

**Grant Agreement number:** 101092696

**Topic:** HORIZON-CL4-2022-DATA-02



# CODECO

Cognitive Decentralised  
Edge Cloud Orchestration

## D20: Use-case Deployment and Demonstrations v2.0

|                                    |  |
|------------------------------------|--|
| <b>Work package</b>                | WP5 – Experimentation Framework, Demonstrations and AI Data Governance and Resilience Testing  |
| <b>Internal Number</b>             | D5.6   |
| <b>Task</b>                        | Task 5.4 – Demonstrations  |
| <b>Due date</b>                    | 28.02.2026   |
| <b>Submission date</b>             | 22.04.2026   |
| <b>Dissemination Type</b>          | Public   |
| <b>Deliverable lead and editor</b> | ALM, Andries Stam  |
| <b>Contributing Partners</b>       | UPRC (Panagiotis Karamolegkos, Efterpi Paraskevoulakou); UGOE (Yanlong Huang, Fabian Wölk); i2CAT (Jordi Marias Parella, Daniel Ulied Guevara); TID (Luis M. Contreras, Alejandro Muñoz); UPM (David Jiménez, Alberto del Río, Javier Serrano); FOR (Rute C. Sofia, Hongyu Zhu); ALM (Andries Stam, Peet van Tooren, Sander Steeghs-Turcina, Kevin Keijzer, Chiel van Diepen, Ludo Stellingwerff); |
| <b>Revision version</b>            | v0.7   |
| <b>Reviewer 1</b>                  | ATH (George Koukis)  |
| <b>Reviewer 2</b>                  | IBM (Luis Garces-Erice)  |



**Funded by  
the European Union**

**Project funded by**



Schweizerische Eidgenossenschaft  
Confédération suisse  
Confederazione Svizzera  
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs,  
Education and Research EAER  
**State Secretariat for Education,  
Research and Innovation SERI**

## Project Partners



## Affiliated Entities



Funded by  
the European Union

## Executive Summary

Deliverable D20 – *Use-case Deployment and Demonstrations v2.0* documents the work carried out under Work Package 5, Tasks 5.3 (Lab Experimentation) and 5.4 (Demonstrations), across the full thirty-eight-month duration of the CODECO project. It presents a comprehensive account of how the CODECO framework was deployed, validated, and demonstrated across six use-cases spanning smart city infrastructure, urban mobility, media delivery, energy management, industrial automation, and smart buildings.

The six pilots — led respectively by the University of Göttingen, i2CAT, Telefónica, the Universidad Politécnica de Madrid, fortiss GmbH, and Almende — each addressed a distinct real-world challenge requiring dynamic orchestration of computational and networking resources across the Edge-Cloud-IoT continuum.

The technical results confirm that CODECO delivers measurable value in each deployment context. In P1, dynamic workload offloading improved LiDAR analytics throughput by 32% under constrained edge conditions, with CODECO responding effectively to pod failures, channel degradation, and latency increases. In P2, CODECO's reinforcement learning-driven orchestration reduced Age of Information by 13.5% and positional tracking Error Distance by 23.6% relative to a centralised baseline at peak vehicular load, while maintaining both deployments within the 3.5-metre lane-width safety threshold. In P3, CODECO's ALTO-based network awareness enabled more efficient placement of media delivery caches and faster adaptation to network disturbances than a static Kubernetes deployment. In P4, the decentralised energy management system achieved the targeted 5–10% improvements in energy efficiency and carbon footprint reduction across three UPM campuses, with fault tolerance validated at the 2-minute recovery target. In P5, CODECO demonstrated reduced CPU consumption, more stable communication patterns, and successful stateful SLAM migration on physical AMR hardware, with all operational KPIs addressed or partially evidenced. In P6, automated deployment and event-driven redeployment — triggered by topology changes and occupancy-derived BLE statistics — demonstrated a class of adaptive smart building management that would be practically unachievable through manual operation.

Cross-pilot analysis of CODECO component engagement reveals that ACM and NetMA were the universal foundation of the framework across all six pilots, while SWM provided workload migration capabilities in five, PDLC provided learning-based orchestration intelligence in four, and MDM provided data-flow observability in three. The variation in component engagement reflects the genuine diversity of the pilots rather than incomplete integration: each partner engaged the subset of CODECO's capabilities that matched their application's operational requirements, and in several cases — P4's energy-domain integration, P6's Virtual Kubelet layer — developed custom bridging components that extended CODECO's reach into previously unsupported device classes and data modalities.

The lessons learned across the experimental phase point to five durable conclusions. Microservice decomposition and CODECO integration must be co-designed from the outset — retrofitting orchestration onto monolithic applications produces limited results. Domain-specific orchestration constraints require domain-specific integration layers that translate operational semantics into CODECO-compatible signals. Orchestration overhead is scale-dependent and disproportionately visible in small environments; evaluation at target scale is essential for a fair assessment. Wireless connectivity remains the dominant performance constraint in mobile edge deployments, and tighter co-design between orchestration and wireless network management represents an important open direction. Environmental context — occupancy, energy prices, traffic density — is an underutilised but powerful source of orchestration triggers, and the pilots that incorporated it produced some of the most operationally compelling results of the project.



Eleven public demonstration events, an open-source software ecosystem, a peer-reviewed publication, and ongoing experiments on physical testbeds collectively constitute the project's dissemination and impact footprint. The work presented in this deliverable confirms that CODECO is technically ready for application across a wide range of Edge-Cloud deployment scenarios, and that the framework's modularity, extensibility, and domain-agnostic design make it a credible foundation for the next generation of cognitive, decentralised container orchestration.

**Keywords:** Use-cases, Deployment, Evaluation, Demonstration.

### Document Revision History

| Version | Date       | Description of change                  | List of contributors                       |
|---------|------------|--|--|
| v0.1    | 03.12.2025 | Initial Proposal of Table of Contents. | Andries Stam (ALM)                         |
| v0.2    | 16.02.2026 | Integration of partner input.          | Andries Stam (ALM), all other contributors |
| v0.3    | 28.02.2026 | Revision of partner input              | Andries Stam (ALM)                         |
| v0.4    | 16.03.2026 | Revision of partner input              | i2CAT, UGOE, TID, UPRC, ALM                |
| v0.5    | 23.03.2026 | Finalization for internal review       | Andries Stam (ALM)                         |
| v0.6    | 30.03.2026 | Revision by internal reviews           | FOR, UGOE                                  |
| v0.7    | 22.04.2026 | Release to the EC                      | Rute C. Sofia (FOR)                        |

### Disclaimer

The information, documentation, and figures available in this deliverable have been developed by the Horizon Europe CODECO project consortium, under the European Union grant Agreement number 101092696. The content does not necessarily reflect the views of the European Commission. The European Commission is not liable for any use that may be made of the information contained herein.

**Copyright notice:** © 2023 – 2026 CODECO Consortium



## Table of Contents

|       |   |    |
|-------|---|----|
| 1     | Introduction .....  | 1  |
| 1.1   | Document Scope .....  | 2  |
| 1.2   | Dependencies.....   | 3  |
| 1.3   | Document Structure.....   | 3  |
| 2     | CODECO Use-Case Portfolio at a Glance .....                     | 3  |
| 2.1   | P1: Smart Monitoring of the Public Infrastructure.....          | 4  |
| 2.2   | P2: Vehicular Digital Twin for Safe Urban Mobility .....        | 4  |
| 2.3   | P3: Media Delivery System across Decentralised Edge-Cloud ..... | 5  |
| 2.4   | P4: Demand Side Management in Decentralised Grids .....         | 6  |
| 2.5   | P5: Wireless AMR Control for Flexible Factories .....           | 6  |
| 2.6   | P6: Smart Buildings .....                                       | 7  |
| 3     | Deployment and Demonstration Methodology.....                   | 7  |
| 4     | P1: Smart Cities .....  | 8  |
| 4.1   | Use-Case Architecture.....                                      | 9  |
| 4.1.1 | Physical / Cluster-level Setup .....                            | 9  |
| 4.1.2 | Internal Service Pipeline .....                                 | 10 |
| 4.1.3 | Application Structure at Pod Level .....                        | 10 |
| 4.2   | Lab Deployment .....  | 11 |
| 4.2.1 | Scenario and Infrastructure.....                                | 12 |
| 4.2.2 | Software Customization .....                                    | 13 |
| 4.3   | Experimentation and Evaluation .....                            | 13 |
| 4.3.1 | Experimentation.....  | 13 |
| 4.3.2 | Evaluation.....   | 14 |
| 4.4   | Demonstration .....   | 16 |
| 4.4.1 | Technical setup.....  | 16 |
| 4.4.2 | Demo Scenarios .....  | 16 |
| 4.5   | Risks and Mitigation Measures .....                             | 18 |
| 4.6   | Summary .....   | 19 |
| 5     | P2: Mobility .....  | 19 |
| 5.1   | Use-Case Architecture.....                                      | 19 |
| 5.1.1 | Key Performance Requirements .....                              | 20 |
| 5.1.2 | Deployment Architecture and CODECO Integration.....             | 20 |
| 5.1.3 | Preconditions, Triggers, and Postconditions .....               | 21 |
| 5.2   | Lab Deployment .....  | 21 |

|       |   |    |
|-------|---|----|
| 5.2.1 | Scenario and Infrastructure.....                  | 22 |
| 5.2.2 | Software Customization .....                      | 22 |
| 5.3   | Experimentation and Evaluation .....              | 23 |
| 5.3.1 | Key Performance Metrics.....                      | 23 |
| 5.3.2 | Experimentation.....                              | 23 |
| 5.4   | Demonstration .....                               | 25 |
| 5.4.1 | Technical setup.....                              | 25 |
| 5.4.2 | Demo Scenario .....                               | 25 |
| 5.4.3 | Events .....                                      | 27 |
| 5.5   | Risks and Mitigation Measures .....               | 27 |
| 5.6   | Summary .....                                     | 27 |
| 6     | P3: MDS across Decentralized Edge-Cloud.....      | 28 |
| 6.1   | Use-Case Architecture.....                        | 28 |
| 6.1.1 | Preconditions and Postconditions .....            | 29 |
| 6.1.2 | CODECO Integration .....                          | 30 |
| 6.2   | Lab Deployment .....                              | 30 |
| 6.3   | Experimentation and Evaluation .....              | 31 |
| 6.3.1 | Evaluation Methodology.....                       | 31 |
| 6.3.2 | Experimentation Results .....                     | 31 |
| 6.4   | Demonstration .....                               | 31 |
| 6.4.1 | Technical setup.....                              | 31 |
| 6.4.2 | Demo Scenarios .....                              | 33 |
| 6.4.3 | Risks and Mitigation Measures .....               | 34 |
| 6.5   | Summary .....                                     | 34 |
| 7     | P4: Demand-side Energy .....                      | 34 |
| 7.1   | Use-Case Architecture.....                        | 34 |
| 7.1.1 | Preconditions, Triggers, and Postconditions ..... | 35 |
| 7.1.2 | CODECO Integration .....                          | 36 |
| 7.1.3 | KPIs.....   | 36 |
| 7.2   | Lab Deployment .....                              | 36 |
| 7.2.1 | Deployment Phases.....                            | 36 |
| 7.2.2 | Software Customization .....                      | 37 |
| 7.3   | Experimentation and Evaluation .....              | 38 |
| 7.3.1 | Evaluation Methodology.....                       | 38 |
| 7.3.2 | Evaluation Metrics and Outcomes .....             | 39 |
| 7.3.3 | Experimentation Results .....                     | 39 |

|       |  |    |
|-------|--|----|
| 7.4   | Demonstration .....                                | 40 |
| 7.4.1 | Technical setup.....                               | 40 |
| 7.4.2 | Demo Scenarios .....                               | 40 |
| 7.4.3 | Demo Event.....                                    | 41 |
| 7.5   | Risks and Mitigation Measures .....                | 41 |
| 7.6   | Summary .....                                      | 41 |
| 8     | P5: AMR and Manufacturing .....                    | 42 |
| 8.1   | Use case Architecture.....                         | 42 |
| 8.1.1 | Preconditions, Triggers, and Postconditions .....  | 43 |
| 8.1.2 | CODECO Integration .....                           | 43 |
| 8.1.3 | KPIs.....  | 44 |
| 8.2   | Lab Deployment .....                               | 44 |
| 8.2.1 | Infrastructure .....                               | 44 |
| 8.2.2 | Software Customization .....                       | 44 |
| 8.3   | Experimentation and Evaluation .....               | 46 |
| 8.3.1 | Evaluation Methodology.....                        | 46 |
| 8.3.2 | Experimentation Results .....                      | 47 |
| 8.3.3 | Evaluation Summary.....                            | 48 |
| 8.4   | Demonstration .....                                | 50 |
| 8.4.1 | Technical Setup.....                               | 50 |
| 8.4.2 | Demo Scenarios .....                               | 50 |
| 8.4.3 | Demo Event.....                                    | 52 |
| 8.4.4 | Risks and Mitigation Measures .....                | 52 |
| 8.5   | Summary .....                                      | 52 |
| 9     | P6: Smart Buildings .....                          | 53 |
| 9.1   | Use Case Architecture.....                         | 53 |
| 9.1.1 | Pre-conditions, Triggers, and Postconditions ..... | 55 |
| 9.1.2 | CODECO Integration .....                           | 55 |
| 9.2   | Lab Deployment .....                               | 56 |
| 9.2.1 | User Stories.....                                  | 56 |
| 9.2.2 | Software Customization .....                       | 56 |
| 9.3   | Experimentation and Evaluation .....               | 56 |
| 9.3.1 | Experimentation Methodology.....                   | 56 |
| 9.3.2 | Experimentation Results .....                      | 57 |
| 9.4   | Demonstration .....                                | 58 |
| 9.4.1 | Technical Setup.....                               | 58 |

|       |  |    |
|-------|--|----|
| 9.4.2 | Events .....   | 60 |
| 9.5   | Risks and Mitigation Measures .....                  | 61 |
| 9.6   | Summary .....  | 61 |
| 10    | Consolidated Results and Conclusions.....            | 62 |
| 10.1  | Cross-functional Coverage of CODECO .....            | 62 |
| 10.2  | Analysis of CODECO Integration Across Use cases..... | 63 |
| 10.3  | Lessons Learned .....                                | 65 |
| 10.4  | Impact .....   | 68 |





## List of Figures

|  |    |
|--|----|
| Figure 1: The CODECO Kubernetes framework and its components. ....   | 2  |
| Figure 2: methodological approach for the CODECO pilot analysis, development, validation. ....                       | 8  |
| Figure 3: P1 – Lab deployment and connectivity. ....   | 9  |
| Figure 4: P1 – Internal pipeline of main pods (ingestion, processing, and collection). ....                          | 10 |
| Figure 5: P1 – Pod deployment across worker nodes (CODECO-managed processing).....                                   | 11 |
| Figure 6: P1 – Intra-Cluster Task Migration.....   | 12 |
| Figure 7: P1 – Inter-Cluster Task Migration.....   | 13 |
| Figure 8: P1 – Deployment time comparison between a Kubernetes-only baseline and the CODECO-enabled deployment. .... | 14 |
| Figure 9: P1 – Effect of workload migration on throughput and end-to-end latency. ....                               | 15 |
| Figure 10: P1 – Effect of workload migration on throughput and end-to-end latency (sample size n = 30). ....         | 15 |
| Figure 11: P1 – CODECO overhead on edge nodes in terms of CPU usage, memory usage, and throughput.....               | 16 |
| Figure 12: P2 – Functional View of the Vehicular Digital Twin Use-Case. ....   | 19 |
| Figure 13: P2 – Deployment Diagram. ....   | 21 |
| Figure 14: P2 – Scenario Facilities.....   | 22 |
| Figure 15: P2 - Infrastructure and actors involved. ....   | 26 |
| Figure 16: P2 - Connections between road users and infrastructure through different RATs. ....                       | 26 |
| Figure 17: P3 – Interaction Scenario between CODECO Client and CODECO Framework. ....                                | 29 |
| Figure 18: P3 lab infrastructure. ....   | 30 |
| Figure 19: P3 - Use case deployment map. ....  | 32 |
| Figure 20: P3 - Use case nodes scheme.....   | 32 |
| Figure 21: P4 infrastructure across the UPM campus. ....   | 35 |
| Figure 22: P4 – Use-Case Architecture.....   | 35 |
| Figure 23: P4 – Energy Measures through ETSIT IoT Infrastructure. ....   | 37 |
| Figure 24: P4 – Kafka Producer. ....   | 38 |
| Figure 25: P4 – Energy Measures Integrated in Prometheus. ....   | 38 |
| Figure 26: P4 - Overview of the demonstration location. ....   | 40 |
| Figure 27: P5 – System Architecture, One Cluster. ....   | 43 |
| Figure 28: P5 – Dashboard for Control of the AMRs. ....  | 46 |
| Figure 29: P5 – Simplified example of the demo environment.....  | 50 |
| Figure 30: P5 – Simplified example of migration offloading due to node/network failure.....                          | 51 |
| Figure 31: P6 – Crownstone Node Inside a Power Outlet. ....  | 54 |

|   |    |
|---|----|
| Figure 32: P6 – Typical Setup of a Crownstone Network. .... | 54 |
| Figure 33: P6 – Technical Setup of Use-Case P6. ....        | 55 |
| Figure 34: P6 - Demonstration location overview. ....       | 58 |

## List of Tables

|   |    |
|---|----|
| Table 1: CODECO-use cases summary.....  | 4  |
| Table 2: P1, demo 1 steps. ....   | 17 |
| Table 3: P1, demo 2 steps. ....   | 17 |
| Table 4: P1, demo 3 steps. ....   | 17 |
| Table 5: P1 - First demo event Overview. ....   | 18 |
| Table 6: P1 - Second Event Overview.....  | 18 |
| Table 7: P1 - Demo risks and mitigation measures. ....                                    | 18 |
| Table 8: P2 – P2 infrastructure, Hardware. ....   | 22 |
| Table 9: P2, Average AoI vs. Number of Connected Vehicles.....                            | 24 |
| Table 10: P2, Average Error Distance (Penalty AoI) vs. Number of Connected Vehicles. .... | 24 |
| Table 11: P2 - V2X Equipment Specs, demo infrastructure.....                              | 25 |
| Table 12: P2 - Event Overview.....  | 27 |
| Table 13: P2 – Demo risks and mitigation measures.....                                    | 27 |
| Table 14: P3 – Metrics Used. ....   | 29 |
| Table 15: P3 - Use case equipment characteristics.....                                    | 32 |
| Table 16: P3 - Demo risks and mitigation measures. ....                                   | 34 |
| Table 17: P4 – Technical KPIs. ....   | 36 |
| Table 18: P4 – Sustainability KPIs. ....  | 36 |
| Table 19: P4 - resilience KPIs. ....  | 36 |
| Table 20: P4 testing scenarios. ....  | 39 |
| Table 21: P4 – Metrics for Comparison. ....   | 39 |
| Table 22: P4 – Outcomes. ....   | 39 |
| Table 23: P4 - Event Overview.....  | 41 |
| Table 24: P4 - Demo risks and mitigation measures. ....                                   | 41 |
| Table 25: P5 – KPIs. ....   | 44 |
| Table 26: P5 – Basic Node Configuration.....  | 44 |
| Table 27: P5 KPI coverage assessment.....   | 48 |
| Table 28: P5 demo camp event. ....  | 52 |
| Table 29: P5 – Demo risks and mitigations. ....   | 52 |
| Table 30: P6, CODECO Deployment Time vs Number of Crownstone Nodes.....                   | 57 |
| Table 31: P6 – Steps in demo scenario A — Automated Application Deployment.....           | 58 |

|  |    |
|--|----|
| Table 32: P6 – Steps in demo scenario B — Topology-Based Redeployment.....   | 59 |
| Table 33: P6 - Steps in demo scenario C — Occupancy-Based Redeployment. .... | 60 |
| Table 34: P6 – First Event Overview. ....                                    | 60 |
| Table 35: P6 – Second Event Overview. ....                                   | 61 |
| Table 36: P6 - Demo risks and mitigation measures. ....                      | 61 |
| Table 37: ACM Functionalities. ....  | 62 |
| Table 38: ACM Functionalities Used .....                                     | 62 |
| Table 39: PDLC Functionalities. ....   | 62 |
| Table 40: PDLC Functionalities Used.....                                     | 62 |
| Table 41: SWM Functionalities.....   | 62 |
| Table 42: SWM Functionalities Used .....                                     | 63 |
| Table 43: NetMA Functionalities. ....  | 63 |
| Table 44:: NetMA Functionalities Used .....                                  | 63 |
| Table 45: MDM Functionalities.....   | 63 |
| Table 46: MDM Functionalities Used.....                                      | 63 |

## List of Acronyms and Definitions

| Acronym       | Meaning                                   |
|---------------|---|
| <b>ACM</b>    | Advanced Configuration and Management     |
| <b>AI</b>     | Artificial Intelligence                   |
| <b>ALM</b>    | Almende                                   |
| <b>ALTO</b>   | Application Layer Transport Optimization  |
| <b>AMR</b>    | Autonomous Mobile Robot                   |
| <b>AoI</b>    | Age of Information                        |
| <b>API</b>    | Application Programming Interface         |
| <b>AR</b>     | Augmented Reality                         |
| <b>AWS</b>    | Amazon Web Services                       |
| <b>B2B</b>    | Business-to-Business                      |
| <b>B2C</b>    | Business-to-Consumer                      |
| <b>BESS</b>   | Battery Energy Storage System             |
| <b>BGP</b>    | Border Gateway Protocol                   |
| <b>BGP-LS</b> | Border Gateway Protocol Link-State        |
| <b>BLE</b>    | Bluetooth Low Energy                      |
| <b>BTP</b>    | Basic Transport Protocol                  |
| <b>CA</b>     | Cooperative Awareness                     |
| <b>CAM</b>    | CODECO Application Model                  |
| <b>CEI</b>    | Cloud-Edge-IoT                            |
| <b>C-ITS</b>  | Cooperative Intelligent Transport Systems |
| <b>CNI</b>    | Container Network Interface               |
| <b>CO2</b>    | Carbon Dioxide                            |
| <b>CPU</b>    | Central Processing Unit                   |
| <b>CR</b>     | Custom Resource                           |
| <b>CRD</b>    | Custom Resource Definition                |
| <b>CV</b>     | Computer Vision                           |
| <b>C-V2X</b>  | Cellular-V2X                              |
| <b>DB</b>     | Database                                  |
| <b>DBMS</b>   | Database Management System                |
| <b>DDS</b>    | Data Distribution Service                 |
| <b>DEN</b>    | Decentralized Environment Notification    |
| <b>DENM</b>   | DEN Messages                              |
| <b>ER</b>     | Extended Reality                          |



|              |  |
|--------------|--|
| <b>ETSIT</b> | Escuela Técnica Superior de Ingenieros de Telecomunicación |
| <b>EV</b>    | Electric Vehicle   |
| <b>FOR</b>   | fortiss  |
| <b>GB</b>    | Gigabyte   |
| <b>GDPR</b>  | General Data Protection Regulation                         |
| <b>GNSS</b>  | Global Navigation Satellite System                         |
| <b>GPS</b>   | Global Positioning System                                  |
| <b>HTTP</b>  | Hypertext Transfer Protocol                                |
| <b>HVAC</b>  | Heating, Ventilation, and Air Conditioning                 |
| <b>ICMP</b>  | Internet Control Message Protocol                          |
| <b>ID</b>    | Identification   |
| <b>IoT</b>   | Internet of Things   |
| <b>IP</b>    | Internet Protocol  |
| <b>JSON</b>  | JavaScript Object Notation                                 |
| <b>K8s</b>   | Kubernetes   |
| <b>KPI</b>   | Key Performance Indicator                                  |
| <b>LTE</b>   | Long Term Evolution  |
| <b>LTS</b>   | Long Term Support  |
| <b>MAC</b>   | Media Access Control                                       |
| <b>MAPEM</b> | Map Extended Message                                       |
| <b>MDM</b>   | Metadata Manager   |
| <b>MDS</b>   | Media Distribution System                                  |
| <b>MEC</b>   | Multi-Access Edge Computing                                |
| <b>MQTT</b>  | Message Queuing Telemetry Transport                        |
| <b>NDN</b>   | Named Data Networking                                      |
| <b>NetMA</b> | Network Management and Adaptation                          |
| <b>OBU</b>   | Onboard Unit   |
| <b>P1</b>    | Pilot 1  |
| <b>P2</b>    | Pilot 2  |
| <b>P3</b>    | Pilot 3  |
| <b>P4</b>    | Pilot 4  |
| <b>P5</b>    | Pilot 5  |
| <b>P6</b>    | Pilot 6  |
| <b>PDLC</b>  | Privacy-preserving Decentralised Learning                  |
| <b>PID</b>   | Prefix Identification                                      |

|               |   |
|---------------|---|
| <b>PV</b>     | Persistent Volume                             |
| <b>PVC</b>    | Persistent Volume Claim                       |
| <b>QoE</b>    | Quality of Experience                         |
| <b>RAM</b>    | Random-Access Memory                          |
| <b>RLT</b>    | Road Lane Topology                            |
| <b>RMSE</b>   | Root Mean Square Error                        |
| <b>ROCOF</b>  | Rate of Change of Failure                     |
| <b>ROS2</b>   | Robotics Operating System v2                  |
| <b>RPi</b>    | Raspberry Pi                                  |
| <b>RRL</b>    | Recurse Requisites List                       |
| <b>RSSI</b>   | Received Signal Strength Indicator            |
| <b>RSU</b>    | Roadside Unit                                 |
| <b>RTT</b>    | Round-Trip Time                               |
| <b>SCM</b>    | Service Configuration Management              |
| <b>SDN</b>    | Software-Defined Networking                   |
| <b>SLAM</b>   | Simultaneous Localization and Mapping         |
| <b>SUMO</b>   | Simulation of Urban Mobility                  |
| <b>SWM</b>    | Scheduling and Workload Migration             |
| <b>TCP</b>    | Transmission Control Protocol                 |
| <b>TEE</b>    | Trusted Execution Environments                |
| <b>TID</b>    | Telefónica Innovación Digital                 |
| <b>UART</b>   | Universal Asynchronous Receiver / Transmitter |
| <b>UC</b>     | Use-Case                                      |
| <b>UGOE</b>   | University of Göttingen                       |
| <b>UI</b>     | User Interface                                |
| <b>UML</b>    | Unified Modeling Language                     |
| <b>UPC</b>    | Universitat Politècnica de Catalunya          |
| <b>UPM</b>    | Polytechnic University of Madrid              |
| <b>UPRC</b>   | University of Piraeus Research Centre         |
| <b>USB</b>    | Universal Serial Bus                          |
| <b>V2X</b>    | Vehicle-to-Everything                         |
| <b>V2XaaS</b> | V2X as a Service                              |
| <b>VAM</b>    | VRU Awareness Message                         |
| <b>VDMA</b>   | Verband Deutscher Maschinen und Anlagenbau    |
| <b>VKs</b>    | Virtual Kubelets                              |

|             |  |
|-------------|--|
| <b>VRU</b>  | Vulnerable Road User                                   |
| <b>WP5</b>  | Work Package 5   |
| <b>WP6</b>  | Work Package 6   |
| <b>WP7</b>  | Work Package 7   |
| <b>YAML</b> | YAML Ain't Markup Language/Yet Another Markup Language |

## Acknowledgements

Many contributors played a crucial role in validating the content and advancing CODECO's Task 5.3 and 5.4 tasks during the development of this deliverable. Gratitude goes to the partners at ATH and FOR for their consistent daily assistance with task 5.3 and the content. The valuable contributions and insights from the partners at ALM and ATH to the Table of Contents are also highly appreciated. Recognition is due to ECL partners for their exceptional support of the CODECO GitLab, which has enabled seamless monitoring of technical progress across all CODECO pilots, efficient resolution of identified issues, and thorough documentation of their solutions. The insights and comments provided by reviewers from ICOM and SIE are acknowledged as well. Appreciation is extended to the development teams of the CODECO components for their timely assistance and dedication to effectively resolving issues, the use case leads for their effort to make the demonstrations a success, and finally to UPRC for providing continuous assistance with the execution of task 5.4.



# 1 Introduction

This deliverable provides a comprehensive overview of the work that was completed between month 6 (M6) and M38 of CODECO's project timeline, with a particular focus on the deployment and demonstration of the six innovative use-cases that have showcased the valuable *Cloud-Edge-IoT (CEI)* orchestration functionalities of the CODECO framework, including functionalities like latency and power efficiency optimization, real-time computation adjustments, as well as flexible and adaptive networking infrastructures from the far Edge to the Cloud. The use-cases were aimed at demonstrating the whole range of CODECO functionalities and features in a wide array of deployment configurations serving the needs of different stakeholders like infrastructure providers and Cloud/Edge application developers. The CODECO containerized application orchestration framework consists of modular micro-services illustrated in Figure 1 to support the following aspects:

- **Automated configuration**, related with application setup and application runtime across Edge-Cloud, by taking into consideration compute, network, and data aspects. Automated configuration is handled by the CODECO Advanced Configuration and Management (**ACM**) component.
- **Data as a resource**. CODECO addresses, via its Metadata Manager (**MDM**) component data as a resource in the sense that available snapshots from the overall Edge-Cloud infrastructure, integrating different perspectives (application, user, system, data, network) at different instants of the CODECO operational workflow can be provided to different CODECO components, to assist in detecting relevant changes.
- **Dynamic scheduling and workload migration** is supported by the CODECO component Scheduling and Workload Migration (**SWM**). SWM integrates a novel concept by Siemens for seamless computing, including a novel scheduler for Kubernetes (K8s) that considers data-network-computation requirements to provide a best match between application requirements and available infrastructure (nodes, their computational and data properties, as well as network nodes and links), and to schedule and re-schedule application workloads across single cluster and federated cluster environments, considering application and user requirements.
- **Context-awareness and privacy preserving decentralised learning**, supported in the component Privacy-preserving Decentralised Learning (**PDLC**). CODECO relies on context-awareness to be able to achieve a joint data-network-compute orchestration, and on privacy-preserving decentralised learning and inference to best support readjustment of aspects such as the processing capability, computational resources, networking resources and interconnections in real-time.
- **Infrastructure adaptation based on a cross-layer data-compute-network approach**. Via the CODECO Network Management and Adaptation component (**NetMA**), CODECO assists in adapting not just computational (node resources) but also the networking infrastructure interconnecting such nodes.



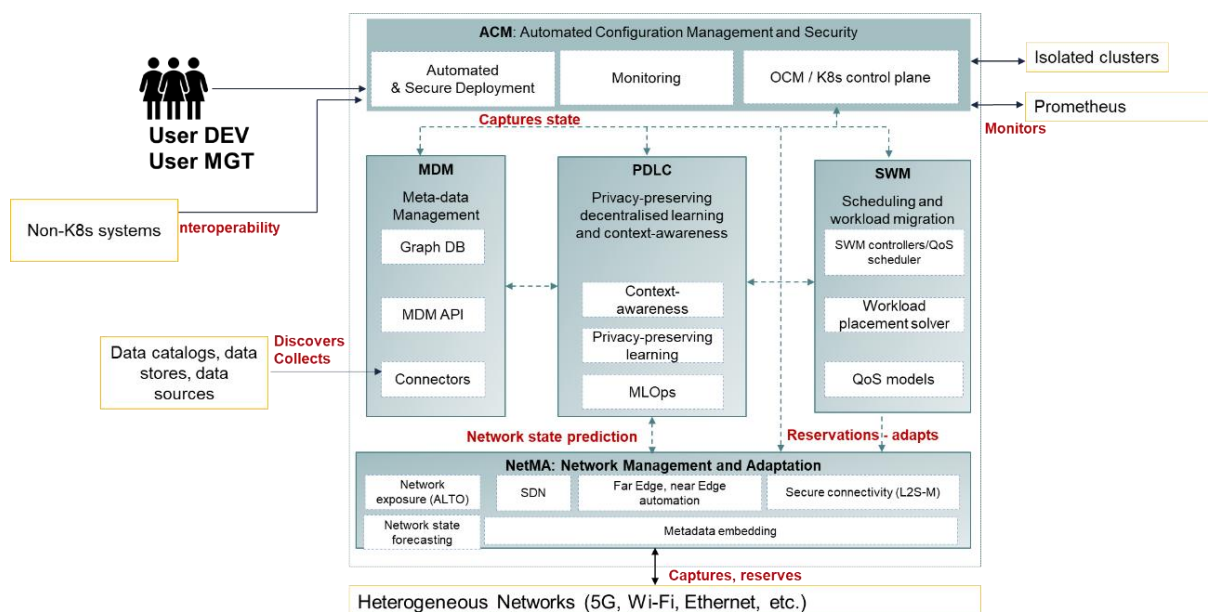


Figure 1: The CODECO Kubernetes framework and its components.

CODECO as a framework supports the setup of applications across clusters (the so-called K8s application deployment) and the cluster runtime management operations for single and multi-cluster environments. Users relying on CODECO during an application deployment are named as **user DEV** in CODECO. Users relying on CODECO during the cluster runtime management are referred to as **user MGT**.

The document commences with a comprehensive summary of all CODECO use-cases. It then allocates specific sections to each pilot, which encompass technical components such as deployment preconditions, triggers for CODECO activation, and postconditions within the operational environment, as well as all aspects related to the technical testing and the set-up and execution of the demonstrations. The CODECO use-cases include:

- **P1: Smart Monitoring of Public Infrastructure**, led by the University of Göttingen (UGOE) and the City of Göttingen.
- **P2: Vehicular Digital Twin for Safe Urban Mobility**, led by i2CAT Foundation (i2CAT).
- **P3: Media Distribution System (MDS) Across Decentralized Edge-Cloud**, led by Telefónica Innovación Digital (TID).
- **P4: Demand-Side Management in Decentralized Grids**, led by the Polytechnic University of Madrid (UPM).
- **P5: Wireless Autonomous Mobile Robot (AMR) Control for Flexible Factories**, led by fortiss (FOR).
- **P6: Smart Buildings**, led by Almende (ALM).

## 1.1 Document Scope

This deliverable documents the work carried out under WP5, Task 5.3 – Lab Experimentation and Task 5.4 – Demonstrations, spanning the full cycle of use-case deployment, experimentation, and demonstration across all six CODECO pilots.

Task 5.3 is concerned with the coordinated deployment of use-cases within the experimental environments of the contributing partners. It encompasses the end-to-end proof-of-concept design for each pilot, covering the selection and configuration of equipment and software, the definition of system actors, the specification of communication protocols, and the design of

operational sequences. In practice, this involved refining the use-case designs first documented in Deliverable D8 into a technically deployable format, a process coordinated by the University of Piraeus Research Centre (UPRC) through biweekly working meetings and ad-hoc discussions with pilot partners throughout the project.

Task 5.4 builds on this foundation by addressing the planning and preparation of demonstration events, the detailed design of demonstration scenarios, and all aspects of their execution and evaluation in front of live audiences. Regular meetings coordinated by Almende (ALM) expanded the technical documentation produced under Task 5.3 into full demonstration plans, covering event logistics, scenario design, stakeholder engagement, and risk management.

For each of the six CODECO pilots, this document provides a structured account of the use-case architecture, the deployment strategies adopted, the lab experimentation carried out and its outcomes, and the design, execution, and evaluation of the associated demonstration. It also explains, in each case, the specific way the CODECO framework was integrated and the functionalities it provided. The document concludes with a consolidated summary of the results produced under Tasks T5.3 and T5.4, offering a unified view of the technical and demonstrative achievements of the project across its experimental phase.

## 1.2 Dependencies

This is a self-contained document. However, we advise the user to read the documents related with the CODECO overall design (D10 [10], D31) and with the CODECO use-cases (D8 [1]).

## 1.3 Document Structure

**Section 1 – Introduction** presents the context and motivation for this deliverable, outlines its scope, and describes the structure of the document.

**Section 2 – Use-Case Overview** provides the reader with the contextual background needed to engage with the detailed pilot sections that follow, offering a concise description of each of the six CODECO use-cases and the domain challenges they address.

**Section 3 – Deployment and Demonstration Methodology** describes the structured information-gathering process and the five instruments used to capture use-case design, deployment, experimentation, and demonstration specifications across all pilot teams.

**Sections 4 through 9 – Pilot Descriptions** each correspond to one CODECO pilot and follow a consistent internal structure throughout. Each section opens with the current architecture of the use-case, including its key components, actors, and operational flow. This is followed by the deployment methodologies adopted in conjunction with the CODECO framework, and a description of the software developed or customised by the use-case partners. The evaluation methodology is then presented, setting out the criteria and metrics used to assess performance. Each section closes with a dedicated account of the lab experimentation carried out and the demonstration executed, together with their respective results and evaluation.

**Section 10 – Consolidated Results and Conclusions** draws together the findings of the deliverable with respect to both lab experimentation and demonstration. It presents a structured overview of the CODECO functionalities employed across all pilots, supported by summary tables, reflects on the lessons learned across the experimental phase, and assesses the technical, dissemination, and community impact of the project.

## 2 CODECO Use-Case Portfolio at a Glance

This section provides the contextual foundation needed to engage with the detailed pilot sections that follow. Each of the six CODECO use-cases addresses a distinct real-world domain — from smart city infrastructure and urban mobility to energy management, media

delivery, industrial automation, and smart buildings — yet all share a common thread: the need to orchestrate computational and networking resources efficiently across Cloud-Edge-IoT (CEI) environments. The descriptions below are intentionally concise, offering enough background to situate each pilot without anticipating the technical depth of the dedicated sections. Table 1 provides a summary overview of the six pilots.

*Table 1: CODECO-use cases summary.*

| Pilot | Title  | Domain             | Lead  | Deployment Environment                        |
|-------|--|--------------------|-------|---|
| P1    | Smart Monitoring of the Public Infrastructure  | Smart Cities       | UGOE  | University of Göttingen Lab                   |
| P2    | Vehicular Digital Twin for Safe Urban Mobility | Mobility           | i2CAT | UPC Campus Nord, Barcelona                    |
| P3    | MDS across Decentralised Edge-Cloud            | Smart Cities/Media | TID   | Telefónica Labs, Madrid, and ICOM Lab, Athens |
| P4    | Demand Side Management in Decentralised Grids  | Energy             | UPM   | UPM Campus, Madrid                            |
| P5    | Wireless AMR Control for Flexible Factories    | Manufacturing      | FOR   | fortiss IIoT Lab, Munich                      |
| P6    | Smart Buildings                                | Smart Buildings    | ALM   | ALM Office, Rotterdam                         |

## 2.1 P1: Smart Monitoring of the Public Infrastructure

**Context and motivation** Cities are under increasing pressure to manage growing volumes of urban traffic while ensuring pedestrian safety and generating actionable insights for planning. LiDAR-based monitoring systems produce dense, variable point-cloud data streams that fluctuate sharply across the day, overwhelming edge devices during peak periods and making real-time analytics difficult to sustain without dynamic resource management.

**Objectives** P1 set out to deploy a real-time traffic and pedestrian analytics system at the far Edge, redistributing compute workloads dynamically in response to load and network conditions. The pilot aimed to optimise traffic flow, reduce congestion, and enhance pedestrian safety, while validating CODECO's capacity to orchestrate resources across decentralised edge environments in real time.

**Key stakeholders and partners** The pilot was led by the University of Göttingen (UGOE) in collaboration with the City of Göttingen. Key stakeholders included municipal transport and planning authorities, infrastructure providers, and citizens as the ultimate beneficiaries of improved traffic management and public safety services.

**High-level architecture and setup** The deployment targets a multi-edge, multi-cluster configuration in Göttingen, where each monitored intersection is treated as an independent Kubernetes cluster comprising NVIDIA Jetson GPU nodes. Validated work to date has focused on a controlled lab deployment capturing the main orchestration challenges of the target system. All data is processed at the Edge prior to storage, supporting privacy-by-design and GDPR compliance.

**Link to CODECO components** CODECO's SWM and PDLC components were central to this pilot, enabling dynamic remapping of compute-intensive inference workloads across nodes and clusters in response to load spikes, channel degradation, or node failures. NetMA provided real-time network monitoring to inform migration decisions.

## 2.2 P2: Vehicular Digital Twin for Safe Urban Mobility

**Context and motivation** Vulnerable Road Users (VRUs) — pedestrians, cyclists, and motorcyclists — face disproportionate safety risks in urban traffic. Existing safety systems are predominantly reactive, lacking the real-time situational awareness needed to intervene before

incidents occur. The Campus Nord of the UPC<sup>1</sup> in Barcelona provided a realistic and diverse urban mobility environment in which to address this challenge.

**Objectives** P2 aimed to develop and validate a Vehicular Digital Twin capable of tracking all moving entities in real time, detecting dangerous situations, and issuing timely alerts to VRUs and vehicle operators. A central objective was to demonstrate the feasibility of ultra-reliable, low-latency Edge services for safety-critical mobility applications, with end-to-end latency targets well below 20 milliseconds.

**Key stakeholders and partners** The pilot was led by i2CAT. Key stakeholders included municipal authorities and mobility planners, mobile communications and cloud providers, academic researchers, and early adopters — pedestrians and vehicle operators — who participated in live testing.

**High-level architecture and setup** The pilot deployed V2X Roadside Units and cameras across the UPC campus to capture real-time positional data on all road users, feeding a continuously updated Digital Twin that generated collision alerts delivered via on-board units and smartphones. Processing was performed as close as possible to the V2X nodes to minimise latency.

**Link to CODECO components** CODECO's ACM, SWM, and PDLC components managed the deployment, scaling, and dynamic migration of the Digital Twin's microservices across the edge-cloud continuum. PDLC's reinforcement learning agent used Age of Information as its optimisation objective, guiding SWM to proactively redistribute workloads before bottlenecks could form.

## 2.3 P3: Media Delivery System across Decentralised Edge-Cloud

**Context and motivation** The rapidly growing demand for video streaming, gaming, and AR/XR content is straining telecommunications infrastructure. A key structural obstacle is the traditional separation between network management and computational orchestration, which prevents the joint optimisation needed to serve users efficiently and cost-effectively.

**Objectives** P3 set out to demonstrate how a Media Delivery System (MDS) could leverage tighter integration of networking and computational resources across a multi-domain, multi-cluster Edge-Cloud environment. The pilot aimed to improve Quality of Experience (QoE) for end users, reduce delivery costs for service providers, and validate a decentralised approach to real-time content source selection.

**Key stakeholders and partners** The pilot was led by Telefónica Innovación Digital (TID). The three principal actors were the application-service provider, responsible for content delivery decisions; the network operator, responsible for exposing network topology and cost information; and the application-service client, the end user consuming the media content.

**High-level architecture and setup** The pilot was deployed on a virtualised Kubernetes environment spanning Madrid and Athens. The network operator published a live representation of network topology using a protocol based on the IETF ALTO standard, which the MDS queried to identify the most suitable content source for each request, considering both network path characteristics and computational availability.

**Link to CODECO components** The NetMA component was central to this pilot, providing a decentralised ALTO implementation that exposed network capabilities and topological information to the MDS in real time. This enabled joint adaptation of networking and compute allocation — a capability not available in standard Kubernetes deployments.

---

<sup>1</sup> <https://www.upc.edu/campusnord/en>

## 2.4 P4: Demand Side Management in Decentralised Grids

**Context and motivation** The urgent need to decarbonise the built environment is driving interest in decentralised energy management systems capable of integrating renewable sources and enabling active demand response. University campuses represent both a significant operational challenge and a compelling opportunity in this context. UPM's commitment to climate neutrality by 2030 provided the motivating framework for this pilot.

**Objectives** P4 aimed to implement a decentralised demand-response management system across university campuses, optimising energy usage, integrating renewable generation, and enabling participation in energy flexibility markets. The pilot also set out to demonstrate CODECO's ability to jointly orchestrate computational and networking resources across the IoT-Edge-Cloud continuum.

**Key stakeholders and partners** The pilot was led by the Universidad Politécnica de Madrid (UPM), acting as both technology deployer and primary energy community. Additional stakeholders included energy service companies, grid operators, technology providers, and EV users seeking smart charging services.

**High-level architecture and setup** The system was deployed across three UPM campuses, each forming a distinct cluster. IoT sensors collected real-time generation and consumption data; edge nodes performed local forecasting and scheduling; cloud resources handled higher-level coordination. Over 100 energy assets were grouped into dynamic clusters, adjusted up to 20 times per day.

**Link to CODECO components** ACM and PDLC were central to this pilot, supporting resource deployment, scalability, and energy demand forecasting across the distributed infrastructure. MDM aggregated real-time sensor data, while NetMA ensured low-latency communication between nodes for timely decision-making.

## 2.5 P5: Wireless AMR Control for Flexible Factories

**Context and motivation** Modern manufacturing environments demand AMR fleets capable of operating flexibly across dynamic, wireless-connected spaces. Current systems rely on pre-defined paths and centralised control that break down under interference and connectivity variability, creating a pressing need for adaptive, decentralised control architectures.

**Objectives** P5 set out to demonstrate that CODECO could provide the orchestration layer for decentralised, resilient AMR fleet control in a flexible factory setting, improving task completion times, reducing collision rates, enhancing energy efficiency, and maintaining availability under wireless disruption. The pilot also validated CODECO's scalability from a single static cluster to a federated multi-cluster architecture.

**Key stakeholders and partners** The pilot was led by fortiss (FOR) and deployed in the fortiss IIoT Labs in Munich. Key stakeholders included factory operators and logistics managers, mobile communications and Edge-Cloud providers, and software developers deploying and managing AMR applications.

**High-level architecture and setup** The pilot progressed through three phases: a single cluster with a static control plane, a single cluster with a mobile control plane, and a federated multi-cluster configuration. Each AMR was treated as a ROS2 Kubernetes worker node running containerised microservices, with compute-intensive tasks such as SLAM offloaded to the controller node and stateful migration supported via persistent storage.

**Link to CODECO components** CODECO's PDLC provided context-aware analysis of infrastructure robustness, energy consumption, and channel conditions, issuing



recommendations to SWM for workload reallocation. NetMA managed wireless path optimisation and interference mitigation, while ACM handled the automated deployment and reconfiguration of AMR application microservices.

## 2.6 P6: Smart Buildings

**Context and motivation** The management of smart building infrastructure is becoming increasingly complex as the number of third-party applications sharing the same physical infrastructure grows. Without robust orchestration, this leads to manual configuration overhead and poor responsiveness to changes in the building environment. ALM's Crownstone platform presented both the opportunity and the technical challenge that motivated this pilot.

**Objectives** P6 set out to demonstrate how CODECO could automate the deployment, monitoring, and dynamic redeployment of applications across a Crownstone smart building network, eliminating manual intervention in response to topology changes or occupancy shifts, while preserving occupant privacy and always ensuring baseline infrastructure functionality.

**Key stakeholders and partners** The pilot was led by Almende (ALM) and deployed in the company's main office in Rotterdam. The primary end user was the facility manager, responsible for defining service levels and managing the building application portfolio. Third-party application developers also featured as stakeholders, as the platform supports multi-party deployment on shared infrastructure.

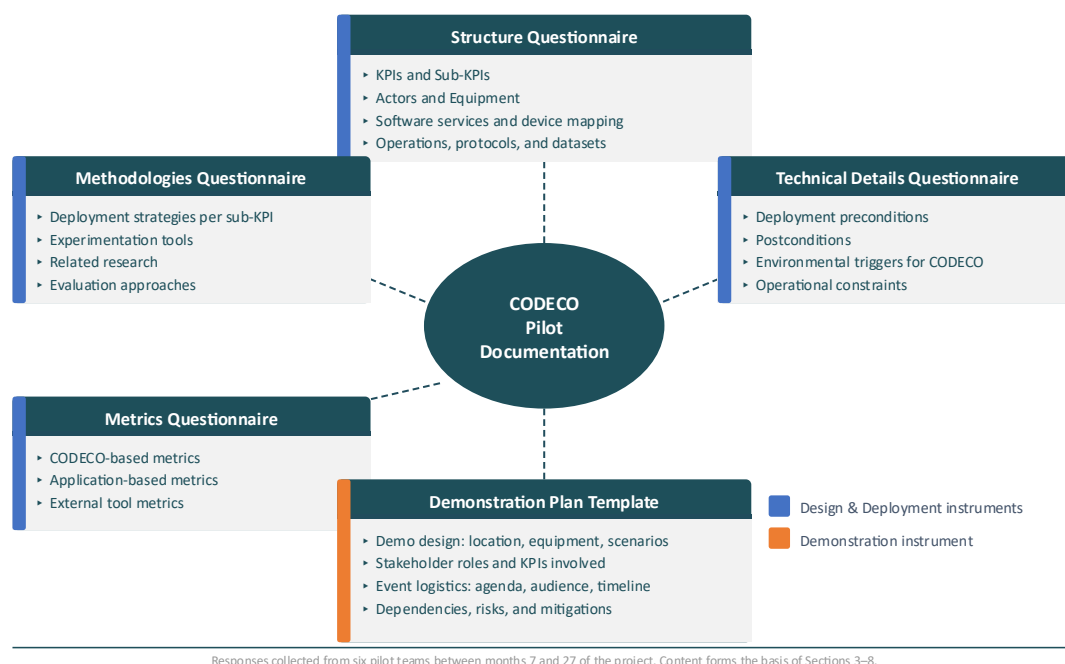
**High-level architecture and setup** The deployment comprised three office rooms with two Crownstone Hubs — Raspberry Pi 5 devices acting as Kubernetes worker nodes — and between twelve and twenty Crownstone nodes mounted inside power outlets. Non-IP Crownstone devices were integrated into Kubernetes via Virtual Kubelets, enabling CODECO to treat them as standard worker nodes. A cloud-based service handled sphere administration and inter-device coordination.

**Link to CODECO components** CODECO's ACM automated initial deployment and ongoing desired-state management across the Crownstone network. NetMA processed BLE network statistics to detect topology changes and occupancy shifts, triggering SWM to redeploy microapps accordingly — enabling fully automated, event-driven building management without human intervention.

## 3 Deployment and Demonstration Methodology

The technical documentation of each use-case presented in Sections 4–9 is grounded in a structured information-gathering process conducted across all six use-cases teams between months seven and twenty-seven of the project. Each team responded to a common set of questionnaires and templates, designed to capture every aspect of use-case design, deployment, experimentation, and demonstration in a format that could be directly translated into deployable specifications. The responses were reviewed, consolidated, and restructured by the task coordinators before forming the basis of the pilot documentation that follows.

The process was organised around five instruments, illustrated in Figure 2, each addressing a distinct layer of the deployment and demonstration cycle.



*Figure 2: methodological approach for the CODECO pilot analysis, development, validation.*

The **Structure Questionnaire** established the technical anatomy of each use-case. It covered the primary and decomposed KPIs, the actors involved and their relationship to each KPI, the hardware required and its minimum and maximum quantities, the software services to be deployed and the devices on which they would run, the inter-service communication flows supported by UML diagrams, and the datasets used for training or validation.

The **Methodologies Questionnaire** examined how each partner intended to achieve their KPIs in practice. It addressed deployment strategies for CODECO integration, the experimentation tools to be used, relevant related research, and the approaches taken for evaluation, covering both the procedure for integrating CODECO into each pilot and the methods for assessing sub-KPI outcomes.

The **Metrics Questionnaire** defined the measurable signals needed to operate and assess each use-case, distinguishing between metrics collected directly from CODECO components and those gathered from external tools or the application itself.

The **Technical Details Questionnaire** specified the preconditions that must be satisfied before deployment, the postconditions expected upon successful execution, and the environmental triggers that initiate CODECO's adaptive behaviour during operation.

The **Demonstration Plan Template** translated the technical work into structured demonstration events. It covered the general technical setup of each demonstration — including location, equipment, and applications — the detailed design of demonstration scenarios with step-by-step descriptions, stakeholder roles, and KPIs involved, the organisational logistics of each event including agenda, audience, duration, and timeline, and a structured assessment of external dependencies, risks, and their mitigations.

## 4 P1: Smart Cities

LiDAR is playing an increasingly important role in modern smart cities, particularly within intelligent transportation systems [2]. However, the density and volume of point cloud data generated by LiDAR sensors varies significantly across the day, driven by fluctuations in traffic

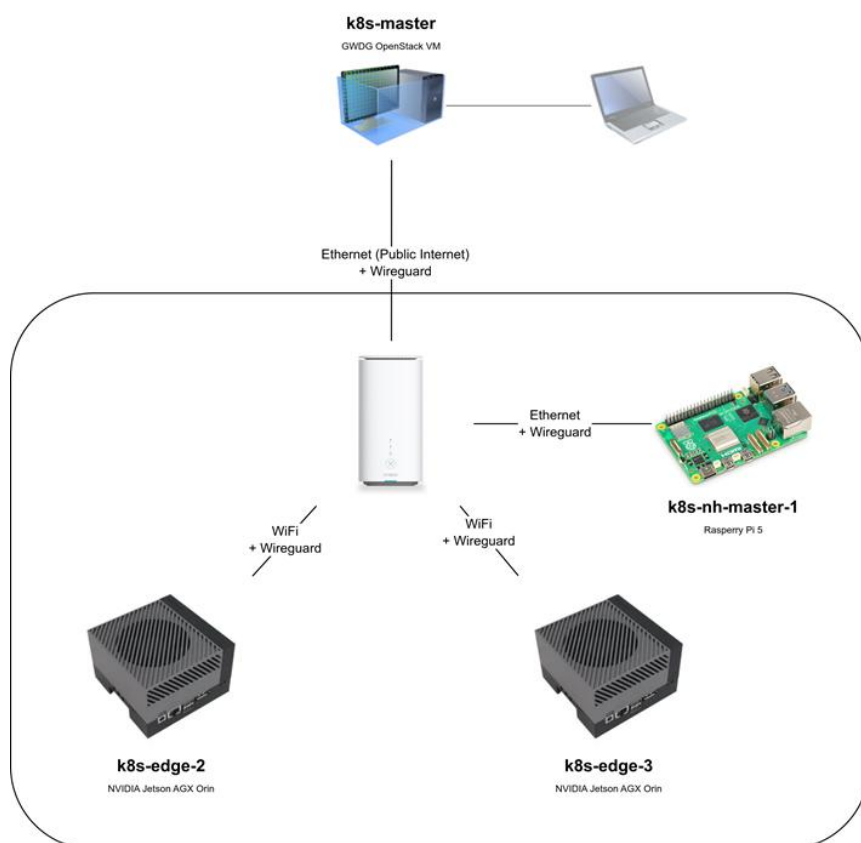
patterns. During periods of heavy traffic, the volume of data and the number of detected objects can rise sharply, making timely traffic analysis difficult to sustain — especially on resource-constrained edge devices. This use-case addresses that challenge by examining how analytics workloads can be dynamically migrated between nodes and clusters using the CODECO framework, ensuring continuous and timely processing regardless of traffic load.

## 4.1 Use-Case Architecture

### 4.1.1 Physical / Cluster-level Setup

As shown in Figure 3, the use case is deployed as a Kubernetes-based edge cluster that mirrors the intended roadside structure: a lightweight control plane plus GPU-capable worker nodes for LiDAR analytics.

- A k8s-master (k8s-nh-master-1, Raspberry Pi 5) hosts the Kubernetes control plane and provides central cluster management.
- A local network hub/router provides connectivity between all nodes; communication is secured using WireGuard, with links established over Ethernet and/or WiFi depending on the device.
- Two GPU edge nodes (k8s-edge-2 and k8s-edge-3, NVIDIA Jetson AGX Orin) act as the primary compute units for real-time traffic analytics at the edge (worker nodes).



*Figure 3: P1 – Lab deployment and connectivity.*

This setup reflects the core deployment assumption of P1: LiDAR data is processed at the far edge on embedded GPU nodes, while orchestration and management functions are provided via Kubernetes (and enhanced via CODECO).



## 4.1.2 Internal Service Pipeline

The detailed pipeline is decomposed into three cooperating pods, as illustrated in Figure 4.

### 1) Data Migration Pod (LiDAR ingestion and forwarding).

This pod interfaces with the LiDAR feed and prepares data for processing:

- A ROS LiDAR node ingests sensor data.
- A ROS master node enables internal ROS communication.
- A subscriber consumes ROS messages and buffers them in a queue.
- The pipeline includes conversion and optional compression (shown as “Convert & Compress with Draco”) before transmission.
- Data is forwarded to processing via a bidirectional WebSocket connection.

### 2) Processing Pod (GPU-based inference).

This pod performs the core analytics:

- It receives LiDAR data via WebSocket and buffers it.
- It performs decompression and conversion (shown as “Decompress & Convert to KITTI”) before inference.
- A PointPillars model is executed using TensorRT and CUDA on the Jetson GPU.
- Outputs (e.g., detected bounding boxes) are queued for downstream consumption.

### 3) Collector Pod (results ingestion and storage).

This pod provides stable output handling:

- A lightweight HTTP Flask app ingests the produced results.
- Results are stored in Postgres for downstream analytics, dashboards, and reporting.

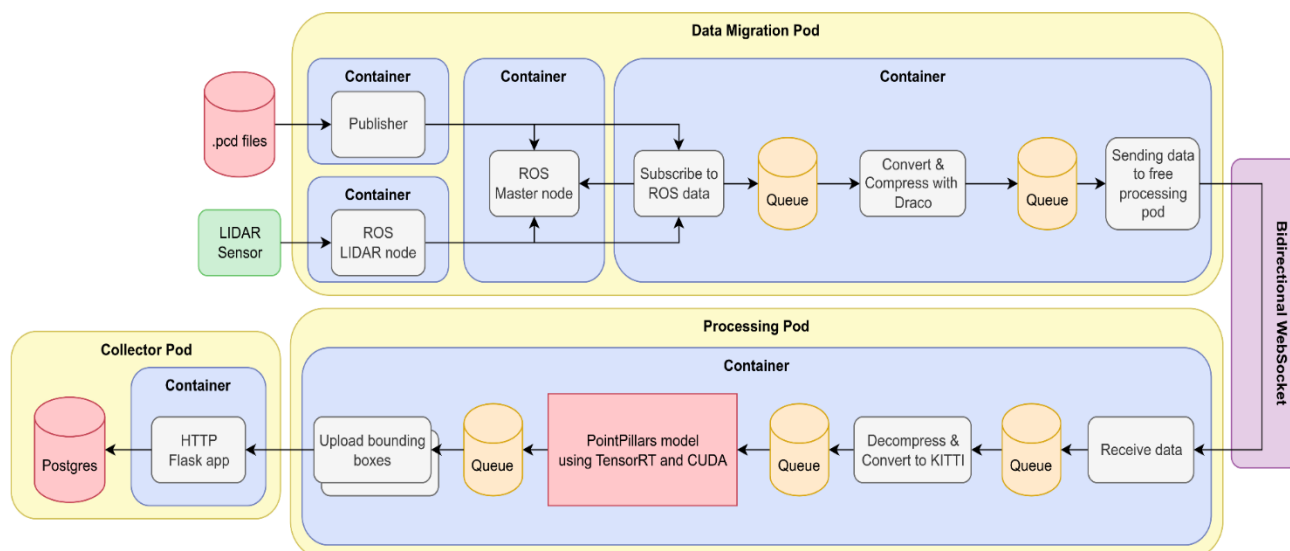


Figure 4: P1 – Internal pipeline of main pods (ingestion, processing, and collection).

## 4.1.3 Application Structure at Pod Level

At the application level, the deployment is organised around two main functional roles running on each worker node: (i) data ingestion/forwarding and (ii) GPU-based processing. The pod-level design is shown in Figure 5.

- A Master node manages the cluster.
- Two Worker Nodes receive LiDAR data streams (one LiDAR feed per node is depicted for simplicity).
- Each worker runs a Data Migration Pod, which ingests LiDAR data locally and can forward it to processing components.
- Each worker also runs multiple Processing Pods, which execute the LiDAR analytics workload (inference).

Importantly, Figure 5 highlights that the Processing Pods are managed by CODECO, meaning CODECO can (re)allocate processing workloads across nodes to support operational objectives (e.g., resilience and maintaining timeliness). The diagram also shows GPU time slicing, indicating that multiple processing pods can share GPU resources on each Jetson node, enabling parallelism and redundancy.

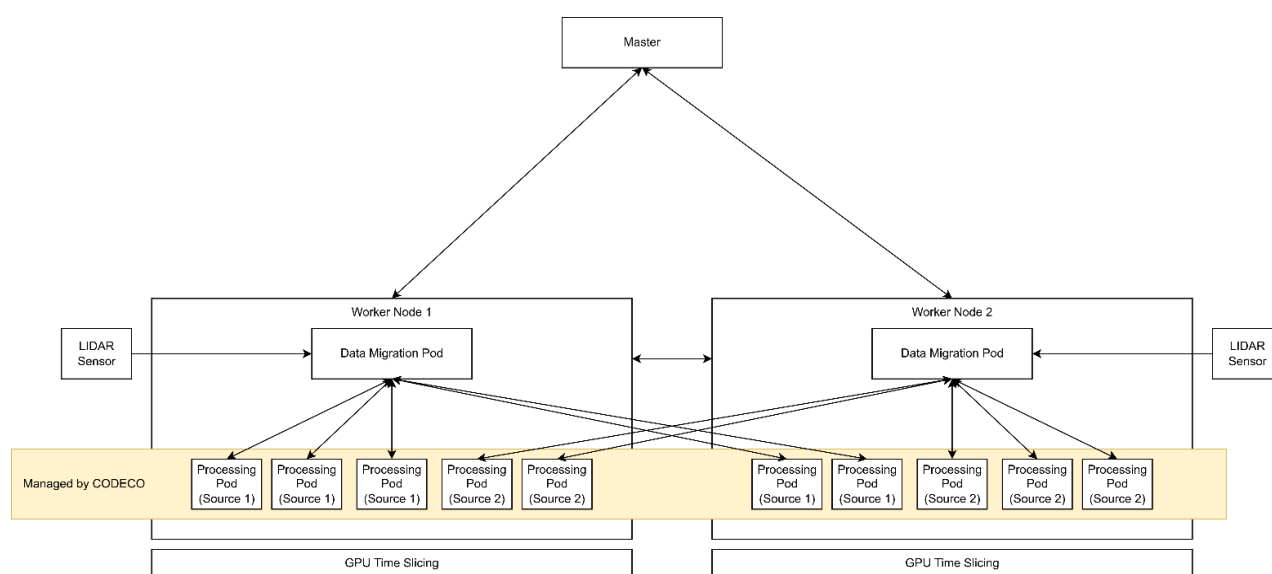


Figure 5: P1 – Pod deployment across worker nodes (CODECO-managed processing).

## 4.2 Lab Deployment

The executed validation of P1 was carried out **as a controlled lab deployment rather than a full city-scale rollout**. A Kubernetes cluster was configured on two NVIDIA Jetson AGX Orin devices acting as resource-constrained edge worker nodes, connected over WiFi, with communication between the control plane and the edge devices supported through a 5G router. All network communication was protected with WireGuard. Since the real traffic monitoring system is not yet deployed in Göttingen, LiDAR input was emulated by streaming point-cloud data from the TUMTraf A9 dataset<sup>2</sup> into the edge devices, allowing repeatable and controlled validation of the use-case pipeline together with CODECO.

The software was implemented as a Kubernetes-native microservice pipeline rather than a monolithic application, a design choice driven not only by modularity but by the need to expose the right control points for runtime orchestration by CODECO. The three pod types described in section 4.1.2 form the core of this pipeline. The processing pipeline is implemented in C++ with CUDA and TensorRT to achieve high-throughput, low-latency inference on Jetson-class

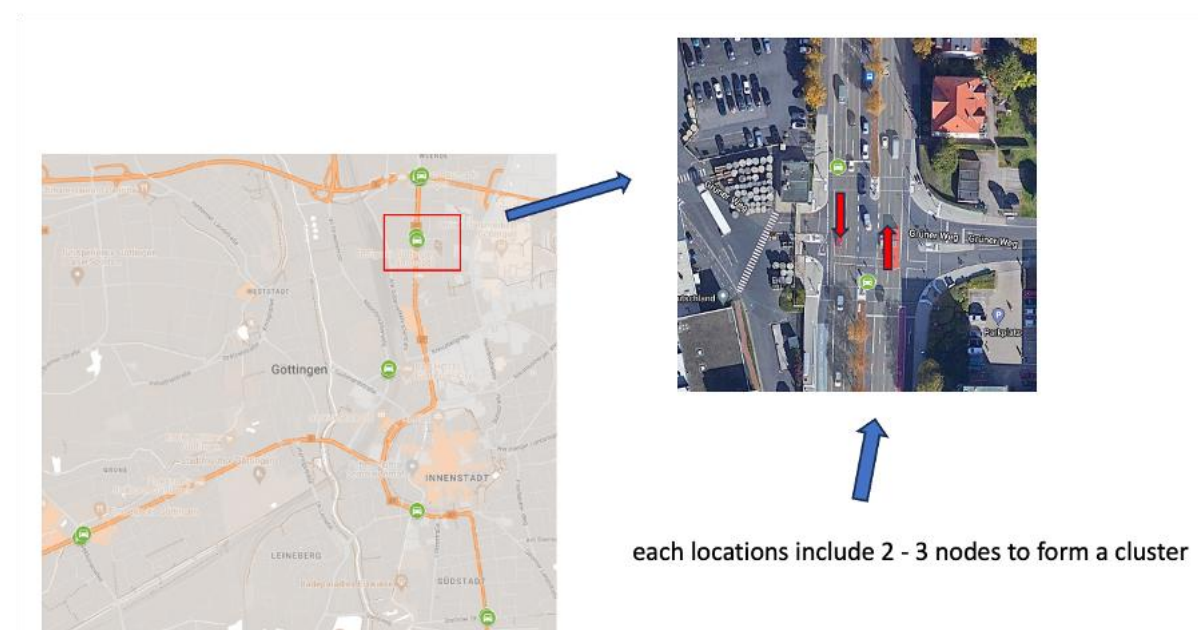
<sup>2</sup> <https://innovation-mobility.com/en/project-providentia/a9-dataset/>

embedded GPUs, with Draco used for point-cloud compression to reduce transmission cost over wireless links.

The key contribution of the software customisation is not the individual tools used in isolation, but the way the pipeline is structured to enable orchestration. **By keeping DMPs fixed to their sensing sources while treating PPs as movable workloads, CODECO can perform runtime remapping and resilience-oriented migration in response to pod failures, channel instability, or latency degradation** — rather than relying on static placement. The P1 software customisation should therefore be understood as the co-design of the LiDAR analytics pipeline and its CODECO integration.

## 4.2.1 Scenario and Infrastructure

**Intra-cluster task migration:** In the target field deployment, each monitored traffic area in Göttingen is treated as one Kubernetes cluster (Figure 6). Each cluster is intended to comprise two to three LiDAR-edge nodes, each covering a different traffic direction. Under this deployment logic, intra-cluster task migration addresses situations where one direction generates substantially more point-cloud data than another, for example during heavy traffic peaks or temporary local congestion. In such cases, CODECO can remap part of the processing workload from the overloaded node to another worker node within the same cluster, thereby reducing backlog and helping to maintain timely analytics. This corresponds to the local load-balancing function of the P1 deployment, where sensing remains tied to the source node while compute-intensive processing can be shifted according to available resources and current link conditions.



*Figure 6: P1 – Intra-Cluster Task Migration.*

**Inter-cluster task migration:** Beyond local load balancing, P1 also targets inter-cluster task migration across monitored traffic areas. This case becomes relevant when the overload cannot be absorbed within one cluster, for instance when multiple directions at one intersection simultaneously generate high data rates or when local node/channel disturbances degrade processing performance. In such situations, CODECO enables workload remapping towards nodes in another cluster with more favourable resource or network conditions. Therefore, the inter-cluster case is not merely a scale extension of the intra-cluster case, but a resilience-oriented deployment logic for preserving service continuity across the broader multi-edge smart-city system. The inter-cluster case can be viewed in Figure 7.

## 4.2.2 Software Customization

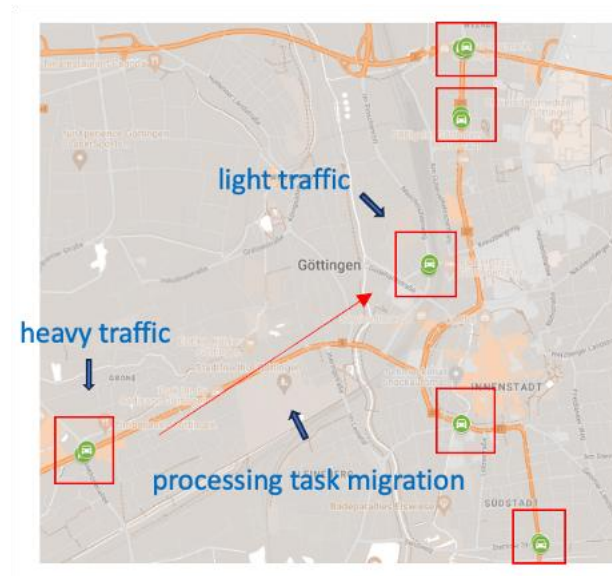


Figure 7: P1 – Inter-Cluster Task Migration.

P1 required software customization at both the application level and the CODECO integration level. At the application level, the LiDAR-based traffic analytics system was implemented as a Kubernetes-native microservice pipeline rather than as a monolithic application. This decomposition was necessary not only for modularity, but also to expose the right control points for runtime orchestration by CODECO. More specifically, the application was split into three main pod types. The DMP interfaces with the LiDAR stream, performs data conversion and compression, and forwards the resulting point-cloud data to the processing stage. The Processing Pod (PP) receives the transferred data, performs decompression and conversion to the KITTI format, and executes

PointPillars-based object detection. The Collector Pod (CP) receives the produced outputs and stores the resulting traffic statistics in a PostgreSQL backend. The processing pipeline is implemented in C++ with CUDA and TensorRT to achieve high-throughput and low-latency inference on Jetson-class embedded GPUs. In addition, Draco is used as the point-cloud compression method to reduce transmission cost over wireless links.

From the CODECO perspective, the key customization is not the use of these tools in isolation, but the way the pipeline is structured for runtime orchestration. Each worker node hosts one DMP that remains attached to the corresponding sensing source, while one or more PPs can be placed locally or on neighbouring nodes depending on current resource and network conditions. This software design allows CODECO to treat the compute-intensive inference stage as a movable workload, enabling runtime remapping and resilience-oriented migration instead of static placement. CODECO supports workload reallocation when processing pod failures occur or when channel instability and latency degradation threaten timely analytics.

Overall, the software customization of P1 should therefore be understood as the co-design of the LiDAR analytics pipeline and its CODECO integration. The main contribution is not merely that PointPillars, Draco, and a database are used, but that the application was restructured into migration-aware microservices that CODECO can deploy, observe, and adapt at runtime.

## 4.3 Experimentation and Evaluation

### 4.3.1 Experimentation

The experimentation was designed to validate whether CODECO improves the operation of a LiDAR-based traffic analytics pipeline under realistic edge constraints, assessing the combined behaviour of the application pipeline and the framework under varying workload placement and disturbance conditions. **Three validation objectives were defined:** evaluating the throughput–latency trade-off between local and offloaded processing configurations; validating CODECO's ability to react to degradation events through pod migration and workload remapping; and quantifying the overhead introduced by the framework and the multi-pod pipeline design.

Validation scenarios were organised into three classes. The first **compared local-only, mixed local/remote, and fully remote processing configurations**. The second evaluated disturbance-triggered migration under increasing processing-pod failure rate, increasing channel failure rate, and increasing channel latency between DMPs and PPs. The third measured deployment overhead, CODECO runtime overhead, and use-case overhead in terms of resource utilisation and performance impact. All scenarios were implemented through dedicated scripts to ensure repeatable, observable behaviour. Each worker node received the point-cloud stream of one specific sensor, and the PointPillars models were pretrained for detection of three object classes: Car, Pedestrian, and Truck.

### 4.3.2 Evaluation

The results confirm that CODECO provides measurable benefits for LiDAR-based traffic analytics under constrained edge conditions, while also revealing the operational trade-offs associated with workload remapping.

Regarding deployment overhead, integrating CODECO increases overall deployment time compared with a Kubernetes-only baseline, but the deployment of the use-case itself adds only negligible delay once the framework is installed (Figure 8).

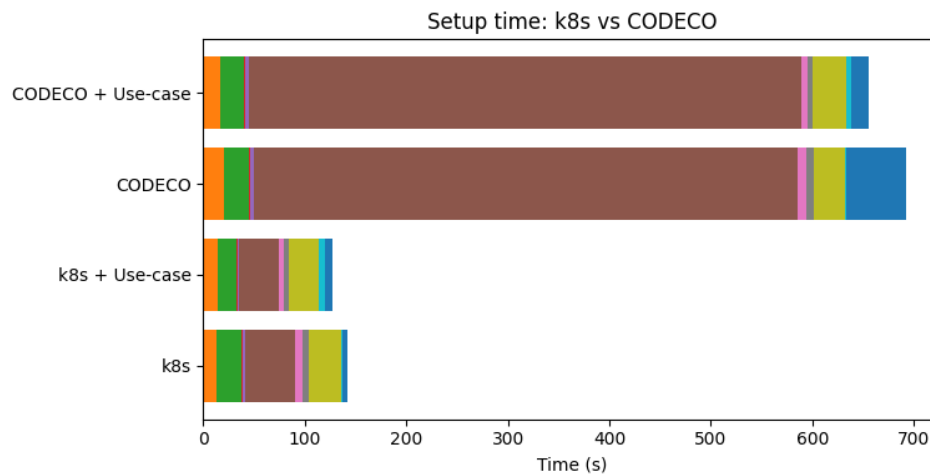


Figure 8: P1 – Deployment time comparison between a Kubernetes-only baseline and the CODECO-enabled deployment.

■ Connection Test, ■ Initializing kubeadm, ■ Joining worker nodes, ■ Disable apparmor, ■ Base deployment, ■ Install CODECO/core deployment, ■ Install NVIDIA device plugin, ■ Install Dashboard, ■ Install Ingress NGINX, ■ Deploy use-case, ■ Waiting for all pods to be running)

Regarding workload offloading, a clear throughput–latency trade-off emerges. At a LiDAR sensor frame rate of 45 fps, a configuration with 50% remote processing achieves 32% higher throughput than the no-offloading baseline. At the same time, at a publisher rate of 30 fps, end-to-end latency increases by up to 43% as the offloading level rises. These results confirm that workload remapping can improve throughput when local GPU resources are constrained, but that the gain is bounded by wireless-network limitations (Figure 9).



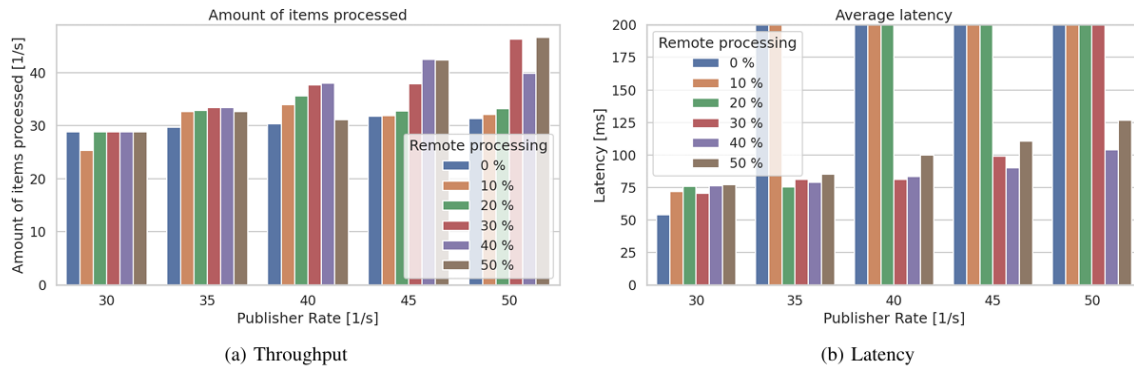


Figure 9: P1 – Effect of workload migration on throughput and end-to-end latency.

Regarding migration behaviour, CODECO reacts effectively to degradation events including pod failures, channel failures, and channel latency increases, though migration itself incurs non-negligible delay. Measured migration time increases with the number of pods relocated, ranging from 18.8 seconds for two pods to 55.8 seconds for ten pods, indicating that large-scale simultaneous remapping carries a meaningful time cost (Figure 10).

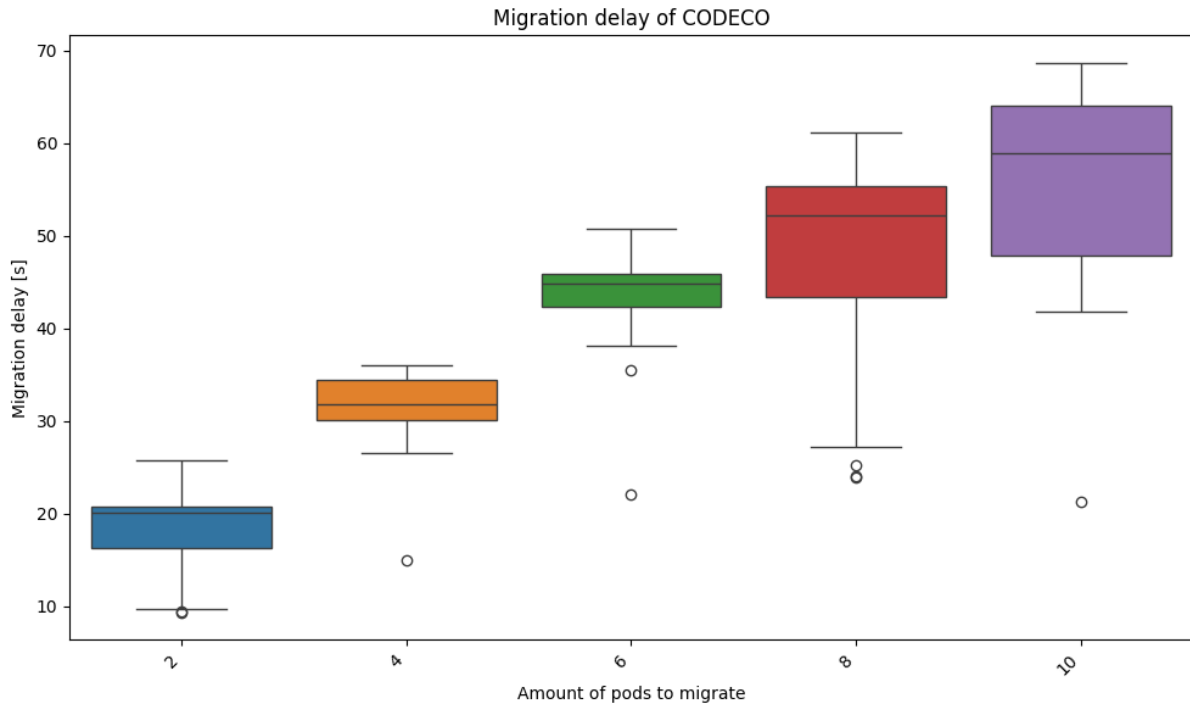


Figure 10: P1 – Effect of workload migration on throughput and end-to-end latency (sample size  $n = 30$ ).

Regarding framework overhead, CODECO does not reduce pipeline throughput, while increasing edge-node memory usage by approximately 0.99 GiB on average with CPU utilisation essentially unchanged. Network-overhead analysis shows that transmission delay is the dominant contributor to the latency increase of remote processing, with inference remaining the second most important factor. Overall, the main performance bottleneck in remote processing is the wireless path rather than the framework itself (Figure 11).

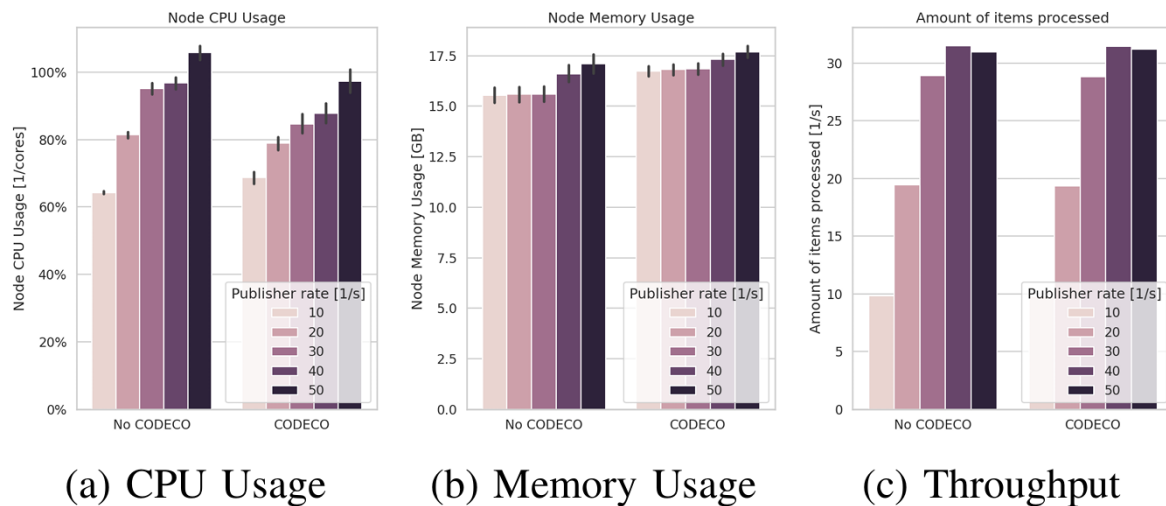


Figure 11: P1 – CODECO overhead on edge nodes in terms of CPU usage, memory usage, and throughput.

## 4.4 Demonstration

### 4.4.1 Technical setup

The P1 demonstration took place in a seminar room of the Institute of Computer Science of the University of Göttingen (Goldschmidtstr. 7, 37077 Göttingen). The following equipment was used:

- 1× LiDAR sensor for capturing detailed 3D point-cloud data
- 2× NVIDIA Jetson AGX Orin edge devices for real-time data processing, integrated with CODECO
- 1× Raspberry Pi 5 as the Kubernetes master node
- 1× Communication module providing Wi-Fi Direct for intra-cluster communication and 5G connectivity for inter-cluster task migration

The application deployed on the CODECO infrastructure was CUDA-PointPillars, a high-performance 3D object detection framework that processes LiDAR point-cloud data in real time using an optimised GPU implementation of the PointPillars model. In the demonstration, CUDA-PointPillars enabled real-time analysis of traffic scenes, generating object detection outputs that drive CODECO's task migration decisions based on system load.

### 4.4.2 Demo Scenarios

Three scenarios were designed to demonstrate CODECO's adaptive behaviour under different types of networks and node degradation.

#### 4.4.2.1 Demo Scenario 1: Channel Latency and Throughput Degradation between DMPs and PPs

**Goal:** Demonstrate CODECO adaptation when the network channel between pipeline components degrades.

**KPIs highlighted:** End-to-end pipeline latency (expected to rise during degradation, then recover after adaptation); node placement changes (expected: PPs migrated to a more suitable node).

Table 2: P1, demo 1 steps.

| Step | Action  | Observation   |
|------|---|---|
| 0    | Pre-defined state: cluster is running (1x Raspberry Pi 5 master, 2x Jetson AGX Orin workers). Grafana dashboard is visible. | Baseline metrics shown on Grafana (steady end-to-end latency / healthy links).  |
| 1    | Deploy PPs on opposite worker nodes (to ensure traffic crosses the network path).   | Current placement can be shown via cluster view and correlated with Grafana.  |
| 2    | Run a script that simulates an increase in channel latency / reduced throughput between DMPs and PPs.                       | Grafana shows degraded network/channel metrics and end-to-end latency rising.   |
| 3    | CODECO reacts by changing recommendations / placement (expected: migrate PPs towards the more suitable node).               | Grafana: latency increases first, then decreases once the new placement stabilizes; placement/recommendation change is visible. |

#### 4.4.2.2 Demo Scenario 2: Increasing Pod failure rate for PPs on a specific node

**Goal:** Demonstrate CODECO resilience and recovery when repeated failures occur on one node.

**KPIs highlighted:** Failure rate and restart events for the affected PP; end-to-end latency before and after adaptation (expected to rise then recover).

Table 3: P1, demo 2 steps.

| Step | Action   | Observation   |
|------|--|---|
| 0    | Pre-defined state: cluster and monitoring running; baseline performance stable.                                    | Grafana shows normal operation.   |
| 1    | Run a script that simulates PP pod failures on a specific node (e.g., repeated termination / crash loop behavior). | Grafana shows increased failures; end-to-end latency may rise due to restarts/unavailability.           |
| 2    | CODECO reacts by updating node recommendations / migrating PPs away from the failing node.                         | Placement change is observable; end-to-end latency increases first, then decreases after stabilization. |

#### 4.4.2.3 Demo Scenario 3: Increasing channel failure rate on a specific network channel

**Goal:** Demonstrate CODECO adaptation to unstable channels affecting pipeline connectivity.

**KPIs highlighted:** Channel failure rate (expected to spike during fault injection, then reduce after adaptation); end-to-end latency trend (expected to rise then recover).

Table 4: P1, demo 3 steps.

| Step | Action  | Observation   |
|------|---|---|
| 0    | Pre-defined state: stable connectivity and baseline metrics.                          | Grafana shows healthy channel behavior.   |
| 1    | Run a script that simulates channel failures on selected network channels.            | Grafana shows increased channel failures, transient performance degradation visible.                |
| 2    | CODECO triggers adaptation (expected: move affected PPs / adjust placement strategy). | Node recommendations/placement changes; end-to-end latency improves after the adaptation completes. |



#### 4.4.2.4 Demo Events

Two demonstrations of P1 were scheduled during the final phase of the project. The first one was held with an early version of the implementation of P1, with a focus on the traffic monitoring application. A final one was held later, when the implementation was in a more mature state.

*Table 5: P1 - First demo event Overview.*

| Date / Time       | October 29, 2025   |
|-------------------|--|
| <b>Organizers</b> | University of Göttingen team   |
| <b>Audience</b>   | Local audience   |
| <b>Duration</b>   | The demonstration spanned around 60 minutes.   |
| <b>Content</b>    | This first demonstration focused on traffic monitoring application. Audience were presented with the results thus far. |

*Table 6: P1 - Second Event Overview.*

| Date / Time       | February 27, 2026   |
|-------------------|---|
| <b>Organizers</b> | University of Göttingen team  |
| <b>Audience</b>   | Local audience, City of Göttingen   |
| <b>Duration</b>   | The demonstration spanned around 60 minutes.  |
| <b>Content</b>    | <p>The intention of the second demonstration was to focus on the interaction of CODECO with our use case application, as now a representatives of the City of Göttingen were present. During the second demonstration, we showed:</p> <ul style="list-style-type: none"> <li>• the migration of pods.</li> <li>• hybrid processing vs. local processing: we have experimented with different ratios of processing pods deployed on the actual node (sensor data processed on same node where data was sensed) and pods deployed on another node (sensor data processed on other node). The result is that a mixture of local processing of the sensor data on the same node and remote processing on other nodes leads to higher throughput than only local processing, which shows the effectiveness of migrating pods to other nodes.</li> <li>• node recommendations by PDLC.</li> <li>• deployment of CODECO &amp; Use Case App, show &amp; explain the User Interface and Grafana dashboard.</li> <li>• Channel Failure Rate/Latency increase leads to changing NetMA performance metrics (bandwidth, etc.), showing that NetMA detects these kind of problems.</li> </ul> |

### 4.5 Risks and Mitigation Measures

Prior to the execution of the demonstrations, we identified several dependencies, risks and mitigation measures, which are summarised in Table 7.

*Table 7: P1 - Demo risks and mitigation measures.*

| Risk  | Mitigation  |
|---|---|
| <b>Contract with the city or approval of suggested data license by the legal department takes too much time</b> | Switch to alternative demo scenario   |
| <b>Scenarios turn out not to work during dry-run</b>  | Analyse issues, have fallback scenarios in place to demonstrate parts of the intended scenarios, hardcoding |
| <b>Equipment not installed in time</b>  | Switch to alternative demo scenario   |

| Risk  | Mitigation                          |
|---|-------------------------------------|
| No heavy traffic at any node during demonstration | Switch to alternative demo scenario |

## 4.6 Summary

The P1 use-case demonstrated how the CODECO framework supports adaptive, LiDAR-based traffic monitoring through dynamic workload orchestration across edge nodes and clusters. The core challenge addressed was the variability in point-cloud data volume across the day, which can overwhelm resource-constrained edge devices during peak traffic periods and compromise the timeliness of real-time analytics.

The architecture decomposed the LiDAR analytics pipeline into migration-aware microservices — Data Migration, Processing, and Collector Pods — coordinated by CODECO across a Kubernetes-based edge cluster. The system was evaluated against its ability to handle resource contention, network variability, and workload spikes. CUDA-PointPillars and Draco provided efficient point-cloud inference and compression over wireless links, while the defined KPIs — processing throughput, end-to-end latency, migration delay, and resource utilisation — provided measurable performance benchmarks.

Experimentation and demonstration results confirmed that dynamic workload migration improves throughput, resilience, and adaptability compared to static processing. A 50% remote processing configuration achieved 32% higher throughput at 45 fps, while CODECO responded effectively to pod failures, channel failures, and latency degradation. The main costs identified were migration delay, which scales with the number of pods relocated, and a moderate increase in memory usage attributable to the framework. The wireless path was identified as the dominant performance bottleneck in remote processing configurations, rather than the CODECO framework itself.

## 5 P2: Mobility

As described in Section 2.2, this pilot addresses VRU protection through a Vehicular Digital Twin deployed across the edge-cloud continuum, using CODECO to orchestrate the V2X services that sustain real-time cooperative awareness between road users. The sections below detail the use-case architecture, deployment, and experimentation.

### 5.1 Use-Case Architecture

To elaborate on the concepts described in the previous section, the Vehicular Digital Twin use-case incorporates several modules that implement specific ETSI Cooperative Intelligent Transport Systems (C-ITS) protocols [4], which form the European V2X protocol stack [5]. Additionally, other modules are responsible for collecting and processing the information generated within the system. A detailed overview of the modules expected to be deployed is presented in Figure 12.

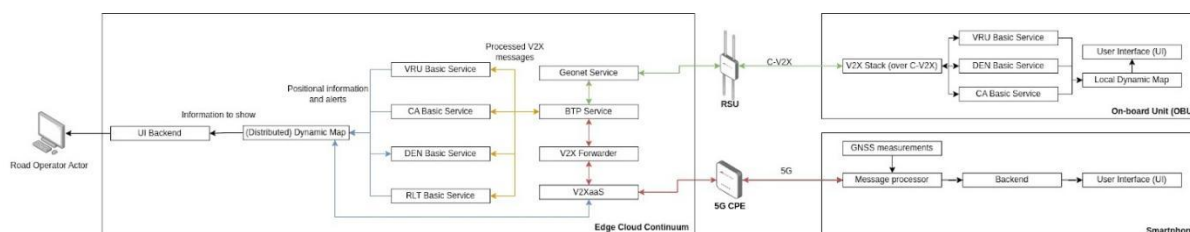


Figure 12: P2 – Functional View of the Vehicular Digital Twin Use-Case.

The Vehicular Digital Twin use-case is built around a set of modules implementing specific ETSI *Cooperative Intelligent Transport Systems (C-ITS)* protocols, which form the European

V2X protocol stack. Additional modules handle the collection and processing of information generated within the system. A functional overview of the full module set is presented in Figure 13.

The modules operating within the edge-cloud continuum are as follows. The **Geonetworking Service** processes GeoNetworking protocol headers from messages sent and received within the Cellular-V2X (C-V2X) environment. The **Basic Transport Protocol (BTP) Service** processes BTP headers for messages destined for the C-V2X environment; since incoming messages must have their GeoNetworking headers handled first, a BTP service module must always be connected to a Geonetworking Service instance. The **VRU Awareness Basic Service** processes VRU Awareness Messages (VAMs), which carry the position and trajectory of one or more VRUs in a highly compressed format. The **Cooperative Awareness (CA) Basic Service** processes Cooperative Awareness Messages, which convey the position and trajectory of vehicles and are disseminated periodically by road users. The **Decentralised Environment Notification (DEN) Basic Service** manages the generation and processing of DEN Messages (DENMs), urgent alerts concerning potential or existing incidents in the mobility environment. The **Road Lane Topology (RLT) Service** distributes Map Extended Messages (MAPEMs) containing high-definition maps of the mobility environment, compressing extensive geographical data into messages of 100–500 bytes.

The **V2X as a Service (V2XaaS)** module implements a protocol stack for VRUs connected via conventional IP networks. It collects GNSS measurements from smartphones, generates VAMs with GeoNetworking and BTP headers on the edge-cloud continuum, returns UI instructions to smartphones, and processes incoming messages from the C-V2X environment. The **V2X Forwarder** relays messages between the C-V2X and IP-based network realms, facilitating bidirectional communication between V2XaaS and Geonetworking modules. The **Dynamic Map**, functioning as the Digital Twin, aggregates data from the Geonetworking, BTP, and CA Basic Service modules to maintain a continuously updated representation of the positions and movements of all connected road users. It detects potential incidents and issues advisories to vehicles and VRUs, while also providing real-time data to the backend for road operator monitoring. The **Backend** receives data from the Dynamic Map and exposes it to users through the frontend. The **Frontend Web Server** serves the frontend over HTTP and operates independently of all other modules, allowing it to be deployed separately if required.

All modules are designed to be horizontally scalable — multiple instances of any module, such as the Geonetworking Service, can run simultaneously across different edge nodes to absorb demand surges.

### 5.1.1 Key Performance Requirements

Given that this use-case is safety-critical, performance and reliability are non-negotiable requirements. Failure to deliver information with sufficient freshness and speed could have severe or fatal consequences in a real mobility environment. The system must therefore maintain end-to-end latencies below 30 ms, with most exchanges expected to fall within the 5–10 ms range. Age of Information (AoI) must remain below 500 ms, with an average typically between 100–200 ms. Packet loss must stay below 5%. These thresholds govern both the design of the deployment and the behaviour of CODECO's orchestration logic during operation.

### 5.1.2 Deployment Architecture and CODECO Integration

The deployment is organised as a single cluster comprising a cloud node, two edge nodes, and two RSUs each directly connected to one edge node, with C-V2X-capable OBUs and 5G smartphones on the client side. The deployment schematic is shown in Figure 13.

ACM initialises and configures all modules for a seamless initial deployment. SWM manages ongoing workload placement and migration based on PDLC recommendations, migrating

protocol processing modules to the nearest suitable edge node as client locations and latency profiles change. NetMA continuously monitors latency and throughput between nodes and pods, sharing this data with PDLC to inform optimisation and trigger migrations where needed. PDLC aggregates metrics from all components, applies heuristic reasoning to determine optimal placement strategies, and operates proactively — anticipating degradation before thresholds are breached.

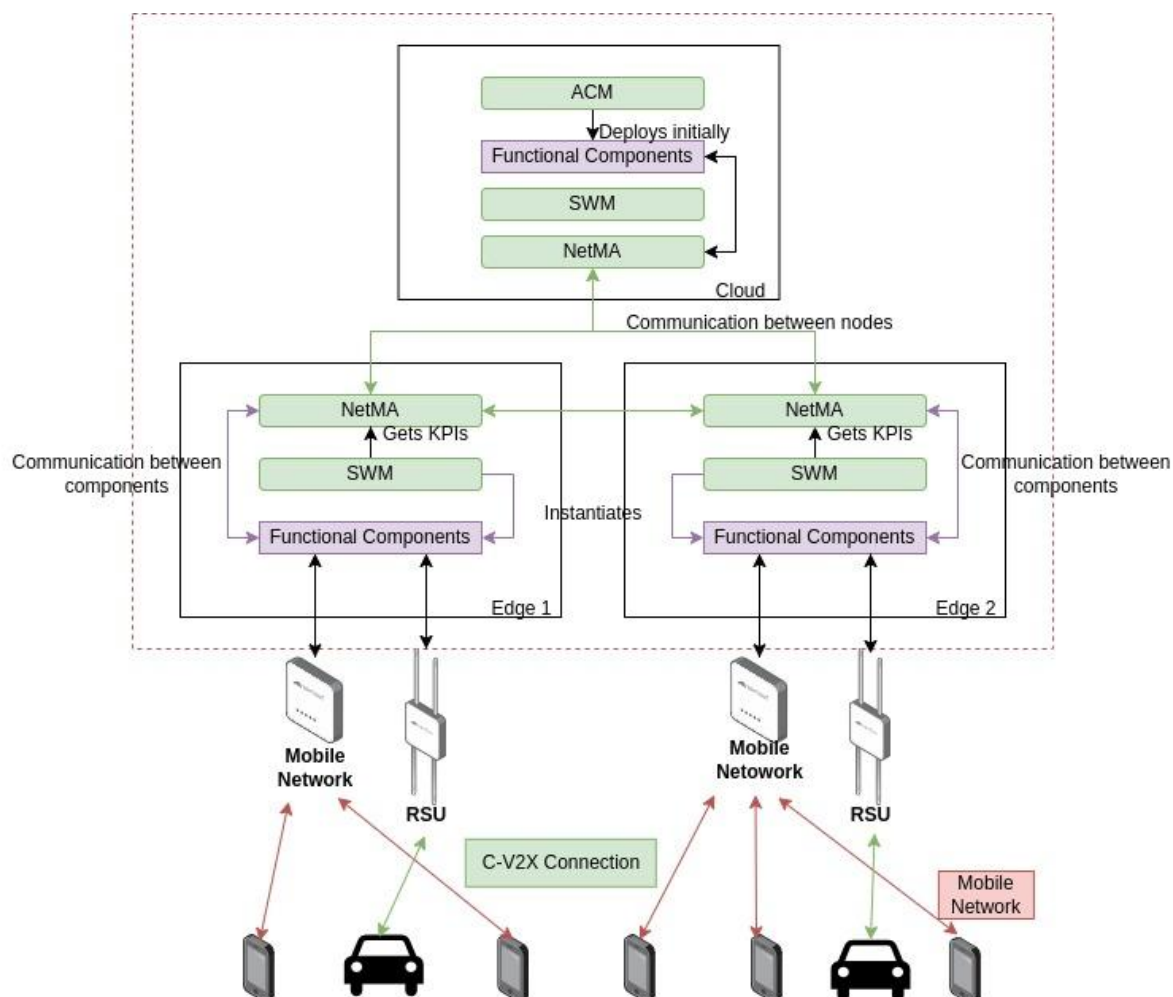


Figure 13: P2 – Deployment Diagram.

### 5.1.3 Preconditions, Triggers, and Postconditions

Before deployment, the mobile network must be properly configured, all RSUs connected to the cluster, and all edge and cloud nodes operational. During operation, CODECO continuously monitors all defined metrics, triggering corrective actions on threshold breaches and applying pre-emptive measures as limits are approached. Upon completion, all operational metrics are logged for performance evaluation and future deployment refinement.

## 5.2 Lab Deployment

The use-case was physically deployed at the i2CAT facilities at Campus Nord, Barcelona. As shown in [Figure 14](#), the site combines pedestrian sidewalks and vehicle roads, with a clearly identified collision-prone intersection. CODECO was required to always keep V2X services available while meeting latency and AoI requirements, even as additional pedestrians and vehicles entered the environment and placed growing demands on network and compute



resources.

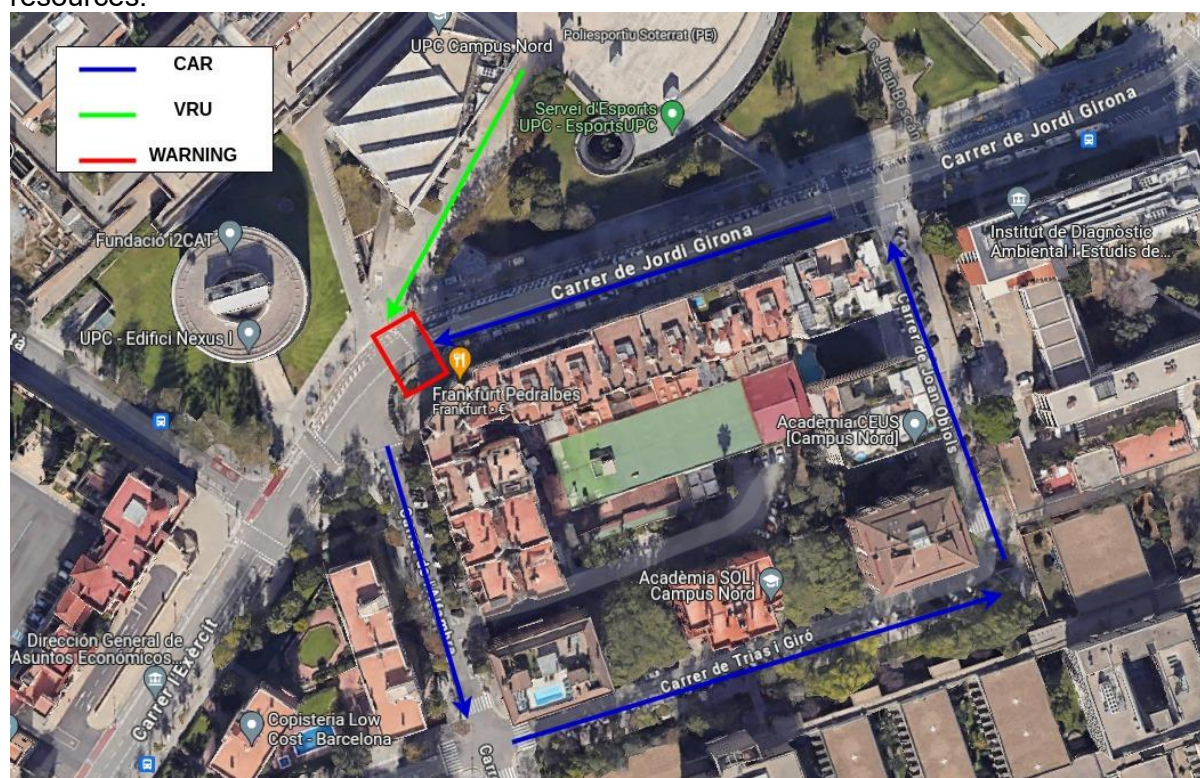


Figure 14: P2 – Scenario Facilities.

## 5.2.1 Scenario and Infrastructure

Three nodes with different computational profiles were used, as detailed in Table 8. The Jetson Orin and Jetson Orin Nano were connected to V2X radio units — Unex V2X modems or Cohda MK6 OBUs — via dedicated Ethernet or USB connections. A public cloud instance provided additional compute capacity. Smartphone-equipped users connected via the mobile 5G network.

Table 8: P2 – P2 infrastructure, Hardware.

| Name                   | Central Processing Unit (CPU)                   | Random-Access Memory (RAM) | Disk     |
|------------------------|---|----------------------------|----------|
| Supermicro SYS-610C-TR | Intel(R) Xeon(R) Silver 4316 20 Cores @ 2.30GHz | 225 Gigabyte (GB)          | 3,840 GB |
| Jetson Orin            | Arm® Cortex®-A78AE 12 Cores @ 2.20GHz           | 64 GB                      | 64 GB    |
| Jetson Orin Nano       | CPU Arm® Cortex®-A78AE 6 Cores @ 1.50GHz        | 8 GB                       | 64 GB    |

## 5.2.2 Software Customization

For this use-case, there is a need to connect to specific radio units (e.g., V2X modems) that require custom-mode plug-ins. These plug-ins had to be tailor-made and deployed simultaneously with the CODECO framework when the use-case is launched. Similarly, to ensure service reliability, there was a scaling on the deployment based on key metrics. These metrics, exposed by the CODECO framework, were measured and evaluated by i2CAT to determine the appropriate scaling actions. In this stage, K8s Horizontal Pod Autoscaler<sup>3</sup> was exploited, which monitors key metrics to scale the deployments of P2 appropriately.

<sup>3</sup> <https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/>

## 5.3 Experimentation and Evaluation

To test the performance of the CODECO framework and its intelligent adaptability, custom created simulators that can mock the sending of V2X messages using the Simulation of Urban Mobility (SUMO) simulator<sup>4</sup> were used. Also, to artificially restrict the throughput or latency of a network link (i.e., Linux interface) the standard Linux networking tools (e.g., tc<sup>5</sup>) were exploited. Finally, to artificially load the nodes, we used the Debian stress function<sup>6</sup> that allows for both RAM and CPU loading. These tools enable testing the performance of the CODECO framework for P2, as well as fine-tuning the infrastructure to its fullest potential.

### 5.3.1 Key Performance Metrics

Four metrics were used to assess system performance, evaluated collectively to avoid the blind spots that arise from measuring any one in isolation.

**Latency** was measured using timestamped beacon packets sent periodically between nodes, rather than conventional ICMP RTT, since V2X packets are predominantly non-unicast and delay may be asymmetric between directions. **Aol** measured information freshness at the Dynamic Map using the formula  $Aol = t - u(t)$ , where  $u(t)$  is the timestamp of the latest received update. **Jitter** — the variation in packet-to-packet delay — was derived from the same timestamped latency data and served as an indicator of link stability. **Packet loss** was estimated from the expected periodicity of V2X awareness packets: since these are sent at fixed intervals, deviations from the expected update count indicate dropped packets, compensating for the absence of acknowledgement mechanisms in BTP and broadcast communications.

### 5.3.2 Experimentation

*Note on experimental environment:* The experimentation described below was conducted in a simulated environment distinct from the physical i2CAT deployment. The lab setup used KinD (Kubernetes in Docker) on a single high-performance workstation to enable controlled, reproducible benchmarking across a wide range of traffic loads not achievable in the physical deployment. The physical setup described in Section 5.2 was used for the demonstration, as described in Section 5.4.

**Objectives:** The experimentation benchmarked a CODECO-managed distributed Digital Shadow (DS) against a centralised baseline, assessing whether distributing microservices across an Edge-Cloud continuum could sustain data freshness and spatial accuracy under growing vehicular load. Two metrics beyond conventional latency were used: Aol and Error Distance. Aol grows linearly between updates and resets on receipt of a fresh one, reflecting the currency of the system's awareness. Error Distance measures the Euclidean gap in metres between the Digital Shadow's kinematically extrapolated position estimate and the vehicle's actual ground-truth position at the time of the next update — a directly safety-relevant quantity for VRU protection.

**Setup:** SUMO generated vehicular dynamics across a 4 km motorway section with two to three lanes per direction, covering approximately 0.08 km<sup>2</sup>. Traffic density ranged from 1 to 300 simultaneous vehicles. Each vehicle was represented by a Docker container generating ETSI C-ITS messages at 1 Hz over UDP with BTP and GeoNetworking headers, routed to the nearest virtual edge node. The ceiling of 300 vehicles was chosen to avoid near-gridlock conditions at higher densities, which artificially suppress positional errors regardless of system latency.

<sup>4</sup> <https://eclipse.dev/sumo/>

<sup>5</sup> <https://www.man7.org/linux/man-pages/man8/tc.8.html>

<sup>6</sup> <https://manpages.debian.org/testing/stress/stress.1.en.html#:~:text=stress%20is%20a%20tool%20that%20imposes%20a%20configurable,errors%20it%20detects.%20stress%20is%20not%20a%20benchmark.>

The KinD cluster comprised three Edge Tier nodes (0.3 CPUs each) and one Cloud Tier node (0.5 CPUs), running on an Intel Core i9-10940X workstation at 3.30 GHz with 28 logical CPUs. The centralised DS ran exclusively on the cloud node; the CODECO-managed distributed DS scheduled and migrated microservices dynamically across all four nodes. PDLC used a modified PPO reinforcement learning agent configured to minimise Aol, issuing placement recommendations to SWM based on Aol values and node resource availability.

**Results: Age of Information:** Below 100 simultaneous vehicles, the centralised deployment performed marginally better, as a monolithic process avoids the inter-service communication overhead of a distributed architecture and the cloud node had sufficient spare capacity. Beyond 100 vehicles the two approaches diverged. The centralised DS hit its processing ceiling, with average Aol rising from 1,230 ms at 100 vehicles to 1,624 ms at 300. The CODECO-managed distributed DS reached 1,405 ms at 300 vehicles — a 13.5% improvement — with a substantially flatter Aol growth curve, reflecting proactive workload redistribution before bottlenecks formed. Both systems operated near the theoretical 500 ms Aol floor at low vehicle counts, confirming correct experimental calibration.

*Table 9: P2, Average Aol vs. Number of Connected Vehicles.*

| Connected Vehicles | Centralised DS — Avg. Aol (ms) | CODECO Distributed DS — Avg. Aol (ms) | Improvement (%) |
|--------------------|--------------------------------|---------------------------------------|-----------------|
| 1                  | 506                            | 543                                   | —               |
| 2                  | 517                            | 555                                   | —               |
| 5                  | 553                            | 592                                   | —               |
| 10                 | 632                            | 654                                   | —               |
| 20                 | 689                            | 712                                   | —               |
| 50                 | 1,040                          | 1,150                                 | —               |
| 100                | 1,230                          | 1,206                                 | 1.9%            |
| 200                | 1,489                          | 1,322                                 | 11.2%           |
| 300                | 1,624                          | 1,405                                 | 13.5%           |

**Error Distance:** Below 100 vehicles, Error Distance was virtually indistinguishable between deployments, hovering around 1.4–1.5 m, as positional accuracy at low load is governed primarily by the kinematic extrapolation model rather than network conditions. At 300 vehicles, the centralised DS reached 2.04 m average Error Distance while the CODECO deployment remained at 1.56 m — a 23.6% reduction. The divergence stems from packet loss caused by cloud node overload in the centralised architecture, forcing extrapolation over longer intervals without fresh ground-truth data. Both deployments remained within the 3.5 m safety threshold (standard highway lane width) at all tested densities, but the CODECO deployment maintained a nearly 0.5 m wider safety margin at peak load — a meaningful difference in a system designed to distinguish a pedestrian on the pavement from one stepping into the road.

*Table 10: P2, Average Error Distance (Penalty Aol) vs. Number of Connected Vehicles.*

| Connected Vehicles | Centralised DS — Avg. Error Dist. (m) | CODECO Distributed DS — Avg. Error Dist. (m) | Improvement (%) |
|--------------------|---------------------------------------|--|-----------------|
| 1                  | 1.374                                 | 1.385  | —               |
| 2                  | 1.394                                 | 1.396  | —               |
| 5                  | 1.397                                 | 1.401  | —               |
| 10                 | 1.443                                 | 1.458  | —               |
| 20                 | 1.443                                 | 1.463  | —               |
| 50                 | 1.461                                 | 1.475  | —               |
| 100                | 1.494                                 | 1.488  | 0.4%            |



| Connected Vehicles | Centralised DS — Avg. Error Dist. (m) | CODECO Distributed DS — Avg. Error Dist. (m) | Improvement (%) |
|--------------------|---------------------------------------|--|-----------------|
| 200                | 1.786                                 | 1.501  | 16.0%           |
| 300                | <b>2.036</b>                          | <b>1.555</b>                                 | <b>23.6%</b>    |

## 5.4 Demonstration

### 5.4.1 Technical setup

The P2 demonstration took place at the Nexus building of Campus Nord, Universitat Politècnica de Catalunya (UPC), next to the i2CAT premises. The location — an active urban intersection adjacent to a university campus with regular movement of cars, motorcycles, trucks, pedestrians, cyclists, and electric scooter users — provided a realistic and operationally representative environment for a VRU protection demonstration.

The infrastructure hardware was as described in Section 5.2, with connectivity to the V2X communication environment established through two RSUs. On the client side, VRUs connected using Android smartphones with integrated GPS, and vehicles used OBUs. The specifications of the RSUs and OBUs are provided in Table 8.

*Table 11: P2 - V2X Equipment Specs, demo infrastructure.*

| Name                 | CPU  | RAM         | Disk       |
|----------------------|--|-------------|------------|
| <b>Cohda Mk6 OBU</b> | NXP i.MX 8M Plus (4× Arm Cortex-A53 @ 1.8 GHz + 1× Cortex-M7 @ 800 MHz); V2X: 2× NXP RoadLink® SAF5400 (DSRC/IEEE 802.11p) + Qualcomm SA515M (C-V2X/LTE-V2X PC5, 3GPP R14); Cellular: 5G NR (3GPP R15) with 4G/3G/2G fallback; GNSS: u-blox F9P multi-band (GPS, Galileo, GLONASS, BeiDou), <2 cm RTK accuracy; Wi-Fi 802.11 a/b/g/n/ac; Bluetooth 5.1                     | 4 GB LPDDR4 | 32 GB eMMC |
| <b>Cohda Mk6 RSU</b> | NXP i.MX 8M Plus (4× Arm Cortex-A53 @ 1.8 GHz + 1× Cortex-M7 @ 800 MHz); V2X: 2× NXP RoadLink® SAF5400 (DSRC/IEEE 802.11p) + Qualcomm SA515M (C-V2X/LTE-V2X PC5, 3GPP R14); Cellular: 5G NR (3GPP R15) with 4G/3G/2G fallback; GNSS: u-blox F9P multi-band (GPS, Galileo, GLONASS, BeiDou); Wi-Fi 802.11 a/b/g/n/ac; Bluetooth 5.1; Enclosure: IP67 / NEMA 4X weatherproof | 4 GB LPDDR4 | 32 GB eMMC |

Three primary actors participated in the demonstration. The VRU, represented by a pedestrian, connected to the Digital Twin via smartphone and conventional mobile network, enabling periodic position updates and real-time alerts about nearby vehicles. The Vehicle, a car navigating the test environment, connected via V2X communication and simulated approaching behaviour in areas where pedestrians might unexpectedly cross. The Road Operator monitored all road actors in real time through a dedicated dashboard, with full visibility of the Digital Twin's state and the system's collision detection outputs.

### 5.4.2 Demo Scenario

The demonstration centred on a simulated collision course between a pedestrian VRU and a vehicle, resolved through proactive alerts generated by the Vehicular Digital Twin. The scenario illustrated how a driver with no direct line of sight to a crossing pedestrian can benefit from the extended situational awareness provided by the system, enabling both parties to avoid a potentially dangerous incident before it occurs.



The core scenario was complemented by a stress-test phase in which multiple vehicles and pedestrians connected simultaneously via V2X and conventional mobile networks, demonstrating system resilience and reliability under increasing load. Throughout the demonstration, the road operator monitored all events through the use-case dashboard, providing a continuous view of system performance and Digital Twin accuracy.

The full deployment of equipment and actors is depicted in Figure 15, with the pedestrian crosswalk highlighted as the primary location of safety-critical interactions. Figure 16 illustrates the communication flows between road users and infrastructure across the different Radio Access Technologies employed, necessitated by hardware constraints in the deployment environment.

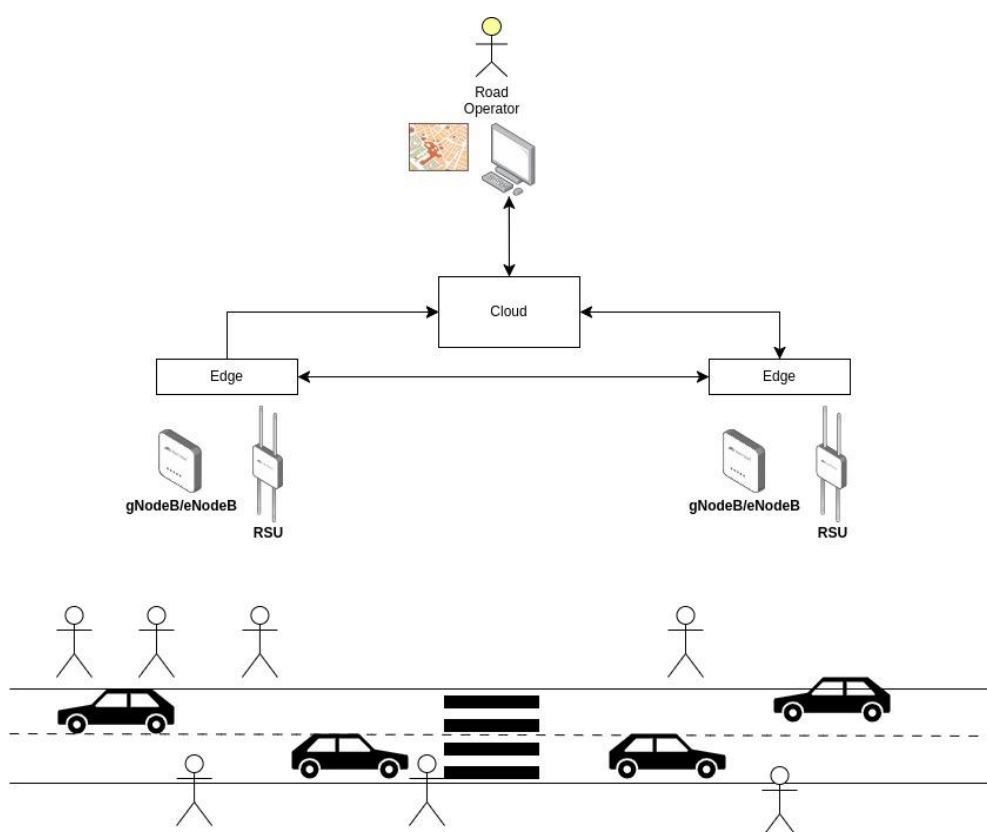


Figure 15: P2 - Infrastructure and actors involved.

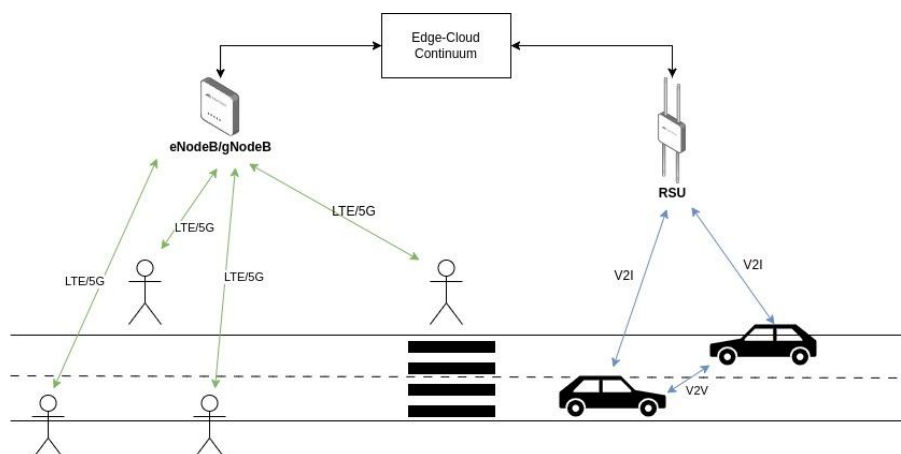


Figure 16: P2 - Connections between road users and infrastructure through different RATs.

### 5.4.3 Events

A demonstration of this use case was given in November 2025 at the i2CAT premises, as explained in Table 12.

*Table 12: P2 - Event Overview.*

| Date / Time       | November 24, 2025   |
|-------------------|---|
| <b>Organizers</b> | The event was organized by the use case leader Jordi Marias along with Daniel Ulied and the support of Rizkallah Touma and Alejandro Espinosa, who oversaw the PDLC on the side of i2CAT.   |
| <b>Audience</b>   | Local audience, CODECO partners   |
| <b>Duration</b>   | The demonstration was held in the morning and took 4 hours.   |
| <b>Content</b>    | The use case and all its technical details were presented, along with the CODECO framework. A live demo was given involving PDLC, the most important component within the CODECO framework for this use case.<br>The attendees were able to participate in the demonstration as VRUs just by using their own smartphone and getting connected to the service. |

### 5.5 Risks and Mitigation Measures

The success of the presented use case relied on several critical dependencies that could have posed risks to the demonstration. A complete overview of the identified risks and their mitigations is provided in Table 13.

*Table 13: P2 – Demo risks and mitigation measures.*

| Risk  | Mitigation   |
|---|--|
| <b>Potential Office Relocation by i2CAT</b>                   | Postpone the demonstration until after the relocation  |
| <b>Revocation of Permissions by Barcelona Council and UPC</b> | Demonstration in different (restricted) environment  |
| <b>Lack of approval from town halls or municipalities</b>     | Plan to deploy RSUs on the rooftops of i2CAT and UPC campus offices, which do not rely on municipal permissions  |
| <b>Degradation of OBUs and RSUs</b>                           | Procurement of replacement and postponement of the demonstration   |
| <b>Equipment Deployment and Power Supply problems</b>         | Demonstration in different (restricted) environment  |
| <b>Inadequate 5G coverage in certain urban areas</b>          | Use LTE connections without Edge servers (directly to the Cloud) for devices unable to connect to RSUs, or deploy a private 5G network to enhance coverage |
| <b>Vandalism targeting deployed infrastructure</b>            | Place all infrastructure on inaccessible rooftops to minimize the risk of vandalism and unauthorized access  |

### 5.6 Summary

The P2 use-case demonstrated how CODECO enables a Vehicular Digital Twin for VRU protection through real-time, distributed V2X communication and edge-cloud orchestration. The core objective was to maintain accurate, low-latency tracking of all road participants, supporting proactive detection and prevention of potential collisions.

Relative to traditional VANET-based approaches, the use-case leverages CODECO to bridge V2X and conventional mobile networks, reducing dependence on dedicated hardware and improving interoperability. VRUs connected via smartphone, vehicles via OBUs, and RSUs together with edge nodes ensured seamless communication across the continuum.

CODECO's orchestration layer continuously monitored latency, throughput, and resource usage across all components, dynamically migrating or scaling services to maintain the strict QoS thresholds — below 30 ms latency, below 500 ms AoI, and below 5% packet loss — that the safety-critical application demands.

Experimental results confirmed that while a centralised deployment performs marginally better at low traffic loads, the CODECO-managed distributed architecture outperforms it significantly as load grows. At 300 simultaneous vehicles, CODECO reduced average AoI by 13.5% and average positional Error Distance by 23.6% relative to the centralised baseline, while maintaining both deployments within the 3.5 m lane-width safety threshold. The additional safety margin provided by CODECO is operationally significant in a VRU protection context, where the difference of half a metre in positional accuracy determines the lead time available to issue a collision warning.

The live demonstration validated these capabilities in a real urban environment, successfully averting a simulated collision between a vehicle and a pedestrian VRU through timely Digital Twin alerts, and confirming system resilience under multi-user load conditions.

## 6 P3: MDS across Decentralized Edge-Cloud

As described in Section 2.3, this pilot demonstrates how a Media Delivery System can leverage tighter integration of networking and computational resources — enabled by CODECO — to optimise content delivery across a multi-domain, multi-cluster Edge-Cloud environment. The sections below detail the use-case architecture, deployment, experimentation, and demonstration.

### 6.1 Use-Case Architecture

The P3 use-case constitutes a complete multimedia content deployment environment in which CODECO handles the deployment, interconnection, and runtime adaptation of ETSI MEC instances and caches, allowing the MDS provider to focus exclusively on content delivery logic. The architectural example is provided in Figure 17. The system involves three principal actors. The **use-case user** is an external component that provides metrics to the Service Configuration Management layer, mirroring the behaviour of a real multimedia application. The **CODECO client** comprises both external use-case logic running outside the CODECO framework and code deployed within it. The **CODECO framework** is responsible for launching, monitoring, and managing the components defined in the CODECO Application Model.

Two components operate within the CODECO client. **MEC APIs** collect metrics from edge environments — including user location, zone information, and subscription data — supplementing the measurements provided by the CODECO framework. The **Client Database** aggregates metrics extracted from all MEC instances. In the deployment, MEC components are placed on edge nodes and the database on cloud nodes, with all communication managed through the CODECO framework. The interaction between the CODECO client and the CODECO framework is illustrated in Figure 17, showing MEC APIs deployed at the edge and metrics exported to the external client via an out-of-band channel.

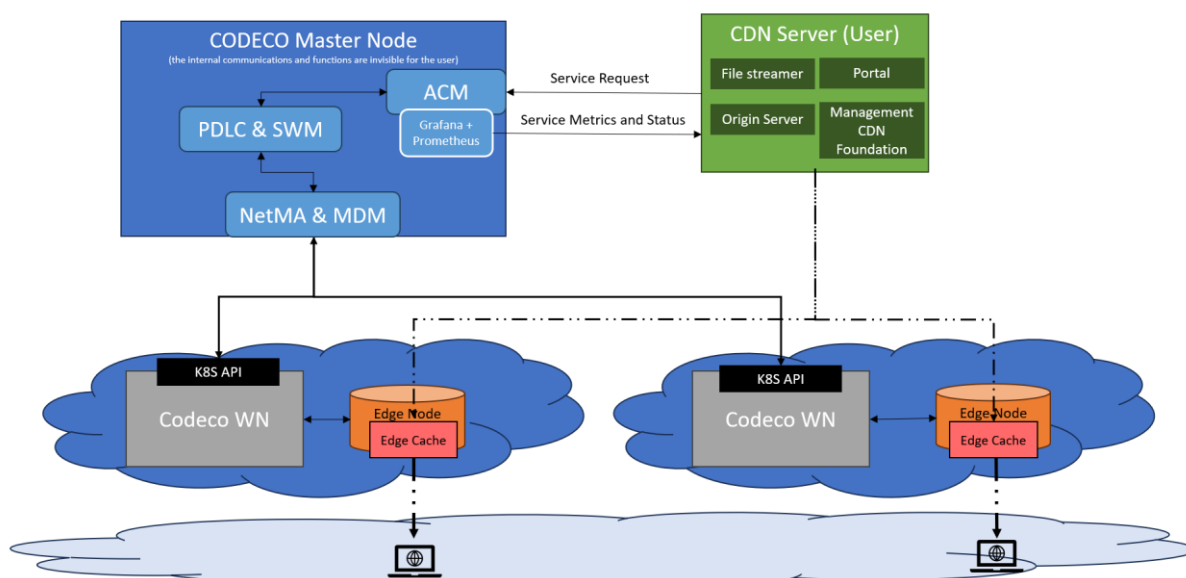


Figure 17: P3 – Interaction Scenario between CODECO Client and CODECO Framework.

The metrics used by the use-case are summarised in Table 1. Application-level metrics — user location, zones, and subscription areas — are managed directly by the MDS logic without passing through CODECO's internal databases. Infrastructure-level metrics — bandwidth, memory, and jitter — are provided by CODECO.

Table 14: P3 – Metrics Used.

| Metric                     | Source      | Description   |
|----------------------------|-------------|---|
| <b>Users</b>               | MEC P3 code | Query location information about a specific UC user or a group of UC users. |
| <b>Zones</b>               | MEC P3 code | Query the information about one or more specific zones or a list of zones   |
| <b>Subscriptions area</b>  | MEC P3 code | Retrieves information about the subscriptions for this requestor.           |
| <b>Bandwidth available</b> | CODECO      | Max bandwidth available for data transfer.                                  |
| <b>Memory available</b>    | CODECO      | Memory available at each endpoint where the cache deployment will occur.    |
| <b>Jitter</b>              | CODECO      | Average variation in the latency between instances deployed.                |

### 6.1.1 Preconditions and Postconditions

**Preconditions:** All edge and cloud devices must be connected, initialised, and provisioned with sufficient capacity to run CODECO, with a minimum of three nodes per cluster. In addition to CODECO-internal connectivity, external client-to-client communication must be possible and monitorable, either through network protocols or an SDN controller.

**Triggers:** Two conditions initiate CODECO adaptation. The first is a detected increase in media file requests requiring scaling or reconfiguration of edge caches. The second is an abnormal rise in failure rate or latency variation in one area, requiring service diversion to another.

**Postconditions:** Successful execution is expected to produce three outcomes: reduced inter-cache communication volume through more efficient deployment compared to a default Kubernetes setup; improved latency between users and caches and reduced storage requirements; and improved scalability and adaptability, with CODECO anticipating network problems and avoiding service outages or bottlenecks.

## 6.1.2 CODECO Integration

CODECO supports this use-case through the following components. ACM orchestrates deployment and handles scaling and reconfiguration requests. SWM enables non-disruptive migration of deployments between nodes. PDLC provides intelligent decision-making, anticipating network issues before they affect service. MDM and NetMA collect real-time node and network data to enable informed orchestration decisions. NetMA additionally ensures secure and transparent connectivity between all use-case components deployed within the framework.

## 6.2 Lab Deployment

The deployment and testing process progressed through four scenarios of increasing complexity, each building on the previous scenario to validate integration and optimise CODECO's orchestration behaviour.

**Scenario 1 — Virtual Single-Cluster:** The initial stage used KinD for virtualisation, with three CODECO nodes (one master, two workers). Two applications were deployed and interconnected through CODECO: a MEC API instance on a worker node, and a MongoDB database without placement constraints. The objective was to confirm successful deployment and inter-component connectivity.

**Scenario 2 — Physical Single-Cluster:** The same application set was deployed on three physical Ubuntu 22 compute nodes, validating that the behaviour observed in KinD translated correctly to a physical environment and that no configuration or environment-specific issues arose.

**Scenario 3 — Hybrid Multi-Cluster:** The physical deployment was extended with a virtualised cluster, creating a multi-cluster environment. ALTO servers were introduced, integrating network information from BGP and SDN controllers to provide visibility of the network topology beyond CODECO's internal view. This information was combined with MEC API data to give the use-case a complete view of the infrastructure.

**Scenario 4 — Full Physical Multi-Cluster:** The final stage deployed the complete system in a fully physical environment, with two edge nodes and one cloud node per cluster, and at least four nodes routed outside CODECO. Each cluster hosted one MEC API instance, one ALTO instance, one integration database, and one edge cache. Edge devices were Raspberry Pi 5 units; cloud nodes ran Ubuntu 22 Linux servers. The schematic is shown in Figure 18. This scenario formed the basis for all evaluation and demonstration activities.

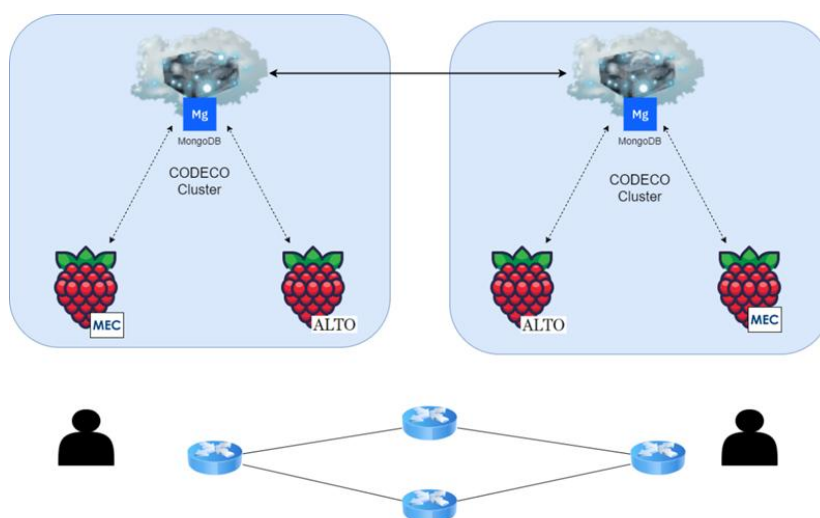


Figure 18: P3 lab infrastructure.

## 6.3 Experimentation and Evaluation

### 6.3.1 Evaluation Methodology

Evaluation was structured around the three postconditions defined in Section 6.1.1, assessed through a combination of closed-ended verification checks and comparative measurement. The same scenario of requests and resources was executed in two contexts: a default Kubernetes deployment and a CODECO-managed deployment. External monitoring tools — tcpdump for network traffic and top for compute resource usage — provided comparable measurements across both contexts.

Communication efficiency was assessed by measuring the volume of bytes exchanged between components in the K8s and CODECO deployments, with a reduction in interactions indicating more efficient resource placement. Scalability and adaptability were evaluated by testing behaviour in resource-constrained and connection-irregularity conditions; faster cache scaling and distribution with fewer service drops or bottlenecks was taken as fulfilment of the postcondition. QoE-related KPIs — latency between users and caches, caching error rate, and storage capacity required to serve clients — were measured to assess the end-user impact of CODECO-managed deployment. All tests used scripted client connection simulators and scripted network disturbance generators, applied identically to both deployment contexts.

### 6.3.2 Experimentation Results

The experimentation validated three core capabilities of CODECO in the MDS context. First, edge location selection: CODECO consistently identified deployment points that satisfied combined latency, bandwidth, CPU, and memory constraints, placing caches closer to demand concentrations than the default Kubernetes scheduler. Second, dynamic migration: the system instantiated new caches, scaled existing ones, and migrated services across clusters in response to changing user demand and network conditions, with reaction times faster than the static baseline. Third, continuous adaptation: by integrating NetMA's real-time network view into orchestration decisions, the system avoided the performance degradation characteristic of static deployments when network conditions shifted.

Overall, the results validate that combining network awareness with dynamic service migration significantly improves the performance and resource efficiency of media delivery systems in distributed multi-cluster environments.

## 6.4 Demonstration

### 6.4.1 Technical setup

The demonstration was configured as a three-cluster, ten-node environment spanning Madrid and Athens, as depicted in Figure 19. Clusters were allocated based on network betweenness — one cluster per location — with one cloud node and at least two edge nodes per cluster. Edge nodes comprised two types: Raspberry Pi 5 devices for low-capability edge roles, and virtualised laptops for higher-capability roles, providing deployment variability. The full node scheme is shown in Figure 20.





Figure 19: P3 - Use case deployment map.

Software deployed across the nodes was as follows. Edge nodes hosted MEC APIs, ALTO agents, and edge caches. Cloud nodes hosted caches, an ALTO server, and a MongoDB integration database. Outside the cluster infrastructure, the master CDN resided at the ICOM premises in Athens, and traffic injectors simulated user activity near the edge APIs.

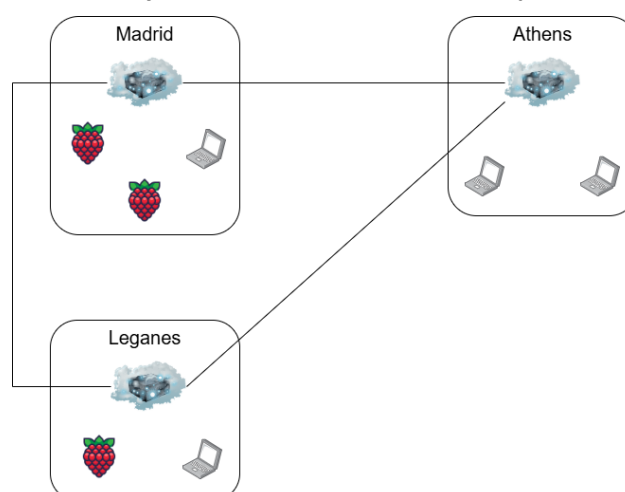


Figure 20: P3 - Use case nodes scheme.

An overview of equipment involved is provided in Table 15. On these nodes, the following software was deployed:

- **Edge:** Edge Caches: To be available in the Multi-Cluster scenario; ALTO agents: To be available in the Multi-Cluster scenario; MEC APIs: already available in the Eclipse Repo.
- **Cloud:** Caches: To be available in the Multi-Cluster scenario; ALTO server: To be available in the Multi-Cluster scenario; MongoDB: integration database to be used to check the metric collection.
- **Outside the nodes:** Main CDN: the master CDN allocated in the ICOM premises in Athens; Traffic injectors: users' simulators to be launched in a local area with the APIs to allow the data collection for the internal logic of the CDN.

Table 15: P3 - Use case equipment characteristics.

| Name       | Device           | OS            |
|------------|------------------|---------------|
| Cloud-Mad1 | Cisco ASR 9000 S | Ubuntu Server |
| Edge1-Mad1 |                  | Ubuntu        |
| Edge2-Mad1 | Raspberry Pi 5C  | Raspberry OS  |
| Edge3-Mad1 | Raspberry Pi 5C  | Raspberry OS  |

| Name       | Device          | OS            |
|------------|-----------------|---------------|
| Cloud-Mad2 |                 | Ubuntu Server |
| Edge1-Mad2 |                 | Ubuntu        |
| Edge2-Mad2 | Raspberry Pi 5C | Raspberry OS  |
| Cloud-Ath  |                 | Ubuntu Server |
| Edge1-Ath  |                 | Ubuntu        |
| Edge2-Ath  |                 | Ubuntu        |

## 6.4.2 Demo Scenarios

Two stakeholder roles were present across all scenarios. **Users** were simulated through traffic generators deployed near edge devices, generating realistic content request patterns. The **Content Provider** — functioning as CODECO's client — was a CDN with internal logic that used CODECO to deploy edge caches intelligently in proximity to users.

Four scenarios were demonstrated, progressing from a stable single-cluster baseline to a complex multi-cluster dynamic environment.

**Demo Scenario A — Stable Operation:** The CDN deploys its resources close to users using CODECO's orchestration capabilities and monitors access under low traffic conditions. The scenario confirms stable operation without connectivity or resource availability issues.

**Actions:** CODECO is deployed on the testbed; once stable, the CDN requests deployment via the CODECO Application Model (CAM); CODECO deploys the use-case components; pod status, API connectivity, and metric availability are verified.

**Demo Scenario B — High Demand in One Cluster:** A traffic generator simulates a surge in user requests to one cluster, increasing Edge Cache resource consumption. CODECO detects the overload and migrates the affected component to a more suitable node.

**Actions:** Traffic generator sends a high volume of content requests; Edge Cache resource usage rises and is detected by CODECO; CODECO migrates the service to a better-placed node; the new deployment becomes active.

**Demo Scenario C — Migration to Another Cluster:** A user changes location, moving from the coverage area of Cluster 1 to that of Cluster 2. The CDN detects that the nearest APIs are now in Cluster 2, updates the CAM to request cache migration, and CODECO executes the migration without service interruption.

**Actions:** A user is activated in Cluster 1; the user ceases sending messages in Cluster 1 and begins in Cluster 2; the CDN modifies the CAM; CODECO applies the migration; the Edge Cache is confirmed active in Cluster 2 with no service loss.

**Demo Scenario D — Multi-Cluster Peak Usage:** Multiple simultaneous load peaks occur across different clusters. CODECO places resources at optimal locations near each peak, while the CDN uses both its own edge probe data and CODECO metrics to issue updated CAM scaling requests. CODECO re-deploys without interrupting service availability.

**Actions:** Starting from a stable multi-cluster deployment, the traffic generator simulates a demand peak in one cluster; the CDN detects increased demand and requests larger resources via the CAM; in parallel, CODECO migrates pods to less-loaded edge nodes; CODECO receives the updated CAM and scales the resources; the service remains available throughout.

### 6.4.3 Risks and Mitigation Measures

An overview of the identified risks and their mitigations is provided in Table 16.

Table 16: P3 - Demo risks and mitigation measures.

| Risk   | Mitigation   |
|--|--|
| Networking issues between Spain and Athens                                       | Simulating the third cluster in one of the Madrid premises and connecting those two - there could be also simulated the three clusters in Athens |
| Problem with access to / modifiability of parts of the code of the ICOM CDN team | Use Open Code CDNs or to simulate it with DBs and traffic generators   |

## 6.5 Summary

The P3 use-case demonstrated how CODECO enables a dynamic Media Delivery System across a decentralised edge-cloud continuum, addressing the fundamental challenge of adaptive, low-latency content delivery in environments where traditional systems cannot efficiently respond to shifting demand and network conditions.

The architecture assigns distinct responsibilities to the MDS provider — content and delivery logic — and to CODECO — deployment, orchestration, and runtime adaptation. MEC APIs at the edge collect user- and service-level metrics, cloud-hosted databases aggregate them, and CODECO-provided infrastructure metrics complete the picture. This separation allows the content provider to operate without managing deployment complexity directly.

Deployment progressed through four phases, from a virtual single-cluster setup to a fully physical multi-cluster environment spanning Madrid and Athens. The progressive approach validated each layer of integration before adding further complexity, with ALTO servers introduced in later phases to extend CODECO's network awareness beyond its internal view through BGP and SDN data.

Evaluation against a baseline Kubernetes deployment confirmed that CODECO reduces inter-component communication volume, improves cache placement in proximity to demand, and reacts more rapidly to both load spikes and network disturbances. The four demonstration scenarios illustrated the full operational range of the system — from stable baseline operation through localised demand spikes, user mobility across clusters, and simultaneous multi-cluster load peaks — validating CODECO's role as an effective orchestration layer for distributed media delivery infrastructure.

## 7 P4: Demand-side Energy

As described in Section 2.4, this pilot implements a decentralised demand-response energy management system across university campuses, demonstrating CODECO's ability to jointly orchestrate computational and networking resources across the IoT-Edge-Cloud continuum in support of real-world sustainability objectives. The sections below detail the use-case architecture, deployment, experimentation, and demonstration.

### 7.1 Use-Case Architecture

The P4 use-case follows a three-layer architecture in which data acquisition, processing, and orchestration operate across edge and cloud environments, as illustrated in Figure 21 and Figure 22.

The **Data Acquisition Layer** comprises IoT sensors distributed across buildings and energy assets, monitoring consumption, grid parameters, and renewable generation in real time. The **Processing Layer** collects this data and feeds it into predictive models executing at the edge or cloud depending on latency and resource requirements. The **Orchestration Layer** is

provided by the CODECO framework, through which ACM, MDM, SWM, NetMA, and PDLC jointly optimise energy usage and task scheduling across the distributed infrastructure.

The use-case was structured around four progressive phases. Phase 1 established a single-building proof-of-concept to validate energy monitoring and forecasting capabilities. Phase 2 expanded the deployment to multiple nodes across the ETSIT campus at UPM. Phase 3 integrated infrastructure across multiple UPM campuses — Moncloa, Sur, and Madrid. Phase 4 constituted a full-scale deployment across all campuses to validate CODECO's scalability in a heterogeneous, real-world environment.

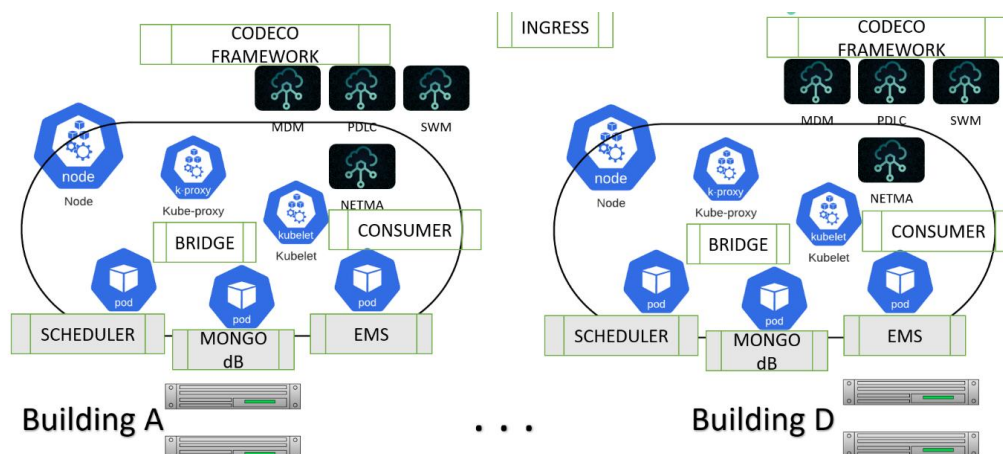


Figure 21: P4 infrastructure across the UPM campus.

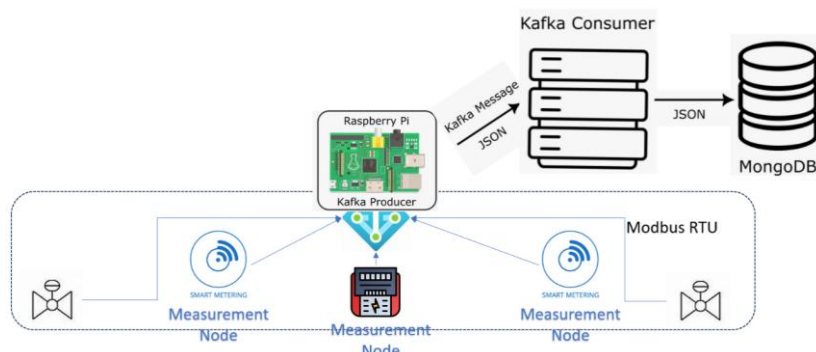


Figure 22: P4 – Use-Case Architecture.

### 7.1.1 Preconditions, Triggers, and Postconditions

**Preconditions:** IoT sensors, communication links, and distributed computing resources must be deployed and initialised across all nodes. Edge nodes and cloud resources must be provisioned for orchestration and processing. Energy data — including consumption profiles, photovoltaic generation, and load data — must be collected and ingested into the CODECO framework before operation begins.

**Triggers:** Three conditions initiate CODECO adaptation. Sudden fluctuations in energy demand trigger dynamic load balancing and task redistribution. Variability in photovoltaic generation activates predictive analytics to adjust resource allocation across the grid. Network or system anomalies trigger fault-tolerant reconfiguration mechanisms.

**Postconditions:** Successful execution is expected to deliver dynamic optimisation of energy consumption with reduced peak load stress; realised efficiency improvements of 5–10%, latency within 3 minutes, and carbon footprint reductions of 5–10%; and seamless scalability to accommodate additional energy assets or nodes without performance degradation.



## 7.1.2 CODECO Integration

**ACM** orchestrates the deployment of energy monitoring nodes and cloud-edge computing resources and supports reconfiguration for scalability and fault-tolerance. **MDM** collects and integrates real-time energy data from IoT sensors, enriching it and feeding insights into predictive models. **PDLC** delivers energy demand and generation forecasts using machine learning without centralising sensitive data, enabling proactive load balancing and task scheduling. **SWM** dynamically redistributes tasks based on current demand and resource availability, with energy-aware scheduling that prioritises low-consumption workloads during off-peak periods. **NetMA** monitors and optimises communication latency between nodes, ensuring reliable real-time data transmission for decision-making.

## 7.1.3 KPIs

The use-case was assessed against technical, sustainability, and resilience KPIs, summarised in Table 17, Table 18, Table 19.

Table 17: P4 – Technical KPIs.

| KPI                              | Target  | Measurement   |
|----------------------------------|---|---|
| <b>Latency</b>                   | Decision-making latency below 3 minutes for edge-to-cloud communication | Time from data ingestion at edge nodes to orchestration decision in the cloud |
| <b>Energy Forecast Accuracy</b>  | Prediction accuracy above 90% for energy demand and PV generation       | RMSE of predictions against actual energy usage data                          |
| <b>Load Balancing Efficiency</b> | Reduce peak energy load by 10%  | Comparison of peak loads before and after dynamic load redistribution         |

Table 18: P4 – Sustainability KPIs.

| KPI                               | Target                                       | Measurement  |
|-----------------------------------|--|--|
| <b>Energy Efficiency</b>          | 5–10% reduction in energy consumption        | Total energy consumption before and after CODECO deployment    |
| <b>Carbon Emissions Reduction</b> | 5–10% reduction in CO <sub>2</sub> emissions | Carbon footprint calculations based on energy consumption data |

Table 19: P4 – resilience KPIs.

| KPI                    | Target   | Measurement  |
|------------------------|--|--|
| <b>Fault Tolerance</b> | Recovery from node failures within 2 minutes                         | Downtime and recovery time after simulated node or network failures            |
| <b>Scalability</b>     | Maintain latency below 3 min and efficiency above 90% as nodes scale | KPI validation under incremental scaling from single to multi-node deployments |

## 7.2 Lab Deployment

### 7.2.1 Deployment Phases

The deployment followed the four-phase structure described in Section 7.1, progressing from a controlled single-building setup to full multi-campus scale.

In **Phase 1**, a single building at UPM served as the pilot environment. IoT sensors collected real-time data from general utilities — HVAC and lighting — and from specialised energy assets including photovoltaic panels and battery systems. The phase validated real-time data ingestion, energy forecasting, and dynamic load optimisation within a controlled and repeatable setting.

In **Phase 2**, the deployment expanded to four buildings within the ETSIT campus at UPM. Each building integrated energy monitoring nodes and edge resources for local processing. Network performance and latency were measured to confirm seamless communication across nodes, and real-time orchestration and optimisation were validated at the edge.

In **Phase 4**, all nodes across UPM campuses were integrated into a single distributed network of energy assets. Each campus contributed data processed locally at the edge, while CODECO's resilience and fault-tolerant mechanisms were validated under large-scale, multi-cluster conditions. Testing focused on orchestration efficiency at scale, system behaviour under node and network failures, and measurement of energy efficiency and CO<sub>2</sub> reduction KPIs.

Two integration challenges required custom software development. First, CODECO does not natively process heterogeneous energy data from multiple sources. Custom pre-processing pipelines were developed to normalise and standardise data across sources before ingestion into MDM, along with an intermediary data integration layer to bridge the use-case data sources and CODECO's orchestration tools. Second, CODECO does not account for domain-specific energy management constraints such as regulatory requirements, peak load tariffs, or grid stability thresholds. Domain-specific algorithms and energy-event triggers were developed and provided to the framework.

```
~ — gatv@raspberrypi: ~/Codeco — ssh gatv@192.168.0.22 +
```

```
RETURN: 19  
Data receivedgatv@raspberrypi:~/Codeco $ ./kafka_test_flask2 192.168.0.18:9092 E  
1 /dev/ttyS0 0 19 s29
```

```
SELECT MODE --> Slave address = 29  
Opening /dev/ttyS0 at 9600 bauds (N, 8, 1)  
[1D][04][00][00][00][13][B3][9B]  
Waiting for a confirmation...  
<1D><04><26><59><E3><00><00><00><00><00><00><00><00><00><00><00><00>  
<00><00><00><00><03><00><00><00><00><02><23><03><E7><FC><40><C3><59><B1><D7><0F>  
<B4><BB><AF>
```

```
RETURN: 19  
Data receivedgatv@raspberrypi:~/Codeco $ ./kafka_test_flask2 192.168.0.18:9092 E  
1 /dev/ttyS0 0 19 s29
```

```
SELECT MODE --> Slave address = 29  
Opening /dev/ttyS0 at 9600 bauds (N, 8, 1)  
[1D][04][00][00][00][13][B3][9B]  
Waiting for a confirmation...  
<1D><04><26><59><E3><00><00><00><00><00><00><00><00><00><00><00><00>  
<00><00><00><00><03><00><00><00><00><02><23><03><E7><FC><40><C3><59><B1><D7><0F>  
<B4><BB><AF>
```

```
RETURN: 19  
Data receivedgatv@raspberrypi:~/Codeco $ █
```

The second, also on the Raspberry Pi, is a Python Flask server that receives JSON energy measurements and converts them into Kafka messages for the Kafka producer (Figure 24).



```

~ — gatv@raspberrypi: ~/Codeco — ssh gatv@192.168.0.22
gatv@raspberrypi:~/Codeco$ python3 local_server.py
* Serving Flask app 'local_server'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [31/Jul/2024 13:14:49] "POST /data HTTP/1.1" 200 -
127.0.0.1 - - [31/Jul/2024 13:14:49] "POST /data HTTP/1.1" 200 -
127.0.0.1 - - [31/Jul/2024 13:14:49] "POST /data HTTP/1.1" 200 -
127.0.0.1 - - [31/Jul/2024 13:15:04] "POST /data HTTP/1.1" 200 -
Data sent to Kafka. Topic: EnergyEdificio1, Partition: 0, Offset: 3
Data sent to Kafka. Topic: EnergyEdificio1, Partition: 0, Offset: 4
Data sent to Kafka. Topic: EnergyEdificio1, Partition: 0, Offset: 5
Data sent to Kafka. Topic: EnergyEdificio1, Partition: 0, Offset: 6
127.0.0.1 - - [31/Jul/2024 13:15:07] "POST /data HTTP/1.1" 200 -

```

Figure 24: P4 – Kafka Producer.

The third, deployed in a Kubernetes pod, is a Python data logger that receives Kafka messages and produces two outputs: one to a MongoDB database and one to a Prometheus server for monitoring (Figure 25).

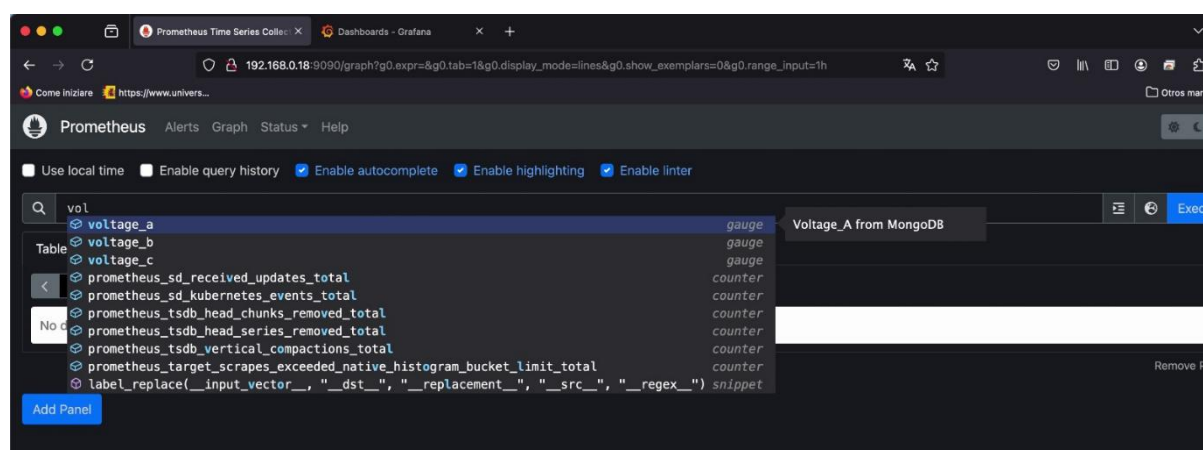


Figure 25: P4 – Energy Measures Integrated in Prometheus.

## 7.3 Experimentation and Evaluation

### 7.3.1 Evaluation Methodology

The evaluation validated CODECO's integration across three dimensions. **Real-time testing and monitoring** assessed CODECO's ability to collect, process, and respond to real-time energy data, measuring latency from data acquisition to actionable decision and validating the accuracy of photovoltaic generation and demand forecasting models. **Incremental scalability testing** measured system performance — latency, energy optimisation, and fault tolerance — as the deployment grew from a single node to multi-node environments, confirming orchestration stability under heterogeneous workloads and variable renewable inputs. **Comparative benchmarking** compared CODECO's performance against a centralised energy management baseline, assessing improvements in energy efficiency, carbon footprint reduction, and decision-making latency under stress conditions such as peak demand.

The validation environment comprised Raspberry Pi edge nodes and industrial edge servers installed across UPM buildings, Kubernetes-based cloud orchestration handling workloads, predictive analytics, and global optimisation, and IoT sensors monitoring power consumption,

PV output, and market pricing data. Three testing scenarios were executed on this infrastructure, as summarised in Table 20.

*Table 20: P4 testing scenarios.*

| Scenario                            | Description  | Validation Focus  |
|-------------------------------------|--|---|
| <b>Real-Time Demand Response</b>    | Sudden HVAC usage spike causing a step increase in energy demand | System ability to predict demand and redistribute loads efficiently           |
| <b>Renewable Energy Integration</b> | Variable PV generation due to changing weather conditions        | Predictive model accuracy and scheduling adaptation to renewable availability |
| <b>Fault Simulation</b>             | Node or network failure  | Recovery time, system stability, and effectiveness of redundancy mechanisms   |

### 7.3.2 Evaluation Metrics and Outcomes

CODECO's performance was compared against a traditional centralised energy management baseline across the metrics in Table 21, with outcomes reported in Table 22.

*Table 21: P4 – Metrics for Comparison.*

| Metric                         | Description   |
|--------------------------------|---|
| <b>Energy Efficiency Gains</b> | Improvement in energy usage optimization compared to centralized systems (based on percentage). |
| <b>System Responsiveness</b>   | Reduction in decision-making latency during real-time energy events.                            |
| <b>Resilience</b>              | Time to recover from node failures compared to baseline recovery times.                         |

*Table 22: P4 – Outcomes.*

| Outcome                            | Description  |
|------------------------------------|--|
| <b>Quantified Energy Savings</b>   | Report energy efficiency improvements and carbon footprint reductions.                 |
| <b>System Scalability Insights</b> | Validate CODECO's performance under scaled deployments.                                |
| <b>Real-Time Effectiveness</b>     | Demonstrate CODECO's ability to handle real-time energy dynamics with minimal latency. |
| <b>Fault Tolerance Proof</b>       | Highlight system robustness through recovery metrics.                                  |

### 7.3.3 Experimentation Results

The experimentation was conducted across the UPM campus, with IoT sensors monitoring photovoltaic generation, HVAC and lighting consumption, electricity prices, and CO<sub>2</sub> emissions. Two experimental scenarios were executed.

In the **single-building scenario**, workloads were adjusted in real time to prioritise energy efficiency and renewable source usage. MDM aggregated sensor data to provide a unified system view; PDLC applied predictive models to forecast demand and guide decisions; SWM performed dynamic scheduling to optimise resource usage; NetMA ensured reliable communication across nodes.

In the **multi-building scenario**, distributed nodes across the ETSIT campus collaborated to balance energy loads and respond to simulated faults. Coordination mechanisms enabled real-time data sharing through MDM and joint optimisation through ACM, PDLC, and SWM.

Results confirmed that CODECO enables effective, adaptive energy management across distributed environments. The system reduced peak loads, improved energy efficiency, and supported renewable integration. Combining real-time monitoring with predictive analytics produced more stable and efficient operation compared to static approaches. Identified challenges included system complexity under heterogeneous device integration and the

coordination overhead in multi-campus scenarios, both of which remained within acceptable bounds.

## 7.4 Demonstration

### 7.4.1 Technical setup

The P4 demonstration took place at the ETSIT Campus of the Universidad Politécnica de Madrid, spanning buildings A, B, C, and D, as shown in Figure 26. The following equipment was deployed:

- IoT sensors monitoring real-time energy consumption, PV generation, battery levels, and HVAC performance
- Raspberry Pi devices and industrial edge servers as energy management edge nodes
- Wi-Fi and Ethernet network infrastructure for low-latency inter-node communication
- Kubernetes-based cloud services for orchestration and predictive analytics

Applications running on the demonstration infrastructure included energy monitoring dashboards for real-time data acquisition, predictive modelling tools for energy demand and PV generation forecasting, dynamic scheduling and load balancing algorithms, and visualisation dashboards displaying energy usage, optimisation results, and KPI status.

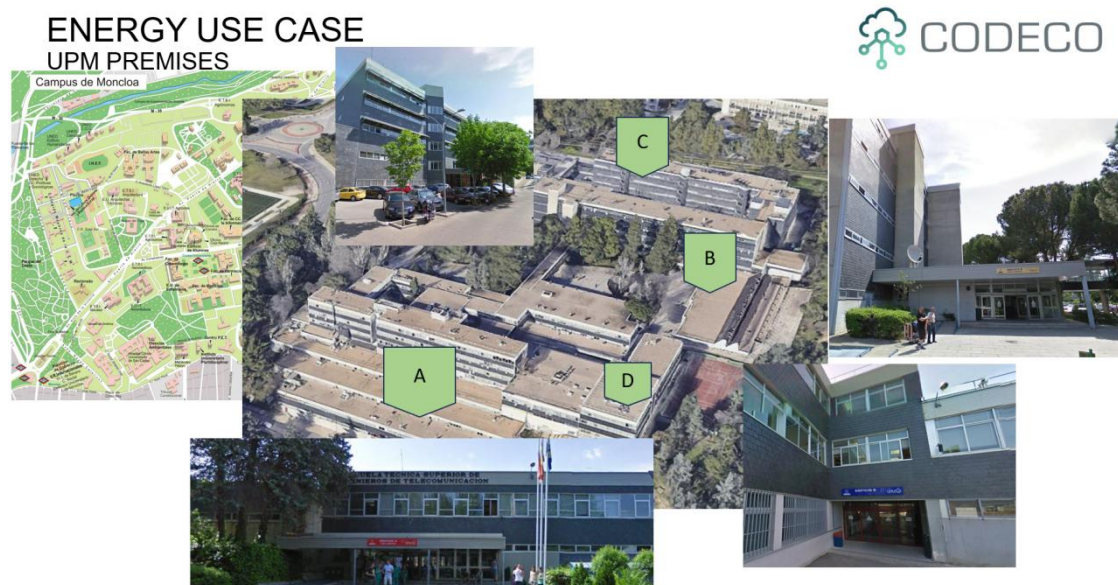


Figure 26: P4 - Overview of the demonstration location.

### 7.4.2 Demo Scenarios

Two scenarios were demonstrated, corresponding to the single-building and multi-building deployment phases.

**Demo Scenario A — Single-Building Energy Optimisation:** This scenario illustrated CODECO's ability to optimise energy usage in real time within a single building. IoT sensors captured consumption data from HVAC systems and lighting. MDM integrated the data into the system; PDLC made intelligent decisions to minimise the energy consumption of the computing and communications infrastructure supporting the energy management system; SWM executed the necessary workload migrations to achieve energy optimisation and

prioritise green energy usage. Results were displayed to observers via the use-case dashboard, showing reduced peak loads and improved energy efficiency.

**Demo Scenario B — Multi-Building Coordination:** In this scenario, buildings A, B, C, and D across the ETSIT campus shared energy data through MDM and collaborated to achieve distributed energy management. ACM orchestrated global optimisation, SWM dynamically reallocated workloads, and NetMA ensured uninterrupted inter-node communication. PDLC provided demand forecasts for the full multi-building system. A simulated node fault triggered CODECO's resilience mechanisms, demonstrating recovery within the 2-minute target. KPIs including fault tolerance, scalability, and carbon emissions reduction were presented to validate system performance.

### 7.4.3 Demo Event

The demonstration event took place in 2025 Q3 (M34-36 of the project timeline), collocated with the European Researchers' Night in Madrid, as described in Table 23. In addition, a short demonstration of the use case was given during the European Researchers' Night exhibition.

Table 23: P4 - Event Overview.

| Date / Time       | September 26, 2025  |
|-------------------|---|
| <b>Organizers</b> | UPM, with technical support from other CODECO partners  |
| <b>Audience</b>   | UPM administration, research faculty, students, and external stakeholders   |
| <b>Duration</b>   | 2 hours   |
| <b>Content</b>    | Overview of CODECO and the P4 use case, live demonstration of demo scenario A, Q&A session. Participants engaged with the demonstration through interactive use-case specific dashboards, question-and-answer sessions, and structured feedback interviews. |

## 7.5 Risks and Mitigation Measures

For this use case, the risks and mitigation measures were identified as specified in Table 24.

Table 24: P4 - Demo risks and mitigation measures.

| Risk   | Mitigation   |
|--|--|
| <b>No permission from EU Researchers' Night organization committee</b> | Organize the event separately at a later stage           |
| <b>Hardware availability</b>   | Order equipment well in advance including backup devices |
| <b>Node failure during demonstration</b>                               | Deploy redundant nodes for failover scenarios            |
| <b>Network disruptions</b>   | Use local storage for critical data                      |
| <b>Inaccurate energy forecasts</b>                                     | Perform extensive model validation before the event      |
| <b>Stakeholder scheduling conflicts</b>                                | Offer alternative event times to ensure participation    |

## 7.6 Summary

The P4 use-case demonstrated how CODECO supports decentralised energy management across a distributed IoT-Edge-Cloud environment, targeting measurable improvements in energy efficiency, carbon emission reduction, and low-latency decision-making.

The three-layer architecture — data acquisition via IoT sensors, local processing at the edge, and higher-level coordination in the cloud — was orchestrated by CODECO's full component set. ACM managed deployment and reconfiguration; MDM aggregated and enriched real-time sensor data; PDLC provided privacy-preserving demand and generation forecasts; SWM performed energy-aware workload scheduling; and NetMA ensured reliable, low-latency communication across nodes and campuses.



Deployment progressed through four phases, from a single building to a full multi-campus environment spanning three UPM sites. This incremental approach validated CODECO's ability to handle increasing architectural complexity, maintain performance targets, and sustain resilience across heterogeneous infrastructure. Custom software — including Modbus-based sensor integration, a Kafka messaging pipeline, and a Prometheus data logger — was developed to bridge the gap between the energy domain's specific data formats and CODECO's orchestration layer.

Experimentation confirmed that CODECO reduces peak energy loads, improves efficiency, and enables proactive adaptation to demand fluctuations, renewable variability, and system faults, outperforming a centralised baseline across all three evaluation dimensions. The demonstration at the European Researchers' Night validated these capabilities in a live public setting, with both single-building and multi-building scenarios successfully executed.

## 8 P5: AMR and Manufacturing

As described in Section 2.5, this pilot demonstrates how CODECO can serve as the orchestration layer for decentralised, resilient AMR fleet control in a flexible factory setting, addressing the challenges of wireless connectivity, limited embedded resources, and dynamic operational conditions. The sections below detail the use-case architecture, deployment, experimentation, and demonstration.

### 8.1 Use case Architecture

The P5 use-case is structured around two user journeys that together cover the full lifecycle of a robotics application in an AMR fleet: the initial deployment of containerised microservices across the fleet, and the runtime management of those services under dynamic and adverse conditions.

The overall system, illustrated in Figure 27, assumes that a robotics application workload for AMR fleets is available, that an AMR fleet of ROS2 devices is operational and wirelessly interconnected, and that CODECO is installed and running on the cluster. Each AMR is treated as a Kubernetes worker node, with the AMR controller co-located with the CODECO control plane on the K8s master node.

CODECO is triggered to act when one or more AMR experience intermittent connectivity, approach battery depletion, or lack the resources necessary to complete a specific task.

**User Journey 1 — Microservice Deployment (DEV):** A developer wishing to deploy a robotics application across the fleet accesses the CODECO ACM interface via the use-case dashboard. The dashboard allows the user to select a pre-defined set of microservices — including a publish-subscribe communication layer (MQTT Sparkplug or NDN), a task handler, and an object detection service — and to specify deployment requirements such as latency targets, fleet size, communication type, and channel properties. These parameters are used to construct the CODECO CAM as a YAML-formatted CRD. SWM then performs the initial placement, deploying one pod per worker node across all nodes within radio range of the controller. NetMA handles wireless path optimisation and interference mitigation throughout. The user monitors the deployment state and cluster topology through the dashboard.

**User Journey 2 — Runtime Management (MGT):** This journey addresses the runtime management of deployed applications, with a focus on autonomous AMR navigation in indoor environments under conditions of energy constraint and intermittent connectivity. Two approaches were considered. In Phase 1, a centralised approach was used: the controller maintained a global view of the K8s infrastructure, continuously updated by CODECO operators. PDLC analysed infrastructure robustness, energy consumption across node pairs, channel conditions, and RTT, and issued adaptation recommendations to SWM. If an AMR

was predicted to run out of battery, its microservices were automatically migrated to another suitable AMR selected by CODECO based on the CAM requirements. In Phase 2, a decentralised approach was adopted: each AMR transmitted its own perspective of the infrastructure to peers, and the control plane was extended to a multi-master cluster with three master nodes selected by NetMA to maximise resilience.

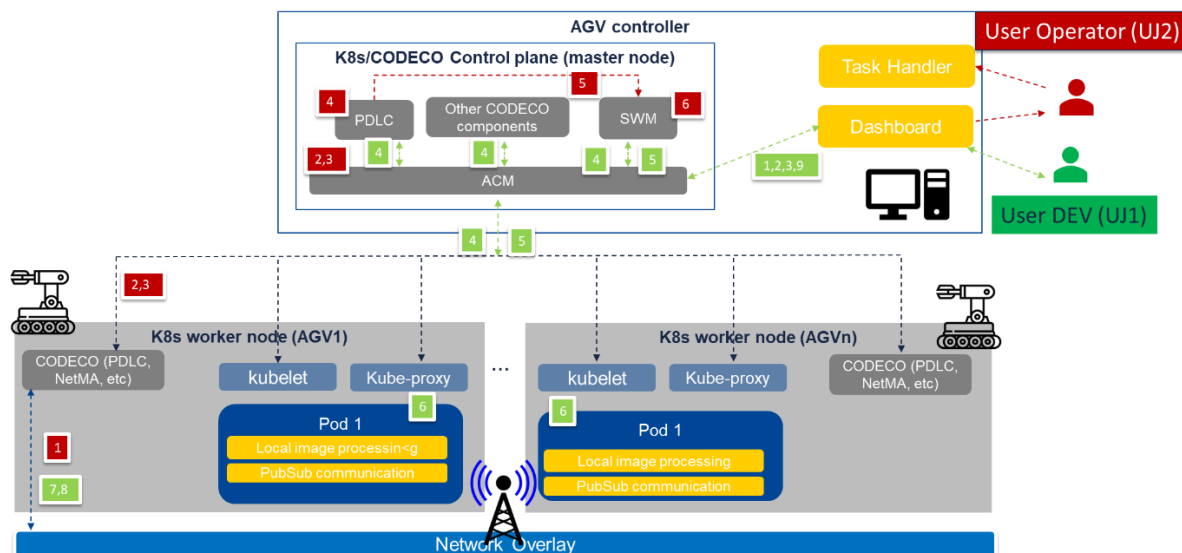


Figure 27: P5 – System Architecture, One Cluster.

### 8.1.1 Preconditions, Triggers, and Postconditions

**Preconditions:** The K8s cluster must be operational and encompass all actors, with at least one active master node and the required worker nodes representing AMRs. All nodes must be remotely accessible and correctly integrated into the CODECO framework. The AMR fleet must be wirelessly interconnected with stable topology and network conditions. The robotics application workload must be available and the *CODECO Application Model* (CAM) configured with relevant fleet parameters, for instance, latency and bandwidth, CPU and memory, ROS2 topics, before deployment begins.

**Triggers:** Three conditions initiate CODECO adaptation. Intermittent or degraded wireless connectivity on one or more AMRs triggers network path optimisation and workload reallocation. Battery depletion on an AMR triggers stateful migration of its active microservices — including accumulated SLAM map data — to another suitable node selected automatically by CODECO. Resource exhaustion, such as insufficient memory or compute capacity to sustain a running microservice, triggers re-scheduling to a node with the required availability.

**Postconditions:** Successful execution is expected to deliver automated deployment of the robotics application across the fleet with maintained task completion and navigation accuracy following stateful workload migration; and sustained system availability under node failure, battery depletion, and connectivity disruption.

### 8.1.2 CODECO Integration

**ACM** manages the automated deployment and reconfiguration of the robotics application workload across the AMR fleet, translating the CAM into concrete pod placements and handling cluster expansion as new AMRs come within radio range. Node resource availability, battery levels are provided via ACM, and network conditions via **NetMA**. **PDLC** generates recommendations concerning greenness or other target profiles requested by the user issuing proactive placement recommendations to **SWM** before performance degrades. **SWM** executes initial workload placement and dynamic reallocation, including stateful migration of



SLAM workloads between AMRs with map data continuity ensured via persistent storage. **NetMA** monitors the networking conditions, collecting link quality metrics and triggering route optimisation and interference mitigation to maintain the communication stability required for real-time robotic control.

### 8.1.3 KPIs

The use-case was validated based on the KPIs proposed in Table 25.

Table 25: P5 – KPIs.

| KPI       | Description  | Sub-KPIs   |
|-----------|--|--|
| <b>K1</b> | Operational: scalability, failure reduction, latency | K1.1: $\geq 3$ nodes per cluster; 10 total nodes; $\geq 3$ clusters in federated operation. K1.2: Task completion time. K1.3: $\sim 10\%$ reduction in setup time.                                     |
| <b>K2</b> | Strategic: performance, energy, resilience           | K2.1: $\sim 10\%$ collision reduction. K2.2: $\sim 10\%$ network lifetime increase. K2.3: $\sim 10\%$ energy consumption reduction. K2.4: $\sim 10\%$ reduction in application setup message overhead. |

## 8.2 Lab Deployment

### 8.2.1 Infrastructure

The fortiss IIoT Lab CODECO cluster comprised one K8s master node, a Lenovo ThinkPad X280 laptop, and eight worker nodes: two TurtleBot3 mobile robots and six Raspberry Pi 4 devices. For the purposes of this use-case, only the TurtleBots and the controller node were used, selected for their cost-effectiveness, modularity, ROS2 compatibility, and compact form factor. The node configuration is shown in Table 26. All devices were interconnected via Wi-Fi (802.11bg) using private IP addressing.

Table 26: P5 – Basic Node Configuration.

| Node                | Operating System                                  | RAM     |
|---------------------|---|---------|
| <b>Controller</b>   | Ubuntu 22.04 Long Term Support (LTS), kernel 5.15 | 15 GiB  |
| <b>Worker nodes</b> | Ubuntu server 22.04 LTS, kernel 5.15              | 7.6 GiB |

### 8.2.2 Software Customization

For the proposed use-case, an analysis of existing AMR fleet management tools has been developed. To create an application that could assist the validation of CODECO, creating a proof-of-concept that could serve the analysis of the capability to handle stateful migration, a specific application has been developed for autonomous navigation of AMRs composed of different micro-services: **bringup**; **Simultaneous Localization and Mapping (SLAM)**; **Navigation** which are available via the CODECO Gitlab<sup>7</sup>.

The **bringup** micro-service is responsible for initializing the AMRs core systems. This includes setting up the communication interfaces, starting the necessary ROS2 nodes, and configuring sensors and actuators. It handles the initialization of hardware components, ensuring that all sensors (e.g., LiDAR, cameras), motors (e.g., wheels), and other hardware components are properly connected and operational. It starts the core ROS2 packages and establishes reliable communication between the worker nodes and controller, for remote control and visualization.

**SLAM** is used for autonomous driving, e.g., to assist robots to explore unknown environments and to be able to build a map by collecting the data from existing sensors. The Google

<sup>7</sup> <https://gitlab.eclipse.org/eclipse-research-labs/codeco-project/use-cases/p5-amr-manufacturing/-/tree/main/docker/>

Cartographer<sup>8</sup> is a widely well-known SLAM algorithm, which is also compatible with ROS2. In this work, the AMRs (TurtleBots) lack the processing power necessary for SLAM, so a local Laptop, with the same IP subnet, is used instead for the required SLAM intensive computation. Each AMR handles the bringup process, including motor control and sensor data acquisition, within a Docker container. Data transmission between the AMR and the controller node (laptop) is handled via Data Distribution Service (DDS), which handles the real-time exchange of sensor data and SLAM results. In this setup, the AMR publishes sensor data such as laser scans and odometry via ROS2 topics. These topics are transmitted over the network using DDS to the controller node, where the Cartographer SLAM algorithm runs in a separate Docker container. The controller node subscribes to the sensor data topics and performs the SLAM computations, publishing the resulting map and localization data back to the AMR. The SLAM Docker image<sup>9</sup> is also available as open-source and uploaded to the CODECO

Dockerhub repository. The **Navigation** micro-service is based on the Navigation System [6] (Nav2<sup>10</sup>), a control system that allows an AMR to autonomously reach a specified goal state, such as a particular position and orientation on a given map. Starting from its current position and using the map and target destination, the navigation system generates a plan to reach the goal. It then issues commands to autonomously guide the robot, ensuring adherence to safety constraints and avoiding any obstacles encountered along the way. Running the Nav2 algorithm for navigation on an AMR with ROS2 using Docker involves a collaborative setup between the AMR and the local laptop which behaves as fleet controller. The controller node subscribes to the topics published by AMR bringup-services, processes the data to build a map using SLAM, and then runs the Nav2 stack for navigation. The Nav2 stack on the controller node consists of several key nodes, including *amcl* for localization, *planner\_server* for path planning, *controller\_server* for following the planned path, and *bt\_navigator* to manage the navigation behaviour tree. The AMR publishes its sensor data, which the controller node uses to update the map and localize the AMR. The controller node then sends velocity commands back to the AMR to move it along the planned path. This setup allows for efficient and accurate navigation, leveraging the computational power of the controller node while keeping the AMR's processing load manageable. Based on Navigation, an AMR can be given a script that defines the locations to be reached (fixed poses script) and then the AMRs will follow these poses moving. This is called Waypoint Follower<sup>11</sup>. To maintain continuous operation, especially in the face of failures such as power depletion, node failure, or high latency, the system can migrate the SLAM workload from one (1) AMR to another. For this specific work, the use of the k8s Persistent Volume<sup>12</sup> (PV) and Persistent Volume Claim (PVC) has been considered, to share data. PV provides a persistent storage solution that can be accessed by any node in the cluster, while PVC ensures that the necessary data, such as the current map, is available to a new selected node which can take over the SLAM task(s) of a node that is running out of battery or experiencing any other triggering condition for CODECO.

A **dashboard**, illustrated in Figure 28, has been deployed in the controller node to assist in the interaction and experimentation. Via the dashboard, users can deploy workloads, monitor the status of nodes and running pods within the cluster effectively. This comprehensive view allows users to oversee the health and performance of the cluster, ensuring that all components are functioning as expected.

<sup>8</sup> <https://google-cartographer-ros.readthedocs.io/en/latest/>

<sup>9</sup> <https://hub.docker.com/r/hecodeco/uc-p5-slam>

<sup>10</sup> <https://docs.nav2.org/>

<sup>11</sup> <https://docs.nav2.org/configuration/packages/configuring-waypoint-follower.html>

<sup>12</sup> <https://kubernetes.io/docs/concepts/storage/persistent-volumes/>



Figure 28: P5 – Dashboard for Control of the AMRs.

## 8.3 Experimentation and Evaluation

### 8.3.1 Evaluation Methodology

The evaluation was structured around four validation objectives, each verified in the fortiss IIoT Lab and, where applicable, in a KinD emulation environment.

**Operational setup control** confirmed that all preconditions were met before experimentation: at least one active cluster encompassing all actors was operational, the cluster was remotely accessible via SSH, AMRs could be dynamically added or removed, and topology and networking conditions were stable.

Two experimental environments were used. The **KinD environment** emulated the FOR CODECO cluster for reproducible benchmarking. After setting the topology, the configuration involved deploying a *Container Network Interface (CNI)* to enable communication between the

cluster nodes. Weave Net2<sup>13</sup> was selected as the CNI due to its capability to handle dynamic, multi-host networking and provide robust network management across the cluster. In CODECO, the default CNI is Multus<sup>14</sup>. Prometheus<sup>15</sup> was deployed as metrics server (requirement in CODECO). Furthermore, Grafana<sup>16</sup> was deployed for monitoring the performance and health of the K8s cluster. Prometheus gets, via CODECO, monitoring of resources at a node and networking level from worker nodes and from the application workload (application, pod and node level). Grafana was used to visualize the Prometheus metrics, providing a user-friendly interface for monitoring the performance and operational status of the cluster.

**fortiss IIoT Lab CODECO testbed:** After the validation of the ROS2 system within the K8s environment, the next step was to evaluate the performance of the overall system for the proposed application workload, when considering CODECO and specific trigger conditions for CODECO to act (e.g., node has no battery). The first version of CODECO, the "CODECO Basic Operation Components and Toolkit version 2.0" [8] has been used and installed following the steps described in the accompanied report. The different CODECO components deployed in their respective `he-codeco` namespaces: `he-codeco-acm`, `he-codeco-netma`, `he-codeco-mdm`, `he-codeco-pdlc`, and `he-codeco-swm`.

**Application workload control** assessed the accuracy and repeatability of AMR navigation using SLAM, evaluating whether the correct location was reached within acceptable position error, whether accuracy was repeatable across multiple runs, and whether accuracy was maintained across varying space sizes.

**Deployment evaluation** (User Journey 1) measured the performance of CODECO-managed microservice deployment against a baseline K8s deployment, focusing on application setup time, messaging overhead, and energy consumed. Scenario variability included different node capabilities, application sizes, dataset sizes, and battery levels.

**Runtime management evaluation** (User Journey 2) assessed CODECO's behaviour during autonomous navigation under adverse conditions, measuring energy consumption, RTT between nodes, task completion latency, and re-scheduling latency when a battery-depleted AMR triggered workload migration.

## 8.3.2 Experimentation Results

The experimentation compared standard K8s against K8s augmented with CODECO across five metrics: CPU usage, memory usage, pod deployment time, pod deletion time, and inter-pod data rates. All experiments were conducted in a KinD environment on a Lenovo ThinkPad T470p with an Intel Core i7-7700HQ processor, 16 GiB RAM, running Ubuntu 22.04 LTS, with each experiment repeated five times. Prometheus and Grafana were deployed for telemetry collection and real-time visualisation. The full results are reported in [11].

**CPU usage.** CODECO consistently produced lower CPU consumption than vanilla Kubernetes across both talker and listener pods. CPU usage was measured as the per-second instant rate over a one-minute window for each container. The reduction is attributed to finer-grained workload placement and reduced background orchestration activity, though further profiling is planned to better isolate the contributing factors. Lower CPU consumption directly supports more sustainable operation of resource-constrained AMRs and is relevant to KPI K2.3.

---

<sup>13</sup> <https://github.com/weaveworks/weave>

<sup>14</sup> <https://github.com/k8snetworkplumbingwg/multus-cni>

<sup>15</sup> <https://github.com/prometheus-operator/kube-prometheus>

<sup>16</sup> <https://grafana.com/>

**Memory usage.** CODECO introduced a memory overhead of approximately 10–15% relative to baseline Kubernetes — for example, 63 MiB versus 54 MiB per pod. This increase reflects the additional services required for telemetry collection, policy management, and migration state handling. The overhead is considered modest in Kubernetes terms and acceptable relative to the orchestration capabilities it enables.

**Communication stability.** CODECO produced substantially more stable and lower transmit and receive data rates than Kubernetes. Kubernetes exhibited bursty behaviour with receive rates exceeding 2–3 Gbps in peaks, whereas CODECO maintained lower, more predictable ranges throughout. This behaviour is consistent with CODECO's QoS-aware network control, which constrains unnecessary traffic bursts and smooths inter-pod communication in alignment with configured performance profiles. For real-time AMR workloads operating over constrained wireless links, communication predictability is operationally more important than peak throughput, and this result directly supports KPI K2.2 and K2.4.

**Pod lifecycle latency.** Kubernetes performed both deployment and deletion operations faster than CODECO — approximately 2 seconds versus 5 seconds in each case. The additional time under CODECO stems from the initialisation and teardown of the VXLAN-based L2S-M secure overlay, which establishes encrypted pod-to-pod connectivity across distributed nodes. Both values are small in absolute terms given that Kubernetes lifecycles typically operate on the order of minutes. For most AMR workflows this overhead is acceptable, though optimisation may be warranted for latency-critical swarm operations. As noted, KPI K1.3 — targeting a 10% reduction in overall setup time — was not met under the current emulation conditions and remains subject to re-evaluation in real-device experiments.

**Stateful workload migration.** The experiments confirmed that CODECO can detect triggering conditions — such as battery depletion or resource exhaustion on a node — and initiate stateful migration of SLAM workloads between nodes, with Kubernetes Persistent Volumes ensuring map data continuity across the handover. This validates the core migration mechanism at the orchestration level and supports uninterrupted mission execution under dynamic conditions. Full validation of migration continuity on physical TurtleBot hardware under real wireless and mobility conditions is ongoing and is part of the planned real-device validation phase.

**Overall assessment.** The results reveal a clear and consistent set of trade-offs. Kubernetes provides faster pod lifecycle operations and marginally lower memory usage, reflecting its simpler, infrastructure-focused design. CODECO, by contrast, delivers lower CPU utilisation, more stable and controlled communication patterns, and resilient stateful migration support — properties that are directly relevant to mobile, resource-constrained AMR deployments where predictable compute and network performance, energy awareness, and operational continuity under failures are primary requirements. As concluded in [11], these trade-offs favour CODECO in dynamic Edge AMR scenarios, particularly those involving communication guarantees, energy-constrained scheduling, and continuous operation under wireless variability.

### 8.3.3 Evaluation Summary

The assessment maps the experimental results reported in [11] against the P5 KPIs defined in Table 27. Where a KPI was not addressed in the current experimental phase, the reason is stated and the planned validation path is noted.

*Table 27: P5 KPI coverage assessment.*

| KPI  | Description   | Status    | Evidence and Assessment  |
|------|---|-----------|--|
| K1.1 | Minimum 3 nodes per cluster; 10 total nodes; at least 3 | Addressed | The testbed deployment has been demonstrated with 7 nodes per cluster and involving three clusters in federated operation.<br><b>Evidence:</b> videos in YouTube, demo camp. |



| KPI         | Description   | Status                          | Evidence and Assessment  |
|-------------|---|---------------------------------|--|
|             | clusters in federated operation                         |                                 |  |
| <b>K1.2</b> | Time to completion of a task                            | <b>Partially addressed</b>      | Pod deployment time was measured at approximately 5 seconds under CODECO versus 2 seconds under Kubernetes. This reflects orchestration-level lifecycle latency rather than end-to-end AMR task completion time. Full task completion time on physical robots navigating to defined waypoints is verified at the fortiss IIoT Lab and via YouTube demonstration videos; publication is under development.  |
| <b>K1.3</b> | Reduce overall setup time by ~10%                       | <b>Not met in current phase</b> | Pod deployment under CODECO takes approximately 5 seconds compared to 2 seconds under Kubernetes, representing a slowdown rather than a reduction. This overhead is attributable to L2S-M secure overlay initialisation. The KPI target is not achieved in the emulation environment and will be re-evaluated at scale in real-device experiments, where CODECO's placement intelligence is expected to reduce total application setup time across larger fleets.  |
| <b>K2.1</b> | Reduce number of collisions by ~10%                     | <b>Not yet assessed</b>         | Collision behaviour requires physical robot interaction in a real environment. The KinD experiments do not include real mobility or robot-to-robot proximity scenarios. This KPI will be addressed in the real-device validation phase using physical TurtleBot robots in the fortiss IIoT Lab.  |
| <b>K2.2</b> | Network lifetime improvement                            | <b>Addressed</b>                | CODECO demonstrated substantially more stable and lower inter-pod data rates than Kubernetes, with Kubernetes exhibiting receive rate peaks exceeding 2–3 Gbps while CODECO maintained controlled, predictable ranges. This QoS-aware traffic management reduces unnecessary link load and is consistent with extended network operational lifetime. The concrete quantification (originally planned for 10%) is not yet available.  |
| <b>K2.3</b> | Reduce overall energy consumption                       | <b>Addressed</b>                | CODECO consistently produced lower CPU consumption than Kubernetes across all measured pods, with reduced background orchestration activity identified as a contributing factor. Lower CPU utilisation is a meaningful proxy for reduced energy expenditure on embedded AMR hardware, where compute and power are tightly coupled. Direct energy measurement — in joules or watt-hours — was not performed in [11] and is currently ongoing for real-device experimentation with embedded platforms.   |
| <b>K2.4</b> | Reduce message exchange required to set up applications | <b>Addressed</b>                | CODECO produced lower and more stable transmit data rates compared to Kubernetes, which exhibited higher peaks and burst behaviour during the same workload. This is consistent with reduced unnecessary messaging, as CODECO's QoS-aware control constrains inter-pod traffic to what is required by the configured performance profile. A precise quantification of messaging reduction against the 10% target requires controlled application-level message counting across repeated deployments, planned as part of the next validation phase. |



## 8.4 Demonstration

### 8.4.1 Technical Setup

The P5 demonstrations took place at the fortiss IIoT Lab in Munich. The equipment comprised eight Raspberry Pi static nodes, three notebook computers, and three TurtleBot3 mobile nodes.

The developed ROS2 AMR application and dashboard, along with all the code, is provided via the CODECO Eclipse GitLab as explained in section 8.2.

### 8.4.2 Demo Scenarios

Two scenarios were demonstrated, corresponding to the two P5 user journeys. A simplified overview of the demo environment is provided in Figure 29.

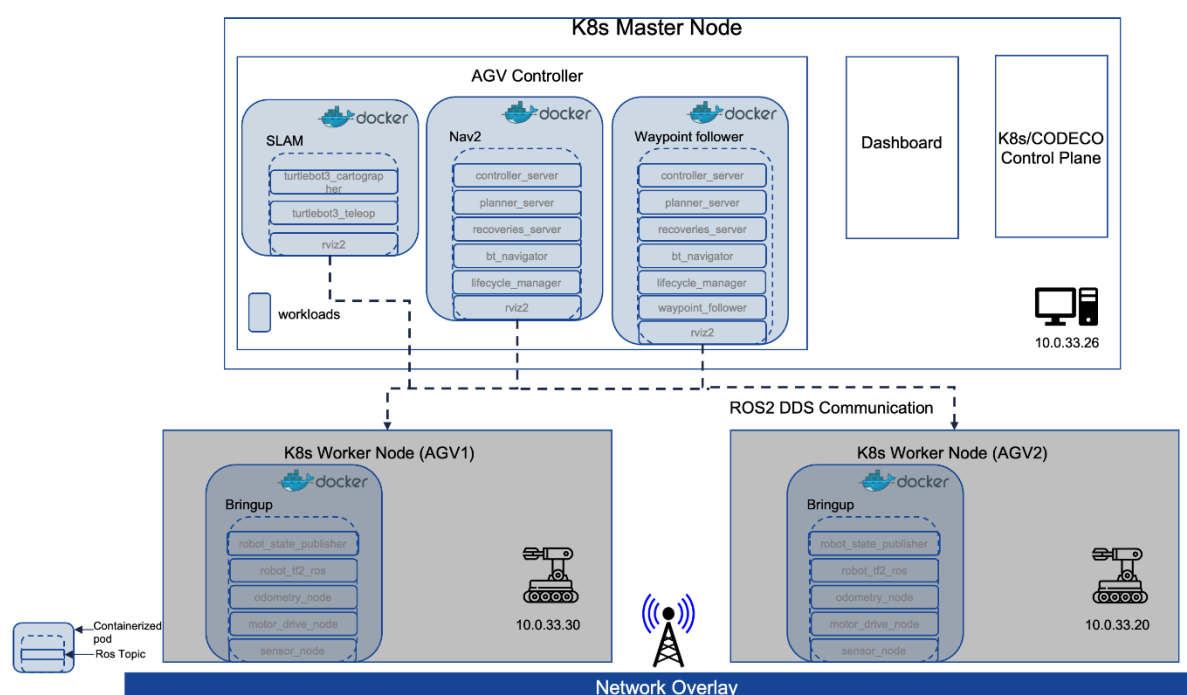


Figure 29: P5 – Simplified example of the demo environment.

**Demo Scenario A — Microservice Deployment in an AMR Fleet:** The user downloads the P5 robotics application from the CODECO GitLab to the controller node and opens the use-case dashboard. Via the dashboard, the user specifies deployment parameters — number of nodes, device characteristics, Wi-Fi SSID, and other networking properties — and selects the set of microservices to deploy. The dashboard provides real-time feedback on deployment progress, drawing on monitoring data from NetMA, ACM, and MDM. CODECO handles network path optimisation and interference mitigation throughout. On completion, the dashboard displays the time to deployment along with robot status — location, battery level, and active microservices. The user then issues a task request to confirm successful deployment by triggering autonomous AMR movement.

**Demo Scenario B — AMR Fleet Control: Resilient Infrastructure:** Starting from a running deployment, the user simulates battery depletion on one AMR (worker node 1) via the dashboard command `kubectl cordon <node-name>`, while that AMR is actively using SLAM to build a map of the area. The map construction is monitored in real time via RViz on the controller node. CODECO detects the node condition and triggers re-scheduling of the SLAM microservice. The current map state is saved and CODECO executes stateful migration —

including the map data — to another suitable AMR. The dashboard notifies the user of migration completion and total migration time. The user then issues a task request originally intended for worker node 1; worker node 2, selected by CODECO, completes the task instead by navigating to destination C. The dashboard displays energy consumption across all MARs throughout. An overview of the migration scenario is provided in Figure 30.

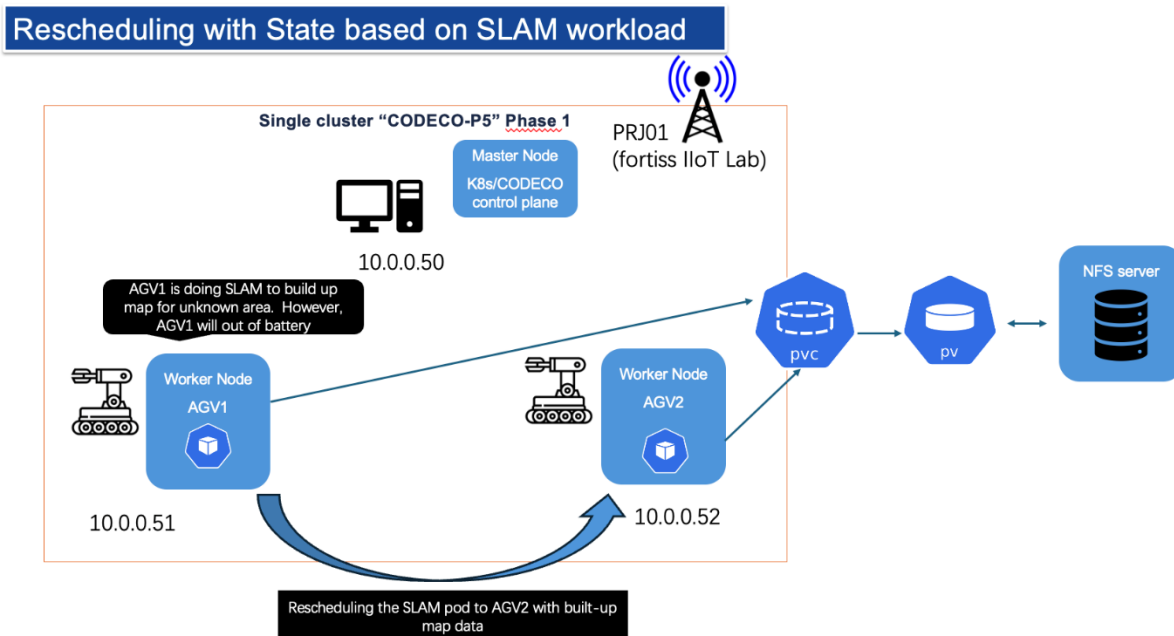


Figure 30: P5 – Simplified example of migration offloading due to node/network failure.

The steps taken in this demonstration scenario are as follows:

1. Via the P5 Dashboard, user depletes the energy of one AMR (worker node 1)
2. While the AMR is using SLAM to build a map of the area.
3. On the controller node, the user can follow the building of the map with RViz.
4. CODECO/K8s detect the node failure and a need to re-schedule the SLAM micro-service. The map data is saved (currently, master node) and CODECO handles the process of stateful migration (including the map data).
5. The user, via the Dashboard, receives a notification of the migration completion with total time.
6. The user performs a task request to the AMR (originally worker node 1). Worker node 2 (new AMR selected by CODECO) moves instead of worker node 1 to complete the task request (move until destination C).
7. Via the dashboard, information concerning the energy processing across all MARs is displayed to the user.

### 8.4.3 Demo Event

An on-premises demonstration has been carried out by fortiss in December 2025, as specified in Table 28.

*Table 28: P5 demo camp event.*

| Date / Time       | December 11, 2025, during the Fortiss Demo Camp  |
|-------------------|--|
| <b>Organizers</b> | FOR, Rute Sofia  |
| <b>Audience</b>   | Local audience, fortiss Demo Camp visitors, CODECO partners (online)   |
| <b>Duration</b>   | 2 hours  |
| <b>Content</b>    | Overview of CODECO and the P5 use case, live demonstration of demo scenarios A and B, Q&A session.<br>See: <a href="https://zenodo.org/records/17989291">https://zenodo.org/records/17989291</a> |

### 8.4.4 Risks and Mitigation Measures

For P5, we have identified the risks and mitigation measures outlined in Table 29.

*Table 29: P5 – Demo risks and mitigation measures.*

| Risk   | Mitigation  |
|--|---|
| <b>Delays in the development of the toolkits or issues derived of the customization for the use-case</b> | Consider an earlier version of the code, shown to work<br>Rely on dedicated hardware for the demo (separated from the validation equipment)                                     |
| <b>Venue availability</b>  | Ensure that the proposed venues controlled by FOR are available in Q1 2025; for external events to the consortium, ensure that the venue is known at an early stage, or opt out |
| <b>Agenda availability</b>   | Contact the organizers  |
| <b>Data protection</b>   | Start the process of confirmation of no blockers in FOR for all planned events, as soon as venues and agenda are known (at least 3 months before)                               |
| <b>Potential problems during demo validation or execution</b>  | Ensure that videos are set and available for all demos  |

## 8.5 Summary

The P5 use-case demonstrated how CODECO enables flexible, scalable, and resilient control of AMR fleets in smart manufacturing environments, representing a fundamental shift from rigid, pre-defined AMR operations to dynamic, autonomously orchestrated systems operating over wireless infrastructure.

The use-case addressed two complementary challenges: the automated deployment of containerised robotics applications across heterogeneous fleets of resource-constrained embedded devices, and the runtime management of those applications under conditions of battery depletion, connectivity disruption, and resource variability. Central to both was CODECO's support for stateful workload migration, which enabled critical services such as SLAM to be transferred between AMRs — along with their accumulated map data — without interrupting operation.

Each AMR functioned as a K8s worker node running containerised ROS2 microservices, with compute-intensive tasks offloaded to the controller node. The three-microservice pipeline — bringup, SLAM, and Navigation — was designed to expose clear orchestration control points for CODECO, allowing SWM and PDLC to make and execute informed placement and migration decisions. Persistent storage via K8s Persistent Volumes and Persistent Volume Claims provided the state continuity required for seamless migration.

Experimental validation across both the KinD emulation environment and the fortiss IIoT Lab confirms that CODECO improves resource efficiency and communication stability compared

to standard Kubernetes. Against the defined KPIs, four of the seven sub-KPIs — K1.1, K2.2, K2.3, and K2.4 — are addressed by the available experimental evidence. K1.1 was validated through testbed deployment involving seven nodes per cluster across three clusters in federated operation. K2.2 is supported by the substantially more stable and controlled inter-pod data rates observed under CODECO, consistent with reduced link load and extended network operational lifetime. K2.3 is supported by consistently lower CPU consumption under CODECO, a meaningful proxy for energy efficiency on embedded AMR hardware where compute and power are tightly coupled. K2.4 is supported by lower and more stable transmit data rates under CODECO relative to the burst behaviour observed under Kubernetes, reflecting the QoS-aware constraint of unnecessary inter-pod traffic.

K1.2 is partially addressed: pod lifecycle latency was measured at approximately 5 seconds under CODECO versus 2 seconds under Kubernetes, and end-to-end task completion on physical robots has been verified at the fortiss IIoT Lab and via demonstration videos, with quantitative reporting under development. K1.3 was not met in the current experimental phase, as the L2S-M secure overlay initialisation introduces additional deployment latency relative to baseline Kubernetes; this KPI will be re-evaluated at scale in real-device experiments where CODECO's placement intelligence is expected to offset this cost across larger fleets. K2.1 — collision reduction — has not yet been assessed, as it requires physical robot interaction under real mobility conditions and will be addressed in the forthcoming real-device validation phase using TurtleBot robots in the fortiss IIoT Lab.

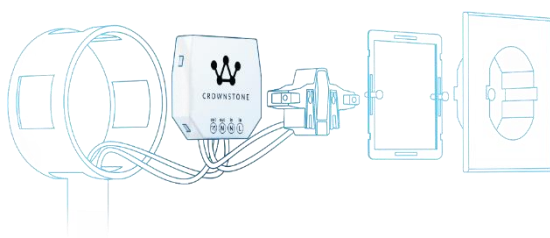
The live demonstration at the fortiss Demo Camp in December 2025 validated both user journeys in a real-world industrial setting, confirming CODECO's readiness to support complex, mobile, and resource-constrained manufacturing applications.

## 9 P6: Smart Buildings

As described in Section 2.6, this pilot applies CODECO to automate the deployment and management of a Crownstone smart building network, demonstrating the framework's ability to orchestrate non-standard edge devices, respond to topology changes, and incorporate environmental context — specifically occupancy — into orchestration decisions. The sections next detail the use-case architecture, deployment, experimentation, and demonstration.

### 9.1 Use Case Architecture

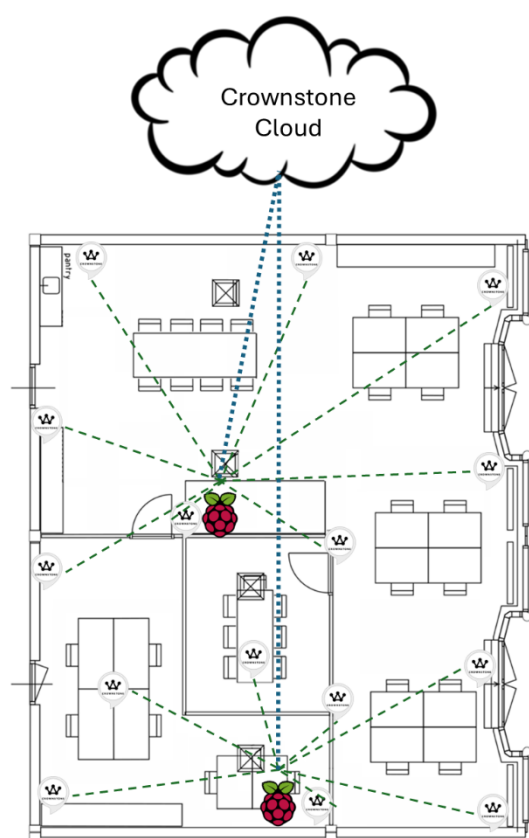
The Crownstone platform comprises five device types that together form a layered smart building infrastructure. **Crownstone nodes** are smart terminals mounted inside standard power outlets, each capable of switching and dimming connected devices, measuring power consumption, maintaining BLE connections with nearby sensors and actuators, communicating with other nodes via Bluetooth mesh, and running lightweight micro-apps on their onboard processor. **Crownstone Bridges** are USB-stick form-factor devices that connect a Bluetooth mesh network to a Crownstone Hub without switching or measurement capability. **Crownstone Hubs** are Raspberry Pi devices with a dedicated software stack, responsible for collecting data from up to 256 Crownstone nodes, performing higher-level analytics, deploying micro-apps to nodes, and connecting to the cloud. The **Crownstone Cloud** provides sphere administration — managing the relationships between buildings, rooms, nodes, and smartphones — and enables connectivity to other Crownstone networks and internet services. The **Crownstone App**, a React Native application, allows smartphones to function as detectable BLE beacons and provides a management dashboard for the platform.



*Figure 31: P6 – Crownstone Node Inside a Power Outlet.*

Applications in a Crownstone network span all three layers: low-level sensor data processing runs inside nodes, higher-level analysis runs on hubs, and coordination and connectivity are handled by the cloud. CODECO sits above this infrastructure as the orchestration layer, managing initial deployment and responding dynamically to two classes of trigger: topology changes — nodes or hubs added or removed — and occupancy shifts — changes in the presence and movement of people in the office.

Three technical challenges had to be resolved before CODECO could orchestrate the Crownstone environment. The first was integrating Crownstone nodes — which lack a TCP/IP stack — into the Kubernetes ecosystem. This was addressed through Virtual Kubelets (VKs), automatically installed by CODECO on each hub.



*Figure 32: P6 – Typical Setup of a Crownstone Network.*

Each Virtual Kubelet registers with the K8s cluster as a standard worker node, providing the Crownstone node with an IP interface and linking it to a custom API that translates K8s and CODECO commands into BLE actions directed at the physical device. Crownstone nodes can also report data back to their Virtual Kubelet via BLE, completing the communication loop. The second challenge was enabling CODECO to maintain awareness of the Crownstone network topology — specifically, determining which hub can communicate with each node, and ensuring each node carries the correct micro-app for its intended function. The third challenge was making occupancy information available to CODECO's orchestration logic.

The key insight was that the presence of people significantly influences BLE mesh network statistics — packet loss rates, signal strength, broadcast conflicts, and latency — which are already reported by nodes to their hubs. Making these statistics available to CODECO provided an indirect but effective occupancy signal. A secondary route was also pursued: surfacing direct occupancy data from the Crownstone Cloud, where smartphone apps report user

presence, down to the edge.

A typical Crownstone network setup is illustrated in Figure 32. Crownstone nodes form a mesh, each associated with a hub that hosts its Virtual Kubelet representation. Hubs are standard Kubernetes worker nodes, and the Crownstone Cloud runs on a cloud worker node — typically an AWS instance. The CODECO control plane operates in the cloud, with CODECO monitoring components running on all three physical worker node types. A detailed technical overview of the full setup is shown in Figure 33.



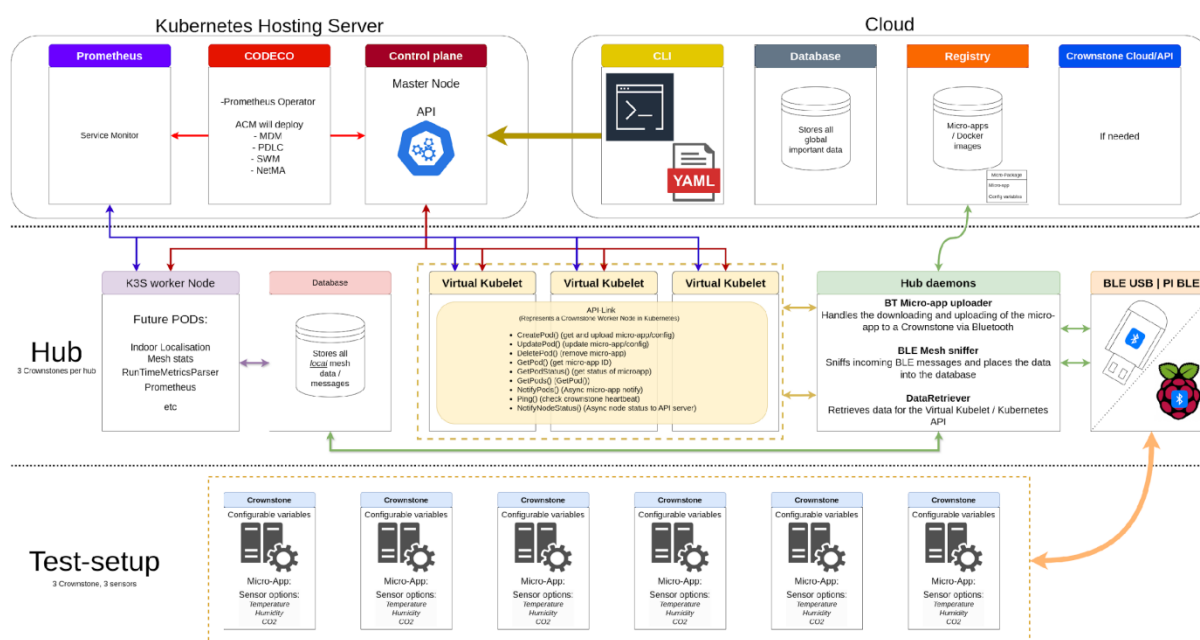


Figure 33: P6 – Technical Setup of Use-Case P6.

### 9.1.1 Pre-conditions, Triggers, and Postconditions

**Preconditions:** All Crownstone hardware must be physically installed in the office environment and added to the Crownstone network via the smartphone app before CODECO is initialised. Both Crownstone Hubs must be operational and connected to the Crownstone Cloud. The CODECO cluster with the control plane in the Cloud, worker components on the hubs (Edge) must be running. A minimum of twelve Crownstone nodes must be active and reachable by at least one hub.

**Triggers:** Three conditions initiate CODECO action. System start-up following hardware installation triggers initial automated deployment of all components (User-Story 1). Detection by a hub of a previously unknown or newly absent Crownstone node triggers topology-based redeployment (User-Story 2). Deviations in BLE mesh statistics, e.g., signal strength, packet loss, broadcast conflicts, or latency beyond predefined thresholds trigger occupancy-based redeployment (User-Story 3).

**Postconditions:** Successful execution delivers automated initial deployment with reduced setup effort compared to manual configuration; real-time dashboard visibility of cluster state and micro-app status; and autonomous redeployment in response to topology and occupancy changes without requiring manual intervention.

### 9.1.2 CODECO Integration

**ACM** manages the automated deployment of all Crownstone Cloud, Hub, and node components based on the YAML configuration file, and maintains the desired state of the network throughout operation. **NetMA** processes BLE network statistics reported by Virtual Kubelets, detecting deviations that indicate topology changes or occupancy shifts, and generates the triggers that initiate redeployment. **SWM** executes redeployment decisions — determining the optimal hub for hosting a new Virtual Kubelet, deploying the required pods, and updating relay agent configurations to maintain seamless connectivity. **MDM** provides data observability across the infrastructure, supporting monitoring of micro-app state and node behaviour. The CODECO control plane enables the facility manager to define and maintain service levels for building applications without manual monitoring or intervention.



## 9.2 Lab Deployment

### 9.2.1 User Stories

The use-case was deployed at the Almende main office in Rotterdam. The infrastructure comprised one AWS cloud server hosting the Crownstone Cloud and the CODECO control plane, two Raspberry Pi 5 Crownstone Hubs acting as edge worker nodes, and at least twelve Crownstone nodes installed in office power outlets. Node positions were not fixed in advance. Some were mounted in wall outlets, others deployed via extension cord plugs. Similarly, the assignment of Virtual Kubelets to hubs was determined dynamically by CODECO based on node visibility and hub load balancing, rather than pre-configured.

**User-Story 1 — Automated Application Deployment:** The first user-story provides the baseline capability on which the other two depend. CODECO automatically deploys the Crownstone Cloud components, Hub components, and micro-apps across all Crownstone nodes based on a YAML configuration file, triggered by system start-up after hardware installation and network initialisation via the smartphone app.

**User-Story 2 — Topology-Based Redeployment:** Starting from a running CODECO-managed Crownstone network, the addition or removal of a Crownstone node triggers CODECO to expand or contract its cluster. When a new node is detected — its messages identified by a hub's mesh sniffer — CODECO determines the most suitable hub to host the node's Virtual Kubelet, deploys the required pod, and updates relay agent configurations to integrate the node seamlessly. Removal of a node triggers the inverse process, including redeployment of any micro-apps that were running on the removed node to an alternative Crownstone.

**User-Story 3 — Occupancy-Based Redeployment:** Starting from a running network, changes in office occupancy that cause BLE mesh statistics to deviate beyond defined thresholds generate a trigger through the Virtual Kubelet monitoring layer. NetMA processes these deviations and initiates redeployment actions — removing micro-apps that are too processing-intensive for the degraded mesh or adjusting micro-apps generating excessive traffic — to restore the network to an optimal operating state. When occupancy decreases and statistics return to normal, the original deployment is restored.

### 9.2.2 Software Customization

The main software challenge was integrating non-IP Crownstone devices into Kubernetes to enable CODECO orchestration. Each Virtual Kubelet registers with the cluster as a standard worker node and exposes a custom API that translates K8s and CODECO commands into BLE operations directed at its corresponding Crownstone node. This API also receives data reported by the Crownstone node via BLE, making device state visible to the orchestration layer. The result is full integration of non-IP devices into the K8s ecosystem without modifying the Crownstone hardware or firmware. Additional customisation was required to make BLE network statistics — packet loss rates, signal strength, broadcast conflicts, and latency — available to CODECO via the Virtual Kubelet reporting pipeline, enabling occupancy-driven orchestration without requiring direct access to cloud-hosted presence data.

## 9.3 Experimentation and Evaluation

### 9.3.1 Experimentation Methodology

Evaluation was structured around the three user-stories, with elaborate methodologies applied to User-Stories 2 and 3, and a simpler approach for User-Story 1, which was treated as a prerequisite for the others.

**User-Story 1** was evaluated against KPIs 1 and 4. For KPI 1 (deployment effort), manual installation time was measured and compared against CODECO-automated deployment time, with the expectation that automation would reduce total setup effort. For KPI 4 (ability to monitor and verify sphere behaviour), the evaluation observed whether the Crownstone dashboard correctly reflected the desired state at initialisation — all nodes active, all micro-apps running — with no errors during setup. Micro-apps were developed specifically to communicate their running state to the dashboard.

**User-Story 2** was evaluated against the same KPIs 1 and 4. For KPI 1, manual addition or removal of a single node was timed and compared against CODECO-automated reconfiguration. For KPI 4, the dashboard was observed to confirm it dynamically updated to reflect topology changes in real time, including the resulting shifts in network statistics.

**User-Story 3** was evaluated against KPIs 2 and 3. KPI 2 (effort to deploy distributed edge-to-cloud AI applications) was assessed by observing micro-app deployment and redeployment in response to occupancy changes, following the same approach as the previous user-stories. KPI 3 (effort to reconfigure far-edge data processing applications) assessed the availability and responsiveness of occupancy-derived triggers: CODECO framework logs were examined to verify that redeployment actions were initiated and completed each time occupancy in the office changed beyond the defined thresholds.

### 9.3.2 Experimentation Results

The experimentation focused on User-Story 1, measuring CODECO framework deployment time across varying numbers of Crownstone nodes, and comparing application deployment time with and without CODECO.

CODECO deployment time scaled moderately with the number of Crownstone nodes, as shown in Table 30, reflecting the additional Virtual Kubelet integration required per node. Variability was bounded across all configurations, indicating stable and predictable deployment behaviour.

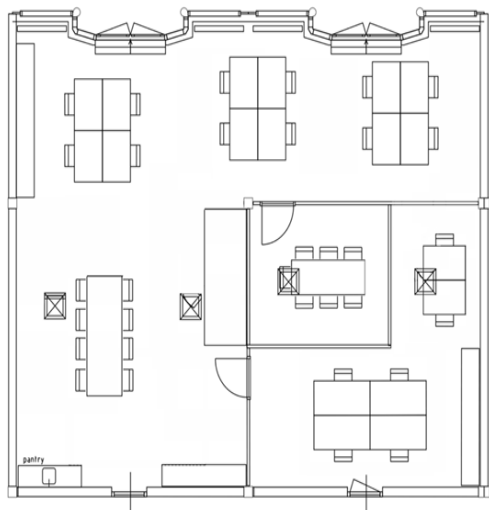
*Table 30: P6, CODECO Deployment Time vs Number of Crownstone Nodes.*

| Number of Crownstones | Avg. Deployment Time (s) | Min (s) | Max (s) |
|-----------------------|--------------------------|---------|---------|
| 1                     | 569                      | 466     | 592     |
| 2                     | 564                      | 466     | 603     |
| 4                     | 609                      | 591     | 630     |
| 8                     | 638                      | 532     | 672     |
| 16                    | 732                      | 715     | 743     |

Application deployment with CODECO was slower than direct baseline deployment without orchestration. This outcome reflects the overhead introduced by CODECO's scheduling and control-plane interactions, monitoring and decision-making processes, and the abstraction layer required to integrate non-IP Crownstone devices via Virtual Kubelets — none of which are present in a direct, unorchestrated deployment. The result highlights a fundamental trade-off characteristic of small-scale environments: orchestration overhead is present from the outset, while the benefits — scalability, automated management, and adaptive redeployment — only become fully realisable as system complexity and dynamism increase. In the target operating context of a real, occupied office building with regularly changing topology and occupancy conditions, these benefits are expected to outweigh the per-deployment latency cost.

## 9.4 Demonstration

### 9.4.1 Technical Setup



**Figure 34:** P6 - Demonstration location overview.

The demonstration took place at the Almende office in Rotterdam, Stationsplein 45, E7.154, as shown in Figure 34. The setup comprised one cloud server running the Crownstone Cloud database and cloud interface components, a second server hosting the CODECO control plane, one or two Raspberry Pi 5 platforms (8 GB RAM, M2 SSD) acting as Crownstone Hubs depending on the scenario, and at least twelve Crownstone nodes represented in the cluster via Virtual Kubelets. CODECO managed the deployment of the cloud interface component, the Virtual Kubelet pods on the hubs, the micro-apps on the Crownstone nodes, and application-specific pods on the hubs.

Three scenarios were demonstrated, corresponding to the three user-stories, A, B, C. The primary end user in all scenarios was the facility manager, with building occupants playing a passive but operationally significant role — their

presence or absence influencing the desired state maintained by CODECO.

**Demo Scenario A — Automated Application Deployment:** This scenario demonstrates the baseline capability of CODECO: the automated initial deployment of a complete Crownstone network from a standing start. All hardware has been physically installed in advance — Crownstone nodes mounted in power outlets, hubs connected and initialised, cloud configuration in place — but no applications are running yet. The facility manager initiates the system through a single action, and CODECO takes over from that point. It reads the YAML configuration file, determines the desired state of the network, and systematically deploys the cloud interface component, the Virtual Kubelet proxy layer on the hubs, and the micro-apps on the individual Crownstone nodes. The deployment dashboard makes this process visible in real time, showing each component reaching its target state. Once complete, the application dashboard confirms that the deployed application is active and reporting data — for example, temperature, CO<sub>2</sub> levels, or occupancy readings from the sensor micro-apps running inside the Crownstone nodes. The scenario establishes the operational baseline from which Scenarios B and C proceed and illustrates how a facility manager can bring an entire smart building application stack online without any manual configuration of individual devices. Table 31 summarizes the steps of the demo A.

**Table 31:** P6 – Steps in demo scenario A — Automated Application Deployment.

| Step | Action   | Observation  |
|------|--|--|
| 0    | Pre-defined state: Crownstones are installed and configured for indoor positioning (trained). Hubs are installed and configured for connections with Crownstones (initialized). Cloud component configuration is initialized. The CODECO cluster exists. | In the office room, we show where the Crownstones and hubs are. We explain the concepts of the use-case specific deployment dashboard and application dashboard. We show the deployment dashboard. |
| 1    | Facility Manager starts Crownstone infrastructure (initiation).  | On the deployment dashboard, we see CODECO starting up and initiating the desired state: cloud component and Crownstone proxies and pre-installed micro-apps.                                      |

| Step | Action   | Observation   |
|------|--|---|
| 2    | Facility Manager starts Crownstone Application(s). | On the deployment dashboard, we see CODECO having deployed the application components (cloud, hub, micro-app).<br>On the application dashboard, we see that the application is active. Idea: have a sensor application which reports temperature, CO2 level or occupancy. |

**Demo Scenario B — Topology-Based Redeployment:** This scenario demonstrates CODECO's ability to detect and respond autonomously to changes in the physical infrastructure of the Crownstone network. Starting from the fully deployed state established in Scenario A, the facility manager physically removes a Crownstone node from its power outlet — a deliberate action that simulates a real-world event such as a device being relocated, replaced, or taken offline for maintenance. The moment the node is unplugged, it becomes unreachable and ceases transmitting in the BLE mesh. One or both hubs detect the absence of the node's messages through the mesh sniffer, and this detection is reported to CODECO as a topology change. CODECO evaluates the impact on the desired state: a micro-app that was running on the removed node is now undeployed. Without any human instruction, CODECO identifies a suitable alternative Crownstone node — one that is reachable by the hub and has available capacity — and redeploys the micro-app there. The entire process is visible on the deployment dashboard, which updates in real time to reflect the change in cluster composition and the new placement of the micro-app. The scenario (Table 32) demonstrates that the Crownstone network can absorb hardware changes gracefully, maintaining its operational state without requiring the facility manager to intervene in the redeployment logic.

*Table 32: P6 – Steps in demo scenario B — Topology-Based Redeployment.*

| Step | Action  | Observation  |
|------|---|--|
| 0    | Pre-defined state: Crownstone infrastructure and applications are running (end of Scenario 1).            | N/A.   |
| 1    | Facility Manager picks a pluggable Crownstone and removes it from a power outlet.                         | We watch the “facility manager” removing the Crownstone from the power outlet.   |
| 2    | The removed Crownstone is detected by one of the hubs, and the hub reports the topology change to CODECO. | On the deployment dashboard, we see something happening: based on the trigger, the desired state changes and a re-deployment of the existing microapp on the removed Crownstone takes place. After some time, we see that the microapp is deployed on another (previously empty) Crownstone. |

**Demo Scenario C — Occupancy-Based Redeployment:** This scenario demonstrates the most contextually sophisticated capability of the P6 use-case: the use of BLE mesh network statistics as an indirect occupancy signal that drives automated micro-app redeployment. Starting from the operational state following Scenario B, the demonstration exploits the fact that the physical presence of people in a room significantly degrades BLE mesh performance — their bodies absorb and reflect radio signals, increasing packet loss, reducing signal strength, and introducing broadcast conflicts and latency spikes across the Crownstone nodes in that space.

In the demonstration, all attendees move simultaneously into the meeting room, creating a deliberately extreme occupancy condition. The Crownstone nodes in the room immediately begin reporting degraded mesh statistics to their hub. The hub aggregates these reports and, when the metrics exceed the predefined thresholds, forwards the signal to CODECO via the

Virtual Kubelet monitoring layer. CODECO's NetMA component processes the deviation and identifies it as an occupancy-driven change in network conditions. In response, CODECO initiates redeployment: micro-apps that are computationally expensive or that generate excessive mesh traffic — and which are therefore contributing to the degradation under constrained radio conditions — are removed from the affected Crownstone nodes or replaced with lighter alternatives. This reduces the load on the struggling mesh and allows the network to maintain an acceptable operating state despite the crowded environment. The deployment dashboard shows the redeployment taking place in real time.

In the final step of the scenario, the attendees move back out of the meeting room. BLE conditions return to normal as the radio environment clears, and the mesh statistics reported by the Crownstone nodes recover to baseline levels. CODECO detects this recovery and reverses its earlier decisions: the micro-apps that were removed or adjusted are restored to their original configuration on the previously affected nodes. The deployment dashboard again reflects the change, showing the network returning to its full desired state. The scenario illustrates how CODECO can incorporate environmental context — occupancy inferred from physical-layer statistics rather than explicit user input — into orchestration decisions, enabling a class of automated, event-driven building management that would be practically impossible to achieve through manual monitoring and intervention. Steps are presented in Table 33.

*Table 33: P6 - Steps in demo scenario C — Occupancy-Based Redeployment.*

| Step | Action  | Observation   |
|------|---|---|
| 0    | Pre-defined state: Crownstone infrastructure is running (end of Scenario 2).  | N/A.  |
| 1    | All attendants move into the meeting room in the office. That hardly fits ☺   | N/A.  |
| 2    | The extreme occupancy is detected by the hub based on reduced signal strength and packet loss reported by the Crownstones. The hub reports the situation to CODECO. | On the deployment dashboard, we see something happening: based on the trigger, the desired state changes and micro-apps are moved away from the Crownstones suffering from the extreme occupancy.   |
| 3    | The attendants move outside the meeting room, being able to breathe normally again.   | On the deployment dashboard, we see something happening: based on the trigger, the desired state changes and micro-apps are moved back to the Crownstones that suffered from the extreme occupancy. |

## 9.4.2 Events

Two demonstration events were held at the Almende office in Rotterdam, summarized in Table 35 and Table 36.

*Table 34: P6 – First Event Overview.*

| Date / Time | October 17, 2025  |
|-------------|---|
| Organizers  | ALM, Andries Stam, Chiel van Diepen, Tymon de Jonge   |
| Audience    | Local audience, CODECO partners (online)  |
| Duration    | 1 hour  |
| Content     | Overview of CODECO and the P6 use case, live demonstration of demo scenarios A and B without CODECO. The audience was actively involved: unplugging Crownstones, walking around, leaving and entering the office collectively, etc.<br>Our home video experts Danny, Teresa and Kevin will record the entire meeting. We are also thinking of interviewing some of the attendees with one or two questions. |



Table 35: P6 – Second Event Overview.

| Date / Time | March 27, 2026   |
|-------------|--|
| Organizers  | ALM, Andries Stam, Chiel van Diepen, Tymon de Jonge  |
| Audience    | Local audience, CODECO partners (online)   |
| Duration    | 1 hour   |
| Content     | Live demonstration of demo scenario A with CODECO, Q&A session. The audience will again be actively involved in the demonstration. |

## 9.5 Risks and Mitigation Measures

For this use case, there were no severe dependencies except for the functioning of the CODECO framework. The identified risks are mentioned in Table 36.

Table 36: P6 - Demo risks and mitigation measures.

| Risk  | Mitigation  |
|---|---|
| Not enough participants in event (< 25)       | Invite members from own daughter companies  |
| Scenarios turn out not to work during dry-run | Analyse issues, have fallback scenarios in place to demonstrate parts of the intended scenarios, hardcoding |

## 9.6 Summary

The P6 use-case applied CODECO to a smart building environment, demonstrating the framework's ability to automate the deployment and adaptive management of a Crownstone IoT network across a heterogeneous system of non-IP edge devices, Raspberry Pi hubs, and cloud services.

The central technical challenge was integrating Crownstone nodes — devices without a TCP/IP stack — into the Kubernetes ecosystem. This was resolved through Virtual Kubelets acting as proxy worker nodes, each providing a Crownstone node with a Kubernetes-compatible interface via a custom BLE translation API. This enabled CODECO to treat Crownstone nodes as standard cluster members and deploy, monitor, and reconfigure micro-apps on them without modifying the underlying hardware. The incorporation of BLE mesh statistics as an occupancy signal represented a further integration contribution, enabling event-driven redeployment based on environmental context rather than explicit user input.

Three user-stories structured the deployment and demonstration: automated initial deployment of the full Crownstone network; topology-based redeployment triggered by node addition or removal; and occupancy-based redeployment driven by BLE mesh statistics. In all three scenarios, CODECO responded to triggers without human intervention, updating cluster composition, redistributing micro-apps, and restoring desired system state automatically.

Experimentation confirmed that CODECO deployment time scales moderately with the number of Crownstone nodes — from an average of 569 seconds for a single node to 732 seconds for sixteen nodes — with bounded variability indicating stable behaviour. Application deployment under CODECO was slower than direct baseline deployment, reflecting the orchestration overhead inherent in small-scale environments where the adaptive benefits are not yet fully exercised. In the target operational context — a dynamic office environment with variable topology and occupancy — these benefits are expected to outweigh the per-deployment latency cost, particularly for scenarios that would otherwise require continuous manual monitoring and reconfiguration. The two demonstration events in Rotterdam validated all three user-stories in a real office environment, with audience members actively participating as occupants and confirming the responsiveness of CODECO's automated management.

## 10 Consolidated Results and Conclusions

### 10.1 Cross-functional Coverage of CODECO

This section emphasizes that each CODECO component is engineered with distinct functionalities that are actively employed by the CODECO pilots and serve distinct purposes. These functionalities are customized to meet the system's distinctive requirements and objectives, thereby guaranteeing its overall efficiency and operational effectiveness. The functionalities associated with each CODECO component are summarized in Tables 37-42. A more comprehensive analysis of CODECO's functionalities can be found in CODECO's Deliverable: "D10: CODECO Technological Guidelines, Reference Architecture, and Open-source Ecosystem Design" [10].

*Table 37: ACM Functionalities.*

| Functionality ID | Description  |
|------------------|--|
| <b>ACM-1</b>     | Installation of the entire CODECO framework.   |
| <b>ACM-2</b>     | Deploy custom applications across CEI (Cloud-Edge-IoT) in an efficient way.                                      |
| <b>ACM-3</b>     | Specifies target performance profiles that are used in PDLC to meet a specific level of greenness or resilience. |

*Table 38: ACM Functionalities Used*

| Functionality ID | P1 | P2 | P3 | P4 | P5 | P6 |
|------------------|----|----|----|----|----|----|
| <b>ACM-1</b>     | X  | X  | X  | X  | X  | X  |
| <b>ACM-2</b>     | X  | X  | X  | X  | X  | X  |
| <b>ACM-3</b>     | X  | X  | X  | X  | X  |    |

*Table 39: PDLC Functionalities.*

| Functionality ID | Description   |
|------------------|---|
| <b>PDLC-1</b>    | Provides an estimate on the overall system stability based on privacy-preserving decentralised learning approaches. |
| <b>PDLC-2</b>    | Provides an aggregated cost view of a specific target performance profile for the available infrastructure.         |

*Table 40: PDLC Functionalities Used.*

| Functionality ID | P1 | P2 | P3 | P4 | P5 | P6 |
|------------------|----|----|----|----|----|----|
| <b>PDLC-1</b>    | X  | X  | X  | X  | X  |    |
| <b>PDLC-2</b>    | X  | X  | X  | X  | X  |    |

*Table 41: SWM Functionalities.*

| Functionality ID | Description  |
|------------------|--|
| <b>SWM-1</b>     | Initial placement of workloads based on the application/user QoS requirements. |
| <b>SWM-2</b>     | Dynamic placement of application workloads (re-scheduling).                    |
| <b>SWM-3</b>     | Application workload deployment and re-deployment.                             |
| <b>SWM-4</b>     | Application workload migration.  |

Table 42: SWM Functionalities Used

| Functionality ID | P1 | P2 | P3 | P4 | P5 | P6 |
|------------------|----|----|----|----|----|----|
| <b>SWM-1</b>     | X  | X  | X  | X  | X  | X  |
| <b>SWM-2</b>     | X  | X  | X  | X  | X  | X  |
| <b>SWM-3</b>     | X  | X  | X  | X  | X  | X  |
| <b>SWM-4</b>     | X  | X  | X  |    | X  |    |

Table 43: NetMA Functionalities.

| Functionality ID | Description                              |
|------------------|--|
| <b>NETMA-1</b>   | Provides network awareness.              |
| <b>NETMA-2</b>   | Handles secure connectivity across pods. |

Table 44: NetMA Functionalities Used

| Functionality ID | P1 | P2 | P3 | P4 | P5 | P6 |
|------------------|----|----|----|----|----|----|
| <b>NETMA-1</b>   | X  | X  | X  | X  | X  | X  |
| <b>NETMA-2</b>   | X  |    | X  |    | X  |    |

Table 45: MDM Functionalities.

| Functionality ID | Description  |
|------------------|--|
| <b>MDM-1</b>     | Provides metrics related to how fresh the data processed by pod is (freshness).                        |
| <b>MDM-2</b>     | Provides metrics related to the level of compliance of the pod given where it is running (compliance). |
| <b>MDM-3</b>     | Provides metrics related to how robust the execution of the pod is where it is running (portability).  |

Table 46: MDM Functionalities Used.

| Functionality ID | P1 | P2 | P3 | P4 | P5 | P6 |
|------------------|----|----|----|----|----|----|
| <b>MDM-1</b>     | X  |    |    | X  |    |    |
| <b>MDM-2</b>     | X  |    |    |    |    | X  |
| <b>MDM-3</b>     | X  |    |    |    |    |    |

## 10.2 Analysis of CODECO Integration Across Use cases

The six pilots collectively cover a broad spectrum of deployment environments, application domains, and orchestration challenges. Examining how each pilot engaged with the CODECO component set reveals both the breadth of the framework's applicability and the specific capabilities that proved most consequential in practice.

**ACM** was the only component engaged by every pilot without exception, consistent with its role as the sole user-facing entry point to CODECO through which every partner initialised the framework, configured deployment parameters, and maintained visibility of system state. ACM-1 (framework installation) was therefore universal across all six pilots. ACM-2 (custom application deployment via the CODECO Application Model) was engaged by all pilots except P4, where the energy management software interfaced with the framework through custom pre-processing pipelines and a Kafka-based data bridge rather than through the standard CAM. ACM-3 (performance profile specification, which feeds target greenness or resilience objectives into PDLC) was engaged by P1, P2, P3, P4, and P5 and was absent only from P6, where redeployment is governed entirely by rule-based BLE threshold triggers rather than by pre-defined performance profiles supplied to a learning component.

**NetMA** was the second universally deployed component, engaged by all six pilots for network awareness (NETMA-1). Its role varied considerably across deployments. In P3, NetMA's most architecturally distinctive contribution was providing a decentralised ALTO implementation — integrating BGP- and SDN-sourced topology data to expose real-time network capabilities to the Media Delivery System across its multi-domain, multi-cluster environment spanning Madrid and Athens, extending CODECO's network awareness well beyond the cluster boundary. In P4, NetMA ensured reliable, low-latency communication between energy monitoring nodes and cloud orchestration resources across the distributed UPM campus infrastructure, providing the network observability required for the three-minute latency KPI from data ingestion to orchestration decision. In P6, NetMA served an unusual sensing function, processing BLE mesh statistics reported by Virtual Kubelets as an indirect occupancy signal and detecting topology changes triggered by node addition or removal — demonstrating the component's adaptability to non-standard physical-layer data modalities. The use of NETMA-2 (secure pod-to-pod connectivity via the L2S-M overlay) was more selective, confirmed only in P1, P3, and P5. In P1, WireGuard-secured links were established between all cluster nodes over Ethernet and Wi-Fi. In P3, NetMA ensured secure and transparent inter-component connectivity across the multi-cluster deployment. In P5, the VXLAN-based L2S-M secure overlay was explicitly confirmed in the experimentation results, where its initialisation and teardown accounted for the additional pod lifecycle latency observed relative to baseline Kubernetes. P2 relied on dedicated V2X radio infrastructure for inter-node communication and did not require CODECO-managed overlay networking; P4 operated over campus Ethernet and Wi-Fi infrastructure without evidence of CODECO overlay engagement; and P6 communicated over the Crownstone BLE mesh, which does not require an IP-level secure overlay.

**SWM** was engaged by all six pilots, though with varying depth. SWM-1 through SWM-3 (initial placement, dynamic rescheduling, and workload deployment and re-deployment) were used across all pilots. SWM-4 (stateful workload migration) was engaged by P1, P2, P3, and P5, and absent from P4 and P6. In P4, energy load redistribution was performed by domain-specific scheduling algorithms operating within the energy management system, which handled the regulatory and operational specificity of peak load tariffs and grid stability thresholds through custom logic rather than through CODECO's generic migration mechanism. In P6, the stateless nature of Crownstone micro-apps meant that redeployment to a different node required no state continuity and therefore no migration mechanism. The most technically demanding application of SWM-4 across the project was in P5, where stateful SLAM migration preserved accumulated map data across AMR handovers via Kubernetes Persistent Volumes and Persistent Volume Claims — the only pilot in which persistent storage formed an integral part of the migration mechanism, ensuring uninterrupted autonomous navigation following workload transfer.

**PDLC** was engaged by five pilots — P1, P2, P3, P4, and P5 — and absent only from P6. In P6, redeployment decisions are fully determined by rule-based thresholds applied to BLE mesh statistics, and the learning-based cost estimation and stability analysis that PDLC provides is neither needed nor invoked. Among the five pilots that engaged PDLC, the component's role varied considerably in character and optimisation objective. In P2, a modified PPO reinforcement learning agent was configured to minimise Age of Information as its primary objective, linking learning-based inference directly to a safety-critical performance metric and guiding SWM to proactively redistribute V2X processing microservices before bottlenecks formed. In P1, PDLC issued node placement recommendations based on channel conditions and resource availability, informing migration decisions under pod failure, channel degradation, and latency increase. In P5, PDLC analysed infrastructure robustness across node pairs — considering battery levels, channel conditions, and round-trip time — and issued proactive re-scheduling recommendations to SWM before AMR performance degraded. In P3, PDLC took a more anticipatory role, predicting network mishaps before they affected service

rather than reacting to observed threshold breaches. In P4, PDLC delivered energy demand and photovoltaic generation forecasts using privacy-preserving decentralised machine learning, enabling proactive load balancing and energy-aware task scheduling across the distributed UPM campus infrastructure without centralising sensitive consumption data.

**MDM** had the most selective cross-pilot engagement of all CODECO components, with confirmed use in P1, P4, and P6. In P4, MDM-1 (freshness of data processed by pod) was explicitly engaged: MDM collected and integrated real-time energy measurements from IoT sensors distributed across UPM buildings, enriching the data and feeding it into PDLC's predictive models. The freshness of these sensor readings was operationally central to the pilot's three-minute latency KPI from data ingestion to orchestration decision. In P6, MDM-2 (compliance — whether pods are running correctly in their assigned locations) was engaged to monitor micro-app state and node behaviour across the Crownstone network, verifying that each Crownstone node carried the correct micro-app for its assigned function within the sphere. For P1, all three MDM functionalities are indicated in the tables; however, the pilot description in Section 4 does not explicitly attribute specific MDM functionality to the deployment, and these entries should be verified with the P1 authors before being treated as confirmed. P2, P3, and P5 did not engage MDM: P2 sourced its primary observability signal from Age of Information metrics derived directly from V2X message timestamps; P3 relied on ALTO-derived topology data and application-level MEC API metrics; and P5 used ROS2 telemetry collected via Prometheus and visualised through Grafana for cluster observability.

Three cross-cutting patterns emerge from this analysis. First, the most comprehensive CODECO engagement — spanning all five components across the full range of defined functionalities — **occurred in the pilots with the most dynamic operational environments**: P1 with variable LiDAR load at resource-constrained edge nodes, P2 with mobile road users and continuously shifting latency profiles, P3 with multi-domain content delivery across geographically distributed clusters, and P5 with battery-depleted and wirelessly-constrained mobile robots. Second, **pilots where the application domain imposed strong regulatory or operational constraints**, as in P4 with energy management tariffs, grid stability thresholds, and forecasting requirements, engaged CODECO deeply but selectively, complementing rather than replacing domain-specific control logic with framework capabilities in the areas — deployment automation, network awareness, predictive scheduling — where CODECO added structural value without requiring domain semantics to be routed through generic mechanisms. Third, NetMA's consistent presence across all six pilots, including those where every other component was used selectively, confirms its role as the foundational network observability layer upon which the rest of CODECO's orchestration logic depends regardless of application domain or deployment environment.

## 10.3 Lessons Learned

The coordinated experimentation and demonstration across six pilots, six partner institutions, and multiple deployment environments produced a set of lessons that extend beyond the technical results of individual pilots and speak to the challenges of deploying a cognitive orchestration framework in real-world conditions.

### **Microservice decomposition is a prerequisite, not a consequence, of orchestration.**

Across the pilots, the ability of CODECO to manage workloads dynamically depended entirely on how the application had been structured beforehand. In P1, the decision to split the LiDAR analytics pipeline into Data Migration Pods, Processing Pods, and Collector Pods keeping Data Migration Pods fixed to their sensing sources while treating Processing Pods as movable workloads was what made runtime remapping possible at all. In P2, the decomposition of the V2X processing stack into independently deployable protocol microservices was the precondition for SWM to migrate processing load to the nearest suitable edge node as road users moved through the environment. In P5, the separation of bringup, SLAM, and Navigation into independent containerised services was the precondition for stateful SLAM migration,



since only a clearly bounded SLAM microservice with well-defined persistent state could be transferred between AMRs without disrupting the navigation stack. Partners that approached CODECO integration as a co-design exercise restructuring their applications to expose the right control points for runtime orchestration obtained significantly richer framework behaviour than those that attempted to retrofit orchestration onto pre-existing monolithic pipelines. This lesson applies at all layers of the stack: the structure of the application, the granularity of its containerisation, and the placement of its data dependencies all directly constrain what CODECO can and cannot do at runtime.

**Domain-specific constraints require domain-specific integration layers, but do not preclude deep framework engagement.** The energy management pilot, P4, illustrated clearly that CODECO's generic orchestration mechanisms do not natively accommodate domain-specific operational constraints such as peak load tariffs, grid stability thresholds, or regulatory compliance requirements. Custom pre-processing pipelines were developed to normalise and standardise heterogeneous energy data before ingestion into MDM, and domain-specific energy-event triggers were developed to translate operational semantics, for instance, fluctuations in photovoltaic generation, demand response signals, fault conditions into CODECO-compatible orchestration inputs. Crucially however, this translation effort did not limit the depth of CODECO integration: P4 engaged ACM, PDLC, SWM, NetMA, and MDM, with PDLC delivering energy demand and generation forecasts through privacy-preserving decentralised learning and SWM performing energy-aware workload scheduling across the distributed campus infrastructure. The lesson is therefore not that domain complexity reduces CODECO's role, but that it shifts the integration challenge to the boundary layer between domain semantics and framework signals. Investing in that boundary layer — as P4 did through its Kafka messaging pipeline, Modbus integration, and Prometheus data logger — unlocks the full orchestration capability of the framework even in operationally constrained environments.

**Orchestration overhead is scale-dependent and must be evaluated at the right scale.** Several pilots observed that CODECO introduced measurable overhead relative to baseline Kubernetes in small-scale experimental environments. In P5, pod deployment took approximately five seconds under CODECO versus two seconds under vanilla Kubernetes, with the additional time attributable to the initialisation of the VXLAN-based L2S-M secure overlay. In P6, per-node deployment time ranged from an average of 569 seconds for a single Crownstone node to 732 seconds for sixteen nodes, reflecting the Virtual Kubelet integration overhead required per device. In both cases, the overhead was disproportionately visible in small-scale setups where the adaptive benefits, i.e., dynamic redeployment, stateful migration, occupancy-triggered reconfiguration, were not yet fully exercised. This points to a general principle: CODECO's value proposition strengthens with the dynamism, fleet size, and redeployment frequency of the target environment, and weakens in stable, small-scale settings where static deployment is adequate. Evaluation at target operational scale is therefore essential for a fair assessment of the framework's net impact, and the emulation results reported across several pilots should be interpreted with this dependency in mind.

**Wireless connectivity is the dominant performance constraint in mobile edge deployments.** In P1, network-overhead analysis identified transmission delay as the dominant contributor to the latency increase of remote processing, exceeding the inference time itself and confirming that the main performance bottleneck in remote processing configurations was the wireless path rather than the CODECO framework. In P5, the key challenges that motivated CODECO's use, namely, interference, intermittent connectivity, and latency variability between AMRs and the controller node, were all wireless in origin, and NetMA's route optimisation and interference mitigation capabilities were central to maintaining the communication stability required for real-time robotic control. In P2, the end-to-end latency requirements of the VRU protection application demanded that processing be placed as close as possible to the V2X roadside units precisely because mobile network latency would

otherwise breach the 30-millisecond threshold regardless of orchestration quality. Across all three pilots, CODECO's network awareness through NetMA was operationally important, but the fundamental constraint remained the wireless medium. This underlines the need for tighter co-design between orchestration frameworks and wireless network management, an area where NetMA's cross-layer data-compute-network approach is well positioned but where further integration with 5G network management interfaces, radio resource allocation, and interference coordination remains an open and important research direction.

**Non-standard edge devices can be integrated into Kubernetes-based orchestration but require dedicated engineering effort.** The P6 use-case demonstrated that Crownstone nodes, which are consumer IoT devices operating exclusively over Bluetooth Low Energy without a TCP/IP stack, could be brought under CODECO management through Virtual Kubelets acting as proxy worker nodes, each providing a Crownstone node with a Kubernetes-compatible interface via a custom BLE translation API. The engineering effort required to build this integration layer was substantial: the API had to translate Kubernetes and CODECO commands into BLE operations directed at physical devices, receive telemetry back from those devices via BLE, expose BLE mesh statistics to NetMA for occupancy detection, and register each Crownstone node with the cluster as a standard worker node without modifying the underlying hardware or firmware. The result was a genuinely novel capability: the ability to deploy, monitor, and reconfigure firmware-level micro-apps on consumer IoT hardware through a standard Kubernetes orchestration interface and a reusable integration pattern applicable across smart building, industrial IoT, and smart city domains. The lesson is that CODECO's principle of device-agnostic orchestration is architecturally sound but that each new class of non-standard device will require a dedicated, non-trivial integration effort before that principle can be realised in practice.

**Environmental context is an underutilised but powerful orchestration input.** Two pilots demonstrated that orchestration triggers need not be limited to compute and network telemetry from within the cluster. In P6, BLE mesh statistics such as packet loss rates, signal strength, broadcast conflicts, and latency served as an indirect proxy for office occupancy, enabling CODECO to initiate micro-app redeployment in response to changes in human presence without requiring explicit occupancy data from the Crownstone Cloud. In P4, electricity price signals, photovoltaic generation variability, and CO<sub>2</sub> emission data served as inputs to PDLC's scheduling decisions, enabling energy-aware workload placement that adapted to external market and environmental conditions rather than relying solely on infrastructure telemetry. In both cases, surfacing physical-world context to the orchestration layer through careful integration engineering at the boundary between domain and framework produced some of the most operationally distinctive capabilities demonstrated in the project. The pilots that incorporated environmental context produced adaptive behaviours that would have been practically unachievable through either manual operation or conventional compute-and-network-only orchestration, and this direction represents a compelling avenue for further development in future work.

**Demonstrations revealed integration fragilities that experimentation alone did not surface.** In several pilots, the process of preparing for a live public demonstration together with city authorities, industry partners, students, or general audiences exposed configuration fragilities, user experience gaps, and integration issues that controlled lab experimentation had not surfaced. The discipline of demonstration-readiness proved to be a valuable quality gate, complementary to but distinct from the validation performed in lab environments: demonstrations impose real-time reliability requirements, heterogeneous audience expectations, and logistical constraints that stress-test the system in qualitatively different ways from scripted benchmarking scenarios. The structured demonstration planning process managed under Task 5.4 with its explicit requirements for fallback scenarios, risk identification, dry-run validation, and stakeholder engagement planning contributed measurably to the

robustness of the final implementations and should be considered an integral part of the validation methodology for future deployments of the framework.

## 10.4 Impact

The work documented in this deliverable has produced impact across three dimensions: technical, dissemination, and community.

**Technical impact.** The six pilots collectively validate CODECO's core thesis: that a cognitive, telemetry-driven orchestration layer extending Kubernetes can deliver measurable improvements in resource efficiency, system resilience, and operational adaptability across diverse real-world deployment scenarios. The quantitative results are concrete and span multiple application domains.

In P1, dynamic workload offloading under CODECO improved LiDAR analytics throughput by 32% at a sensor frame rate of 45 fps relative to a local-only processing configuration, while the framework responded effectively to pod failures, channel degradation, and latency increases. The wireless path was identified as the dominant performance bottleneck in remote processing configurations rather than the framework itself, a finding with direct implications for future co-design between edge orchestration and wireless network management.

In P2, CODECO's reinforcement learning-driven orchestration using a PPO agent configured to minimise Aol reduced average Aol by 13.5% and positional Error Distance by 23.6% at peak load of 300 simultaneous vehicles relative to a centralised baseline, while maintaining both deployments within the 3.5-metre lane-width safety threshold. The additional positional accuracy margin provided by CODECO is operationally significant in a VRU protection context, where half a metre in tracking accuracy determines the lead time available to issue a collision warning.

In P3, CODECO's ALTO-based network awareness enabled more efficient placement of media delivery caches and faster adaptation to network disturbances than a static Kubernetes deployment, validating the value of joint networking and compute orchestration in multi-domain content delivery environments spanning geographically distributed clusters.

In P4, the decentralised energy management system achieved the targeted 5–10% improvements in energy efficiency and carbon footprint reduction across three UPM campuses, with fault tolerance validated at the two-minute recovery target. PDLC's privacy-preserving decentralised learning delivered energy demand and photovoltaic generation forecasts that enabled proactive load balancing without centralising sensitive consumption data.

In P5, CODECO demonstrated reduced CPU consumption and substantially more stable inter-pod communication patterns relative to baseline Kubernetes, alongside successful stateful SLAM migration on physical AMR hardware with map data continuity preserved via Kubernetes Persistent Volumes. The trade-offs identified — faster pod lifecycle operations in vanilla Kubernetes versus lower CPU utilisation, controlled communication behaviour, and resilient stateful migration under CODECO — consistently favour the framework in dynamic, resource-constrained, wirelessly-connected manufacturing environments.

In P6, automated deployment and event-driven redeployment — triggered by topology changes and occupancy-derived BLE mesh statistics — demonstrated a class of adaptive smart building management that would be practically unachievable through manual monitoring and intervention. The Virtual Kubelet integration layer developed for this pilot enables CODECO to treat non-IP, Bluetooth-only consumer IoT devices as standard Kubernetes worker nodes, extending the framework's reach into device classes not previously supported by container orchestration systems.

Beyond the individual pilot results, the project has produced a set of reusable technical assets that constitute a lasting contribution to the broader Edge-Cloud orchestration ecosystem. These include open-source microservice pipelines for LiDAR-based traffic analytics (P1), V2X Digital Twin services implementing ETSI C-ITS protocols (P2), Media Delivery System components with ALTO-based network awareness (P3), energy monitoring and Kafka integration pipelines with Modbus sensor integration (P4), ROS2 AMR application workloads including SLAM, Navigation, and bringup microservices (P5), and the Virtual Kubelet integration layer for non-IP IoT devices with BLE translation API (P6). All are publicly available via the CODECO Eclipse GitLab repository and Docker Hub and are documented to a standard that supports reuse and extension by external researchers and developers.

**Dissemination impact.** The pilots generated a programme of eleven demonstration events across six partner sites in four countries — Germany, Spain, the Netherlands, and Greece — reaching audiences that included municipal authorities (City of Göttingen), university administrations (UPM), industry visitors (fortiss Demo Camp), and the public (European Researchers' Night in Madrid). These events established CODECO's visibility as a deployable, production-relevant framework rather than a research prototype.

**Community and ecosystem impact.** CODECO's integration with the Eclipse research ecosystem, its open-source licensing, and its participation in IETF standardisation discussions — particularly around the ALTO protocol used in P3 — position the project's outputs for uptake beyond the consortium. The fortiss IIoT Lab, made available as an open-access infrastructure, provides a continuing experimental platform for external researchers to validate and extend the P5 results. The Virtual Kubelet integration pattern developed in P6 offers a reusable blueprint for incorporating non-IP IoT devices into Kubernetes-based orchestration environments — a contribution with applicability across the smart building, industrial IoT, and smart city domains.

## References

- [1] Raquel Sousa (Ed), "CODECO D8 - Fine-grained Use-case Design and Business Impact", Zenodo, July 2023. DOI: 10.5281/zenodo.8143800.
- [2] Adreani, Lorenzo, et al. "Implementing integrated digital twin modelling and representation into the Snap4City platform for smart city solutions." *Multimedia Tools and Applications* 83.12 (2024): 37121-37146.
- [3] Asadi, Arash, Qing Wang, and Vincenzo Mancuso. "A survey on device-to-device communication in cellular networks." *IEEE Communications Surveys & Tutorials* 16.4 (2014): 1801-1819.
- [4] Intelligent Transport Systems (ITS); Communication Architecture for Multi-Channel Operation (MCO); Release 2, ETSI TS 103 696 V2.1.1, European Telecommunications Standard Institute (ETSI), Sophia-Antipolis, France, Nov. 2021
- [5] Lone, Faisal & Verma, Harsh & Sharma, Krishna. (2024). ETSI ITS: A Comprehensive Overview of the Architecture, Challenges and Issues. *International Journal of Sensors, Wireless Communications and Control*. 14. 85-103.
- [6] Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 1: Media-Independent Functionality; Release 2, ETSI TS 103 836-4-1 V2.1.1, European Telecommunications Standard Institute (ETSI), Sophia-Antipolis, France, Nov. 2022
- [7] J. Marias-i-Parella et al. "Interoperability between cellular and V2X networks (802.11 p/LTE-PC5) under a cloud native edge scenario". In: *IEEE INFOCOM 2023-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE. 2023, pp. 1–2
- [8] N. Psaromanolakis (Ed.), "CODECO D12 - CODECO Basic Operation Components and Toolkit Version 2.0"
- [9] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano and S. Ulukus, "Age of Information: An Introduction and Survey," in *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1183-1210, May 2021
- [10] Rute C. Sofia (Ed.), "CODECO D10 - Technological Guidelines, Reference Architecture, and Initial Open-source Ecosystem Design"
- [11] H. Zhu, T. Samizadeh, R.C. Sofia. A CODECO Case Study and Initial Validation for Edge Orchestration of Autonomous Mobile Robots. To appear, *IEEE Computer*, May 2026. ArXiv pre-print <https://arxiv.org/abs/2511.08354>