



**Finanziato dall'Unione Europea**  
NextGenerationEU

*"Progetto Finanziato dall'Unione europea – NextGenerationEU"*



**Università  
di Brescia**

*DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE*

*DOTTORATO DI RICERCA IN TECHNOLOGY FOR HEALTH*

---

settore scientifico disciplinare  
ING-INF/07

CICLO  
XXXVIII

TITOLO TESI

Industrial-Grade Edge Computing and IoT Paradigms for Smart Mobility:  
Enabling Health and Safety Monitoring in Light Electric Vehicles

Massimiliano Gaffurini  
Matricola:  
719138

Primo SUPERVISORE

Paolo Ferrari, Università degli Studi di Brescia

Secondo SUPERVISORE

Salvatore Dello Iacono, Università degli Studi di Brescia

## **Abstract**

Il settore dei trasporti sta vivendo una trasformazione profonda, guidata dall'urgenza della decarbonizzazione e dal crescente riconoscimento del ruolo della mobilità attiva sulla salute pubblica. I Veicoli Elettrici Leggeri (LEV), tra cui e-bike e monopattini elettrici, si affermano come soluzione strategica per la mobilità urbana sostenibile, nel quadro del PNRR e del Centro Nazionale MOST (Spoke 5 – Veicoli Leggeri e Mobilità Attiva). Tuttavia, l'ecosistema LEV presenta criticità strutturali ancora aperte: rigidità delle architetture IoT di bordo, gestione di flussi di dati ad alta frequenza su dispositivi embedded a risorse limitate, complessità della raccolta di dati biometrici in contesti reali, assenza di sistemi autonomi di rilevamento degli incidenti e mancanza di metodologie per infrastrutture di ricarica off-grid sostenibili. Questa tesi colma tali lacune attraverso la trasposizione sistematica di paradigmi maturi nell'ambito industriale (Edge Computing, containerizzazione, TinyML e protocolli IoT a bassa potenza) nel dominio della Smart Mobility e della Technology for Health.

La tesi adotta un approccio architetturale basato sulla disaggregazione software tramite containerizzazione (Docker), integrato da un'infrastruttura di comunicazione multi-accesso (LoRaWAN e 5G Standalone). Per l'intelligenza di bordo, è stato sviluppato un prototipo di nodo edge modulare denominato "Black Box", capace di acquisire segnali inerziali (IMU a 103 Hz) e dati GNSS, con successiva elaborazione e trasmissione on-device. Due strategie complementari di gestione dei dati sono state validate: un modello di compressione basato su Autoencoder e algoritmi incrementali di anomaly detection (MST e MPT), entrambi ottimizzati per l'esecuzione su dispositivi embedded tramite TinyML. La validazione sul campo ha incluso una campagna di acquisizione biometrica durante

---

le 20 tappe del GiroE-2024 e metodologie di simulazione per stazioni di ricarica fotovoltaiche off-grid.

I risultati sperimentali confermano la validità dell'approccio. La virtualizzazione leggera introduce overhead trascurabile, con tempi di ciclo fino a 2 ms validati in ambito industriale e compatibili con le latenze richieste in scenari di Smart Mobility. Il prototipo Black Box raggiunge 8 ore di autonomia operativa, con un consumo medio di 2 W e di picco di 7 W (durante l'acquisizione dati), e una validità GPS del 96,17%. Gli algoritmi TinyML di compressione basati su Autoencoder raggiungono un rapporto di compressione dell'80%, con un F1 score vicino al riferimento del 90% durante la successiva validazione tramite un test di classificazione, validato su dati OBD automotive; gli algoritmi di anomaly detection MST e MPT raggiungono rispettivamente un F1 score del 99,84% e del 33,67%, gli errori di correzione confermano che le correzioni del MST sono localizzate e conservative mentre le correzioni MPT risultano meno localizzate per via della sostituzione dell'intero campione in caso di canali anomali. Si è studiato il consumo energetico di un sistema per la fall detection, operante con 12  $\mu$ A in idle garantendo un'autonomia superiore a 87 giorni e latenza media di allerta di 221 ms. Il dataset biometrico raccolto durante il GiroE-2024, pari a 180,93 ore e 3.749,84 km su 20 tappe, costituisce la prima raccolta pubblica su larga scala di parametri fisiologici da ciclisti non professionisti. I risultati dimostrano che i paradigmi industriali possono essere trasposti con successo su piattaforme mobili embedded, confermando modularità software e intelligenza distribuita all'Edge come requisiti abilitanti per ecosistemi di mobilità leggera sicuri, scalabili e mantenibili.

## **Abstract**

The transportation sector is undergoing a profound transformation driven by the urgency of decarbonization and the growing recognition of the role that active mobility plays in public health. Light Electric Vehicles (LEVs), including e-bikes and electric scooters, are emerging as a strategic solution for sustainable urban mobility within the Italian NRRP/PNRR framework and the MOST National Center for Sustainable Mobility (Spoke 5 – Light Vehicles and Active Mobility). However, the current LEV ecosystem presents unresolved structural challenges: rigid on-board IoT architectures, management of high-frequency data streams on resource-constrained embedded devices, complexity of biometric data collection in real-world conditions, absence of autonomous incident-detection systems, and lack of methodologies for sustainable off-grid charging infrastructures. This thesis addresses these gaps through the systematic transposition of mature industrial paradigms (Edge Computing, containerization, TinyML, and low-power IoT protocols) into the domain of Smart Mobility and Technology for Health.

The thesis adopts an architectural approach based on software disaggregation through containerization (Docker), integrated with a multi-access communication infrastructure (LoRaWAN and 5G Standalone). For on-board intelligence, a modular edge node prototype called the “Black Box” was developed, capable of acquiring high-frequency inertial signals (IMU at 103 Hz) and GNSS data, with subsequent on-device processing and transmission. Two complementary on-device data management strategies were validated: an Autoencoder-based compression model and incremental anomaly detection algorithms (MST and MPT), both optimized for execution on embedded devices through TinyML. Field validation included a biometric data acquisition campaign across all 20 stages

---

of the GiroE-2024 national e-bike tour and simulation-based methodologies for photovoltaic-powered off-grid charging stations.

The experimental results confirm the validity of the approach. Lightweight virtualization introduces negligible overhead, with cycle times as low as 2 ms validated in the industrial domain and compatible with the latency requirements of Smart Mobility scenarios. The Black Box prototype achieves 8 hours of operational autonomy, with an average power consumption of 2 W and a peak of 7 W (during data acquisition), and a GPS validity rate of 96.17%. The Autoencoder-based TinyML compression algorithms achieve an 80% compression ratio, with an F1 score close to the 90% reference during subsequent classification validation on automotive OBD data; the MST and MPT anomaly detection algorithms reach an F1 score of 99.84% and 33.67% respectively, with correction error analysis confirming that MST corrections are localized and conservative, while MPT corrections are less localized due to full-sample substitution in the presence of anomalous channels. The energy consumption of a fall detection system was also studied, operating at 12  $\mu$ A idle current and guaranteeing a battery lifetime exceeding 87 days and a median alert latency of 221 ms. The biometric dataset collected during the GiroE-2024, totaling 180.93 hours and 3,749.84 km across 20 stages, represents the first large-scale publicly available physiological dataset from non-professional e-bike riders. The results demonstrate that industrial paradigms can be successfully transposed to mobile embedded platforms, confirming software modularity and distributed Edge intelligence as fundamental enabling requirements for safe, scalable, and maintainable light mobility ecosystems.

# Contents

|          |                                                                                                |           |
|----------|------------------------------------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                                                            | <b>1</b>  |
| 1.1      | Context and Motivation: The Rise of Sustainable Mobility . . . .                               | 1         |
| 1.2      | Problem Statement . . . . .                                                                    | 2         |
| 1.3      | The Technological Bridge: From Industry 4.0 to personal safety .                               | 6         |
| 1.4      | Research Objectives and Contributions . . . . .                                                | 10        |
| 1.5      | Thesis Structure . . . . .                                                                     | 13        |
| <b>2</b> | <b>Opportunities of Industry 4.0 in Smart Mobility: Background and Industrial Case Studies</b> | <b>15</b> |
| 2.1      | Introduction to Edge Computing . . . . .                                                       | 16        |
| 2.1.1    | Edge Computing Architecture . . . . .                                                          | 17        |
| 2.1.2    | Cloud Computing vs. Edge Computing . . . . .                                                   | 18        |
| 2.1.3    | Advantages of Edge Computing . . . . .                                                         | 19        |
| 2.1.4    | Representative Use Cases . . . . .                                                             | 20        |
| 2.1.5    | Virtualization . . . . .                                                                       | 22        |
| 2.1.6    | Case Studies in Industry 4.0 . . . . .                                                         | 26        |
| 2.2      | IoT Communication Protocols for Smart Mobility . . . . .                                       | 44        |
| 2.2.1    | LPWAN - LoRaWAN . . . . .                                                                      | 44        |
| 2.2.2    | 5G Mobile Networks . . . . .                                                                   | 51        |

|          |                                                                                        |           |
|----------|----------------------------------------------------------------------------------------|-----------|
| 2.3      | Summary . . . . .                                                                      | 54        |
| <b>3</b> | <b>IoT Communication Protocols in Smart Mobility Scenarios</b>                         | <b>55</b> |
| 3.1      | LoRaWAN End-Device Disaggregation via<br>Lightweight virtualization . . . . .          | 56        |
| 3.1.1    | Proposed Architectures . . . . .                                                       | 57        |
| 3.1.2    | Experimental Evaluation . . . . .                                                      | 60        |
| 3.2      | End-to-End Emulation of LoRaWAN Infrastructure for Smart City<br>Scenarios . . . . .   | 66        |
| 3.2.1    | Reference Smart-City Scenario . . . . .                                                | 66        |
| 3.2.2    | Emulation Architecture . . . . .                                                       | 67        |
| 3.2.3    | Performance Metrics . . . . .                                                          | 68        |
| 3.2.4    | Experimental Platforms . . . . .                                                       | 69        |
| 3.2.5    | Results . . . . .                                                                      | 71        |
| 3.3      | A Software-Defined LoRaWAN Node Based on SDR . . . . .                                 | 74        |
| 3.3.1    | Architecture of the Software-Defined Node . . . . .                                    | 75        |
| 3.3.2    | Relation to the Hybrid Architecture . . . . .                                          | 76        |
| 3.3.3    | Experimental Setup . . . . .                                                           | 77        |
| 3.4      | 5G Standalone Private Network for Smart Campus Multi-Access<br>Communication . . . . . | 81        |
| 3.4.1    | Experimental Setup . . . . .                                                           | 81        |
| 3.4.2    | Results . . . . .                                                                      | 83        |
| 3.5      | Summary . . . . .                                                                      | 86        |
| <b>4</b> | <b>The Core Monitoring Platforms</b>                                                   | <b>87</b> |
| 4.1      | Design of the Black Box Prototype . . . . .                                            | 88        |
| 4.1.1    | Processing Unit . . . . .                                                              | 89        |
| 4.1.2    | Power Section . . . . .                                                                | 90        |

|          |                                                                                                                     |            |
|----------|---------------------------------------------------------------------------------------------------------------------|------------|
| 4.1.3    | Communication Modules . . . . .                                                                                     | 90         |
| 4.1.4    | Sensors and Actuators . . . . .                                                                                     | 91         |
| 4.1.5    | Prototype Implementation . . . . .                                                                                  | 92         |
| 4.1.6    | Preliminary Experimental Results . . . . .                                                                          | 94         |
| 4.2      | Autoencoder-Based Compression of Vehicular Sensor Data . . .                                                        | 97         |
| 4.2.1    | Autoencoder Variants and Architectures . . . . .                                                                    | 97         |
| 4.2.2    | Dataset and Preprocessing . . . . .                                                                                 | 99         |
| 4.2.3    | Model Selection and Embedded Deployment . . . . .                                                                   | 100        |
| 4.2.4    | Results . . . . .                                                                                                   | 100        |
| 4.3      | Lightweight Incremental Algorithms for Multivariate Anomaly<br>Detection and Correction on TinyML Devices . . . . . | 103        |
| 4.3.1    | Theoretical Background: TEDA and RLS . . . . .                                                                      | 103        |
| 4.3.2    | Proposed Algorithms: MST and MPT . . . . .                                                                          | 105        |
| 4.3.3    | Experimental Setup and Results . . . . .                                                                            | 106        |
| 4.4      | Summary . . . . .                                                                                                   | 109        |
| <b>5</b> | <b>Technology for Health: Safety and Biometric Monitoring</b>                                                       | <b>110</b> |
| 5.1      | Biometric Data Collection in Cycling Events . . . . .                                                               | 111        |
| 5.1.1    | Study Context: the GiroE-2024 . . . . .                                                                             | 112        |
| 5.1.2    | Data Acquisition System Architecture . . . . .                                                                      | 112        |
| 5.1.3    | Methodological Challenges and Scientific Value . . . . .                                                            | 116        |
| 5.2      | Spatial Aggregation of IoT Mobility Data . . . . .                                                                  | 118        |
| 5.2.1    | Reference Node Construction . . . . .                                                                               | 118        |
| 5.2.2    | Aggregation Pipeline . . . . .                                                                                      | 119        |
| 5.2.3    | Experimental Results . . . . .                                                                                      | 120        |
| 5.2.4    | Discussion and Application Perspectives . . . . .                                                                   | 121        |
| 5.3      | Summary . . . . .                                                                                                   | 123        |

|          |                                                                                    |            |
|----------|------------------------------------------------------------------------------------|------------|
| <b>6</b> | <b>Energy Sustainability and Off-Grid Infrastructures</b>                          | <b>124</b> |
| 6.1      | Fall Detection Systems for Light Mobility . . . . .                                | 124        |
| 6.1.1    | System Architecture . . . . .                                                      | 125        |
| 6.1.2    | Experimental Characterisation . . . . .                                            | 127        |
| 6.1.3    | Discussion . . . . .                                                               | 130        |
| 6.2      | ICT-Equipped Portable E-Bike Charging Station for Mass Cycling<br>Events . . . . . | 131        |
| 6.2.1    | System Design and Constraints . . . . .                                            | 131        |
| 6.2.2    | Simulation Methodology . . . . .                                                   | 133        |
| 6.2.3    | Results and ICT Sizing Guidelines . . . . .                                        | 134        |
| 6.3      | Techno-Environmental Assessment of<br>PV-Powered Charging Stations . . . . .       | 135        |
| 6.3.1    | Use Case and System Architecture . . . . .                                         | 135        |
| 6.3.2    | Simulation Framework . . . . .                                                     | 136        |
| 6.3.3    | GHG Emission Model and Key Performance Indicators . . . . .                        | 137        |
| 6.3.4    | Results . . . . .                                                                  | 138        |
| 6.3.5    | Discussion . . . . .                                                               | 139        |
| 6.4      | Summary . . . . .                                                                  | 140        |
| <b>7</b> | <b>Conclusion and Future Work</b>                                                  | <b>142</b> |
| 7.1      | Revisiting the Problem Statement . . . . .                                         | 142        |
| 7.2      | Summary of Contributions . . . . .                                                 | 143        |
| 7.3      | General Discussion: A Validated Transposition . . . . .                            | 144        |
| 7.4      | Limitations . . . . .                                                              | 146        |
| 7.5      | Future Work . . . . .                                                              | 147        |
| <b>A</b> | <b>Appendix</b>                                                                    | <b>148</b> |
| A.1      | Siemens Industrial Edge . . . . .                                                  | 148        |

---

|     |                             |     |
|-----|-----------------------------|-----|
| A.2 | S7Comm Protocol . . . . .   | 150 |
| A.3 | PROFINET Protocol . . . . . | 152 |

# Chapter 1

## Introduction

### **1.1 Context and Motivation: The Rise of Sustainable Mobility**

The health and well-being of individuals are shaped not only by direct medical factors, but also by the broader environmental and social systems in which people live. Among these, urban mobility plays an increasingly recognized role: sedentary lifestyles, air pollution, noise, and road accidents are all direct consequences of car-dominated transportation models, and all carry measurable impacts on public health. Active and sustainable mobility, by contrast, has been consistently associated with improved cardiovascular health, reduced stress, lower exposure to pollutants, and greater social inclusion. It is within this framing that the present thesis positions itself: not merely as a contribution to transportation engineering, but as a response to a health imperative. The global imperative to mitigate climate change has placed the decarbonisation of the transportation sector at the forefront of international policy. As urban centres face increasing congestion and environmental degradation, Light Electric Vehicles (LEVs), in-

cluding e-bikes and electric scooters, have emerged as a pivotal solution for both urban commuting and sustainable tourism.

Within the Italian framework, the National Recovery and Resilience Plan (NRRP/PNRR) has recognized this shift by funding the MOST National Centre for Sustainable Mobility, specifically Spoke 5, which is dedicated to Light Vehicles and Active Mobility. The mission of Spoke 5 is to transform light vehicles from simple mechanical tools into intelligent, connected nodes within a smart city ecosystem. However, the transition to widespread LEV adoption faces significant hurdles, including range anxiety, a lack of standardized charging infrastructure, and critical safety concerns for vulnerable road users. Addressing these hurdles is not only a matter of technological convenience: removing barriers to LEV adoption means making active mobility accessible to a broader population, with direct positive effects on individual health outcomes and on the collective burden of chronic diseases linked to physical inactivity and urban pollution. This thesis addresses these challenges by proposing a technological framework that bridges the gap between robust industrial automation and personal mobility, with the ultimate goal of making light electric vehicles safer, more reliable, and more conducive to the health and well-being of their users.

## 1.2 Problem Statement

Despite the growing adoption of LEVs and the broader momentum behind micro-mobility, several critical gaps referred to as Problem Statement (PS) remain in the current ecosystem, spanning the dimensions of safety, system design, data management, infrastructure, and health monitoring. Together, they collectively limit the ability to deploy reliable, scalable, and user-centric LEV solutions in real-world conditions.

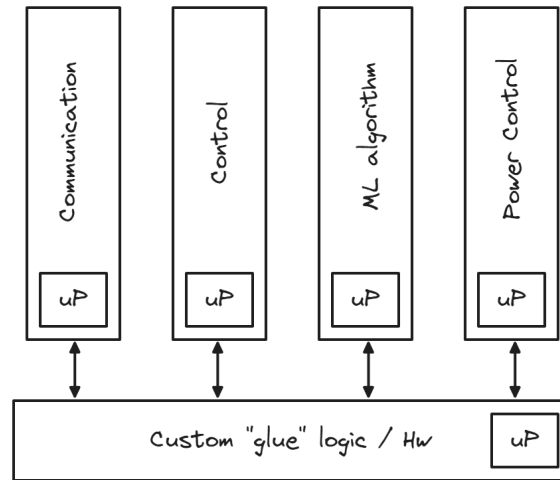


Figure 1.1: Actual view of the Problem Statement landscape

Figure 1.1 illustrates the architectural fragmentation that characterizes the current state of on-board IoT systems in the LEV ecosystem. Each functional domain (Communication, Control, ML algorithm, and Power Control) is realized as a vertically isolated stack, in which a dedicated microprocessor ( $\mu\text{P}$ ) is tightly coupled to its specific software function. All these stacks ultimately converge on a layer of custom “glue” logic and hardware, which acts as the sole integration point for the entire system. This architecture exposes three critical limitations. First, the one-to-one binding between a  $\mu\text{P}$  and a single function makes the system inherently rigid: updating or replacing one component requires redesigning the surrounding hardware and custom logic, slowing down the development cycle and raising long-term maintenance costs. Second, the proliferation of independent microprocessors, each managing its own resources, timings, and interfaces, introduces significant integration complexity: any attempt to add a new functionality or to port the system to a different vehicle platform requires re-engineering the glue layer from scratch, rather than simply

deploying a new software module. Third, the reliance on custom hardware integration prevents the adoption of standard, off-the-shelf development practices, leaving no shared abstraction layer, no common execution environment, and no possibility of hardware-independent software reuse. The result is an ecosystem in which each LEV platform tends to be a closed, one-off solution, a concrete instance of the broader modularity and flexibility gap identified as (PS 1), and a recurring structural theme across the remaining Problem Statements described below.

- (PS 1) Lack of Flexibility and Modularity in On-Board IoT Systems: current embedded solutions are based on common IoT devices that tend to be tightly coupled to specific hardware platforms, making them difficult to update, maintain, or port across different vehicle types. This rigidity slows down the development cycle, raises long-term maintenance costs, and hinders the adoption of new functionalities. A more modular and hardware-independent approach to software design is needed to support the rapid evolution of LEV platforms and the diversity of deployment contexts.
- (PS 2) Data Volume and Quality on Resource-Constrained Devices: LEVs are undergoing a technological evolution that closely mirrors the one experienced by the automotive sector over the past decades: from purely mechanical platforms to increasingly instrumented systems equipped with inertial sensors, GNSS receivers, biometric wearables, and real-time control units. However, unlike automobiles, where this transition is supported by progressively more powerful onboard computers, dedicated communication modules, and standardized diagnostic interfaces, LEVs operate under far tighter constraints in terms of computational capacity, memory, energy budget, and communication bandwidth. As the number of sensors

grows, so does the volume of continuous, high-frequency multivariate data streams generated on board, which rapidly exceeds the storage and transmission capacity of the embedded hardware typically adopted in this domain. Forwarding raw data to remote systems is often impractical due to power limitations, duty-cycle restrictions on wireless links, and the intermittent connectivity characteristic of outdoor mobility scenarios. At the same time, ensuring data integrity and detecting anomalies in real time is essential for both safety and performance monitoring. This tension between data richness and hardware constraints calls for new approaches capable of managing, compressing, and validating sensor data directly on-device, without delegating these tasks to cloud infrastructure.

- (PS 3) Complexity of Biometric and Data Collection in Real-World Conditions: monitoring the health and physical performance of LEV riders in naturalistic settings is significantly more challenging than in controlled laboratory environments. Data from multiple wearable and on-board devices is inherently asynchronous, subject to connectivity dropouts, and influenced by confounding factors such as motor assistance or terrain variation. Furthermore, the spatial dimension of mobility data adds a layer of complexity that requires dedicated frameworks to meaningfully aggregate and interpret physiological observations across heterogeneous routes and user profiles.
- (PS 4) Safety and Timely Accident Response: light vehicles are routinely operated in settings where rider safety cannot rely on the presence of bystanders or constant network connectivity. Unlike motor vehicles, LEVs carry no embedded systems capable of autonomously detecting incidents and alerting emergency services. This gap leaves riders exposed in the event of a fall or

sudden health crisis, particularly in remote or low-coverage environments, where riders are known to navigate road-safety hazards with little technological support [1]. There is therefore a need for on-board, autonomous safety mechanisms that can guarantee timely emergency response regardless of infrastructure availability.

(PS 5) Energy Sustainability and Resilience of Charging Infrastructures. Extending LEV adoption to rural, touristic, and off-grid areas requires charging solutions that are independent from the main power grid. Deploying such infrastructure in a sustainable and cost-effective way poses open questions around energy availability, sizing of monitoring hardware, and trade-offs between service quality and energy consumption. The lack of established methodologies for the design and assessment of off-grid charging stations represents a concrete barrier to the widespread rollout of LEV infrastructure.

### **1.3 The Technological Bridge: From Industry 4.0 to personal safety**

The five Problem Statement identified in Section 1.2 share a common underlying trait: they all require systems that are simultaneously reliable, autonomous, resource-efficient, and capable of operating in uncontrolled, real-world environments. Individually, each challenge could be addressed with ad-hoc solutions; taken together, they call for a coherent architectural vision. The core thesis of this work is that such a vision already exists, not in the mobility domain, but in industrial automation. Industry 4.0 and, more recently, Industry 5.0 have driven the development of paradigms and technologies specifically designed to tackle

reliability, modularity, data management, and energy efficiency in constrained, distributed environments. The proposition of this research is therefore to deliberately transpose these industrial paradigms into the realm of Smart Mobility and Technology for Health. The parallel is more direct than it might appear.

The need for modular, hardware-independent software architectures on LEV platforms (PS 1) mirrors a transformation that the industrial sector has already undergone. The advent of Industry 4.0 introduced the imperative of interconnecting heterogeneous machines, sensors, and control systems within the same production environment: everything needed to communicate with everything else, across different vendors, protocols, and generations of hardware. This drove the development of flexible, reconfigurable software architectures capable of abstracting away the underlying hardware, ultimately giving rise to Edge Computing platforms based on virtualized and containerized execution environments. The same need for interoperability, flexibility, and hardware independence is now emerging in the LEV domain, where vehicles, charging stations, and monitoring systems must coexist and exchange data within a common ecosystem.

The problem of managing sensor streams on embedded devices with limited resources (PS 2) has a direct industrial precedent. As manufacturing systems evolved, the information exchanged on the factory floor progressively expanded beyond real-time process control data to include monitoring and diagnostic information, such as vibration signatures, thermal profiles, and energy consumption traces. This data, non-time-critical in nature, is increasingly collected and transmitted through non-real-time protocols, both wired and wireless, often via dedicated IoT devices installed alongside existing automation hardware. The challenge is no longer only to control processes in real time, but also to efficiently acquire, compress, and route large volumes of monitoring data from resource-constrained nodes to higher-level analytics systems. LEVs face the same struc-

tural challenge, with the added complication of operating in highly dynamic, mobile, and energy-limited environments.

The complexity of collecting and aggregating biometric data from LEV users in heterogeneous field conditions (PS 3) extends this trajectory further. Once industrial IoT devices began producing rich monitoring streams, the next challenge became making sense of data collected from multiple, geographically distributed, and asynchronous sources. Fusing telemetry from different assets, aligning it in time and space, and extracting meaningful indicators required dedicated data aggregation frameworks. In the LEV and health monitoring context, this challenge takes on an additional spatial dimension: physiological observations must be interpreted not only across multiple riders and devices, but also in relation to the physical route, the terrain, and the varying conditions encountered along the way.

The safety and responsiveness required to detect an emergency situation and dispatch an alert (PS 4) are not fundamentally different from the real-time control requirements of an automated production line. In industrial environments, safety has long been treated as a first-class design requirement rather than an afterthought: dedicated safety protocols such as PROFIsafe and FSoE (Fail Safe over EtherCAT) are developed precisely to guarantee that safety-critical signals, whether an emergency stop or a guard door status, are transmitted and acted upon within deterministic time bounds, even in the presence of communication faults. Crucially, industrial safety systems are designed to protect not only the operators but also the assets themselves, since an undetected fault can cause damage to machinery, production loss, or cascading failures across the line. The same dual imperative applies to LEVs: an undetected fall endangers the rider, but a delayed or missed alert also compromises the integrity of the broader monitoring infrastructure and erodes user trust in the system. Transposing this safety-

first design philosophy to the mobility domain means embedding reliability and fault tolerance directly into the on-board architecture, rather than delegating them to cloud connectivity that may not always be available.

Finally, the need for energy-sustainable, off-grid charging infrastructures (PS 5) reflects a challenge that the industrial sector has already confronted in a different but structurally analogous form. As manufacturing plants have progressively integrated renewable energy sources and moved towards more distributed production models, the energy footprint of instrumentation, control, and communication hardware has become an explicit design variable rather than a secondary concern. In smart factories, the power consumed by sensors, gateways, edge nodes, and communication modules must be accounted for in the overall energy budget of the plant, particularly in contexts where energy costs are high or supply is constrained. The transition towards Industry 5.0 has further reinforced this imperative by placing sustainability at the centre of the design process, alongside productivity and flexibility. Off-grid LEV charging stations face the same challenge in a more acute form: the energy harvested from photovoltaic panels is finite and variable, and every watt allocated to the ICT monitoring subsystem is a watt unavailable for vehicle charging.

This transposition is not merely conceptual. The industrial domain has produced mature, validated solutions (edge computing architectures, lightweight virtualization, low-power wide-area communication protocols, embedded machine learning) that can be adapted and applied to the LEV context. Crucially, the industrial background also provides a rigorous performance baseline: the case studies presented in Chapter 2 demonstrate what these paradigms can achieve on factory-floor hardware, and the experimental contributions of Chapters 3 through 6 systematically evaluate whether equivalent performance can be attained on the resource-constrained platforms typical of light mobility.

## 1.4 Research Objectives and Contributions

Building on the Problem Statement outlined in Section 1.2 and the industrial approach introduced in Section 1.3, this thesis pursues five research objectives, each directly addressing one of the identified problems and contributing an original experimental validation.

- **Modular and Disaggregated Edge Architectures for LEV Platforms (PS 1).** The first objective is to demonstrate that the software disaggregation and virtualization paradigms developed in industrial automation can be successfully transposed to the LEV domain. This entails designing and validating containerized, hardware-independent architectures for on-board IoT nodes and communication infrastructure, and evaluating their performance against the baseline established by the industrial case studies of Chapter 2.
- **Embedded Data Management under Resource and Connectivity Constraints (PS 2).** The second objective is to address the tension between the growing volume of sensor data generated on board LEVs and the limited computational, memory, and communication resources available on embedded hardware. This thesis proposes and validates algorithms for on-device data compression and anomaly detection that operate without relying on cloud infrastructure, drawing on the same design principles adopted for condition-monitoring in industrial IoT systems.
- **Complexity of Biometric and Data Collection in Real-World Conditions (PS 3).** The third objective is to design and deploy a data acquisition framework capable of collecting physiological and data from LEV riders in naturalistic and large-scale settings. This objective extends the industrial data fusion

paradigm to the health monitoring domain, where asynchronous, multi-device streams must be aligned along physical routes rather than production timelines.

- Safety and Timely Accident Response (PS 4). The fourth objective is to develop and validate an on-board safety monitoring system capable of autonomously detecting emergency situations and dispatching alerts, independently of network availability. The system must satisfy the same reliability and latency requirements imposed on safety-critical functions in industrial automation, while operating within the tight energy budget of a battery-powered, mobile platform.
- Energy Sustainability and Resilience of Charging Infrastructures (PS 5). The fifth objective is to establish methodologies for the design, simulation, and techno-environmental assessment of PV-powered e-bike charging stations, with explicit consideration of the power budget allocated to the embedded ICT monitoring subsystem. This objective mirrors the industrial practice of accounting for the energy footprint of instrumentation and control hardware as an integral part of system design.

It is useful to distinguish, among these objectives, the methodological contributions from the system-level integration outcomes. The methodological (scientific) contributions are the novel algorithms and models proposed and validated here: the Autoencoder-based multivariate compression scheme and the incremental MST/MPT anomaly-detection algorithms for TinyML-class devices (PS 2), and the graph-based spatial aggregation pipeline for asynchronous multi-rider physiological data (PS 3). The integration outcomes, by contrast, demonstrate that mature industrial paradigms can be transposed and engineered into a coherent, working system under LEV constraints: the disaggregated containerised

LoRaWAN/5G communication stack (PS 1), the Black Box edge node (PS 1, PS 2), the autonomous fall-detection subsystem (PS 4), and the PV-charging assessment methodology (PS 5). Both classes of contribution are validated experimentally, but only the former claim algorithmic novelty; the latter claim feasibility and quantitative parity with the industrial baseline.

Figure 1.2 presents the proposed architectural response to the limitations identified in Figure 1.1, organized as a four-layer stack that separates concerns across hardware, abstraction, communication, and application. At the base lies the Hardware (HW) layer, which accommodates heterogeneous physical components: a Software-Defined Radio (SDR) node, a general-purpose  $\mu$ P, and a dedicated power management unit, without imposing any fixed binding between hardware and software functions. Above it, the Hardware Abstraction (HA) layer introduces the virtualization substrate that decouples software execution from the specific underlying hardware, enabling containerised deployments to run portably across different physical platforms. The OS and Communication layer sits at the centre of the architecture and provides the two enabling technologies that address (PS 1) and (PS 2): Edge Computing, which supports local data processing and reduces reliance on cloud connectivity, and multi-access communication via both LoRaWAN and 5G Standalone networks, which provides flexible and resilient connectivity for heterogeneous deployment contexts. At the top, the Application (App.) layer hosts the functional modules: TinyML inference, Control logic, and Power Control, that directly address the remaining Problem Statements. Crucially, these modules are shown with dashed borders, signifying that they are software-defined and hardware-independent: they can be deployed, updated, or replaced without modifying the underlying layers. Compared to the monolithic, hardware-coupled architecture of Figure 1.1, this layered design achieves two fundamental objectives. It eliminates the custom glue

logic by replacing it with a standardized abstraction and virtualization layer, and it enables the functional modules to be developed, tested, and deployed as independent software units. This modularity is essential for supporting the rapid evolution of LEV platforms and the diversity of deployment contexts, while also facilitating maintenance, updates, and the integration of new functionalities over time.

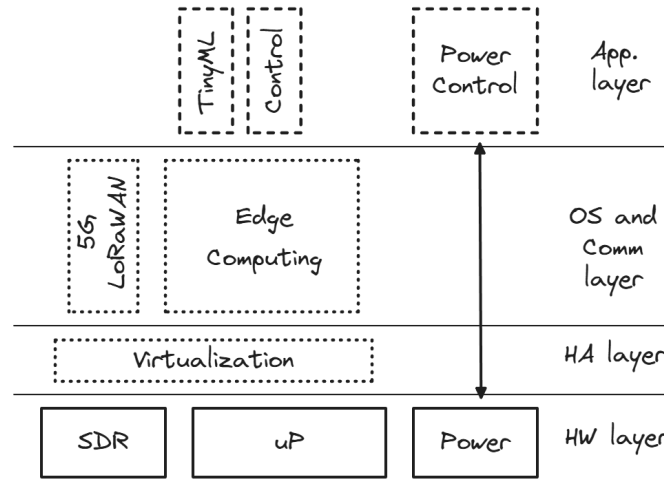


Figure 1.2: Proposed view of the Problem Statement landscape, with the five research objectives addressing the identified gaps

## 1.5 Thesis Structure

The remainder of this thesis is organized as follows.

Chapter 2 establishes the technical foundations of the work, introducing Edge Computing, lightweight virtualization, and a set of industrial case studies on containerized deployments on Industrial Grade Edge Computing platforms to support (PS 1) technical requirements. These case studies provide the performance baseline against which the LEV-oriented contributions are evaluated.

Chapter 3 addresses the communication infrastructure layer, presenting four experimental contributions on LoRaWAN disaggregation, network emulation, Software-Defined Radio nodes, and 5G Standalone private networks in smart mobility scenarios, in support of (PS 1).

Chapter 4 presents the Black Box, the modular containerized edge node developed for on-board LEV deployment, together with two on-device data management contributions addressing (PS 2): an Autoencoder-based compression model and a set of incremental anomaly detection algorithms for TinyML-class devices.

Chapter 5 addresses (PS 3), presenting the biometric data acquisition system deployed during the GiroE-2024 e-bike tour and a graph-based spatial aggregation pipeline for multi-rider physiological data analysis.

Chapter 6 addresses (PS 4) and (PS 5), presenting an autonomous fall detection system for light vehicles and simulation-based methodologies for the design and techno-environmental assessment of PV-powered e-bike charging stations.

Read together, Chapters 3 through 5 form a single end-to-end pipeline: the communication infrastructure of Chapter 3 provides the connectivity envelope, the edge node and on-device intelligence of Chapter 4 acquire and process the data within that envelope, and the health-monitoring application of Chapter 5 consumes the resulting streams at scale. Each chapter is therefore a stage of the same data path, from radio link to rider insight, rather than an isolated contribution.

Chapter 7 summarizes the main findings, discusses limitations, and outlines directions for future research.

## **Chapter 2**

# **Opportunities of Industry 4.0 in Smart Mobility: Background and Industrial Case Studies**

This chapter addresses the foundational question underlying (PS 1) and (PS 2): what paradigms and technologies has the industrial domain developed to tackle modularity, data management, and distributed intelligence, and how mature are they? It introduces Edge Computing architectures and lightweight virtualization as the enabling technologies for hardware-independent, disaggregated IoT deployments, and presents a set of industrial case studies in which these paradigms are validated on Industrial Edge platforms. The results of these case studies serve as the performance baseline for all subsequent LEV-oriented contributions.

## 2.1 Introduction to Edge Computing

The origins of Edge Computing can be traced back to the late 1990s, when Akamai introduced its Content Delivery Network (CDN), placing geographically distributed nodes closer to end users to serve cached content such as images and video. Edge Computing generalizes and extends the CDN concept by leveraging a broader cloud-edge infrastructure[2, 3].

An early demonstration of the potential of this paradigm is provided by Noble et al. in their 1997 work “*Agile application-aware adaptation for mobility*” [4], which showed how resource-constrained mobile devices can support demanding applications including: web browsing, video streaming, and speech recognition by offloading computation to nearby servers. This idea was subsequently generalized in the context of pervasive computing by Satyanarayanan et al. in their 2001 paper “*Pervasive computing: vision and challenges*” [5].

Edge Computing can be broadly defined as the practice of processing, analyzing, and storing data in proximity to where it is generated, so as to enable near real-time responses [6]. In recent years many organisations have pursued a centralised model, consolidating storage and computation in remote cloud data centres. However, the emergence of billions of connected devices and the requirements of new use cases, such as advanced warehouse management, vision-guided robotic production lines, and smart-city traffic control, have exposed the limitations of this approach. The volume of data generated at the network periphery is growing exponentially, driven by devices including smart cameras, mobile point-of-sale terminals, medical sensors, industrial PCs, and IoT gateways. Industry projections estimate that by 2025 as much as 75% of all data will be produced outside centralised data centres, yet today the vast majority of processing still occurs there. Furthermore, a large fraction of the data currently col-

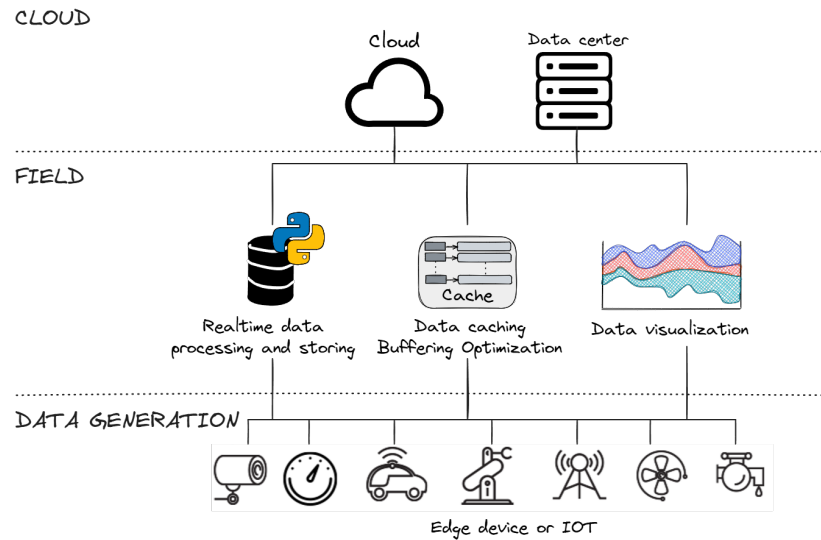


Figure 2.1: A representative Edge Computing architecture, showing the hierarchical relationship between the cloud, edge, and devices.

lected by organisations is never exploited, a problem that Edge Computing can help address by enabling high-performance, low-latency processing on secure platforms closer to the data source.

### 2.1.1 Edge Computing Architecture

A representative Edge Computing architecture is illustrated in Figure 2.1. It is organized into three hierarchical layers [7].

The **CLOUD layer** corresponds to the cloud data centre, which acts as the ultimate repository for aggregated information and hosts global analytics and long-term storage services. Applications running at the edge may operate independently of this layer when connectivity is unavailable.

The **FIELD layer** constitutes the edge itself and may correspond to any facility, plant, or infrastructure deployment point. It comprises three main compo-

nents:

- *Edge data centres*, which can be thought of as local cloud nodes that reduce the data path from devices, thereby alleviating latency and bandwidth constraints;
- *Edge servers and gateways*, which distribute applications to devices and maintain continuous communication with them through installed agents; a single edge server may supervise thousands of devices;
- *Edge and IoT devices*, which are capable of performing local analytics and storing a subset of data to support edge-level operations without relying on upstream connectivity.

While the **DATA GENERATION layer** is not strictly part of the Edge Computing architecture, it is composed of the myriad of sensors, actuators, and devices that generate the data to be processed at the edge.

### 2.1.2 Cloud Computing vs. Edge Computing

Cloud Computing and Edge Computing are both distributed computing paradigms, yet they differ in a fundamental way [2]. Cloud Computing centralises computational resources in large, geographically remote data centres accessible over the Internet under a pay-per-use model. While this approach provides virtually unlimited scalability and simplifies infrastructure management, it introduces constraints that become critical whenever applications require low-latency or continuous operation.

The principal limitation is latency: data must traverse long-distance networks before being processed and before any response is returned, a delay that is unacceptable for applications such as industrial control, telemedicine, or real-time fall

detection on a moving vehicle, all of which demand near-instantaneous feedback. A second constraint is bandwidth: the volume of raw data generated by dense IoT deployments (smart cameras, inertial measurement units, biosensors) can rapidly saturate wide-area network links, leading to bottlenecks and degraded quality of service. Beyond performance, security and privacy represent a further concern, since transmitting sensitive data (biometric measurements, vehicle telemetry, location traces) to third-party data centres increases the attack surface and raises questions of data sovereignty and regulatory compliance. Finally, cloud-centric architectures exhibit an inherent connectivity dependence: in environments where Internet access is intermittent or unavailable (remote cycling routes, rural charging stations, underground facilities), the system becomes unreachable and any safety-critical service is suspended.

These limitations are not merely theoretical. In the context of smart mobility and Technology for Health, where a delayed emergency alert or a missed fall detection event can have life-threatening consequences, the architectural assumption that every decision can be delegated to a remote data centre is fundamentally untenable [3].

### 2.1.3 Advantages of Edge Computing

Edge Computing addresses the limitations described above by relocating computation to nodes that are physically proximate to the data source, eliminating or drastically shortening the network path that separates sensing from decision-making [2, 7].

The most immediate benefit is reduced latency and increased responsiveness. By processing data locally on an edge node embedded in the vehicle or installed at a roadside unit, reaction times that are previously bounded by wide-area network propagation become bounded only by local computation, dropping from

hundreds of milliseconds to single-digit milliseconds. This improvement is decisive for safety-critical services such as autonomous emergency braking, real-time heart-rate anomaly detection, or the actuation of electrovalves in precision-irrigation systems.

A second advantage is the optimisation of network traffic. Rather than forwarding every raw sample to the cloud, the edge node filters, aggregates, and compresses data locally, transmitting only features or anomaly alerts upstream. This reduces bandwidth consumption by orders of magnitude, lowers transmission costs, and alleviates congestion on shared infrastructure, an important consideration in dense urban deployments where many vehicles share the same cellular or LoRaWAN channel.

Edge architectures also offer improved reliability and offline operation. An edge node that has locally stored the relevant application logic can continue to function and record data autonomously even when the uplink to the cloud is degraded or interrupted. For a cyclist travelling through a tunnel or a rural area beyond cellular coverage, this property ensures that safety monitoring is never silently disabled by a transient connectivity loss.

Finally, enhanced security is achieved by reducing both the frequency and the volume of data transmitted over public networks. Sensitive information (biometric signals, vehicle identifiers, location history) can be anonymised or aggregated at the edge before any transmission, limiting the exposure to interception and minimising the data footprint that must comply with privacy regulations such as the GDPR.

### 2.1.4 Representative Use Cases

The advantages described in the previous section have motivated the adoption of Edge Computing across a wide spectrum of application domains. Three areas

are of particular relevance to the research presented in this thesis: industrial automation, smart mobility, and Technology for Health.

In industrial automation, edge platforms have become a key enabler of Industry 4.0 and the emerging Industry 5.0 paradigm. Programmable Logic Controllers (PLCs), vision systems, and condition-monitoring sensors generate high-frequency data streams that must be processed with deterministic latency to guarantee the correct operation of production lines. Edge nodes handle real-time closed-loop control locally while forwarding aggregated telemetry to cloud-based analytics services for predictive maintenance and production optimisation. The case studies presented in Section 2.1.6 demonstrate that lightweight containerization makes it possible to host these functions on commodity hardware without compromising timing requirements [8, 9].

In smart mobility, the edge is the natural home for the processing pipeline of connected Light Electric Vehicles (LEVs). Position tracking, battery-state estimation, over-the-air software updates, and V2X communication all require low-latency data paths that are ill-suited to a cloud-only architecture [10]. A roadside edge node can aggregate telemetry from a fleet of e-bikes, detect anomalies (sudden deceleration consistent with a fall, battery thermal runaway), and issue alerts within the time window required for an effective emergency response, independently of the backhaul quality.

In the domain of Technology for Health, the requirements that edge computing must satisfy are in many respects identical to those of industrial automation: high reliability, bounded latency, and the ability to operate in environments where network connectivity cannot be guaranteed. Wearable biosensors and on-board inertial units generate continuous data streams from which fall events, cardiac arrhythmias, or excessive physical stress must be detected within seconds. Running inference directly on an edge node embedded in the vehicle, rather than

streaming raw signals to the cloud, reduces response time, preserves the rider's privacy, and ensures that the monitoring service remains active even in areas outside cellular coverage. The convergence of these two domains, industrial-grade determinism applied to personal health and safety, is the central design principle of the MOST project and of the contributions described in the following chapters.

### 2.1.5 Virtualization

Virtualization is a software technique that abstracts the underlying physical hardware, presenting each running process or group of processes with the illusion of executing on a dedicated machine. By decoupling software from the specific hardware on which it runs, virtualization enables multiple isolated execution environments to coexist on a single physical host, improving resource utilisation, portability, and manageability. Two broad families of virtualization are relevant to the context of this thesis: full (or hardware-level) virtualization based on virtual machines, and lightweight virtualization based on containers.

#### Full Virtualization and Virtual Machines

In full virtualization, a software layer known as the *hypervisor* sits between the physical hardware and one or more guest operating systems in the case of hypervisor type 1, and on the top of the host OS in the case of hypervisor type 2 (see Figure 2.2 (a) and (b)). Each guest OS, together with its applications, is encapsulated in a *Virtual Machine* (VM) that believes it has exclusive access to a complete hardware platform, including virtual CPU, memory, storage, and network interfaces. The hypervisor intercepts privileged instructions issued by the guest kernels and translates them into operations on the physical hardware, enforcing strict isolation between VMs. This architecture supports *strong security*

*boundaries* and the concurrent execution of heterogeneous operating systems (e.g., Windows and Linux on the same server), but it comes at a significant cost: each VM carries a full OS stack, which implies substantial memory overhead (typically hundreds of megabytes per instance), slow boot times (on the order of tens of seconds to minutes), and a non-negligible CPU penalty due to hypervisor mediation. For resource-constrained edge nodes, these characteristics make full virtualization impractical as the primary deployment mechanism.

### Lightweight Virtualization and Containers

Lightweight virtualization (LV) addresses these limitations by exploiting isolation mechanisms already built into the OS kernel, rather than introducing an additional emulation layer. The key building blocks are *Linux namespaces* and *control groups* (cgroups). Namespaces partition the global view of OS resources (process identifiers, network interfaces, file system mount points, inter-process communication objects, user identifiers and hostname) so that each container perceives its own independent instance of these resources, while remaining unaware of the resources belonging to other containers or to the host. Cgroups complement namespaces by enforcing quantitative resource limits: the amount of CPU time, physical memory, block I/O bandwidth, and network bandwidth that each container is permitted to consume can be bounded independently, preventing a single misbehaving service from starving the others [11].

Because all containers on a host share the same OS kernel (see Figure 2.2 (c) ), no per-instance kernel copy is required and the overhead of hardware emulation is absent. Container images are correspondingly compact (often a few tens of megabytes), start-up times are measured in milliseconds rather than minutes, and the same physical node can host a substantially larger number of concurrent isolated workloads than would be feasible with full VMs. These properties make

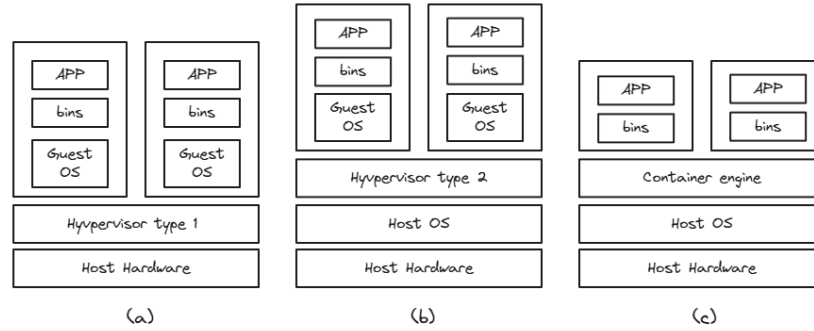


Figure 2.2: Comparison of full virtualization (VMs) (a), (b) and lightweight virtualization (containers) (c).

containers particularly attractive in IoT and edge-computing scenarios, where the host hardware may be a low-power single-board computer or an industrial IPC with limited RAM and no GPU [12].

### Docker and Container Orchestration

Among the container runtimes available today, *Docker* has become the de facto standard for packaging and deploying containerised applications. Docker introduces the concept of a *container image*, an immutable, layered file system snapshot that bundles the application binary, its runtime dependencies, and its configuration into a single portable artifact. Images are stored in registries (public or private) and can be pulled to any Docker-compatible host regardless of the underlying OS distribution, guaranteeing bit-for-bit reproducibility across development, testing, and production environments.

At runtime, Docker instantiates one or more *containers* from an image. Each container runs as an isolated process tree on the host kernel, with its own virtual network interface, file system overlay, and resource budget as defined by cgroups. Communication between containers on the same host is handled through

virtual Docker networks (bridge, host, or overlay mode), while external connectivity is provided through configurable port mappings or host-network sharing. For multi-container applications, *Docker Compose* allows the entire service graph (application logic, middleware, database, message broker) to be described declaratively in a single YAML file and brought up or torn down atomically, a property exploited in the containerised LoRaWAN architectures described in Chapter 3.

For production deployments that span multiple hosts, container *orchestration* platforms such as Kubernetes or Docker Swarm extend these capabilities with automatic scheduling, health checking, rolling updates, and horizontal scaling. In the edge context, lightweight orchestrators (K3s, MicroK8s) have been specifically designed to operate within the memory and CPU envelope of embedded hardware, making cloud-native DevOps practices applicable to roadside units and on-vehicle computers.

### Relevance to Smart Mobility and Technology for Health

The adoption of container-based virtualization in this thesis is motivated by three concrete requirements. First, *modularity*: the software stack of a smart e-bike (sensor acquisition, LoRaWAN communication, fall-detection inference, health-data logging) [13] can be decomposed into independent containers, each updated or replaced without restarting the others, enabling over-the-air software maintenance on deployed vehicles. Second, *hardware independence*: the same container images developed and tested on a desktop workstation can be deployed on an ARM-based edge board embedded in the vehicle frame, without recompilation or configuration changes, reducing the development and validation effort. Third, *resource isolation*: safety-critical containers (e.g., the fall-detection engine) can be assigned dedicated CPU and memory budgets via cgroups, preventing non-critical background tasks (logging, telemetry compression) from introduc-

ing latency jitter in the real-time inference path. The experimental evidence supporting the feasibility of this approach in industrial-grade real-time scenarios is presented in the case studies of Section 2.1.6 [8].

### 2.1.6 Case Studies in Industry 4.0

Industrial automation has undergone significant evolution over the past decades, driven by the increasing demand for efficiency, flexibility, and reliability in manufacturing environments. This section provides an overview of the classical SCADA and PLC-based control architecture, and examines how emerging virtualization technologies are reshaping its implementation [8, 9].

A well-established industrial automation paradigm, depicted in Figure 2.3 (a), integrates both software and hardware elements to oversee and coordinate complex manufacturing processes. At its core, this paradigm relies on two fundamental components: SCADA systems and Programmable Logic Controllers (PLCs).

The SCADA system acts as a centralized supervisory layer, continuously gathering and processing data from geographically distributed points within the industrial plant. Its infrastructure encompasses a supervisory computing unit, a Human-Machine Interface (HMI), and the necessary communication backbone. PLCs, by contrast, are dedicated embedded controllers responsible for executing deterministic, real-time control tasks. They interface directly with field-level equipment, including sensors and actuators, and relay relevant operational data back to the SCADA layer, which in turn provides operators with a unified view of the entire production workflow.

Data exchange between SCADA and PLCs relies on Machine-to-Machine (M2M) or Controller-to-Controller (C2C) communication protocols [14]. These protocols handle the bidirectional flow of information: configuration and command data are pushed toward the PLCs, while process-related telemetry is for-

warded to the SCADA system. While communication between PLCs and field devices may use a variety of real-time communication protocols (RT protocols).

As highlighted in [15], virtualization and containerization technologies are accelerating the digital transformation of the manufacturing sector. The increasing maturity of these technologies has unlocked novel opportunities for industrial deployments. Traditional physical devices typically demand specialized, expensive hardware platforms, which constrains their adaptability and scalability. Virtual counterparts, on the other hand, can run on commodity general-purpose hardware, significantly reducing costs while offering improved scalability and a reduced environmental footprint [16].

Figure 2.3 (b) illustrates an example of such a virtualization-based architecture. The logical communication structure remains the same as in Figure 2.3 (a); however, it does change how individual components are implemented. PLCs, sensors, and SCADA systems can now be instantiated within virtual environments. M2M and sensor communication protocols are preserved and can be deployed directly within the virtual SCADA or virtual PLC instances, or alternatively as standalone microservices [17].

Nevertheless, the transition to virtual devices requires careful assessment of their applicability to specific industrial scenarios, particularly those with stringent real-time requirements. Whereas physical devices operate on bare-metal hardware, meaning their performance is directly determined by the underlying hardware characteristics, the behavior of virtual devices is influenced by multiple factors, including:

1. the chosen virtualization technique;
2. the operating system in use;
3. the hypervisor layer.

The primary virtualization approaches can be summarized as follows:

- **Full Virtualization:** This approach provides strong isolation between environments, enabling multiple distinct operating systems to run concurrently on a single physical host.
- **Containerization (Lightweight Virtualization):** Containers represent a more resource-efficient form of virtualization, in which the host kernel is shared across multiple isolated container instances [18].

Each of these approaches entails different trade-offs in terms of isolation guarantees, resource management, deployment complexity, and operational flexibility. Consequently, the selection and configuration of the virtualization strategy must be approached with particular care when deploying virtual industrial devices.

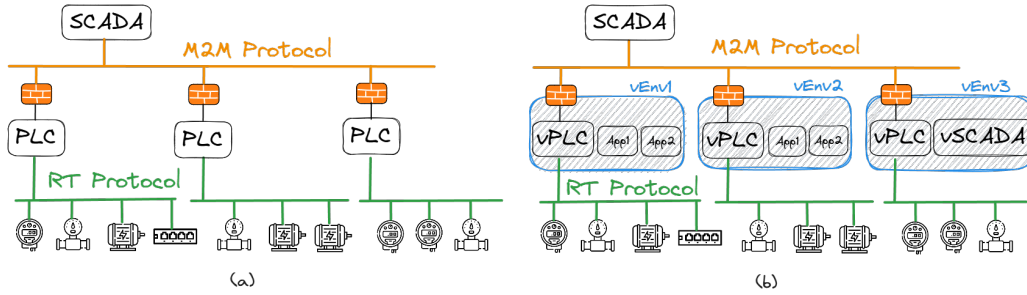


Figure 2.3: Comparison of old OT paradigm (a) and new OT paradigm using edge computing and containerization technology (b).

In the following studies we present three different case studies focused on benchmarking virtual PLCs, M2M communication, and real-time communication protocols in industrial automation, based on the work respectively presented in [19], [20] and [21].

Figure 2.4 illustrates the architectures of the three case studies, which share the same physical infrastructure but differ in the PLC implementation: (a) virtual

PLCs running on a Virtual Machine (VM), (b) virtual PLCs running on an IPC, and (c) real PLCs. The first case study evaluates PLC virtualization performance, while the second and third assess M2M and real-time communication protocols, respectively.

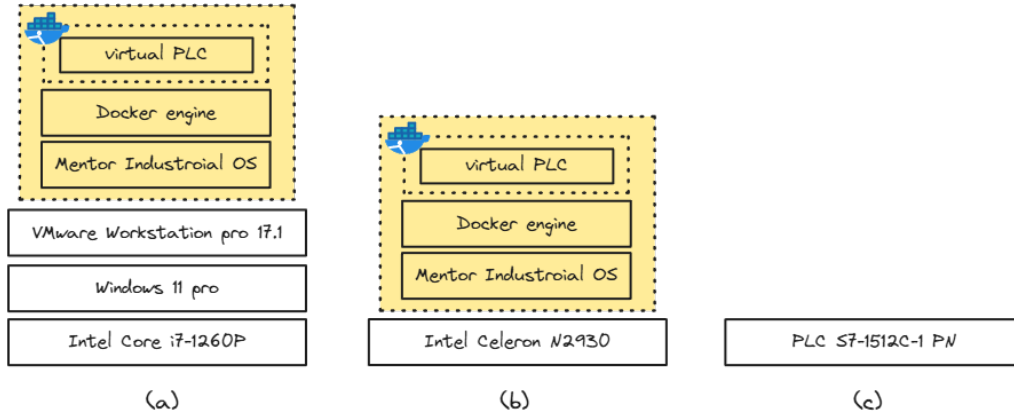


Figure 2.4: Architectures of the three case studies: (a)  $V_{IPC}$ s on Virtual Machine, (b)  $V_{IPC}$ s on IPC, and (c) real PLCs.

Containerized implementations leverage the same Industrial Edge Platform by Siemens, described in detail in Appendix A.1. More specifically, the three case studies are based on the following devices:

- **CPU1517v on desktop PC ( $V_{PC}$ ) (Figure 2.4 (a))**: A desktop machine with a 4.70GHz Intel Core i7-1260P CPU, 16GB of RAM and Windows 11 pro as the host operating system. The CPU1517v is deployed via VMware Workstation Pro 17.1, with 4GB memory, 2 virtual processors, and two bridged network adapters assigned to the VM, running IED OS.
- **CPU1517v on Simatic IPC227E ( $V_{IPC}$ ) (Figure 2.4 (b))**: A compact industrial PC equipped with a quad-core Intel Celeron N2930 at 1.83 GHz,

8 GB RAM, and a 240 GB SSD. It runs the IED OS, based on Debian Linux real-time with containerization middleware.

- **S7-1512C-1 PN (R) (Figure 2.4 (c)):** A Siemens SIMATIC S7-1500 series PLC featuring 250 KB program memory, 1 MB data memory, and native support for real-time protocols (Profibus, Profinet, Ethernet/IP).

### Case Study: Benchmarking virtual PLCs

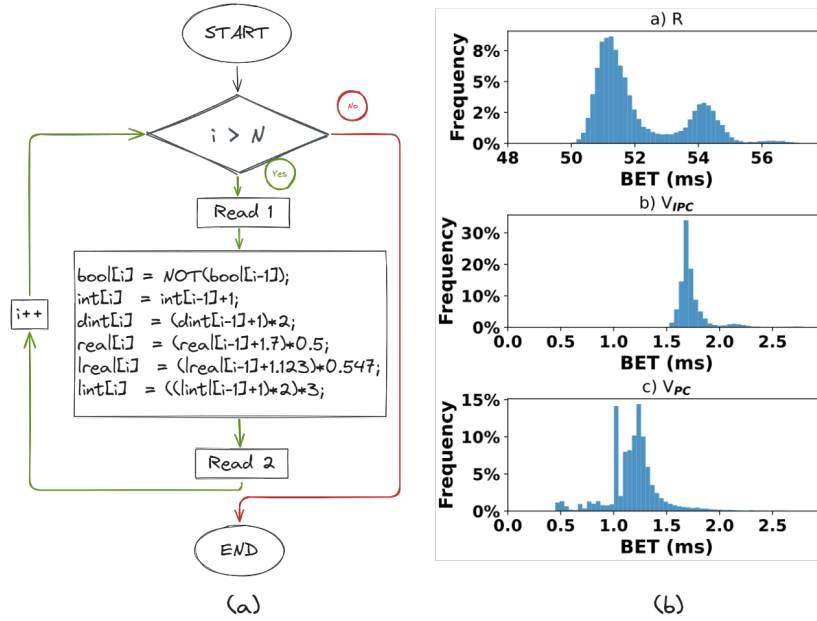


Figure 2.5: (a) Benchmark program and (b) benchmark results for different PLC implementations.

Industrial automation is undergoing a significant transition from fixed hardware controllers to Virtual PLCs (vPLCs), which are containerized versions of traditional PLC firmware. This case study aims to characterize the computational performance of vPLCs executed on different edge platforms compared to stan-

dard hardware units. The evaluation focuses on the Benchmark Execution Time (BET), assessing the efficiency of the virtualized controller in executing complex logic tasks while managing the overhead introduced by the virtualization layer.

This study [19] assesses the computational performance of different PLC implementations across multiple data types. BET is measured using the RUNTIME executing a standard benchmark program (Figure 2.5 (a)), which performs a batch of arithmetic operations instruction, specific to S7 PLCs, which records two timestamps, *Read 1* and *Read 2* in the flow diagram, and computes their difference to determine the execution time of the arithmetic block.

The benchmark is executed with  $N = 30000$  iterations. The BET distributions are reported in Figure 2.5 (b), while the corresponding statistics are summarized in Table 2.1. *R* exhibits an average BET of 52 ms, significantly higher than the 1.7 ms recorded for  $V_{IPC}$  and the 1.2 ms for  $V_{PC}$ , a difference attributable to the superior processing capabilities of the hardware platforms hosting the virtual PLCs.

Comparing the two virtual implementations,  $V_{PC}$  introduces greater timing variability due to the additional hypervisor layer: its BET ranges from 0.4 ms to 21.7 ms, whereas  $V_{IPC}$  remains more consistent, between 1.5 ms and 6.0 ms. The bimodal shape of the distributions in Figure 2.5 (b) is explained by the presence of auxiliary OS-level operations that periodically interrupt benchmark execution, causing occasional spikes in the recorded BET. Although this effect is more pronounced in absolute terms for slower devices such as *R*, the relative overhead remains limited, accounting for approximately 6% of the average execution time.

Table 2.1: Experimental results for BET test.

| <b>Metric</b>                    | $R$   | $V_{IPC}$ | $V_{PC}$ |
|----------------------------------|-------|-----------|----------|
| $N$                              | 30000 | 30000     | 30000    |
| Average (ms)                     | 52    | 1.7       | 1.2      |
| Min (ms)                         | 50    | 1.5       | 0.4      |
| Max (ms)                         | 70    | 6.0       | 21.7     |
| std (ms)                         | 1.5   | 0.19      | 0.39     |
| 90 <sup>th</sup> percentile (ms) | 54    | 1.9       | 1.4      |

### Case Study: S7Comm Protocol in Industrial Automation

While internal processing speed is vital, the ability of controllers to interact efficiently is equally critical. This section investigates the performance of the S7Comm protocol (Appendix A.2), a standard for M2M data exchange between Siemens-based automation devices and SCADA systems. The study [20] focuses on evaluating the communication stack delays and the round-trip times (RTT) for 'Request and Response' transactions when one or both partners are virtualized (Combination reported in Table 2.2), establishing a baseline for supervision and coordination tasks in distributed IoT environments.

The metrics of the proposed methodology are defined, without lack of generality, in the case of transaction of type "Request and Response". As a matter of fact, the other type of transaction is the "Publish" where one partner emits a message without being asked for. In practical systems "Publish with Acknowledge" and "Publish without Acknowledge" styles are possible, and the approach proposed here can deal also with them, as described after the metrics definitions.

iii) at time  $T_3$  the Request is read from PLC edge; iv) at time  $T_4$  the Response is visible on the network; v) at time  $T_5$  the Response is read by PLC client. The

Table 2.2: Experiment matrix for the comparison.

| Ref. # | PLC Client | PLC Edge  |
|--------|------------|-----------|
| 1      | $R$        | $R$       |
| 2      | $R$        | $V_{IPC}$ |
| 3      | $R$        | $V_{PC}$  |
| 4      | $V_{IPC}$  | $R$       |
| 5      | $V_{IPC}$  | $V_{IPC}$ |
| 6      | $V_{IPC}$  | $V_{PC}$  |
| 7      | $V_{PC}$   | $R$       |
| 8      | $V_{PC}$   | $V_{IPC}$ |
| 9      | $V_{PC}$   | $V_{PC}$  |

interaction between PLC client and PLC edge is shown in Figure 2.6: i) at time  $T_1$  is generated the Request; ii) at time  $T_2$  the Request is seen on the network via a network sniffer; The following latencies are defined and evaluated:

- $L_{15} = T_5 - T_1$ , the end-to-end latency of the transaction;
- $L_{24} = T_4 - T_2$ , the latency of the transaction as seen on the network;
- $L_{12} = T_2 - T_1$ , the latency from the generation of the Request to the visibility of the Request on the network;
- $L_{23} = T_3 - T_2$ , the latency from the visibility of the Request on the network to the reading of the Request by PLC edge;
- $L_{34} = T_4 - T_3$ , the latency from the reading of the Request by PLC edge to the visibility of the Response on the network;

- $L_{45} = T_5 - T_4$ , the latency from the visibility of the Response on the network to the reading of the Response by PLC client;
- $L_{13} = T_3 - T_1$ , the latency from the generation of the Request to the reading of the Request by PLC edge;
- $L_{35} = T_5 - T_3$ , the latency from the reading of the Request by PLC edge to the reading of the Response by PLC client.

The proposed methodology aims to provide meaningful performance insights for the system under test. The experimental results are summarized in Table 2.4 and Table 2.5.

As a reference, experiment #1 (real PLCs only) yields a round-trip time  $L_{15}$  with an average value of 3.3 ms. The validity of the measurement setup is confirmed by observing that the average value of  $L_{15}$  consistently equals the sum of the average values of its components  $L_{12}$ ,  $L_{23}$ ,  $L_{34}$ , and  $L_{45}$  (within the expanded uncertainty), as reported in Table 2.4.

Regarding the individual latency contributions, when PLC client is a virtual PLC (experiments #4–9),  $L_{45}$  ranges on average from 37 to 102 ms, with values depending on the synchronization state of PLC client and PLC edge at the start of the experiment. When PLC edge is a virtual PLC (experiments #2, 3, 5, 6, 8, 9),  $L_{23}$  averages between 47 and 54 ms and follows a uniform distribution.

In conclusion, real PLCs outperform virtual PLCs in terms of average S7 transaction latency. However, when minimum values are compared, the two implementations exhibit equivalent performance, indicating that the additional delay in virtual PLCs stems solely from the IP stack implementation rather than from the virtualization layer itself. No significant differences are observed between the two virtual environments tested. Therefore, the primary recommendation for improving performance in this use case is to optimize the IP stack of

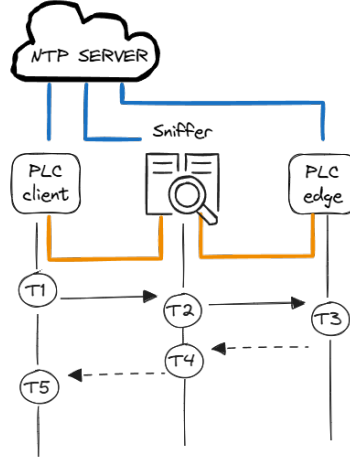


Figure 2.6: A representative architecture of a machine-to-machine communication system, showing the flow of data between devices and timestamping points for latency measurement.

the virtual PLC.

Table 2.3: Standard Uncertainty and Expanded Uncertainty for Every Metric and Experiment.

| # | $L_{15}$ (ms) |     | $L_{24}$ (ms) |     | $L_{12}$ (ms) |     | $L_{23}$ (ms) |     | $L_{34}$ (ms) |     | $L_{45}$ (ms) |     | $L_{13}$ (ms) |     | $L_{35}$ (ms) |     |
|---|---------------|-----|---------------|-----|---------------|-----|---------------|-----|---------------|-----|---------------|-----|---------------|-----|---------------|-----|
|   | $u_c$         | $U$ | $u_c$         | $U$ | $u_c$         | $U$ | $u_c$         | $U$ | $u_c$         | $U$ | $u_c$         | $U$ | $u_c$         | $U$ | $u_c$         | $U$ |
| 1 | 0.03          | 0.1 | 0.79          | 1.6 | 0.64          | 1.3 | 0.64          | 1.3 | 0.64          | 1.3 | 0.64          | 1.3 | 0.03          | 0.1 | 0.03          | 0.1 |
| 2 | 0.03          | 0.1 | 0.79          | 1.6 | 0.64          | 1.3 | 0.87          | 1.7 | 0.87          | 1.7 | 0.64          | 1.3 | 0.59          | 1.2 | 0.59          | 1.2 |
| 3 | 0.03          | 0.1 | 0.79          | 1.6 | 0.64          | 1.3 | 0.66          | 1.3 | 0.66          | 1.3 | 0.64          | 1.3 | 0.14          | 0.3 | 0.14          | 0.3 |
| 4 | 0.74          | 1.5 | 0.79          | 1.6 | 0.87          | 1.7 | 0.64          | 1.3 | 0.64          | 1.3 | 0.87          | 1.7 | 0.59          | 1.2 | 0.59          | 1.2 |
| 5 | 0.74          | 1.5 | 0.79          | 1.6 | 0.87          | 1.7 | 0.87          | 1.7 | 0.87          | 1.7 | 0.87          | 1.7 | 0.74          | 1.5 | 0.74          | 1.5 |
| 6 | 0.74          | 1.5 | 0.79          | 1.6 | 0.87          | 1.7 | 0.66          | 1.3 | 0.66          | 1.3 | 0.87          | 1.7 | 0.60          | 1.2 | 0.60          | 1.2 |
| 7 | 0.20          | 0.4 | 0.79          | 1.6 | 0.66          | 1.3 | 0.64          | 1.3 | 0.64          | 1.3 | 0.66          | 1.3 | 0.14          | 0.3 | 0.14          | 0.3 |
| 8 | 0.20          | 0.4 | 0.79          | 1.6 | 0.66          | 1.3 | 0.87          | 1.7 | 0.87          | 1.7 | 0.66          | 1.3 | 0.60          | 1.2 | 0.60          | 1.2 |
| 9 | 0.20          | 0.4 | 0.79          | 1.6 | 0.66          | 1.3 | 0.66          | 1.3 | 0.66          | 1.3 | 0.66          | 1.3 | 0.20          | 0.4 | 0.20          | 0.4 |

Table 2.4: Round-Trip Latencies, Forward and Backward Latencies (‘—’ means under resolution.)

| # | $L_{15}$ (ms) |       |     |      | $L_{24}$ (ms) |     |     |     | $L_{13}$ (ms) |     |     |     | $L_{35}$ (ms) |     |     |      |
|---|---------------|-------|-----|------|---------------|-----|-----|-----|---------------|-----|-----|-----|---------------|-----|-----|------|
|   | Avg           | max   | min | Std  | Avg           | max | min | Std | Avg           | max | min | Std | Avg           | max | min | Std  |
| 1 | 3.3           | 4.6   | 2.6 | 0.3  | —             | —   | —   | —   | 0.9           | 2.2 | 0.2 | 0.4 | 2.41          | 3.4 | 1.3 | 0.42 |
| 2 | 54.6          | 102.8 | 3.3 | 28.8 | 52            | 100 | —   | 29  | 51            | 100 | —   | 29  | 2             | 4   | —   | —    |
| 3 | 50.8          | 103.6 | 3.4 | 28.7 | 48            | 101 | —   | 29  | 48            | 101 | 1   | 29  | 1             | 3   | —   | 0.5  |
| 4 | 45            | 102   | 3   | 32   | —             | 2   | —   | —   | —             | 3   | —   | —   | 38            | 97  | —   | 30   |
| 5 | 118           | 175   | 15  | 37   | 52            | 101 | 1   | 30  | 51            | 100 | —   | 30  | 61            | 71  | —   | 22   |
| 6 | 134           | 188   | 82  | 29   | 52            | 106 | 1   | 29  | 52            | 107 | —   | 29  | 82            | 83  | 81  | 29   |
| 7 | 57            | 114   | 4   | 29   | —             | 2   | —   | —   | —             | 1   | 2   | —   | 60            | 116 | 7   | 29   |
| 8 | 151           | 201   | 95  | 30   | 53            | 101 | —   | 29  | 54            | 104 | —   | 30  | 96            | 98  | 95  | —    |
| 9 | 137           | 191   | 86  | 29   | 48            | 100 | —   | 29  | 50            | 106 | —   | 29  | 95            | 98  | 93  | —    |

Table 2.5: Transmission Latencies Between Devices (‘—’ means under resolution).

| # | $L_{12}$ (ms) |     |     |     | $L_{23}$ (ms) |     |     |     | $L_{34}$ (ms) |     |     |     | $L_{45}$ (ms) |     |     |     |
|---|---------------|-----|-----|-----|---------------|-----|-----|-----|---------------|-----|-----|-----|---------------|-----|-----|-----|
|   | Avg           | max | min | Std | Avg           | max | min | Std | Avg           | max | min | Std | Avg           | max | min | Std |
| 1 | —             | 2   | —   | —   | —             | —   | —   | —   | —             | 2   | —   | —   | —             | —   | —   | —   |
| 2 | 2             | 2   | 1   | —   | 50            | 98  | —   | 29  | 3             | 4   | —   | —   | —             | 2   | —   | —   |
| 3 | 2             | 2   | —   | —   | 46            | 99  | —   | 29  | —             | 2   | —   | —   | —             | 2   | —   | —   |
| 4 | —             | 3   | —   | —   | —             | 2   | —   | —   | —             | 3   | —   | —   | 37            | 97  | —   | 29  |
| 5 | 2             | 3   | —   | —   | 51            | 100 | —   | 30  | —             | 2   | —   | —   | 65            | 74  | 4   | 22  |
| 6 | —             | —   | —   | —   | 54            | 109 | 2   | 29  | —             | 2   | —   | —   | 79            | 79  | 17  | 77  |
| 7 | —             | —   | —   | —   | —             | 2   | —   | —   | —             | 2   | 3   | —   | 59            | 116 | 6   | 29  |
| 8 | —             | —   | —   | —   | 47            | 99  | —   | 30  | —             | 2   | —   | —   | 102           | 103 | 101 | —   |
| 9 | —             | 2   | —   | —   | 52            | 100 | 6   | 23  | 3             | 4   | —   | —   | 91            | 94  | 89  | —   |

### Case Study: PROFINET Protocol in Industrial Automation

The most critical layer of industrial automation is real-time communication at the field level, where controllers interact directly with sensors and actuators. The goal is to verify if Industrial Edge platforms can support the stringent timing con-

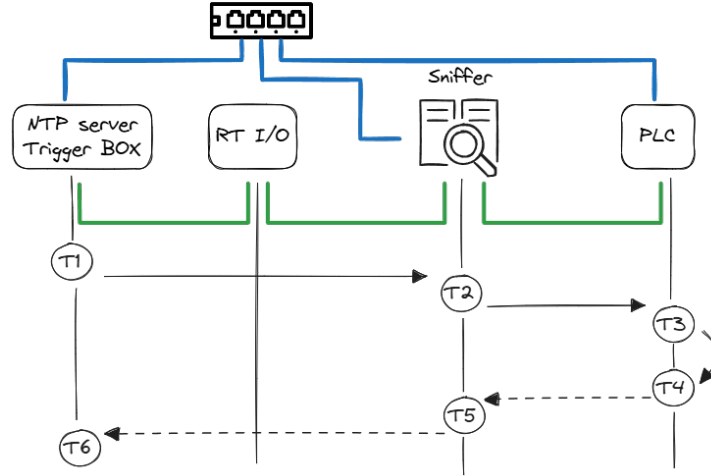


Figure 2.7: Proposed architecture for benchmarking real-time communication protocols, showing the flow of data between devices and timestamping points for latency measurement.

straints in virtualized environments. This study [21] extends the investigation to the PROFINET protocol, evaluating its performance and deterministic behavior required for rapid control loops, ensuring that safety-critical alerts, such as those needed for light vehicle monitoring, are handled with the necessary temporal precision

Building on the previous case study focused on M2M communication where supervision and control data exchange between PLCs (virtual and real), this case study extends the investigation to real-time communication, which represents the most critical layer of industrial automation. Real-time communication governs the direct interaction between the PLC and field devices such as sensors and actuators, where stringent timing constraints must be met to ensure correct control behavior. For this purpose, the PROFINET protocol (Appendix A.3) is selected as a representative real-time communication protocol widely adopted in

industrial settings.

A key difference with respect to the previous case study concerns the synchronization approach: here, a local NTP server is employed to improve timestamping accuracy across all devices, significantly reducing synchronization errors, as shown in Table 2.6 compared to the previous approach (Table 2.3). This choice directly impacts the quality of the latency measurements, as timestamping errors are a primary source of uncertainty in distributed measurement campaigns.

Table 2.6: Standard Uncertainty and Expanded Uncertainty for Every Metric and Experiment for real-time communication.

| #         | $L_{16}$ (ms) |       | $L_{12}$ (ms) |       | $L_{23}$ (ms) |     | $L_{34}$ (ms) |     | $L_{45}$ (ms) |     | $L_{56}$ (ms) |       | $L_{13}$ (ms) |     | $L_{25}$ (ms) |     | $L_{46}$ (ms) |       |
|-----------|---------------|-------|---------------|-------|---------------|-----|---------------|-----|---------------|-----|---------------|-------|---------------|-----|---------------|-----|---------------|-------|
|           | $u_c$         | $U$   | $u_c$         | $U$   | $u_c$         | $U$ | $u_c$         | $U$ | $u_c$         | $U$ | $u_c$         | $U$   | $u_c$         | $U$ | $u_c$         | $U$ | $u_c$         | $U$   |
| $V_{IPC}$ | 0.0008        | 0.002 | 0.0015        | 0.003 | 0.18          | 0.4 | 0.26          | 0.5 | 0.18          | 0.4 | 0.0015        | 0.003 | 0.18          | 0.4 | 0.18          | 0.4 | 0.0019        | 0.004 |
| $R$       | 0.0008        | 0.002 | 0.0015        | 0.003 | 0.24          | 0.5 | 0.33          | 0.7 | 0.24          | 0.5 | 0.0015        | 0.003 | 0.24          | 0.5 | 0.18          | 0.4 | 0.0019        | 0.004 |

The central question addressed is whether a virtual PLC can meet the real-time requirements imposed by industrial protocols such as PROFINET, and how the additional layers of virtualization affect latency, jitter, and cycle stability when compared to a real PLC.

The measurement setup, shown in Figure 2.7, involves three main components: a Trigger Box (equipped by a local NTP server), a Network Traffic Sniffer, and the PLC under test (either  $V_{IPC}$  or  $R$ ). The Trigger Box generates a stimulus signal and captures the system response, while Sniffer monitors all network traffic with synchronized timestamps.

Referring to Figure 2.7, the interaction between the Trigger Box and the PLC is described as follows. At time  $T_1$ , the trigger signal is generated via the output port designated as `ppsrel`, connected to the input port `event2` to produce a

packet carrying temporal information  $T_1$ . At  $T_2$ , the trigger signal is identified on the network via Sniffer. At time  $T_3$ , the application level of the PLC reads the trigger signal. Subsequently, at  $T_4$ , the response is generated by processing the trigger signal and transmitted back to the network. At  $T_5$ , the response is detected on the network via Sniffer. Finally, at  $T_6$ , the Trigger Box reads the response via the input port designated as event 1.

The latencies are defined as follows:

- $L_{16} = T_6 - T_1$ : round-trip time from trigger signal generation to system response;
- $L_{25} = T_5 - T_2$ : elaboration latency, encompassing input reading, processing, and response generation;
- $L_{34} = T_4 - T_3$ : PLC program execution time;
- $L_{12} = T_2 - T_1$ : trigger traverse latency from Trigger Box to Sniffer;
- $L_{23} = T_3 - T_2$ : trigger traverse latency from Sniffer to PLC ( $R$  or  $V_{IPC}$ );
- $L_{45} = T_5 - T_4$ : response traverse latency from PLC to Sniffer;
- $L_{56} = T_6 - T_5$ : response traverse latency from Sniffer to Trigger Box;
- $L_{13} = T_3 - T_1$ : total trigger traverse latency from Trigger Box to PLC;
- $L_{46} = T_6 - T_4$ : total response traverse latency from PLC to Trigger Box.

The stability of the PROFINET cycle is assessed by computing the deviation from the nominal cycle time for each device under test ( $R$ ,  $V_{IPC}$ , and  $RT I/O$ ). The deviation from the nominal PROFINET cycle is evaluated as:

$$\Delta_{PN} = T_{PN,nom} - T_{PN,meas} \quad (2.1)$$

where  $T_{PN,nom}$  is the nominal PROFINET cycle time and  $T_{PN,meas}$  is the measured PROFINET cycle time of the devices  $R$ ,  $V_{IPC}$ , and  $RT I/O$ .

Observing minimum and maximum values reported in Table 2.7, jitter is clearly more pronounced in the  $V_{IPC}$ , with a peak deviation of  $\pm 8.035$  ms, compared to  $\pm 1.075$  ms for the  $R$  and  $\pm 0.512$  ms for the  $RT I/O$ . This difference is significantly attenuated when considering the 5th and 95th percentiles, which filter out sporadic misbehaviors attributable to network fluctuations or virtualization overhead; in this range, both  $R$  and  $V_{IPC}$  exhibit a much more symmetric and comparable behavior, suggesting that extreme deviations are isolated events rather than systematic issues.

The relative clock deviation from the nominal value  $\gamma_{PN,nom}$  is estimated as:

$$\gamma_{PN,nom} = \frac{\Delta_{PN}}{T_{PN,nom}} \quad (2.2)$$

The analysis of  $\gamma_{PN,nom}$  provides further insight into the internal clock quality of each device. The  $RT I/O$  presents a stable deviation of 75 ppm, and the  $R$  an equally stable 3 ppm, both consistent with the behavior expected from hardware-based PROFINET interfaces with dedicated timing hardware. The  $V_{IPC}$ , on the other hand, shows a varying deviation ranging from 0 to 18 ppm, indicative of a software clock implementation tied to the host operating system rather than a dedicated hardware timer. This distinction has direct implications for the achievable timing precision and cycle regularity of virtualized control systems.

The latency analysis highlights the impact of the virtualization layer on different segments of the communication path, as reported in Table 2.8 and Table 2.9.

Latencies  $L_{12}$  and  $L_{56}$  which describe the network path between the Trigger Box and the  $RT I/O$  remain consistent regardless of whether a  $V_{IPC}$  or  $R$  is

under test, as expected, since these segments do not involve the PLC directly. For a PROFINET cycle time of 8 ms, the mean value of  $L_{12}$  is 8.556 ms ( $V_{IPC}$ ) and 8.602 ms ( $R$ ), while  $L_{56}$  averages 0.607 ms and 0.525 ms respectively. Notably,  $L_{56}$  is independent of the PROFINET cycle time and thus remains stable across different experimental configurations.

Latency  $L_{34}$ , representing the PLC program execution time, is extremely low for the  $V_{IPC}$  often below the measurement resolution owing to the high-performance hardware hosting the container. This suggests that, from a purely computational standpoint, the virtualization layer does not introduce a meaningful bottleneck for control program execution.

The round-trip latency  $L_{16}$  is the most critical metric, as it encompasses two full PROFINET cycles: one for the *RT I/O* and one for the PLC. The maximum value of  $L_{16}$  occurs when data must wait for two complete PROFINET cycles before being processed, while the minimum corresponds to data being immediately available at the end of a cycle. At 8 ms cycle time, the  $V_{IPC}$  shows a minimum of 7.711 ms, a maximum of 23.616 ms, and a mean of 15.257 ms; the  $R$  behaves comparably, with values of 6.744 ms (minimum), 23.375 ms (maximum), and 14.688 ms (mean). The close alignment between  $R$  and  $V_{IPC}$  in both mean and worst-case  $L_{16}$  confirms that the virtualization overhead does not substantially degrade end-to-end real-time communication under nominal conditions.

Regarding operational limits, the vendor documentation for the  $V_{IPC}$  version under test specifies that cycle times below 16 ms may cause issues with real-time communication. However, the experimental results present a considerably more favorable picture: communication losses are observed only at a PROFINET cycle time of 1 ms, indicating that the  $V_{IPC}$  can reliably operate at cycle times down to 2 ms. This is a significantly better outcome than vendor expectations and demonstrates that the proposed measurement methodology can provide more

precise and informative characterizations of  $V_{IPC}$  behavior than conservative manufacturer specifications alone.

Table 2.7: Deviation from the nominal PROFINET cycle,  $\Delta_{PN}$  (ns).

| $T_{PN,nom}$      | 32 ms             |                   |                   | 16 ms             |                   |                   | 8 ms              |                   |                   |
|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
|                   | $RTI/O$           | $V_{IPC}$         | $R$               | $RTI/O$           | $V_{IPC}$         | $R$               | $RTI/O$           | $V_{IPC}$         | $R$               |
| mean              | $-2.4 \cdot 10^3$ | $-8.0 \cdot 10^5$ | $1.1 \cdot 10^6$  | $-1.2 \cdot 10^3$ | -40               | 48                | $-6.1 \cdot 10^2$ | 46                | 23                |
| min               | $-1.5 \cdot 10^4$ | $-8.0 \cdot 10^5$ | $-1.1 \cdot 10^6$ | $-5.4 \cdot 10^4$ | $-4.8 \cdot 10^5$ | $-1.9 \cdot 10^5$ | $-5.4 \cdot 10^4$ | $-8.9 \cdot 10^5$ | $-2.0 \cdot 10^5$ |
| max               | $9.9 \cdot 10^3$  | $7.0 \cdot 10^6$  | $1.1 \cdot 10^6$  | $5.2 \cdot 10^4$  | $2.0 \cdot 10^6$  | $2.7 \cdot 10^5$  | $5.2 \cdot 10^4$  | $4.2 \cdot 10^6$  | $2.1 \cdot 10^5$  |
| Perc 5th          | $-1.5 \cdot 10^4$ | $-2.4 \cdot 10^4$ | $-5.7 \cdot 10^3$ | $-1.2 \cdot 10^3$ | $-1.9 \cdot 10^4$ | $-5.3 \cdot 10^3$ | $-6.3 \cdot 10^2$ | $-1.7 \cdot 10^4$ | $-3.8 \cdot 10^3$ |
| Perc 95th         | $-1.5 \cdot 10^4$ | $1.9 \cdot 10^4$  | $6.1 \cdot 10^3$  | $-1.2 \cdot 10^3$ | $2.4 \cdot 10^4$  | $5.5 \cdot 10^3$  | $-5.7 \cdot 10^2$ | $2.2 \cdot 10^4$  | $4.0 \cdot 10^3$  |
| $\gamma_{PN,nom}$ | -75               | 15                | 3                 | -75               | -2                | 3                 | -76               | 6                 | 3                 |

| $T_{PN,nom}$      | 4 ms              |                   |                   | 2 ms              |                   |                   | 1 ms              |                   |                   |
|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
|                   | $RTI/O$           | $V_{IPC}$         | $R$               | $RTI/O$           | $V_{IPC}$         | $R$               | $RTI/O$           | $V_{IPC}$         | $R$               |
| mean              | $-3.0 \cdot 10^2$ | 73                | 11                | $-1.5 \cdot 10^2$ | 1                 | 6                 | -76               | 10                | 3                 |
| min               | $-1.3 \cdot 10^5$ | $-1.0 \cdot 10^6$ | $-3.1 \cdot 10^5$ | $-1.8 \cdot 10^5$ | $-1.0 \cdot 10^6$ | $-3.4 \cdot 10^5$ | $-5.1 \cdot 10^5$ | $-9.6 \cdot 10^5$ | $-9.3 \cdot 10^5$ |
| max               | $1.3 \cdot 10^5$  | $8.0 \cdot 10^6$  | $3.3 \cdot 10^5$  | $1.8 \cdot 10^5$  | $3.0 \cdot 10^6$  | $3.4 \cdot 10^5$  | $5.1 \cdot 10^5$  | $2.5 \cdot 10^6$  | $1.0 \cdot 10^6$  |
| Perc 5th          | $-3.3 \cdot 10^2$ | $-2.1 \cdot 10^4$ | $-3.3 \cdot 10^3$ | $-3.8 \cdot 10^2$ | $-1.4 \cdot 10^4$ | $-3.4 \cdot 10^3$ | $-4.1 \cdot 10^2$ | $-1.3 \cdot 10^4$ | $-3.1 \cdot 10^3$ |
| Perc 95th         | $-2.6 \cdot 10^2$ | $2.1 \cdot 10^4$  | $3.4 \cdot 10^3$  | 70                | $1.3 \cdot 10^4$  | $3.4 \cdot 10^3$  | 48                | $1.3 \cdot 10^4$  | $3.1 \cdot 10^3$  |
| $\gamma_{PN,nom}$ | -75               | 18                | 3                 | -75               | 0                 | 3                 | -76               | 10                | 3                 |

Table 2.8: Use case transmission latencies  $V_{IPC}$  (“—” means under resolution).

| $T_{PN,nom}$ |      | $L_{16}$ (ms) | $L_{12}$ (ms) | $L_{23}$ (ms) | $L_{34}$ (ms) | $L_{45}$ (ms) | $L_{56}$ (ms) | $L_{13}$ (ms) | $L_{46}$ (ms) | $L_{25}$ (ms) |
|--------------|------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| 32 ms        | mean | 37.910        | 20.564        | 1.2           | —             | 15.5          | 0.609         | 21.8          | 16.1          | 16.736        |
|              | min  | 7.979         | 4.533         | —             | —             | 0.7           | —             | 4.9           | 0.8           | 2.042         |
|              | max  | 70.676        | 36.579        | 2.5           | —             | 32.9          | 1.118         | 38.7          | 34.0          | 35.018        |
| 16 ms        | mean | 23.577        | 12.545        | 1.2           | —             | 9.2           | 0.609         | 13.8          | 9.8           | 10.424        |
|              | min  | 7.777         | 4.527         | —             | —             | 0.8           | —             | 5.1           | 1.1           | 2.043         |
|              | max  | 39.315        | 20.581        | 2.3           | —             | 16.9          | 1.114         | 22.4          | 17.9          | 18.864        |
| 8 ms         | mean | 15.257        | 8.556         | 1.3           | —             | 4.8           | 0.607         | 9.8           | 5.4           | 6.094         |
|              | min  | 7.711         | 4.523         | —             | —             | 0.4           | —             | 4.8           | 0.6           | 2.056         |
|              | max  | 23.616        | 12.589        | 2.5           | —             | 9.2           | 1.115         | 14.8          | 10.3          | 10.991        |
| 4 ms         | mean | 11.337        | 6.563         | 1.1           | —             | 3.0           | 0.611         | 7.7           | 3.6           | 4.163         |
|              | min  | 7.645         | 4.522         | —             | —             | 0.9           | —             | 5.1           | 1.2           | 2.040         |
|              | max  | 15.669        | 8.582         | 2.2           | —             | 5.1           | 1.118         | 10.4          | 6.2           | 7.036         |
| 2 ms         | mean | 9.261         | 5.550         | 1.2           | —             | 1.9           | 0.613         | 6.7           | 2.5           | 3.098         |
|              | min  | 7.630         | 4.517         | —             | —             | 0.7           | —             | 5.0           | 0.9           | 2.021         |
|              | max  | 12.528        | 6.588         | 2.4           | —             | 3.2           | 1.549         | 8.6           | 4.2           | 5.036         |
| 1 ms         | mean | 8.247         | 5.052         | 1.0           | —             | 1.6           | 0.607         | 6.0           | 2.2           | 2.588         |
|              | min  | 7.631         | 4.518         | —             | —             | 1.1           | —             | 4.8           | 1.2           | 2.018         |
|              | max  | 9.558         | 5.584         | 2.0           | —             | 2.3           | 1.117         | 7.2           | 3.2           | 4.008         |

Table 2.9: Use case transmission latencies  $R$  (“—” means under resolution).

| $T_{PN,nom}$ |      | $L_{16}$ (ms) | $L_{12}$ (ms) | $L_{23}$ (ms) | $L_{34}$ (ms) | $L_{45}$ (ms) | $L_{56}$ (ms) | $L_{13}$ (ms) | $L_{46}$ (ms) | $L_{25}$ (ms) |
|--------------|------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| 32 ms        | mean | 38.615        | 20.575        | 0.9           | 1.2           | 15.4          | 0.526         | 21.5          | 15.9          | 17.513        |
|              | min  | 7.598         | 4.613         | —             | 0.7           | 0.5           | —             | 5.2           | —             | 1.245         |
|              | max  | 70.606        | 36.619        | 2.8           | 3.0           | 31.5          | 1.092         | 38.0          | 32.4          | 33.997        |
| 16 ms        | mean | 22.632        | 12.620        | 0.9           | 1.2           | 7.4           | 0.526         | 13.5          | 7.9           | 9.486         |
|              | min  | 6.663         | 4.590         | —             | 0.8           | —             | —             | 5.1           | —             | 1.118         |
|              | max  | 38.536        | 20.669        | 2.8           | 2.9           | 15.5          | 1.077         | 22.0          | 16.4          | 17.873        |
| 8 ms         | mean | 14.688        | 8.602         | 0.7           | 1.2           | 3.7           | 0.525         | 9.3           | 4.3           | 5.561         |
|              | min  | 6.744         | 4.573         | —             | 0.7           | —             | —             | 4.7           | —             | 0.998         |
|              | max  | 23.375        | 12.714        | 3.0           | 2.5           | 7.8           | 1.081         | 13.7          | 8.7           | 11.248        |
| 4 ms         | mean | 10.731        | 6.634         | 0.7           | 1.2           | 1.7           | 0.527         | 7.3           | 2.2           | 3.570         |
|              | min  | 5.851         | 4.528         | —             | 0.7           | —             | —             | 4.7           | —             | 0.998         |
|              | max  | 15.163        | 8.696         | 2.4           | 2.9           | 3.8           | 1.080         | 9.9           | 4.7           | 6.378         |
| 2 ms         | mean | 8.723         | 5.615         | 0.7           | 1.2           | 0.7           | 0.527         | 6.3           | 1.3           | 2.581         |
|              | min  | 5.916         | 4.533         | —             | 0.8           | —             | —             | 4.7           | —             | 0.999         |
|              | max  | 14.755        | 6.690         | 2.1           | 2.9           | 1.8           | 1.079         | 8.4           | 2.7           | 4.754         |
| 1 ms         | mean | 7.634         | 5.114         | 0.8           | 1.2           | —             | 0.530         | 5.9           | 0.6           | 1.990         |
|              | min  | 5.857         | 4.533         | —             | —             | —             | —             | 4.8           | —             | 0.997         |
|              | max  | 10.527        | 5.693         | 2.8           | 3.0           | 0.8           | 1.080         | 8.3           | 1.6           | 4.625         |

### Summary

The core philosophy of this research is the transposition of robust Industry 4.0 and 5.0 paradigms, traditionally reserved for manufacturing, into personal safety and health monitoring. The stringent requirements for reliability, low latency, and determinism found in automated production lines are identical to those required for real-time heart rate monitoring or fall detection on a light vehicle. Our experimental results on vPLCs demonstrate that lightweight containerized virtualization (Docker) can manage complex control tasks with minimal jitter, suggesting that vPLCs can meet the real-time requirements of industrial control applications. Ensuring this temporal precision is not merely an industrial requirement but an enabling prerequisite for hosting safety-critical services, such as remote diagnostics or automated emergency alerts, on mobile edge platforms.

## 2.2 IoT Communication Protocols for Smart Mobility

### 2.2.1 LPWAN - LoRaWAN

The term LoRaWAN defines an open standard managed by the LoRa Alliance, a nonprofit association that brings together members from both the industry and the academia. In particular, the standard focuses on upper levels of the communication protocol stack. The radio, i.e., the Physical level (PHY), usually exploits LoRa modulation, an enhanced version of the chirp spread (CSS) spectrum technique operating in sub-GHz unlicensed bands. In LoRa, up to SF (a tunable parameter named Spreading Factor) bits per symbol can be coded [22]. Indeed, the fixed upchirp bandwidth  $B_{UC} = f_{HI} - f_{LO}$ , is divided into  $2^{SF}$  inter-

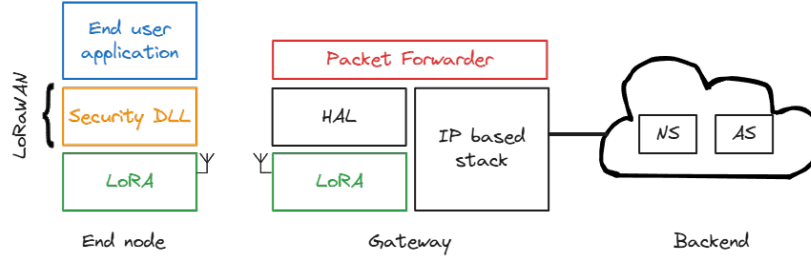


Figure 2.8: The stack of an hardware end node, the stack of a hardware gateway device, and the backend in cloud.

vals, and the actual initial upchirp frequency  $f_{START}$  is defined as  $f_{START}(i) = (B_{UC}/2^{SF}) \cdot i + f_{LO}, i = 0, \dots, SF - 1$ ; when the upchirp frequency reaches the value  $f_{HI}$ , the upchirp rolls over and starts increasing from  $f_{LO}$ . The upchirp duration  $T_{UC}$  depends on the  $SF$  value, in agreement with  $T_{UC} \cdot B_{UC} = 2^{SF}$ ; in other words, changing  $SF$  also modifies the data rate.

Regarding the Data Link Level (DLL), LoRaWAN exploits pure ALOHA, even if the specifications include possible Channel Activity Detection (CAD) for supporting Listen Before Talk (LBT) medium access strategy [23]. Data exchange is event-based [24]; the end device starts the transmission asynchronously and opens two receiving windows RX1 and RX2 after fixed but configurable amount of time from the frame end (namely,  $T_{RX1}$  and  $T_{RX2} = T_{RX1} + 1s$ ). Consequently, bidirectional traffic in the uplink and downlink directions is permitted; interesting to note, the in-phase and quadrature components of upchirps are swapped depending on the direction, in order to reduce the interference. Indeed, frames sent using different SF values are quasi-orthogonal, allowing to reduce the impact of resource scarcity in unlicensed bands.

The network topology is a star, at the center of which a gateway (GW) is located. Referring to Figure 2.8, the GW tunnels wireless uplink frames into the

backend wired backbone and vice versa. It is worth to notice that the GW actually consists of two parts: the first includes the radio interface, which is hidden under a hardware abstraction layer furnishing services to the packet forwarder; the second part is used by the packet forwarder for the (generally IP-based) connectivity with the backend [25].

LoRaWAN specifications define logical entities in the backend in charge on managing the network, the user integration and the ciphering key distribution. The Network Server (NS) has to deduplicate messages from the GWs, verify the integrity using network session key and provide acknowledgment for confirmed messages; it can also send commands for modifying node behaviors (e.g., for adaptive data rate strategies). The Application Server (AS) provides desired information to the end users, ensuring confidentiality by means of decryption via the application session key. Starting from specs release 1.0, keys are managed by the so-called Join Server (JS).

For the sake of clarity, the different stacks implanted in the end device and the GW of a LoRaWAN network are also visible in Figure 2.8. As regards the provisioning of a LoRaWAN node, it typically occurs using the Over The Air Activation (OTAA), i.e. a procedure, based on the exchange of a join request and join accept messages, for deriving the actual network and application session keys from root keys preliminary configured in both the end device and the backend.

As a concluding remark, it must be stressed that, operating in unlicensed bands, LoRaWAN transmissions in both directions must satisfy different norms, in terms of maximum transmitting power and duty cycle, depending on the region where the network is operated. For instance, in Europe the so called EU868 frequency plan is adopted, satisfying the ETSI regulation. Accordingly, the maximum power is  $\{14, 27\}$  dBm and the duty cycle is  $\{0.1\%, 1\%, 10\%\}$ , depending on the actual used channel. Therefore, in most cases uplink is the preferred di-

rection and downlink messages are minimized, or not used at all, to not consume the transmission opportunities in the opposite way. Advantages and limitations of LoRa and LoRaWAN for (massive) IoT applications have been deeply analyzed in the available literature, e.g. [26].

The applicability of LoRaWAN to mobile IoT scenarios has attracted growing research interest [27]. Comparative surveys of LPWAN technologies, including LoRaWAN, NB-IoT, and SigFox, have assessed their respective trade-offs in terms of range, capacity, and energy efficiency [28, 29]. Of particular relevance to the light-mobility context is the study by Al Mojamed [30], which investigates LoRaWAN performance specifically for *mobile* IoT applications, including bike-rental systems and vehicle fleet monitoring, using network simulation. The results confirm that LoRaWAN can sustain reliable uplink connectivity for slowly moving devices such as electric bicycles, even though signal-quality metrics degrade with mobility, an aspect that informs the network dimensioning choices adopted in the MOST project infrastructure.

### **Case Study: Hybrid LoRa/LoRaWAN Networks for Rural IoT Deployments**

A concrete illustration of LoRaWAN's flexibility and of the practical trade-offs between standard and custom LPWAN deployments is provided by a field experiment conducted in Praia, Cabo Verde [31]. The study targets precision irrigation in a rural, semi-arid environment where neither cellular coverage nor stable power supply can be assumed. Conditions that are representative of a broad class of off-grid IoT scenarios and that share several constraints with remote cycling-route monitoring in the MOST project context.

Each sensorised water tank is equipped with an ESP32-S3 node connected to a Lambda62-8D LoRa module and to a set of environmental and hydraulic sensors: a SEN55-SDN-T multi-parameter air-quality unit (particulate matter,

VOC, temperature, humidity), a SEN0312 ultrasonic water-level sensor, and a SEN0236 anemometer. An electrovalve controls irrigation. The study compares two communication configurations, illustrated in Figure 2.9, that share identical node hardware but differ in the protocol stack:

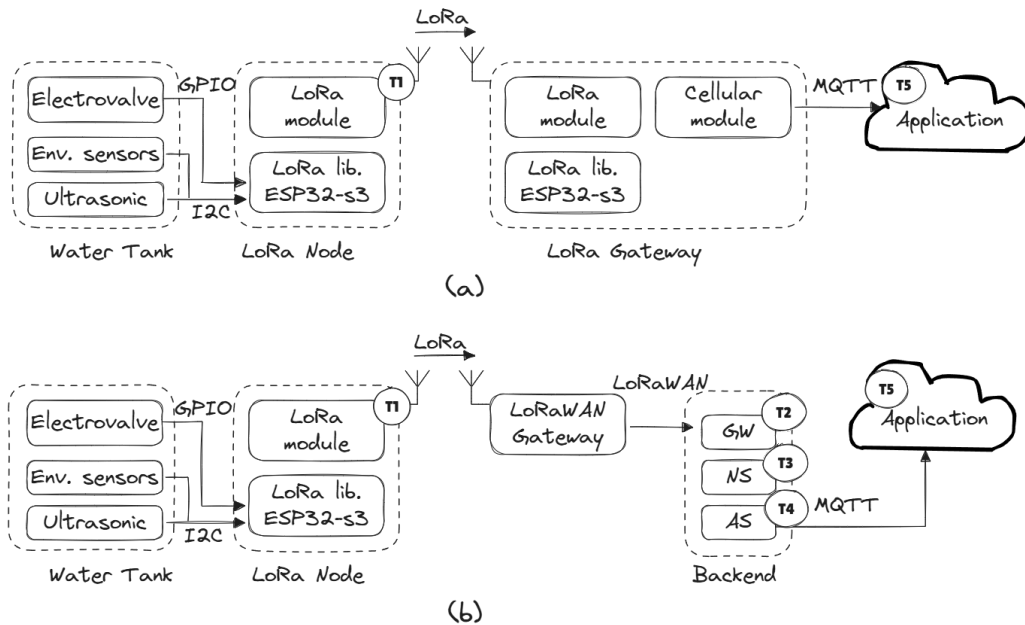


Figure 2.9: Hybrid LoRa/LoRaWAN system architecture deployed in Praia, Cabo Verde. (a) Custom LoRa configuration; (b) LoRaWAN configuration.

In the *custom LoRa* configuration, the node firmware communicates directly with the gateway over raw LoRa frames (no LoRaWAN MAC layer). The gateway, an ESP32-S3 board with a SIM800 cellular module, acts as a transparent bridge, forwarding received frames to the backend via MQTT over a cellular connection. In the *LoRaWAN* configuration, the same node hardware runs the MCCI LMIC library implementing the full LoRaWAN Class A MAC, and transmits at SF = 7, BW = 125 kHz in the EU868 band. Uplink packets are received by a Milesight UG64 commercial gateway and forwarded to The Things Stack (TTN) Network

Server, which delivers decoded payloads to the application via MQTT. Both data streams are unified in a Node-RED dashboard, making the hybrid architecture transparent to the end user.



Figure 2.10: Geographical deployment of the three sensor nodes and the gateway at the field site in Praia, Cabo Verde.

Figure 2.10 shows the geographical deployment of the three nodes at increasing distances from the gateway (N1: 26 m, N2: 151 m, N3: 267 m), enabling a comparative link-quality analysis across short, medium, and longer-range conditions within the same rural environment.

End-to-end latency is measured by embedding a firmware-generated timestamp  $T_1$  in each uplink payload and recording reception times at each hop of the LoRaWAN delivery chain ( $T_2$ : gateway,  $T_3$ : NS,  $T_4$ : AS,  $T_5$ : user application). The defined latency components are:

- $L_{15} = T_5 - T_1$ : Total end-to-end latency.

- $L_{12} = T_2 - T_1$ : Latency from node to gateway.
- $L_{23} = T_3 - T_2$ : Latency from gateway to network server.
- $L_{34} = T_4 - T_3$ : Latency from network server to application server.
- $L_{45} = T_5 - T_4$ : Latency from application server to user application.

In the custom LoRa configuration only  $L_{15}$  is measurable (no intermediate network-layer timestamps). Results are summarised in Table 2.10 and discussed below.

Table 2.10: Custom LoRa vs LoRaWAN ('-' values under std)

|      |        | LoRa    |                | LoRaWAN |         |         |         |         |            |          |       |
|------|--------|---------|----------------|---------|---------|---------|---------|---------|------------|----------|-------|
| Node | D. (m) | L15 (s) | $\Delta s$ (s) | L15 (s) | L12 (s) | L23 (s) | L34 (s) | L45 (s) | RSSI (dBm) | SNR (dB) |       |
| N1   | 26     | avg     | 0.118          | 606.187 | 61.024  | 0.016   | 60.789  | 0.206   | 0.012      | -63.16   | 13.31 |
|      |        | std     | 0.009          | 0.229   | 0.031   | 0.014   | 0.021   | 0.003   | 0.004      | 7.62     | 0.95  |
|      |        | 5perc   | 0.102          | 606.134 | 60.989  | -       | 60.757  | 0.203   | 0.008      | -75.00   | 11.50 |
|      |        | 95perc  | 0.142          | 606.251 | 61.067  | 0.033   | 60.820  | 0.211   | 0.018      | -49.00   | 14.00 |
| N2   | 151    | avg     | 0.132          | 606.244 | 63.232  | 0.015   | 63.002  | 0.205   | 0.010      | -91.30   | 10.36 |
|      |        | std     | 0.015          | 0.234   | 0.078   | 0.015   | 0.073   | 0.002   | 0.003      | 1.31     | 2.37  |
|      |        | 5perc   | 0.105          | 606.193 | 63.147  | -       | 62.887  | 0.203   | 0.007      | -93.50   | 5.75  |
|      |        | 95perc  | 0.148          | 606.208 | 63.329  | 0.033   | 63.116  | 0.209   | 0.015      | -89.00   | 13.80 |
| N3   | 267    | avg     | 0.124          | 606.312 | 63.179  | 0.017   | 62.947  | 0.208   | 0.013      | -104.08  | 6.68  |
|      |        | std     | 0.012          | 0.238   | 0.071   | 0.014   | 0.068   | 0.003   | 0.005      | 3.67     | 2.16  |
|      |        | 5perc   | 0.108          | 606.267 | 63.094  | -       | 62.841  | 0.205   | 0.009      | -110.00  | 2.80  |
|      |        | 95perc  | 0.145          | 606.374 | 63.261  | 0.032   | 63.072  | 0.213   | 0.019      | -99.00   | 9.00  |

The custom LoRa path achieves a sub-second end-to-end latency of approximately 0.12–0.13 s across all three distances, with minimal variability ( $\sigma < 15$  ms). The LoRaWAN path introduces a dramatically higher end-to-end latency of 61–63 s, almost entirely attributable to the  $L_{23}$  component (gateway-to-NS). The authors attribute this to the use of a free-tier TTN account operating in best-effort mode, where NS processing is not prioritised; dedicated private deploy-

ments, such as the ChirpStack-based infrastructure adopted in the MOST project, achieve significantly lower values, typically in the 0.2–1 s range [32]. The remaining components are negligible:  $L_{12} \approx 16$  ms (radio propagation plus gateway forwarding),  $L_{34} \approx 206$  ms (NS-to-AS routing), and  $L_{45} \approx 10$ –13 ms (MQTT delivery).

RSSI and SNR values, reported in Table 2.10, decrease monotonically with distance as expected. At the maximum tested range of 267 m, RSSI reaches  $-104$  dBm while SNR remains at 6.7 dB, well above the LoRa demodulation threshold for  $SF = 7$  (approximately  $-7.5$  dB). No packet losses or anomalous latency events are observed at any distance, confirming the robustness of the LoRaWAN link under rural conditions characterised by high humidity and partial vegetation obstructions. These conditions make the propagation considerably more challenging than those encountered on open cycling paths.

### 2.2.2 5G Mobile Networks

The fifth generation of mobile networks (5G) represents a substantial evolution from previous cellular standards, designed to satisfy a broad and heterogeneous set of application requirements that a single radio access technology can no longer address alone. Unlike 4G LTE, whose design is primarily driven by mobile broadband, 5G is built around three canonical service classes defined by 3GPP: enhanced Mobile BroadBand (eMBB), targeting peak data rates of tens of gigabits per second for bandwidth-intensive applications; massive Machine Type Communication (mMTC), optimised for dense deployments of low-power IoT devices with sporadic transmissions; and Ultra-Reliable Low-Latency Communication (URLLC), which targets one-way delays below 1 ms and packet error rates below  $10^{-5}$  for critical control applications [33, 34].

The 5G system is logically partitioned into two principal domains: the con-

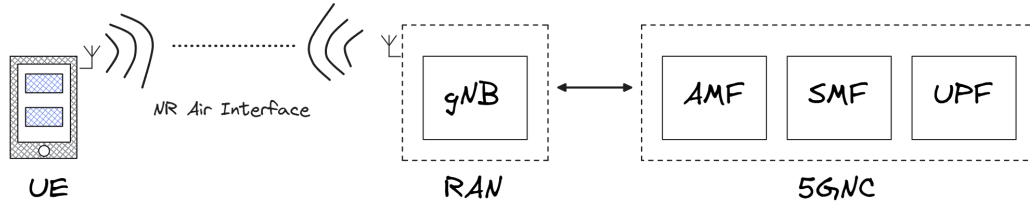


Figure 2.11: A representative 5G architecture.

sists of base stations referred to as gNodeBs (gNB), which Radio Access Network (RAN) and the Core Network (5GCN) as shown in Figure 2.11. The RAN communicate with User Equipment (UE) over the air interface using the New Radio (NR) standard. The 5GCN adopts a Service-Based Architecture (SBA), in which network functions are implemented as software modules that expose well-defined APIs and can be deployed as containerised microservices on commodity hardware. Key network functions include the Access and Management Function (AMF), which handles UE registration and mobility; the Session Management Function (SMF), which controls data sessions; and the User Plane Function (UPF), which routes and forwards user data between the RAN and external networks through the so-called N6 interface. This separation of the user plane from the control plane, inherited from the 4G paradigm and extended in 5G, enables independent scaling and placement of the two planes, which is a prerequisite for flexible network slicing.

The gNB implements the physical layer (PHY), the Medium Access Control (MAC) sublayer, and the Radio Link Control (RLC) and Packet Data Convergence Protocol (PDCP) layers. It connects to the 5GCN over the NG interface and to neighbouring gNBs over the Xn interface to support handover and coordinated transmission. On the radio side, 5G NR supports a flexible numerology in which the sub-carrier spacing can be set to  $15 \cdot 2^\mu$  kHz, with  $\mu \in \{0, 1, 2, 3, 4\}$ , allowing

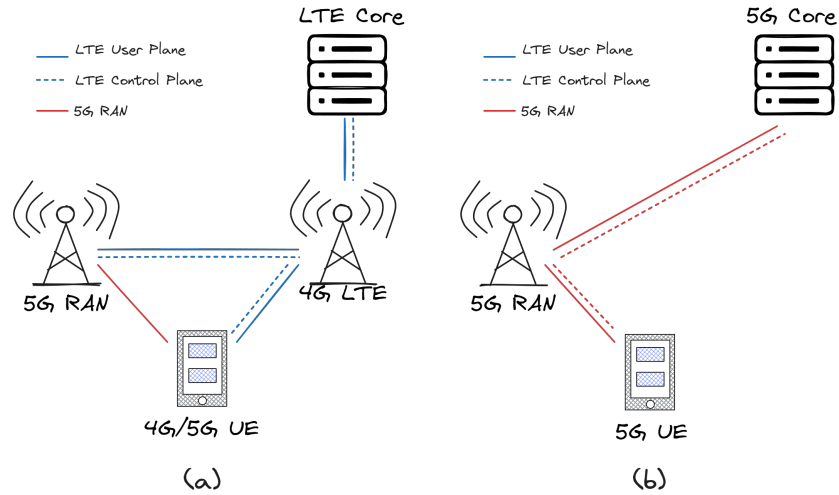


Figure 2.12: 5G deployments architecture: (a) Non-Standalone (NSA) and (b) Standalone (SA).

the same waveform to operate from sub-6 GHz bands (FR1) up to millimetre-wave frequencies (FR2). The physical resources are organized in Physical Resource Blocks (PRBs), each comprising 12 sub-carriers, and are allocated dynamically by the MAC scheduler on a per-slot basis in both uplink (UL) and downlink (DL).

Two deployment modes are defined and shown in Figure 2.12. Non-Standalone (NSA) mode anchors the 5G RAN to an existing 4G core for control-plane operations, reducing deployment cost at the expense of full 5G capability (Figure 2.12 (a)). Standalone (SA) mode operates exclusively on 5G network functions and unlocks the full feature set, including network slicing, edge-computing integration, and URLLC support (Figure 2.12 (b)).

### Open-Source Implementations and Private Networks

The softwarization of 5G network functions has enabled the release of several vendor-independent open-source frameworks that implement the complete protocol stack on commodity hardware. Notable examples include Open5GS, free5GC, and the Open Air Interface (OAI) project, the last of which is developed collaboratively by universities and industrial partners (EURECOM, Nokia Bell-Labs, Qualcomm, among others) with the explicit goal of providing a realistic and scalable research platform [35]. The OAI software stack covers both the 5GCN and the gNB, and can be interfaced with Software Defined Radio (SDR) front-ends such as the Ettus USRP family, enabling fully functional over-the-air experiments without proprietary radio hardware.

## 2.3 Summary

This chapter established the technical and methodological foundations of the thesis by demonstrating that the reliability, low latency, and determinism demanded by industrial automation are not exclusive to the factory floor, but constitute enabling prerequisites for the LEV-oriented contributions developed in subsequent chapters. The experimental results on virtualized PLCs show that lightweight containerized virtualization can manage industrial real-time tasks with minimal jitter, and, by extension, the real-time requirements of safety-critical services on mobile edge platforms (PS 1). Ensuring this temporal precision is not merely an industrial requirement: it directly enables the hosting of autonomous safety functions, such as remote diagnostics and emergency alerting, on resource constrained on-board hardware. The industrial case studies presented here serve as the performance baseline against which the LEV-oriented contributions of Chapters 3 through 6 are evaluated.

## Chapter 3

# IoT Communication Protocols in Smart Mobility Scenarios

Modular and disaggregated architectures (PS 1) are only as effective as the communication infrastructure that connects their components. This chapter investigates how industrial communication paradigms, built on the principle of protocol independence and multi-access connectivity, can be transposed to smart mobility scenarios. The background on LoRaWAN LPWAN technology and 5G Standalone architectures has been laid out in Chapter 2; the present chapter moves from technology description to original research contributions, presenting four experimental studies that collectively characterise and extend the communication infrastructure available for smart-campus and sustainable-mobility applications. The first three studies share a common design philosophy: the application of software-defined and containerization principles to the end-device layer of a LoRaWAN network. The first study proposes and evaluates three architectures of increasing virtualization depth, Minimal, Hybrid, and Full, in which the application logic, the LoRaWAN protocol stack, and the radio hardware are

progressively disaggregated into isolated Docker containers, and quantifies the latency overhead introduced at each decomposition level [36]. The second study builds on the Full architecture to construct a complete end-to-end LoRaWAN emulator capable of reproducing smart-city scenarios with hundreds of concurrent nodes on commodity hardware or affordable cloud instances, with application to a real multi-utility deployment in Brescia [37]. The third study pushes the decomposition down to the physical layer, replacing the hardware LoRa modem with a Software Defined Radio (SDR) and moving the LoRa baseband modulation into a dedicated container, yielding a fully software-defined and reconfigurable LPWAN node [38]. The fourth study evaluates the end-to-end communication performance of a 5G Standalone network deployed on the University of Brescia smart campus, measuring one-way and two-way delay under realistic interfering load conditions using an open-source OAI-based core and a USRP SDR base station [39]. The 5G infrastructure is envisioned as the complementary high-bandwidth link to the LoRaWAN layer in a multi-access campus architecture. Together, these four contributions define the communication envelope within which the monitoring and mobility platforms described in subsequent chapters operate.

### **3.1 LoRaWAN End-Device Disaggregation via Lightweight virtualization**

The growing heterogeneity of IoT deployments has pushed practitioners and researchers to reconsider the rigid, hardware-centric design of Low-Power Wide-Area Network (LPWAN) nodes. In most commercial LoRaWAN deployments the end device is a monolithic piece of hardware that combines the radio front-end,

the LoRaWAN protocol stack, and the application logic on a single, resource-constrained microcontroller. While this approach minimises cost and power consumption for simple, single-function sensing tasks, it becomes a limiting factor whenever flexibility, reconfigurability, or integration with heterogeneous systems is required.

Concepts borrowed from the fifth-generation (5G) mobile-network domain, notably *disaggregation*, i.e. the separation of software from its underlying hardware, and *decomposition*, i.e. the splitting of different protocol-layer functions across distinct execution environments, offer a principled way to overcome these limitations [40, 41]. Applying analogous ideas to LoRaWAN end devices opens the door to a class of implementations in which each functional block of the communication stack (application logic, LoRaWAN security and data-link layer, packet-forwarder) can be individually virtualized and deployed as a lightweight software container.

### 3.1.1 Proposed Architectures

Starting from the reference LoRaWAN stack decomposition shown in Figure 2.8, in [36] three containerization architectures of increasing virtualization depth are considered.

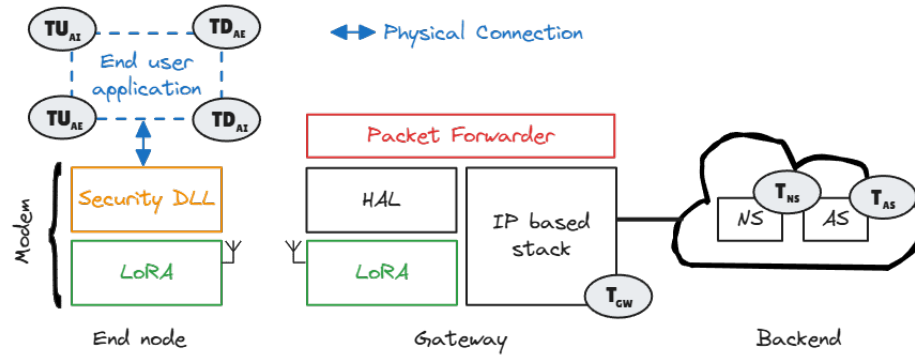


Figure 3.1: Simplified diagram of the first scenario (Minimal); dotted boxes represent containers. Ovals define timestamping points.

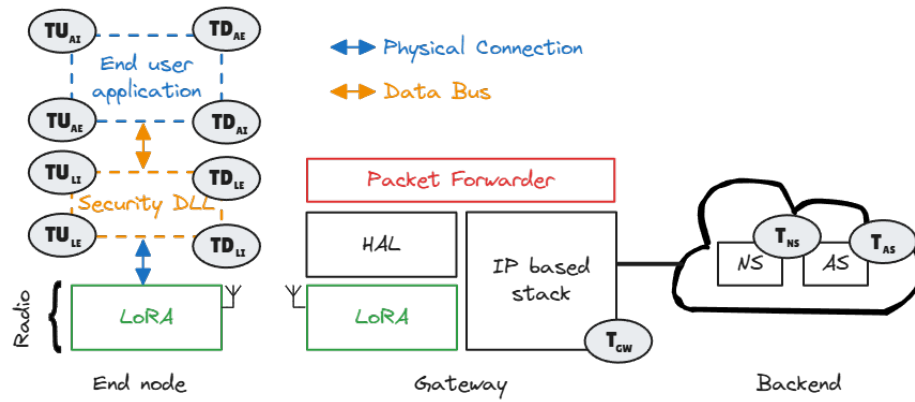


Figure 3.2: Simplified diagram of the second scenario (Hybrid); dotted boxes represent containers. Ovals define timestamping points.

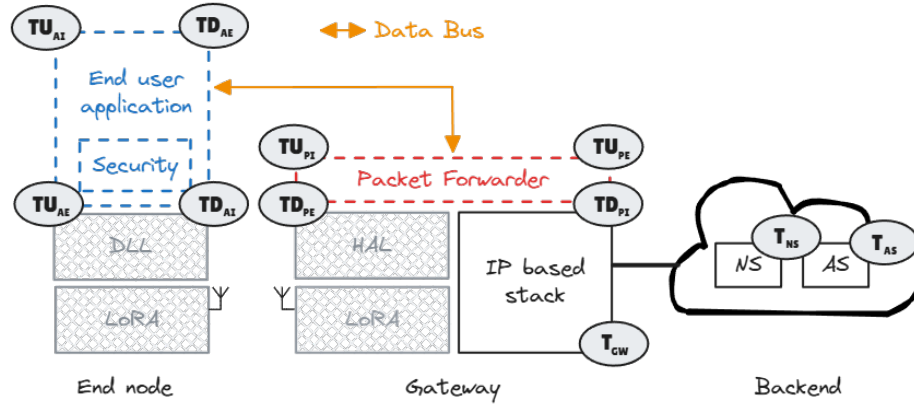


Figure 3.3: Simplified diagram of the third scenario (Full); dotted boxes represent containers. Ovals define timestamping points.

In the *Minimal* architecture (Figure 3.1) only the end-user application is virtualized inside a container. The LoRaWAN stack and the LoRa radio remain co-located on a dedicated hardware modem, which is accessible from the container host via a serial link (e.g. a USB virtual COM port). This configuration requires the smallest software effort and leaves the most safety-critical timing logic on proven, certified hardware, while still allowing the application to be deployed, updated, and monitored through standard container-orchestration tooling.

The *Hybrid* architecture (Figure 3.2) extends virtualization to the LoRaWAN protocol stack: both the application and the LoRaWAN security and data-link layers run in separate, dedicated containers that communicate through a message-oriented middleware (e.g. MQTT over a virtual Docker network). Only the LoRa radio remains on external hardware, connected to the container host via a serial link. Compared with the Minimal case, this configuration decouples software updates of the application from updates of the LoRaWAN stack, and makes intermediate protocol states accessible for inspection and testing.

The *Full* architecture (Figure 3.3) is the most radical: the application, the Lo-

RaWAN security and data-link management, and the packet-forwarder logic all run as containers. Crucially, no physical LoRa radio is needed. LoRaWAN frames are transported between the packet-forwarder container and the backend using any IP-based transport (e.g. the standard Semtech UDP protocol), so the wireless link is entirely replaced by a wired or virtual network. The result is a fully virtual LoRaWAN node that is indistinguishable from a hardware node from the perspective of the existing backend infrastructure.

This architecture is particularly powerful for scenarios in which non-LoRaWAN field devices, connected via fieldbuses, Ethernet, or other wired protocols, need to be exposed to a LoRaWAN backend with minimal integration effort: the containerized stack acts as a transparent adapter, applying standard LoRaWAN encapsulation and cryptographic protection to payloads that originate outside the radio domain.

### 3.1.2 Experimental Evaluation

#### Setup and Configurations

All three architectures are implemented on both a physical Industrial Edge Device (IED) and a virtualized Industrial Edge Virtual Device (IEVD) running inside a commercial hypervisor, in order to characterise the performance impact of the additional virtualization layer. The configurations evaluated are summarised in Table 3.1.

Containers within the same host are interconnected via an MQTT-based industrial data bus. A three-hour experiment is conducted for each configuration, transmitting one uplink frame every 60 seconds with a 40-byte random payload (SF7, CR 4/5, resulting in a 103 ms air time). For each uplink, a corresponding downlink frame (2-byte payload) is scheduled at the backend; the LoRaWAN

Table 3.1: Experimental configurations.

| Arch | Setup | Application | LoRaWAN Stack | Packet Forwarder |
|------|-------|-------------|---------------|------------------|
| 1    | 1     | IED         | RN2483        | RG186            |
|      | 2     | IEVD        | RN2483        | RG186            |
| 2    | 1     | IED         | IED           | RG186            |
|      | 2     | IEVD        | IEVD          | RG186            |
| 3    | 1     | IED         | IED           | IED              |
|      | 2     | IEVD        | IEVD          | IEVD             |

Network Server forwarded it in the first available receive window (RX1, with  $T_{RX1} = 5$  s).

### Time-Related Metrics

A set of timestamping points is defined at the ingress and egress of each containerized functional block (application, LoRaWAN stack, packet forwarder) in both the uplink and downlink directions. From these timestamps the following latency metrics are derived:

- $\Delta U_A = T_{UA_E} - T_{UA_I}$ ,  $\Delta D_A = T_{DA_E} - T_{DA_I}$ : processing time inside the application container for uplink and downlink messages, respectively.
- $\Delta U_L = T_{UL_E} - T_{UL_I}$ ,  $\Delta D_L = T_{DL_E} - T_{DL_I}$ : processing time inside the LoRaWAN stack container.
- $\Delta U_P = T_{UP_E} - T_{UP_I}$ ,  $\Delta D_P = T_{DP_E} - T_{DP_I}$ : processing time inside the packet-forwarder container.
- $\Delta U_{GW} = T_{UGW_E} - T_{UGW_I}$ : elapsed time from application trigger to gateway reception.

- $\Delta U_{NS} = T_{UNSE} - T_{UNSI}$ ,  $\Delta U_{AS} = T_{UASE} - T_{UASI}$ : incremental latency from gateway to Network Server and from Network Server to Application Server, respectively.
- $\Delta UD = T_{DAE} - T_{UAI}$ : overall round-trip time of a complete uplink-downlink transaction.

Time synchronisation between the edge device and the backend is achieved via NTP, with a measured standard uncertainty of approximately 1.3 ms over three-hour runs (reduced to 0.9 ms when considering differences of same-clock timestamps).

## Results

The experimental results for Architectures 1, 2, and 3 on both the IED and the IEVD are reported in Tables 3.2–3.4.

Table 3.2: Architecture 1 (Minimal): latency statistics for IED (Setup 1) and IEVD (Setup 2).

| Setup | (ms)    | $\Delta U_A$ | $\Delta U_{GW}$ | $\Delta U_{NS}$ | $\Delta U_{AS}$ | $\Delta D_A$ |
|-------|---------|--------------|-----------------|-----------------|-----------------|--------------|
| 1     | Avg     | 0.9          | 201.2           | 16.1            | 207.0           | 2.1          |
|       | Std     | 0.3          | 8.6             | 7.9             | 3.4             | 0.5          |
|       | Median  | 1.1          | 199.5           | 19.7            | 205.9           | 2.3          |
|       | 05 Perc | 0.4          | 189.8           | 0.8             | 204.3           | 1.2          |
|       | 95 Perc | 1.3          | 215.2           | 20.7            | 212.2           | 2.6          |
| 2     | Avg     | 0.4          | 1411.8          | 6.6             | 216.2           | 1.1          |
|       | Std     | 0.2          | 706.7           | 8.2             | 5.4             | 0.5          |
|       | Median  | 0.4          | 1725.0          | 1.2             | 214.7           | 1.0          |
|       | 05 Perc | 0.1          | 269.8           | 0.9             | 212.0           | 0.3          |
|       | 95 Perc | 0.8          | 2165.5          | 19.1            | 231.6           | 2.1          |

Table 3.3: Architecture 2 (Hybrid): latency statistics for IED (Setup 1) and IEVD (Setup 2).

| Setup | (ms)    | $\Delta U_A$ | $\Delta U_L$ | $\Delta U_{GW}$ | $\Delta U_{NS}$ | $\Delta U_{AS}$ | $\Delta D_A$ | $\Delta D_L$ |
|-------|---------|--------------|--------------|-----------------|-----------------|-----------------|--------------|--------------|
| 1     | Avg     | 0.7          | 13.5         | 168.8           | 20.0            | 209.1           | 0.4          | 29.0         |
|       | Std     | 0.3          | 0.8          | 7.1             | 1.7             | 3.7             | 0.4          | 22.1         |
|       | Median  | 0.7          | 13.3         | 169.4           | 19.7            | 208.2           | 0.3          | 14.0         |
|       | 05 Perc | 0.3          | 12.8         | 158.5           | 19.0            | 206.5           | 0.2          | 3.7          |
|       | 95 Perc | 1.0          | 14.2         | 177.2           | 20.7            | 215.7           | 0.8          | 54.3         |
| 2     | Avg     | 0.2          | 451.4        | 602.7           | 23.3            | 210.4           | 0.2          | 17.3         |
|       | Std     | 0.1          | 281.4        | 280.8           | 1.5             | 3.9             | 0.1          | 19.7         |
|       | Median  | 0.2          | 382.3        | 532.2           | 22.7            | 209.5           | 0.2          | 4.9          |
|       | 05 Perc | 0.1          | 64.1         | 219.1           | 22.4            | 206.8           | 0.1          | 1.0          |
|       | 95 Perc | 0.4          | 945.7        | 1096.2          | 24.7            | 219.7           | 0.3          | 47.3         |

Table 3.4: Architecture 3 (Full): latency statistics for IED (Setup 1) and IEVD (Setup 2).

| Setup | (ms)    | $\Delta U_A$ | $\Delta U_P$ | $\Delta U_{GW}$ | $\Delta U_{NS}$ | $\Delta U_{AS}$ | $\Delta D_A$ | $\Delta D_P$ |
|-------|---------|--------------|--------------|-----------------|-----------------|-----------------|--------------|--------------|
| 1     | Avg     | 1.0          | 0.2          | 10.6            | 23.4            | 207.1           | 0.7          | 0.5          |
|       | Std     | 0.5          | 0.2          | 4.6             | 19.8            | 3.6             | 0.5          | 0.1          |
|       | Median  | 1.1          | 0.2          | 10.1            | 21.8            | 205.9           | 0.6          | 0.5          |
|       | 05 Perc | 0.4          | 0.1          | 5.3             | 21.3            | 204.1           | 0.3          | 0.3          |
|       | 95 Perc | 1.4          | 0.4          | 15.2            | 23.9            | 213.9           | 1.0          | 0.7          |
| 2     | Avg     | 0.3          | 0.3          | 7.6             | 27.3            | 207.3           | 0.2          | 0.2          |
|       | Std     | 0.3          | 0.1          | 4.2             | 4.3             | 3.5             | 0.1          | 0.1          |
|       | Median  | 0.3          | 0.3          | 7.3             | 26.6            | 206.5           | 0.2          | 0.2          |
|       | 05 Perc | 0.1          | 0.1          | 2.5             | 23.1            | 204.5           | 0.1          | 0.1          |
|       | 95 Perc | 0.5          | 0.5          | 11.7            | 32.3            | 214.0           | 0.4          | 0.3          |

Several key observations emerge from the results. First, metrics that depend exclusively on the backend ( $\Delta U_{NS}$ ,  $\Delta U_{AS}$ ) are consistent across all configurations, as expected. Second, application-level latencies ( $\Delta U_A$ ,  $\Delta D_A$ ) are systematically higher on the IED than on the IEVD, reflecting the more capable host CPU of the latter. Third, and most notably, the virtualized serial COM port used by the IEVD to communicate with the LoRaWAN modem (Architectures 1 and 2) introduces significant and highly variable latency in  $\Delta U_{GW}$  and  $\Delta U_L$  (roughly an order of magnitude above the physical IED; Figure 3.4). This behaviour is attributed to the polling-based management of the virtualized COM port inside the hypervisor.

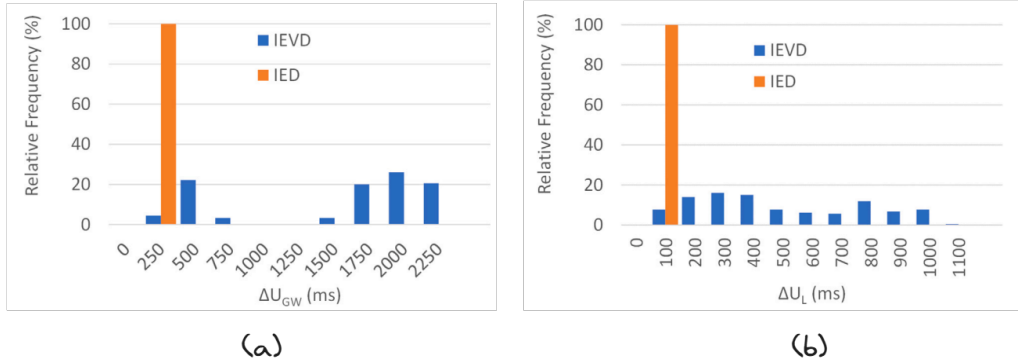


Figure 3.4: (a) Distribution of  $\Delta U_{GW}$  for Architecture 1, (b) distribution of  $\Delta U_L$  for Architecture 2.

Architecture 3 (Full) eliminates the serial port bottleneck entirely, as no physical radio is involved. As a result, the overall round-trip time  $\Delta U D$  drops below 5 s, a significant reduction compared with the 5.2 s typical of Architectures 1 and 2, which are constrained by the 5 s RX1 window of the LoRaWAN Class A timing.

### Comparison with Non-containerized Implementation

To quantify the overhead introduced by containerization, the three architectures are compared against a baseline non-containerized implementation in which the same RN2483 modem is driven directly by a native (non-containerized) application. The overall uplink trip time  $\Delta U$  and the round-trip time  $\Delta UD$  for all scenarios are reported in Table 3.5.

Table 3.5: Comparison of non-containerized vs. containerized architectures: overall uplink trip time  $\Delta U$  and round-trip time  $\Delta UD$  for IED (Setup 1) and IEVD (Setup 2).

| Setup | (ms)    | Not containerized |             | Minimal    |             | Hybrid     |             | Full       |             |
|-------|---------|-------------------|-------------|------------|-------------|------------|-------------|------------|-------------|
|       |         | $\Delta U$        | $\Delta UD$ | $\Delta U$ | $\Delta UD$ | $\Delta U$ | $\Delta UD$ | $\Delta U$ | $\Delta UD$ |
| 1     | Avg     | 413.3             | 5229.3      | 424.3      | 5269.8      | 398.0      | 5249.7      | 241.1      | 4256.5      |
|       | Std     | 5.9               | 2.8         | 6.9        | 2.6         | 8.5        | 22.9        | 23.2       | 23.2        |
|       | Median  | 413.1             | 5229.5      | 423.2      | 5269.4      | 397.9      | 5238.1      | 239.0      | 4254.9      |
|       | 05 Perc | 405.9             | 5227.5      | 414.9      | 5267.5      | 386.5      | 5224.2      | 232.7      | 4248.7      |
|       | 95 Perc | 430.2             | 5264.7      | 435.2      | 5274.7      | 409.2      | 5276.9      | 248.1      | 4261.9      |
| 2     | Avg     | 1654.4            | 6488.1      | 1634.6     | 6508.1      | 836.4      | 5678.9      | 242.2      | 4259.2      |
|       | Std     | 716.3             | 718.7       | 706.7      | 717.2       | 280.4      | 281.8       | 7.1        | 6.5         |
|       | Median  | 1889.4            | 6789.3      | 1949.8     | 6799.8      | 764.7      | 5595.9      | 241.1      | 4258.5      |
|       | 05 Perc | 464.3             | 5332.3      | 484.9      | 5328.3      | 456.4      | 5301.8      | 233.9      | 4251.5      |
|       | 95 Perc | 2306.6            | 7242.1      | 2386.6     | 7251.1      | 1330.6     | 6164.5      | 256.1      | 4267.4      |

The non-containerized and Minimal containerized scenarios exhibit nearly identical performance, confirming that the overhead associated with Docker-based lightweight virtualization is negligible for typical LoRaWAN payloads and transmission rates. The Full architecture, by eliminating the radio link and its associated timing constraints, consistently achieves the lowest round-trip latency across all configurations.

## 3.2 End-to-End Emulation of LoRaWAN Infrastructure for Smart City Scenarios

The Full containerised architecture introduced in Section 3.1 removes any dependency on physical radio hardware, making every component of the end-to-end LoRaWAN stack, end nodes, gateway, and backend, available as a Docker image that can be instantiated on a single host. This property has a direct and practically important consequence: the same images used in a real deployment can be composed into a complete network emulator without procuring any hardware. Unlike system-level simulators such as FLoRa (OMNeT++) or ns-3-based tools, which abstract protocol behaviour into high-level models, the container-based emulator runs the actual production code, including OTAA provisioning, AES-CMAC integrity checks, and AES-CTR payload ciphering, and can therefore expose implementation-level bottlenecks that a simulator would miss.

This section characterizes the scalability of such an emulation platform, targeting the node densities encountered in real smart-city deployments such as the A2A initiative in Brescia and, more broadly, the sustainable-mobility scenarios of the MOST project, where hundreds of concurrent sensors must be managed on affordable infrastructure.

### 3.2.1 Reference Smart-City Scenario

The emulation targets a smart-city deployment representative of real-world large-scale IoT infrastructures, specifically inspired by the A2A Smart City initiative in Brescia (Italy), which operates approximately 3000 installed sensors and a publicly accessible Application Server hosting more than 500 active devices. The initiative is launched in 2016 and leverages LoRaWAN as the communication in-

infrastructure for diverse use cases including smart parking, smart metering, smart bin management, and sustainable mobility services. The backend is based on the proprietary PTNetSuite network manager and the CityEye application platform, offering REST APIs for data retrieval and user-facing dashboards.

For the emulation experiments, the open-source ChirpStack backend has been adopted as the reference Network and Application Server, owing to its full LoRaWAN compliance and native Docker support. Each emulated end node follows the Class A uplink-centric behaviour described in Section 2.2.1: it transmits one uplink message every 20 s per node, and no downlink traffic is considered.

### 3.2.2 Emulation Architecture

The emulation topology, depicted in Figure 3.5, is hosted on a single physical machine (Host Z) running two virtual machines, VMx and VMy, interconnected by a virtual network at port 1700 (Semtech UDP). VMx hosts the ChirpStack backend (NS + AS + JS, each in its own container). VMy hosts all the emulated field-level entities:  $N$  end-node containers, one gateway container, an Eclipse-Mosquitto MQTT broker (image v1.6.14), and a Controller container implemented in Node-RED that orchestrates the emulation by triggering each end node at the appropriate time.

The number of end nodes is swept over  $N \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512\}$ , covering the range from a minimal single-device test up to a density comparable with the A2A Smart City deployment. Each experiment lasts 15 min. No resource limits are enforced in the Compose files, so that the natural footprint of the containers is measured without artificial constraints.

A hard architectural limit of 1024 containers per virtual machine exists, imposed by the maximum number of ports supported by the Linux bridge kernel driver. For larger networks, the topology can be extended by distributing end

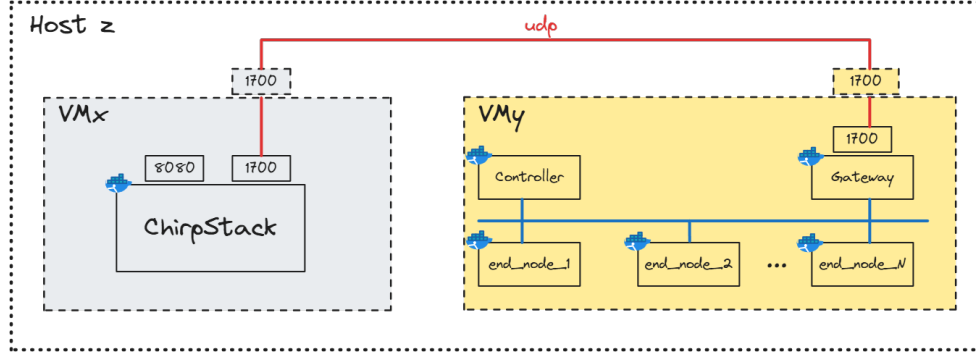


Figure 3.5: Topological view of the proposed end-to-end LoRaWAN emulation setup. VMx hosts the ChirpStack backend; VMy hosts  $N$  containerized end nodes, the gateway, the MQTT broker, and the Controller.

nodes across additional virtual machines, each with its own MQTT broker, while the Controller, gateway, and backend remain centralised.

### 3.2.3 Performance Metrics

Container-level resource usage is collected via `docker stats` [42], which queries the cgroup hierarchy (`/sys/fs/cgroup/. . .`) for CPU and memory figures, and via `netstat` [43] for network I/O bytes. The per-container metrics are defined as:

$$\text{CPU} = \frac{\Delta \text{cpuTime}_{\text{container}}}{\Delta \text{cpuTime}_{\text{host}}} \times N_{\text{CPU}} \times 100 \% \quad (3.1)$$

$$\text{MEM} = \text{memTotalUsage} - \text{memCache} \quad (3.2)$$

$$\text{NET} = \text{NET\_I} + \text{NET\_O} \quad (3.3)$$

Where  $\Delta \text{cpuTime}_{\text{container}}$  is obtained subtracting the previous CPU time from the current one,  $\Delta \text{cpuTime}_{\text{host}}$  is the elapsed time since the last measurement

and  $N_{\text{CPU}}$  is the number of CPUs that the container is using. MEM is calculated as the total memory usage of the container (memTotalUsage) minus the cache (memCache), to reflect the actual memory footprint of the container. NET is the sum of incoming and outgoing bytes.

The host-level aggregates are denoted TCPU, TMem, and TNET, and represent the overall burden imposed on the virtual machine by the full emulation. These are the primary indicators for scalability assessment.

### 3.2.4 Experimental Platforms

Two classes of emulation platforms are evaluated.

#### On-Premises (Local) Platform

Two local hardware configurations are considered, summarised in Table 3.6. Local Use Case 1 runs on an AMD Ryzen 5 5600U host (40 GB RAM, Windows 11) with VMware Workstation 17 as hypervisor; Local Use Case 2 runs on an Intel Core i7-8550U host (16 GB RAM) with VirtualBox 7. Both VMs are assigned 1–2 cores and 4–8 GB of RAM and run Ubuntu 20.04 or Ubuntu 22.04 with Docker engine 19–24.

#### Cloud Platform

Three Google Cloud configurations are evaluated (see Table 3.7), differing in the number of vCPUs assigned to VMs and in the processor series (N2 fixed-frequency vs. E2 variable-availability). All instances run Debian 12 with Docker 25.0.2 and Docker Compose 2.24.5.

Table 3.6: Local host and virtual-machine configurations.

|                 | Local Use Case 1          | Local Use Case 2    |
|-----------------|---------------------------|---------------------|
| Host CPU        | AMD Ryzen 5 5600U         | Intel Core i7-8550U |
| Host RAM        | 40.0 GB                   | 16.0 GB             |
| Host OS         | Windows 11 Pro            | Windows 11 Home     |
| Hypervisor      | VMware Workstation 17.0.2 | VirtualBox 7.0.6    |
| VMx cores / RAM | 1 core / 8 GB             | 2 cores / 2 GB      |
| VMy cores / RAM | 1 core / 8 GB             | 2 cores / 4 GB      |
| VMy OS          | Ubuntu 20.04.6 LTS        | Ubuntu 22.04 LTS    |
| Docker engine   | 19.03.15                  | 24.0.5              |

Table 3.7: Cloud virtual-machine configurations (Google Cloud).

|                    | Cloud UC 1           | Cloud UC 2           | Cloud UC 3 |
|--------------------|----------------------|----------------------|------------|
| VMx series / vCPUs | N2 / 2               | N2 / 2               | E2 / 2     |
| VMx price          | 0.11 USD/h           | 0.11 USD/h           | 0.10 USD/h |
| VMy series / vCPUs | N2 / 2               | N2 / 4               | E2 / 2     |
| VMy price          | 0.11 USD/h           | 0.17 USD/h           | 0.10 USD/h |
| VMy OS             | Debian 12            | Debian 12            | Debian 11  |
| CPU type           | Fixed (Cascade Lake) | Fixed (Cascade Lake) | Variable   |

### 3.2.5 Results

#### On-Premises Emulation

Tables 3.8 reports the maximum CPU, MEM, and NET usage of the Controller, gateway, and Eclipse-Mosquitto containers for the two local configurations.

Table 3.8: Local use cases: maximum resource usage of infrastructure containers (Controller, Gateway, Eclipse-Mosquitto) vs. number of end nodes  $N$ .

| Metric   | $N$ | Local UC 1 |        |       | Local UC 2 |        |       |
|----------|-----|------------|--------|-------|------------|--------|-------|
|          |     | Ctrl       | GW     | Mosq. | Ctrl       | GW     | Mosq. |
| CPU (%)  | 1   | 7.89       | 1.18   | 0.21  | 0.20       | 0.94   | 0.13  |
|          | 2   | 0.98       | 1.46   | 0.21  | 0.32       | 1.02   | 0.15  |
|          | 4   | 7.68       | 1.13   | 0.50  | 1.36       | 0.69   | 0.14  |
|          | 8   | 6.85       | 1.38   | 0.86  | 0.90       | 1.03   | 0.27  |
|          | 16  | 2.02       | 1.25   | 0.40  | 1.11       | 1.14   | 0.57  |
|          | 32  | 7.20       | 1.62   | 0.69  | 1.01       | 1.25   | 0.42  |
|          | 64  | 2.98       | 1.07   | 1.56  | 1.79       | 1.58   | 0.50  |
|          | 128 | 2.85       | 1.57   | 5.95  | 1.41       | 1.42   | 5.69  |
| MEM (MB) | 1   | 110.9      | 0.94   | 0.66  | 99.1       | 3.24   | 0.72  |
|          | 2   | 109.3      | 0.96   | 0.66  | 93.4       | 3.27   | 0.73  |
|          | 4   | 111.2      | 1.03   | 0.66  | 89.6       | 1.07   | 0.73  |
|          | 8   | 94.1       | 1.10   | 0.70  | 95.0       | 1.35   | 3.09  |
|          | 16  | 86.5       | 1.24   | 0.73  | 109.3      | 1.20   | 0.75  |
|          | 32  | 93.3       | 1.45   | 0.76  | 124.4      | 5.72   | 4.53  |
|          | 64  | 118.0      | 0.86   | 1.88  | 131.3      | 1.93   | 0.88  |
|          | 128 | 115.4      | 2.96   | 1.02  | 132.9      | 2.25   | 1.08  |
| NET (kB) | 1   | 57         | 78.5   | 121   | 1855       | 61.5   | 88    |
|          | 2   | 66         | 137.9  | 230   | 1764       | 107.0  | 175   |
|          | 4   | 77         | 240.6  | 441   | 1003       | 182.1  | 332   |
|          | 8   | 402        | 443.0  | 864   | 1113       | 320.0  | 666   |
|          | 16  | 4410       | 905.0  | 1916  | 11561      | 694.0  | 1497  |
|          | 32  | 584        | 1677.0 | 4045  | 5629       | 1800.6 | 4639  |
|          | 64  | 3860       | 9830.4 | 3318  | 7598       | 2908.2 | 9114  |
|          | 128 | 1149       | 7137.3 | 24136 | 10455      | 4761.6 | 24371 |

Memory and network usage scale approximately linearly with  $N$  in both local configurations. CPU usage of the Eclipse-Mosquitto broker is particularly significant: it jumps sharply between 64 and 128 nodes (from 1.6 % to 6.0 % in UC 1 and from 0.5 % to 5.7 % in UC 2), most likely because the default broker settings become saturated by the growing number of MQTT topics and subscriptions. The total TCPU, shown in Fig 3.6(a), usage at 128 nodes reaches about 30 % in UC 1 and 28 % in UC 2, confirming that a standard off-the-shelf PC can emulate full-stack LoRaWAN networks up to this scale.

### Cloud Emulation and Scalability

Table 3.9 reports the corresponding measurements for the three cloud configurations, extended to  $N \in \{128, 256, 512\}$ .

Table 3.9: Cloud use cases: maximum resource usage of infrastructure containers vs. number of end nodes  $N$ .

| Metric   | $N$ | Cloud UC 1 |       |        | Cloud UC 2 |       |        | Cloud UC 3 |       |        |
|----------|-----|------------|-------|--------|------------|-------|--------|------------|-------|--------|
|          |     | Ctrl       | GW    | Mosq.  | Ctrl       | GW    | Mosq.  | Ctrl       | GW    | Mosq.  |
| CPU (%)  | 128 | 1.45       | 0.52  | 0.40   | 0.65       | 0.20  | 2.16   | 2.21       | 0.97  | 12.19  |
|          | 256 | 1.51       | 0.51  | 11.43  | 0.58       | 0.35  | 8.86   | 2.26       | 0.90  | 8.08   |
|          | 512 | 1.13       | 0.52  | 16.84  | 0.54       | 0.24  | 10.94  | 1.63       | 0.95  | 12.20  |
| MEM (MB) | 128 | 116.0      | 2.88  | 1.07   | 115.5      | 2.84  | 1.32   | 117.5      | 2.75  | 1.02   |
|          | 256 | 122.3      | 2.90  | 1.43   | 119.9      | 6.16  | 1.68   | 116.5      | 4.58  | 2.37   |
|          | 512 | 127.0      | 7.93  | 2.13   | 121.4      | 8.36  | 4.93   | 115.8      | 8.23  | 4.48   |
| NET (kB) | 128 | 2576       | 6676  | 23951  | 2533       | 6574  | 26931  | 2540       | 6502  | 26419  |
|          | 256 | 5712       | 6973  | 39834  | 10547      | 18217 | 75162  | 5110       | 13066 | 90726  |
|          | 512 | 10291      | 24402 | 291840 | 14090      | 26092 | 288768 | 10230      | 24955 | 325632 |

The key findings are: (i) memory consumption is independent of the number of vCPUs and scales linearly, reaching approximately 500 MB total at  $N = 512$ ;

(ii) CPU usage benefits from additional vCPUs only beyond  $N = 128$  nodes, below this threshold, the overhead of Docker's CPU scheduler outweighs the parallelism gain; (iii) network traffic, dominated by Eclipse-Mosquitto, grows super-linearly above  $N = 256$  due to the accumulation of MQTT keep-alive messages when containers are queued waiting to complete transactions on a saturated host. Cloud Use Case 3, using an E2 instance with variable CPU allocation, provides the lowest cost but yields non-deterministic TCPU behaviour (Fig 3.6(b)), posing a risk of emulation failures at high node counts.

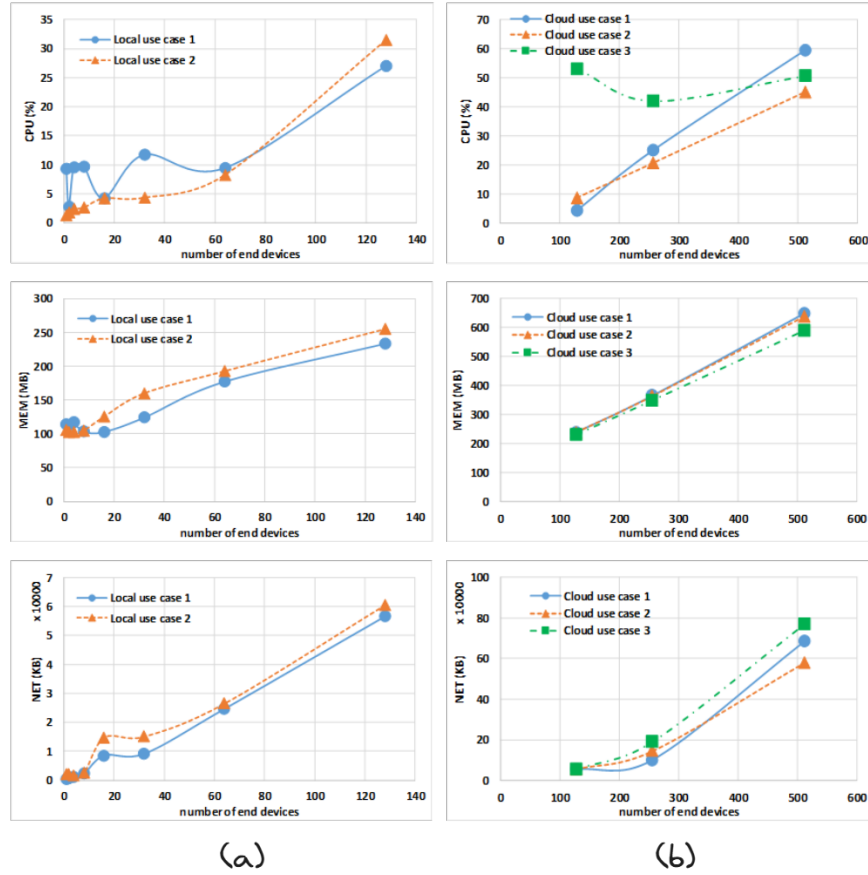


Figure 3.6: TCPU, TMEM, and TNET for (a) Local Use Case, (b) Cloud Use Case

### Cloud Emulation Costs

Table 3.10 reports the measured startup, emulation, and teardown times and the associated cloud costs for each configuration.

Table 3.10: Cloud emulation costs for the three use cases (15 min emulation phase). Costs are contract-specific and given for reference only.

| UC      | $N$ | Startup   | Emu    | Stop     | Cost (USD) |
|---------|-----|-----------|--------|----------|------------|
| Cloud 1 | 128 | 38 s      | 15 min | 19 s     | 0.03       |
|         | 256 | 3 m 14 s  | 15 min | 47 s     | 0.05       |
|         | 512 | 15 m 21 s | 15 min | 2 m 25 s | 0.06       |
| Cloud 2 | 128 | 38 s      | 15 min | 19 s     | 0.04       |
|         | 256 | 3 m 15 s  | 15 min | 41 s     | 0.06       |
|         | 512 | 13 m 1 s  | 15 min | 2 m 7 s  | 0.09       |
| Cloud 3 | 128 | 54 s      | 15 min | 20 s     | 0.03       |
|         | 256 | 2 m 2 s   | 15 min | 49 s     | 0.03       |
|         | 512 | 8 m 47 s  | 15 min | 2 m 6 s  | 0.04       |

The emulation cost scales linearly with the experiment duration and is remarkably low. Emulating 512 nodes for one hour costs between USD 0.12 and USD 0.21 depending on the cloud configuration; a full 24-hour emulation remains below USD 4.10 even in the most expensive configuration. The startup phase grows sub-linearly with  $N$  and represents a one-time preparation cost that can be amortised over long experiments.

## 3.3 A Software-Defined LoRaWAN Node Based on SDR

The Hybrid architecture described in Section 3.1 separates the application and LoRaWAN protocol-stack layers into individual containers while keeping the phys-

ical radio on a dedicated hardware device connected via a serial link. A natural and more radical evolution of this concept is to extend virtualization down to the physical layer itself, replacing the hardwired LoRa transceiver with a SDR. This yields a *fully software-defined* LoRaWAN node in which every functional block, from application logic to I/Q sample generation and transmission, is implemented in software and isolated in a dedicated Linux Container (LXC).

### 3.3.1 Architecture of the Software-Defined Node

Following the decomposition of a Class A LoRaWAN uplink transaction into its constituent processing steps, four distinct functional blocks have been identified and mapped to individual Docker containers:

1. **User application (LXC1).** A Node-RED application, written in JavaScript, that periodically triggers the transmission of a message of preset length.
2. **LoRaWAN layer (LXC2).** A Python script that handles LoRaWAN payload formatting, including frame ciphering, header insertion, and session-key management.
3. **LoRa baseband (LXC3).** A MATLAB-derived standalone application that performs LoRa encoding and baseband modulation, producing the in-phase (I) and quadrature (Q) discrete-time sample stream.
4. **SDR serialiser (LXC4).** A C programme that streams the I/Q samples to the SDR device over a USB 2.0 interface, exploiting the Linux Industrial Input/Output (IIO) driver.

Containers LXC1–LXC3 exchange data through an MQTT-based virtual databus (including a dedicated Broker container); the LXC3–LXC4 link uses a TCP/IP

client–server scheme owing to the high-throughput requirement of the raw sample stream. The physical radio is an ADALM-PLUTO SDR from Analog Devices, a low-cost device operating up to 6 GHz with a 20 MHz instantaneous bandwidth and a USB 2.0 host interface. The architecture is depicted in Figure 3.7.

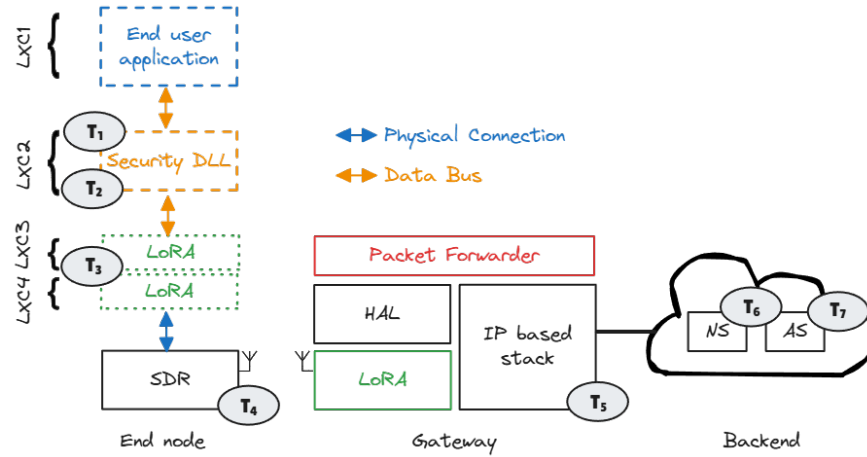


Figure 3.7: Architecture of the software-defined LoRaWAN node. LXC1–LXC4 are Docker containers; inter-container communication uses MQTT except for the LXC3–LXC4 link, which uses TCP/IP. The SDR is connected to the host via USB.

### 3.3.2 Relation to the Hybrid Architecture

Compared with the Hybrid architecture of Section 3.1.1, the software-defined node introduces two additional decomposition levels. First, the LoRa Physical Layer (PHY), which in the Hybrid case is still executed by the firmware of the RN2483 hardware modem, is now fully virtualized in LXC3. Second, the hardware-abstraction boundary is pushed down to the SDR driver level: LXC4 communicates with the ADALM-PLUTO exclusively through the standard Linux

IIO interface, making the solution independent of any specific radio hardware that exposes the same driver API. As a consequence, the serial link between host and modem is replaced by a USB stream of raw I/Q samples, whose throughput becomes the dominant performance constraint.

### 3.3.3 Experimental Setup

The host machine runs Ubuntu 20.04.6 LTS on an Intel i5-1135G7 CPU (2.40 GHz, 16 GB RAM, 256 GB SSD), with Docker 26.0.1 and Compose 1.28.5. The LoRaWAN gateway is the same RG186 from Laird used in the previous experiments, running the Semtech UDP packet-forwarder protocol, and the backend is *The Things Stack*. The SDR sample rate is fixed at  $F_S = 2 \text{ MSa/s}$ , sufficient for the bandwidth of LoRa signals at all spreading factors. Experiments lasting three hours are conducted for spreading factors SF7–SF10 and two user payload sizes (25 B and 50 B), with one uplink frame transmitted every 60 s.

#### Container Resource Usage

Table 3.11 reports the average Docker runtime statistics collected during the experiment (SF10, 50 B payload). The most resource-intensive container is LXC3, owing to the computational cost of the MATLAB-derived LoRa baseband model; porting the model to a native language such as C would substantially reduce both CPU and memory footprint. The output network traffic of LXC4 (approximately 41 MB per message) reflects the use of a fixed-length 10 MSa buffer: with a sample rate of 2 MSa/s and 2 bytes per I/Q pair, each message occupies about 5.3 MB of actual signal data, padded to 40 MB by the fixed buffer.

Table 3.11: Average Docker container statistics (SF10, 50 B payload, 3 h experiment).

|        | CPU<br>(%) | MEM<br>(MiB) | PIDs | Block I<br>(MB) | Block O<br>(kB) | $\Delta$ Net I<br>(MB) | $\Delta$ Net O<br>(MB) |
|--------|------------|--------------|------|-----------------|-----------------|------------------------|------------------------|
| LXC1   | <0.01      | 81.45        | 21   | 0.67            | 32.80           | 0.00                   | —                      |
| LXC2   | <0.01      | 2.88         | 3    | 2.24            | 4.10            | <0.01                  | <0.01                  |
| LXC3   | 0.16       | 589.44       | 37   | 10.93           | 12.30           | 0.01                   | 5.70                   |
| LXC4   | <0.01      | 167.74       | 5    | 2.12            | <0.01           | 6.54                   | 41.11                  |
| Logger | 0.03       | 10.37        | 3    | 0.12            | <0.01           | <0.01                  | <0.01                  |
| Broker | 0.04       | 2.76         | 1    | 2.37            | 45.10           | <0.01                  | <0.01                  |

### Time-Based Metrics

Seven timestamps are collected along the uplink path to derive the following latency metrics (see Figure 3.7):

- $\Delta_1 = T_2 - T_1$ : processing time of the LoRaWAN layer (LXC2).
- $\Delta_2 = T_3 - T_2$ : processing time of the LoRa baseband (LXC3).
- $\Delta_3 = T_4 - T_3$ : time to set up the connection with the SDR (LXC4).
- $\Delta_4 = T_5 - T_4$ : I/Q streaming time plus over-the-air propagation to the gateway.
- $\Delta_5 = T_6 - T_5$ : additional latency to reach the Network Server.
- $\Delta_6 = T_7 - T_6$ : additional latency to reach the Application Server.
- $\Delta_7 = T_7 - T_1$ : overall end-to-end uplink trip time.

The complete statistics for all SF/payload combinations are reported in Table 3.12. The dominant contribution to the overall trip time  $\Delta_7$  is  $\Delta_4$ , which

accounts for the I/Q sample transfer over the USB interface and the over-the-air propagation. The 40 MB fixed buffer transferred at the USB 2.0 throughput of the IIO driver (approximately 40 MB/s) requires around 1 s regardless of the actual payload size, establishing a practical floor on  $\Delta_4$ .

Despite this overhead, the minimum values of  $\Delta_4$  and  $\Delta_7$  across configurations differ by amounts consistent with the difference in airtime duration between spreading factors: incrementing SF by one approximately doubles the frame duration, and the corresponding shift in  $\Delta_4$  follows this expectation closely. Residual variability is attributable to jitter in the  $T_5$  timestamp and NTP synchronisation uncertainty (approximately 0.5 ms for same-clock differences).

The backend latencies  $\Delta_5$  and  $\Delta_6$  are stable across all configurations, as expected:  $\Delta_6$  is consistently around 207 ms, confirming that the backend behaviour is independent of the node implementation.

Table 3.12: Latency statistics for all SF/payload combinations (all values in ms).

| Config                            | Stat     | $\Delta_1$ | $\Delta_2$ | $\Delta_3$ | $\Delta_4$ | $\Delta_5$ | $\Delta_6$ | $\Delta_7$ |
|-----------------------------------|----------|------------|------------|------------|------------|------------|------------|------------|
| SF7<br>25 B<br>$T_{OA}=82.2$ ms   | Mean     | 0.16       | 45.06      | 4.45       | 1143.18    | 18.46      | 207.31     | 1418.62    |
|                                   | Std      | 0.41       | 15.87      | 2.38       | 18.49      | 2.01       | 2.07       | 26.20      |
|                                   | Median   | 0.13       | 43.68      | 4.10       | 1146.56    | 18.63      | 206.96     | 1418.69    |
|                                   | 5th Pct  | 0.06       | 33.16      | 2.80       | 1095.64    | 18.43      | 205.34     | 1390.45    |
|                                   | 95th Pct | 0.19       | 56.02      | 7.08       | 1160.04    | 19.57      | 210.03     | 1443.06    |
| SF7<br>50 B<br>$T_{OA}=118.0$ ms  | Mean     | 0.17       | 52.91      | 5.53       | 1179.89    | 18.68      | 207.33     | 1464.51    |
|                                   | Std      | 0.07       | 17.10      | 2.53       | 21.51      | 0.95       | 2.97       | 29.45      |
|                                   | Median   | 0.16       | 49.66      | 5.02       | 1184.84    | 18.61      | 206.87     | 1465.40    |
|                                   | 5th Pct  | 0.08       | 39.12      | 3.62       | 1114.24    | 18.43      | 205.37     | 1401.76    |
|                                   | 95th Pct | 0.23       | 81.48      | 9.01       | 1200.17    | 19.64      | 210.35     | 1498.18    |
| SF8<br>25 B<br>$T_{OA}=143.9$ ms  | Mean     | 0.13       | 43.23      | 5.84       | 1213.61    | 5.60       | 207.37     | 1475.78    |
|                                   | Std      | 0.05       | 30.20      | 3.50       | 22.02      | 4.37       | 1.45       | 35.44      |
|                                   | Median   | 0.13       | 35.78      | 4.95       | 1219.17    | 4.59       | 207.06     | 1475.64    |
|                                   | 5th Pct  | 0.07       | 32.93      | 4.24       | 1151.79    | 1.23       | 205.65     | 1409.98    |
|                                   | 95th Pct | 0.18       | 56.76      | 10.77      | 1232.11    | 20.00      | 209.70     | 1505.20    |
| SF8<br>50 B<br>$T_{OA}=215.6$ ms  | Mean     | 0.15       | 48.18      | 7.83       | 1278.32    | 13.65      | 207.51     | 1555.63    |
|                                   | Std      | 0.05       | 16.31      | 2.05       | 22.30      | 8.48       | 2.22       | 27.83      |
|                                   | Median   | 0.15       | 46.63      | 7.53       | 1283.23    | 18.82      | 207.06     | 1557.94    |
|                                   | 5th Pct  | 0.07       | 38.67      | 6.15       | 1216.63    | 2.03       | 205.51     | 1491.17    |
|                                   | 95th Pct | 0.19       | 58.51      | 10.04      | 1298.63    | 20.54      | 209.51     | 1575.47    |
| SF9<br>25 B<br>$T_{OA}=267.3$ ms  | Mean     | 0.13       | 47.77      | 9.46       | 1334.59    | 9.20       | 207.58     | 1608.74    |
|                                   | Std      | 0.04       | 15.77      | 2.57       | 23.19      | 7.26       | 2.08       | 28.16      |
|                                   | Median   | 0.13       | 43.71      | 8.92       | 1339.86    | 4.98       | 207.28     | 1611.10    |
|                                   | 5th Pct  | 0.07       | 38.61      | 7.65       | 1270.56    | 1.25       | 205.44     | 1546.20    |
|                                   | 95th Pct | 0.16       | 61.21      | 11.94      | 1355.31    | 19.10      | 210.03     | 1632.20    |
| SF9<br>50 B<br>$T_{OA}=390.1$ ms  | Mean     | 0.16       | 66.62      | 16.05      | 1465.49    | 6.68       | 207.30     | 1762.29    |
|                                   | Std      | 0.04       | 19.67      | 3.81       | 20.45      | 6.28       | 1.40       | 29.86      |
|                                   | Median   | 0.16       | 64.67      | 14.97      | 1470.34    | 4.57       | 207.10     | 1762.69    |
|                                   | 5th Pct  | 0.07       | 47.82      | 12.40      | 1404.46    | 0.48       | 205.58     | 1718.48    |
|                                   | 95th Pct | 0.23       | 90.23      | 23.04      | 1484.27    | 18.88      | 209.37     | 1796.79    |
| SF10<br>25 B<br>$T_{OA}=493.6$ ms | Mean     | 0.12       | 58.68      | 18.39      | 1568.45    | 9.80       | 207.49     | 1862.93    |
|                                   | Std      | 0.02       | 13.19      | 3.31       | 18.85      | 7.52       | 1.76       | 23.21      |
|                                   | Median   | 0.12       | 57.15      | 17.48      | 1571.97    | 5.14       | 207.12     | 1864.00    |
|                                   | 5th Pct  | 0.06       | 49.42      | 16.16      | 1533.59    | 1.21       | 205.51     | 1837.78    |
|                                   | 95th Pct | 0.16       | 69.47      | 23.04      | 1590.10    | 19.62      | 210.78     | 1884.47    |
| SF10<br>50 B<br>$T_{OA}=698.4$ ms | Mean     | 0.15       | 78.82      | 28.71      | 1775.86    | 6.58       | 207.43     | 2097.54    |
|                                   | Std      | 0.03       | 21.20      | 5.91       | 18.63      | 5.83       | 2.42       | 29.35      |
|                                   | Median   | 0.15       | 74.83      | 26.67      | 1780.18    | 4.62       | 207.11     | 2097.76    |
|                                   | 5th Pct  | 0.08       | 64.75      | 25.07      | 1729.21    | 0.99       | 205.48     | 2064.35    |
|                                   | 95th Pct | 0.19       | 106.18     | 39.22      | 1793.42    | 17.64      | 209.84     | 2136.01    |

## **3.4 5G Standalone Private Network for Smart Campus Multi-Access Communication**

While LoRaWAN addresses the low-rate, low-power segment of the smart-campus communication infrastructure, periodic telemetry from e-bikes, environmental sensors, and parking monitors, a complementary high-bandwidth, low-latency link is required for real-time applications such as video analytics, vehicle localisation, and time-sensitive control loops. The University of Brescia is developing a multi-access architecture that combines the existing private LoRaWAN network with a lab-grade private 5G Standalone (SA) infrastructure, both deployed within the MOST PNRR project framework. This section documents the performance characterisation of the 5G side of that architecture performed in [39].

### **3.4.1 Experimental Setup**

The 5G infrastructure is based on the Open Air Interface (OAI) open-source framework, customised and integrated by Firecell through the Labkit 40 product. The base station (gNB) functions are realised by a USRP B200 SDR from National Instruments, operating in the n78 band (FR1, below 6 GHz) with 30 kHz sub-carrier spacing and 40 MHz channel bandwidth (SISO), yielding up to 106 Physical Resource Blocks. Both the 5G Core Network (5GCN) and the Radio Access Network (RAN) components are Docker-containerized and executed on an Intel i9-12900 server (32 GB RAM, Ubuntu 20.04 LTS with low-latency kernel). A preliminary bandwidth stress test demonstrated a maximum uplink throughput of approximately 15 Mbps and a maximum downlink throughput of approximately 160 Mbps.

User Equipment (UE) connectivity is provided by Quectel RM500Q 5G mod-

ules, each equipped with a preconfigured SIM. The reference UE is a PC (Intel Pentium G2130, 8 GB RAM, Windows 7) that exchanges a 10 B TCP/IP payload with a Python server co-hosted on the 5GCN machine at an update rate of approximately 1 message per second. Controlled interfering traffic is generated by a second PC (Intel Core i7-10710U, Windows 11) connected to three additional RM500Q modules, each running an independent iperf3 client that injects 1 Mbps of uplink traffic towards a co-located iperf3 server.

Since the deployment of on-premises private 5G networks is not yet licensed in Italy, the testbed operates in a confined, close-proximity environment; coverage characterisation is therefore out of scope, and the focus is exclusively on time-related performance.

### **Time-Synchronisation and Metrics**

Both client and server applications record four timestamps per transaction (Figure 3.8):

- $T_1$ : client timestamp before sending the request;
- $T_2$ : server timestamp upon receiving the request;
- $T_3$ : server timestamp before sending the response;
- $T_4$ : client timestamp upon receiving the response.

A common time reference is established using the Italian metrological NTP server `ntp1.inrim.it`, queried through the Python `ntplib` library, providing an accuracy of a few milliseconds, sufficient for the latency ranges of interest. The

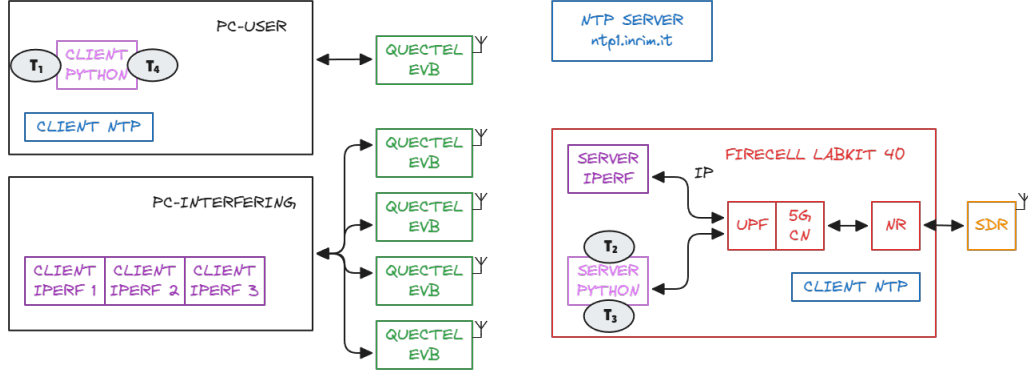


Figure 3.8: Experimental setup for the 5G network scenario. The four timestamps  $T_1$ – $T_4$  collected by the client and server applications are indicated.

metrics are:

$$OTT_{\text{FWD}} = T_2 - T_1 \quad (\text{client-to-server OTT}) \quad (3.4)$$

$$OTT_{\text{BWD}} = T_4 - T_3 \quad (\text{server-to-client OTT}) \quad (3.5)$$

$$RTT = T_4 - T_1 \quad (\text{round-trip time, incl. server processing}) \quad (3.6)$$

### 3.4.2 Results

Approximately 4000 timestamp sets are collected per scenario (each experiment lasting more than one hour). Four configurations are tested, with zero to three active interfering UEs. Table 3.13 summarizes the statistics of the RTT across all conditions and Figure 3.9 shows the distribution of the OTT and RTT metrics.

The average RTT is 40 ms across all test conditions, with 90 % of samples falling in the narrow range  $[39, 40]$  ms, indicating very limited spread. The OTT exhibits a notable asymmetry:  $OTT_{\text{FWD}} \approx 10$  ms (client to server) versus  $OTT_{\text{BWD}} \approx 14$  ms (server to client). This asymmetry is consistent with findings reported in the literature for equivalent OAI-based setups and is attributed to dif-

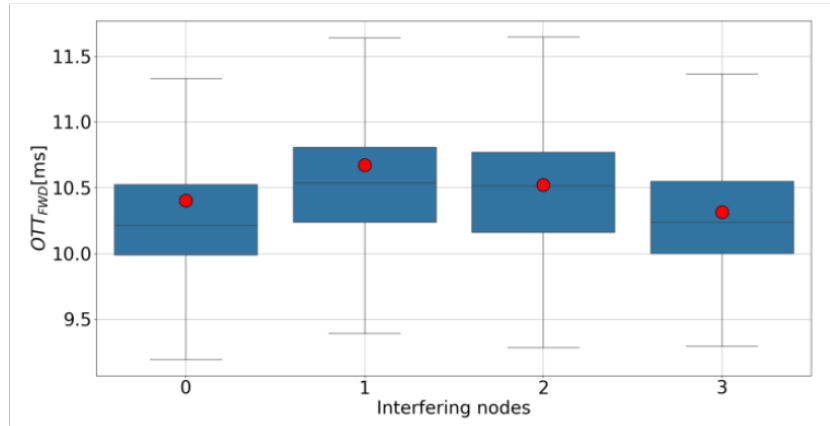
Table 3.13: Statistics of the *RTT* metric under variable interfering load.

| Interfering nodes | $\mu(\text{ms})$ | $\sigma(\text{ms})$ | $P_{5\%}(\text{ms})$ | $P_{95\%}(\text{ms})$ |
|-------------------|------------------|---------------------|----------------------|-----------------------|
| 0                 | 39.91            | 5.12                | 39.18                | 39.98                 |
| 1                 | 40.07            | 3.24                | 39.33                | 40.34                 |
| 2                 | 39.85            | 2.76                | 39.15                | 40.26                 |
| 3                 | 39.58            | 3.20                | 39.05                | 40.00                 |

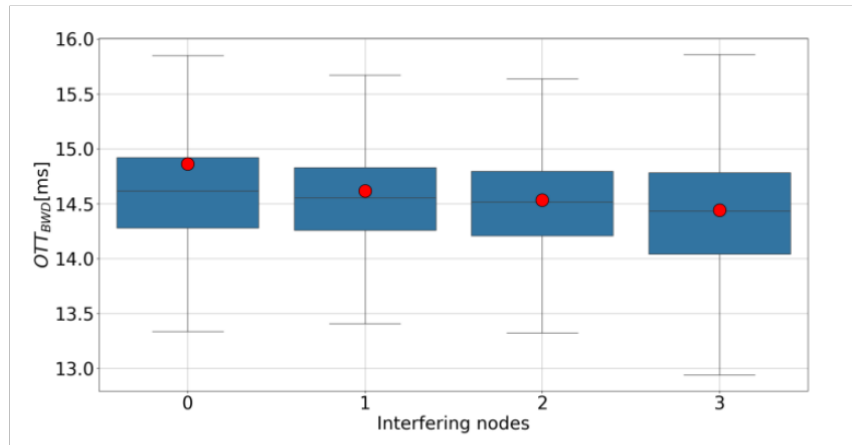
ferences in uplink and downlink scheduling and queuing, rather than to payload size or application-level processing.

Importantly, the introduction of up to three interfering UEs each injecting 1 Mbps of continuous traffic causes no statistically significant variation in the average RTT or OTT. The network maintains stable latency performance even under increased load, confirming the suitability of the private 5G SA infrastructure for the latency requirements of smart-campus applications.

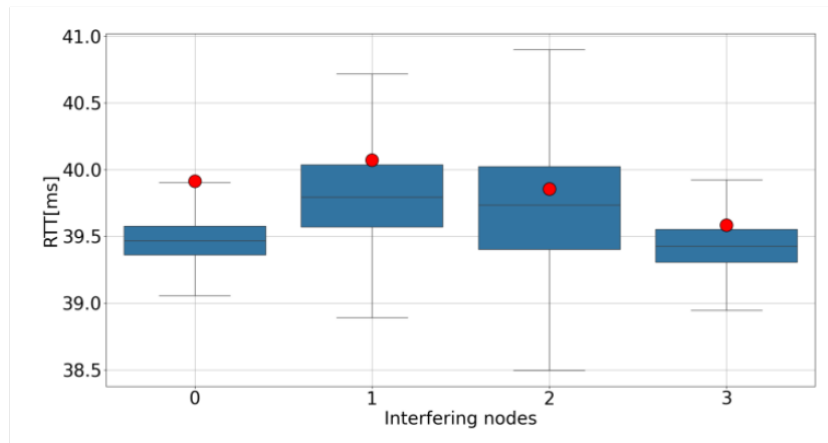
The computational load of the User Plane Function container (SPGWU) is also monitored via `docker stats` during the stress-test phases. Even at peak aggregate loads exceeding 100 Mbps in the downlink direction, the container CPU usage remained within the capacity of the host server, validating the adequacy of the hardware configuration for the envisioned deployment.



(a)



(b)



(c)

Figure 3.9: Boxplot showing the distribution of (a)  $OTT_{FWD}$ , (b)  $OTT_{BWD}$ , and (c)  $RTT$  under variable interfering load.

## 3.5 Summary

The contributions presented in this chapter collectively define a layered communication infrastructure for sustainable-mobility and smart-campus scenarios, directly addressing (PS 1) by demonstrating that the software disaggregation and containerization principles developed in industrial automation can be successfully applied to the communication layer of LEV platforms. On the LoRaWAN side, three contributions form a progressive decomposition of the end-device stack: the first establishes that virtualization overhead is negligible and that the dominant performance factor is the hardware interface retained at the system boundary; the second pushes the decomposition down to the physical layer via an SDR, yielding a fully software-defined and reconfigurable LPWAN node; the third leverages the fully virtualized architecture to emulate complete LoRaWAN networks at smart-city scale on commodity hardware. On the 5G side, a private Standalone network characterised under realistic load conditions achieves an average RTT of 40 ms, independent of co-located interfering traffic, establishing 5G as the natural complement to LoRaWAN for latency-critical services. Together, these four contributions define the communication envelope within which the monitoring and mobility platforms described in subsequent chapters operate.

## Chapter 4

# The Core Monitoring Platforms

With the communication layer established, this chapter turns to the on-board intelligence of the LEV platform. It presents the Black Box, a modular and containerised edge node that embodies the disaggregation principles introduced in response to (PS 1). The chapter then addresses (PS 2) directly, proposing two complementary on-device data management strategies: an Autoencoder-based compression model for multivariate sensor streams and a family of incremental anomaly detection algorithms designed to operate within the strict memory and computational constraints of TinyML-class hardware. Within the end-to-end pipeline of this thesis, this chapter occupies the central stage: it builds directly on the communication envelope characterised in Chapter 3, moving from the link layer to the edge node that acquires and processes data over it, and in turn feeds the health-monitoring application of Chapter 5.

## 4.1 Design of the Black Box Prototype

The Black Box is a standalone, battery-powered data-logging and monitoring device designed to be mounted on light vehicles, primarily bicycles. Two hardware prototypes are developed and tested: a *Raspberry Pi-based* implementation and an *Arduino-based* one. Both are evaluated in terms of sensor integration, data acquisition performance, power consumption, and suitability for real-world deployment on light electric vehicles.

While both solutions successfully demonstrate capture, storage, and transmission of critical mobility parameters, this section focuses on the Raspberry Pi-based implementation (architecture shown in Figure 4.1), which is selected as the primary development platform due to its superior computational capabilities, flexible software ecosystem, and ease of sensor integration.

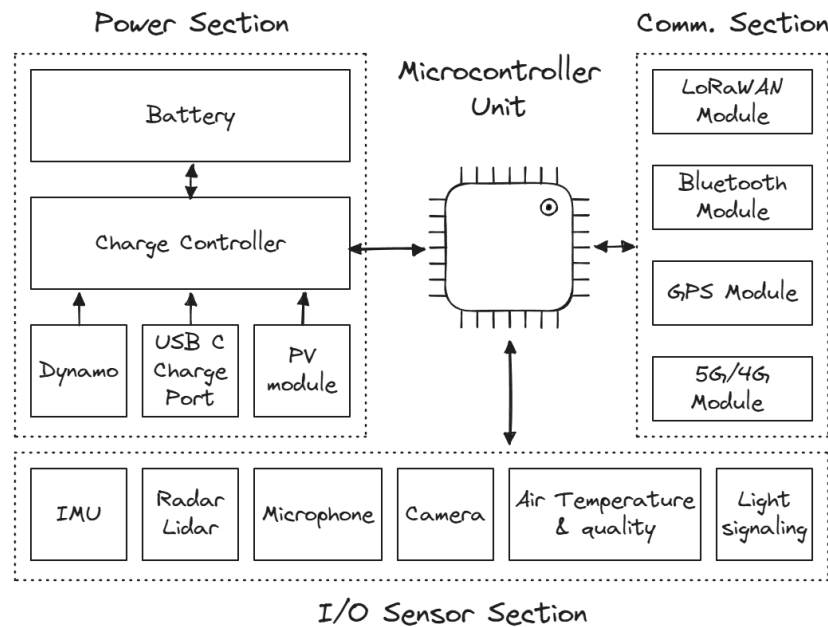


Figure 4.1: Proposed architecture of the Raspberry Pi-based Black Box prototype.

### 4.1.1 Processing Unit

The processing unit represents the computational core of the Black Box. Given the constraints of battery-powered, on-bike deployment, the board selection requires a careful trade-off between processing power, energy consumption, physical footprint, and peripheral support.

The evaluation is initially restricted to Raspberry Pi boards, owing to their ease of programming, extensive library support, compact form factor, and affordability. Among the models examined (including the Raspberry Pi 5, 4 Model B, 3 Model B+, Zero W, and Zero 2W) the two most compact variants prove best suited to the application.

A detailed comparison between the **Raspberry Pi Zero W** and the **Raspberry Pi Zero 2W** is carried out, focusing on idle and peak power consumption, as summarised in Table 4.1.

Table 4.1: Power consumption comparison: Raspberry Pi Zero W vs. Zero 2W (5V supply).

| Board       | Idle [mA] | Idle [W] | Heavy load [mA] | Heavy load [W] |
|-------------|-----------|----------|-----------------|----------------|
| RPi Zero W  | 260       | 1.30     | 370             | 1.85           |
| RPi Zero 2W | 280       | 1.40     | 580             | 2.90           |

Despite its higher power draw under load, the **Raspberry Pi Zero 2W** is selected as the processing unit. Its quad-core ARM Cortex-A53 processor at 1 GHz offers significantly better performance for multi-threaded workloads (essential for simultaneously managing GPS parsing, IMU sampling, LiDAR reading, audio recording, and LoRaWAN communication). With a 2500 mAh lithium-ion battery, the estimated idle autonomy is approximately 8.9 hours, which is com-

patible with the expected duration of a typical cycling session.

#### 4.1.2 Power Section

The Black Box is designed to operate autonomously from a rechargeable lithium-ion battery pack, composed of the following elements:

- Two **INR 18650-25R** cells (2500 mAh each) connected in parallel, providing a nominal voltage of 3.7 V;
- A **TP4056** charging module with USB-C input for recharging from standard power sources;
- A **step-up converter** (MT3608) to regulate the output to 5 V, as required by the Raspberry Pi Zero 2W and the peripheral modules;
- A **power supply switch** for manual device activation.

A photovoltaic panel (KS-M5555P, peak power 0.4 W at 5 V) is also considered as a supplementary energy source. Under standard irradiation conditions (1000 W/m<sup>2</sup>), it supplies approximately 80 mA of charging current, potentially extending operational time during outdoor use.

#### 4.1.3 Communication Modules

Long-range data transmission and precise geolocation are both handled by a single compact module: the **Waveshare SX1262 LoRaWAN/GNSS HAT**. It integrates:

- A **LoRaWAN** radio transceiver (SX1262) for low-power, long-range uplink to a network server;

- An integrated **GNSS** receiver supporting GPS and GLONASS, providing real-time position and speed estimation via UART.

The GNSS module parses NMEA-0183 messages (specifically GGA for timestamp, latitude, longitude, and altitude, and VTG for ground speed) using the `pynmea2` Python library, at a nominal acquisition rate of 1 Hz.

#### 4.1.4 Sensors and Actuators

##### Inertial Measurement Unit (IMU)

The **LSM6DS3** (STMicroelectronics) integrates a 3-axis accelerometer and a 3-axis gyroscope, connected to the Raspberry Pi via the **I2C** bus. Its key features include:

- Accelerometer full-scale range:  $\pm 2\text{ g}$  to  $\pm 16\text{ g}$ ;
- Gyroscope full-scale range:  $\pm 125\text{ dps}$  to  $\pm 2000\text{ dps}$ ;
- Configurable output data rate (ODR) up to 6664 Hz;
- Low-power mode and hardware free-fall interrupt support.

Acceleration and angular velocity on all three axes are logged with timestamps at an effective acquisition frequency of approximately 103 Hz, enabling reliable detection of abnormal motion patterns such as sudden falls.

##### LiDAR Range Sensor

The **TF-Luna** is a compact time-of-flight sensor used to estimate the distance of approaching objects or vehicles from behind. Its main specifications are summarised in Table 4.2.

Table 4.2: TF-Luna LiDAR main specifications.

| Parameter          | Value                                   |
|--------------------|-----------------------------------------|
| Detection range    | 0.2 m – 8 m                             |
| Accuracy           | $\pm 6$ cm (0.2–3 m), $\pm 2\%$ (3–8 m) |
| Sampling frequency | 1–250 Hz (adjustable)                   |
| Communication      | I2C / UART                              |
| Supply voltage     | 3.7–5.2 V                               |
| Average current    | <70 mA                                  |

The sensor is connected via **I2C** in parallel with the IMU, minimising GPIO usage. LiDAR and IMU data are acquired synchronously and stored in the same log file, to facilitate joint analysis for fall and proximity detection.

### Microphones

Two **INMP441** MEMS microphones are included for environmental sound monitoring, interfaced via the **I2S** bus (Data, Bit Clock, and Word Select lines). Audio is captured using the `pyaudio` Python library and stored in `.wav` format, with each recording timestamped and capped at 4-minute segments to manage file size.

#### 4.1.5 Prototype Implementation

Two successive hardware iterations are realised on the Raspberry Pi platform.

### **Preliminary Prototype**

The first prototype is assembled on a breadboard to validate the proposed architecture. It integrates the Raspberry Pi Zero 2W, the LoRaWAN/GNSS HAT, the TF-Luna LiDAR (I2C), the LSM6DS3 IMU (I2C), and the INMP441 microphone pair (I2S). This configuration confirms the correct operation of all communication interfaces and the coexistence of multiple peripherals on shared buses.

### **Final Prototype**

The final version is assembled on a dedicated prototype board, consolidating all components in a compact, field-deployable form:

1. Raspberry Pi Zero 2W (processing unit);
2. LSM6DS3 IMU module;
3. Two INR 18650-25R lithium-ion batteries;
4. TP4056 USB-C charging module;
5. Power supply switch;
6. MT3608 step-up converter;
7. TF-Luna LiDAR Range Sensor;
8. INMP441 stereo microphone pair;
9. External GPS antenna;
10. Waveshare SX1262 LoRaWAN/GNSS HAT.

The assembled device is housed inside a standard bicycle saddle bag (Figure 4.2), demonstrating its suitability for practical deployment without any structural modification to the bicycle.



Figure 4.2: Raspberry Pi-based Black Box inside the bike saddle bag.

### 4.1.6 Preliminary Experimental Results

#### Field Data Logging Test

A field test is performed by mounting the prototype inside a vehicle travelling approximately 25 km along the SP11 and SP510 provincial roads. GNSS and IMU data are logged in .csv format over a 20-minute session: 1200 GPS samples are collected, of which 96.17% are valid (46 invalid frames). The average ground speed computed from VTG messages is 76.20 km/h, consistent with the road conditions during the test. Over the same session, 123 797 IMU records are stored, confirming a stable effective sampling frequency of approximately 103.19 Hz.

### Power Consumption and Battery Life

The prototype is designed to log sensor data automatically for the entire duration of its power supply. A bash script (invoked at each board reboot via `crontab`) launches the Python data-logging routine and creates three timestamped output files, ensuring uniqueness and easy identification.

Indoor battery life tests are conducted under two conditions:

- **Without GPS signal:** 6 hours and 15 minutes;
- **With GPS signal available:** 8 hours and 6 minutes.

The longer duration in the second case reflects the reduced computational overhead once the GPS receiver locks onto satellites, compared to the continuous polling attempts in the signal-absent scenario.

Extended measurements performed with a Keithley DMM6500 digital multi-meter over three full discharge cycles are reported in Table 4.3 and illustrated in Figures 4.3 and 4.4.

Table 4.3: Raspberry Pi-based Black Box: power consumption over three discharge cycles.

| Cycle | Max $I$ [A] | Mean $I$ [A] | Max $P$ [W] | Mean $P$ [W] | Energy [Wh] |
|-------|-------------|--------------|-------------|--------------|-------------|
| 1     | 1.78        | 0.583        | 6.70        | 2.034        | 17.69       |
| 2     | 1.72        | 0.592        | 5.95        | 2.063        | 17.70       |
| 3     | 1.92        | 0.586        | 7.77        | 2.071        | 16.97       |

Average power consumption across the three cycles is approximately 2.06 W, with peak values reaching 7.77 W during transient operations. These results confirm the feasibility of the prototype for real-world cycling applications, where sessions typically last 2 to 6 hours.

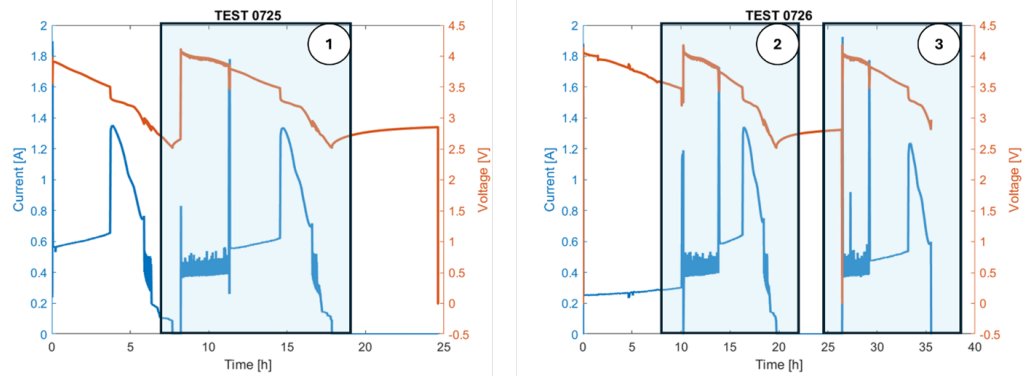


Figure 4.3: Current and voltage measurements over three battery discharge cycles.

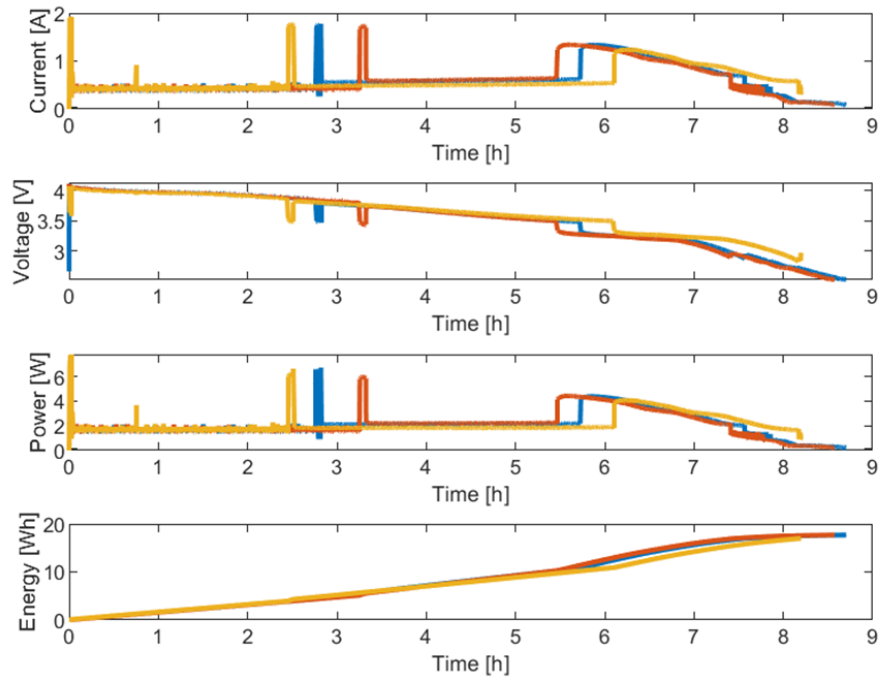


Figure 4.4: Current, voltage, power, and energy consumption of the prototype.

## 4.2 Autoencoder-Based Compression of Vehicular Sensor Data

The proliferation of IoT devices in vehicular and smart-mobility contexts generates continuous, high-frequency sensor streams that must be transmitted, stored, and processed under strict resource constraints. Edge computing mitigates the communication bottleneck by moving part of the processing logic to the device itself, but the severe limitations of embedded microcontrollers in terms of memory, compute, and energy require compression algorithms that are both lightweight and faithful to the original signal [44, 45]. The proposed work [46] addresses this challenge by proposing an autoencoder-based approach for sensor data compression that is fully deployable on low-cost IoT hardware, validated on a real-world vehicular case study.

### 4.2.1 Autoencoder Variants and Architectures

An autoencoder is a neural network trained to reproduce its input at the output through a bottleneck layer of reduced dimensionality, forcing the model to learn a compact latent representation of the data [47]. Three variants are considered in this work. The *Feedforward Autoencoder* minimises the mean squared error (MSE) between input and reconstruction without additional constraints. The *Sparse Autoencoder* [48] augments the loss with a Kullback–Leibler divergence penalty that drives the average activation of each latent neuron towards a target sparsity rate  $\rho$ :

$$\mathcal{L}_{\text{Sparse}} = \mathcal{L}_{\text{MSE}} + \beta \sum_{j=1}^n \left[ \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j} \right] \quad (4.1)$$

where  $\hat{\rho}_j$  is the empirical mean activation of neuron  $j$  and  $\beta$  controls the strength of the penalty. The *Contractive Autoencoder* [49] instead penalises the Frobenius norm of the Jacobian of the latent activations with respect to the input, encouraging representations that are insensitive to small input perturbations:

$$\mathcal{L}_{\text{Contractive}} = \mathcal{L}_{\text{MSE}} + \lambda \left\| \frac{\partial h}{\partial x} \right\|_F^2 \quad (4.2)$$

where  $h$  denotes the bottleneck activations and  $\lambda$  is the regularisation coefficient.

Each variant is combined with two structural families. In *symmetric* architectures the decoder mirrors the encoder layer by layer; in *asymmetric* architectures the decoder is deliberately deeper and wider than the encoder, allowing a richer reconstruction from a very compact latent code. Six concrete configurations are evaluated (Table 4.4), crossing two symmetric and four asymmetric designs with fifteen bottleneck sizes  $N \in [1, 15]$  and three autoencoder types, for a total of 270 trained models.

Table 4.4: Evaluated autoencoder architectures.  $N$  denotes the bottleneck size.

| Architecture  | Encoder          | Decoder                      |
|---------------|------------------|------------------------------|
| Symmetric C1  | [128, 64, $N$ ]  | [ $N$ , 64, 128]             |
| Symmetric C2  | [256, 128, $N$ ] | [ $N$ , 128, 256]            |
| Asymmetric C1 | [128, 64, $N$ ]  | [ $N$ , 64, 128, 256]        |
| Asymmetric C2 | [128, 64, $N$ ]  | [ $N$ , 64, 128, 256, 512]   |
| Asymmetric C3 | [256, 128, $N$ ] | [ $N$ , 128, 256, 512]       |
| Asymmetric C4 | [256, 128, $N$ ] | [ $N$ , 128, 256, 512, 1024] |

The proposed approaches focus on lightweight compression and encryption of vehicular sensor data, emphasizing its feasibility for deployment on low-cost platforms, such as the Freematics One+. As shown in Figure 4.5, the proposal

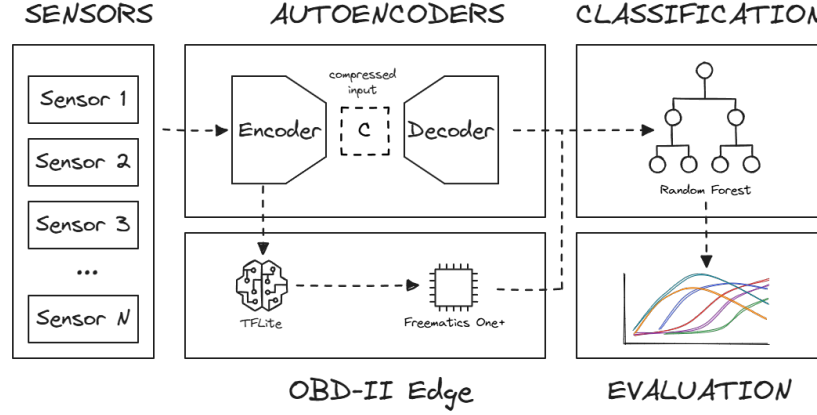


Figure 4.5: Autoencoder proposed architectures.

involves developing, training, and evaluating Autoencoder models specified in Table 4.4.

### 4.2.2 Dataset and Preprocessing

The training data are collected from eight OBD-II datasets spanning five vehicle models (Nissan Kicks, Honda Fit, Hyundai Creta, Hyundai HB20, Fiat Mobi) using the Torque Pro application over a Bluetooth OBD-II interface [50]. Five engine parameters are selected for their availability across all vehicles and their relevance to combustion dynamics: Engine Load, Engine RPM, Vehicle Speed, Throttle Position, and Timing Advance. To capture short-term temporal dependencies without the overhead of recurrent layers, a sliding-window expansion is applied: each sample is composed of three consecutive readings at times  $t - 2$ ,  $t - 1$ , and  $t$ , yielding a 15-dimensional input vector ( $5 \text{ sensors} \times 3 \text{ time steps}$ ). All variables are normalised to  $[0, 1]$  using MinMaxScaler parameters computed on the training split. The final dataset comprises 164 599 training samples, 20 575 validation samples, and 20 575 test samples.

### 4.2.3 Model Selection and Embedded Deployment

After training, models whose validation MSE exceeded the upper outlier fence ( $Q_3 + 1.5 \cdot \text{IQR}$ ) are discarded. The remaining models are ranked by a composite score that linearly combines the normalised bottleneck size and the normalised MSE:

$$\text{Score} = \alpha \cdot \frac{N}{N_{\max}} + \beta \cdot \frac{\text{MSE}}{\text{MSE}_{\max}} \quad (4.3)$$

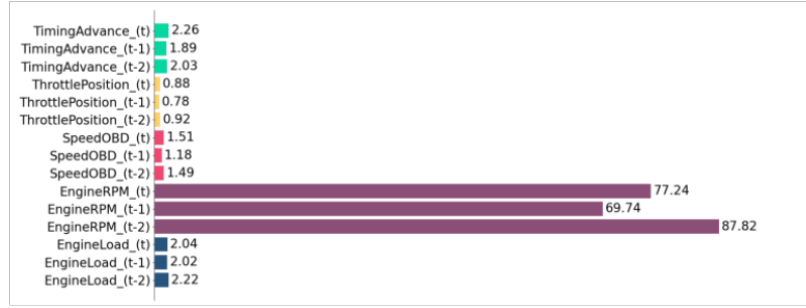
Three weight configurations  $(\alpha, \beta) \in \{(0.2, 0.8), (0.3, 0.7), (0.4, 0.6)\}$  are tested and the intersection of the top-10 lists is retained, yielding three models with consistent performance across all balance criteria. These three models are converted to TensorFlow Lite format with fixed-point quantisation and deployed on the OBD-II Edge Freematics One+ device (ESP32 microcontroller, 16 MB flash, native TFLite support).

### 4.2.4 Results

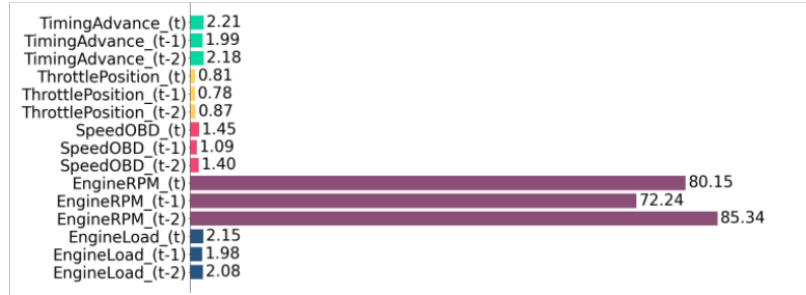
All three selected models share a bottleneck size of  $N = 3$ , corresponding to an 80 % compression ratio (15 input dimensions reduced to 3). The selected architectures are `feedforward_symmetric_c2`, `feedforward_asymmetric_c3`, and `feedforward_asymmetric_c4`.

Reconstruction errors are lowest for Throttle Position and Vehicle Speed, whose signals exhibit limited variability, while Engine RPM produces the highest MAE values (up to approximately 87 normalised units for Symmetric C2) due to the high amplitude and rapid fluctuations of that signal. Asymmetric C4 achieves the lowest overall MAE, followed closely by Asymmetric C3, confirming that the deeper decoder compensates for the aggressive bottleneck compression.

Despite the compression, the latent representations retain sufficient discriminative information for downstream tasks. A Random Forest classifier trained on



(a)



(b)



(c)

Figure 4.6: Mean Absolute Error per sensor variable at time instances  $t$ ,  $t-1$ , and  $t-2$  for the three selected autoencoder architectures.

the compressed codes achieves an F1-score between 0.86 and 0.88 on a fuel-type classification task (gasoline vs. ethanol), compared to 0.90 when trained on the full 15-dimensional input (Table 4.5).

Table 4.5: Fuel-type classification performance from compressed representations.

| Model                | Accuracy | Precision | Recall | F1   |
|----------------------|----------|-----------|--------|------|
| Full data (baseline) | 0.90     | 0.91      | 0.89   | 0.90 |
| Symmetric C2         | 0.89     | 0.88      | 0.87   | 0.87 |
| Asymmetric C3        | 0.88     | 0.86      | 0.87   | 0.86 |
| Asymmetric C4        | 0.90     | 0.89      | 0.88   | 0.88 |

On the embedded device, all three models operate well within the 1 Hz sampling constraint. The asymmetric architectures (C3 and C4) are the fastest, averaging approximately 22.8 ms per inference with a standard deviation of only 36  $\mu$ s, indicating highly stable execution. The symmetric C2 model is slower (28.3 ms mean) and more variable (1.2 ms standard deviation). This result is counterintuitive at first glance, since symmetric architectures are structurally simpler, but is explained by the larger hidden-layer widths of C2 (256 neurons in the first encoder layer) relative to the asymmetric configurations evaluated. Overall, the asymmetric designs offer the best compromise between compression fidelity, discriminative preservation, and embedded execution efficiency.

### 4.3 Lightweight Incremental Algorithms for Multivariate Anomaly Detection and Correction on TinyML Devices

A complementary challenge to data compression is the detection and correction of anomalies in the sensor stream before transmission or storage. Sensor faults, electrical noise, and connectivity glitches produce outliers that, if left uncorrected, propagate errors into all downstream analytics [51]. The work [52] extends the univariate TEDA-RLS framework proposed by Andrade et al. [53] to multivariate time series, proposing two algorithms, Multivariate Sequential TEDA (MST) and Multivariate Parallel TEDA (MPT), that perform real-time anomaly detection and correction on TinyML devices with no prior training and negligible memory footprint.

#### 4.3.1 Theoretical Background: TEDA and RLS

The TEDA framework (Typicality and Eccentricity Data Analytics), introduced by Angelov [54], characterizes each incoming sample by two complementary metrics derived from accumulated proximity.

For a univariate stream  $X = [x_1, x_2, \dots, x_k, \dots]$ , the *eccentricity*  $\xi$  of sample  $x_k$  is computed recursively from the running mean  $\mu_k$  and variance  $\sigma_k^2$ :

$$\xi_k(x) = \frac{1}{k} + \frac{(\mu_k - x_k)^T(\mu_k - x_k)}{k \sigma_k^2} \quad (4.4)$$

The eccentricity measures how different a sample  $x_k$  is from the rest of the collected data.

From the eccentricity, the *typicality*  $\tau$ , that quantifying how well the sample fits the accumulated data distribution, is derived as follow:

$$\tau_k(x) = 1/(1 + \xi_k) \quad (4.5)$$

These definitions of eccentricity and typicality resemble fuzzy set membership functions since being values between 0 and 1, summing up to a value larger than 1. We can also introduce normalized eccentricity  $\zeta$  and normalized typicality  $t$  which integrate to 1:

$$\zeta_k(x) = \frac{\xi_k(x)}{2}, \quad t_k(x) = \frac{\tau_k(x)}{k-2} \quad (4.6)$$

A sample  $x_k$  is flagged as an outlier when its normalized eccentricity exceeds the Chebyshev-derived threshold [53]:

$$\zeta_k(x_k) \geq \theta = \frac{m^2 + 1}{2k} \quad (4.7)$$

Where  $m$  represents the threshold of sensitivity to outliers, and  $k$  is the number of samples observed so far. The threshold  $\theta$  decreases with  $k$ , reflecting the increasing confidence in the data distribution as more samples are collected.

TEDA requires no stored data history and updates its sufficient statistics (mean and variance) with  $\mathcal{O}(1)$  arithmetic operations per sample, making it directly compatible with microcontroller constraints.

When an outlier is detected, a Recursive Least Squares (RLS) filter provides the corrected estimate. The RLS weight vector  $w_k$  is updated incrementally using the inverse correlation matrix  $R_k$  and a forgetting factor  $\mu$ :

$$R_k = \frac{1}{\mu} \left( R_{k-1} - \frac{R_{k-1} x_k x_k^T R_{k-1}}{\mu + x_k^T R_{k-1} x_k} \right) \quad (4.8)$$

The RLS filter adapts continuously to the local signal trend and replaces the anomalous value with a prediction based on recent history, restoring data integrity without interrupting the acquisition loop.

### 4.3.2 Proposed Algorithms: MST and MPT

Both algorithms extend TEDA-RLS to multivariate signals but differ in how they handle the joint structure of the observation vector.

**Multivariate Sequential TEDA (MST).** MST applies the univariate TEDA-RLS pipeline independently to each sensor channel  $i$ . For a new sample  $x_k$ , the eccentricity  $e_k[i]$  is evaluated per component; if it exceeds the threshold  $\theta$ , only that component is replaced by the RLS estimate:

$$\hat{x}_k[i] = \begin{cases} \text{RLS}(x_k[i]) & \text{if } e_k[i] > \theta \\ x_k[i] & \text{otherwise} \end{cases} \quad (4.9)$$

The corrected multivariate sample  $\hat{x}_k$  is assembled from the per-channel outputs. This localized strategy preserves the original values of unaffected channels and limits the propagation of correction errors across variables.

**Multivariate Parallel TEDA (MPT).** MPT computes a single joint anomaly score  $E_k$  for the entire observation vector  $x_k$ , based on the sum of squared standardised deviations across all  $d$  channels:

$$E_k = \sum_{i=1}^d \frac{(x_k[i] - \mu_k[i])^2}{\sigma_k^2[i]} \quad (4.10)$$

The decision rule is:

$$\hat{x}_k = \begin{cases} \text{RLS}(x_k) & \text{if } E_k > \theta \\ x_k & \text{otherwise} \end{cases} \quad (4.11)$$

This joint-analysis strategy can capture cross-variable anomaly patterns that channel-wise inspection might miss, but it also risks correcting normal channels when only one variable is anomalous.

### 4.3.3 Experimental Setup and Results

Both algorithms are implemented in Python for offline hyperparameter tuning and subsequently translated to C++ for deployment on the Freematics One+ (ESP32, OBD-II interface). The five OBD-II channels used are the same as in the compression study of Section 4.2: Vehicle Speed, Engine RPM, Engine Load, Throttle Position, and Timing Advance, sampled at 1 Hz. Real-world tests are conducted on two vehicles (Fiat Fastback 2025 and Volkswagen Polo 2019) over 6 km urban/highway routes of approximately 10 minutes each.

Detection performance is evaluated offline on synthetic data with injected  $+3\sigma$  spike anomalies (Table 4.6). MST achieves near-perfect precision (98.96 %) and 100 % recall, yielding an F1-score of 99.84 %. MPT shows high recall (91.65 %) but very low precision (20.63 %), reflecting a strong tendency to flag entire samples as anomalous whenever any single channel deviates.

Table 4.6: Anomaly detection performance in Python simulation.

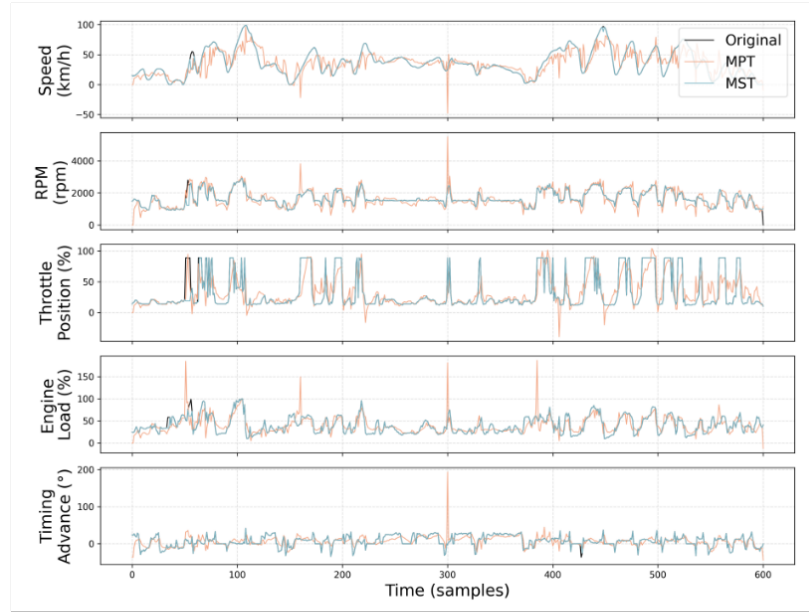
| Algorithm | Accuracy | Precision | Recall   | F1      |
|-----------|----------|-----------|----------|---------|
| MST       | 99.96 %  | 98.96 %   | 100.00 % | 99.84 % |
| MPT       | 92.78 %  | 20.63 %   | 91.65 %  | 33.67 % |

On the embedded device, MST averages 445–447  $\mu\text{s}$  per inference with a standard deviation of 9–11  $\mu\text{s}$  across both vehicles, comfortably within the 1 Hz real-time budget and with negligible variability. MPT requires approximately 4310–4322  $\mu\text{s}$ , an order of magnitude slower, approaching the timing limit for constrained devices. The correction errors introduced by the two algorithms (Table 4.7) confirm this trade-off quantitatively: MST corrections are localised and conservative (e.g. MAE of 8.0 RPM units on the Fastback), while MPT corrections are far larger (MAE of 222.8 RPM units), reflecting the damage done by

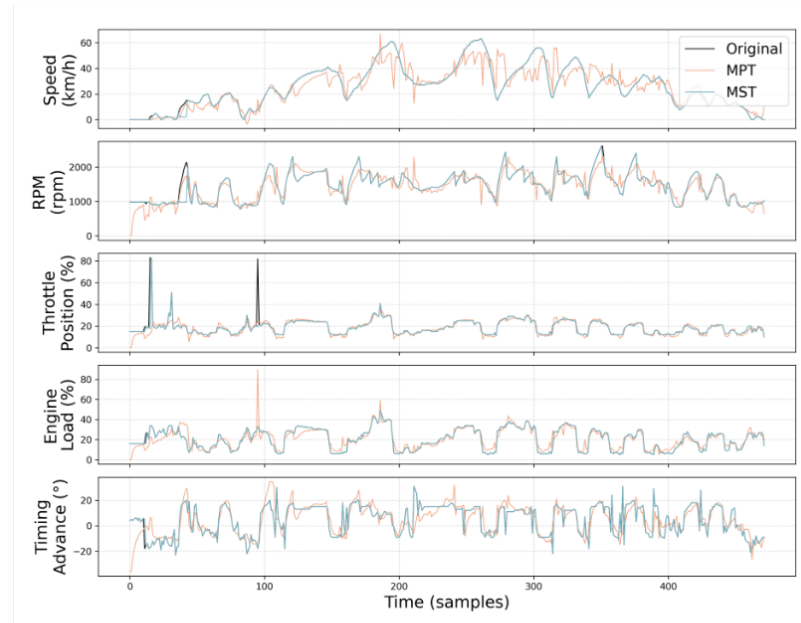
full-sample replacement when only one channel is anomalous.

Table 4.7: Deviation metrics for MST and MPT corrections.

| Algorithm | Variable     | MAE    | MSE       | RMSE   |
|-----------|--------------|--------|-----------|--------|
| MST       | Speed        | 0.15   | 3.90      | 1.97   |
|           | Engine RPM   | 8.03   | 10123.49  | 100.62 |
|           | Throttle Pos | 0.70   | 49.23     | 7.02   |
|           | Engine Load  | 0.23   | 6.66      | 2.58   |
|           | Timing Adv.  | 0.06   | 2.18      | 1.48   |
| MPT       | Speed        | 9.87   | 196.95    | 14.03  |
|           | Engine RPM   | 222.81 | 106735.97 | 326.70 |
|           | Throttle Pos | 12.67  | 373.14    | 19.32  |
|           | Engine Load  | 10.83  | 279.11    | 16.71  |
|           | Timing Adv.  | 10.10  | 234.73    | 15.32  |



(a)



(b)

Figure 4.7: Original and corrected sensor time series for MST and MPT on the two test vehicles. (a) Fiat Fastback and (b) Volkswagen Polo.

## 4.4 Summary

This chapter presented the design, implementation, and experimental evaluation of the core on-board monitoring platform and two complementary data management strategies, collectively addressing (PS 1) and (PS 2). The Black Box, a modular and containerised edge node mounted on the bicycle, embodies the hardware-independent, disaggregated architecture advocated in response to (PS 1): field tests confirmed a stable IMU acquisition rate of approximately 103 Hz, a GPS validity rate of 96.17% over a 25 km route, and an operational autonomy of up to 8 hours. In response to (PS 2), two on-device data management contributions were proposed and validated. The Autoencoder-based compression scheme achieves an 80% compression ratio on multivariate OBD-II streams while preserving downstream discriminative performance, with inference times below 30 ms on an ESP32 microcontroller. The MST and MPT incremental anomaly detection algorithms, derived from the TEDA-RLS framework, enable real-time anomaly detection and correction on TinyML-class devices: MST achieves near-perfect detection ( $F_1 = 99.84\%$ ) with an inference latency of 445–447  $\mu\text{s}$  and is the recommended choice for resource-constrained deployments. Taken together, the three components form a coherent edge analytics pipeline in which the Black Box acquires data, MST ensures its integrity, and the Autoencoder compresses it for efficient transmission over LoRaWAN or 5G.

## Chapter 5

# Technology for Health: Safety and Biometric Monitoring

This chapter addresses (PS 3) by moving from the vehicle platform to the rider. Monitoring the health and physical performance of LEV users in real-world conditions requires not only appropriate sensing hardware, but also data acquisition architectures capable of handling asynchronous, multi-device streams collected over large spatial extents. Beyond their role in personal transportation, IoT-equipped vehicles are increasingly recognized as valuable mobile sensing platforms: as they traverse cities and rural areas, these vehicles act as distributed probes, collecting data that would otherwise require costly fixed infrastructure. In doing so, this chapter closes the end-to-end pipeline of the thesis: where Chapter 4 established how data is acquired and processed on board over the connectivity characterised in Chapter 3, here those streams are consumed at scale to monitor rider health in real-world conditions. The proliferation of IoT technologies has enabled the large-scale collection of mobility-related data from various moving entities, including bicycles, cars, and pedestrians. In the context of ur-

ban cycling, particularly with the rise of e-bikes, onboard sensors and connected devices offer unprecedented opportunities to monitor physical, environmental, and behavioural parameters across time and space [55]. E-bikes equipped with GPS, inertial sensors, and communication modules can capture a wide variety of information, including road surface roughness, temperature, noise levels, vibration, wireless signal strength, and the physiological state of the rider. Air quality analysis, for instance, has already been investigated by some of the authors [56]. More broadly, a wide range of applications requires spatial sampling of quantities of interest, including environmental monitoring, infrastructure assessment, urban planning and mobility analysis, predictive maintenance, and communication diagnostics. However, the inherently mobile and asynchronous nature of these data sources poses significant challenges for spatial analysis and aggregation. Irregular sampling rates, varied trajectories, and heterogeneous user behaviours make it non-trivial to align and aggregate such data in a meaningful way. This chapter addresses these challenges by presenting the biometric data acquisition system deployed during the GiroE-2024 national e-bike tour and the graph-based spatial aggregation pipeline developed to align physiological observations to road-network nodes, enabling meaningful analysis across riders and stages.

## 5.1 Biometric Data Collection in Cycling Events

The systematic collection of biometric data during mass cycling events represents an emerging frontier at the intersection of sport science, wearable technology, and sustainable mobility research. Advances in multisensor smartwatch platforms have made it possible to acquire a wide range of physiological and environmental parameters in real time, opening application scenarios that span

from sports medicine to urban planning, and from infrastructure monitoring to cyclist safety assessment.

### 5.1.1 Study Context: the GiroE-2024

A compelling case study [57] is provided by the data acquisition campaign conducted during the GiroE-2024, a non-competitive e-bike event that runs concurrently with the Giro d'Italia (the second most prestigious stage race in professional cycling), sharing the same roads, stages, and daily schedule as the professional race [58]. During this event, a group of non-competitive cyclists completed 20 consecutive stages between 4 and 26 May 2024, covering approximately 1,100 km across diverse terrains, elevation profiles, and weather conditions. All participants rode on homologated pedal-assisted road bicycles (250 W motor, maximum assisted speed 25 km/h, 500 Wh battery), ensuring a consistent mechanical platform across the dataset.

The distinguishing technological feature of this campaign is the deployment of nine Garmin Fēnix® 7 Standard Edition smartwatches, a premium multisport device identified as a commercial reference for cycling activity by the non-competitive athletes and enthusiasts consulted. A custom application, the *MOST Garmin Raw Logger* [59], is installed on each device and purposely designed to log raw sensor data at the maximum available sampling frequency, without any filtering or preprocessing, in contrast to conventional activity-tracking applications that process data before displaying it.

### 5.1.2 Data Acquisition System Architecture

The resulting dataset is structured as a single comma-separated values file (`giroe2024.csv`), approximately 455 MB uncompressed, containing 625,703



Figure 5.1: Geographical locations of the 20 stages of the GiroE-2024 [58].

rows across 24 columns. Each row corresponds to a one-second sample from a specific device and includes the fields defined in Table 5.1.

Table 5.1: Description of the columns in the `giroe2024.csv` dataset.

| Column                 | Unit              | Description                                                               |
|------------------------|-------------------|---------------------------------------------------------------------------|
| Device                 | –                 | Integer serial number identifying the smartwatch                          |
| Session                | –                 | Timestamp of the first sample of a recording session                      |
| timestamp              | –                 | Timestamp of the individual sample                                        |
| position_lat           | deg               | Latitude (GNSS, LOCATION_CONTINUOUS)                                      |
| position_long          | deg               | Longitude (GNSS, LOCATION_CONTINUOUS)                                     |
| Distance               | km                | Cumulative distance from session start                                    |
| heart_rate             | bpm               | Heart rate from smartwatch OS interface                                   |
| SystemBattery          | %                 | Smartwatch battery level                                                  |
| PositionLatitude       | deg               | Latitude (Garmin IQ sensor interface)                                     |
| PositionLongitude      | deg               | Longitude (Garmin IQ sensor interface)                                    |
| PositionAltitude       | m                 | Altitude above sea level (GNSS)                                           |
| PositionSpeed          | m/s               | Speed from GNSS (fallback: accelerometer)                                 |
| PositionHeading        | rad               | Heading w.r.t. true north (GNSS)                                          |
| PositionAccuracy       | –                 | GNSS fix quality: 0 = n/a, 1 = last known, 2 = poor, 3 = usable, 4 = good |
| SensorAltitude         | m                 | Barometric or GNSS altitude                                               |
| SensorSpeed            | m/s               | Speed from GNSS (fallback: accelerometer)                                 |
| SensorHeading          | rad               | Heading w.r.t. true north (Garmin IQ sensor interface)                    |
| SensorPressure         | Pa                | Barometric pressure                                                       |
| SensorAccelerationX_HD | $g \cdot 10^{-3}$ | Array of 25 acceleration samples along the $x$ -axis at 25 Hz             |
| SensorAccelerationY_HD | $g \cdot 10^{-3}$ | Array of 25 acceleration samples along the $y$ -axis at 25 Hz             |
| SensorAccelerationZ_HD | $g \cdot 10^{-3}$ | Array of 25 acceleration samples along the $z$ -axis at 25 Hz             |
| SensorHeartRate        | bpm               | Heart rate from Garmin IQ sensor interface (PPG)                          |
| SensorTemperature      | °C                | Ambient temperature                                                       |
| Date                   | –                 | Date of the recorded session                                              |

All sensors are sampled at 1 Hz with the exception of the triaxial accelerometer, which operates at 25 Hz. The total recorded riding time amounts to 180.93 hours, covering a cumulative distance of 3,749.84 km across all devices and stages.

Data is available for each of the 20 GiroE-2024 stages, starting from 04/05/2024 to 26/05/2024, but due to the willingness of participants, which could change

stage by stage, the number of recorded devices can vary. A summary of the collected data by each device, stage by stage, is reported in Table 5.2.

Table 5.2: Recording duration (minutes) per device and stage during the GiroE-2024.

| Device    | Stage |    |     |     |     |     |    |     |     |     |     |     |     |     |     |     |     |     |     |    |
|-----------|-------|----|-----|-----|-----|-----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|----|
|           | 1     | 2  | 3   | 4   | 5   | 6   | 7  | 8   | 9   | 10  | 11  | 12  | 13  | 14  | 15  | 16  | 17  | 18  | 19  | 20 |
| 4493      | 100   | 0  | 170 | 0   | 0   | 0   | 0  | 0   | 0   | 0   | 143 | 189 | 0   | 110 | 0   | 145 | 0   | 195 | 0   | 0  |
| 4526      | 148   | 0  | 167 | 0   | 180 | 0   | 0  | 0   | 0   | 260 | 0   | 0   | 0   | 13  | 0   | 145 | 0   | 195 | 0   | 90 |
| 4595      | 89    | 0  | 0   | 0   | 0   | 0   | 0  | 0   | 0   | 0   | 143 | 0   | 0   | 0   | 240 | 145 | 0   | 195 | 0   | 0  |
| 3818      | 0     | 73 | 0   | 255 | 0   | 171 | 0  | 0   | 133 | 0   | 0   | 0   | 0   | 0   | 240 | 0   | 215 | 0   | 0   | 90 |
| 4141      | 0     | 75 | 0   | 255 | 0   | 0   | 0  | 182 | 0   | 0   | 0   | 0   | 150 | 0   | 240 | 0   | 215 | 195 | 0   | 90 |
| 4253      | 0     | 75 | 0   | 255 | 0   | 171 | 0  | 195 | 0   | 260 | 0   | 190 | 0   | 110 | 0   | 145 | 0   | 195 | 166 | 90 |
| 4530      | 0     | 75 | 0   | 255 | 0   | 0   | 86 | 0   | 133 | 0   | 0   | 0   | 0   | 0   | 240 | 0   | 215 | 195 | 0   | 90 |
| 4642      | 0     | 75 | 0   | 0   | 0   | 0   | 0  | 0   | 0   | 0   | 143 | 0   | 150 | 110 | 192 | 0   | 164 | 0   | 166 | 79 |
| 4155      | 0     | 0  | 180 | 0   | 0   | 0   | 0  | 0   | 133 | 0   | 143 | 0   | 126 | 0   | 240 | 0   | 0   | 0   | 166 | 0  |
| # devices | 3     | 5  | 3   | 4   | 1   | 2   | 1  | 2   | 3   | 2   | 4   | 2   | 3   | 4   | 6   | 4   | 4   | 6   | 3   | 6  |

## Operational Workflow and Data Management

The field protocol required a researcher to be present at the team paddock at both the start and finish of each stage. Before departure, athletes provided written informed consent, received the assigned smartwatch, and started the MOST Garmin Raw Logger application; upon arrival, devices are collected and data are immediately backed up to a cloud repository. Participation is entirely voluntary and could vary from stage to stage; the number of active devices ranged from 1 to 6 per stage, with a mean of approximately 3–4.

A notable technical consideration arose from the length and itinerant nature of the recording sessions. Automatic cloud synchronisation via Bluetooth or Wi-Fi proved unreliable due to session durations often exceeding three hours and

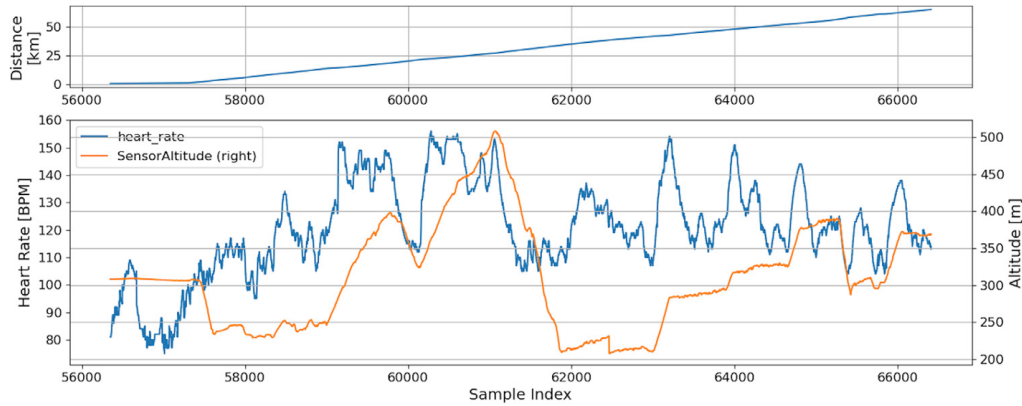


Figure 5.2: Example of performance data recorded by a single smartwatch during a GiroE-2024 stage. The heart rate profile reflects not only changes in road gradient but also the cyclist’s use of motor assistance, a distinctive feature of e-bike riding

the frequent absence of stable internet connectivity at remote stage locations, conditions that repeatedly triggered Garmin Cloud timeouts. For this reason, the application is designed to permanently store data in FIT files on the device, which are retrieved via USB at the end of the event, converted to JSON using the `fitdecode` Python library, and subsequently exported to CSV format. Each trace is then manually inspected and trimmed around the expected stage start and finish times extracted from the official race roadbook, in order to remove pre-race warm-up and post-race cool-down segments.

### 5.1.3 Methodological Challenges and Scientific Value

Collecting biometric data in a real-world competitive context such as the GiroE-2024 introduces a set of methodological challenges that simultaneously enrich the scientific value of the resulting dataset. First, the participant cohort is delib-

erately heterogeneous, ranging from beginners to experienced cyclists, allowing the observation of markedly different physiological responses to identical route demands. Second, each participant obtained a non-competitive sports medical certificate issued by a sports medicine specialist, as required by Italian law; this qualification enables the dataset to serve as a ground truth for subjects in certified health condition, a valuable reference for the validation of clinical algorithms or heart rate modelling approaches.

Unlike the majority of cycling datasets that focus on elite professional athletes, this corpus of data is unique as it characterizes non-competitive athletes who have obtained a mandatory non-agonistic sports medical certificate. This allows the dataset to serve as a ground truth for public health and urban micromobility research, providing a baseline for 'normal' physiological responses under real-world strenuous conditions. Furthermore, the implementation of the custom MOST Garmin Raw Logger application was pivotal in bypassing the filtering and pre-processing typically found in commercial fitness apps, enabling the acquisition of raw data from biometric (1 Hz) and accelerometric (25 Hz) sensors at their maximum sampling frequency. This level of granularity facilitates the identification of specific e-bike peculiarities, such as heart rate variations driven not only by terrain slope but also by the dynamic selection of motor assistance levels chosen by the rider

The heart rate signal acquires a particularly nuanced significance in the e-bike context. Unlike conventional cycling, the availability of motor assistance (under direct rider control) introduces discontinuities in cardiovascular load that cannot be attributed solely to changes in gradient or speed. This interaction, visible in Figure 5.2, makes heart rate modelling in e-bike scenarios an open research problem, potentially more complex than in conventional cycling and of direct relevance to personalised training and cardiac risk assessment.

The dataset is publicly available on Zenodo under a CC BY licence, and can be used for evaluating road surface estimation techniques, cyclist behaviour analysis, heart rate modelling, and as a benchmark for smartwatch data fusion with other sensing modalities.

## 5.2 Spatial Aggregation of IoT Mobility Data

The availability of georeferenced, multi-user datasets such as the GiroE-2024 corpus immediately raises a fundamental analysis challenge: how to meaningfully aggregate asynchronous measurements, sampled at variable rates, from cyclists following non-perfectly overlapping trajectories? In [60] is proposed a graph-based spatial aggregation framework for IoT data collected from light electric vehicles.

### 5.2.1 Reference Node Construction

A central design choice of the framework concerns the definition of spatial reference nodes. The default road network nodes provided by the OSMnx library [61] (parameter extracted using the `network_type='drive'`) are found to be unsuitable, as they failed to cover the full extent of the cycling course, particularly on sections not coinciding with driveable roads. Instead, reference nodes are manually defined along the official route geometry using Google Maps, exported as GPX files via the mapsToGPX tool, and imported using the gpxpy library. The spatial density of nodes is adapted to route geometry: denser placement in curves, intersections, and topographically complex sections; sparser in straight segments. The graph is then constructed using NetworkX [62] and the GeoDataFrame containing the Reference point geometries, preserving their geographic coordinates.

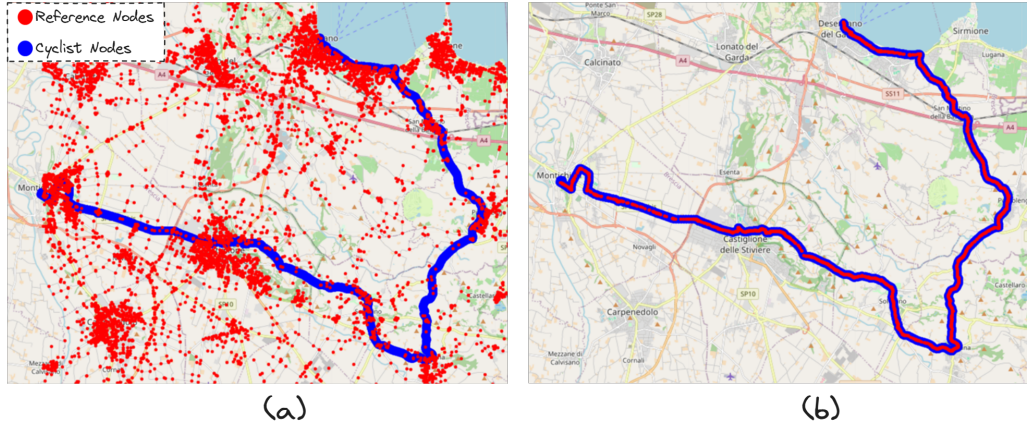


Figure 5.3: Comparison between OSMnx road network nodes and manually defined reference nodes along a GiroE-2024 stage.

### 5.2.2 Aggregation Pipeline

The aggregation pipeline proceeds through four stages, illustrated schematically in Figure 5.4.

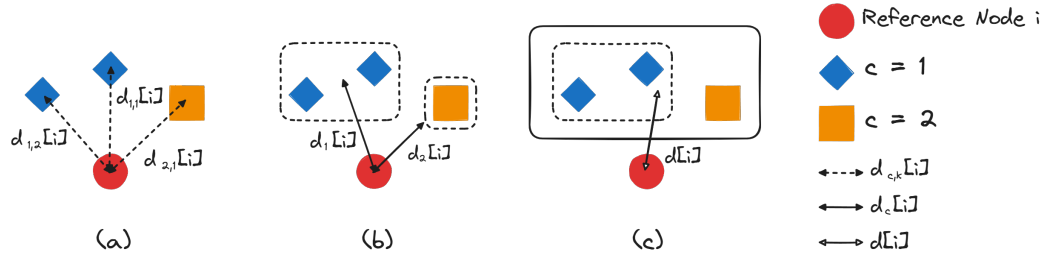


Figure 5.4: Schematic of the spatial aggregation pipeline. (a) Position matching. (b) Per-cyclist aggregation. (c) Cross-cyclist aggregation.

**Dataset preparation.** Only tracks involving more than one simultaneously active cyclist are retained. Entries with missing critical features (e.g., heart rate) are discarded. Track boundaries are aligned to the interval during which all cyclists have active data collection, and a speed-based filter removes non-racing

segments.

**Position matching.** (Figure 5.4(a)) Since the graph operates in unprojected geographic coordinates, nearest-node searches employ a haversine distance metric on a ball tree structure. Each GPS sample from cyclist  $c$  is assigned to its nearest reference node  $i$  distanced  $d_{c,i}$ , yielding a set of  $K_c[i]$  matched instances per cyclist per node.

**Per-cyclist aggregation.** (Figure 5.4(b)) Multiple instances of the same cyclist matched to the same node are averaged to produce a single per-cyclist feature value and mean distance:

$$f_c[i] = \frac{\sum_{k=1}^{K_c[i]} f_{c,k}[i]}{K_c[i]}, \quad d_c[i] = \frac{\sum_{k=1}^{K_c[i]} d_{c,k}[i]}{K_c[i]} \quad (5.1)$$

**Cross-cyclist aggregation.** (Figure 5.4(c)) For each reference node  $i$ , the values of all  $P[i]$  cyclists with valid data are combined:

$$f[i] = \frac{\sum_{c=1}^C f_c[i]}{P[i]}, \quad d[i] = \frac{\sum_{c=1}^C d_c[i]}{P[i]}, \quad s[i] = \sqrt{\frac{\sum_{c=1}^C (d_c[i] - d[i])^2}{P[i]}} \quad (5.2)$$

where  $s[i]$  quantifies the spread of cyclist-to-node alignment and provides a qualitative indicator of aggregation fidelity at each node.

### 5.2.3 Experimental Results

The framework is validated on three stages of the GiroE-2024 (stages 4, 13, and 14), whose aggregation quality statistics are summarised in Table 5.3.

Mean distances between GPS samples and their assigned reference nodes ranged from 6.19 to 11.48 m, with a limited number of points exceeding the 95th percentile threshold. Stage 13, characterised by extended straight sections,

Table 5.3: Spatial aggregation statistics for three GiroE-2024 stages.  $N$ : number of cyclists;  $I$ : number of reference nodes; distances in metres.

| Stage | $N$ | $I$   | Min (m) | Max (m) | 5th pct (m) | 95th pct (m) |
|-------|-----|-------|---------|---------|-------------|--------------|
| 4     | 4   | 1,302 | 0.09    | 38.55   | 1.85        | 13.62        |
| 13    | 3   | 1,533 | 0.29    | 85.49   | 2.09        | 31.48        |
| 14    | 3   | 2,287 | 0.13    | 68.34   | 1.47        | 18.08        |

exhibited the highest maximum distance (85.49 m) and the largest standard deviation (10 m), suggesting that sparse node placement in geometrically simple segments reduces spatial fidelity. The spatial distribution of  $d[i]$  and  $s[i]$  along all three stages is visualised in Figure 5.5.

#### 5.2.4 Discussion and Application Perspectives

The integration of wearable biometric acquisition and graph-based spatial aggregation establishes a methodological foundation for several application domains relevant to cyclist safety and well-being. On the infrastructure side, the correlation between high-frequency accelerometric data and spatial position enables the construction of continuous road surface quality maps, with the potential to identify potholes, vibrations, or surface degradation at metric granularity. On the physiological side, the joint analysis of elevation profile, speed, heart rate, and motor assistance level opens pathways toward personalised cardiovascular load modelling, with implications both for training optimisation and for risk assessment in subjects with borderline cardiac conditions.

Compared to traditional map-matching approaches based on Hidden Markov Models, which typically assume uniform sampling or rely on predefined road infrastructure, the proposed graph-based framework adapts directly to the actual

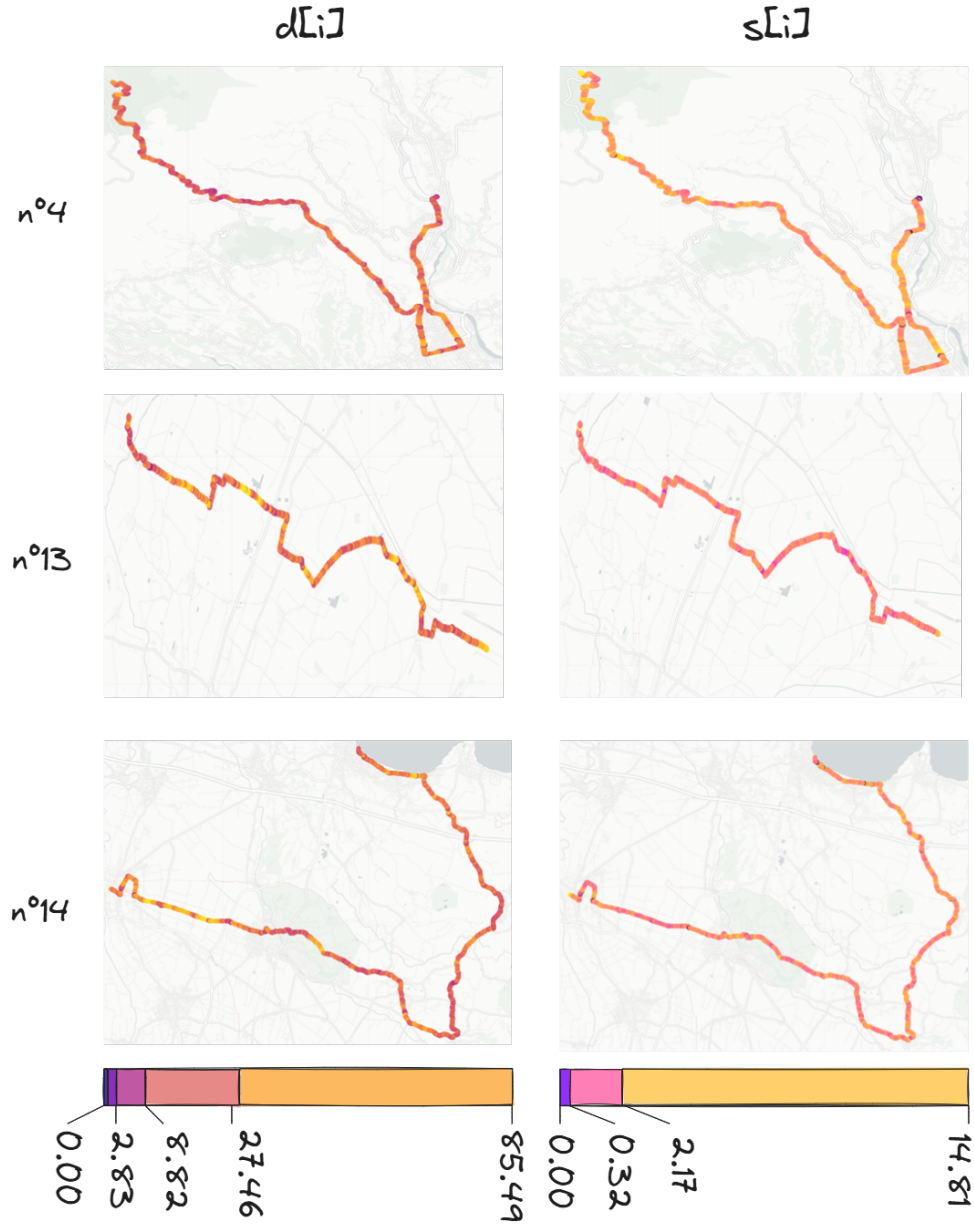


Figure 5.5: Spatial distribution of mean distance  $d[i]$  (left) and its standard deviation  $s[i]$  (right) along the routes of three GiroE-2024 stages.

route geometry and accommodates inter-user variability through multi-user averaging. The observed alignment metrics and data retention rates confirm that the approach achieves a favourable trade-off between spatial fidelity and computational simplicity, making it suitable for integration into higher-level analytical pipelines.

### 5.3 Summary

This chapter addressed (PS 3) by developing the data acquisition and spatial analysis framework needed to monitor the health and physical performance of LEV riders in real-world, large-scale conditions. The biometric data acquisition system, deployed during all 20 stages of the GiroE-2024 national e-bike tour, collected 180.93 hours of riding data covering 3,749.84 km, including physiological parameters, GNSS traces, and motion signals from a cohort of non-professional riders. The graph-based spatial aggregation pipeline, built on OSMnx road-network nodes, aligns physiological observations from multiple riders to shared spatial references using a haversine-based nearest-node assignment, enabling meaningful cross-rider and cross-stage analysis with a mean matching distance between 6.19 m and 11.48 m across the evaluated stages. The dataset is publicly available on Zenodo under a CC BY licence [63], making it a reusable benchmark for future research in smart mobility and Technology for Health.

## **Chapter 6**

# **Energy Sustainability and Off-Grid Infrastructures**

This chapter addresses the two remaining challenges. In response to (PS 4), it presents an autonomous fall detection system for light vehicles, combining low-power inertial sensing with LoRaWAN communication to guarantee emergency alerting independently of network availability. In response to (PS 5), it then presents simulation-based methodologies for the design and techno-environmental assessment of PV-powered e-bike charging stations, with particular attention to the trade-off between the energy demand of the embedded ICT monitoring subsystem and the energy available for vehicle charging.

### **6.1 Fall Detection Systems for Light Mobility**

The growing adoption of light vehicles in multimodal urban mobility scenarios introduces a category of users that remains largely unmonitored from a safety standpoint. Unlike motor vehicles, standard bicycles carry no embedded elec-

tronics capable of detecting incidents or alerting emergency services in the event of a fall. At the same time, the operational constraints of a bicycle-mounted IoT device are severe: the solution must be low-cost, unobtrusive, require minimal maintenance, and sustain autonomous operation over extended periods without user intervention. The work of Dello Iacono et al. [64] addresses this problem by proposing and characterising a proof-of-concept (PoC) IoT node for fall detection based on LoRaWAN communication and solar energy harvesting. Recent work on pre-impact fall detection for e-scooters likewise confirms that sub-50 kB on-device models can achieve high detection accuracy with lead times compatible with real-time intervention [65].

### 6.1.1 System Architecture

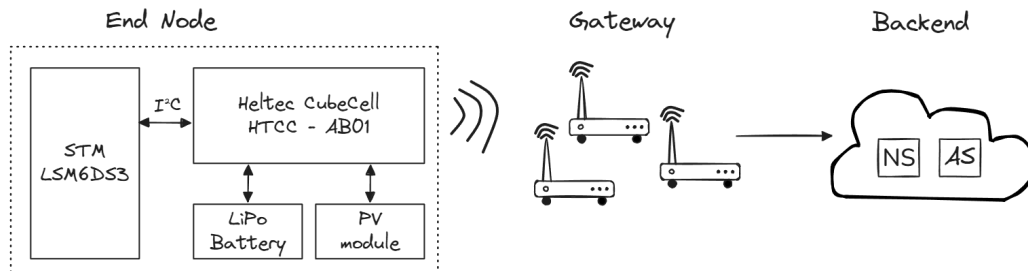


Figure 6.1: Block diagram of the proposed fall detection architecture.

The end node is built around the *CubeCell HTCC-AB01* development board by Heltec Automation, which integrates an ARM Cortex-M0 microcontroller unit (MCU) and a Semtech SX1262 LoRa transceiver on a single platform derived from the Infineon PSoC 4000 architecture. Fall and motion detection are delegated to the *STMicroelectronics LSM6DS3* Inertial Measurement Unit (IMU), a triaxial accelerometer and gyroscope communicating with the MCU via I<sup>2</sup>C. A key design advantage of this component is its hardware free-fall interrupt line, which allows

the MCU to remain in deep sleep until a candidate fall event is detected by the IMU autonomously, without continuous polling.

The primary power source is a LiPo battery, supplemented by a small photovoltaic (PV) panel whose output is managed by the on-board charge circuitry. The overall footprint of the assembled prototype does not exceed the panel dimensions, approximately  $84 \times 40$  mm.

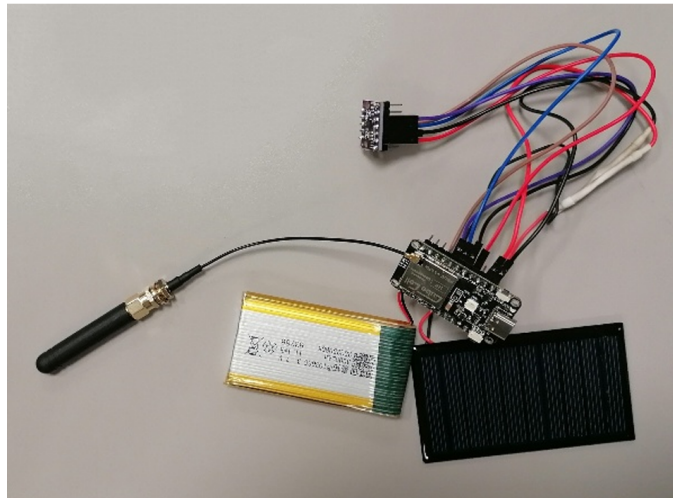


Figure 6.2: Photograph of the developed prototype. The overall footprint is constrained by the solar panel dimensions ( $84 \times 40$  mm).

### Firmware and State Machine

The firmware is structured around a software Finite State Machine (FSM) that coordinates the LoRaWAN stack and the IMU driver. On power-up, the device joins the network via Over-The-Air Activation (OTAA) and configures the IMU; it then enters deep sleep, waiting for a hardware interrupt from the IMU free-fall detection output. Periodically, the MCU wakes up to transmit a heartbeat message containing the device identifier and the instantaneous battery voltage, measured

by the internal 12-bit ADC, and to verify LoRaWAN connectivity, attempting a re-join if the link has been lost.

### Backend Infrastructure

The backend relies on TTN, a decentralised LoRaWAN infrastructure that exposes received frames to downstream applications via an MQTT broker. This choice enables straightforward integration with monitoring dashboards such as Grafana and with custom data collection scripts, while avoiding dependence on a proprietary third-party provider.

## 6.1.2 Experimental Characterisation

### Current Consumption

Static current consumption is measured using a Keysight 34450A bench multimeter in series with the supply line, averaging over 180 s per test condition. The results, summarised in Table 6.1, confirm the effectiveness of the interrupt-driven approach: the idle current in interrupt mode (approximately 1.1 mA) is more than ten times lower than in polling mode (14.15 mA), and the power source (USB or battery) has negligible impact.

Transient consumption during LoRa transmission is characterised with a Keysight DSOX2004A oscilloscope and a 10  $\Omega$  shunt resistor. Three phases are identified: an idle phase at approximately 1.1 mA, a radio wake-up phase at approximately 10 mA lasting about 1 s, and an active transmission phase peaking at approximately 80 mA for about 1 s. The resulting current consumption profile is illustrated in Figure 6.3.

Assuming a heartbeat period of 300 s and a supply voltage of 3.3 V, the estimated average power consumption is approximately 4.72 mWh. For a 3000 mAh

Table 6.1: Average static current consumption of the PoC under different operating conditions.

| Power source | MCU mode  | IMU mode  | Avg. current |
|--------------|-----------|-----------|--------------|
| USB          | Low Power | OFF       | 16.5 $\mu$ A |
| USB          | Low Power | Interrupt | 1.05 mA      |
| Battery      | Low Power | OFF       | 17.0 $\mu$ A |
| Battery      | Low Power | Interrupt | 1.11 mA      |
| Battery      | Low Power | Polling   | 14.15 mA     |

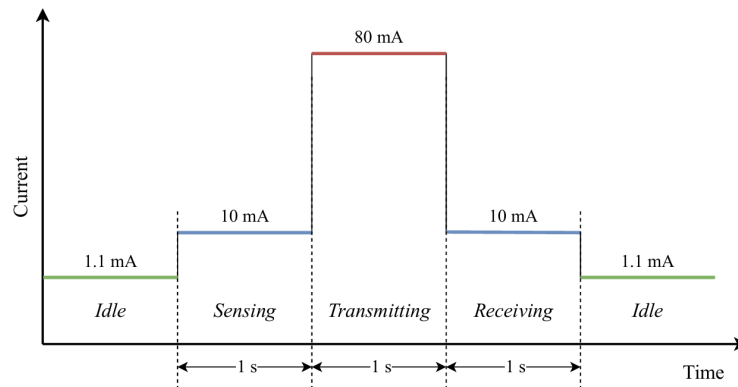


Figure 6.3: Average current consumption model during a LoRa uplink transmission cycle.

Li-Ion cell operated in its constant-current region, this yields an estimated battery lifetime of about 87 days without solar harvesting.

### LoRaWAN Network Performance

Network latency and signal quality are characterized over a five-day campaign using two prototypes deployed approximately 500 m from the University of Brescia LoRa gateway. A total of 1500 heartbeat messages are sent at 300 s intervals and timestamped at the gateway ( $T_{\text{GW}}$ ), network server ( $T_{\text{NS}}$ ), and application server ( $T_{\text{AS}}$ ). The results are reported in Table 6.2.

Table 6.2: LoRaWAN network performance statistics over 1500 uplink messages.

| <b>Metric</b> | $T_{\text{NS}} - T_{\text{GW}}$ | $T_{\text{AS}} - T_{\text{GW}}$ | <b>RSSI</b> |
|---------------|---------------------------------|---------------------------------|-------------|
| $\mu$         | 0.016 s                         | 0.221 s                         | −116.5 dBm  |
| $\sigma$      | 0.005 s                         | 0.007 s                         | 4.2 dBm     |
| $Q_{95\%}$    | 0.035 s                         | 0.243 s                         | −121.6 dBm  |
| $Q_{5\%}$     | <0.001 s                        | 0.204 s                         | −105.4 dBm  |

The end-to-end latency from gateway reception to application server delivery has a mean of 221 ms with a standard deviation of only 7 ms, confirming the suitability of TTN as a reliable backend for time-sensitive event reporting. The mean RSSI of −116.5 dBm, recorded at 500 m from the gateway in an urban environment, is well within the LoRa sensitivity margin, leaving considerable link budget for larger coverage scenarios typical of rural or suburban cycling routes.

Battery voltage is monitored throughout the five-day experiment via the on-board ADC. Starting from a 90% charge level, the terminal voltage decreased lin-

early from 4.008 V to 3.975 V, corresponding to a discharge rate of approximately 6.6 mV/day, consistent with the estimated power budget.

### **6.1.3 Discussion**

The results demonstrate that LoRaWAN, combined with an interrupt-driven IMU and solar energy harvesting, constitutes a viable communication and power architecture for unattended safety monitoring of light vehicles. The hardware free-fall interrupt of the LSM6DS3 is central to achieving sub-milliampere idle consumption, while the periodic heartbeat mechanism ensures both network keepalive and continuous battery monitoring without significant energy overhead. The estimated 87-day autonomous lifetime without solar input already exceeds the typical maintenance cycle of a shared bicycle fleet; the addition of even a small PV panel is expected to extend operation indefinitely under normal outdoor conditions.

From a network perspective, the sub-250 ms end-to-end latency is adequate for post-event notification of falls, where the primary requirement is reliable delivery to a monitoring centre rather than real-time reaction. The standardised TTN backend and MQTT integration further simplify the deployment of fleet-scale monitoring infrastructure without reliance on proprietary connectivity providers, a relevant consideration for the rural and peri-urban routes typical of leisure cycling and bike-sharing services.

## 6.2 ICT-Equipped Portable E-Bike Charging Station for Mass Cycling Events

The rapid growth of the e-bike market, combined with the increasing popularity of mass cycling events oriented toward sustainability and green tourism, creates concentrated charging demand that existing fixed infrastructure is not designed to absorb. A portable, fully off-grid charging station powered by photovoltaic modules represents a practical response to this need, provided that the ICT equipment is carefully sized so as not to compromise the limited energy budget available from a compact PV system. The work of Astolfi et al. [66] addresses precisely this design challenge, presenting a simulation-based methodology for tuning the battery storage and ICT load of a portable charging station intended for deployment at the GiroE-2024 competition.

### 6.2.1 System Design and Constraints

The charging station is designed to operate off-grid at 24 V DC, with a rated PV power of 200 W supplied by two compact 100 W modules selected to satisfy portability constraints. The complete structure is dismountable, with a total weight below 15 kg, making it transportable by a small electric vehicle and assembleable by a single operator. The power chain includes an MPPT controller, a Battery Management System (BMS), a battery balancer, a smart shunt for state-of-charge estimation, and a 250 VA hybrid inverter connected to two Bosch Compact Chargers (2 A each) for e-bike recharging.

Two ICT scenarios are considered. In the *cloud-based* scenario, the communication centre monitors PV voltage and current, MPPT output, DC link voltage, battery voltage and current, and inverter parameters, transmitting aggregated

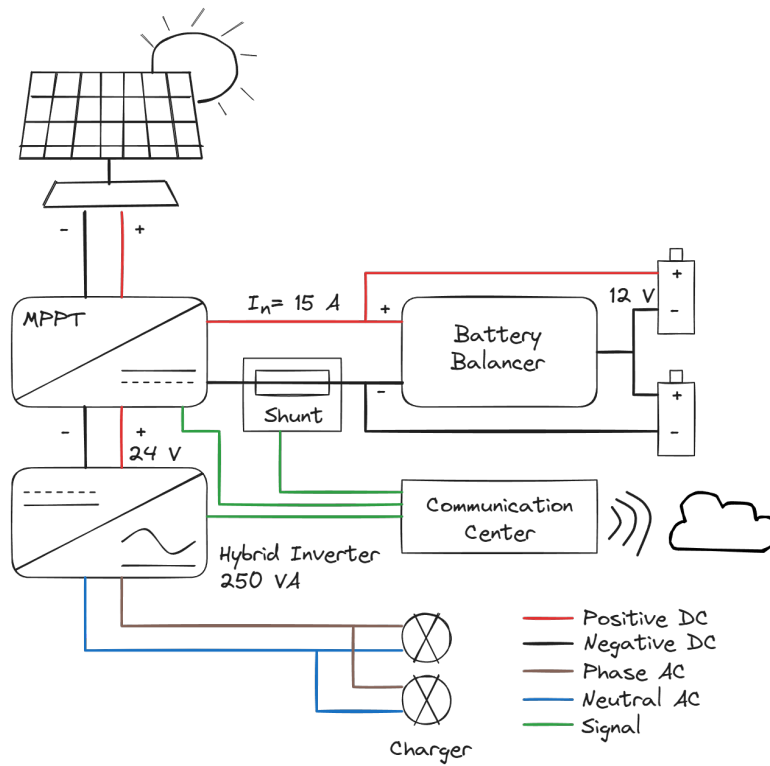


Figure 6.4: Electrical diagram of the proposed portable e-bike charging station.

data to a remote cloud. The estimated continuous power demand of all components in this scenario totals approximately 12 W, of which about 4 W is attributable to the communication centre and the remainder to power electronics (MPPT, BMS, inverter, chargers). In the *edge-based* scenario, an industrial router (6 W) and an edge computer (12 W) are added to enable on-site smart charging algorithms, raising the total ICT and electronics load to approximately 30 W, equivalent to 15% of the rated PV power.

### 6.2.2 Simulation Methodology

PV production is estimated using the PVGIS-SARAH2 database for a representative location along the GiroE route (Montichiari, latitude 45.425°N, longitude 10.382°E), with a tilt of 40° and an azimuth of  $-2$ , at hourly resolution for the year 2020. Conversion efficiencies are set to  $\eta_{DC/DC} = 87\%$  and  $\eta_{DC/AC} = 88\%$ , with a battery round-trip efficiency of 92% and a self-discharge of 1% per day. The e-bike charging load is modelled as a constant 135 W demand over a two-hour charge session, consistent with measured profiles of commercial 500 Wh e-bike batteries.

Three charging demand scenarios are evaluated, spanning the main categories of user behaviour. Scenario 1 places both daily charges in the morning (8–10 a.m. and 10 a.m.–noon), stressing the system because the battery is depleted by the first charge before irradiance is sufficient to replenish it. Scenario 2 combines an early morning charge with a late afternoon charge (7–9 p.m.), where evening irradiance is insufficient and the second charge relies entirely on stored energy. Scenario 3 places both charges during peak irradiance hours (10 a.m.–noon and 2–4 p.m.), representing the least critical case.

Three performance metrics are used to evaluate the system. The *Quality of Service* (QoS) is the annual fraction of requested charging time during which

power is actually available, with a minimum acceptable threshold of 50%. The *Loss of Main Power Supply* is the annual fraction of expected operational time during which no power is available at all, with a threshold of 30% (acknowledging that nighttime unavailability is inherently acceptable). The *Off-Tracking Loss* measures PV energy curtailed due to a full battery, with a target below 10%.

### 6.2.3 Results and ICT Sizing Guidelines

In the cloud-based ICT scenario, the daily energy consumed by ICT is 288 Wh, comparable to about half the energy required for the two e-bike charges (540 Wh). A battery storage of 500 Wh is identified as the appropriate selection: in Scenarios 1 and 3 the QoS reaches 92% and 95% of its asymptotic value at this capacity, while in the more critical Scenario 2 it stands at 60%, which is 81% of the value reachable with 1500 Wh. The QoS values for the cloud-based and edge-based scenarios with 500 Wh of storage are summarised in Table 6.3.

Table 6.3: Quality of Service (%) for the cloud-based and edge-based ICT scenarios with 500 Wh of battery storage.

| Scenario           | Cloud ICT | Edge ICT |
|--------------------|-----------|----------|
| 1 (morning peak)   | 69.7%     | 32.9%    |
| 2 (evening charge) | 58.2%     | 40.6%    |
| 3 (midday charges) | 76.0%     | 64.3%    |

In the edge-based scenario the total ICT energy consumption (720 Wh/day) exceeds the e-bike charging demand by 33%, and the QoS in Scenario 1 drops to 32.9%, less than half the cloud-based value. A sensitivity analysis, in which the electronics and ICT power demand is varied from 12 W to 30 W in 2 W steps,

reveals a linear relationship between ICT load and both QoS and loss of main power supply ( $R^2 = 0.998$  and  $R^2 = 0.985$  respectively). To maintain a QoS above 50% in the most critical scenario, the ICT load must not exceed 20 W.

The principal design guideline emerging from this analysis is that, in a 200 W PV system of this type, the purely ICT power consumption should remain below a few percent of the rated PV power (2.5% in the cloud-based case) and should not exceed one third of the total baseline electronics load. This constraint effectively rules out full edge-computing equipment in the most demanding charging scenarios, while cloud-based monitoring remains compatible with acceptable service quality across all three scenarios.

## 6.3 Techno-Environmental Assessment of PV-Powered Charging Stations

The adoption of PV-powered charging stations for Light Electric Vehicles (LEVs) is intuitively appealing from an environmental standpoint, but the actual reduction in greenhouse gas (GHG) emissions depends on a complex interplay between storage capacity, demand profile, operating mode, and system lifetime. Pasetti et al. [67] address this question through a comprehensive simulation framework that models a representative LEV charging station over a 20-year lifecycle, evaluating both island-mode and hybrid operation under five distinct daily demand profiles.

### 6.3.1 Use Case and System Architecture

The reference system is a charging station designed for up to 8 LEVs, equipped with a 2.6 kWp PV array (6 modules LG440QAC-A6, 440.5 W each), a 3 kWp

hybrid inverter with 90% rated efficiency, and a  $\text{LiFePO}_4$  battery storage whose nominal capacity  $C$  is treated as a free parameter varying from 200 Wh to 6 kWh in 200 Wh steps . The system operates in two modes. In *island mode*, the LEV charging bar is fed exclusively by the PV-storage system, completely disconnected from the public grid. When the combined PV and storage capacity is insufficient, the station switches to *grid mode*, connecting the charging bar to the utility while the PV-storage system continues to operate in island mode to recharge the battery if solar generation is available. Grid-tied PV export is not considered, due to the regulatory complexity associated with small PV capacities.

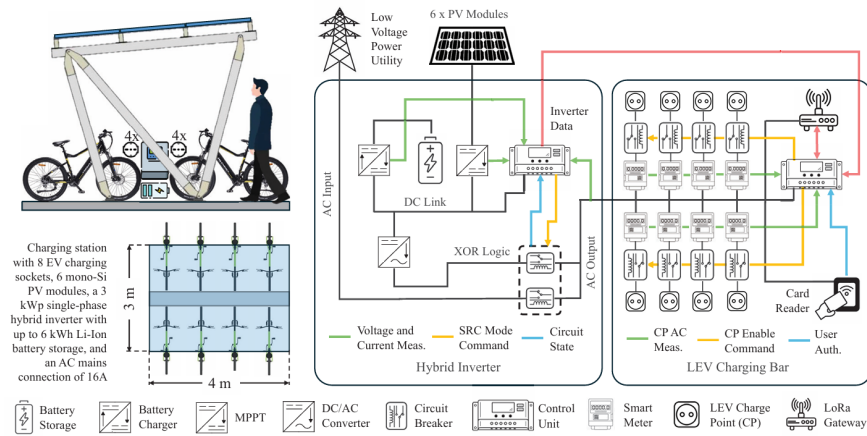


Figure 6.5: Conceptual layout of the PV-powered LEV charging station.

### 6.3.2 Simulation Framework

PV production is computed from Typical Meteorological Year (TMY) data retrieved from PVGIS for a coastal location in Sardinia ( $\varphi = 39.212^\circ\text{N}$ ,  $\lambda = 9.104^\circ\text{E}$ ), representative of a touristic site with strong solar resource. The PV cell temperature is estimated iteratively using the NOCT-based thermal model of Eq. 6.1, and

the maximum power point output is derived from the single-diode equivalent circuit:

$$\hat{T}_{\text{cell}}[k] = T_a[k] + \frac{9.5 G_T[k] (T_{c,n} - T_{a,n})}{G_{T,n}(5.7 + 3.8 w_s)} \left( 1 - \frac{\hat{\eta}_{\text{cell}}}{\tau \alpha} \right) \quad (6.1)$$

The battery model implements constant-current (CC) and constant-voltage (CV) charging phases, with a maximum charge and discharge rate of 1C, DC charge and discharge efficiencies of 94%, and a self-discharge rate of 5%/day from 100% SoC. The minimum and maximum SoC are fixed at 20% and 100% respectively, with the CC-to-CV transition occurring at 90% SoC.

Five daily demand profiles are considered: average load, full load (average plus one standard deviation), and morning, afternoon, and evening peak variants, each constructed from 10,000 Monte Carlo simulations of sharing-service and private user arrivals.

### 6.3.3 GHG Emission Model and Key Performance Indicators

The global GHG emission  $\varepsilon_S$  (kg CO<sub>2</sub> eq./kWh delivered) is computed over the 20-year system lifetime  $L_S$ , accounting for the manufacturing footprint of PV modules ( $\varepsilon_{\text{PV}} = 623$  kg CO<sub>2</sub>/kWp), Li-Ion batteries ( $\varepsilon_{\text{batt}} = 122$  kg CO<sub>2</sub>/kWh, replaced every 10 years), and grid energy under the Italian 2020 energy mix ( $\varepsilon_U = 0.2134$  kg CO<sub>2</sub>/kWh):

$$\varepsilon_S = \frac{\varepsilon_{\text{PV}} E_{\text{PV},n} + \frac{\varepsilon_{\text{batt}} C L_S}{L_{\text{batt}}} + L_S \varepsilon_U \sum_k E_{\text{FU}}[k]}{L_S (\sum_k (E_{\text{PV}}[k] + E_{\text{FS}}[k]) \eta_{\text{inv}} + \sum_k E_{\text{FU}}[k])} \quad (6.2)$$

In addition to GHG emissions, four technical KPIs are evaluated: the *Quality of Service* (ratio of energy delivered to theoretical demand, applicable in is-

land mode only), the *Battery Stress Rate* (fraction of time with SoC outside the  $[SoC_{Low}, SoC_{High}]$  interval), the *PV Derating Rate* (fraction of theoretically available PV energy lost due to full battery), and the *Battery Energy Supply Rate* (fraction of LEV demand covered by the battery).

### 6.3.4 Results

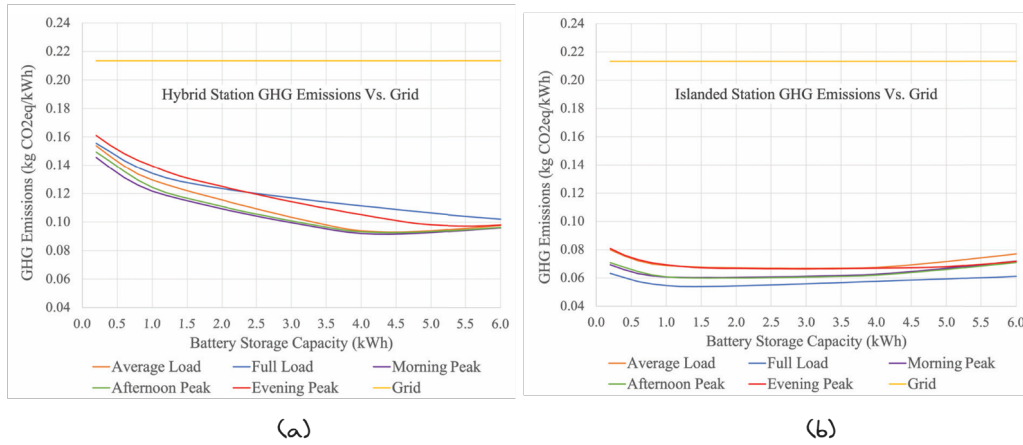
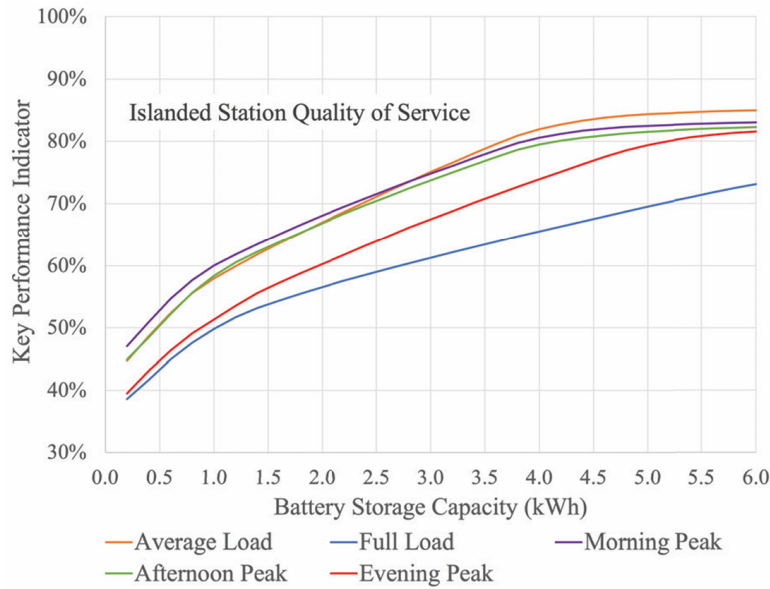


Figure 6.6: GHG emissions as a function of battery storage capacity for the hybrid (a) and island (b) configurations, compared against the grid-only baseline.

In both operating modes, GHG emissions are substantially lower than the grid reference across all demand scenarios and storage capacities. The evening-peak profile is the most penalising: in island mode, even with 6 kWh of storage the Quality of Service only marginally improves compared to 3 kWh, and the asymptotic QoS ceiling is approximately 80% for average, morning, and afternoon profiles, dropping to 70% under full load.

The analysis of technical KPIs for the average load profile reveals a threshold around 4 kWh, below which increasing storage capacity produces concurrent improvements in PV derating losses, grid energy supply rate, and battery supply



rate. Beyond this threshold, energy flow metrics improve only marginally while the battery stress rate continues to decrease, suggesting that larger batteries may be justified in hybrid operation to extend battery lifetime rather than to increase self-sufficiency.

In hybrid mode, the impact of storage capacity is most pronounced near evening demand peaks: comparing 3 kWh and 6 kWh configurations, the average hourly grid power demand differs significantly only in the evening hours and only under the evening-peak scenario. This result indicates that smart charging coordination among multiple stations becomes necessary to effectively manage peak grid demand, since isolated operation of each station is insufficient to absorb the worst-case demand scenarios.

### 6.3.5 Discussion

The main conclusion of the study is that PV-powered LEV charging stations, in both island and hybrid configurations, achieve a meaningful reduction in life-

cycle GHG emissions compared to conventional grid recharging under the Italian energy mix. However, the grid connection remains a necessary complement in hybrid mode to maintain an adequate Quality of Service, particularly under evening-peaked demand distributions typical of recreational and tourism scenarios. The sizing of the battery storage presents a diminishing returns characteristic: once a threshold capacity (approximately 4 kWh for this system) is reached, further investment in storage is better justified by lifetime considerations through reduced battery stress than by additional energy self-sufficiency gains. These results motivate the development of coordinated smart charging strategies across multiple connected stations, where ICT and communication infrastructure, as discussed in Section 6.2, play a central enabling role.

## 6.4 Summary

This chapter addressed the two remaining challenges identified in the Problem Statement. In response to (PS 4), an autonomous fall detection system for light vehicles is presented, combining IMU-based free-fall detection with LoRaWAN communication and solar energy harvesting. The interrupt-driven architecture achieves an idle current of approximately 12  $\mu\text{A}$ , yielding an estimated battery lifetime exceeding 87 days without solar contribution, while maintaining a median alert delivery latency of 221 ms. In response to (PS 5), two simulation-based contributions established methodologies for the design and techno-environmental assessment of PV-powered e-bike charging stations. The portable station study derived ICT sizing guidelines showing that monitoring hardware must be carefully dimensioned to avoid competing with the charging load for the finite energy budget. The techno-environmental assessment demonstrated that PV-powered LEV charging stations, in both island and hybrid configurations, achieve a mean-

---

ingful reduction in lifecycle GHG emissions compared to conventional grid recharging under the Italian energy mix, with battery storage presenting a diminishing returns characteristic beyond approximately 4 kWh.

# Chapter 7

## Conclusion and Future Work

### 7.1 Revisiting the Problem Statement

This thesis originated from five concrete gaps in the current LEV ecosystem, identified in Section 1.2: the lack of modular and hardware-independent on-board architectures (PS 1), the difficulty of managing large volumes of sensor data on resource-constrained devices (PS 2), the complexity of collecting and aggregating biometric data from riders in real-world conditions (PS 3), the absence of autonomous safety mechanisms capable of operating independently of network infrastructure (PS 4), and the lack of established methodologies for the design of energy-sustainable off-grid charging stations (PS 5). These challenges, while emerging from the specific context of light electric mobility, are not unique to it. As argued in Section 1.3, each of them has a structural counterpart in industrial automation, a domain that has already developed mature, validated responses. The central thesis of this work is therefore not to invent new paradigms, but to demonstrate that industrial-grade paradigms can be successfully transposed to the LEV domain, and to provide rigorous experimental evidence of that transpo-

sition.

## 7.2 Summary of Contributions

The experimental contributions of this thesis were organized into four chapters, each addressing one or more of the identified problems.

Chapter 3 addressed (PS 1) at the communication layer, demonstrating that software disaggregation and containerization principles, well established in industrial IoT deployments, can be applied to LoRaWAN end-devices with negligible overhead. The progressive decomposition of the end-device stack into isolated Docker containers, from the Minimal to the Full architecture, confirmed that flexibility and performance are not in conflict: deeper virtualization reduces latency variability rather than increasing it. The complementary 5G Standalone characterisation established a clear operational boundary between the two radio technologies, with LoRaWAN suited to sporadic low-rate telemetry and 5G to latency-critical broadband services, providing the communication envelope for the entire monitoring framework.

Chapter 4 addressed (PS 1) and (PS 2) at the platform level. The Black Box demonstrated that a modular, containerised edge node can be deployed on a bicycle with an autonomy of up to 8 hours, a GPS validity rate of 96.17%, and a stable IMU acquisition rate of 103 Hz, performance figures comparable to those achieved by industrial edge nodes in factory deployments. The Autoencoder compression scheme and the MST/MPT anomaly detection algorithms then addressed the data management challenge directly on the device, achieving an 80% compression ratio and near-perfect anomaly detection ( $F_1 = 99.84\%$ ) at inference latencies of 445–447  $\mu\text{s}$ , without any reliance on cloud infrastructure.

Chapter 5 addressed (PS 3) by deploying the acquisition framework at scale

during the GiroE-2024 national e-bike tour. The resulting dataset, covering 180.93 hours and 3,749.84 km across 20 stages, is the first large-scale, publicly available biometric dataset collected from non-professional e-bike riders in naturalistic conditions. The graph-based spatial aggregation pipeline demonstrated that asynchronous, multi-device physiological streams can be reliably aligned to road-network nodes with a mean matching distance ranging from 6.19 m to 11.48 m, enabling the kind of cross-rider and cross-stage analysis that would otherwise require costly fixed infrastructure.

Chapter 6 addressed (PS 4) and (PS 5). The fall detection system confirmed that autonomous, low-power safety monitoring is achievable on a bicycle platform: the interrupt-driven architecture sustains an idle current of 12  $\mu\text{A}$ , an estimated battery lifetime exceeding 87 days, and a median alert latency of 221 ms, meeting the responsiveness requirements typical of industrial safety systems. The two charging station studies established that PV-powered LEV infrastructure is both technically viable and environmentally beneficial, reducing lifecycle GHG emissions compared to grid recharging under the Italian energy mix, provided that ICT hardware is dimensioned with explicit awareness of the energy budget shared with the charging load.

## 7.3 General Discussion: A Validated Transposition

Taken together, the five contributions converge on a single overarching conclusion: the problems facing the LEV ecosystem are not fundamentally new. They are instances of challenges that industrial automation has already confronted and, to a large extent, solved. Modular software architectures, on-device

data compression, distributed telemetry fusion, safety-first system design, and energy-aware ICT dimensioning are all established practices in Industry 4.0 environments. What this thesis has demonstrated is that the same approaches, applied with appropriate adaptation to the constraints of mobile, battery-powered, and cost-sensitive platforms, deliver performance that is quantitatively comparable to their industrial counterparts.

This conclusion has an important implication beyond the specific contributions reported here. The technology transfer documented in this thesis has been unidirectional: from industry to mobility. But the direction can be reversed. The algorithms developed for resource-constrained LEV platforms, precisely because they are designed to operate under tighter constraints than typical industrial hardware, may offer advantages when redeployed in industrial contexts where energy efficiency, miniaturisation, or wireless operation are priorities. The MST anomaly detection algorithm, for instance, is validated on a vehicle OBD-II stream, but its incremental, memory-efficient design makes it directly applicable to condition-monitoring on low-power industrial sensors.

This bidirectionality reflects a broader dynamic in the research landscape. The boundary between industrial IoT and consumer or mobility IoT is becoming increasingly permeable, driven by the convergence of hardware platforms, communication standards, and software frameworks. Research investment is following this convergence: Industry 5.0, with its explicit emphasis on human-centricity and sustainability alongside productivity, is precisely the paradigm in which the contributions of this thesis find their most natural home.

## 7.4 Limitations

The results reported in this thesis should be read in light of several limitations, which also delineate the boundaries of the claims made.

First, the experimental validation is centred on the LEV domain and, in several studies, on a single class of vehicle (the e-bike). While the underlying paradigms are deliberately transposed from industrial automation and are therefore general by construction, the quantitative figures (autonomy, GPS validity, anomaly-detection  $F_1$ , alert latency) are tied to the specific hardware, sensor placement, and operating conditions tested. Generalisation to e-scooters, heavier L-category vehicles, or different sensor suites would require dedicated re-characterisation.

Second, the scalability of the proposed architectures was demonstrated at the scale of the deployments actually carried out, most notably the 20-stage GiroE-2024 campaign, but not at the city-wide fleet scale that a production deployment would entail. Aspects such as multi-tenant network contention, large-scale device provisioning, and backend aggregation under thousands of concurrent nodes remain outside the scope of the present validation.

Third, the studies were conducted under controlled or semi-controlled field conditions with prototype hardware. Practical large-scale deployment introduces further challenges, such as long-term hardware robustness, weatherproofing, certification, and maintenance, that were not addressed here and are left to engineering follow-up.

Finally, to support reproducibility, the experimental setups and performance metrics have been described throughout the manuscript, and the GiroE-2024 dataset has been released publicly on Zenodo under a CC BY licence; nevertheless, some results depend on specific firmware versions and environmental conditions that cannot be fully reproduced from the manuscript alone.

## 7.5 Future Work

The contributions of this thesis open several directions for future research.

At the platform level, the Black Box prototype represents a first-generation implementation whose modularity is designed to facilitate extension. A natural next step is the integration of V2X communication capabilities, enabling direct vehicle-to-infrastructure and vehicle-to-vehicle data exchange without relying on centralised backend services. This would extend the scope of (PS 1) from on-board disaggregation to ecosystem-level interoperability.

At the data management level, the Autoencoder compression and MST anomaly detection algorithms were validated on fixed datasets from automotive applications. Extending them to online, continual learning settings, where the model adapts incrementally to distribution shifts in the sensor data without full re-training, would significantly increase their practical applicability and bring them closer to the adaptive behaviour expected of industrial condition-monitoring systems.

At the health monitoring level, the GiroE-2024 dataset opens a range of analytical possibilities that were not fully explored in this thesis. In particular, disentangling the effect of motor assistance from intrinsic physiological response remains an open problem, as does the development of personalised exertion models that account for rider fitness level and route characteristics.

At the infrastructure level, the charging station simulation framework could be extended to incorporate dynamic demand forecasting, integrating real-time weather data, event calendars, and historical usage patterns to optimise energy dispatch. The deployment of the derived sizing guidelines within the MOST national testbeds would provide the field validation that the simulation studies, by design, could not offer.

# Appendix A

## Appendix

### A.1 Siemens Industrial Edge

Siemens Industrial Edge (SIE) is a container-based edge computing platform designed to run containerized applications on dedicated edge devices. As illustrated in Fig. A.1, the platform is organized around three main components:

- **Industrial Edge Hub (IEH):** a cloud-based repository hosting middle-ware and official containerized applications available for deployment.
- **Industrial Edge Management (IEM):** a local (or cloud) management system responsible for configuring and monitoring edge devices and their applications.
- **Industrial Edge Devices (IEDs):** physical devices running the Industrial Edge Runtime, a Debian Linux-based software layer for executing Docker containerized applications (referred to as Industrial Edge Apps). When deployed on a Virtual Machine, the device is designated as a Virtual Industrial Edge Device (vIED).

- **Industrial Edge App Publisher (IEAP):** a software tool for packaging and publishing custom containerized applications to the IEM, enabling their deployment on IEDs.

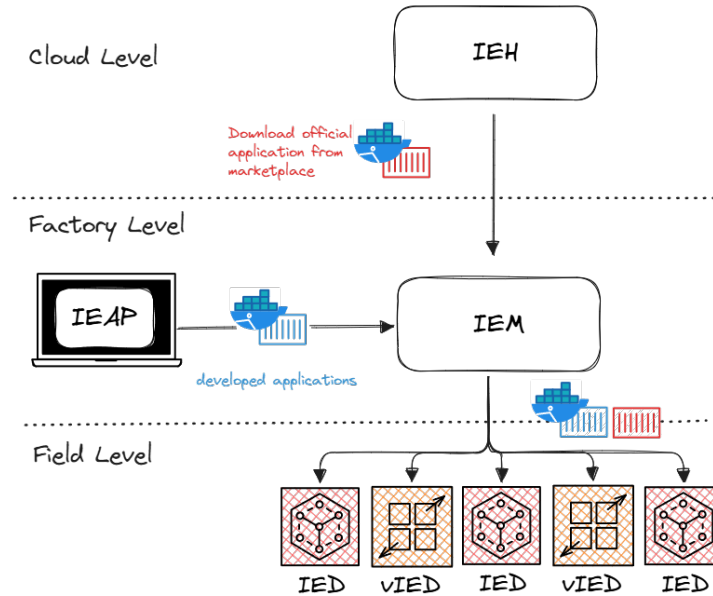


Figure A.1: Siemens Industrial Edge components.

IEDs are connected to both the factory network, for IEM access, and the field network, for data acquisition and control. All traffic is routed through HTTPS proxies to ensure secure communication: inbound traffic is handled by an Nginx reverse proxy, while outbound traffic is managed via a redsocks-based proxy. Direct TCP/IP port exposure is technically supported but generally discouraged; containerized applications typically rely on Docker internal virtual networks instead.

Inter-application communication is facilitated by the Industrial Edge Databus, an MQTT broker whose topics must be pre-configured within the IEM. Proxy

settings are pre-configured by default and can be integrated directly into the application's Docker Compose file.

For custom application development, the platform provides the Industrial Edge Flow Creator, a tailored distribution of the Node-RED visual programming tool, which allows users to build data processing pipelines using JavaScript functions and MQTT connectors linked to the IE Databus. Custom containerized applications can be imported into the ecosystem through the Industrial Edge App Publisher (IEAP).

## A.2 S7Comm Protocol

To enable seamless communication between PLCs and supervisory systems, specific communication protocols are often employed. Siemens developed S7comm, which is the primary communication protocol for M2M, C2C, and SCADA. It is used by Siemens S7-300, S7-400, S7-1200, and S7-1500 families and external devices [25]. The protocol runs on ISO transport services on top of the TCP (TPKT) and all the communications occur on port 102. S7comm data are encapsulated in connection-oriented transport protocol (COTP) packets. The protocol incorporates security mechanisms such as authentication, integrity checks, and confidentiality using encryption algorithms. However, S7comm has faced vulnerabilities and attacks (see [26], [27], [28]). There are three steps to establish a S7 connection with the PLCs [29]: 1) establish a COTP connection by sending a request and receiving the corresponding ACK, 2) S7 communication setup, and 3) exchange of S7 function code GET\_DB instruction, it gets data from the desired data area related to the transaction. In Fig A.2, it is shown an example of the server PLC and assigns them to the data area in the client PLC. The S7 request is handled by the PLC operating system and it allows access to the specified

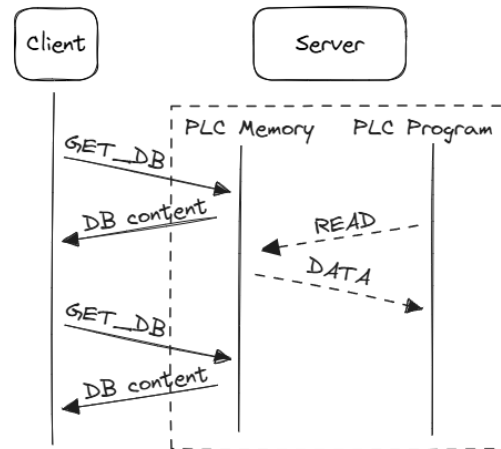


Figure A.2: Sequence diagram of S7comm GET Request.

memory area without disturbing the normal behavior of the PLC program.

[25] (2023). S7comm. Accessed: Jul. 1, 2023. [Online]. Available: [wire-shark.org](http://wire-shark.org) [26] L. Martín-Liras, M. A. Prada, J. J. Fuertes, A. Morán, S. Alonso, and M. Domínguez, “Comparative analysis of the security of configuration protocols for industrial control devices,” *Int. J. Crit. Infrastruct. Protection*, vol. 19, pp. 4–15, Dec. 2017. [27] D. Beresford, “Exploiting Siemens simatic S7 PLCs,” *Black Hat USA*, vol. 16, no. 2, pp. 723–733, 2011. [28] R. Spennenberg, M. Bruggemann, and H. Schwartke, “PLC-blaster: A worm living solely in the PLC,” *Black Hat Asia*, vol. 16, pp. 1–16, Jul. 2016. [29] F. Xiao, E. Chen, and Q. Xu, “S7commtrace: A high interactive honeypot for industrial control system based on S7 protocol,” in *Information and Communications Security*. Beijing, China: Springer, 2018

## A.3 PROFINET Protocol

The PROFINET protocol is standardized under IEC 61158-5-10 and IEC 61158-6-10, and represents one of the most widespread industrial communication solutions based on Ethernet technology. It supports multiple traffic classes hard real-time, soft real-time, and non-real-time (TCP/IP) coexisting on the same physical network. Non real-time communication can use up to 40% of the total bandwidth, ensuring that control and diagnostic data can share the same medium without compromising deterministic communication.

Within a PROFINET network, three main device roles are defined: the IO-Controller, which acts as the automation controller; the IO-Device, representing sensors and actuators; and the IO-Supervisor, typically corresponding to engineering or diagnostic stations. Communication between these entities follows a cyclic exchange mechanism, where each device transmits its process data at regular intervals determined during configuration. In most implementations, cycle timing is managed by dedicated hardware counters, achieving a high degree of temporal precision. Nevertheless, standard Ethernet introduces potential non-deterministic delays that must be mitigated at the protocol level.

To handle different timing requirements, PROFINET defines multiple real-time communication classes as shown in Figure A.3. The RT\_Class 1/2 mode is designed for applications that can tolerate small timing jitter typically within a few communication cycles or a few milliseconds. Conversely, the RT\_Class 3 (Isochronous Real-Time, IRT) mode targets highly synchronized applications requiring sub-millisecond cycle times with jitter below 1  $\mu s$ .

A typical PROFINET communication cycle is divided into distinct transmission phases.

- The Isochronous phase is reserved for IRT communication (RT\_Class 3) and

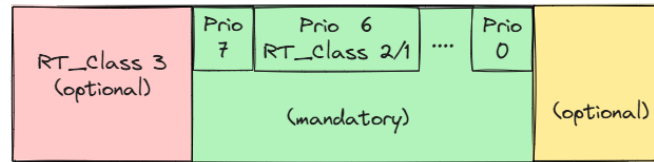


Figure A.3: PROFINET IO cycle (from IEC61158-6-10).

guarantees deterministic data exchange.

- The Standard real-time phase handles RT\_Class 1/2 communication and non-real-time traffic.
- Between these, a Transition phase of variable duration may occur.

Among these, only the standard real-time phase is mandatory. In RT\_Class 3, transmission scheduling is computed deterministically based on the network topology, thus eliminating collisions and delays. This class, however, requires dedicated PROFINET compliant switches and hardware. In contrast, RT\_Class 1/2 leverages standard Ethernet VLAN priority tagging to prioritize PROFINET frames using a best-effort approach, making it the most commonly deployed configuration in industrial automation systems. For instance, the sources of the dataset considered in this paper are all based on PROFINET RT\_Class 1/2.

As shown in Figure A.4, PROFINET supports different topologies:

- The star topology configuration is optimal for deployments within constrained geographical boundaries. This architecture emerges inherently when multiple communication nodes establish connections through a centralized switch. In this configuration, the failure or disconnection of an individual PROFINET node does not compromise the operational integrity of remaining nodes. However, a critical vulnerability exists at the cen-

tral switch: its malfunction results in complete communication disruption across all connected nodes.

- The tree topology represents a hierarchical network architecture formed through the integration of multiple star-configured sub-networks. In industrial implementations, functionally related plant components are consolidated at star points, which are subsequently interconnected via adjacent switches. Within each star point, a designated switch functions as a signal distribution hub. The address-based message routing capability of these switches ensures that only destination-relevant data packets are forwarded to neighboring distribution points, thereby optimizing network efficiency.
- The line topology employs a daisy chain connection, where the communication nodes are arranged sequentially. This architecture presents a significant limitation: switch failure at any point in the chain eliminates communication capability for all downstream nodes. Implementation of a linear structure necessitates the deployment of devices equipped with dual-port switching functionality. Despite this vulnerability, linear network architectures offer the advantage of minimal cabling requirements, making them cost-effective for some types of applications.
- The ring topology provides enhanced redundancy through its closed loop architecture. Implementation of this configuration requires the designation of both a redundancy manager and redundancy clients within the network. This topology offers superior fault tolerance by maintaining alternative communication paths: in the event of a single point failure, data transmission can proceed through the redundant pathway, ensuring continuous network operation. The ring structure effectively addresses the single point

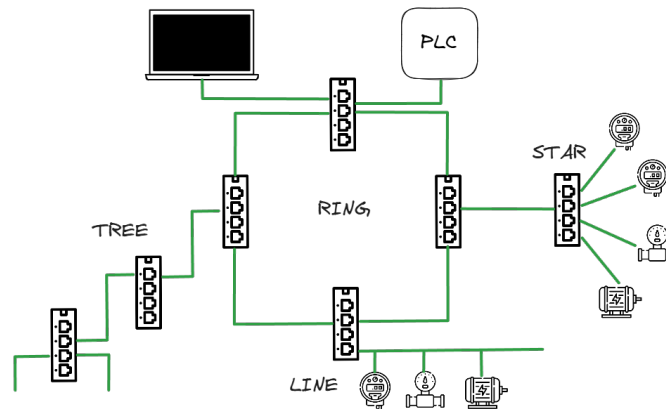


Figure A.4: Profinet Topology

of failure limitations inherent in line and star topologies, though it requires more sophisticated configuration and management protocols.

# Bibliography

- [1] Juliane Anke, Madlen Ringhand, David Schackmann, and Tibor Petzoldt. How e-scooter riders navigate road safety hazards – understanding the perceptions and strategies of regular riders. *Journal of Cycling and Micromobility Research*, 3:100065, 2025. doi: 10.1016/j.jcmr.2025.100065.
- [2] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016. doi: 10.1109/JIOT.2016.2579198. Cited by 12 000; foundational definition of edge computing.
- [3] Weisong Shi and Schahram Dustdar. The promise of edge computing. *IEEE Computer*, 49(5):78–81, 2016. doi: 10.1109/MC.2016.145.
- [4] Brian D. Noble, M. Satyanarayanan, Dushyanth Narayanan, James Eric Tilton, Jason Flinn, and Kevin R. Walker. Agile application-aware adaptation for mobility. *SIGOPS Oper. Syst. Rev.*, 31(5):276–287, oct 1997. ISSN 0163-5980. doi: 10.1145/269005.266708. URL <https://doi.org/10.1145/269005.266708>.
- [5] M. Satyanarayanan. Pervasive computing: vision and challenges. *IEEE Personal Communications*, 8(4):10–17, 2001. doi: 10.1109/98.943998.

- [6] Stephen Soltesz, Herbert Pötzl, Marc E Fiuczynski, Andy Bavier, and Larry Peterson. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. In *Proceedings of the 2Nd ACM SIGOPS/EuroSys european conference on computer systems 2007*, pages 275–287, 2007.
- [7] Bo Li, Qiang He, Feifei Chen, Hai Jin, Yang Xiang, and Yun Yang. Edge-computing-driven internet of things: A survey. *ACM Computing Surveys*, 55(8):1–41, 2022. doi: 10.1145/3555308.
- [8] Stefan Grüner, Johannes Pfrommer, and Florian Palm. Evaluating Docker for lightweight virtualization of distributed and time-sensitive applications in industrial automation. *IEEE Transactions on Industrial Informatics*, 17(3): 2109–2116, 2020. doi: 10.1109/TII.2020.3007803.
- [9] Yu Liu, Dapeng Lan, Zhibo Pang, Magnus Karlsson, and Shaofang Gong. Performance evaluation of containerization in edge-cloud computing stacks for industrial applications: A client perspective. *IEEE Open Journal of the Industrial Electronics Society*, 2:153–168, 2021. doi: 10.1109/OJIES.2021.3055901.
- [10] Franklin Oliveira, Dilan Nery, Daniel G. Costa, Ivanovitch Silva, and Luciana Lima. A survey of technologies and recent developments for sustainable smart cycling. *Sustainability*, 13(6):3422, March 2021. ISSN 2071-1050. doi: 10.3390/su13063422. URL <http://dx.doi.org/10.3390/su13063422>.
- [11] Roberto Morabito and Nicklas Beijar. Enabling data processing at the network edge through lightweight virtualization technologies. In *Proceedings of the IEEE International Conference on Sensing, Communication*

- and Networking Workshops (SECON Workshops)*, pages 1–6, 2016. doi: 10.1109/SECONW.2016.7746807.
- [12] Rui Queiroz, Tiago Cruz, Jérôme Mendes, Pedro Sousa, and Paulo Simões. Container-based virtualization for real-time industrial systems—a systematic review. *ACM Computing Surveys*, 56(3):1–38, October 2023. ISSN 1557-7341. doi: 10.1145/3617591. URL <http://dx.doi.org/10.1145/3617591>.
- [13] E. Sisinni, A. Flammini, M. Gaffurini, and P. Ferrari. Exploiting container-based microservices for reliable smart mobility applications. *ACM SIGAda Ada Letters*, 43(2):52–56, June 2024. ISSN 1094-3641. doi: 10.1145/3672359.3672365. URL <http://dx.doi.org/10.1145/3672359.3672365>.
- [14] Andreas Eckhardt and Sebastian Müller. Analysis of the round trip time of opc ua and tsn based peer-to-peer communication. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 161–167, 2019. doi: 10.1109/ETFA.2019.8869060.
- [15] Theodor Borangiu, Damien Trentesaux, André Thomas, Paulo Leitão, and Jose Barata. Digital transformation of manufacturing through cloud services and resource virtualization. *Computers in Industry*, 108:150–162, 2019. ISSN 0166-3615. doi: <https://doi.org/10.1016/j.compind.2019.01.006>. URL <https://www.sciencedirect.com/science/article/pii/S0166361519300107>.
- [16] Núbia Carvalho, Omar Chaim, Edson Cazarini, and Mateus Gerolamo. Manufacturing in the fourth industrial revolution: A positive

- prospect in sustainable manufacturing. *Procedia Manufacturing*, 21:671–678, 2018. ISSN 2351-9789. doi: <https://doi.org/10.1016/j.promfg.2018.02.170>. URL <https://www.sciencedirect.com/science/article/pii/S2351978918302105>. 15th Global Conference on Sustainable Manufacturing.
- [17] Ricardo P. Pontarolli, Massimiliano Gaffurini, Eduardo P. Godoy, and Paolo Ferrari. Analyzing microservice communication performance in industrial edge platform. *IEEE Access*, 13:167744–167755, 2025. ISSN 2169-3536. doi: 10.1109/access.2025.3612701. URL <http://dx.doi.org/10.1109/ACCESS.2025.3612701>.
- [18] Emiliano Sisinni, Paolo Bellagente, Alessandro Depari, Alessandra Flammini, Massimiliano Gaffurini, Marco Pasetti, Stefano Rinaldi, and Paolo Ferrari. Assessment of time performance of lightweight virtualization for edge computing applications. In *2023 IEEE 19th International Conference on Factory Communication Systems (WFCS)*, page 1–4. IEEE, April 2023. doi: 10.1109/wfcs57264.2023.10144238. URL <http://dx.doi.org/10.1109/wfcs57264.2023.10144238>.
- [19] Massimiliano Gaffurini, Paolo Bellagente, Alessandro Depari, Alessandra Flammini, Dennis Brandão, Stefano Rinaldi, Emiliano Sisinni, and Paolo Ferrari. On computing and real-time communication performance of containerized virtual plcs. In *2024 IEEE 20th International Conference on Factory Communication Systems (WFCS)*, page 1–4. IEEE, April 2024. doi: 10.1109/wfcs60972.2024.10540795. URL <http://dx.doi.org/10.1109/WFCS60972.2024.10540795>.
- [20] Massimiliano Gaffurini, Paolo Bellagente, Alessandro Depari, Alessandra

- Flammini, Emiliano Sisinni, and Paolo Ferrari. Virtual plc in industrial edge platform: Performance evaluation of supervision and control communication. *IEEE Transactions on Instrumentation and Measurement*, 73:1–10, 2024. doi: 10.1109/TIM.2024.3370746.
- [21] Massimiliano Gaffurini, Dennis Brandão, Stefano Rinaldi, Alessandra Flammini, Emiliano Sisinni, and Paolo Ferrari. Characterizing the real-time communication performance of virtual plc in industrial edge platform. *IEEE Open Journal of Instrumentation and Measurement*, 4:1–11, 2025. ISSN 2768-7236. doi: 10.1109/ojim.2025.3559573. URL <http://dx.doi.org/10.1109/OJIM.2025.3559573>.
- [22] Lorenzo Vangelista and Alessandro Cattapan. Start of packet detection and synchronization for lorawan modulated signals. *IEEE Transactions on Wireless Communications*, 21(6):4608–4621, June 2022. ISSN 1558-2248. doi: 10.1109/twc.2021.3131802. URL <http://dx.doi.org/10.1109/twc.2021.3131802>.
- [23] Jorge Ortin, Matteo Cesana, and Alessandro Redondi. Augmenting lorawan performance with listen before talk. *IEEE Transactions on Wireless Communications*, 18(6):3113–3128, June 2019. ISSN 1558-2248. doi: 10.1109/twc.2019.2910512. URL <http://dx.doi.org/10.1109/twc.2019.2910512>.
- [24] Rene Brandborg Sorensen, Dong Min Kim, Jimmy Jessen Nielsen, and Petar Popovski. Analysis of latency and mac-layer performance for class a lorawan. *IEEE Wireless Communications Letters*, 6(5):566–569, October 2017. ISSN 2162-2345. doi: 10.1109/lwc.2017.2716932. URL <http://dx.doi.org/10.1109/lwc.2017.2716932>.

- [25] Nikolay Petkov Manchev, Krasen Kirov Angelov, Panagiotis Gregorios Kogias, and Stanimir Mihailov Sadinov. Development of multichannel lorawan gateway for educational applications in low-power wireless communications. In *2019 IEEE XXVIII International Scientific Conference Electronics (ET)*, page 1–4. IEEE, September 2019. doi: 10.1109/et.2019.8878492. URL <http://dx.doi.org/10.1109/et.2019.8878492>.
- [26] Mohammed Jouhari, Nasir Saeed, Mohamed-Slim Alouini, and El Mehdi Amhoud. A survey on scalable lorawan for massive iot: Recent advances, potentials, and challenges. *IEEE Communications Surveys & Tutorials*, 25(3): 1841–1876, 2023. ISSN 2373-745X. doi: 10.1109/comst.2023.3274934. URL <http://dx.doi.org/10.1109/COMST.2023.3274934>.
- [27] Massimiliano Gaffurini, Enrico Brunelli, Dennis Brandão, Alessandro Depari, Alessandra Flammini, Stefano Rinaldi, Emiliano Sisinni, and Paolo Ferrari. Wide area, iot-based pid control of slow systems over lorawan. In *2025 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0 & IoT)*, page 23–28. IEEE, July 2025. doi: 10.1109/metroind4.0iot66048.2025.11122052. URL <http://dx.doi.org/10.1109/metroind4.0iot66048.2025.11122052>.
- [28] Muhalim Ali Almuahaya, Waheb A. Jabbar, Norashidah Sulaiman, and Saleh Abdulmalek. A survey on LoRaWAN technology: Recent trends, opportunities, simulation tools and future directions. *Electronics*, 11(1):164, 2022. doi: 10.3390/electronics11010164.
- [29] Christos Milarokostas, Dimitrios Tsolkas, Nikos Passas, and Lazaros Merakos. A comprehensive study on LPWANs with a focus on the potential of

- LoRa/LoRaWAN systems. *IEEE Communications Surveys & Tutorials*, 25(1): 825–867, 2023. doi: 10.1109/COMST.2022.3229846.
- [30] Mana Al Mojamed. On the use of LoRaWAN for mobile internet of things: The impact of mobility. *Applied System Innovation*, 5(1):5, 2022. doi: 10.3390/asi5010005. Evaluates LoRaWAN performance for bike rentals and fleet monitoring.
- [31] Massimiliano Gaffurini, Stefano Rinaldi, Emiliano Sisinni, Paolo Ferrari, and Sonia Semendo. Hybrid lora / lorawan networks integration for rural areas communication infrastructure: lesson learned from a real use case. In *2026 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0 & IoT)*. IEEE, 2026. Under review (minor revision).
- [32] Davide Botturi, Alessandro Depari, Paolo Ferrari, Alessandra Flammini, Simone Pasinetti, Matteo Soprani, and Emiliano Sisinni. Loravine: Using lorawan for smart vineyards microclimate monitoring. In *2023 IEEE Sensors Applications Symposium (SAS)*, page 1–6. IEEE, July 2023. doi: 10.1109/sas58821.2023.10254177. URL <http://dx.doi.org/10.1109/SAS58821.2023.10254177>.
- [33] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B. Letaief. Mobile edge computing: Survey and research outlook. *arXiv preprint arXiv:1701.01090*, 2017. URL <https://arxiv.org/abs/1701.01090>. Comprehensive MEC survey; over 2 500 citations.
- [34] Ejaz Ahmed, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Imran Khan, Ahmad Ijaz Ahmad, Muhammad Imran, and Athanasios V. Vasilakos. A survey of multi-access edge computing in 5G and beyond: Fundamentals,

- technology integration, and state-of-the-art. *IEEE Access*, 8:116974–117017, 2020. doi: 10.1109/ACCESS.2020.2993530.
- [35] Gianfranco Nencioni, Rosario G. Garroppo, and Ruxandra F. Olimid. 5G multi-access edge computing: A survey on security, dependability, and performance. *IEEE Access*, 11:68739–68800, 2023. doi: 10.1109/ACCESS.2023.3288334.
- [36] Emiliano Sisinni, Alessandra Flammini, Massimiliano Gaffurini, Marco Pasetti, Stefano Rinaldi, and Paolo Ferrari. Lorawan end device disaggregation and decomposition by means of lightweight virtualization. *Internet of Things*, 25:101033, April 2024. ISSN 2542-6605. doi: 10.1016/j.iot.2023.101033. URL <http://dx.doi.org/10.1016/j.iot.2023.101033>.
- [37] Massimiliano Gaffurini, Alessandra Flammini, Paolo Ferrari, Dhiego Fernandes Carvalho, Eduardo Paciencia Godoy, and Emiliano Sisinni. End-to-end emulation of lorawan architecture and infrastructure in complex smart city scenarios exploiting containers. *Sensors*, 24(7), 2024. ISSN 1424-8220. doi: 10.3390/s24072024. URL <https://www.mdpi.com/1424-8220/24/7/2024>.
- [38] Hossein Khalilnasl, Alessandro Depari, Paolo Ferrari, Alessandra Flammini, Massimiliano Gaffurini, and Emiliano Sisinni. Implementing a software defined lorawan node exploiting container-based lightweight virtualization. In *2024 IEEE International Symposium on Measurements & Networking*, pages 1–6, 2024. doi: 10.1109/MN60932.2024.10615549.
- [39] Enrico Brunelli, Salvatore Dello Iacono, Alessandro Depari, Paolo Ferrari, Alessandra Flammini, Massimiliano Gaffurini, Stefano Rinaldi, and Emil-

- iano Sisinni. A smart campus multi-access communication infrastructure: Performance of the 5g side. In *2025 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0 & IoT)*, pages 17–22, 2025. doi: 10.1109/MetroInd4.0IoT66048.2025.11122105.
- [40] Chia-Yu Chang and Navid Nikaein. Closing in on 5g control apps: Enabling multiservice programmability in a disaggregated radio access network. *IEEE Vehicular Technology Magazine*, 13(4):80–93, December 2018. doi: 10.1109/mvt.2018.2857408. URL <https://doi.org/10.1109/mvt.2018.2857408>.
- [41] Daniele Moro, Giacomo Verticale, and Antonio Capone. Network function decomposition and offloading on heterogeneous networks with programmable data planes. *IEEE Open Journal of the Communications Society*, 2:1874–1885, 2021. doi: 10.1109/ojcoms.2021.3101366. URL <https://doi.org/10.1109/ojcoms.2021.3101366>.
- [42] Docker, Inc. stats\_helpers.go – Docker CLI source file. GitHub. URL <https://github.com/docker/cli/blob/6c12a82f330675d4e2cfff4f8b89a353bcb1fecb/cli/command/container/statshelpers.go#L180>. Accessed: March 13, 2026.
- [43] *netstat(8) – Linux Manual Page*. The Linux man-pages project. URL <https://linux.die.net/man/8/netstat>. Accessed: March 13, 2026.
- [44] Sergio Bemposta Rosende, Sergio Ghisler, Javier Fernandez Andres, and Javier Sánchez-Soriano. Implementation of an edge-computing vision sys-

- tem on reduced-board computers embedded in uavs for intelligent traffic management. *Drones*, 7:682, 11 2023. doi: 10.3390/drones7110682.
- [45] Vlad-Eusebiu Baciú, An Braeken, Laurent Segers, and Bruno da Silva. Secure tiny machine learning on edge devices: A lightweight dual attestation mechanism for machine learning. *Future Internet*, 17(2), 2025. ISSN 1999-5903. doi: 10.3390/fi17020085. URL <https://www.mdpi.com/1999-5903/17/2/85>.
- [46] Matheus Andrade, Miguel Amaral, Morsinaldo Medeiros, Marianne Silva, Ivanovitch Silva, Massimiliano Gaffurini, Dennis Brandão, and Paolo Ferrari. Autoencoders for embedded sensor data compression: A case study on vehicular iot systems. In *2025 IEEE 30th International Conference on Emerging Technologies and Factory Automation (ETFA)*, page 1–8. IEEE, September 2025. doi: 10.1109/etfa65518.2025.11205629. URL <http://dx.doi.org/10.1109/ETFA65518.2025.11205629>.
- [47] Mateus S. Gilbert, Marcello L.R. de Campos, and Miguel Elias M. Campista. Asymmetric autoencoders: An nn alternative for resource-constrained devices in iot networks. *Ad Hoc Networks*, 156:103412, 2024. ISSN 1570-8705. doi: <https://doi.org/10.1016/j.adhoc.2024.103412>. URL <https://www.sciencedirect.com/science/article/pii/S1570870524000234>.
- [48] Tianyue Zhang, Wei Chen, Yuxiao Liu, and Lifa Wu. An intrusion detection method based on stacked sparse autoencoder and improved gaussian mixture model. *Computers & Security*, 128:103144, 2023. ISSN 0167-4048. doi: <https://doi.org/10.1016/j.cose.2023>.

103144. URL <https://www.sciencedirect.com/science/article/pii/S0167404823000548>.
- [49] Mengke Guo, Xiucai Ye, Dong Huang, and Tetsuya Sakurai. Robust feature learning using contractive autoencoders for multi-omics clustering in cancer subtyping. *Methods*, 233:52–60, 2025. ISSN 1046-2023. doi: <https://doi.org/10.1016/j.ymeth.2024.11.013>. URL <https://www.sciencedirect.com/science/article/pii/S1046202324002500>.
- [50] Conect2ai Research Group. Conect2ai – research group on artificial intelligence and machine learning. GitHub. URL <https://conect2ai.dca.ufrn.br/>. Accessed: March 13, 2026.
- [51] S. Li, Z. Wang, X. Wang, Z. Yin, and M. Yao. Frequency-enhanced and decomposed transformer for multivariate time series anomaly detection. *Applied Intelligence*, 55(6):1–18, 2025.
- [52] Morsinaldo Medeiros, Thaís Medeiros, Marianne Silva, Ivanovitch Silva, Massimiliano Gaffurini, Dennis Brandão, and Paolo Ferrari. Mst and mpt: Lightweight incremental algorithms for multivariate anomaly detection and correction on tinyml devices. In *2025 IEEE 30th International Conference on Emerging Technologies and Factory Automation (ETFA)*, page 1–4. IEEE, September 2025. doi: 10.1109/etfa65518.2025.11205746. URL <http://dx.doi.org/10.1109/ETFA65518.2025.11205746>.
- [53] Pedro Andrade, Marianne Silva, Morsinaldo Medeiros, Daniel G. Costa, and Ivanovitch Silva. Teda-rls: A tinyml incremental learning approach for outlier detection and correction. *IEEE Sensors Journal*, 24(22):38165–38173, 2024. doi: 10.1109/JSEN.2024.3458917.

- [54] Plamen Angelov. Anomaly detection based on eccentricity analysis. In *2014 IEEE Symposium on Evolving and Autonomous Learning Systems (EALS)*, pages 1–8, 2014. doi: 10.1109/EALS.2014.7009497.
- [55] Matteo Verzeroli, Luigi Gaioni, Andrea Galliani, Luca Ghislotti, Paolo Lazzaroni, and Valerio Re. Advancing sustainable mobility: A data acquisition system for light vehicles and active mobility. *Electronics*, 13(21), 2024. ISSN 2079-9292. doi: 10.3390/electronics13214249. URL <https://www.mdpi.com/2079-9292/13/21/4249>.
- [56] Stefano Botticini, Elisabetta Comini, Salvatore Dello Iacono, Alessandra Flammini, Luigi Gaioni, Andrea Galliani, Luca Ghislotti, Paolo Lazzaroni, Valerio Re, Emiliano Sisinni, Matteo Verzeroli, and Dario Zappa. Index air quality monitoring for light and active mobility. *Sensors*, 24(10), 2024. ISSN 1424-8220. doi: 10.3390/s24103170. URL <https://www.mdpi.com/1424-8220/24/10/3170>.
- [57] Paolo Bellagente, Dennis Brandão, Salvatore Dello Iacono, Paolo Ferrari, Alessandra Flammini, Massimiliano Gaffurini, Luigi Gaioni, Paolo Malighetti, Stefano Rinaldi, Emiliano Sisinni, and Matteo Verzeroli. Smart-watch data from non-competitive athletes during the giroe-2024 e-bike multi-stage race. *Data in Brief*, 64:112440, February 2026. ISSN 2352-3409. doi: 10.1016/j.dib.2025.112440. URL <http://dx.doi.org/10.1016/j.dib.2025.112440>.
- [58] RCS Sport S.p.A. On the road of history. <https://www.giroe.it/wp-content/uploads/2024/04/Y7VGesFIrq5l6MZZiEkG230424-101510.pdf>, 2024. Accessed: 4 December 2025.
- [59] S. Delloiacono. most-garmin-rawlogger: Garmin ConnectIQ application de-

- signed to log raw sensor data. GitHub. URL <https://github.com/delloiaconos/most-garmin-rawlogger>. Accessed: March 13, 2026.
- [60] Massimiliano Gaffurini, Dennis Brandão, Salvatore Dello Iacono, Alessandra Flammini, Emiliano Sisinni, Paolo Ferrari, and Ivanovitch Silva. Spatial aggregation of iot data in sustainable mobility scenarios. In *2025 Symposium on Internet of Things (SIoT)*, pages 1–4, 2025. doi: 10.1109/SIoT68426.2025.11368796.
- [61] Geoff Boeing. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *CoRR*, abs/1611.01890, 2016. URL <http://arxiv.org/abs/1611.01890>.
- [62] Aric Hagberg, Pieter Swart, and Daniel Chult. Exploring network structure, dynamics, and function using networkx. *Proceedings of the 7th Python in Science Conference*, 06 2008. doi: 10.25080/TCWV9851.
- [63] Paolo Bellagente, Dennis Brandão, Salvatore Dello Iacono, Paolo Ferrari, Alessandra Flammini, Massimiliano Gaffurini, Luigi Gaioni, Paolo Malighetti, Stefano Rinaldi, Emiliano Sisinni, and Matteo Verzeroli. Giroe 2024: Smartwatch data from non-competitive athletes during an e-bike multi-stage race, 2025. URL <https://zenodo.org/doi/10.5281/zenodo.14882222>.
- [64] Salvatore Dello Iacono, Davide Astolfi, Alessandro Depari, Paolo Ferrari, Alessandra Flammini, Massimiliano Gaffurini, Marco Pasetti, and Emiliano Sisinni. Suitability of lorawan and solar harvesting in iot-ready fall detection solutions for light mobility. In *2024 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0 & IoT)*,

- page 210–215. IEEE, May 2024. doi: 10.1109/metroind4.0iot61288.2024.10584218. URL <http://dx.doi.org/10.1109/MetroInd4.0IoT61288.2024.10584218>.
- [65] Seunghye Lee, Bummo Koo, and Youngho Kim. Pre-impact fall detection for e-scooter riding using an IMU: Threshold-based, supervised, and unsupervised approaches. *Applied Sciences*, 14(22):10443, 2024. doi: 10.3390/app142210443.
- [66] Davide Astolfi, Paolo Bellagente, Dennis Brandão, Salvatore Dello Iacono, Alessandro Depari, Paolo Ferrari, Alessandra Flammini, Massimiliano Gaffurini, Marco Pasetti, Stefano Rinaldi, Emiliano Sisinni, and Antony Vasile. Ict-equipped portable e-bike charging station powered by renewables for mass cycling events. In *2024 IEEE International Workshop on Metrology for Automotive (MetroAutomotive)*, page 141–146. IEEE, June 2024. doi: 10.1109/metroautomotive61329.2024.10615741. URL <http://dx.doi.org/10.1109/MetroAutomotive61329.2024.10615741>.
- [67] Marco Pasetti, Antony Vasile, Davide Astolfi, Salvatore Dello Iacono, Emiliano Sisinni, Massimiliano Gaffurini, Paolo Ferrari, and Stefano Rinaldi. Techno-environmental assessment of photovoltaic-powered charging stations for light electric vehicles. In *2025 International Conference on Clean Electrical Power (ICCEP)*, page 948–955. IEEE, June 2025. doi: 10.1109/iccep65222.2025.11143696. URL <http://dx.doi.org/10.1109/ICCEP65222.2025.11143696>.

## Acknowledgements

Banale paragonare questo percorso di dottorato a delle montagne russe, ma la sensazione è proprio quella: "Bella! Ma finalmente è finito il giro". Voglio ringraziare innanzitutto il mio supervisore, il Prof. Paolo Ferrari che definirei una guida in tutto e per tutto. Grazie a lui ho potuto finalmente trovare cosa mi appassiona davvero, e soprattutto come farlo. Fin dal primo giorno, a partire dalla tesi magistrale, è stato un uomo di parola e diretto in ogni occasione, dandomi sempre fiducia e un immenso supporto. Abbiamo molti piani in futuro che ora è il momento di iniziare a concretizzare, e non vedo l'ora di continuare a lavorare insieme (così posso sdebitarmi di tutti i caffè offerti). Un grazie a tutti gli altri membri del gruppo di ricerca ES3, in particolare ad Emiliano, come Paolo persona di riferimento, a Salvatore, mio co-supervisore, sempre pronto a consigliarmi e a darmi una mano. Ai miei colleghi di Open Space, in particolare Martina e Ileana, con cui ho da recuperare molti aperitivi. Avete reso tutto più facile e divertente. Una menzione speciale anche a Lorenzo, mio migliore amico (nonché collega, ex coinquilino, parente, compagno di serate, calcio e praticamente di tutto). Grazie per essere sempre stato lì. Ti auguro i milioni nelle mille startup che lancerai e di cui fai parte.

Massimiliano Gaffurini, Brescia, 04/2026

