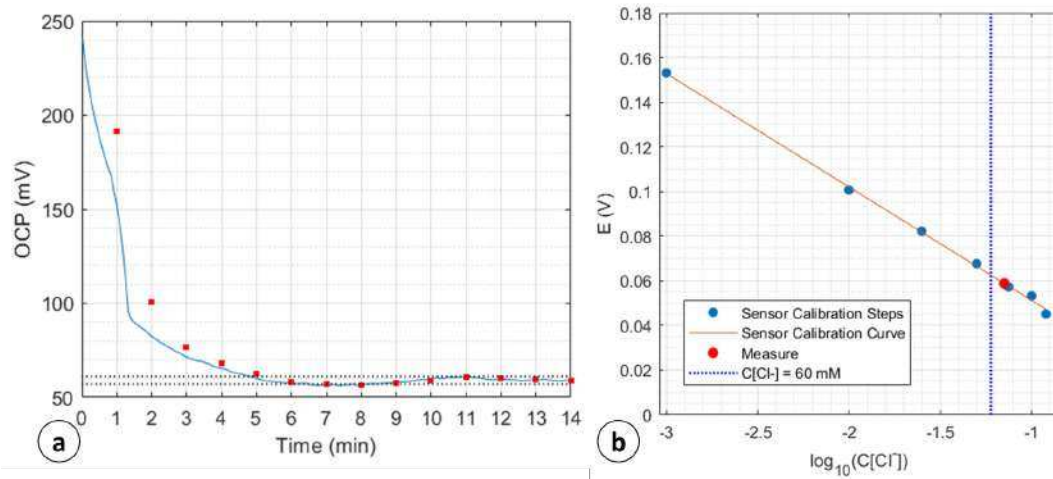


Figure 5.9: (a). Experimental response curve of the wearable chloride analyzer showing stabilization of the OCP during sweat test measurement: acquired values (blue line), computed values (red marker), and settling boundaries (black dashed lines). (b) Calibration curve showing calibration steps, extracted log-linear fitting (calibration curve), extracted measure, and pathological limit ($C[Cl^-] = 60$ mM) [64].

Therapy Monitoring and Follow-Up

To verify the long-term capabilities of the system, let consider a follow-up study over 18 months on a child previously diagnosed with CF. As summarized in Figure 5.10, the chloride concentration progressively decreased with treatment, moving from pathological levels (> 60 mM) to a normal range (< 30 mM) after one year of pharmacological therapy. This result highlights the system's ability to provide reliable feedback on the effectiveness of CFTR-modulator drugs through at-home monitoring. The experimental

results confirm that the proposed wearable analyzer matches the performance of clinical laboratory equipment, providing a resolution of ± 3.4 mM (worst case) and a sensitivity of 56.9 mV/dec in the 5–100 mM range. The combination of accurate potentiometric measurement, autonomous operation, and secure cloud-based data exchange establishes the platform as a promising tool for remote CF diagnosis and long-term patient follow-up.

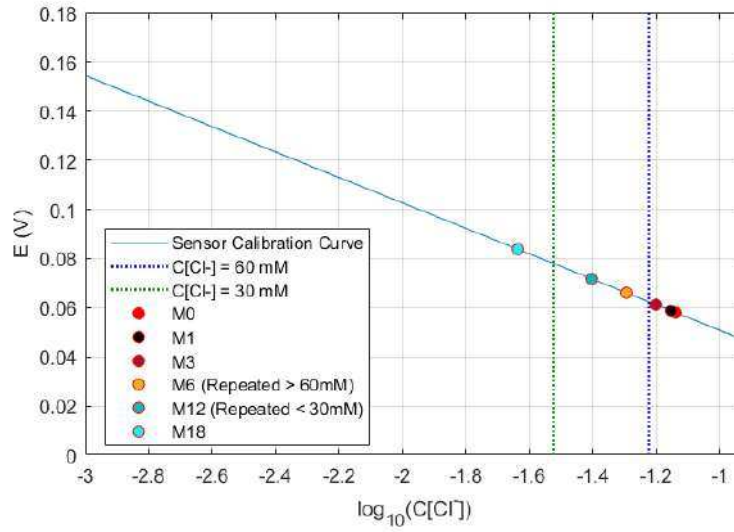


Figure 5.10: Follow-up analysis of chloride concentration over 18 months of CF therapy showing progressive normalization of chloride levels [64].

5.3 On-device XAI

5.3.1 Materials and methods

The experimental campaign evaluates the efficiency and accuracy of the MAFALDA 2.0 framework, including examples of IL, FL, and explainability, across multiple datasets.

Two publicly available datasets were used for both training and prediction.

The *Breast Cancer Wisconsin* dataset [225] includes 569 instances with 30 continuous predictors derived from cytological images; features are organized into mean, standard-error, and worst-case statistics and capture shape/texture descriptors (*e.g.*, radius, perimeter, area, smoothness, symmetry). Labels are binary (benign vs. malignant). The 30 numerical features are extracted from digitalized fine-needle aspirate images of breast masses.

Features are grouped into mean (suffix 1), standard error (suffix 2), and worst-case (suffix 3) measurements for attributes such as texture, radius, perimeter, area, smoothness, and symmetry, and the task is to classify tissues as benign (B) or malignant (M).

The *Heart Disease dataset* [113] comprises 303 records with 13 clinical and test-derived variables spanning demographics, *e.g.*, age; hemodynamic measures, *e.g.*, blood pressure; lipid profile, *e.g.*, cholesterol; and exercise testing indicators, *e.g.*, maximum heart rate. The target is the detection of the presence or absence of heart disease as binary classification.

These datasets were chosen because comparable on-device AI results exist in the literature, and they represent classification problems with numerical and categorical features on MCU-oriented algorithms.

The distributed testbed comprises four interconnected devices, each representative of a distinct class of computing hardware:

- *Edge computing node: Raspberry Pi 4*, Broadcom BCM2711 quad-core Cortex-A72 (ARM v8) SoC at 1.8 GHz, 8 GB synchronous dynamic Random Access Memory (RAM), Gigabit Ethernet, dual-band WiFi (2.4/5.0 GHz, IEEE 802.11ac) and Universal Serial Bus (USB) 3.0.
- *High-end MCU: B-L4S5I-IOT01A* development board with STM32L4S5VIT6 ARM Cortex-M4 @ 120 MHz, 2 MB Flash, 640 kB RAM, WiFi (802.11 b/g/n), Bluetooth 4.1 and onboard sensors.
- *Mid-range MCU: Nordic nRF52840* development kit with ARM Cortex-M4 @ 64 MHz, 1 MB Flash, 256 kB static RAM.
- *Low-end MCU: Arduino Due* based on AT91SAM3X8E Cortex-M3 @ 84 MHz, 512 kB Flash, 96 kB static RAM. In absence of native wireless, a WiFi Shield is used for network connectivity.

Table 5.1 summarizes the comparison of the selected devices.

MAFALDA 2.0 is implemented in C to maximize portability to resource-constrained embedded platforms. All MCUs run *ZephyrOS* (<https://zephyrproject.org/>) as RTOS. Zephyr enables integrating MAFALDA 2.0 within a single codebase and deploying models to different nodes via configuration, without requiring code changes.

Table 5.1: Testbed Nodes Overview.

Node	Chip	Processor	RAM
E	BCM2711	A72 (4C, 1.8GHz)	8 GB
N_{high}	STM32L4S5VIT6	M4 (1C, 120MHz)	640 kB
N_{mid}	nRF52840	M4 (1C, 64MHz)	256 kB
N_{low}	AT91SAM3X8E	M3 (1C, 84MHz)	96 kB

Ontology preprocessing is performed off-device with a Python script that uses *Owl-ready2* [127] for ontology manipulation and *Scikit-learn* [163] for feature transformation. The generated resulting ontology is deployed to all nodes at initialization.

The ontology generation starts with an empty ontology containing two main classes `mafalda:Input` and `mafalda:Output` and two annotation properties `mafalda:minValue` and `mafalda:maxValue` for numerical ranges. Then, following the process column by column, a subclass of `mafalda:Input` is created with the name of each feature; if the feature is categorical, each category is a subclass of the feature, otherwise if the feature is numerical, the values are divided in intervals, a.k.a. binning, partitioning its values into discrete intervals. In other words, each bin is represented as a class with `mafalda:minValue` and `mafalda:maxValue` annotation properties specifying its boundaries and is added as a subclass of the feature class. For each label column, a subclass of `mafalda:Output` is created; each possible value of the label is a subclass of this new label class created.

The binning methods, a.k.a. discretizations, to divide the numerical features into intervals are two:

- **KBins**: an unsupervised method that use K-Means clustering for each feature. The number of clusters, $k \in \mathbb{N}$ is chosen in the range $[k_{\min}, k_{\max}]$, considering the maximum *silhouette coefficient*. The bin boundaries are set at the midpoints between adjacent centroids and for each feature, the numbers of bins depends on the distribution of features.
- **DTree**: the supervised method uses a binary decision tree and chooses the threshold that maximizes the Gini index reduction. The procedure is recursively done until the maximum depth is reached or until each leaf contains at least two samples.

For model optimization and evaluation, each dataset has been split into a training set (70%) and a test set (30%), with 5-fold cross-validation applied to the training data. The

final model was then evaluated on the test set to obtain accuracy metrics. The 5-fold cross-validation is performed on the training set to determine the optimal combination of hyperparameters and ontology via grid search within the following search space:

- **Threshold:** 0.1 to 1.0, with increments of 0.1;
- **Number of classes:** ranges from 5 to 9 in increments of 2, limiting the Decision Tree depth in ontology preprocessing or setting the maximum k_{max} in KBins.
- **Number of descriptions:** 1 to 5.
- **Ontology preprocessor:** KBins or Decision Trees.

Since the case-study data are randomized and lack temporal continuity, the shifting mechanism is disabled by setting the observation window to $+\infty$ (or the maximum storable integer for the data type) and the depth to 1. The search intervals for classes and descriptions are kept small to balance expressiveness and resource constraints: increasing them enlarges the training matrix and memory footprint, which must remain compatible with the target devices.

The framework is assessed on each device with standard metrics. *Accuracy* is computed after training as the proportion of correctly classified instances. *Inference time* is measured per node on single-sample predictions; *training time* is measured on edge and MCUs over the full training set. *Model size* includes the serialized parameters and metadata. *Model loading* (deserialization) and *model saving* (serialization) times are recorded to evaluate storage, sharing, and retrieval. *Peak memory* on MCUs is estimated using `malloc_stats` from *newlib* (<https://www.sourceware.org/newlib/>) plus the maximum stack size; edge-node memory profiling is performed with *Massif* [153].

5.3.2 Experimental outcomes

The evaluation of MAFALDA 2.0 is conducted in the heterogeneous environment of Table 5.1. Figure 5.11 reports the confusion matrices for both datasets, while Table 5.2 summarizes the corresponding performance metrics.

For Breast Cancer, the best configuration uses the Decision Tree for ontology generation with four classes, a single description, and threshold 0.1. For Heart Disease, the best model uses KBins for ontology generation, again with four classes and one description, but with threshold 0.3.

Table 5.2: Performance Metrics for Heart Disease and Breast Cancer Datasets

Dataset	Precision	Recall	F1	Acc
Breast Cancer (Malignant)	1.000	0.857	0.923	0.947
Heart Disease (Positive)	0.886	0.848	0.867	0.868

Table 5.3: Comparison of Breast Cancer Wisconsin dataset results with related works

Algorithm	Device	Acc (%)	Training Time (ms)	Inference Time (ms)	Flash (kB)	RAM (kB)
ML-MCU 2.0 [203]	nRF52840	63	273	0.005	111.864 ¹	77.32 ¹
	STM32F103C8	78	3837	0.1	161.9 (+33.9) ¹	74.33 (+54.33) ¹
	ESP32	62	112	0.005	287.43 ¹	241.73 ¹
	ATSAMD21G18	78	13806	0.33	288.6 (+32.6) ¹	72.58 (+40.58) ¹
Train++ [204]	nRF52840	85.0	5	1	110.41 ¹	77.32 ¹
	STM32F103C8	78.0	66	9	181.6 (+53.6) ¹	49.0 (+29) ¹
	HUZZAH32	78.0	2	<1	285.94 ¹	859.32 ¹
	ESP32	78	2	<1	285.94 ¹	859.32 ¹
	ATSAMD21G18	85.0	286	286	302.7 (+46.7) ¹	72.58 (+40.58) ¹
AIfES [228]	nRF52840 1 x 10 arch	91 ²	15.04	N/A	20.96	8.34
	nRF52840 1 + 10 arch	91 ³	40.34	N/A	21.22	21.27
	nRF52840 10 x 10 arch	91 ³	55.74	N/A	23.33	29.85
TDC [117]	nRF52840	_ ⁴	_ ⁴	_ ⁴	_ ⁴	_ ⁴
	ESP32	92	2.27	N/A	N/A	N/A
MAFALDA 2.0	BCM2711	94	14	0.524	395 ²	94.27
	STM32L4S5VIT6	94	1513	82	358.80	49.90
	nRF52840	94	1941	102	363.19	49.90
	AT91SAM3X8E	94	1722	90	321.62	49.44

¹ Compiler-generated estimates; values in parentheses represent the portion that exceeds available device memory.

² This measurement refers to the executable size of MAFALDA 2.0.

³ [228] do not include accuracy results. The experiments with results were reproduced by [117].

⁴ The model could not fit into the MCU memory.

Table 5.4: Comparison of Heart Disease Binary dataset results with related works

Algorithm	Device	Acc (%)	Training Time (ms)	Inference Time (ms)	Flash (kB)	RAM (kB)
ML-MCU 2.0 [203]	nRF52840	80.1	6	0.015	53.63 ¹	18.76 ¹
	STM32F103C8	82	77	0.03	40.832 ¹	16.24 ¹
	ESP32	83	3	0.015	228.84 ¹	27.38 ¹
	ATSAMD21G18	83	260	0.093	35.73 ¹	17.26 ¹
Train++ [204]	nRF52840	80.18	<1	<1	51.95 ¹	18.74 ¹
	STM32F103C8	82.08	13	1	36.93 ¹	16.23 ¹
	HUZZAH32	80	<1	<1	227.35 ¹	27.35 ¹
	ESP32	83	<1	<1	227.35 ¹	27.35 ¹
	ATSAMD21G18	80.18	44	44	20.47 ¹	19.17 ¹
MAFALDA 2.0	BCM2711	86	42	0.236	395 ²	46.16
	STM32L4S5VIT6	86	204	22	270.07	30.19
	nRF52840	86	261	27	231.46	30.19
	AT91SAM3X8E	86	308	29	244.78	30.19

¹ Compiler-generated estimates.² This measurement refers to the executable size of MAFALDA 2.0.

Malignant	108	0	Positive	39	7
Benign	9	54	Negative	5	40
	M	B		P	N

(a) Breast Cancer

(b) Heart Disease

Figure 5.11: Classification results: confusion matrices with color-coded values

Training and prediction

Experiments are executed on the edge node E and on MCUs N_{high} , N_{mid} , and N_{low} . Each node loads data and ontology; accuracy, training time, and peak memory consumption have been measured. Table 5.3 reports Breast Cancer Wisconsin results, and Table 5.4 reports Heart Disease results, alongside state-of-the-art on-device AI baselines. The comparison spans different ML algorithms, indicating MCU class, accuracy, training time, per-sample inference time, and memory footprint (Flash and SRAM peaks).

When comparing the obtained results with existing on-device learning frameworks, MAFALDA 2.0 consistently achieves the highest predictive performance across all evaluated devices and datasets. On the Heart Disease dataset, the proposed framework attains an average accuracy of 86%, improving upon ML-MCU 2.0 and Train++, whose performance ranges between 80% and 83%. This advantage is maintained across all hardware classes, from the high-end STM32L4S5VIT6 to the low-end AT91SAM3X8E, demonstrating the robustness of the proposed semantic learning approach under heterogeneous computational conditions. For the Breast Cancer Wisconsin dataset, MAFALDA 2.0 reaches 94% accuracy on all tested nodes, outperforming the compared models, which exhibit broader and generally lower accuracy ranges. Specifically, ML-MCU 2.0 achieves between 62% and 78%, while Train++ attains between 78% and 85%. Other lightweight neural or symbolic methods, such as AIfES [228] and the TinyDecisionTreeClassifier (TDC) [117], report slightly lower accuracy values (91–92%), confirming the superior predictive capability of MAFALDA 2.0. These numerical results confirm that the ontology-based reasoning adopted by MAFALDA 2.0 can provide measurable improvements in classification accuracy while remaining executable on resource-constrained microcontrollers.

Overall, results indicate that MAFALDA 2.0 supports on-device AI training and inference on resource-constrained platforms. The extra computational burden compared to purely numeric models is attributable to symbolic representations and the associated preprocess-

ing/transformation steps. This additional complexity, however, enables explainability and high-level reasoning in SWoE contexts.

The higher computational cost observed in MAFALDA 2.0 with respect to purely numerical models is primarily due to the use of symbolic representations and the associated data preprocessing steps required before model execution. Unlike conventional ML approaches that directly operate on raw numerical features, MAFALDA 2.0 encodes and reasons over semantic descriptions, which naturally increases processing time and memory usage. Nevertheless, these additional operations remain compatible with the available resources of mid-range MCUs, ensuring that both training and inference can be performed locally without offloading. Furthermore, this computational overhead brings advantages: the semantic representation enables intrinsic explainability of model decisions and facilitates high-level reasoning and knowledge integration in SWoE scenarios. Thus, the symbolic layer in MAFALDA 2.0, while slightly increasing execution time and memory footprint, enhances interpretability and opens the path to federated and incremental learning workflows on embedded ML models.

Federated Learning

To support FL and model merging, MAFALDA 2.0 integrates Protocol Buffers (<https://protobuf.dev/>) for efficient serialization/ deserialization. Profiling of these operations is reported in Table 5.5. Encoded model sizes are 6.38 kB (Breast Cancer) and 2.43 kB (Heart Disease). As expected, device *E* outperforms MCUs by roughly an order of magnitude in serialization and nearly two orders in deserialization. Specifically *E* reached the fastest serialization times across both datasets, with 0.848 ms for the Breast Cancer dataset and 0.587 ms for the Heart Disease dataset. In contrast, MCU nodes demonstrate a marked decline in serialization speed, with times ranging between 2.59–3.61 ms for the Breast Cancer dataset and 1.48–2.05 ms for the Heart Disease dataset. In deserialization, *E* outperforms MCU nodes 17.41 ms vs. 1292.34 ms (N_{lox}) for Breast Cancer dataset; and 4.61 ms vs. 228.79–302.35 ms for Heart Disease dataset. These disparities in deserialization time, wherein MCU nodes necessitate a substantially greater duration compared to the edge node, indicates that MCUs necessitate a greater computational effort to load and interpret the model, which could be a limiting factor in real-time applications.

A second trial evaluates IL and FL capabilities using the Breast Cancer dataset. Data are split into train/test with a 20% hold-out; the training set is then partitioned into

four parts. The first partition is used on E for model selection (5-fold) and ontology generation. The trained model and ontology are then dispatched to the three MCU nodes, each receiving one of the remaining partitions. Each MCU performs local refinement using its own partition and the shared ontology, evaluates on the original test set, and sends its updated model back to E . Finally, E aggregates the received models, without modification, and evaluates the final version to assess improvements. Accuracies are reported in Table 5.6.

Edge node E attains 0.918 after the initial training. Among MCUs, N_{high} reaches 0.924, N_{mid} attains 0.93 (the highest), and N_{low} equals N_{high} at 0.924. Despite hardware heterogeneity, all MCUs successfully train and update locally, demonstrating the feasibility of FL on embedded devices. The aggregated model reaches 0.93 accuracy, matching or exceeding individual MCU results and confirming the effectiveness of the federated approach. Its slight decrease compared to the 0.94 achieved when the full dataset is locally available (Table 5.3) reflects the expected trade-off between local optimizations and global generalization.

Table 5.5: Serialization and Deserialization Times

N	Breast Cancer (6.38 kB)		Heart Disease (2.43 kB)	
	Ser. (ms)	Des. (ms)	Ser. (ms)	Des. (ms)
E	0.848	17.41	0.587	4.61
N_{high}	2.61	964.83	1.50	228.79
N_{mid}	2.59	1232.63	1.48	284.65
N_{low}	3.61	1292.34	2.05	302.35

Table 5.6: Accuracy for Breast Cancer Dataset of all devices

Device	E	N_{high}	N_{mid}	N_{low}	Aggregated
Accuracy	0.918	0.924	0.93	0.924	0.93

Energy consumption

To appraise the efficacy of MAFALDA 2.0 on heterogeneous datasets and to assess the energy consumption, the performance of MAFALDA 2.0 with ML-MCU and Train++ [204]

was compared. The comparison has been carried out on the common hardware platform, nRF52840, using the *MNIST handwritten digits* image dataset. Results are displayed in Table 5.7. Furthermore, Table 5.8 presents an estimated comparison with several TensorFlow Lite Micro configurations documented in literature [57], and evaluated on the *UCI-HAR* multivariate dataset for human activity recognition [14]. This evaluation was carried out using the same ESP32 board and a 50 Hz sampling rate for Inertial Measurement Unit (IMU) signals. This comparison is intended to position MAFALDA 2.0 relative to widely adopted industry-standard solutions for on-device activity recognition. The measurement of energy is performed through the utilization of the *Monsoon High Voltage Power Monitor*, developed by Monsoon Solutions Inc. (<https://www.msoon.com/high-voltage-power-monitor>). Reported values are the energy mean over five independent runs, computed over the entire dataset and normalized by the number of samples.

For MNIST, task setup, data split, and preprocessing follow the one reproduces from [204]. For UCI-HAR dataset, the extraction of features has been conducted by employing a 50-sample sliding window over raw inertial data, computing mean and standard deviation per segment. Contrary to the original study, which employed convolutional models on raw sequences, MAFALDA 2.0 requires explicit feature engineering; preprocessing is included in both time and energy for training and inference.

Table 5.7 shows that MAFALDA 2.0 consumes more time and energy for both training and inference than simpler models that rely on basic vector operations at inference. However, as indicated by Table 5.8, it is more efficient than several general-purpose TinyML neural network configurations in TensorFlow Lite Micro. Taken together with comparable predictive accuracy, these results indicate that MAFALDA 2.0 is a good option for on-device training and inference, while offering on-device FL and eXplainability, that the other approaches do not address.

5.3.3 Explainability analysis

The explainability of MAFALDA 2.0 was experimentally assessed using the Breast Cancer dataset to evaluate both global and local interpretability. In this experiment, the system represents domain knowledge through symbolic OWL descriptions and performs classification as a semantic matchmaking task, allowing the resulting reasoning process to be directly interpretable (see Section 4.3).

Table 5.7: Comparison on the MNIST Handwritten Digits dataset: training and inference time (milliseconds), energy consumption (millijoules), and accuracy. All values are reported per sample.

Algorithm	Training		Inference		Accuracy (%)
	Time	Energy	Time	Energy	
ML-MCU [203]	5.09	0.68	2.38	0.34	89.7
Train++ [204]	0.0659	0.0098	0.0138	0.002	98.0
MAFALDA 2.0	139	21	41	6	97.2

Table 5.8: Comparison on the UCI-HAR dataset: training and inference time (milliseconds) energy consumption (millijoules), and accuracy. All values are computed per sample and account for feature extraction and preprocessing overhead.

Algorithm	Training		Inference		Accuracy (%)
	Time	Time	Energy	Energy	
TFMicro [57]	n/a	5–72	253–324	86–92	
MAFALDA 2.0	94.875	9	3.55	90.5	

Figure 5.12 illustrates an example of the semantic structures generated by the model reported in Table 5.3. These symbolic descriptions encode the learned associations between input features and class labels, providing an explicit semantic representation of what the model has inferred during training.

Given the dataset’s 30 features, the resulting semantic representations can become quite articulated. For this reason, MAFALDA 2.0 provides a *cutoff parameter* that regulates how many concepts are shown when explanations are converted from internal symbolic structures into human-readable form. Concepts whose TM value falls below a user-defined threshold are omitted from visualization, thus controlling the trade-off between completeness and readability. Lower cutoff values preserve every activated concept, offering a full, high-fidelity account of the symbolic reasoning process. Conversely, higher cutoffs produce more compact and cognitively accessible explanations by filtering out less influential concepts.

As shown in Figure 5.12, a cutoff of 0.65 yields a more detailed explanation, while a value of 0.75 results in a shorter and more abstract representation. In both cases, concepts are sorted from the most to the least frequent across the dataset, enabling a global interpretation of which features most characterize each class.

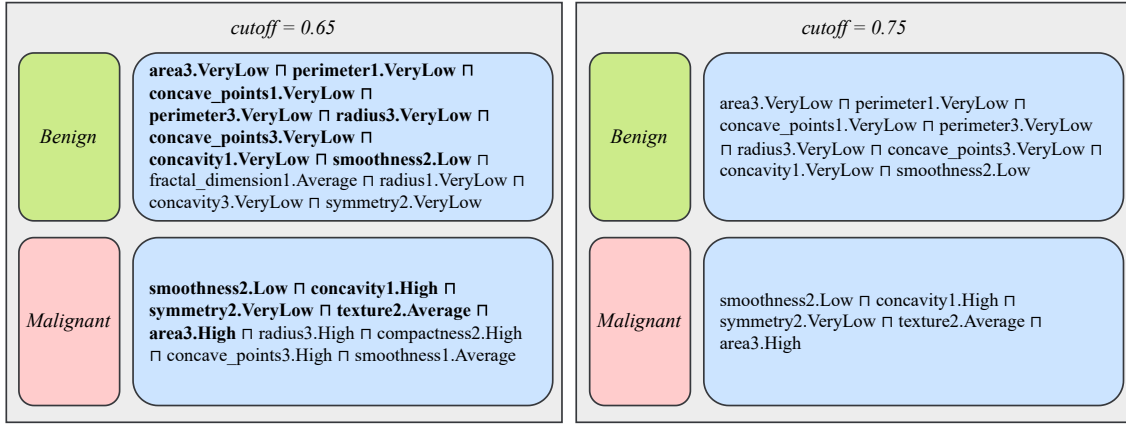


Figure 5.12: MAFALDA 2.0 global explanation: learned semantic description of output class

In particular, benign tumors are associated with low values of area, perimeter, concave points, radius, and concavity across mean, standard deviation, and maximum measurements, describing smaller and more regular cellular shapes. Malignant samples, instead, display high concavity (mean), low symmetry (standard deviation), and larger area (maximum), corresponding to irregular morphologies with rougher edges.

In addition to model-level interpretability, MAFALDA 2.0 also provides local explanations for individual samples, clarifying the reasoning behind each classification decision. Figure 5.13 presents one such example, where a sample correctly classified as benign is explained by the combination of low area, perimeter, radius, concavity, and smoothness values. These factors correspond to ontological concepts activated during reasoning, and their alignment with the input demonstrates the semantic transparency of the model. Furthermore, MAFALDA 2.0 can generate *contrastive* and *counterfactual* explanations, indicating which features would need to change for the model to produce a different outcome. For instance, in this case, a malignant classification would require higher compactness and increased texture variation. The missing features (H' in the input and H'' in the model) represent the counterfactual elements that, if modified, would lead to a different inferential conclusion.

According to the taxonomy introduced in Section 4.3 and summarized in Table 4.2, the explanations generated by MAFALDA 2.0 correspond to a subset of the categories defined in the Explanation Ontology [49].

Contrastive and counterfactual explanations. At the local level, MAFALDA 2.0

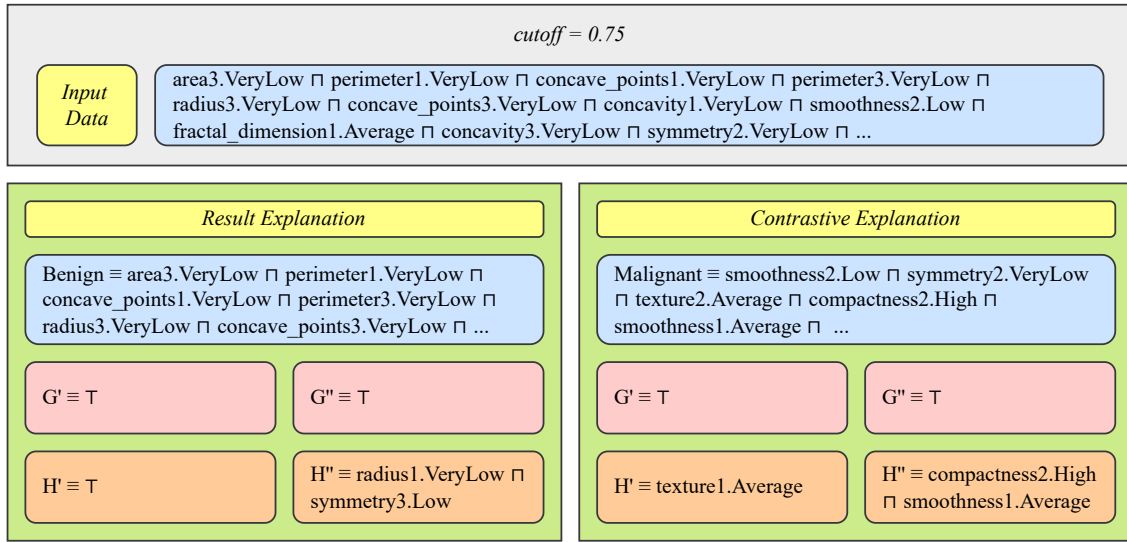


Figure 5.13: MAFALDA local explanation: classification outcome explained by means of reasoning services.

clarifies why a given instance has been classified in one way rather than another (contrastive explanation) and indicates which feature changes would alter the decision (counterfactual explanation). In the Breast Cancer case study, a benign sample is justified by low area, perimeter, and concavity, whereas a malignant classification would require increased compactness and higher texture variation. These mechanisms are implemented through the logical operations of *concept contraction* and *concept abduction*, respectively, providing semantically grounded interpretability at the instance level.

Contextual explanations. The ontology encodes biomedical knowledge (*e.g.*, relationships among morphological features and diagnostic classes), enabling MAFALDA 2.0 to situate each explanation within its semantic context. For example, the association between “low concavity” and “benign morphology” is not purely statistical but also reflects domain semantics, supporting context-aware interpretability.

Trace-based explanations. These are not directly supported, as the semantic match-making process is executed internally and does not expose a readable reasoning trace to the user. Future extensions could incorporate provenance tracking or rule logging to reconstruct the full inference chain.

Case-based explanations. Also not implemented in the current version, but potentially achievable by storing previous reasoning instances or semantic exemplars. Such an addition would allow the system to justify current outcomes by analogy with previously

analyzed cases.

Overall, MAFALDA 2.0 currently supports contextual, contrastive, and counterfactual explanations, while the absence of trace-based and case-based mechanisms highlights possible directions for extending its XAI capabilities in future developments.

Chapter 6

Conclusion and perspectives

6.1 Hazard-driven design

The problem of safety addressed the assurance in fully automated railway systems operating under Grade of Automation 4 (GoA4), with particular focus on the role of hazard analysis as a design-driving and validation-enabling process. In the absence of an onboard driver, safety can no longer rely on human supervision, but must be enforced through embedded onboard subsystems capable of detecting hazardous conditions and reacting within well-defined temporal constraints. Within this context, the contribution of this work lies in the systematic extraction of hazards through a norm-compliant methodology and in the quantitative assessment of the system response to such hazards.

A first key outcome of this study is the confirmation that hazard-driven design represents a fundamental quality enabler for safety-critical railway systems. Rather than being confined to a preliminary or documentary phase, hazard analysis directly informs architectural decisions, functional decomposition, and system-level constraints. By adopting a structured HAZOP-based approach compliant with [111] and [44], hazardous scenarios were identified in a systematic and traceable manner, ensuring consistency with functional safety principles throughout the system lifecycle. This approach contributes not only to risk reduction, but also to improved robustness of the overall system specification.

The core aspect of the hazard-driven validation approach is the traceability established between identified hazards, derived safety requirements, and measurable behavioral constraints. The hazard-to-requirement mapping performed enabled the translation of abstract safety concerns into concrete system-level requirements, distinguishing between safety-

critical and system-functional constraints. This traceability is particularly relevant in highly integrated architectures such as ATO over ETCS systems, where hazardous conditions emerge from the interaction of multiple subsystems rather than from isolated component failures. By maintaining a link between hazards and the functions responsible for their mitigation, the proposed approach facilitates both verification activities and future system evolution.

Beyond the identification and classification of hazards, this work emphasized the importance of quantitatively assessing system behavior under hazardous conditions. In GoA4 scenarios, the timeliness of safety actions is as critical as their correctness. For this reason, intervention time was adopted as a key performance metric for evaluating the effectiveness of hazard mitigation mechanisms. The experimental analysis demonstrated that the system under study is capable of reacting to hazardous events within temporal margins that are well below the limits prescribed by relevant railway standards. This result highlights how hazard-driven requirements can be meaningfully validated through time-based performance indicators, rather than through qualitative compliance alone.

The use of intervention time as a safety-critical metric represents an important step toward bridging the gap between hazard analysis and experimental validation. While traditional safety assessments often focus on functional correctness, the approach presented in this work shows that temporal behavior can be explicitly incorporated into the safety argument. This is particularly relevant for fully automated railway systems, where delayed reactions may lead to unsafe states even in the absence of functional faults. The proposed methodology thus supports a more comprehensive view of safety, encompassing both logical correctness and real-time performance.

It is important to acknowledge the scope and limitations of the presented study. The experimental evaluation focused on a representative subset of preliminary hazardous scenarios derived from the hazard analysis, and the results are therefore conditioned by the assumptions made on system configuration and operational context. Moreover, the hardware and software implementation of the Vital Control Module were not developed within the scope of this work, but it was considered as a reference platform for assessing hazard-driven behavioral constraints. Nevertheless, these limitations do not detract from the generality of the proposed approach, which is applicable to a wide range of safety-critical fully automated embedded railway subsystems.

From a broader perspective, the results of this work suggest that the integration of structured hazard analysis with quantitative performance evaluation is a promising direction for

the development of future GoA4 railway systems. As railway automation continues to advance, the complexity of onboard architectures and the criticality of their interactions will further increase. In this scenario, methodologies capable of systematically linking hazards, requirements, and measurable system behavior will play a crucial role in ensuring safety, reliability, and regulatory compliance.

In conclusion, this study demonstrates that hazard analysis, when treated as an active and design-driving process, can provide a solid foundation for both system specification and experimental validation in safety-critical railway applications. By combining norm-compliant hazard extraction with time-based performance assessment, the proposed approach contributes to a more rigorous and transparent safety engineering process, supporting the safe deployment of fully automated railway systems operating under GoA4 conditions and motivating the extension of current safety standards to explicitly address the requirements of fully autonomous railway operations.

6.2 Sensing

The sensing research activities described throughout this work led to the design, development, and experimental validation of a fully wearable and flexible platform for real-time chloride activity sensing in human sweat. The proposed system targets the early diagnosis and continuous monitoring of *Cystic Fibrosis* (CF), addressing the need for a portable, non-invasive, and low-power diagnostic tool capable of complementing or replacing conventional laboratory-based sweat tests. The technological pathway followed across the three studies, the early prototypes presented in [65], through the miniaturized platform in [63], up to the fully integrated mixed-signal system in [64], demonstrates a coherent evolution from concept to a near-market-ready device.

The research focused on defining a reliable chloride ion-selective electrode pair capable of operating on flexible substrates. In the 2023 prototype [65], silver and silver chloride electrodes have been fabricated on a PET substrate using a stencil-printing process. This configuration achieved a pseudo-Nernstian sensitivity of 53 mV/dec and successfully validated the feasibility of real-time chloride detection. The following iteration [63] optimized the electrode geometry and curing process on a Mylar substrate, improving the adhesion and mechanical stability of the silver ink. Morphological analyses confirmed high chemical uniformity and purity, and the measured sensitivity increased to 57 mV/dec, approaching

the theoretical Nernstian value of 59 mV/dec. Finally, the third generation [64] introduced a controlled immersion process in sodium hypochlorite to obtain a homogeneous $\text{Ag}_2\text{O}/\text{AgCl}$ passivation layer on the reference electrode. This innovation significantly reduced potential drift, yielding an accuracy between ± 0.6 mM and ± 3.4 mM across the 10–100 mM range and ensuring stability better than ± 2 mV over 10 minutes. These results confirmed that the electrochemical performance of the wearable electrodes is comparable to clinical-grade chloride analyzers. The electronics evolved from a bench-top prototype to a fully flexible and autonomous platform, with replaceable electrodes pair.

The conducted research provides a multi-disciplinary contribution at the intersection of electrochemistry, low-power electronics, and biomedical sensing. From a scientific standpoint, it validates the application of potentiometric principles on flexible substrates with near-Nernstian behavior, introducing new fabrication methodologies for Ag/AgCl-based electrodes. From a technological viewpoint, it demonstrates the feasibility of a **fully integrated wearable analyzer** that combines chemical sensing, mixed-signal electronics, and wireless data communication in a single flexible system. Such integration establishes a foundation for the development of smart diagnostic devices capable of reasoning on the acquired data and enabling remote clinical decision-making.

Building upon the presented results, several future research directions can be pursued:

- **Sweat solution simulation:** testing the feasibility of the electrodes on a sweat solution could provide a more comprehensive evaluation of the reaction with silver and silver-chloride.
- **Integration of autonomous sweat stimulation:** the inclusion of micro-heaters or iontophoresis drivers on the flexible substrate would enable sweat induction without external agents.
- **Multiparametric analysis:** combining chloride detection with additional biomarkers (*e.g.*, Na^+ , K^+ , lactate, or glucose) could provide a more comprehensive evaluation of metabolic and hydration states.
- **AI-assisted data analytics:** embedding lightweight ML algorithms within the microcontroller could support trend recognition, anomaly detection, and predictive diagnostics considering not only the chloride concentrations but other vital parameters.

- **Energy harvesting and management:** integrating energy harvesting circuits (*e.g.*, thermoelectric or photovoltaic) could further extend autonomy and sustainability of the platform.

In summary, the three-stage evolution described in [65, 63, 64] led to the realization of low-power and biocompatible system capable of reliable chloride measurement directly on the human body. The proposed platform not only matches the performance of specialized clinical equipment but also opens the way to a new generation of connected biomedical devices for continuous health monitoring with replaceable electrodes system.

6.3 On-device XAI

The AI research activity has presented *MAFALDA 2.0*, a XAI framework specifically designed for deployment on resource-constrained embedded platforms. The work addressed one of the major challenges in modern intelligent systems, achieving a balance between computational efficiency and interpretability, by combining symbolic reasoning with statistical learning within a unified software architecture. The proposed approach reformulates the learning and inference process as a *semantic matchmaking* task, where knowledge is represented through ontological structures that explicitly describe the relationships among features, classes, and contextual information. This paradigm enables not only effective classification but also transparent and semantically interpretable reasoning.

From a software engineering standpoint, *MAFALDA 2.0* was fully implemented in C to ensure cross-platform portability and real-time performance. The integration with lightweight real-time operating systems and the modular design of its components enable deployment across heterogeneous hardware, including microcontrollers, single-board computers, and standard computing platforms, with minimal modification effort. This represents a significant step toward *embedded* AI systems that can learn, reason, and act autonomously without relying on external infrastructures (cloud or edge or fog).

Experimental results confirmed the feasibility of performing both training and inference directly on low-power devices. Across the evaluated datasets, *MAFALDA 2.0* achieved accuracy levels comparable to state-of-the-art TinyML models while maintaining controlled memory, execution time, and energy consumption. The transformation of numerical features into ontology-based semantic representations has proven effective in increasing model interpretability with a limited impact on computational efficiency. These

findings validate the hypothesis that it is possible to integrate explainability into embedded AI systems without compromising performance.

In addition to its on-device learning capabilities, MAFALDA 2.0 supports IL and FL modes. These features allow knowledge to be refined over time and distributed across devices, thereby enabling the emergence of *symbiotic learning networks* aligned with the paradigms of the IoE and SWoE. This integration positions the framework as a concrete step toward fully decentralized intelligent systems capable of adapting to evolving environments while preserving data privacy and autonomy.

Regarding explainability, the introduction of a *cutoff parameter* allows the system to regulate the verbosity of explanations generated from symbolic structures. This mechanism provides a tunable trade-off between fidelity and comprehensibility, allowing the level of detail to be adapted to the user or application context. In future developments, these explainability dimensions will be formally evaluated to establish quantitative metrics for assessing the quality and effectiveness of generated explanations.

Despite its advantages, some limitations remain. The generation and management of ontology structures introduce a moderate computational overhead that can affect execution time in real-time applications. Future research will focus on optimizing ontology creation and matching processes, possibly through hybrid symbolic–subsymbolic approaches that dynamically adapt the semantic representation during execution. Another important perspective concerns the possibility of adapting MAFALDA 2.0 to other explanation types, like statistical and scientific.

Beyond technical optimization, a key research direction involves validating MAFALDA 2.0 in real-world scenarios involving streaming data and dynamic environments. Demonstrating continuous adaptation and learning in such contexts would consolidate the framework’s role in next-generation wearable, IoT, and cyber-physical systems.

In conclusion, the research presented in this thesis contributes to the advancement of *On-Device Explainable AI*, bridging the gap between symbolic knowledge representation and efficient embedded learning. MAFALDA 2.0 demonstrates that interpretable models can coexist with the stringent resource constraints of embedded devices, paving the way toward transparent, sustainable, and autonomous AI systems. The proposed framework not only represents a technological achievement but also sets the foundations for future investigations into explainable, distributed, and self-adaptive intelligent agents operating at the edge of the network. The development of intuitive and user-friendly visualization tools that present explanations in a way non-experts can easily understand [152]. Tools

that integrate visual representations of feature importance or decision boundaries will support decision-makers in interpreting outputs quickly and effectively.

Bibliography

- [1] Fadwa Aaboub, Hasna Chamlal, and Tayeb Ouaderhman. Statistical analysis of various splitting criteria for decision trees. *Journal of Algorithms & Computational Technology*, 17:17483026231198181, 2023.
- [2] Ryan Aalund and Vincent Philip Paglioni. Enhancing reliability in embedded systems hardware: A Literature survey. *IEEE Access*, 2025.
- [3] Muhammad Abubakar, Adbul Sattar, Hamid Manzoor, Khola Farooq, and Muhammad Yousif. Iiot: An infusion of embedded systems, tinymml, and federated learning in industrial iot. *Journal of Computing & Biomedical Informatics*, 8(02), 2025.
- [4] Malina Adach, Nazakat Ali, Kaj Hänninen, and Kristina Lundqvist. Hazard analysis on a system of systems using the hazard ontology. In *2023 18th Annual System of Systems Engineering Conference (SoSe)*, pages 1–6. IEEE, 2023.
- [5] Giovanni Agosta, Andrea Galimberti, and Davide Zoni. Deep learning on risc-v platforms at the edge: A perspective on the hardware and software support. *ACM Comput. Surv.*, 58(5), 2025.
- [6] Dharma Prakash Agrawal. *Embedded sensor systems*. Springer, 2017.
- [7] Marzia Ahmed, Mohammad Abul Kashem, Mostafijur Rahman, and Sabira Khatun. Review and Analysis of Risk Factor of Maternal Health in Remote Area Using the Internet of Things (IoT). *Lecture Notes in Electrical Engineering*, 2020.
- [8] Shamil Al-Ameen, Bharath Sudharsan, Tejus Vijayakumar, Tomasz Szydlo, Tejal Shah, and Rajiv Ranjan. Poly Instance Recurrent Neural Network for Real-time Lifelong Learning at the Low-power Edge. In *2024 IEEE International Conference on Big Data (BigData)*, pages 5907–5916. IEEE, 2024.

- [9] Dean Allemang and James Hendler. *Semantic web for the working ontologist: effective modeling in RDFS and OWL*. Elsevier, 2011.
- [10] M AlNusif. Explainable ai in edge devices: A lightweight framework for real-time decision transparency. *Int. J. Eng. Comput. Sci*, 14(07):27447–27472, 2025.
- [11] Hanin Amara, Nadeem Tariq Beigh, and Nouha Alcheikh. Smart resonant micro-sensor and micro-actuator: high-performance, wide range bi-axial magnetic sensitive/insensitive micro-device for multifunctional sensing applications. *Microsystems & Nanoengineering*, 11(1):174, 2025.
- [12] Arturo Amendola, Mario Barbareschi, Salvatore De Simone, Giovanni Mezzina, Alberto Moriconi, Cataldo Luciano Saragaglia, Diana Serra, and Daniela De Venuto. A real-time vital control module to increase capabilities of railway control systems in highly automated train operations. *Real-Time Systems*, 59(4):636–661, 2023.
- [13] Rashid Amin, Mohammed A Al Ghamdi, Sultan H Almotiri, Meshrif Alruily, et al. Healthcare techniques through deep learning: issues, challenges and opportunities. *IEEE Access*, 9:98523–98541, 2021.
- [14] D. Anguita, Alessandro Ghio, L. Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A Public Domain Dataset for Human Activity Recognition using Smartphones. In *The European Symposium on Artificial Neural Networks*, pages 437–442, 2013.
- [15] Mattia Antonini, Miguel Pincheira, Massimo Vecchio, and Fabio Antonelli. An adaptable and unsupervised TinyML anomaly detection system for extreme industrial environments. *Sensors*, 23(4):2344, 2023.
- [16] R. Arthi and V. Krishnaveni. Optimized Tiny Machine Learning and Explainable AI for Trustable and Energy-Efficient Fog-Enabled Healthcare Decision Support System. *Scientific Reports*, 2024.
- [17] Peter GV Axelberg, Irene Yu-Hua Gu, and Math HJ Bollen. Support vector machine for classification of voltage disturbances. *IEEE Transactions on power delivery*, 22(3):1297–1303, 2007.
- [18] Franz Baader, Diego Calvanese, Deborah L McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2002. 2nd Ed.

- [19] James Balfour, William Dally, David Black-Schaffer, Vishal Parikh, and JongSoo Park. An energy-efficient processor architecture for embedded systems. *IEEE Computer Architecture Letters*, 7(1):29–32, 2008.
- [20] José Luis Corcuera Bárcena, Mattia Daole, Pietro Ducange, Francesco Marcelloni, Alessandro Renda, Fabrizio Ruffini, and Alessio Schiavo. Fed-XAI: Federated Learning of Explainable Artificial Intelligence Models. In *XAI. it@ AI* IA*, pages 104–117. Udine, 2022.
- [21] Alexander Barkalov, Larysa Titarenko, and Ma lgorzata Mazurkiewicz. *Foundations of embedded systems*, volume 195. Springer, 2019.
- [22] Firas Bayram and Bestoun S Ahmed. Towards trustworthy machine learning in production: An overview of the robustness in mlops approach. *ACM Computing Surveys*, 57(5):1–35, 2025.
- [23] Justin Beaurivage, Messaoud Ahmed Ouameur, and Frédéric Domingue. A Memory Representation of Random Forests Optimized for Resource-Limited Embedded Devices. *IEEE Embedded Systems Letters*, 2025.
- [24] Ilja Behnke. *Real-Time Aware IP-Networking for Resource-Constrained Embedded Devices*. PhD thesis, Technische Universität Berlin, 2024.
- [25] Tim Berners-Lee. Linked Data: Design Issues. Technical report, World Wide Web Consortium, 2010. <https://www.w3.org/DesignIssues/LinkedData.html>.
- [26] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25(2):197–227, 2016.
- [27] Ivano Bilenchi, Filippo Gramegna, Giuseppe Loseto, Saverio Ieva, Floriano Scioscia, and Michele Ruta. Cowl: Pushing OWL 2 over the Edge. *Internet of Things*, 29:101439, 2025.
- [28] Ivano Bilenchi, Agnese Pinto, Grazia Mascellaro, Filippo Gramegna, Giuseppe Loseto, and Michele Ruta. Reasoning over Personal Health Knowledge Graph for healthcare monitoring on Apple Watch. In *The Inclusive Web: Realizing Safe, Accessible, Inclusive, and Sustainable Web Engineering (ICWE) - 3rd International*

Workshop on the Semantic Web of EveryThing (SWEET), pages 85–97. Springer Nature Switzerland, 2026.

- [29] Christopher M Bishop and Hugh Bishop. *Deep learning: Foundations and concepts*. Springer Nature, 2023.
- [30] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.
- [31] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. *Advances in neural information processing systems*, 31, 2018.
- [32] M Greenwell Brandon. pdp: An R Package for Constructing Partial Dependence Plots. *The R Journal*, 9(1):421, 2017.
- [33] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [34] Leo Breiman, Jerome Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Chapman and Hall/CRC, 2017.
- [35] Richard G Brereton and Gavin R Lloyd. Support vector machines for classification and regression. *Analyst*, 135(2):230–267, 2010.
- [36] Giorgio Buttazzo. Reti Neurali in grado di apprendere. *Ithaca, Viaggio nella Scienza*, 16:202, 2020.
- [37] Giorgio C. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Springer, 2011.
- [38] Yujian Cai, Xingguang Li, and Jinsong Li. Emotion recognition using different sensors, emotion models, methods and datasets: A comprehensive review. *Sensors*, 23(5):2455, 2023.
- [39] Federico Calà, Mustafa Can Gursesli, Pietro Tarchi, Lorenzo Frassinetti, Andrea Guazzini, and Antonio Lanata. Pupil Dilation Dynamics in Serious Game Emotional Activation. In *2024 IEEE Gaming, Entertainment, and Media Conference (GEM)*, pages 1–4. IEEE, 2024.

- [40] Roberta Calegari, Giovanni Ciatto, and Andrea Omicini. On the integration of symbolic and sub-symbolic techniques for XAI: A survey. *Intelligenza Artificiale*, 14(1):7–32, 2020.
- [41] Luigi Capogrosso, Federico Cunico, Dong Seon Cheng, Franco Fummi, and Marco Cristani. A machine learning-oriented survey on tiny machine learning. *IEEE Access*, 12:23406–23426, 2024.
- [42] CENELEC. CENELEC EN 50128: Railway applications – Communication, signalling and processing systems – Software for railway control and protection systems. Technical Report EN 50126, Comitato Elettrotecnico Italiano (CEI), 2011. Versione del 2011.
- [43] CENELEC. CENELEC EN 50129: Railway applications – Safety related electronic systems for signalling. Technical Report EN 50129, Comitato Elettrotecnico Italiano (CEI), 2011. Versione del 2011.
- [44] CENELEC. CENELEC EN 50126: Railway applications - The specification and demonstration of reliability, availability, maintainability and safety (RAMS). Technical Report EN 50126, Comitato Elettrotecnico Italiano (CEI), 2017. Versione del 2017.
- [45] EN61508 CENELEC. 61508-functional safety of electrical /electronic /programmable electronic safety-related systems. *Book EN*, 61508, 2011.
- [46] EN61882 CENELEC. 61882-hazard and operability studies (hazop studies) - application guide. *Book EN*, 61882, 2016.
- [47] Bahzad Charbuty and Adnan Abdulazeez. Classification based on decision tree algorithm for machine learning. *Journal of applied science and technology trends*, 2(01):20–28, 2021.
- [48] Shruthi Chari, Oshani Seneviratne, Mohamed Ghalwash, Sola Shirai, Daniel M Gruen, Pablo Meyer, Prithwish Chakraborty, and Deborah L McGuinness. Explanation Ontology: A general-purpose, semantic representation for supporting user-centered explanations. *Semantic Web*, 15(4):959–989, 2024.

- [49] Shruthi Chari, Oshani Seneviratne, Daniel M Gruen, Morgan A Foreman, Amar K Das, and Deborah L McGuinness. Explanation ontology: a model of explanations for user-centered AI. In *International semantic web conference*, pages 228–243. Springer, 2020.
- [50] X. Chen et al. Process hazard analysis: Past, present, and future. *Journal of Loss Prevention in the Process Industries*, 40:1–12, 2016.
- [51] M. F. Chia and P. K. Naraharisetti. HAZOP using Stateflow software: Methodology and case study. *Process Safety and Environmental Protection*, 179:137–156, 2023.
- [52] Dong-Hoon Choi, Jin Seob Kim, Garry R Cutting, and Peter C Searson. Wearable potentiometric chloride sweat sensor: the critical role of the salt bridge. *Analytical chemistry*, 88(24):12241–12247, 2016.
- [53] Dong-Hoon Choi, Grant B Kitchen, Mark T Jennings, Garry R Cutting, and Peter C Searson. Out-of-clinic measurement of sweat chloride using a wearable sensor during low-intensity exercise. *NPJ digital medicine*, 3(1):49, 2020.
- [54] Dong-Hoon Choi, Yi Li, Garry R Cutting, and Peter C Searson. A wearable potentiometric sensor with integrated salt bridge for sweat chloride measurement. *Sensors and Actuators B: Chemical*, 250:673–678, 2017.
- [55] Mahesh Chowdhary and Swapnil Sayan Saha. On-Sensor Online Learning and Classification Under 8 KB Memory. In *2023 26th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2023.
- [56] Simona Colucci, Francesco Maria Donini, Verdiana Schena, Floriano Scioscia, and Eugenio Di Sciascio. LISE: a Logic-based Interactive Similarity Explainer for clusters of RDF data. *IEEE Access*, 2025.
- [57] Chiara Contoli and Emanuele Lattanzi. A Study on the Application of TensorFlow Compression Techniques to Human Activity Recognition. *IEEE Access*, 11:48046–48058, 2023.
- [58] Martina Curran, Enda Howley, and Jim Duggan. Classification of System Dynamics model outputs using decision trees. *Machine Learning with Applications*, page 100713, 2025.

- [59] Francesco Daghero, Alessio Burrello, Chen Xie, Luca Benini, Andrea Calimera, Enrico Macii, Massimo Poncino, and Daniele Jahier Pagliari. Adaptive random forests for energy-efficient inference on microcontrollers. In *2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6. IEEE, 2021.
- [60] T Daniya, Mithra Geetha, and K Suresh Kumar. Classification and regression trees with gini index. *Advances in Mathematics: Scientific Journal*, 9(10):8237–8247, 2020.
- [61] Mattia Daole, Alessio Schiavo, José Luis Corcuera Bárcena, Pietro Ducange, Francesco Marcelloni, and Alessandro Renda. OpenFL-XAI: Federated learning of explainable artificial intelligence models in Python. *SoftwareX*, 23:101505, 2023.
- [62] Soma Dasgupta and Swarnava Dey. MCU-based TinyML Camera for Face Recognition. In *2024 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 391–393. IEEE, 2024.
- [63] Daniela De Venuto, Domenica De Venuto, Giovanni Mezzina, Grazia Mascellaro, and Luisa Torsi. Chloride activity sensing in sweat for cystic fibrosis diagnosis by biocompatible flexible tag. In *Annual Meeting of the Italian Electronics Society*, pages 137–149. Springer, 2023.
- [64] Daniela De Venuto, Grazia Mascellaro, Giuseppe Spadavecchia, Domenica De Venuto, Paolo Bollella, and Luisa Torsi. Smart, Wearable and Power-Controlled Mixed-Signal Platform for Screening and Follow-Up of Cystic Fibrosis Based on Real-Time Chloride Concentration Evaluation in Sweat. *IEEE Sensors Journal*, 2024.
- [65] Daniela De Venuto, Giovanni Mezzina, Angelo Tricase, Paolo Bollella, Grazia Mascellaro, and Luisa Torsi. Wearable and flexible fibrosis cystic tag with potentiometric chloride activity sensing. In *2023 9th International Workshop on Advances in Sensors and Interfaces (IWASI)*, pages 115–120. IEEE, 2023.

- [66] Barry De Ville. Decision trees. *Wiley Interdisciplinary Reviews: Computational Statistics*, 5(6):448–455, 2013.
- [67] Fabrizio De Vita, Giorgio Nocera, Dario Bruneo, Valeria Tomaselli, and Mirko Falchetto. On-device training of deep learning models on edge microcontrollers. In *2022 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pages 62–69. IEEE, 2022.
- [68] Mohammad Mahdi Derakhshani, Ivona Najdenkoska, Tom van Sonsbeek, Xiantong Zhen, Dwarikanath Mahapatra, Marcel Worring, and Cees GM Snoek. Lifelonger: A benchmark for continual disease classification. In *International conference on medical image computing and computer-assisted intervention*, pages 314–324. Springer, 2022.
- [69] Miral M Desai and Jignesh J Patoliya. Design & implementation of data acquisition & controlling system based on free RTOS platform. *International Journal of Advanced Networking and Applications*, 10(2):3816–3821, 2018.
- [70] Alessandro Di Benedetto, G Narracci, CL Saragaglia, and D Serra. Autonomous Train Operations. *Applications in Electronics Pervading Industry, Environment and Society: APPLEPIES 2021*, 866:317, 2022.
- [71] Tommaso Di Noia, Eugenio Di Sciascio, and Francesco M Donini. Semantic matchmaking as non-monotonic reasoning: A description logic approach. *Journal of Artificial Intelligence Research*, 29:269–307, 2007.
- [72] Ente Italiano di Normazione. UNI CEI EN ISO 13485:2021. Technical report, UNI Ente Italiano di Normazione, 2021. Sistemi di gestione. Apparecchiature mediche. Apparecchiature mediche in generale. [1].
- [73] Ente Italiano di Normazione. UNI CEI EN ISO 14971:2022 con variante A11(2022). Technical report, UNI Ente Italiano di Normazione, 2022. Dispositivi medici - Applicazione della gestione dei rischi ai dispositivi medici.

- [74] Maha S Diab and Esther Rodriguez-Villegas. Embedded machine learning using microcontrollers in wearable and ambulatory systems for health and care applications: A review. *IEEE Access*, 10:98450–98474, 2022.
- [75] Pierpaolo Dini, Lorenzo Diana, Abdussalam Elhanashi, and Sergio Saponara. Overview of ai-models and tools in embedded iiot applications. *Electronics*, 13(12), 2024.
- [76] Simone Disabato and Manuel Roveri. Incremental On-Device Tiny Machine Learning. In *Proceedings of the 2nd International workshop on challenges in artificial intelligence and machine learning for internet of things*, pages 7–13, 2020.
- [77] P Doherty, J Kvarnström, V Lifschitz, F van Harmelen, and F Porter. The Handbook of Knowledge Representation. *Lifschitz, V., Van Harmelen, F., Porter, F.(eds.)*, pages 709–757, 2006.
- [78] Francesco M Donini, Maurizio Lenzerini, Daniele Nardi, Andrea Schaerf, et al. Reasoning in description logics. *Principles of knowledge representation*, 1:191–236, 1996.
- [79] Derek Doran, Sarah Schulz, and Tarek R Besold. What Does Explainable AI Really Mean? A New Conceptualization of Perspectives. In *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML, co-located with 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017)*. CEUR-WS, 2018.
- [80] Pietro Ducange, Francesco Marcelloni, Alessandro Renda, and Fabrizio Ruffini. Federated learning of XAI models in healthcare: a case study on Parkinson’s disease. *Cognitive Computation*, 16(6):3051–3076, 2024.
- [81] Rudresh Dwivedi, Devam Dave, Het Naik, Smiti Singhal, Rana Omer, Pankesh Patel, Bin Qian, Zhenyu Wen, Tejal Shah, Graham Morgan, et al. Explainable AI (XAI): Core ideas, techniques, and solutions. *ACM Computing Surveys*, 55(9):1–33, 2023.
- [82] Ehab El-Adawi, Ehab Essa, Mohamed Handosa, and Samir Elmougy. Wireless body area sensor networks based human activity recognition using deep learning. *Scientific Reports*, 14(1):2702, 2024.

- [83] Sam Emaminejad, Wei Gao, Eric Wu, Zoe A Davies, Hnin Yin Yin Nyein, Samyuktha Challa, Sean P Ryan, Hossain M Fahad, Kevin Chen, Ziba Shahpar, et al. Autonomous sweat extraction and analysis applied to cystic fibrosis and glucose monitoring using a fully integrated wearable platform. *Proceedings of the National Academy of sciences*, 114(18):4625–4630, 2017.
- [84] Clifton A Ericson et al. *Hazard analysis techniques for system safety*. John Wiley & Sons, 2015.
- [85] Kaouther Ezzameli and Hela Mahersia. Emotion recognition from unimodal to multimodal analysis: A review. *Information Fusion*, 99:101847, 2023.
- [86] Omobayode Fagbohungebe and Lijun Qian. The effect of batch normalization on noise resistant property of deep learning models. *IEEE Access*, 10:127728–127741, 2022.
- [87] Kazi Fatema, Samrat Kumar Dey, Mehrin Anannya, Risala Tasin Khan, Mohammad Mamunur Rashid, Chunhua Su, and Rashed Mazumder. Federated XAI IDS: An explainable and safeguarding privacy approach to detect intrusion combining federated learning and SHAP. *Future Internet*, 17(6):234, 2025.
- [88] Matthias Feurer and Frank Hutter. Hyperparameter optimization. In *Automated machine learning: Methods, systems, challenges*, pages 3–33. Springer International Publishing Cham, 2019.
- [89] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.
- [90] Han Gao, Shida Zhong, Lei Dong, and Tao Yuan. Performance Analysis of Machine Learning Algorithm on GD32 Microcontrollers. In *2023 6th International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pages 763–767. IEEE, 2023.
- [91] Maurizio Garbarino, Matteo Lai, Dan Bender, Rosalind W Picard, and Simone Tognetti. Empatica E3—A wearable wireless multi-sensor device for real-time computerized biofeedback and data acquisition. In *2014 4th international conference on wireless mobile communication and healthcare-transforming healthcare*

- through innovations in mobile and wireless technologies (MOBIHEALTH)*, pages 39–42. IEEE, 2014.
- [92] Christian Garbin, Xingquan Zhu, and Oge Marques. Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia tools and applications*, 79(19):12777–12815, 2020.
 - [93] Giorgos Giannakakis, Dimitris Grigoriadis, Katerina Giannakaki, Olympia Simantiraki, Alexandros Roniotis, and Manolis Tsiknakis. Review on psychological stress detection using biosignals. *IEEE transactions on affective computing*, 13(1):440–460, 2019.
 - [94] Javier Gonzalo-Ruiz, Roser Mas, Carmen de Haro, Enric Cabruja, Rafael Camero, M Asuncion Alonso-Lomillo, and F Javier Munoz. Early determination of cystic fibrosis by electrochemical chloride quantification in sweat. *Biosensors and bioelectronics*, 24(6):1788–1791, 2009.
 - [95] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.
 - [96] Nicola Guarino. *Formal ontology in information systems: Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy*, volume 46. IOS press, 1998.
 - [97] Guoqing Gui, Hong Pan, Zhibin Lin, Yonghua Li, and Zhijun Yuan. Data-driven support vector machine with optimization techniques for structural health monitoring and damage detection. *KSCE Journal of Civil Engineering*, 21(2):523–534, 2017.
 - [98] Stuti Gulati, Priyal Shah, Lakshin Pathak, Chinmay Trivedi, Rajesh Gupta, and Sudeep Tanwar. NanoXAI: Explainable AI for Anomaly Detection in Electromagnetic Nano-Networks of CAVs. In *2025 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, pages 103–108. IEEE, 2025.
 - [99] Kiriyaadhatshini Gunaratnam, Azma Abdullah, Rohani Abu Bakar, and Fadhl Hujainah. Hazard analysis techniques, methods and approaches: a review. *Inter-*

- national Journal of Advanced Research in Engineering Innovation*, 4(1):23–34, 2022.
- [100] Oliver Hahm, Emmanuel Baccelli, Hauke Petersen, and Nicolas Tsiftes. Operating systems for low-end devices in the internet of things: a survey. *IEEE Internet of Things Journal*, 3(5):720–734, 2015.
 - [101] Seung-Mi Ham, Hye-Min Lee, Jae-Hyun Lim, and Jeongwook Seo. A negative emotion recognition system with internet of things-based multimodal biosignal data. *Electronics*, 12(20):4321, 2023.
 - [102] Prasanna Hambarde, Rachit Varma, and Shivani Jha. The survey of real time operating system: RTOS. In *2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies*, pages 34–39. IEEE, 2014.
 - [103] Bo Han, Jiangchao Yao, Tongliang Liu, Bo Li, Sanmi Koyejo, Feng Liu, et al. Trustworthy machine learning: From data to models. *Foundations and Trends® in Privacy and Security*, 7(2-3):74–246, 2025.
 - [104] Jennifer A Healey and Rosalind W Picard. Detecting stress during real-world driving tasks using physiological sensors. *IEEE Transactions on intelligent transportation systems*, 6(2):156–166, 2005.
 - [105] Seungtae Hong, Gunju Park, and Jeong-Si Kim. Automated deep-learning model optimization framework for microcontrollers. *ETRI Journal*, 47(2):179–192, 2025.
 - [106] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.
 - [107] Ruihui Hou, Xiaojun Wang, Weiyan Zhang, Zhexin Song, Kai Wang, Yifei Chen, Jingping Liu, and Tong Ruan. Decision Tree Extraction for Clinical Decision Support System With If-Else Pseudocode and PlanSelect Strategy. *IEEE Journal of Biomedical and Health Informatics*, 2025.
 - [108] William W Hsieh. Evolution of machine learning in environmental science—A perspective. *Environmental Data Science*, 1:e3, 2022.
 - [109] Terence KL Hui and R Simon Sherratt. Coverage of emotion recognition for common wearable biosensors. *Biosensors*, 8(2):30, 2018.

- [110] IEEE. IEEE Standard for Local and metropolitan area networks - Part 15.6: Wireless Body Area Networks. <https://standards.ieee.org/ieee/802.15.6/5364/>, 2012.
- [111] International Electrotechnical Commission. IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems. Technical report, IEC, 2010. Standard.
- [112] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.
- [113] Andras Janosi, William Steinbrunn, Matthias Pfisterer, and Robert Detrano. Heart Disease. UCI Machine Learning Repository, 1989. DOI: <https://doi.org/10.24432/C52P4X>.
- [114] R Jegan and WS Nimi. On the development of low power wearable devices for assessment of physiological vital parameters: a systematic review. *Journal of Public Health*, 32(7):1093–1108, 2024.
- [115] Meera Joshi, Fahad M Iqbal, Mansour Sharabiani, Hutan Ashrafian, Sonal Arora, Kenny McAndrew, Sadia Khan, Graham Cooke, and Ara Darzi. Performance of Continuous Digital Monitoring of Vital Signs with a Wearable Sensor in Acute Hospital Settings. *Sensors*, 25(9):2644, 2025.
- [116] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.
- [117] Aleksei Karavaev, Jan Hejda, Patrik Kutilek, Petr Volf, Marek Sokol, and Lydie Leova. TinyDecisionTreeClassifier: Embedded C++ library for training and applying decision trees on the edge. *SoftwareX*, 27:101778, 2024.
- [118] Faten A Khalifa, Hatem M Abdelkader, and Asmaa H Elsaid. An analysis of ensemble pruning methods under the explanation of Random Forest. *Information Systems*, 120:102310, 2024.

- [119] G Khyathi, KP Indumathi, A Jumana Hasin, M Lisa Flavin Jency, G Krishnaprakash, FLAVIN JENCY M LISA, et al. Support Vector Machines: A Literature Review on Their Application in Analyzing Mass Data for Public Health. *Cureus*, 17(1), 2025.
- [120] Carl Kingsford and Steven L Salzberg. What are decision trees? *Nature biotechnology*, 26(9):1011–1013, 2008.
- [121] Trevor A Kletz. *Hazop & Hazan: identifying and assessing process industry hazards*. CRC Press, 2018.
- [122] Marcin Kołodziej, Andrzej Majkowski, and Marcin Jurczak. Acquisition and analysis of facial electromyographic signals for emotion recognition. *Sensors*, 24(15):4785, 2024.
- [123] Sotiris B Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39(4):261–283, 2013.
- [124] Christos Koulamas and Mihai T. Lazarescu. Real-Time Embedded Systems: Present and Future. *Electronics*, 7:205, 2018.
- [125] Dimitri Kraft, Karthik Srinivasan, and Gerald Bieber. Deep learning based fall detection algorithms for embedded systems, smartwatches, and IoT devices using accelerometers. *Technologies*, 8(4):72, 2020.
- [126] Prashant Kumar, Salman Khalid, and Heung Soo Kim. Prognostics and health management of rotating machinery of industrial robot with deep learning applications—A review. *Mathematics*, 11(13):3008, 2023.
- [127] Jean-Baptiste Lamy. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. *Artificial intelligence in medicine*, 80:11–28, 2017.
- [128] Phillip A. Laplante. *Real-Time Systems Design and Analysis*. Wiley, 2012.
- [129] Benoît Latré, Bart Braem, Ingrid Moerman, Chris Blondia, and Piet Demeester. A survey on wireless body area networks. *Wireless networks*, 17(1):1–18, 2011.
- [130] Justin Leo and Jugal Kalita. Survey of continuous deep learning methods and techniques used for incremental learning. *Neurocomputing*, 582:127545, 2024.

- [131] Jelin Leslin, Martin Trapp, and Martin Andraud. Hardware-efficient tractable probabilistic inference for TinyML Neurosymbolic AI applications. In *2025 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, pages 1–6. IEEE, 2025.
- [132] Chang Li, Yingchi Mao, Qian Huang, Weiliang Xie, Xiaoming He, and Jie Wu. A Real-Time Emotion-Aware System Based on Wireless Body Area Network for IoMT Applications. *IEEE Internet of Things Journal*, 2024.
- [133] Hailun Lian, Cheng Lu, Sunan Li, Yan Zhao, Chuangao Tang, and Yuan Zong. A survey of deep learning-based multimodal emotion recognition: Speech, text, and face. *Entropy*, 25(10):1440, 2023.
- [134] Jia Zheng Lim, James Mountstephens, and Jason Teo. Emotion recognition using eye-tracking: taxonomy, review and current challenges. *Sensors*, 20(8):2384, 2020.
- [135] Davide Loconte, Saverio Ieva, Grazia Mascellaro, Agnese Pinto, Giuseppe Loseto, Floriano Scioscia, and Michele Ruta. On-device explainable artificial intelligence for the semantic web of everything. *Future Generation Computer Systems*, 179:108310, 2026.
- [136] Davide Loconte, Saverio Ieva, Agnese Pinto, Giuseppe Loseto, Floriano Scioscia, and Michele Ruta. Expanding the cloud-to-edge continuum to the IoT in serverless federated learning. *Future Generation Computer Systems*, 155:447–462, 2024.
- [137] Wei-Yin Loh. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1):14–23, 2011.
- [138] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.
- [139] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, 2017.
- [140] Kuldeep Mahato and Joseph Wang. Electrochemical sensors: From the bench to the skin. *Sensors and Actuators B: Chemical*, 344:130178, 2021.

- [141] Ngoc-Dau Mai, Ha-Trung Nguyen, and Wan-Young Chung. Real-time on-chip machine-learning-based wearable behind-the-ear electroencephalogram device for emotion recognition. *Ieee Access*, 11:47258–47271, 2023.
- [142] Enni Mattern, Roxanne R Jackson, Roya Doshmanziari, Marieke Dewitte, Damiano Varagnolo, and Steffi Knorn. Emotion recognition from physiological signals collected with a wrist device and emotional recall. *Bioengineering*, 10(11):1308, 2023.
- [143] Ankur Maurya and Divya Kumar. Reliability of safety-critical systems: A state-of-the-art review. *Quality and Reliability Engineering International*, 36(7):2547–2568, 2020.
- [144] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.
- [145] S Baskiyar N Meghanathan. A survey of contemporary real-time operating systems. *Informatica*, 29(2), 2005.
- [146] Bjoern H Menze, B Michael Kelm, Ralf Masuch, Uwe Himmelreich, Peter Bachert, Wolfgang Petrich, and Fred A Hamprecht. A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data. *BMC bioinformatics*, 10(1):213, 2009.
- [147] Giovanni Mezzina, Arturo Amendola, Mario Barbareschi, Salvatore De Simone, Grazia Mascellaro, Alberto Moriconi, Cataldo Luciano Saragaglia, Diana Serra, and Daniela De Venuto. A Step Toward Safe Unattended Train Operations: A Pioneer Vital Control Module. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–4. IEEE, 2023.
- [148] Giovanni Mezzina, Mario Barbareschi, Salavatore De Simone, Alessandro Di Benedetto, G Narracci, Cataldo Luciano Saragaglia, Diana Serra, and Daniela De Venuto. Smart on-board surveillance module for safe autonomous train operations. In *International Conference on Applications in Electronics Pervading Industry, Environment and Society*, pages 317–325. Springer, 2021.

- [149] Laura Montanini, Antonio Del Campo, Davide Perla, Susanna Spinsante, and Ennio Gambi. A footwear-based methodology for fall detection. *IEEE Sensors Journal*, 18(3):1233–1242, 2017.
- [150] Luc Moreau and Paul Groth. *Provenance: an introduction to PROV*. Springer Nature, 2022.
- [151] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.
- [152] Sidra Naveed, Gunnar Stevens, and Dean Robin-Kern. An Overview of the Empirical Evaluation of Explainable AI (XAI): A Comprehensive Guideline for User-Centered Evaluation in XAI. *Applied Sciences*, 14(23):11288, 2024.
- [153] Nicholas Nethercote and Julian Seward. Valgrind: a framework for heavyweight dynamic binary instrumentation. *ACM Sigplan notices*, 42(6):89–100, 2007.
- [154] Emil Njor, Mohammad Amin Hasanpour, Jan Madsen, and Xenofon Fafoutis. A holistic review of the TinyML stack for predictive maintenance. *IEEE Access*, 2024.
- [155] Pierre J Obeid, Nouha Sari-Chmayssem, Paolo Yammine, Doris Homsy, Hanna El-Nakat, Zeinab Matar, Soumaya Hamieh, Diala Koumeir, and Ayman Chmayssem. Designs and Materials of Electrodes for Electrochemical Sensors. *ChemElectroChem*, page 2500230, 2025.
- [156] Franklin Oliveira, Daniel G Costa, Flávio Assis, and Ivanovitch Silva. Internet of Intelligent Things: A convergence of embedded systems, edge computing and machine learning. *Internet of Things*, page 101153, 2024.
- [157] Giovanni Pagliarini, Simone Scaboro, Giuseppe Serra, Guido Sciavicco, and Ionel Eduard Stan. Neural-symbolic temporal decision trees for multivariate time series classification. *Information and Computation*, 301:105209, 2024.
- [158] Maurizio Palumbo. Requirements Management for Safety Critical Systems, 2015.
- [159] Alexandros Pantelopoulos and Nikolaos G Bourbakis. A survey on wearable sensor-based systems for health monitoring and prognosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(1):1–12, 2009.

- [160] A Paramasivam, M Jenath, TS Sivakumaran, Sakthivel Sankaran, Pavan Sai Kiran Reddy Pittu, S Vijayalakshmi, et al. Development of artificial intelligence edge computing based wearable device for fall detection and prevention of elderly people. *Heliyon*, 10(8), 2024.
- [161] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.
- [162] Bijan Parsia, Sebastian Rudolph, Markus Krötzsch, Peter Patel-Schneider, and Pascal Hitzler. OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation, W3C, December 2012. <http://www.w3.org/TR/owl2-primer>.
- [163] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [164] Guanxiong Pei, Haiying Li, Yandi Lu, Yanlei Wang, Shizhen Hua, and Taihao Li. Affective computing: Recent advances, challenges, and future trends. *Intelligent Computing*, 3:0076, 2024.
- [165] Rosalind W Picard. Affective Computing. Technical report, MIT press Cambridge Massachusetts, 1997.
- [166] Rosalind W Picard. *Affective computing*. MIT press, 2000.
- [167] Agnese Pinto, Saverio Ieva, Arnaldo Tomasino, Giuseppe Loseto, Floriano Scioscia, Michele Ruta, and Francesco De Feudis. A Framework for Automatic Knowledge Base Generation from Observation Data Sets. In *International Conference on Web Engineering*, pages 89–100. Springer, 2023.
- [168] Antonios Pliatsios, Konstantinos Kotis, and Christos Goumopoulos. A systematic review on semantic interoperability in the IoE-enabled smart cities. *Internet of Things*, 22:100754, 2023.
- [169] Robi Polikar, Lalita Upda, Satish S Upda, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE transactions*

- on systems, man, and cybernetics, part C (applications and reviews)*, 31(4):497–508, 2001.
- [170] Pranav Prashant, Patel Archana, and Jain Sarika. *Machine learning in healthcare and security: advances, obstacles, and solutions*. CRC Press, 2024.
 - [171] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.
 - [172] Kanthavel Radhakrishnan, Dhaya Ramakrishnan, and R Adline Freeda. Federated Learning for Artificial Intelligence in Embedded Systems. *ICCK Transactions on Emerging Topics in Artificial Intelligence*, 2(2):91–115, 2025.
 - [173] Visal Rajapakse, Ishan Karunanayake, and Nadeem Ahmed. Intelligence at the Extreme Edge: A Survey on Reformable TinyML. *ACM Computing Surveys*, 55(13s):1–30, 2023.
 - [174] Marvin Rausand. *Reliability of safety-critical systems: theory and applications*. John Wiley & Sons, 2014.
 - [175] Kamran Razzaq and Mahmood Shah. Machine learning and deep learning paradigms: From techniques to practical applications and research frontiers. *Computers*, 14(3):93, 2025.
 - [176] Gopinath Rebala, Ajay Ravi, and Sanjay Churiwala. *An introduction to machine learning*. Springer, 2019.
 - [177] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “Why should i trust you?” Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.
 - [178] Wolfgang Roth, Günther Schindler, Bernhard Klein, Robert Peharz, Sebastian Tschitschek, Holger Fröning, Franz Pernkopf, and Zoubin Ghahramani. Resource-efficient neural networks for embedded systems. *Journal of Machine Learning Research*, 25(50):1–51, 2024.
 - [179] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education Limited, Harlow, England, 4 edition, 2021. Global Edition.

- [180] Michele Ruta, Eugenio Di Sciascio, and Floriano Scioscia. Concept Abduction and Contraction in Semantic-based P2P Environments. *Web Intelligence and Agent Systems*, 9(3):179–207, 2011.
- [181] Michele Ruta, Floriano Scioscia, Ivano Bilenchi, Filippo Gramegna, Giuseppe Loseto, Saverio Ieva, and Agnese Pinto. A multiplatform reasoning engine for the Semantic Web of Everything. *Journal of Web Semantics*, 73:100709, 2022.
- [182] Michele Ruta, Floriano Scioscia, Giuseppe Loseto, Agnese Pinto, and Eugenio Di Sciascio. Machine learning in the Internet of Things: A semantic-enhanced approach. *Semantic Web*, 10(1):183–204, 2019.
- [183] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.
- [184] Swapnil Sayan Saha, Sandeep Singh Sandha, Mohit Aggarwal, Brian Wang, Liying Han, Julian De Gortari Briseno, and Mani Srivastava. TinyNS: Platform-aware neurosymbolic auto tiny machine learning. *ACM Transactions on Embedded Computing Systems*, 23(3):1–48, 2024.
- [185] Swapnil Sayan Saha, Sandeep Singh Sandha, and Mani Srivastava. Machine learning for microcontroller-class hardware: A review. *IEEE Sensors Journal*, 22(22):21362–21390, 2022.
- [186] Fouad Sakr, Riccardo Berta, Joseph Doyle, Alessandro De Gloria, and Francesco Bellotti. Self-learning pipeline for low-energy resource-constrained devices. *Energies*, 14(20):6636, 2021.
- [187] Imrus Salehin and Dae-Ki Kang. A review on dropout regularization approaches for deep neural networks within the scholarly domain. *Electronics*, 12(14):3106, 2023.
- [188] Deepti Saraswat, Pronaya Bhattacharya, Ashwin Verma, Vivek Kumar Prasad, Sudeep Tanwar, Gulshan Sharma, Pitshou N Bokoro, and Ravi Sharma. Explainable AI for healthcare 5.0: opportunities and challenges. *IEEE Access*, 10:84486–84517, 2022.

- [189] Nikolaos Schizas, Aristeidis Karras, Christos Karras, and Spyros Sioutas. TinyML for ultra-low power AI and large scale IoT deployments: A systematic review. *Future Internet*, 14(12):363, 2022.
- [190] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.
- [191] Guus Schreiber and Fabien Gandon. RDF 1.1 XML Syntax. Recommendation, W3C, February 2014. <http://www.w3.org/TR/rdf-syntax-grammar/>.
- [192] Anjaneyulu Babu Shaik and Sujatha Srinivasan. A brief survey on random forest ensembles in classification model. In *International Conference on Innovative Computing and Communications: Proceedings of ICICC 2018, Volume 2*, pages 253–260. Springer, 2018.
- [193] Shaghayegh Shajari, Kirankumar Kuruvinashetti, Amin Komeili, and Uttandaraman Sundararaj. The emergence of AI-based wearable sensors for digital health technology: a review. *Sensors*, 23(23):9498, 2023.
- [194] Elham Ali Shammar and Ammar Thabit Zahary. The Internet of Things (IoT): a survey of techniques, operating systems, and trends. *Library Hi Tech*, 38(1):5–66, 2020.
- [195] FM Javed Mehedi Shamrat, Sovon Chakraborty, Md Masum Billah, Protiva Das, Jannatun Naeem Muna, and Rumesh Ranjan. A comprehensive study on pre-pruning and post-pruning methods of decision tree classification algorithm. In *2021 5th International conference on trends in electronics and informatics (ICOEI)*, pages 1339–1345. IEEE, 2021.
- [196] Shift2Rail Project. Shift2Rail Project — Innovation Programme on ATO over ETCS. <https://projects.shift2rail.org/>. Accessed Sept. 2025.
- [197] Abhigya Pote Shrestha and Chen-Hsiang Yu. ECG data analysis with IoT and machine learning. In *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0323–0327. IEEE, 2022.

- [198] Manan Shukla and Oshani Seneviratne. MentalHealthAI: Utilizing Personal Health Device Data to Optimize Psychiatry Treatment. In *AMIA Annual Symposium Proceedings*, volume 2023, page 641. American Medical Informatics Association, 2023.
- [199] Shruti Shukla, Madhavi Dachawar, Harshal Logade, and Sana Kadu. Self-Supervised Learning: The Core of Next-Gen Machine Learning and a Paradigm Shift in AI. In *2025 International Conference on Computational, Communication and Information Technology (ICCCIT)*, pages 486–491. IEEE, 2025.
- [200] Shriyank Somvanshi, Md Monzurul Islam, Gaurab Chhetri, Rohit Chakraborty, Mahmuda Sultana Mimi, Sawgat Ahmed Shuvo, Kazi Sifatul Islam, Syed Javed, Sharif Ahmed Rafat, Anandi Dutta, et al. From tiny machine learning to tiny deep learning: A survey. *ACM Computing Surveys*, 2025.
- [201] Russell Stuart and Norvig Peter. Artificial intelligence a modern approach third edition, 2010.
- [202] Bharath Sudharsan, John G Breslin, and Muhammad Intizar Ali. Edge2Train: A Framework to Train Machine Learning Models (SVMs) on Resource-Constrained IoT Edge Devices. In *Proceedings of the 10th International Conference on the Internet of Things*, pages 1–8, 2020.
- [203] Bharath Sudharsan, John G Breslin, and Muhammad Intizar Ali. ML-MCU: A Framework to Train ML Classifiers on MCU-Based IoT Edge Devices. *IEEE Internet of Things Journal*, 9(16):15007–15017, 2021.
- [204] Bharath Sudharsan, Piyush Yadav, John G Breslin, and Muhammad Intizar Ali. Train++: An Incremental ML Model Training Algorithm to Create Self-Learning IoT Devices. In *2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)*, pages 97–106. IEEE, 2021.
- [205] Enrico Tabanelli, Giuseppe Tagliavini, and Luca Benini. Optimizing random forest-based inference on RISC-V MCUs at the extreme edge. *IEEE Transactions on*

- Computer-Aided Design of Integrated Circuits and Systems*, 41(11):4516–4526, 2022.
- [206] Andrew S. Tanenbaum and Herbert Bos. *Modern Operating Systems*. Pearson, 2015.
- [207] K Tawsif, Nor Azlina Ab Aziz, J Emerson Raja, J Hossen, and MZH Jesmeen. A systematic review on emotion recognition system using physiological signals: data acquisition and methodology. *Emerging Science Journal*, 6(5):1167–1198, 2022.
- [208] My T Thai, Weili Wu, and Hui Xiong. *Big data in complex and social networks*. CRC Press, 2016.
- [209] Archana Tikayat Ray, Bjorn F Cole, Olivia J Pinon Fischer, Anirudh Prabhakara Bhat, Ryan T White, and Dimitri N Mavris. Agile methodology for the standardization of engineering requirements using large language models. *Systems*, 11(7):352, 2023.
- [210] UNISIG. SUBSET-041 — Performance Requirements for Interoperability. Technical specification, 2012.
- [211] Dirk Valkenburg, Axel-Jan Rousseau, Melvin Geubbelmans, and Tomasz Burzykowski. Support vector machines. *American journal of orthodontics and dentofacial orthopedics*, 164(5):754–757, 2023.
- [212] Gido M Van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022.
- [213] Leonardo Vanneschi and Sara Silva. Artificial neural networks. In *Lectures on Intelligent Systems*, pages 161–204. Springer, 2023.
- [214] Kush R Vashney. *Trustworthy machine learning*. Independently published, 2022.
- [215] Satvik Vats, Vikrant Sharma, Karan Singh, Anvesha Katti, Mazeyanti Mohd Ariffin, Mohammad Nazir Ahmad, Ali Ahmadian, and Soheil Salahshour. Incremental learning-based cascaded model for detection and localization of tuberculosis from chest x-ray images. *Expert Systems with Applications*, 238:122129, 2024.

- [216] W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview (Second Edition). W3C recommendation, W3C, December 2012. <https://www.w3.org/TR/owl2-overview/>.
- [217] Omar Abdel Wahab. Intrusion Detection in the IoT Under Data and Concept Drifts: Online Deep Learning Approach. *IEEE Internet of Things Journal*, 9(20):19706–19716, 2022.
- [218] Chen Wang, Jian Xu, Shanru Tan, and Long Yin. Secure decision tree classification with decentralized authorization and access control. *Computer Standards & Interfaces*, 89:103818, 2024.
- [219] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024.
- [220] Pengcheng Wang, Hui Xiong, and Haoxiang He. Bearing fault diagnosis under various conditions using an incremental learning-based multi-task shared classifier. *Knowledge-based systems*, 266:110395, 2023.
- [221] Xubin Wang, Zhiqing Tang, Jianxiong Guo, Tianhui Meng, Chenhao Wang, Tian Wang, and Weijia Jia. Empowering edge intelligence: A comprehensive survey on on-device ai models. *ACM Computing Surveys*, 57(9):1–39, 2025.
- [222] Yan Wang, Wei Song, Wei Tao, Antonio Liotta, Dawei Yang, Xinlei Li, Shuyong Gao, Yixuan Sun, Weifeng Ge, Wei Zhang, et al. A systematic review on affective computing: Emotion models, databases, and recent advances. *Information Fusion*, 83:19–52, 2022.
- [223] Philippe Westbroek, Georgios Priniotakis, and Paul Kiekens. *Analytical electrochemistry in textiles*. Elsevier, 2005.
- [224] Achmad Widodo and Bo-Suk Yang. Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical systems and signal processing*, 21(6):2560–2574, 2007.
- [225] William Wolberg, Olvi Mangasarian, and Nick Street. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1993. DOI: <https://doi.org/10.24432/C5DW2B>.

- [226] David Wood, Marsha Zaidman, Luke Ruth, and Michael Hausenblas. *Linked Data*. Manning publications co., 2014.
- [227] Jie Wu, Hong Liu, Weiwei Chen, Biao Ma, and Huangxian Ju. Device integration of electrochemical biosensors. *Nature Reviews Bioengineering*, 1(5):346–360, 2023.
- [228] Lars Wulfert, Johannes Kühnel, Lukas Krupp, Justus Viga, Christian Wiede, Pierre Gembaczka, and Anton Grabmaier. AIfES: A Next-Generation Edge AI Framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [229] Lars Wulfert, Christian Wiede, and Anton Grabmaier. TinyFL: On-Device Training, Communication and Aggregation on a Microcontroller for Federated Learning. In *2023 21st IEEE Interregional NEWCAS Conference (NEWCAS)*, pages 1–5. IEEE, 2023.
- [230] Gang Xu, Chen Cheng, Wei Yuan, Zhaoyang Liu, Lihang Zhu, Xintong Li, Yanli Lu, Zetao Chen, Jinglong Liu, Zheng Cui, et al. Smartphone-based battery-free and flexible electrochemical patch for calcium and chloride ions detections in biofluids. *Sensors and Actuators B: Chemical*, 297:126743, 2019.
- [231] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.
- [232] Shijie Zhang, Tao Tang, and Jintao Liu. A hazard analysis approach for the SOTIF in intelligent railway driving assistance systems using STPA and complex network. *Applied Sciences*, 11(16):7714, 2021.
- [233] J. Zhou, K. Hänninen, K. Lundqvist, and L. Provenzano. An ontological approach to hazard identification for safety-critical systems. In *2017 Second International Conference on Reliability Systems Engineering (ICRSE)*, pages 1–7. IEEE, 2017.
- [234] Tongtian Zhu. Analysis on the Applicability of the Random Forest. In *Journal of Physics Conference Series*, volume 1607 of *Journal of Physics Conference Series*, page 012123. IOP, August 2020.

List of publications

Peer-reviewed international journal articles

1. Daniela De Venuto, Grazia Mascellaro, Giuseppe Spadavecchia, Domenica De Venuto, Paolo Bollella, and Luisa Torsi. Smart, Wearable and Power-Controlled Mixed-Signal Platform for Screening and Follow-Up of Cystic Fibrosis Based on Real-Time Chloride Concentration Evaluation in Sweat. *IEEE Sensors Journal*, 2024
2. Davide Loconte, Saverio Ieva, Grazia Mascellaro, Agnese Pinto, Giuseppe Loseto, Floriano Scioscia, and Michele Ruta. On-device explainable artificial intelligence for the semantic web of everything. *Future Generation Computer Systems*, 179:108310, 2026

Peer-reviewed international conference papers

1. Giovanni Mezzina, Arturo Amendola, Mario Barbareschi, Salvatore De Simone, Grazia Mascellaro, Alberto Moriconi, Cataldo Luciano Saragaglia, Diana Serra, and Daniela De Venuto. A Step Toward Safe Unattended Train Operations: A Pioneer Vital Control Module. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–4. IEEE, 2023
2. Daniela De Venuto, Giovanni Mezzina, Angelo Tricase, Paolo Bollella, Grazia Mascellaro, and Luisa Torsi. Wearable and flexible fibrosis cystic tag with potentiometric chloride activity sensing. In *2023 9th International Workshop on Advances in Sensors and Interfaces (IWASI)*, pages 115–120. IEEE, 2023
3. Daniela De Venuto, Domenica De Venuto, Giovanni Mezzina, Grazia Mascellaro, and Luisa Torsi. Chloride activity sensing in sweat for cystic fibrosis diagnosis by

biocompatible flexible tag. In *Annual Meeting of the Italian Electronics Society*, pages 137–149. Springer, 2023

4. Ivano Bilenchi, Agnese Pinto, Grazia Mascellaro, Filippo Gramegna, Giuseppe Loseto, and Michele Ruta. Reasoning over Personal Health Knowledge Graph for healthcare monitoring on Apple Watch. In *The Inclusive Web: Realizing Safe, Accessible, Inclusive, and Sustainable Web Engineering (ICWE) - 3rd International Workshop on the Semantic WEb of EveryThing (SWEET)*, pages 85–97. Springer Nature Switzerland, 2026