SUCCESS-6G: DEVISE

WP4 Deliverable E10

Initial testing and preliminary validation of service KPIs

Project Title:	SUCCESS-6G: DEVISE
Title of Deliverable:	Initial testing and preliminary validation of service KPIs
Status-Version:	v1.0
Delivery Date:	14/03/2025
Contributors:	Francisco Paredes, Miguel Fornell (IDNEO), Javier Santaella (CELLNEX), Carmen Vicente (CELLNEX), Pavol Mulinka, Roshan Sedar, Charalampos Kalalas, Miquel Payaro (CTTC), Guillermo Candela Belmonte (Optare), Michail Dalgitsis, Eftychia Datsika, Maria Serrano, Angelos Antonopoulos (NBC)
Lead editor:	Francisco Paredes (IDNEO)
Reviewers:	Charalampos Kalalas, Miquel Payaro (CTTC)
Keywords:	Testing; validation; service architecture; KPI

Document revision history

Version	Date	Description of change
v0.1	29/05/2024	ToC created
v0.2	12/07/2024	Revision of ToC
v0.3	24/01/2025	5G network KPI revision
v0.4	28/02/2025	Main content added
v0.5	07/03/2025	Internal review
v1.0	14/03/2025	Final version of the deliverable

Acknowledgment

The research conducted by SUCCESS-6G - TSI-063000-2021-39/40/41 receives funding from the Ministerio de Asuntos Económicos y Transformación Digital and the European Union-NextGenerationEU under the framework of the "Plan de Recuperación, Transformación y Resiliencia" and the "Mecanismo de Recuperación y Resiliencia".

[1] http://creativecommons.org/licenses/by-nc-nd/3.0/deed.en_US

Executive Summary

This deliverable elaborates on the initial testing activities and methodology followed for the validation of the Key Performance Indicators (KPIs) associated with the user story “Vehicular condition monitoring with security guarantees” for the SUCCESS-6G-DEVISE project. The primary objective of this phase is to verify the functional readiness of the considered architecture and the alignment of the associated enablers, serving as a foundational step toward full-scale validation. The document describes the two SUCCESS-6G testbeds considered in the project, including details of the testbed configurations, tools, and metrics employed across different scenarios. The methodology followed for KPI measurement and assessment is also presented setting the ground for the holistic KPI evaluation performed in the forthcoming E11 deliverable.

Table of Contents

List of Figures

List of Tables

1 Introduction

The "Initial Testing and Preliminary Validation of Service KPIs" deliverable outlines the initial phase of testing and validation for the Key Performance Indicators (KPIs) of the vehicular condition monitoring service under evaluation. This phase is crucial for establishing the baseline metrics and methodologies that will guide the overall performance assessment.

The report includes a detailed description of the various testbeds utilized, which are designed to simulate real-world conditions within controlled environments. Additionally, the methodology for obtaining the KPIs is detailed in this document. The approach combines automated monitoring tools, manual testing procedures, and statistical analysis to ensure a rigorous data collection process. Key network-related metrics such as latency, throughput, service availability, and error rates are measured against predefined thresholds to validate that the services meet the expected performance standards.

These initial results will serve as an early indication of whether the KPIs are aligned with the operational benchmarks, while also identifying potential areas for optimization before moving to the full-scale validation phase.

2 SUCCESS-6G Testbeds

This section elaborates on the two SUCCESS-6G experimental platforms used in the project for the validation of the service KPIs.

2.1 SUPERCOM platform

The Sustainable and High-Performance Computing (SUPERCOM) platform (https:// supercom.cttc.es) is an advanced experimental platform designed for machine learning (ML) research, integrating cutting-edge sensing, computing, and communication capabilities. The testbed, owned and maintained by the Sustainable Artificial Intelligence (SAI) research unit at CTTC, supports end-to-end ML development, ensuring reproducibility, efficiency, and scalability in real-world applications. It enables the monitoring of key performance indicators, such as model accuracy, complexity, memory usage, computational energy consumption, and communication overhead.

2.1.1 Architecture

In the SUCCESS-6G-DEVISE project, SUPERCOM has been used as a testbed for the proof-of-concept of user story 1.2 within a resource-efficient environment. Figure 1 depicts a high-level architecture for an edge computing solution designed for ML model serving, monitoring, and management, and orchestrated by Kubernetes. This architecture exemplifies edge computing by pushing the model inference to the edge nodes, reducing latency and bandwidth requirements. A detailed description of the building blocks of our modular monitoring solution where the involved interactions (data flows) between individual components are specified, is provided in [1].

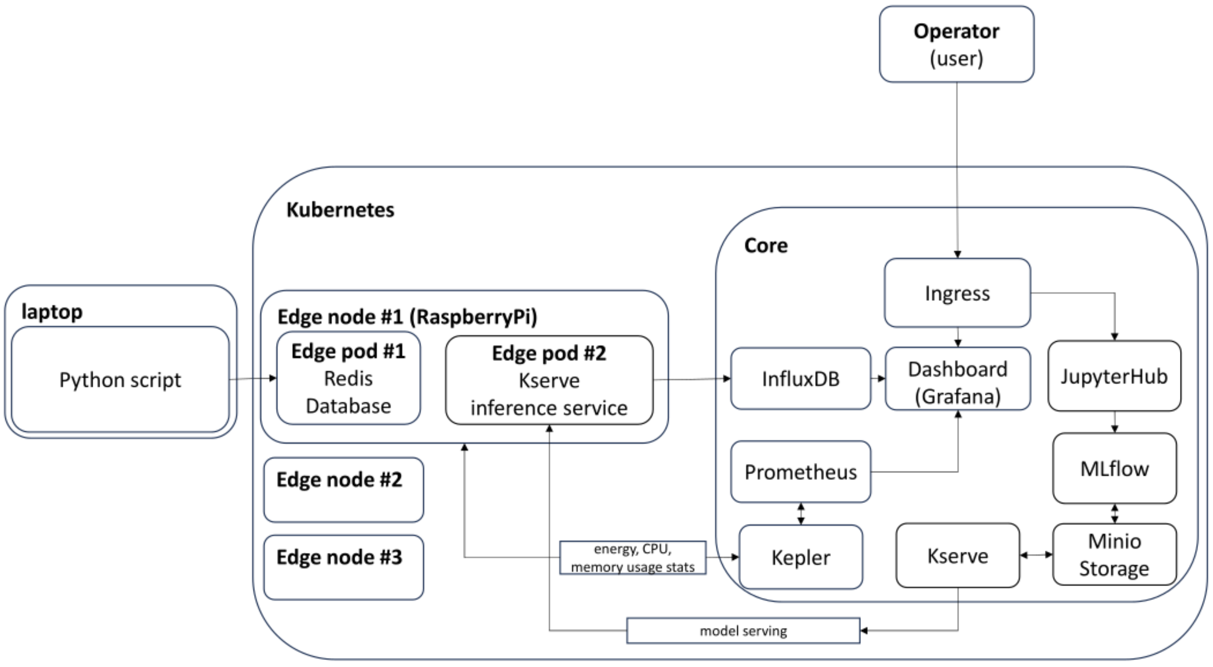


Figure 1: Basic architecture for the use case 1 at the SUPERCOM platform

In this architecture, Kubernetes is the central cluster management system, responsible for deploying and managing applications across the edge nodes. The entire system is deployed on Kubernetes, containing both edge nodes and a core system. Regarding the edge infrastructure, several edge nodes (e.g., Raspberry Pis) can be considered at distributed locations in a resource-constrained setting. Each edge node runs pods, which are the smallest deployable units in Kubernetes. Specifically:

- Edge Pod #1 (Redis Database): Runs a Redis in-memory data store, likely used for caching or real-time data processing.

- Edge Pod #2 (Kserve Inference Service): Runs the actual ML model that performs inference on the edge.

The edge nodes send statistics like energy, CPU, and memory usage to the core system for monitoring and optimization. On the other hand, the core infrastructure is responsible for managing model serving, monitoring, and logging. Several key services are integrated into the core. In particular:

- Ingress: Handles external access to the services running within the Kubernetes cluster, acting as a reverse proxy and load balancer.
- InfluxDB: A time-series database used for storing and querying metrics, likely for monitoring the performance of the edge nodes and the models.
- Dashboard (Grafana): A visualization tool that creates dashboards to monitor the data stored in InfluxDB, providing insights into system performance and model behavior.
- JupyterHub: Provides a web-based interactive development environment for software developers to work with models, analyze data, and create visualizations.
- MLflow: An open-source platform for managing the machine learning lifecycle, including experiment tracking, model packaging, and deployment.
- Prometheus: A monitoring system that collects performance metrics from the Kubernetes nodes and applications.
- Kepler: Used for monitoring energy consumption and resource utilization of the edge nodes.
- Kserve: Model-serving component integrated within both the core and edge nodes.
- Minio Storage: An object storage service, potentially used for storing model artifacts, datasets, or other data.

The inclusion of Prometheus, InfluxDB, and Grafana emphasizes the importance of monitoring and managing the performance of the edge system. The use of MLflow and Kserve suggests an emphasis on MLOps principles, aiming to streamline the machine learning lifecycle from development to deployment and monitoring. We assume that a user interacts with the system through a Python script, possibly for deploying new models, updating configurations, or monitoring the system (e.g., system health, resource consumption). For the purpose of facilitating the development of algorithms responsible for processing the information that is collected and stored in this REDIS database, a vehicle injector tool has been developed by IDNEO, as shown in Figure 2. This tool facilitates the incorporation of real data from a vehicle's sensors, stored in CSV Datasets.

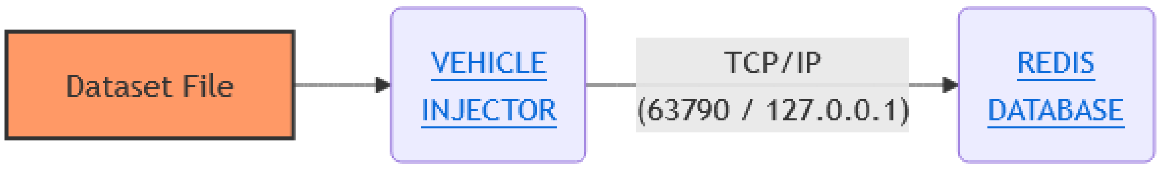


Figure 2: Vehicle injector tool

Edge pods process the data; specifically, Redis stores relevant intermediate data and Kserve performs model inference. The core's Kserve component deploys the ML model to the edge nodes. The edge nodes collect data (energy, CPU, memory usage stats) and send it to the core for monitoring and analysis. The collected data flows from the edge nodes to Prometheus, then to InfluxDB, and finally visualized through Grafana dashboards. MLflow is used to manage the lifecycle of the models, potentially including packaging and deploying them to the edge nodes through Kserve. Finally, Minio provides storage for model artifacts and potentially other data needed by the edge applications.

2.1.2 Advantages and disadvantages of a single-cluster architecture

The following advantages and disadvantages of a single-cluster architecture, as the one followed in SUPERCOM platform, can be identified:

Advantages:

- Simplified Management:
  - Easier to manage and maintain as there is only one cluster.
  - Simplified network configuration and service discovery within the cluster.
- Resource Efficiency:
  - Better utilization of resources since there is no need to allocate separate resources for multiple clusters.
- Cost-Effective:
  - Lower operational costs due to reduced overhead in managing multiple clusters.
  - Reduced complexity in monitoring and logging, leading to potential cost savings.
- Unified Security and Policy Management:
  - Simplified implementation of security policies, access controls, and compliance measures.
- Consistent Environment:
  - Ensures a consistent environment across all nodes, simplifying development and deployment processes.

Disadvantages:

- Scalability Limitations:
  - May face scalability issues as the number of edge devices or workloads increases.
  - Single point of failure; if the cluster goes down, all connected devices and services are affected.
- Performance Bottlenecks:
  - Potential for performance bottlenecks if the cluster becomes overloaded.
  - Limited fault isolation; issues in one part of the cluster can impact the entire system.
- Geographical Constraints:
  - Less optimal for geographically distributed edge nodes as latency might increase.
  - May not efficiently handle diverse network conditions across different locations.

On the contrary, a multi-cluster deployment that can utilize dUPF metrics (e.g., number of connected users) to decide if the application should be deployed in a neighboring micro cluster is adopted in the CELLNEX Mobility Lab testbed.

We provide the details in the following section.

2.2 CELLNEX Mobility Lab

2.2.1 Monitoring Service

Similar to the proof-of-concept solution at SUPERCOM described in Section 2.1.1, Figure 3 illustrates the monitoring service solution for real-world testing and data acquisition at the CELLNEX mobility lab.

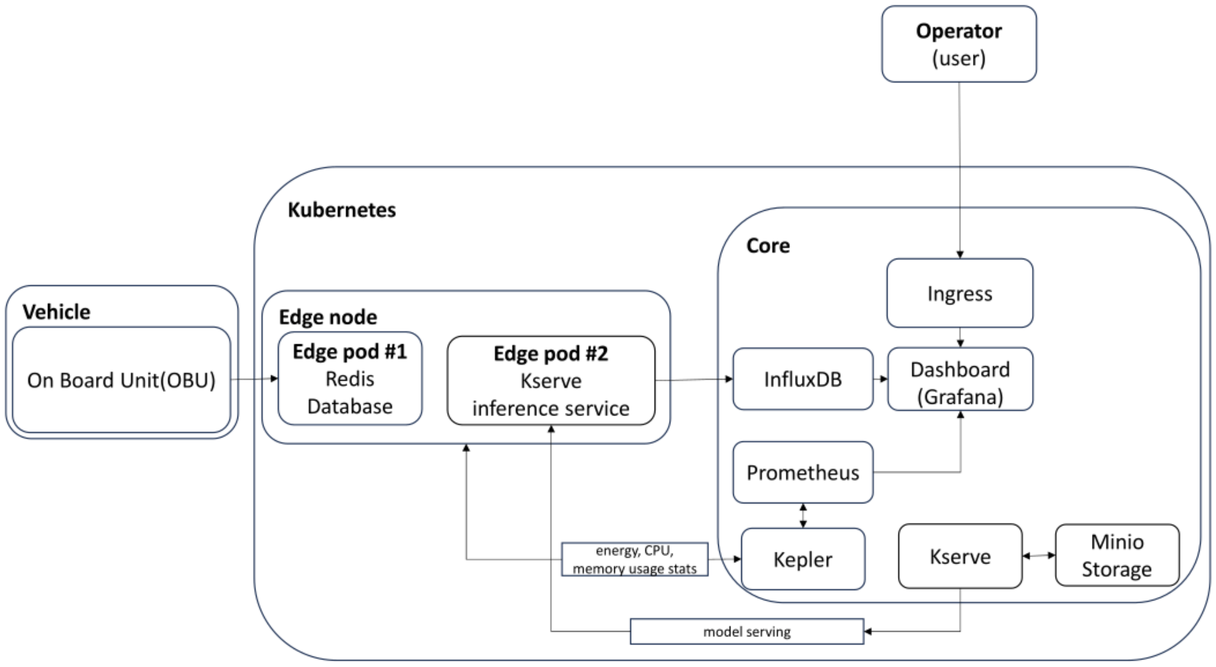


Figure 3: Architecture overview for use case 1 at Castelloli

In the following subsections, the traffic flows are described in detail.

2.2.1.1 Vehicular data processing

A vehicle collects measurements and sends them to the vehicle API service (provided by Idneo). The vehicle API service preprocesses this data and stores it in a Redis database for quick access. It is noted that the inference model requires as input the following specific features of the vehicular measurements to perform correct classification:

- Brake_pressure
- Normed_load_value
- Oil_fill_level
- Engine_oil_temperature
- Time_since_engine_start
- Accelerator_pedal_position
- Fuel_level
- Fuel_consumption
- Engine_torque
- Efficiency_of_the_SCR_catalytic_converter

The network forwards the data to the closest Kubernetes edge node using:

1. Anycast IP address – A single IP address shared by multiple edge nodes. The closest node (based on routing metrics) handles the request.

2. NodePort service – A Kubernetes service that exposes the application on a static port.

The edge node receives the data and:

1. Checks the IP table in kube-proxy to determine where to forward the request.
2. Forwards the request to Pod A (running the Vehicle API service) as the service is configured with externalTrafficPolicy: Local, as follows:
   - If a Pod for the Vehicle API service exists on the edge node, the request is handled.
   - If no local Pod exists, the packet is dropped (i.e., the request is lost).

The Vehicle API service sends the preprocessed data to the ML prediction service (Kserve inference). The ML service runs inside Pod B and is configured with internalTrafficPolicy: Local, as follows:

- If there is a local Pod for the Kserve inference service, the data is processed.
- If there is no local Pod, the packet is dropped.

Instead of waiting for incoming data, the ML prediction service (Pod B) can also periodically query the local Redis DB in the Vehicle API service. Since the Vehicle API service is also configured with internalTrafficPolicy: Local, the request follows these rules:

- If a local Pod for the Vehicle API service exists, the query is processed.
- If no local Pod exists, the query is dropped.

The Kserve inference service pushes the processed data into InfluxDB, a time-series database optimized for analytics. Grafana, a visualization tool, periodically queries data from InfluxDB to generate real-time dashboards and insights.

2.2.1.2 ML model development/update

The user opens a web browser and enters the MLflow Fully Qualified Domain Name (FQDN). The request is sent to the Ingress controller in the Kubernetes cluster. The Ingress controller receives the request, translates the FQDN to the appropriate Kubernetes service, and routes the request to the MLflow GUI service. The MLflow GUI loads and displays the available models. The user can browse, monitor, and manage the current models. The user updates a machine learning model locally or in the cloud. The updated model is then uploaded to the MLflow API. The MLflow API receives the uploaded model and forwards it to the backend database. The backend storage used is MinIO, an object storage service similar to AWS S3. MinIO provides a scalable, S3-compatible backend to store ML models efficiently.

2.2.1.3 ML model deployment

An admin user instructs Kserve (a model serving framework for Kubernetes) to deploy a machine learning model. The request is sent to the Kserve control plane, which manages model deployment. Kserve queries MinIO (the backend object storage) to retrieve the requested model. MinIO responds by providing the stored model files to Kserve. Kserve loads the model and deploys it across all Kserve inference Pods. Each inference Pod now has the updated model and is ready to serve predictions.

2.2.2 Infrastructure

2.2.2.1 Network

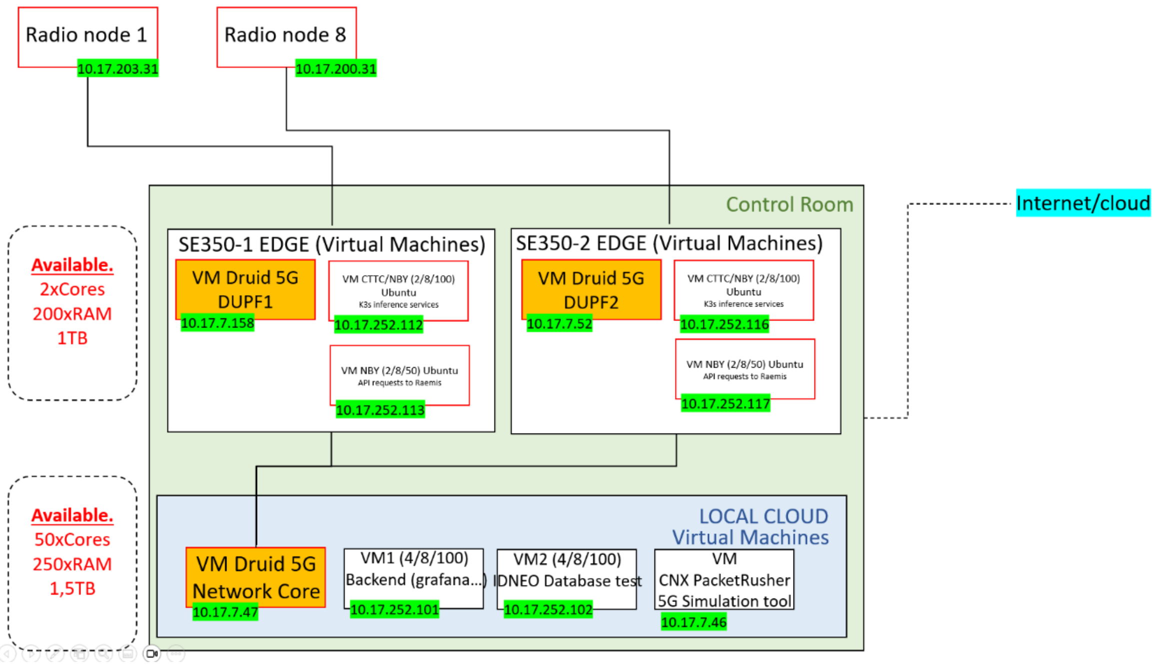


Figure 4: SUCCESS-6G overall architecture and the supported services

The final architecture deployed for the SUCCESS-6G project follows the structure outlined below (see Figure 4). Services related to the 5G standalone (SA) network are highlighted in orange, while all other services pertain to the specific use case.

The LOCAL CLOUD consists of two virtualized SR650 servers configured in a cluster (see deliverables E5 [2] and E6 [3]), enabling the creation of multiple virtual machines for various services and applications. The 5G network core is deployed within a dedicated virtual machine, isolated from all other services. Another isolated virtual machine hosts all backend components, including Grafana. The vehicle REDIS database has been deployed in a third virtual machine, ensuring its separation from other applications. Additionally, a fourth virtual machine is allocated for network measurement tools and network KPIs (5G simulation tool, iperf3, etc.).

Leveraging the infrastructure established for other projects (e.g., FREE6G), two EDGE servers (SE350-1 and SE350-2) have been deployed and virtualized separately. Figure 5 depicts the two EDGE servers in the control room.

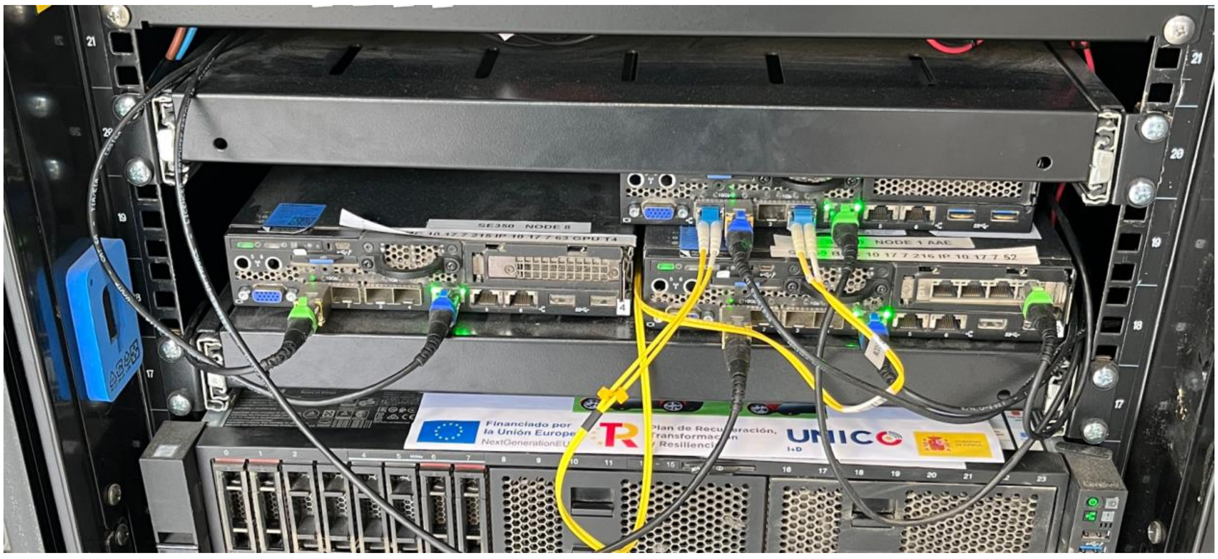


Figure 5: Edge servers in the control room

Each EDGE server is associated with a node (Node-1 and Node-8) and supports virtual machines for critical services, ensuring isolation between them. Figure 6 depicts the radio node 1.

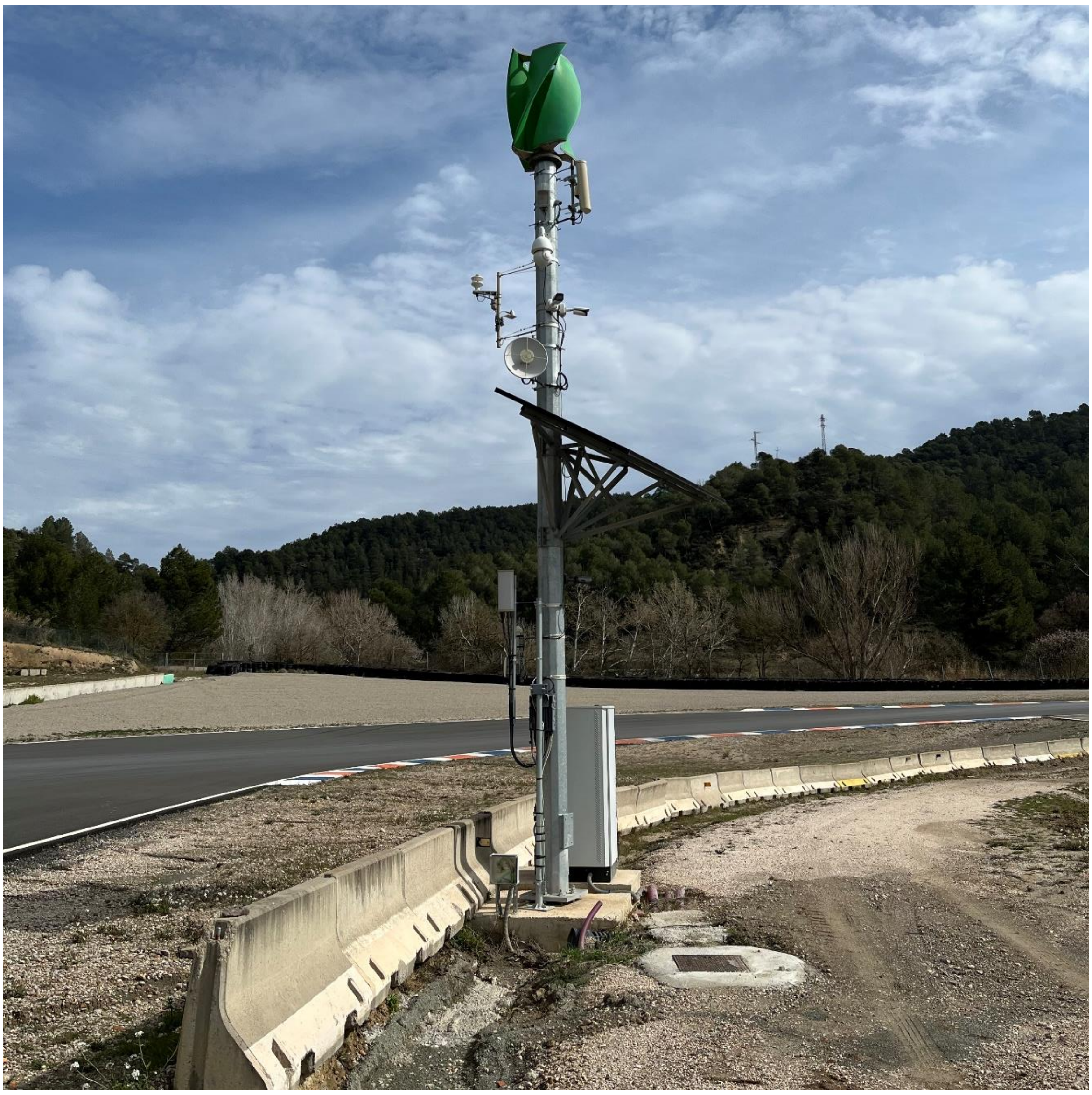


Figure 6: Radio node 1 in the CELLNEX Mobility Lab at Castelloli.

On each EDGE server:

- A dedicated virtual machine hosts the distributed user plane function (UPF) of its associated node.
- A second virtual machine is reserved for critical services (i.e., monitoring service described in Section 2.2.1).
- A third virtual machine, isolated from all others, is used for the 5G network API and network event calls.

2.2.2.2 C-V2X OBU

The hardware platform designed to meet the requirements and use cases for the SUCCESS-6G project is primarily based on two latest-generation modules specifically designed to address the latest advancements in C-V2X systems within the 5G NR environment. This OBU (depicted in Figure 7) is one of the first automotive-grade compliant devices with 5G NR Sub-6 GHz capabilities, supporting both stand-alone (SA) and Non-Stand Alone (NSA) modes.

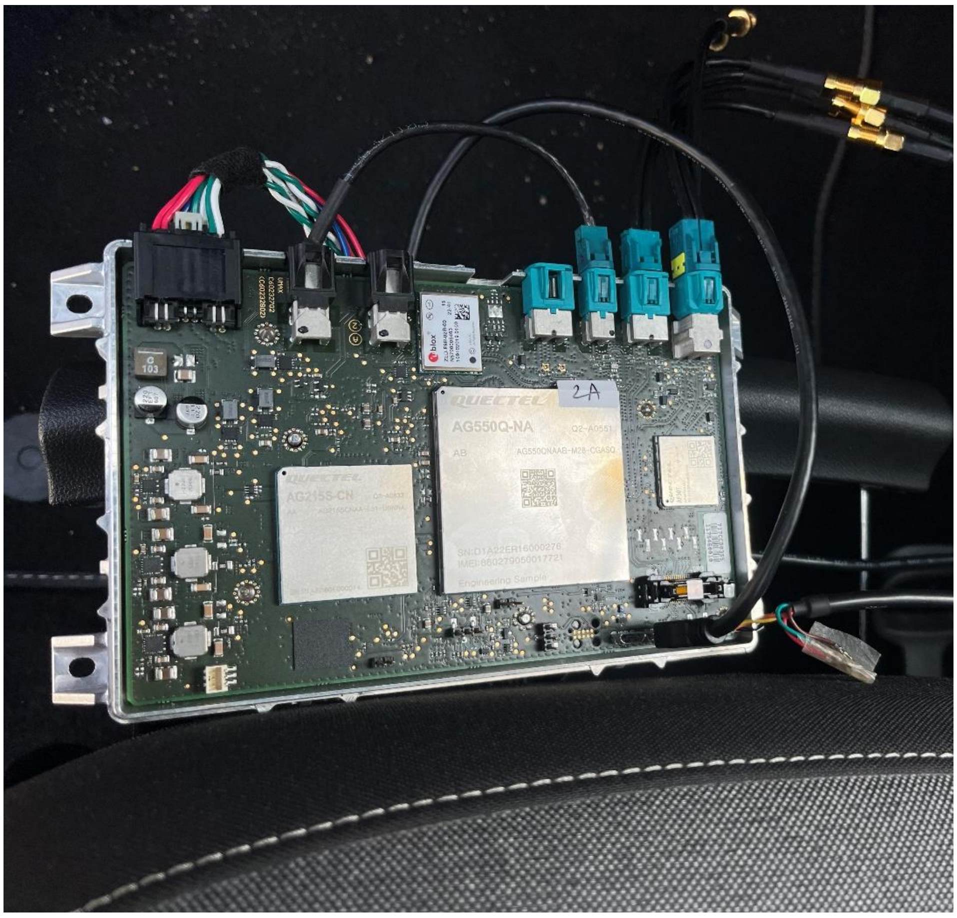


Figure 7: On-board unit (OBU) used in SUCCESS-6G trials

The OBU is specifically designed for C-V2X vehicle communications, such as advanced driving safety, autonomous driving, Intelligent Transportation Systems (ITS), and Advanced Driver Assistance Systems (ADAS). Apart from that, it also provides critical security functions in C-V2X through its ARM processors, ECDSA cryptographic module for message verification, and HSM module for message signing. This way, the AP provides all the necessary elements for executing a C-V2X Stack, a C-V2X messaging software that manages, among other functions, the verification of received messages and secure signing of sent messages.

To conduct all necessary tests, the equipment provides access to a range of communication and debugging buses, which we will utilize during testing. As this type of equipment lacks a graphical user interface, all services and applications must be managed through a command line interface (CLI). The CLI can be accessed via wired connections, such as Ethernet, USB, or serial interfaces including tools like ADB (Android Debug Bridge) and protocols such as SSH.

2.2.2.3 Service Orchestrator

The NearbyOne orchestrator is used as the service orchestrator to perform service migration across the available edge nodes based on events coming from the 5G core. To achieve this functionality, NearbyOne enables the deployment and lifecycle management of a distributed agent in the same network where the 5G core service is running. This distributed agent plays the role of a Decision Engine (DE), which is to subscribe to 5G core events, to process them, and if necessary to inform NearbyOne to perform the migration actions. Therefore, to facilitate the orchestration of applications in the edge, NearbyOne utilizes two types of interfaces: the northbound interface (NBI) and the southbound interface (SBI). The NBI provides endpoints for third-party applications, such as the DE, allowing an API client within the DE to communicate with the NearbyOne's NBI API server. Subsequently, NearbyOne generates orchestration action requests and transmits them through the SBI for execution on the underlying resource orchestration platforms, like Kubernetes, in which finally the applications are going to run.

The system architecture implements a clear separation of responsibilities across three distinct management domains (Figure 8). The 5G Core Network (blue components) maintains direct management control over core network resources including UPF, gNB ID, and TAI, while the NearbyOne Service Orchestrator (green components) independently manages Edge infrastructure (Site ID) and handles service orchestration actions. The DE (orange component) serves as a coordinator, mapping resources and processing events without direct infrastructure management. Instead, it works alongside both the 5G Core Network and NearbyOne to enable coordinated operation across these separate management domains.

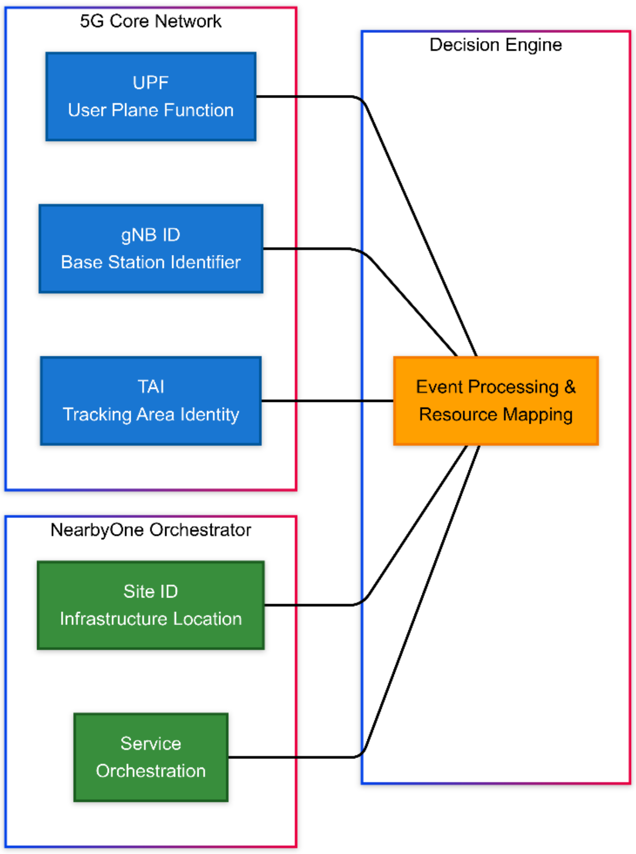


Figure 8: Management domains of service orchestrator

The DE is a sophisticated cloud-native application designed to handle events from a 5G core network and inform Nearbyone to orchestrate services based on these events. This application operates in both client and server roles, each with distinct responsibilities.

In its client role, the DE subscribes to events from the 5G core network using Kubernetes lifecycle hooks, as shown in Figure 9. This is achieved through the *postStart* and *preStop* hooks defined in its Kubernetes Deployment configuration. When the DE Pod starts, the *postStart* hook sends a subscription request to the 5G core, registering a DE's endpoint to receive specific events. This ensures that the application is notified of relevant events, such as data session create and update requests from the vehicle to the 5G core. The *preStop* hook is executed when the pod is about to terminate, sending a request to unsubscribe from the 5G core events. This cleanup step ensures that the application does not receive events when it is no longer running, maintaining the integrity of the event subscription process.

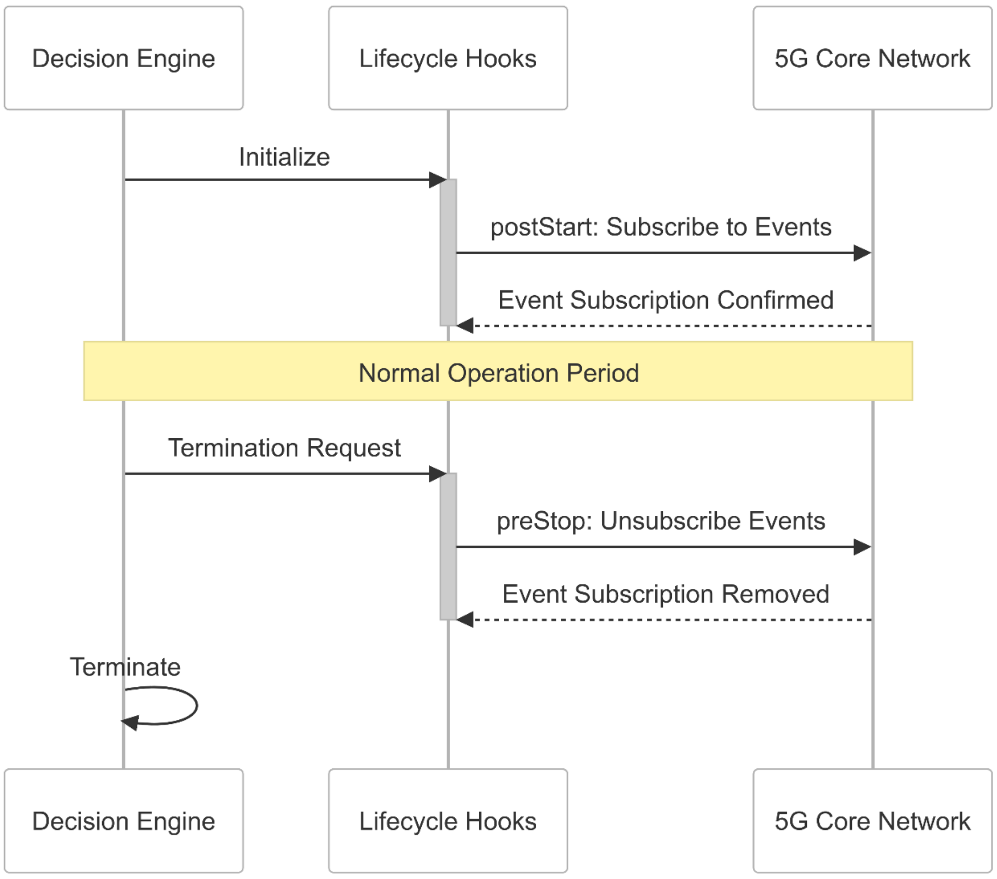


Figure 9: Operation of Decision Engine as client

In its server role, the DE listens for incoming HTTP POST requests at the */form* endpoint, which are expected to contain event data from the 5G core (Figure 10). Upon receiving an event, the server logs the event details and checks if the event's Subscriber Permanent Identifier (SUPI) matches a predefined value of interest, which is configurable via an environment variable. If the SUPI matches, the server initiates a series of actions to manage the orchestration of services within the telco infrastructure.

The core functionality of the server revolves around the *perform_action* function, which is triggered when a relevant event is received. This function maps the gNodeB ID (gNB ID) from the event to a specific edge in the telco infrastructure, identified by its User Plane Function (UPF), Tracking Area Identity (TAI), and site ID. The server then authenticates with the NearbyOne to obtain a session token, which is used to make authenticated requests to manage services via the NBI server exposed by

NearbyOne. The DE retrieves the list of services and identifies the specific service of interest, in this case, "cttc_kserve_model". If the service is running on a different site than the one specified in the event, the server updates the service's configuration to migrate it to the new site. To ensure the service migration is successful, the DE polls the service status at regular intervals (every 0.5 seconds) until the status changes to "SERVICESTATUS_IN_SYNC". This status indicates that the resources at the Kubernetes level are ready. The DE polls the NearbyOne to get the status of the migrated service. The time taken for the service migration is recorded and saved to a JSON file stored in a persistent volume, ensuring the data is not lost even if the DE Pod restarts. In addition to handling events, the DE server provides several endpoints for monitoring and managing its operations. The */health* endpoint returns the health status of the server, while the */events* endpoint allows users to view the log of received events. The */telco_infra* endpoint provides a detailed mapping of the telco infrastructure, including the UPF, gNB ID, TAI, and site ID for each edge. These endpoints facilitate easy monitoring and troubleshooting of the DE server's operations, ensuring that it functions smoothly and effectively.

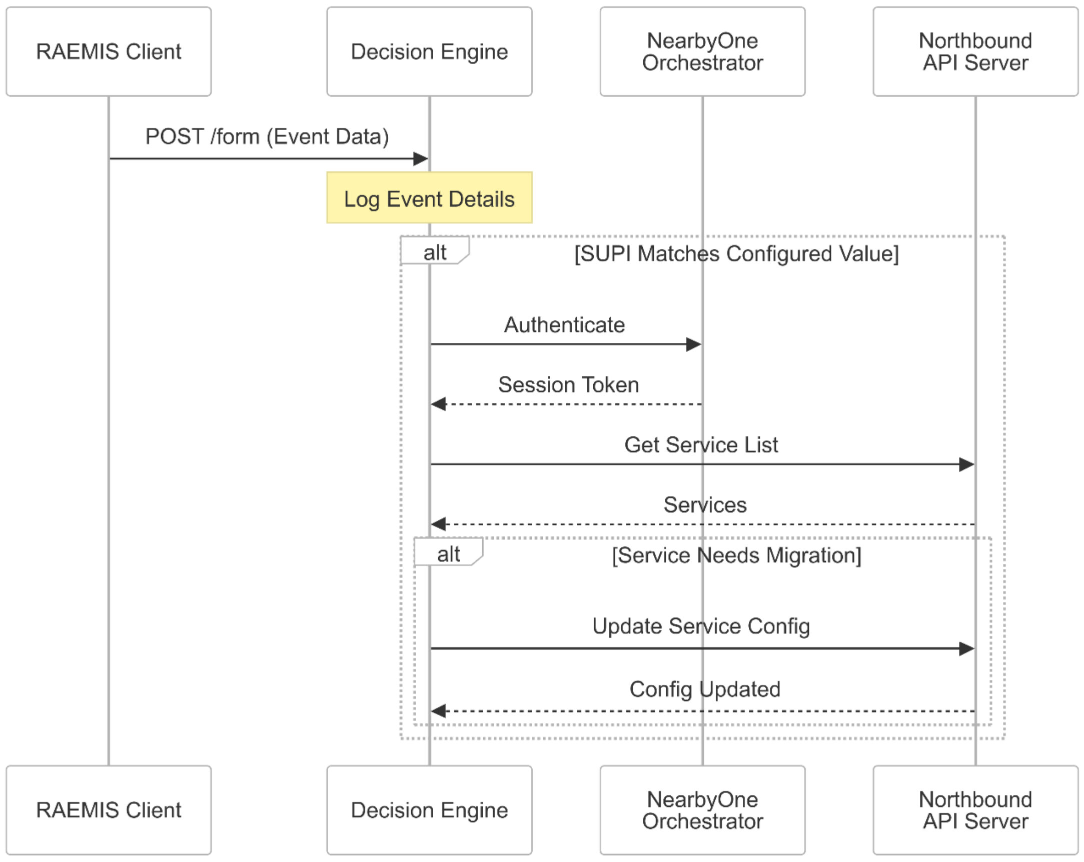
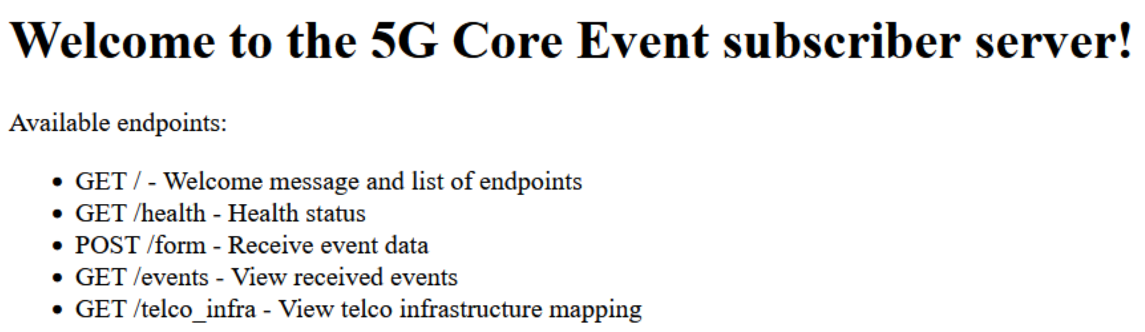


Figure 10: Operation of Decision Engine as server

In conclusion, DE’s functionalities are crucial for the dynamic and automated service orchestration based on real-time events.

The DE can be reached on the following IP and Port number: http://10.17.252.112:30080/. The landing page lists the available endpoints, as shown in Figure 11.



Welcome to the 5G Core Event subscriber server!

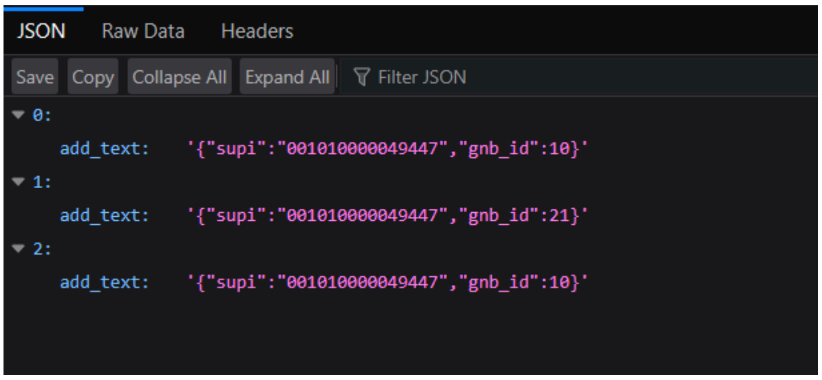
Available endpoints:

- GET / - Welcome message and list of endpoints
- GET /health - Health status
- POST /form - Receive event data
- GET /events - View received events
- GET /telco_infra - View telco infrastructure mapping

Figure 11: Available endpoints

The GET/events endpoints lists all the received events (http://10.17.252.112:30080/events).

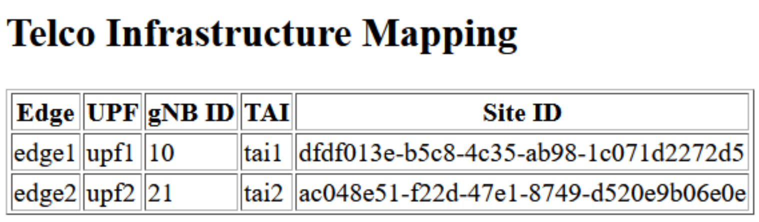
A high-level example event list with the SUPI and gNB ID can be seen in Figure 12.



```
0:
    add_text:   '{"supi":"001010000049447","gnb_id":10}'
1:
    add_text:   '{"supi":"001010000049447","gnb_id":21}'
2:
    add_text:   '{"supi":"001010000049447","gnb_id":10}'
```

Figure 12: Example event list

The GET/telco_infra endpoint (http://10.17.252.112:30080/telco_infra) shows the mapping of the 5G network and the edge infrastructure orchestrated by the Service Orchestrator (Figure 13).

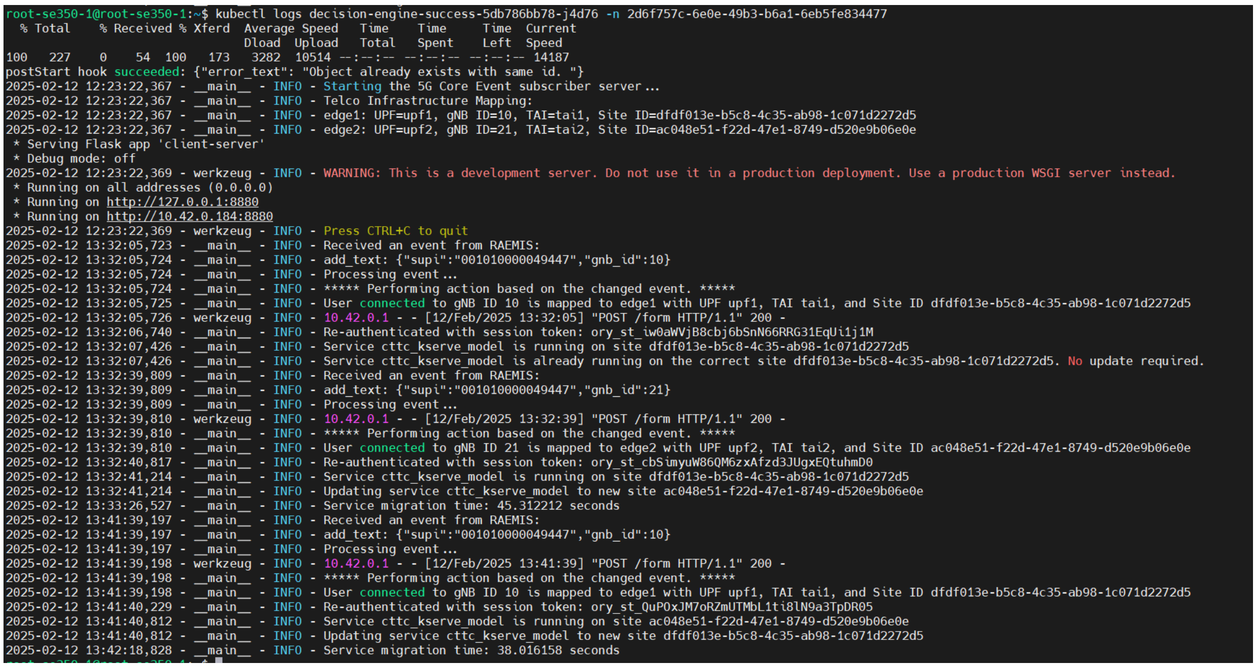


Telco Infrastructure Mapping

Edge	UPF	gNB ID	TAI	Site ID
edge1	upf1	10	tai1	dfdf013e-b5c8-4c35-ab98-1c071d2272d5
edge2	upf2	21	tai2	ac048e51-f22d-47e1-8749-d520e9b06e0e

Figure 13: Mapping of 5G network and edge infrastructure

In Figure 14, logs from the running DE capturing 5G SMF events are depicted.



```
root-se350-1@root-se350-1:~$ kubectl logs decision-engine-success-5db786bb78-j4d76 -n 2d6f757c-6e0e-49b3-b6a1-6eb5fe834477
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   227    0    54  100   173   3282  10514 --:--:-- --:--:-- --:--:-- 14187
postStart hook succeeded: {"error_text": "Object already exists with same id. "}
2025-02-12 12:23:22,367 - __main__ - INFO - Starting the 5G Core Event subscriber server...
2025-02-12 12:23:22,367 - __main__ - INFO - Telco Infrastructure Mapping:
2025-02-12 12:23:22,367 - __main__ - INFO - edge1: UPF=upf1, gNB ID=10, TAI=tai1, Site ID=dfdf013e-b5c8-4c35-ab98-1c071d2272d5
2025-02-12 12:23:22,367 - __main__ - INFO - edge2: UPF=upf2, gNB ID=21, TAI=tai2, Site ID=ac048e51-f22d-47e1-8749-d520e9b06e0e
 * Serving Flask app 'client-server'
 * Debug mode: off
2025-02-12 12:23:22,369 - werkzeug - INFO - WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8880
 * Running on http://10.42.0.184:8880
2025-02-12 12:23:22,369 - werkzeug - INFO - Press CTRL+C to quit
2025-02-12 13:32:05,723 - __main__ - INFO - Received an event from RAEMIS:
2025-02-12 13:32:05,724 - __main__ - INFO - add_text: {"supi":"001010000049447","gnb_id":10}
2025-02-12 13:32:05,724 - __main__ - INFO - Processing event...
2025-02-12 13:32:05,724 - __main__ - INFO - ***** Performing action based on the changed event. *****
2025-02-12 13:32:05,725 - __main__ - INFO - User connected to gNB ID 10 is mapped to edge1 with UPF upf1, TAI tai1, and Site ID dfdf013e-b5c8-4c35-ab98-1c071d2272d5
2025-02-12 13:32:05,726 - werkzeug - INFO - 10.42.0.1 - - [12/Feb/2025 13:32:05] "POST /form HTTP/1.1" 200 -
2025-02-12 13:32:06,740 - __main__ - INFO - Re-authenticated with session token: ory_st_iw0aWVjB8cbj6bSnN66RRG31EqUi1j1M
2025-02-12 13:32:07,426 - __main__ - INFO - Service cttc_kserve_model is running on site dfdf013e-b5c8-4c35-ab98-1c071d2272d5
2025-02-12 13:32:07,426 - __main__ - INFO - Service cttc_kserve_model is already running on the correct site dfdf013e-b5c8-4c35-ab98-1c071d2272d5. No update required.
2025-02-12 13:32:39,809 - __main__ - INFO - Received an event from RAEMIS:
2025-02-12 13:32:39,809 - __main__ - INFO - add_text: {"supi":"001010000049447","gnb_id":21}
2025-02-12 13:32:39,809 - __main__ - INFO - Processing event...
2025-02-12 13:32:39,810 - werkzeug - INFO - 10.42.0.1 - - [12/Feb/2025 13:32:39] "POST /form HTTP/1.1" 200 -
2025-02-12 13:32:39,810 - __main__ - INFO - ***** Performing action based on the changed event. *****
2025-02-12 13:32:39,810 - __main__ - INFO - User connected to gNB ID 21 is mapped to edge2 with UPF upf2, TAI tai2, and Site ID ac048e51-f22d-47e1-8749-d520e9b06e0e
2025-02-12 13:32:40,817 - __main__ - INFO - Re-authenticated with session token: ory_st_cbSimyuW86QM6zxAfzd3JUgxEQtuhmD0
2025-02-12 13:32:41,214 - __main__ - INFO - Service cttc_kserve_model is running on site dfdf013e-b5c8-4c35-ab98-1c071d2272d5
2025-02-12 13:32:41,214 - __main__ - INFO - Updating service cttc_kserve_model to new site ac048e51-f22d-47e1-8749-d520e9b06e0e
2025-02-12 13:33:26,527 - __main__ - INFO - Service migration time: 45.312212 seconds
2025-02-12 13:41:39,197 - __main__ - INFO - Received an event from RAEMIS:
2025-02-12 13:41:39,197 - __main__ - INFO - add_text: {"supi":"001010000049447","gnb_id":10}
2025-02-12 13:41:39,197 - __main__ - INFO - Processing event...
2025-02-12 13:41:39,198 - werkzeug - INFO - 10.42.0.1 - - [12/Feb/2025 13:41:39] "POST /form HTTP/1.1" 200 -
2025-02-12 13:41:39,198 - __main__ - INFO - ***** Performing action based on the changed event. *****
2025-02-12 13:41:39,198 - __main__ - INFO - User connected to gNB ID 10 is mapped to edge1 with UPF upf1, TAI tai1, and Site ID dfdf013e-b5c8-4c35-ab98-1c071d2272d5
2025-02-12 13:41:40,229 - __main__ - INFO - Re-authenticated with session token: ory_st_QuPOxJM7oRZmUTMbL1ti8lN9a3TpDR05
2025-02-12 13:41:40,812 - __main__ - INFO - Service cttc_kserve_model is running on site ac048e51-f22d-47e1-8749-d520e9b06e0e
2025-02-12 13:41:40,812 - __main__ - INFO - Updating service cttc_kserve_model to new site dfdf013e-b5c8-4c35-ab98-1c071d2272d5
2025-02-12 13:42:18,828 - __main__ - INFO - Service migration time: 38.016158 seconds
```

Figure 14: Logs from the running DE capturing 5G SMF events

The steps for network-aware service migration flow in NearbyOne are the following (Figure 15):

1. The DE subscribes to 5G Core events (SMF_sm_context).
2. Events are sent to DE.
3. The DE checks for a specific SUPI in which gNB the UE is connecting to. The DE has a Telco-Edge-Cloud Table to associate gNBs and Edge Nodes. If the UE has performed handover and the service does not exist in Edge Node, the Kserve Service is migrated, otherwise no actions are performed.
4. DE has an NBI client that sends the migration request to the NBI server, which speaks to Service Manager in order to change the Edge Node.
5. The Kserve_service is now migrated.

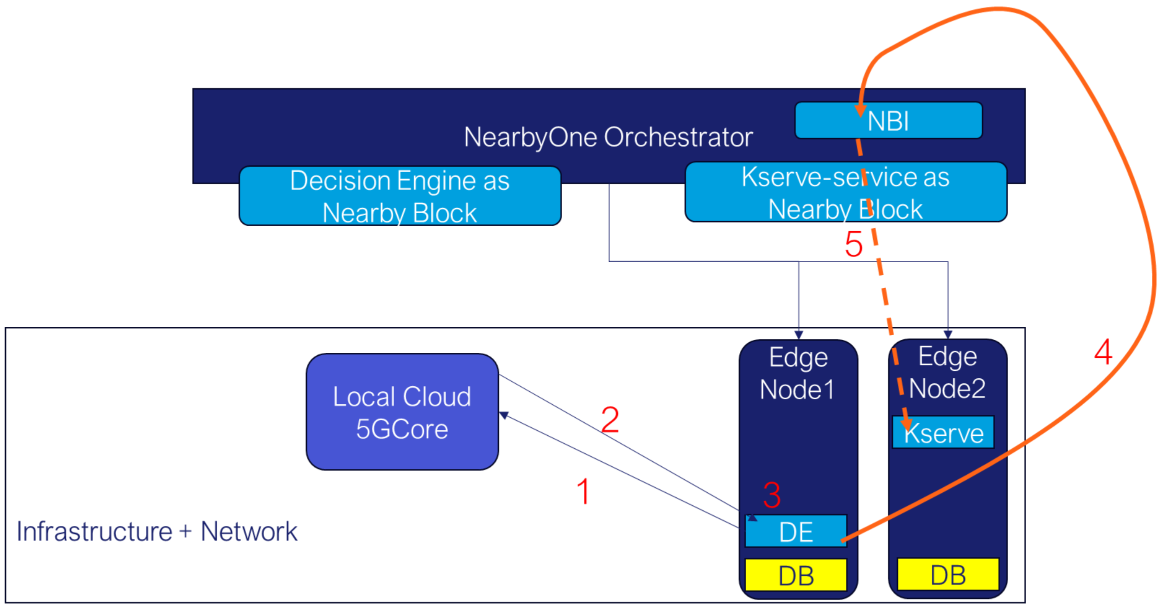
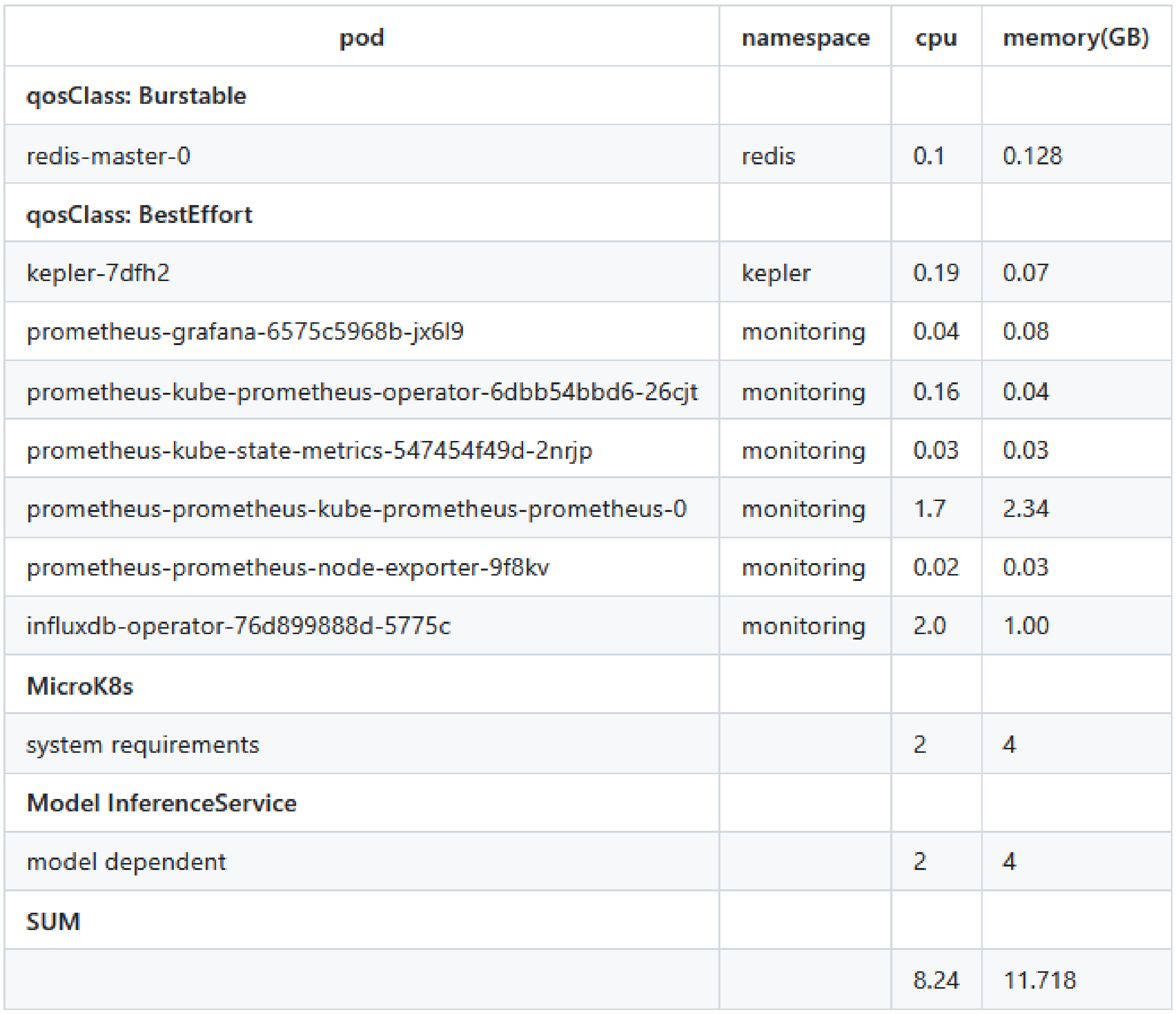


Figure 15: Steps of service migration in NearbyOne

2.2.2.4 Testing Requirements

The following minimum testing requirements have been identified for Kubernetes services' deployment at the CELLNEX mobility lab:



pod	namespace	cpu	memory(GB)
qosClass: Burstable			
redis-master-0	redis	0.1	0.128
qosClass: BestEffort			
kepler-7dfh2	kepler	0.19	0.07
prometheus-grafana-6575c5968b-jx6l9	monitoring	0.04	0.08
prometheus-kube-prometheus-operator-6dbb54bbd6-26cjt	monitoring	0.16	0.04
prometheus-kube-state-metrics-547454f49d-2nrjp	monitoring	0.03	0.03
prometheus-prometheus-kube-prometheus-prometheus-0	monitoring	1.7	2.34
prometheus-prometheus-node-exporter-9f8kv	monitoring	0.02	0.03
influxdb-operator-76d899888d-5775c	monitoring	2.0	1.00
MicroK8s			
system requirements		2	4
Model InferenceService			
model dependent		2	4
SUM			
		8.24	11.718

Table 1: Minimum testing requirements

2.2.2.5 Advantages and disadvantages of a multi-cluster architecture

The following advantages and disadvantages of a multi-cluster architecture can be identified:

Advantages:

- Scalability and fault Isolation:
  - Better scalability by distributing workloads across multiple clusters.
  - Improved fault isolation; problems in one cluster do not affect others.
- Geographical distribution:
  - Optimized for geographically distributed deployments, reducing latency by placing clusters closer to the edge devices.
  - Can handle diverse network conditions more effectively.
- Performance optimization:

    - Easier to optimize performance by dedicating clusters to specific workloads or regions.
    - Enhanced load balancing and resource distribution.
- Resilience and high availability:
    - Higher resilience and availability; failure in one cluster can be mitigated by other clusters.
    - Enhanced disaster recovery capabilities.

Disadvantages:

- Increased complexity:
    - More complex to manage and maintain multiple clusters.
    - Higher administrative overhead in coordinating and synchronizing clusters.
- Resource overhead:
    - Requires more resources for cluster management (e.g., control planes for each cluster).
    - Potentially higher infrastructure costs due to multiple clusters.
- Security and policy management:
    - More complex to implement and manage security policies and compliance across multiple clusters.
    - Potential challenges in maintaining consistent security postures across clusters.
- Inter-cluster communication:
    - Requires robust inter-cluster communication mechanisms, which can add to the complexity.
    - More complicated service discovery and networking between clusters.

2.2.3 Integration of Enablers

For the integration of the different enablers in the project, the implementation of an additional Mediator service was necessary, as shown in Figure 16. This microservice acts as a Redis consumer to adapt OBU messages into the appropriate inference request format and is responsible for handling inference responses and forwarding them to the observability backend (InfluxDB).

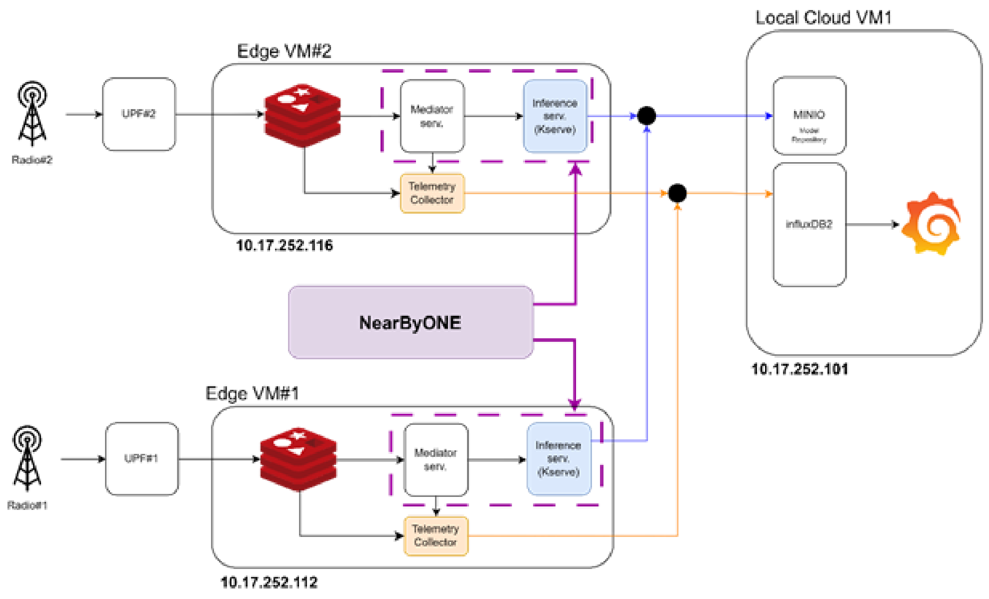
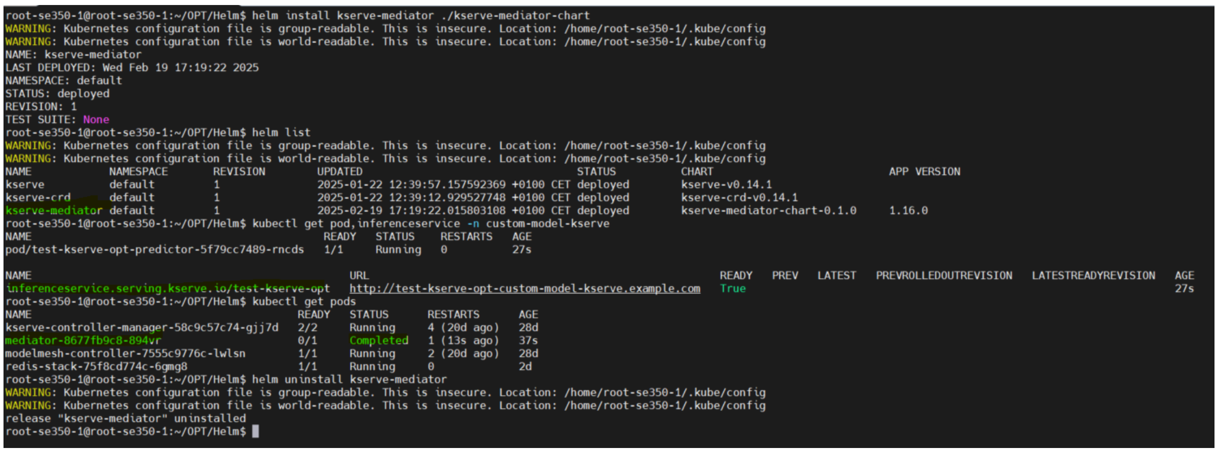


Figure 16: Integration of enablers

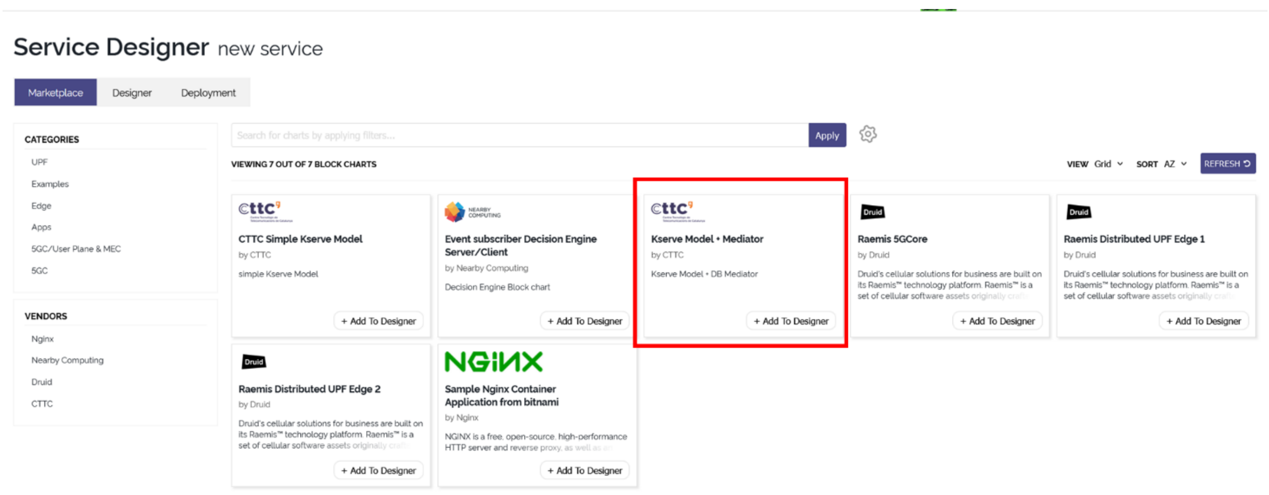
Both the Kserve and Mediator services have been packaged into a single Helm Chart, designed to be fully plug-and-play, allowing for seamless deployment on VM1 and VM2. This ensures an automated deployment process with minimal configuration required. A manual Helm test is shown in Figure 17.



```
root-se350-1@root-se350-1:~/OPT/Helm$ helm install kserve-mediator ./kserve-mediator-chart
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /home/root-se350-1/.kube/config
WARNING: Kubernetes configuration file is world-readable. This is insecure. Location: /home/root-se350-1/.kube/config
NAME: kserve-mediator
LAST DEPLOYED: Wed Feb 19 17:19:22 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
root-se350-1@root-se350-1:~/OPT/Helm$ helm list
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /home/root-se350-1/.kube/config
WARNING: Kubernetes configuration file is world-readable. This is insecure. Location: /home/root-se350-1/.kube/config
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART                           APP VERSION
kserve          default         1               2025-01-22 12:39:57.157592369 +0100 CET deployed        kserve-v0.14.1
kserve-crd      default         1               2025-01-22 12:39:12.929527748 +0100 CET deployed        kserve-crd-v0.14.1
kserve-mediator default         1               2025-02-19 17:19:22.015803108 +0100 CET deployed        kserve-mediator-chart-0.1.0     1.16.0
root-se350-1@root-se350-1:~/OPT/Helm$ kubectl get pod,inferenceservice -n custom-model-kserve
NAME                                           READY   STATUS    RESTARTS   AGE
pod/test-kserve-opt-predictor-5f79cc7489-rncds   1/1     Running   0          27s

NAME                                             URL                                                     READY   PREV   LATEST   PREVROLLEDOUTREVISION   LATESTREADYREVISION   AGE
inferenceservice.serving.kserve.io/test-kserve-opt   http://test-kserve-opt-custom-model-kserve.example.com   True                                                                  27s
root-se350-1@root-se350-1:~/OPT/Helm$ kubectl get pods
NAME                                       READY   STATUS      RESTARTS      AGE
kserve-controller-manager-58c9c57c74-gjj7d   2/2     Running     4 (20d ago)   28d
mediator-8677fb9c8-894vr                   0/1     Completed   1 (13s ago)   37s
modelmesh-controller-7555c9776c-lwlsn      1/1     Running     2 (20d ago)   28d
redis-stack-75f8cd774c-6gmg8               1/1     Running     0             2d
root-se350-1@root-se350-1:~/OPT/Helm$ helm uninstall kserve-mediator
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /home/root-se350-1/.kube/config
WARNING: Kubernetes configuration file is world-readable. This is insecure. Location: /home/root-se350-1/.kube/config
release "kserve-mediator" uninstalled
root-se350-1@root-se350-1:~/OPT/Helm$ █
```

Figure 17: Helm test

Integration with NearbyOne is then performed, following the steps depicted in the subsequent figures. First, Figure 18 shows the NearbyOne marketplace, where all the Nearby Blocks are available to be selected and deployed in the available infrastructure, which are demonstrated in Figure 19. Then, in the NearbyOne marketplace, the *Kserve Model + Mediator* Nearby Block can be selected.



Service Designer new service

Marketplace | Designer | Deployment

CATEGORIES
- UPF
- Examples
- Edge
- Apps
- 5GC/User Plane & MEC
- 5GC

VENDORS
- Nginx
- Nearby Computing
- Druid
- CTTC

Search for charts by applying filters... Apply

VIEWING 7 OUT OF 7 BLOCK CHARTS

VIEW Grid SORT AZ REFRESH

CTTC Simple Kserve Model
by CTTC
simple Kserve Model
+ Add To Designer

Event subscriber Decision Engine Server/Client
by Nearby Computing
Decision Engine Block chart
+ Add To Designer

Kserve Model + Mediator
by CTTC
Kserve Model + DB Mediator
+ Add To Designer

Raemis 5GCore
by Druid
Druid's cellular solutions for business are built on its Raemis™ technology platform. Raemis™ is a set of cellular software assets originally cra
+ Add To Designer

Raemis Distributed UPF Edge 1
by Druid
Druid's cellular solutions for business are built on its Raemis™ technology platform. Raemis™ is a set of cellular software assets originally cra
+ Add To Designer

Raemis Distributed UPF Edge 2
by Druid
Druid's cellular solutions for business are built on its Raemis™ technology platform. Raemis™ is a set of cellular software assets originally cra
+ Add To Designer

Sample Nginx Container Application from bitnami
by Nginx
NGINX is a free, open-source, high-performance HTTP server and reverse proxy, as well as an
+ Add To Designer

Figure 18: NearbyOne marketplace

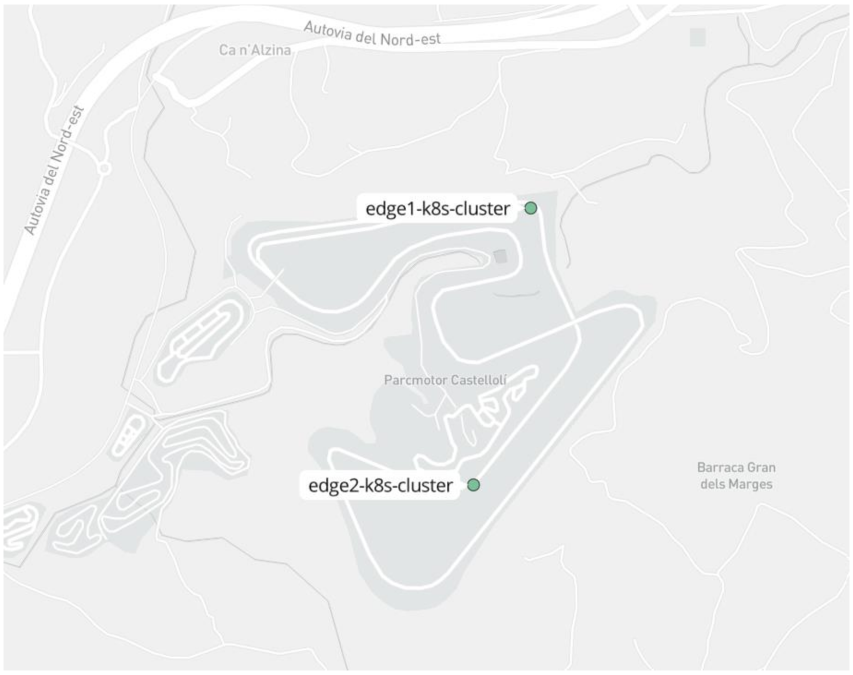
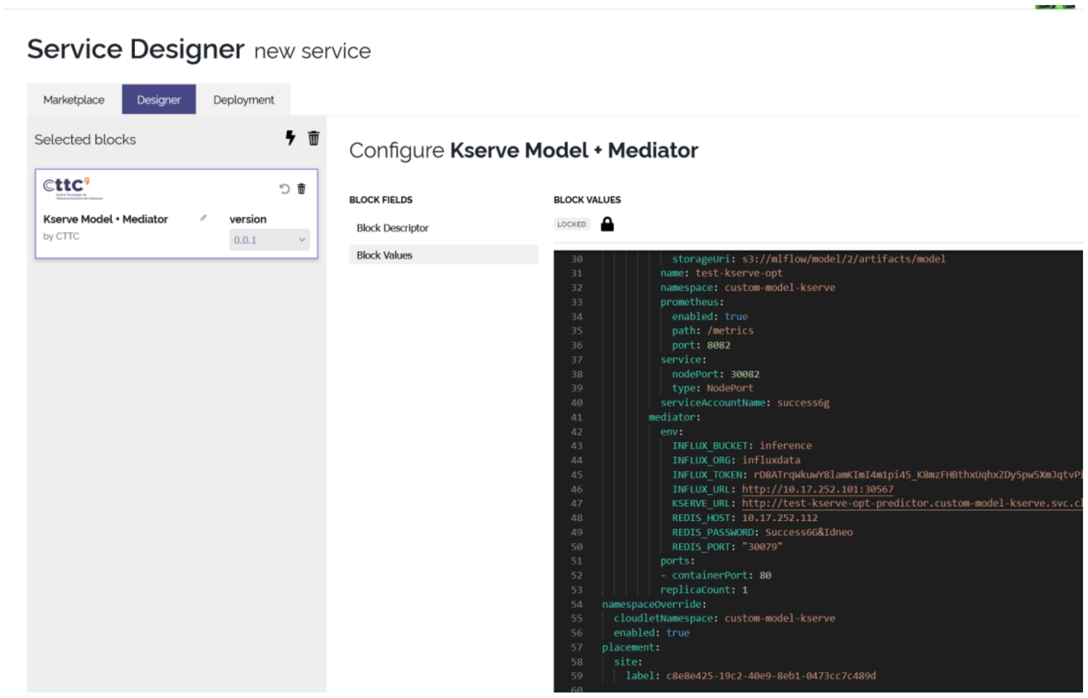


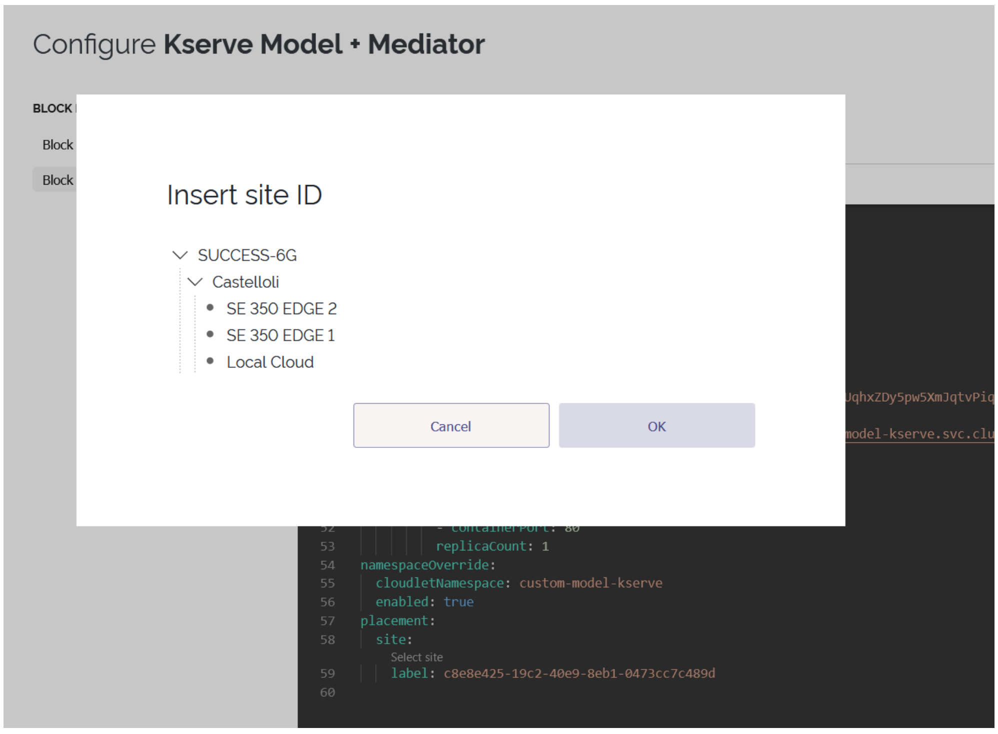
Figure 19: Edge nodes locations

Figure 20 demonstrates the block values. These values can be configured upon a user's preference. A user can select a site identifier (site ID), which corresponds to an edge node, for the service to be deployed (Figure 21).



```
            storageUri: s3://mlflow/model/2/artifacts/model
          name: test-kserve-opt
          namespace: custom-model-kserve
          prometheus:
            enabled: true
            path: /metrics
            port: 8082
          service:
            nodePort: 30082
            type: NodePort
          serviceAccountName: success6g
        mediator:
          env:
            INFLUX_BUCKET: inference
            INFLUX_ORG: influxdata
            INFLUX_TOKEN: rD8ATrqWkuwY8lamKImI4m1pi45_K8mzFHBthxUqhxZDy5pw5XmJqtvPi
            INFLUX_URL: http://10.17.252.101:30567
            KSERVE_URL: http://test-kserve-opt-predictor.custom-model-kserve.svc.cl
            REDIS_HOST: 10.17.252.112
            REDIS_PASSWORD: Success6G&Idneo
            REDIS_PORT: "30079"
          ports:
          - containerPort: 80
          replicaCount: 1
namespaceOverride:
  cloudletNamespace: custom-model-kserve
  enabled: true
placement:
  site:
    label: c8e8e425-19c2-40e9-8eb1-0473cc7c489d
```

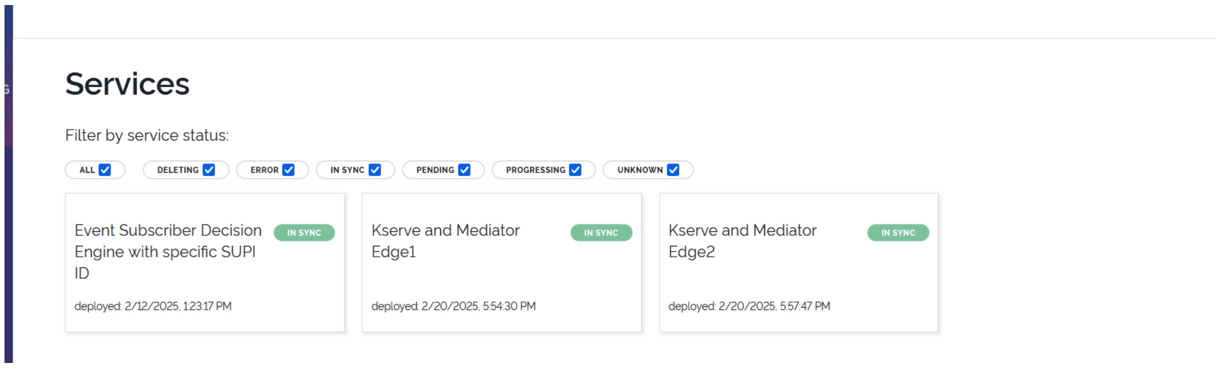
Figure 20: Block values of *Kserve Model + Mediator* Nearby block



```
UqhxZDy5pw5XmJqtvPiq
model-kserve.svc.clu
          replicaCount: 1
namespaceOverride:
  cloudletNamespace: custom-model-kserve
  enabled: true
placement:
  site:
    label: c8e8e425-19c2-40e9-8eb1-0473cc7c489d
```

Figure 21: Selection of site ID

Finally, the selected service is deployed and running. Figure 22 illustrates the *Kserve + Mediator* service to be deployed in both edge nodes and running successfully.


Services

Filter by service status:

Event Subscriber Decision Engine with specific SUPI ID — IN SYNC — deployed: 2/12/2025, 12:23:17 PM

Kserve and Mediator Edge1 — IN SYNC — deployed: 2/20/2025, 5:54:30 PM

Kserve and Mediator Edge2 — IN SYNC — deployed: 2/20/2025, 5:57:47 PM

Figure 22: NearbyOne dashboard showing the integration of Kserve and Mediator services

3 Methodology for KPI collection

3.1 Deployment of Idneo applications at Castelloli

The data acquisition architecture deployed for the SUCCESS-6G use case 1 can be seen in Figure 23. The deployment of the applications involved migrating the V2X data parser and receiver from the vehicle and deploying a REDIS database on the server side. On the OBU side, the application responsible for use case 1 has been migrated from a European variant OBU to an American variant model, as the band used in the Castellolí tests is N77.

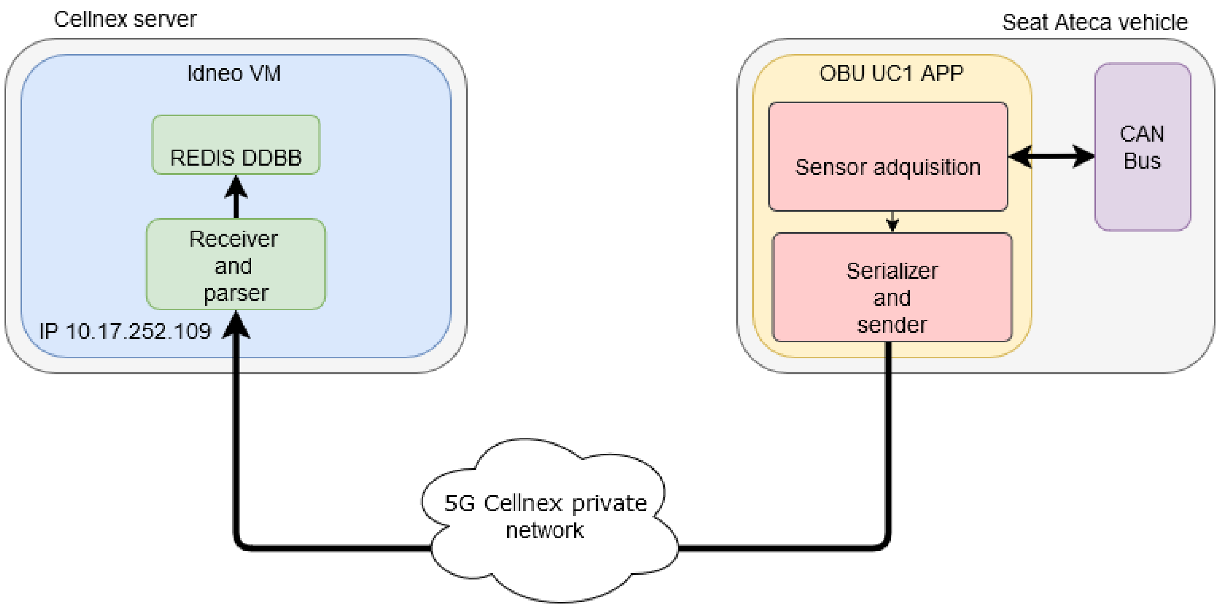
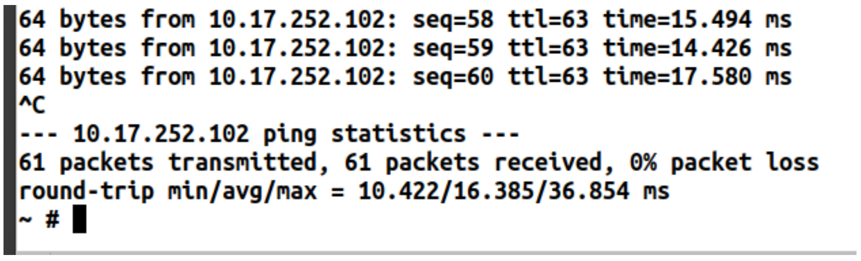


Figure 23: Data acquisition architecture

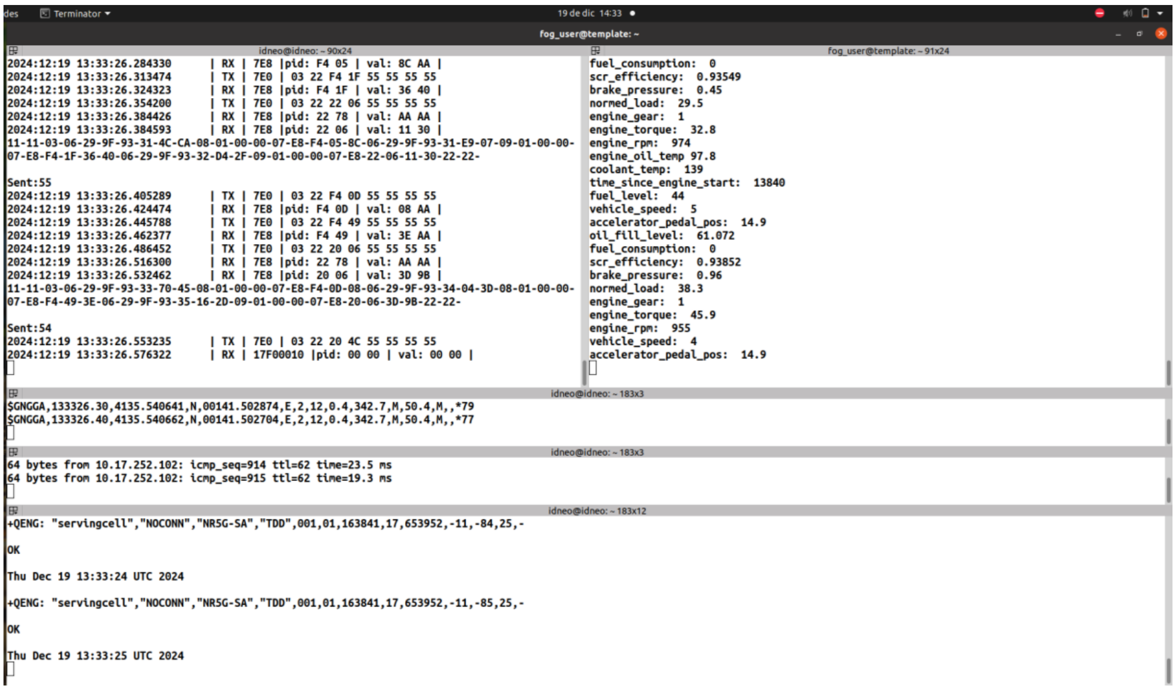
The deployment is carried out after a successful test of the connectivity at the network layer level, between the OBU and the Host that contains the applications. The test is a simple TCP/IP communication. In a ping test between the OBU and the Application Host, an average response of less than 17 milliseconds was obtained, as shown in Figure 24.



```
64 bytes from 10.17.252.102: seq=58 ttl=63 time=15.494 ms
64 bytes from 10.17.252.102: seq=59 ttl=63 time=14.426 ms
64 bytes from 10.17.252.102: seq=60 ttl=63 time=17.580 ms
^C
--- 10.17.252.102 ping statistics ---
61 packets transmitted, 61 packets received, 0% packet loss
round-trip min/avg/max = 10.422/16.385/36.854 ms
~ #
```

Figure 24: Successful connectivity test

The applications on the host that receive the data are containerized in docker. Figure 25 depicts the reception of data in the deployed applications. On the left side, it is possible to observe the acquisition of binary data in the vehicle, while on the right side, the data is already being interpreted in its characteristic units.



```
2024:12:19 13:33:26.284330      | RX | 7E8 |pid: F4 05 | val: 8C AA |
2024:12:19 13:33:26.313474      | TX | 7E0 | 03 22 F4 1F 55 55 55 55
2024:12:19 13:33:26.324323      | RX | 7E8 |pid: F4 1F | val: 36 40 |
2024:12:19 13:33:26.354200      | TX | 7E0 | 03 22 22 06 55 55 55 55
2024:12:19 13:33:26.384426      | RX | 7E8 |pid: 22 78 | val: AA AA |
2024:12:19 13:33:26.384593      | RX | 7E8 |pid: 22 06 | val: 11 30 |
11-11-03-06-29-9F-93-31-4C-CA-08-01-00-00-07-E8-F4-05-8C-06-29-9F-93-31-E9-07-09-01-00-00-
07-E8-F4-1F-36-40-06-29-9F-93-32-D4-2F-09-01-00-00-07-E8-22-06-11-30-22-22-

Sent:55
2024:12:19 13:33:26.405289      | TX | 7E0 | 03 22 F4 0D 55 55 55 55
2024:12:19 13:33:26.424474      | RX | 7E8 |pid: F4 0D | val: 08 AA |
2024:12:19 13:33:26.445788      | TX | 7E0 | 03 22 F4 49 55 55 55 55
2024:12:19 13:33:26.462377      | RX | 7E8 |pid: F4 49 | val: 3E AA |
2024:12:19 13:33:26.486452      | TX | 7E0 | 03 22 20 06 55 55 55 55
2024:12:19 13:33:26.516300      | RX | 7E8 |pid: 22 78 | val: AA AA |
2024:12:19 13:33:26.532462      | RX | 7E8 |pid: 20 06 | val: 3D 9B |
11-11-03-06-29-9F-93-33-70-45-08-01-00-00-07-E8-F4-0D-08-06-29-9F-93-34-04-3D-08-01-00-00-
07-E8-F4-49-3E-06-29-9F-93-35-16-2D-09-01-00-00-07-E8-20-06-3D-9B-22-22-

Sent:54
2024:12:19 13:33:26.553235      | TX | 7E0 | 03 22 20 4C 55 55 55 55
2024:12:19 13:33:26.576322      | RX | 17F00010 |pid: 00 00 | val: 00 00 |
```

```
fuel_consumption:  0
scr_efficiency:  0.93549
brake_pressure:  0.45
normed_load:  29.5
engine_gear:  1
engine_torque:  32.8
engine_rpm:  974
engine_oil_temp 97.8
coolant_temp:  139
time_since_engine_start:  13840
fuel_level:  44
vehicle_speed:  5
accelerator_pedal_pos:  14.9
oil_fill_level:  61.072
fuel_consumption:  0
scr_efficiency:  0.93852
brake_pressure:  0.96
normed_load:  38.3
engine_gear:  1
engine_torque:  45.9
engine_rpm:  955
vehicle_speed:  4
accelerator_pedal_pos:  14.9
```

```
$GNGGA,133326.30,4135.540641,N,00141.502874,E,2,12,0.4,342.7,M,50.4,M,,*79
$GNGGA,133326.40,4135.540662,N,00141.502704,E,2,12,0.4,342.7,M,50.4,M,,*77
```

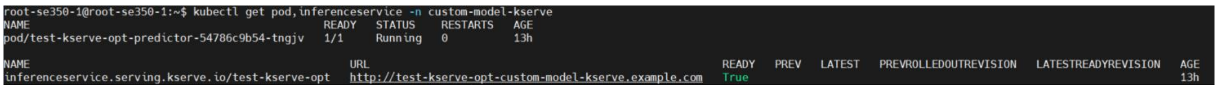
```
64 bytes from 10.17.252.102: icmp_seq=914 ttl=62 time=23.5 ms
64 bytes from 10.17.252.102: icmp_seq=915 ttl=62 time=19.3 ms
```

```
+QENG: "servingcell","NOCONN","NR5G-SA","TDD",001,01,163841,17,653952,-11,-84,25,-

OK

Thu Dec 19 13:33:24 UTC 2024

+QENG: "servingcell","NOCONN","NR5G-SA","TDD",001,01,163841,17,653952,-11,-85,25,-

OK

Thu Dec 19 13:33:25 UTC 2024
```

Figure 25: Reception of vehicular data

3.2 Monitoring service tests

As shown in Figure 26 for VM1 (10.17.252.112), the Kserve inference service was deployed and showed to be running successfully.



```
root-se350-1@root-se350-1:~$ kubectl get pod,inferenceservice -n custom-model-kserve
NAME                                          READY   STATUS    RESTARTS   AGE
pod/test-kserve-opt-predictor-54786c9b54-tngjv   1/1     Running   0          13h

NAME                                          URL                                                 READY   PREV   LATEST   PREVROLLEDOUTREVISION   LATESTREADYREVISION   AGE
inferenceservice.serving.kserve.io/test-kserve-opt   http://test-kserve-opt-custom-model-kserve.example.com   True                                                                     13h
```

Figure 26: Successful Kserve inference test

Two inference tests were performed:

- Test 1: Normal Scenario (**No Anomaly Detected**): In this test, normal metrics were sent with the label class 0 from the test dataset (using cURL commands executed within a temporary test container running inside the cluster).

Inference Result:

{"model_name":"test-kserve-opt","id":"91870bcb-6f38-41fb-820a-3882a0608f58","parameters":{"content_type":"np"},"outputs":[{"name":"output-1","shape":[1,1],"datatype":"FP64","parameters":{"content_type":"np"},"data":[**1.729031182342022e-7**]}]}/

The inference result is **~0** (highlighted in bold), which correctly indicates **no anomaly**.

- Test 2: Simulated **Anomalous** Scenario: In this case, random values were sent to trigger anomaly detection.

Inference Result:

{"model_name":"test-kserve-opt","id":"e03360a1-b93c-4557-b495-d4d03240a770","parameters":{"content_type":"np"},"outputs":[{"name":"output-1","shape":[1,1],"datatype":"FP64","parameters":{"content_type":"np"},"data":[**0.9999785508707971**]}]}/ #

The inference result is **~1** (highlighted in bold), indicating the detection of an **anomaly** as expected.

3.3 Methodology for KPI collection and visualization

To conduct the tests necessary to collect the KPIs for use case 1, an initial test plan has been designed with measurements to be taken at 6 points relative to Node 1, as shown in Figure 27. As these points are in the line of sight of both nodes 1 and 8, they can be used as measurement points in the future. At each of these points, latency, download and upload throughput, and signal quality measurements have been taken.

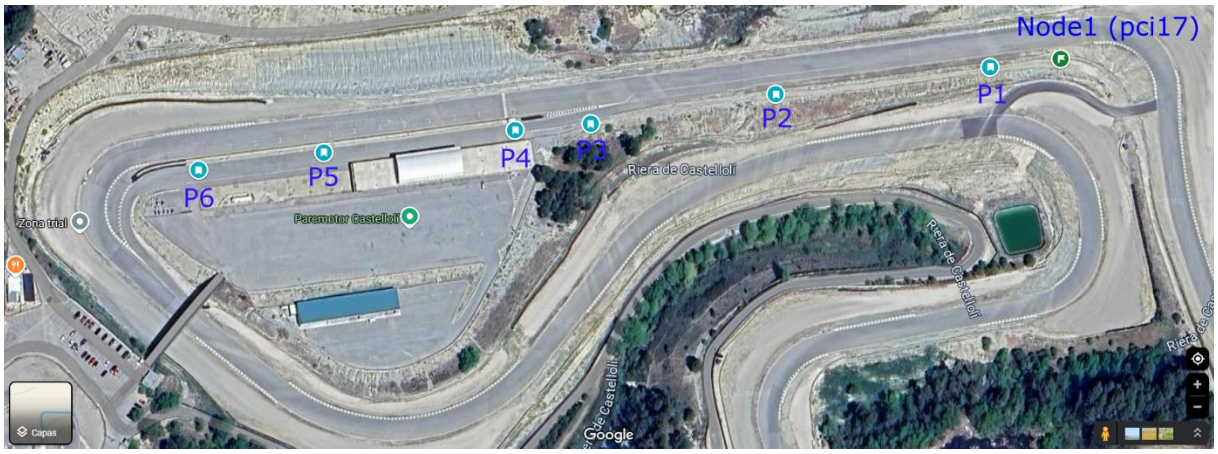
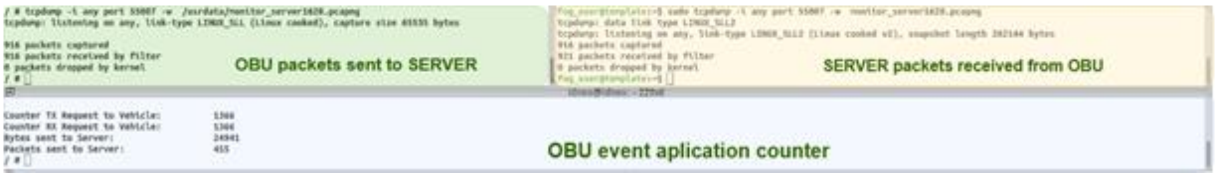


Figure 27: Castellolí testbed and reference points for measures

Regarding data collection and visualization on the OBU, given that this is an embedded platform with limited processing power, all tools must be run manually using the CLI and tools such as TCPDUMP for network traffic analysis, PING for latency measurement, IPERF3 for throughput, and the Qualcomm ql_sdk_api_test API for signal quality.

An example of how KPI measurements regarding the number of lost packages or anomalies have been carried out is shown in Figure 28.


OBU packets sent to SERVER
SERVER packets received from OBU
OBU event aplication counter

```
/ # tcpdump -i any port 55007 -w  /usrdata/monitor_server1628.pcapng
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 65535 bytes

916 packets captured
916 packets received by filter
0 packets dropped by kernel
/ #
```

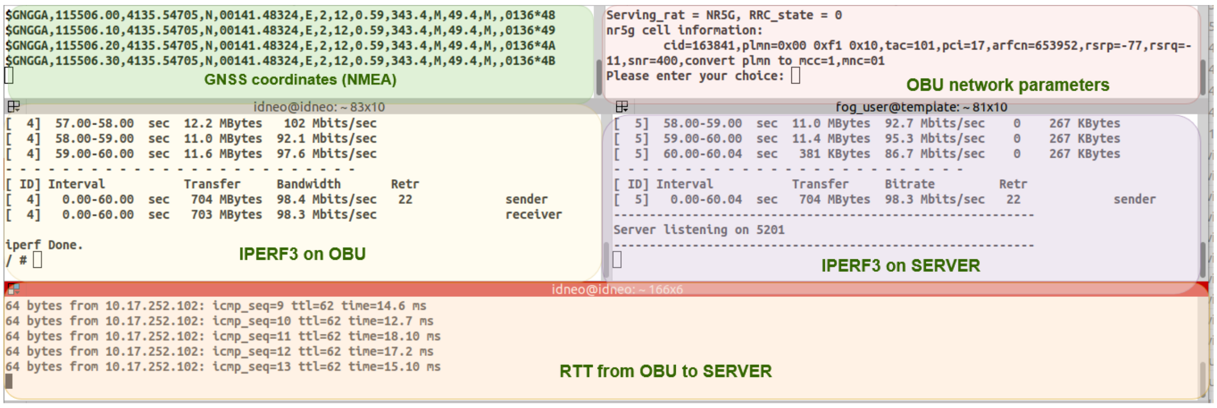
```
[illegible]:~$ sudo tcpdump -i any port 55007 -w  monitor_server1628.pcapng
tcpdump: data link type LINUX_SLL2
tcpdump: listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
916 packets captured
921 packets received by filter
0 packets dropped by kernel
[illegible]:~$
```

```
Counter TX Request to Vehicle:        1366
Counter RX Request to Vehicle:        1366
Bytes sent to Server:                 24941
Packets sent to Server:               455
/ #
```

Figure 28: Packet counter for lost packets or anomalies KPIs

Controlled tests with fixed time intervals were conducted to evaluate the performance of the application responsible for acquiring sensor data and subsequently transmitting it to the server. To analyze the traffic generated by the On-Board Unit (OBU) and the virtual machine, packet capture was performed on the TCP port designated for data transfer, specifically port 55007. Within the application, counters were implemented to record both requests and responses for vehicle data obtained through the OBD-II port. As observed in the image above, the number of packets captured on the TCP port is twice the number recorded by the application. This discrepancy occurs because the capture tool accounts not only for data packets but also for handshakes and ACK response packets.

An example of how KPI measurements regarding the throughput and latency have been carried out is shown in Figure 29.



```
$GNGGA,115506.00,4135.54705,N,00141.48324,E,2,12,0.59,343.4,M,49.4,M,,0136*48
$GNGGA,115506.10,4135.54705,N,00141.48324,E,2,12,0.59,343.4,M,49.4,M,,0136*49
$GNGGA,115506.20,4135.54705,N,00141.48324,E,2,12,0.59,343.4,M,49.4,M,,0136*4A
$GNGGA,115506.30,4135.54705,N,00141.48324,E,2,12,0.59,343.4,M,49.4,M,,0136*4B
```

```
Serving_rat = NR5G, RRC_state = 0
nr5g cell information:
        cid=163841,plmn=0x00 0xf1 0x10,tac=101,pci=17,arfcn=653952,rsrp=-77,rsrq=-
11,snr=400,convert plmn to mcc=1,mnc=01
Please enter your choice:
```

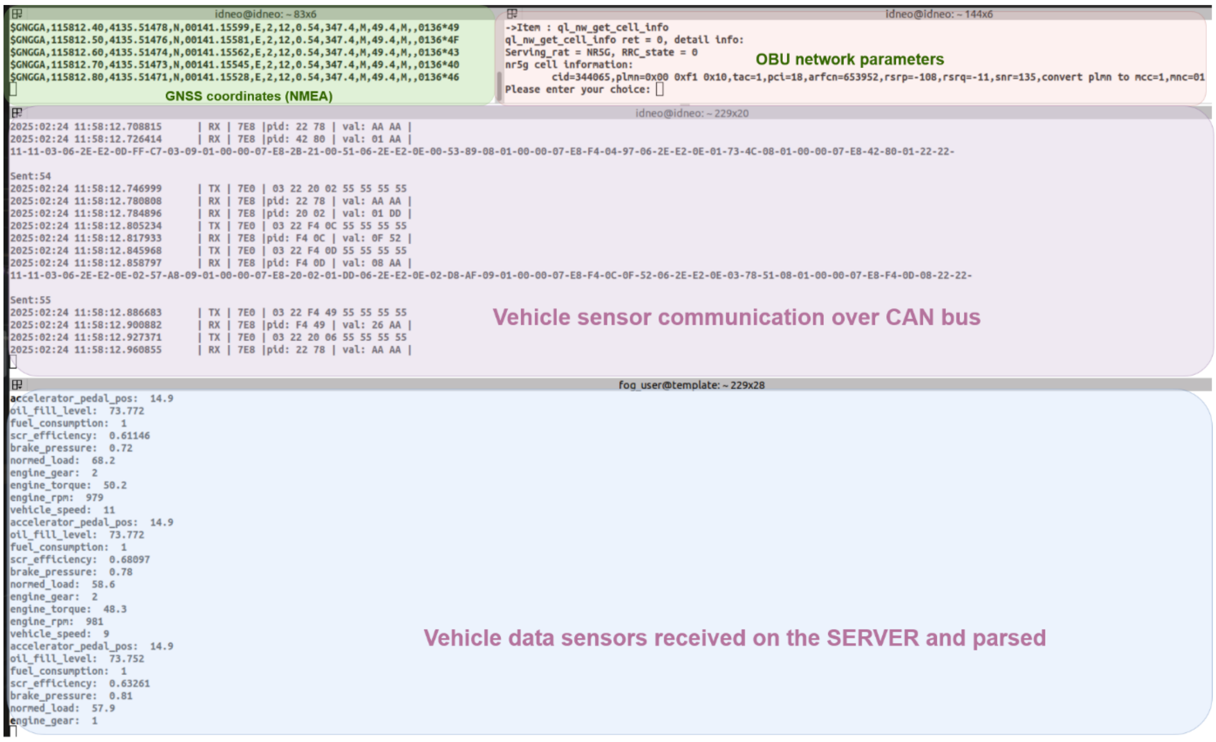
```
[  4]  57.00-58.00  sec  12.2 MBytes   102 Mbits/sec
[  4]  58.00-59.00  sec  11.0 MBytes  92.1 Mbits/sec
[  4]  59.00-60.00  sec  11.6 MBytes  97.6 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-60.00  sec   704 MBytes  98.4 Mbits/sec   22             sender
[  4]   0.00-60.00  sec   703 MBytes  98.3 Mbits/sec                  receiver

iperf Done.
/ #
```

```
[  5]  58.00-59.00  sec  11.0 MBytes  92.7 Mbits/sec    0    267 KBytes
[  5]  59.00-60.00  sec  11.4 MBytes  95.3 Mbits/sec    0    267 KBytes
[  5]  60.00-60.04  sec   381 KBytes  86.7 Mbits/sec    0    267 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-60.04  sec   704 MBytes  98.3 Mbits/sec   22             sender
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

```
64 bytes from 10.17.252.102: icmp_seq=9 ttl=62 time=14.6 ms
64 bytes from 10.17.252.102: icmp_seq=10 ttl=62 time=12.7 ms
64 bytes from 10.17.252.102: icmp_seq=11 ttl=62 time=18.10 ms
64 bytes from 10.17.252.102: icmp_seq=12 ttl=62 time=17.2 ms
64 bytes from 10.17.252.102: icmp_seq=13 ttl=62 time=15.10 ms
```

Figure 29: Throughput and latency KPIs

An example of how KPI measurements regarding the CAN data have been carried out is shown in Figure 30.


idneo@idneo: ~ 83x6
GNSS coordinates (NMEA)
idneo@idneo: ~ 144x6
OBU network parameters
idneo@idneo: ~ 229x20
Vehicle sensor communication over CAN bus
fog_user@template: ~ 229x28
Vehicle data sensors received on the SERVER and parsed

```
$GNGGA,115812.40,4135.51478,N,00141.15599,E,2,12,0.54,347.4,M,49.4,M,,0136*49
$GNGGA,115812.50,4135.51476,N,00141.15581,E,2,12,0.54,347.4,M,49.4,M,,0136*4F
$GNGGA,115812.60,4135.51474,N,00141.15562,E,2,12,0.54,347.4,M,49.4,M,,0136*43
$GNGGA,115812.70,4135.51473,N,00141.15545,E,2,12,0.54,347.4,M,49.4,M,,0136*40
$GNGGA,115812.80,4135.51471,N,00141.15528,E,2,12,0.54,347.4,M,49.4,M,,0136*46
```

```
->Item : ql_nw_get_cell_info
ql_nw_get_cell_info ret = 0, detail info:
Serving_rat = NR5G, RRC_state = 0
nr5g cell information:
        cid=344065,plmn=0x00 0xf1 0x10,tac=1,pci=18,arfcn=653952,rsrp=-108,rsrq=-11,snr=135,convert plmn to mcc=1,mnc=01
Please enter your choice:
```

```
2025:02:24 11:58:12.708815      | RX | 7E8 |pid: 22 78 | val: AA AA |
2025:02:24 11:58:12.726414      | RX | 7E8 |pid: 42 80 | val: 01 AA |
11-11-03-06-2E-E2-0D-FF-C7-03-09-01-00-00-07-E8-2B-21-00-51-06-2E-E2-0E-00-53-89-08-01-00-00-07-E8-F4-04-97-06-2E-E2-0E-01-73-4C-08-01-00-00-07-E8-42-80-01-22-22-

Sent:54
2025:02:24 11:58:12.746999      | TX | 7E0 | 03 22 20 02 55 55 55 55
2025:02:24 11:58:12.780808      | RX | 7E8 |pid: 22 78 | val: AA AA |
2025:02:24 11:58:12.784896      | RX | 7E8 |pid: 20 02 | val: 01 DD |
2025:02:24 11:58:12.805234      | TX | 7E0 | 03 22 F4 0C 55 55 55 55
2025:02:24 11:58:12.817933      | RX | 7E8 |pid: F4 0C | val: 0F 52 |
2025:02:24 11:58:12.845968      | TX | 7E0 | 03 22 F4 0D 55 55 55 55
2025:02:24 11:58:12.858797      | RX | 7E8 |pid: F4 0D | val: 08 AA |
11-11-03-06-2E-E2-0E-02-57-A8-09-01-00-00-07-E8-20-02-01-DD-06-2E-E2-0E-02-D8-AF-09-01-00-00-07-E8-F4-0C-0F-52-06-2E-E2-0E-03-78-51-08-01-00-00-07-E8-F4-0D-08-22-22-

Sent:55
2025:02:24 11:58:12.886683      | TX | 7E0 | 03 22 F4 49 55 55 55 55
2025:02:24 11:58:12.900882      | RX | 7E8 |pid: F4 49 | val: 26 AA |
2025:02:24 11:58:12.927371      | TX | 7E0 | 03 22 20 06 55 55 55 55
2025:02:24 11:58:12.960855      | RX | 7E8 |pid: 22 78 | val: AA AA |
```

```
accelerator_pedal_pos:  14.9
oil_fill_level:  73.772
fuel_consumption:  1
scr_efficiency:  0.61146
brake_pressure:  0.72
normed_load:  68.2
engine_gear:  2
engine_torque:  50.2
engine_rpm:  979
vehicle_speed:  11
accelerator_pedal_pos:  14.9
oil_fill_level:  73.772
fuel_consumption:  1
scr_efficiency:  0.68097
brake_pressure:  0.78
normed_load:  58.6
engine_gear:  2
engine_torque:  48.3
engine_rpm:  981
vehicle_speed:  9
accelerator_pedal_pos:  14.9
oil_fill_level:  73.752
fuel_consumption:  1
scr_efficiency:  0.63261
brake_pressure:  0.81
normed_load:  57.9
engine_gear:  1
```

Figure 30: Vehicle sensors acquisition and communication

In addition to the KPIs, a table has been created showing values for throughput, RTT, RSRP, RSRQ, and SNR, where the behavior of these parameters referenced to the six points can be observed.

POINT	TYPE	THROUGHPUT (Mbps)	RTT (ms)	RSRP (dBm)	RSRQ (dB)	SNR (dB)
P1	UL	32	13.3 – 20.1	-77	-11	40.0
P1	DL	98.4	12.7 – 18.1	-77	-11	40.0
P2	UL	31.8	13.5 – 18.4	-90	-11	28.5
P2	DL	94.8	12.3 – 26.4	-91	-11	26.5
P3	UL	43	18.4 – 25.3	-106	-12	3.5

P3	DL	92.7	18.4 – 25.3	-106	-12	6.5
P4	UL	51.9	14.6 – 25.2	-103	-11	11.5
P4	DL	92.4	13.8 – 22.9	-102	-11	9.5
P5	UL	18.1	23.8 – 31.8	-103	-11	13.5
P5	DL	52.8	28.1 – 34.2	-106	-12	10.5
P6	UL	32	13.3 – 20.1	-77	-11	40.0
P6	DL	98.4	12.7 – 18.1	-77	-11	40.0

Table 2: Throughput, RTT, RSRP, RSRQ, and SNR at different points

It can be seen in Table 2, the measurement points of the Uplink and Downlink. Points P1 and P6 exhibit the best signal quality, with an RSRP of -77 dBm and a great SNR, resulting in high download speeds (98.4 Mbps). In contrast, P3 shows the weakest signal, yet maintains a reasonable throughput (92.7 Mbps). RTT varies significantly across locations. P5, with a moderate signal (-103 dBm RSRP), experiences higher latency (23.8–31.8 ms) and the lowest download speed (52.8 Mbps), highlighting the impact of weak signals on network performance. Conversely, P1 and P6 maintain lower RTT values (12.7–20.1 ms), ensuring a more stable connection.

Nearby Computing monitored the status of services from NearbyOne dashboard throughout the experiments (Figure 31) and the events related to the vehicle data session (Figure 32).

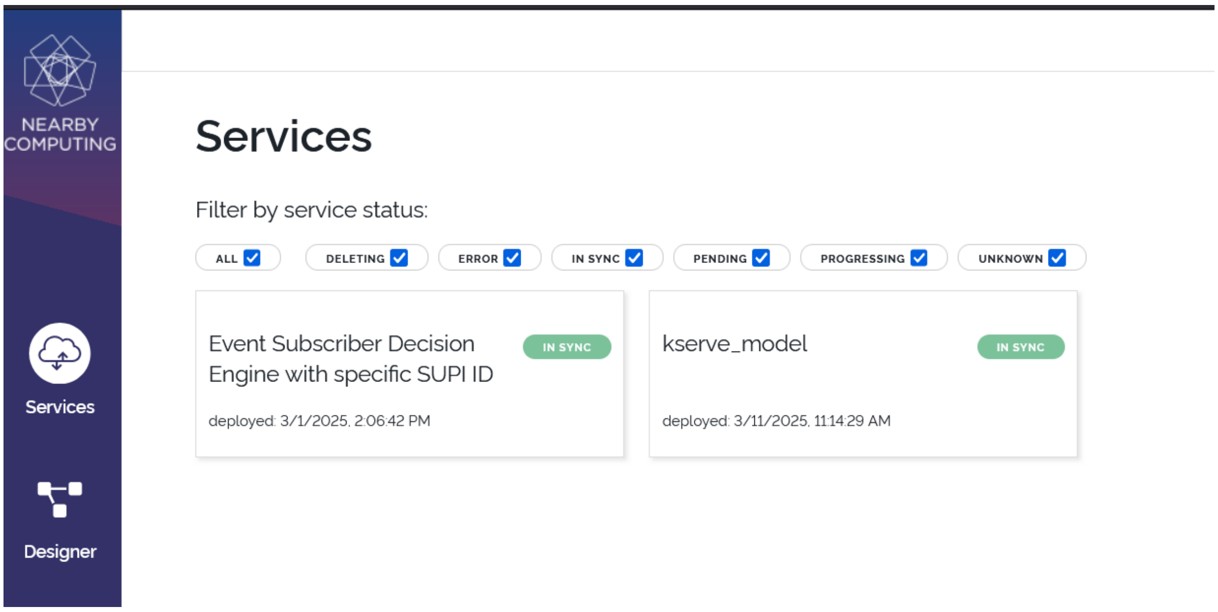
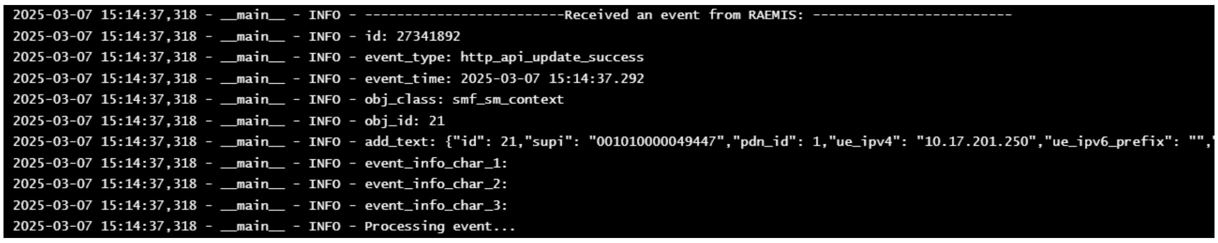

NEARBY COMPUTING
Services
Designer
Services
Filter by service status:
ALL
DELETING
ERROR
IN SYNC
PENDING
PROGRESSING
UNKNOWN
Event Subscriber Decision Engine with specific SUPI ID
IN SYNC
deployed: 3/1/2025, 2:06:42 PM
kserve_model
IN SYNC
deployed: 3/11/2025, 11:14:29 AM

Figure 31: Service monitoring through NearbyOne



```
2025-03-07 15:14:37,318 - __main__ - INFO - ------------------------Received an event from RAEMIS: ------------------------
2025-03-07 15:14:37,318 - __main__ - INFO - id: 27341892
2025-03-07 15:14:37,318 - __main__ - INFO - event_type: http_api_update_success
2025-03-07 15:14:37,318 - __main__ - INFO - event_time: 2025-03-07 15:14:37.292
2025-03-07 15:14:37,318 - __main__ - INFO - obj_class: smf_sm_context
2025-03-07 15:14:37,318 - __main__ - INFO - obj_id: 21
2025-03-07 15:14:37,318 - __main__ - INFO - add_text: {"id": 21,"supi": "001010000049447","pdn_id": 1,"ue_ipv4": "10.17.201.250","ue_ipv6_prefix": "",
2025-03-07 15:14:37,318 - __main__ - INFO - event_info_char_1:
2025-03-07 15:14:37,318 - __main__ - INFO - event_info_char_2:
2025-03-07 15:14:37,318 - __main__ - INFO - event_info_char_3:
2025-03-07 15:14:37,318 - __main__ - INFO - Processing event...
```

Figure 32: Events received by DE of NBC

4 Key performance indicators

4.1 Service KPIs

Table 3 summarizes the measured KPIs for user story 1.2 “Vehicular condition monitoring with security guarantees” in this initial testing phase of the project.

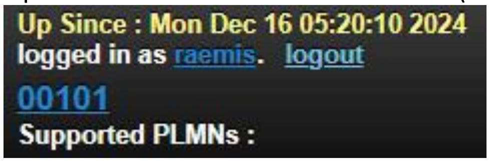
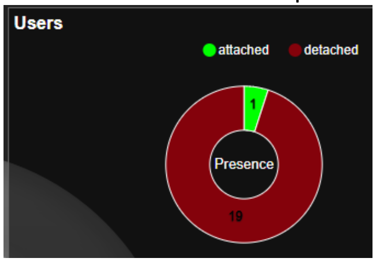
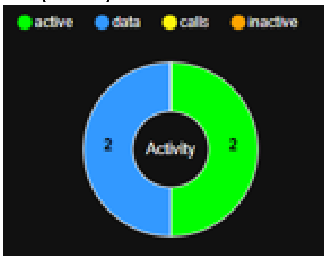
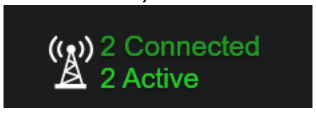
KPI	Definition	Unit	Relevant SUCCESS-6G enabler	Achieved value
Number and frequency of failures	Number and occurrence of the anomalies in the vehicular equipment operation.	Absolute number	C-V2X OBU	In the measurements retrieved from the OBU, no anomalies were detected.
Downtime	Defined as the time duration during which the vehicular component’s working conditions are affected by a failure which results in a faulty operation. - Failures tested: Over CPU consumption, Temperature triggering protection Action in OBU : reboot automatically Boot Time = 30s - 40s Service application run Time = 50ms Networking Establishment with gNB is in boot Time.	Time	C-V2X OBU	30 – 40 s
Reliability	The reliability rate for C-V2X message delivery is typically 99%, ensuring that critical messages are successfully	%	C-V2X OBU	100% during test

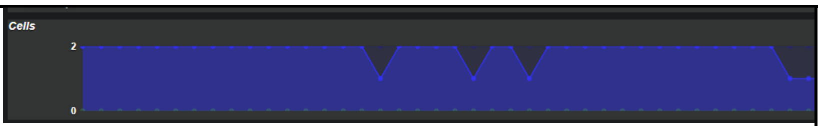
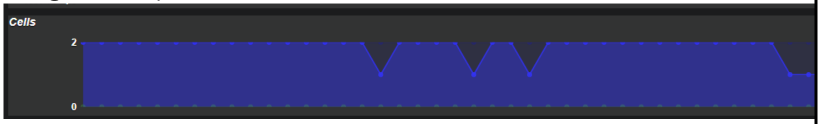
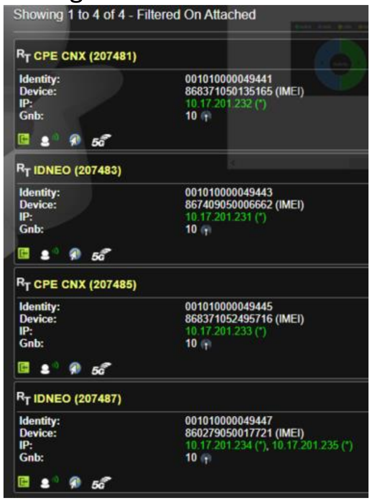
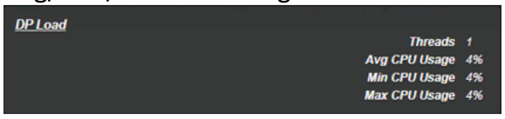
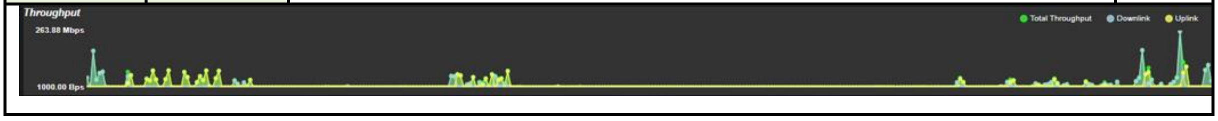
	transmitted.			
Coverage	C-V2X systems aim to achieve network coverage over 95% of the designated area, providing reliable communication across a wide range.	%	C-V2X OBU	100 assuming LOS
Velocity	Speed and direction of motion of a vehicle	Km/h	C-V2X OBU	134

Table 3: Service KPIs for user story 1.2

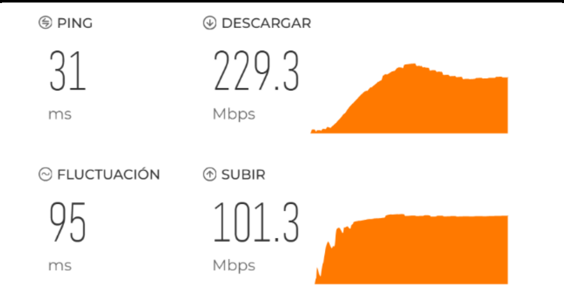
4.2 5G network

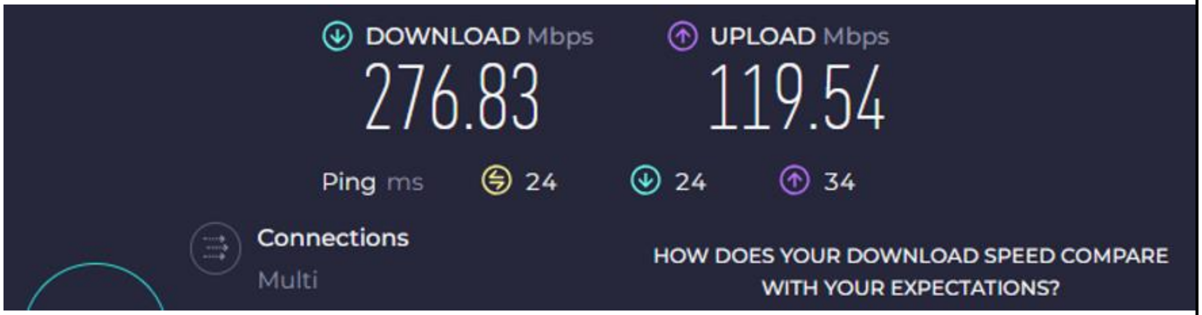
4.2.1 Core

KPI	**Category**	**Definition**	**Unit**
Number of seconds this system has been running	SYSTEM STATUS	Up Since : Mon Dec 16 05:20:10 2024 (from last restart)  Up Since : Mon Dec 16 05:20:10 2024 logged in as raemis. logout 00101 Supported PLMNs :	Hours, minutes, seconds
max_attached users permitted	SYSTEM STATUS	The max attached users permitted (SIMs) at the same time is 20  	Number
Max attached radios permitted	SYSTEM STATUS	The max attached radios permitted at the same time is 2 (Node 1 and Node 8)  2 Connected 2 Active	Number
Number of attached Radios	RADIOS	The number of attached Radios (node 1 and node 8) is 2 (See image above).	Number

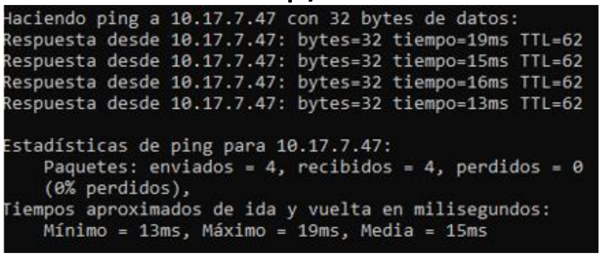
			
Number of active radios (more than 1 user attached)	RADIOS	Number of active radios when Node 1 and Node 8 are up is 2 (See image above). 	Number
Number of attached Users	USERS	During the tests the number of attached users is 4/20:  Showing 1 to 4 of 4 - Filtered On Attached RT CPE CNX (207481) Identity: 001010000049441 Device: 868371050135165 (IMEI) IP: 10.17.201.232 (*) Gnb: 10 RT IDNEO (207483) Identity: 001010000049443 Device: 867409050006662 (IMEI) IP: 10.17.201.231 (*) Gnb: 10 RT CPE CNX (207485) Identity: 001010000049445 Device: 868371052495716 (IMEI) IP: 10.17.201.233 (*) Gnb: 10 RT IDNEO (207487) Identity: 001010000049447 Device: 860279050017721 (IMEI) IP: 10.17.201.234 (*), 10.17.201.235 (*) Gnb: 10	Number
Number of Active Users (not idle mode)	USERS	During the tests the number of Active users is 4/20 (See image above).	Number
average CPU usage for PS	CPU USAGE	Avg/Min/Max CPU Usage is 4%  DP Load Threads 1 Avg CPU Usage 4% Min CPU Usage 4% Max CPU Usage 4%	%
Current UL bits per second on S1-U/N3	THROUGHPUT	UL: 72.08 (Peak)	Mbps
Current DL bits per second on S1-U/N3	THROUGHPUT	DL: 263.88 (Peak)	Mbps
			

Current UL bits per second on Sgi/N6 (Internet)	THROUGHPUT	UL: 101.3-119.54 (Avg)	Mbps
Current DL bits per second on Sgi/N6 (Internet)	THROUGHPUT	DL: 229.3-276.83 (Avg)	Mbps





Network Latency	LATENCY	From the device to the core network: • Interfaces N1 and N2: The latency from the device (UE) to the core network involves communication through the N1 interface (UE to AMF) and the N2 interface (RAN to AMF) for signaling and control. • **~15ms round-trip / 2 = 7ms.**	ms

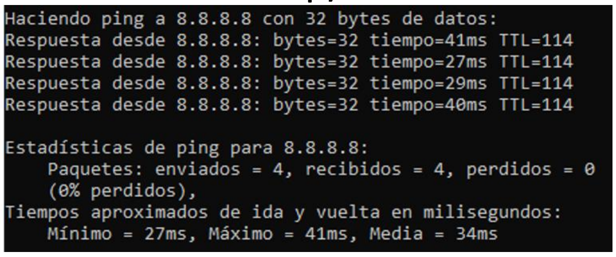


```
Haciendo ping a 10.17.7.47 con 32 bytes de datos:
Respuesta desde 10.17.7.47: bytes=32 tiempo=19ms TTL=62
Respuesta desde 10.17.7.47: bytes=32 tiempo=15ms TTL=62
Respuesta desde 10.17.7.47: bytes=32 tiempo=16ms TTL=62
Respuesta desde 10.17.7.47: bytes=32 tiempo=13ms TTL=62

Estadísticas de ping para 10.17.7.47:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 13ms, Máximo = 19ms, Media = 15ms
```

From the device to the internet:

- Interfaces N3 and N6: The latency from the device to the internet includes communication through the N3 interface (RAN to UPF) and the N6 interface (UPF to external networks like the internet).

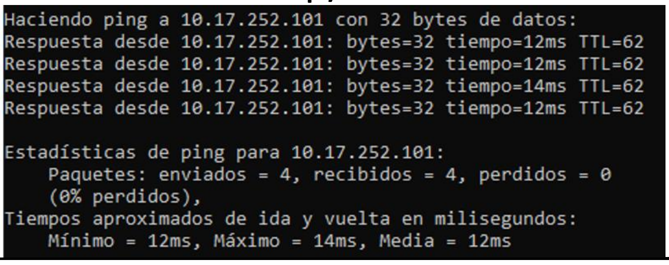
- **~34ms. Round-trip / 2 = 17ms.**



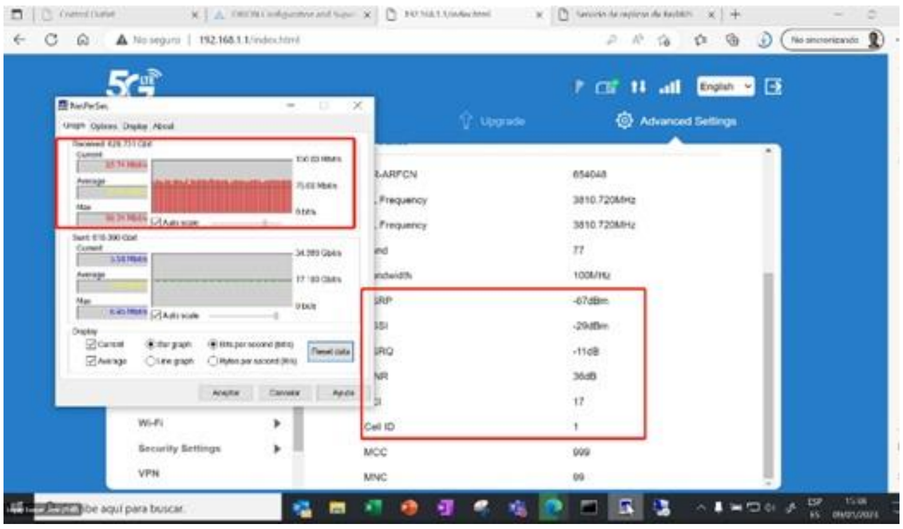
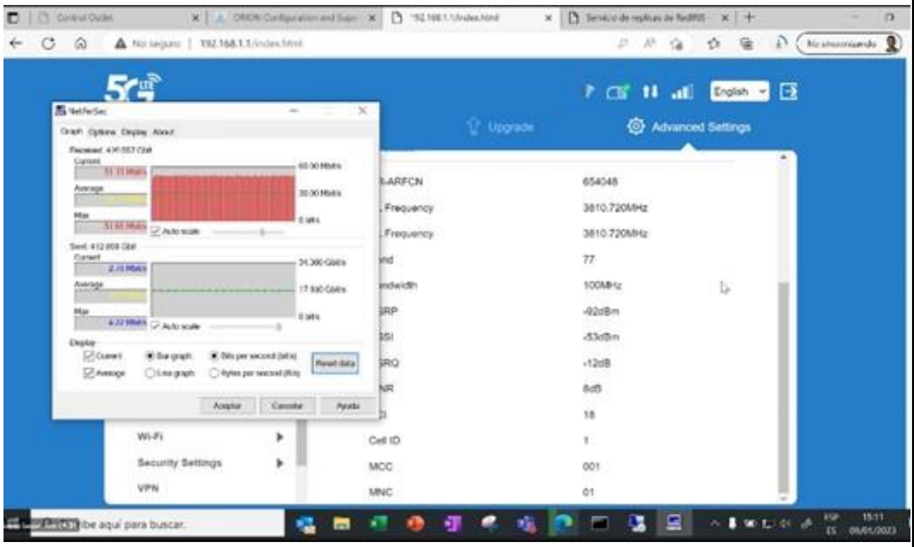
```
Haciendo ping a 8.8.8.8 con 32 bytes de datos:
Respuesta desde 8.8.8.8: bytes=32 tiempo=41ms TTL=114
Respuesta desde 8.8.8.8: bytes=32 tiempo=27ms TTL=114
Respuesta desde 8.8.8.8: bytes=32 tiempo=29ms TTL=114
Respuesta desde 8.8.8.8: bytes=32 tiempo=40ms TTL=114

Estadísticas de ping para 8.8.8.8:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 27ms, Máximo = 41ms, Media = 34ms
```

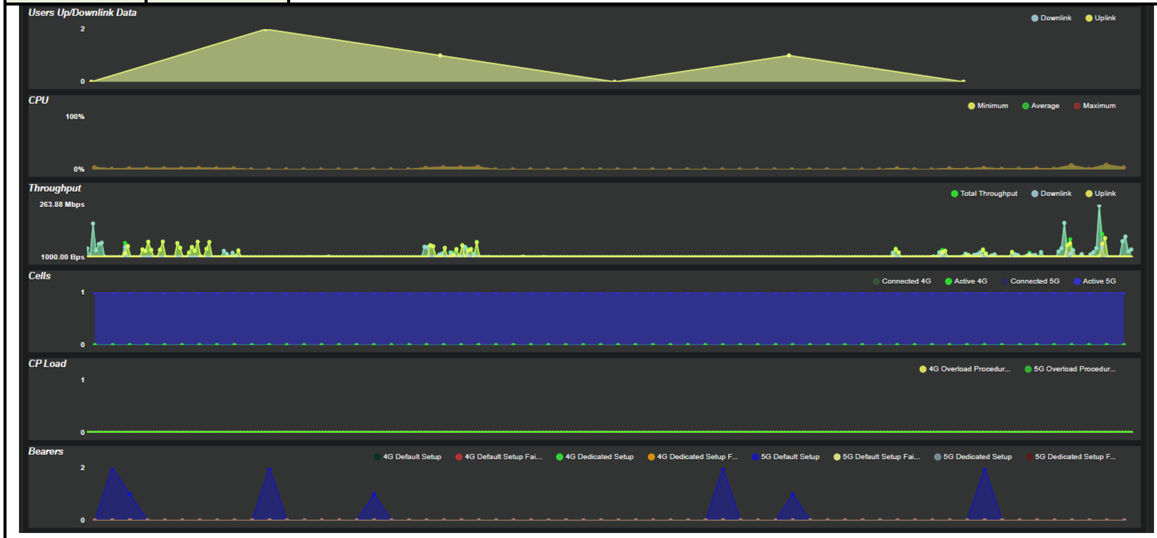
From the device to an edge service:

- Interfaces N3 and N6: The latency from the device to an edge service involves communication through the N3 interface (RAN to UPF) and the N6 interface (UPF to external networks). In this scenario, the edge service is typically located close to the UPF to minimize latency.
- **~12ms. Round-trip / 2 = 6 ms.**



```
Haciendo ping a 10.17.252.101 con 32 bytes de datos:
Respuesta desde 10.17.252.101: bytes=32 tiempo=12ms TTL=62
Respuesta desde 10.17.252.101: bytes=32 tiempo=12ms TTL=62
Respuesta desde 10.17.252.101: bytes=32 tiempo=14ms TTL=62
Respuesta desde 10.17.252.101: bytes=32 tiempo=12ms TTL=62

Estadísticas de ping para 10.17.252.101:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 12ms, Máximo = 14ms, Media = 12ms
```

FTP Throughput	THROUGHPUT	Node 1 : FTP DL 93.74 Mbit/s  ARFCN 654048 Frequency 3810.720MHz Frequency 3810.720MHz nd 77 ndwidth 100MHz RP -67dBm SI -29dBm RQ -11dB NR 36dB I 17 Cell ID 1 MCC 999 MNC 99 Node 8: FTP DL 51.93 Mbit/s  ARFCN 654048 Frequency 3810.720MHz Frequency 3810.720MHz nd 77 ndwidth 100MHz RP -92dBm SI -53dBm RQ -12dB NR 8dB I 18 Cell ID 1 MCC 001 MNC 01



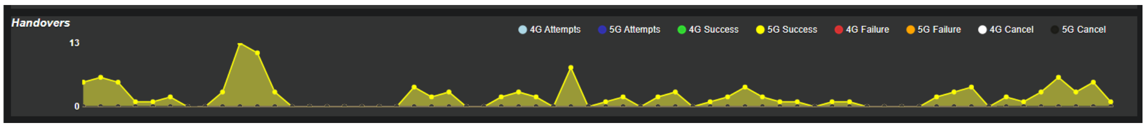


Table 4: Network KPIs that can be extracted from the core.

4.2.2 Final user (On-Board Unit)

KPI	Definition	Unit	Achieved value
Uptime	Measures the amount of time the network is available and functioning properly. High uptime is an indicator of a stable and reliable network.	%	100% during Testing
Latency	The time it takes for a data packet to travel from its source to its destination. Low latency is crucial for ensuring fast and smooth communication on the network.	ms	Node 1 12.617/ 17.703/ 31.385 Min/ Avg/ Max
Bandwidth	Measures the amount of data that can be transmitted through the network in a given period of time. Adequate bandwidth is essential for supporting traffic load and avoiding bottlenecks.	Hz	5- 100 Mhz Estimation: 25- 30 Mhz
DL (downlink) throughput **- Very good radio conditions** **- Good radio conditions** **- Medium radio conditions**	Indicates the speed at which data can be downloaded across the network. High throughput is important for efficient communication and a smooth user experience.	Mbps	Node 8 NO DATA Node 1 -VERY GOOD (RSRP: -82dbm, RSRQ: -11 db, SNR: 25db) 165 Mbps -GOOD (RSRP: -86dbm, RSRQ: -11 db, SNR: 25db) 162 Mbps -MEDIUM (RSRP: -100dbm, RSRQ: -11 db, SNR: 21db) 162 Mbps

UL (uplink) throughput **- Very good radio conditions** **- Good radio conditions** **- Medium radio conditions**	Indicates the speed at which data can be sent across the network. High throughput is important for efficient communication and a smooth user experience.	Mbps	Node 8 NO DATA Node 1 -VERY GOOD (RSRP: -81dbm, RSRQ: -11 db, SNR: 25db) 59.6 Mbps -GOOD (RSRP: -87dbm, RSRQ: -11 db, SNR: 25db) 49.5 Mbps -MEDIUM (RSRP: -103dbm, RSRQ: -11 db, SNR: 21db) 51.5 Mbps
Reliability	Measures the likelihood that the network operates without errors or interruptions. A reliable network minimizes downtime and ensures constant connectivity.	%	100%
Communication range	Communication range is the maximum distance between a transmitter and its intended receiver allowing communication with a targeted packet size, latency, and reliability, and for a given effective transmit power and receiver sensitivity.	Meters/ Kilometres	> 400m
RSRQ	Reference Signal Received Quality: Quality considering also RSSI and the number of used Resource Blocks (N) RSRQ = (N * RSRP) / RSSI measured over the same bandwidth. RSRQ is a C/I type of measurement, and it indicates the quality of the received reference signal. The RSRQ measurement provides additional information when RSRP is not sufficient to make a reliable handover or cell reselection decision.	dB	-14 <= RSRQ <= -11
RSRP	Reference Signal Received Power: RSRP is a RSSI type of measurement, as follows there are some definitions of it and some details as well. It is the power of the LTE Reference Signals spread over the full	dB	-121 <= RSRP <= -81

	bandwidth and narrowband. A minimum of -20 dB SINR (of the S-Synch channel) is needed to detect RSRP/RSRQ		
SNR	Compares the level of a desired signal to the level of background noise	dB	13 <= SNR <= 25

Table 5: Network KPIs that can be extracted from the final user (e2e)

5 Conclusions

In the preliminary tests carried out over two days in Castellolí, certain service KPIs of the user story 1.2 and the 5G network were validated, and the integration of the different software components on the two SUCCESS-6G testbeds was completed. A third visit to the Castellolí Circuit facilities has been planned to carry out the remaining necessary tests for the acquisition of the complete list of service KPIs and to refine the service migration during the network node handover.

6 Annex A

6.1 Photos from the Castelloli trials on 19/12/2024

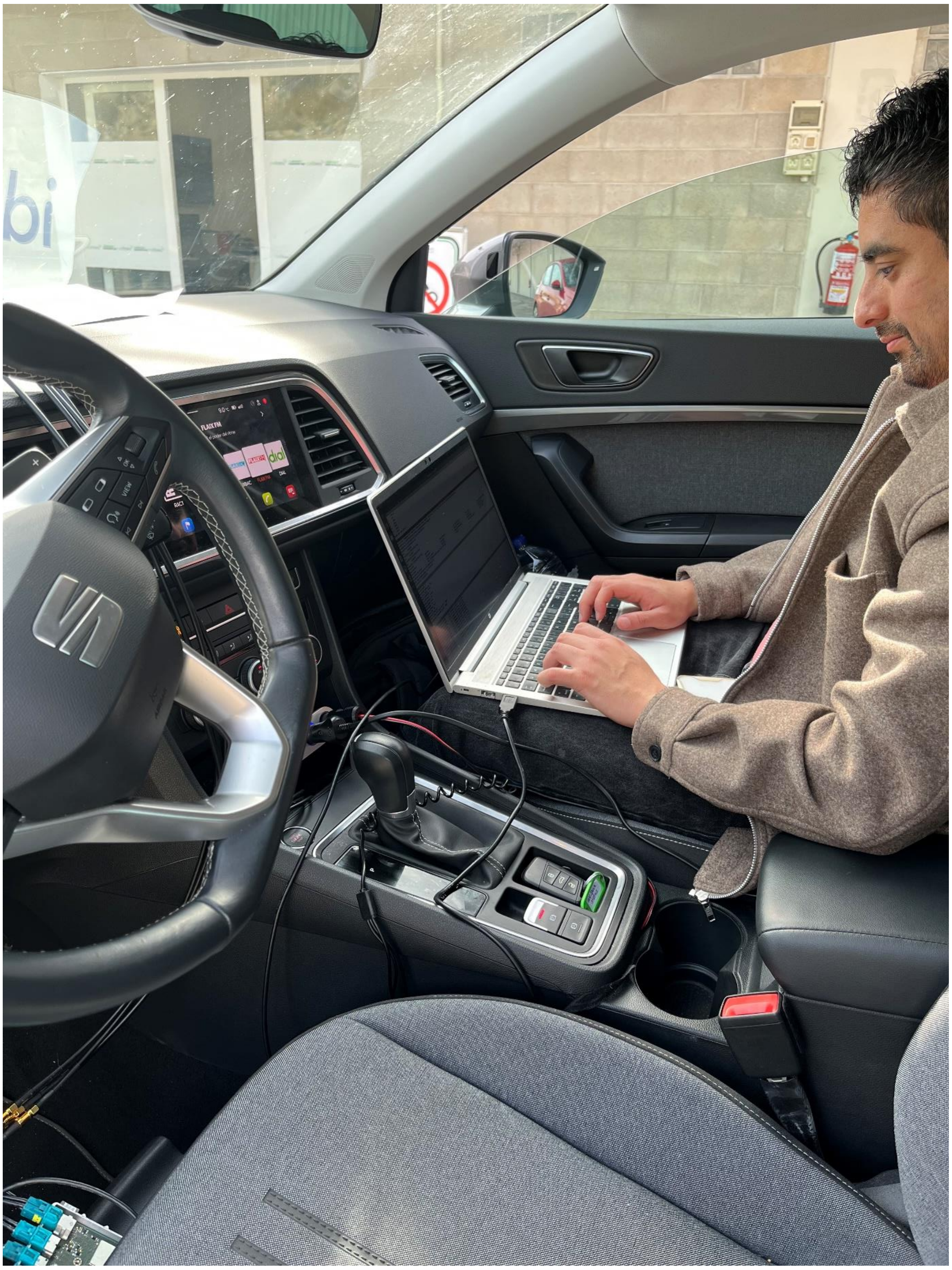
3109 MCZ
Ateca

idneo

3109 MCZ

6.2 Photos from the Castelloli trials on 24/02/2025

3109 MCZ
Ateca



References

[1] P. Mulinka et al., "Information Processing and Data Visualization in Networked Industrial Systems," 2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Helsinki, Finland, 2021, pp. 1-6, doi: 10.1109/PIMRC50174.2021.9569603

[2] Deliverable E5: https://success-6g-project.cttc.es/images/deliverables/E5-DEVISE-2.pdf

[3] Deliverable E6: https://success-6g-project.cttc.es/images/deliverables/E6.%20Diseno%20de%20la%20arquitectura%20del%20sistema-DEVISE.pdf