REPORT ON IMPLEMENTATION OF THE WIRELESS EDGE CONTROL ROBOTICS USE-CASE

Deliverable D2.3

The TARGET-X project has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No: 101096614

REPORT ON IMPLEMENTATION OF THE WIRELESS EDGE CONTROL ROBOTICS USE-CASE

GRANT AGREEMENT	101096614
PROJECT TITLE	Trial PlAtform foR 5G EvoluTion – Cross-Industry On Large Scale
PROJECT ACRONYM	TARGET-X
PROJECT WEBSITE	www.target-x.eu
PROJECT IDENTIFIER	https://doi.org/10.3030/101096614
PROGRAMME	HORIZON-JU-SNS-2022-STREAM-D-01-01 — SNS Large Scale Trials and Pilots (LST&Ps) with Verticals
PROJECT START	01-01-2023
DURATION	30 Months
DELIVERABLE TYPE	Deliverable
CONTRIBUTING WORK PACKAGES	2, 3, 6
DISSEMINATION LEVEL	Public
DUE DATE	M22
ACTUAL SUBMISSION DATE	2024-10-31
RESPONSIBLE ORGANIZATION	RWTH-WZL
EDITOR(S)	Lucas Manassés Pinheiro de Souza (RWTH-WZL)
VERSION	1.0
STATUS:	Final
SHORT ABSTRACT	This document outlines the implementation process for the edge-controlled mobile manipulation use case in WP2. It details the deployment of a mobile manipulator integrated with 5G connectivity at the WZL Testbed, focusing on real-time processing, localization, and motion planning. The report highlights key components, including the ROS 2-based software stack, NVIDIA Isaac-Sim for simulation, and the use of 5G NR for positioning.
KEY WORDS	Implementation, edge robotics, 5G, real-time, mobile manipulator, simulation, ROS 2, motion planning, localization, digital twin.

CONTRIBUTOR(S)	Lucas Manassés Pinheiro de Souza (RWTH-WZL) Manuel Fuentes, Carlos Ravelo, Alberto Alonso (5CMM)

Disclaimer

Co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the other granting authorities. Neither the European Union nor the granting authority can be held responsible for them.

Executive Summary

This deliverable, D2.3, documents the successful implementation of the "Edge-Controlled Automation with Mobile Manipulation" use case as part of Work Package 2 (WP2) within the TARGET-X project. Building on the findings from previous deliverables, specifically D2.2, the focus of this report is on the practical deployment and integration of mobile manipulators with 5G-enabled edge computing technologies. The key objectives include enhancing real-time responsiveness, scalability, and dynamic capabilities of mobile robots in industrial environments, particularly in line-less mobile assembly systems. The report provides a comprehensive overview of the system architecture and implementation at the WZL Testbed, highlighting the hardware components, such as mobile manipulators and LiDAR sensors, and the software stack based on ROS 2. Critical to the success of the deployment is the use of advanced tools like NVIDIA's Isaac-Sim for simulation, real-time localization using Cartographer SLAM, motion planning via MoveIt2, and 5G NR for positioning. The deliverable also explores the challenges faced during the implementation, such as managing high-bandwidth data flows from sensors and ensuring low-latency communication between robots and edge servers. To address these, technologies such as containerization (Docker), CI/CD pipelines, and middleware like DDS and Zenoh were integrated to facilitate efficient and reliable communication between the mobile manipulator and the edge server. In terms of the next steps, the validation and benchmarking of network requirements will be further explored and presented in Deliverable D2.5. This will focus on assessing the performance of the network under different conditions and enhancing the communication infrastructure for future deployments.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

3GPP	3rd Generation Partnership Project
AGV	Automated Guided Vehicle
AI/ML	Artificial Intelligence/Machine Learning
CI/CD	Continuous Integration and Continuous Deployment
Connext DDS	Connext Data Distribution Service
CycloneDDS	Cyclone Data Distribution Service
DDS	Data Distribution Service
DOF	Degrees of Freedom
E2E	End-to-End
Fast RTPS	Fast Real-Time Publish-Subscribe
FR	Frequency Range
GPS	Global Positioning System
IMU	Inertial Measurement Unit
JCAS	Joint Communication and Sensing
LMAS	Line-less Mobile Assembly Systems
LiDAR	Light Detection and Ranging
MPC	Model Predictive Control
NR	New Radio
NSA	Non-Standalone
NW	Network
OMPL	Open Motion Planning Library
PRM	Probabilistic Roadmap
QoS	Quality of Service
QP	Quadratic Programming
RAN	Radio Access Network
RF	Radio Frequency
RRT	Rapidly Exploring Random Tree
ROS	Robot Operating System
ROS1	Robot Operating System 1

ROS2	Robot Operating System 2
RMW	ROS 2 Middleware
RWTH	Rheinisch-Westfälische Technische Hochschule (German)
TSN	Time-Sensitive Networking
TTFF	Time To First Fix
UE	User Equipment
URLLC	Ultra-Reliable and Low-Latency Communication
WZL	Werkzeugmaschinenlabor der RWTH Aachen (German)
WZL	Laboratory for Machine Tools and Production Engineering of RWTH Aachen University
WP1	Work Package 1
WP2	Work Package 2
WP6	Work Package 6
5G-ICE	5G-Industry Campus Europe

1 Introduction

The manufacturing industry continues to face significant challenges, including a shortage of skilled workers, market volatility, and increasing demands for mass customization [1]. These factors have driven the need for flexible automation solutions capable of adapting to evolving production requirements. Traditional manufacturing paradigms, characterized by rigid and fixed assembly lines, are proving inadequate in addressing these modern challenges [1]. To meet this demand, *Lineless Mobile Assembly Systems* (LMAS) have emerged as a promising solution, leveraging mobile robotics to enhance flexibility and adaptability in production environments. LMAS offers the ability to continuously optimize production by dynamically allocating jobs, goods, resources, and work locations within the factory, thereby allowing the movement of either robots to resources or resources to robots [1].

As introduced in *Deliverable 2.2 (D2.2)* [2], the shift from traditional assembly lines to assembly matrices requires deploying flexible assembly resources, with mobile robots, particularly mobile manipulators, being key in LMAS scenarios. These systems must be adaptable, addressing challenges such as the adaptive control of mobile robots, process management, and path planning [3]. In LMAS, resources move either actively or passively, with actively moving robots picking up passive resources, highlighting the need for advanced robotic perception [4]. AI-based vision algorithms, as shown in recent studies [5] and [6], enable real-time object detection, allowing robots to autonomously adjust and execute tasks seamlessly across various scenarios.

In this context, 5G (and future 6G) communication technologies are critical for LMAS, enabling smooth interaction between mobile robots and control systems via edge cloud servers, ensuring dynamic adaptation to production demands. Figure 1.1 illustrates the shift from current challenges in assembly systems to flexible assembly scenarios that leverage edge computing to enhance the autonomy of mobile robots, where autonomy refers to a robot's capability to perform tasks and make decisions independently of human intervention.

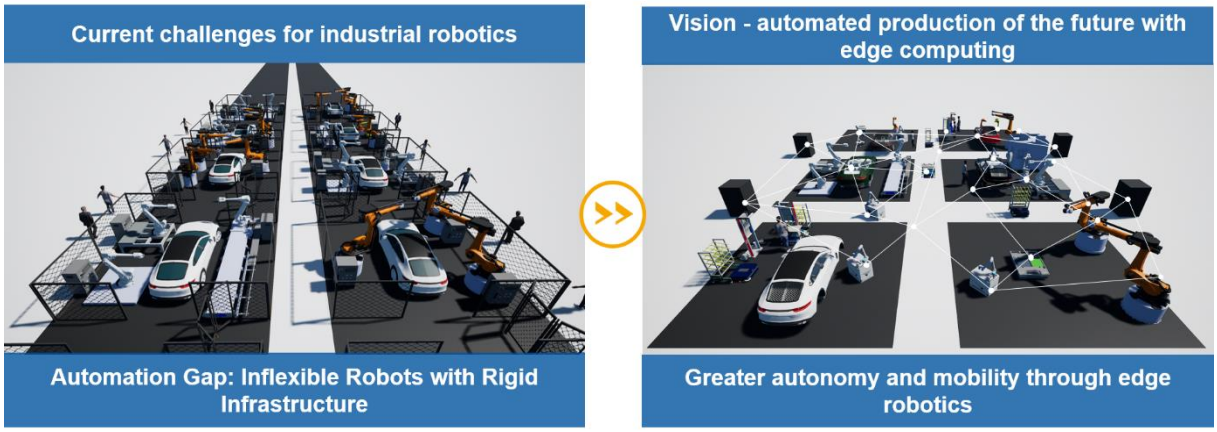

Figure 1.1: Illustration of the transition towards flexible assembly systems with edge computing capabilities. Each assembly station is an element of the assembly matrix. Source: adapted from [7].

In *Deliverable 2.2 (D2.2)*, the exploration of potential system designs and implementation options for edge-controlled mobile manipulators within LMAS was extensively covered. The report highlighted the technological challenges associated with adaptive control, process management, and path planning for mobile robots in assembly systems. Additionally, the adoption of 5G communication

technologies was emphasized as an enabler for seamless interaction between mobile robotics and edge computing systems, facilitating the real-time adaptation required in modern manufacturing [2, 8].

Building on the findings and analyses from D2.2, the present document report shifts its focus from exploration to practical implementation. The aim is to document the concrete steps taken to deploy the *Edge-Controlled Automation with Mobile Manipulation* use case at the *Werkzeugmaschinenlabor* (German) or Laboratory for Machine Tools and Production Engineering (WZL) of the *Rheinisch-Westfälische Technische Hochschule* (RWTH) Aachen University [9].

This implementation is a pivotal milestone within the TARGET-X Work Package 2 (WP2) – Manufacturing, where the mobile manipulators are integrated with 5G-enabled edge computing to achieve real-time responsiveness and dynamic assembly capabilities.

The implementation focuses on addressing several key elements:

- The practical deployment of mobile manipulators in flexible assembly systems to meet real-world assembly challenges.
- Integration of 5G communication technologies to facilitate efficient data processing at the edge.
- Overcoming network-related limitations such as uplink capacity and ensuring low-latency communication for high-bandwidth sensor data, particularly from AI-driven vision systems and LiDAR.

This report will delve into the specific technical solutions, challenges encountered during deployment, and the performance evaluation of the system. By focusing on these aspects, it demonstrates how the theoretical framework established in D2.2 is materializing into a functional and scalable system at the WZL testbed, furthering the project's goals of creating a flexible, future-proof manufacturing environment.

Within the scope of the TARGET-X project, Deliverable 2.3 focuses on the implementation of the intended use case, building upon the system design and options explored in D2.2. This document outlines the concrete steps taken to deploy the use case, including the integration of mobile robotics with 5G technologies, and evaluates the performance of the system against established requirements for various applications and communication streams. The terminology used in this deliverable adheres to the common terminology defined by the TARGET-X project, available as a public contribution on the project website [10].

1.1 Objectives of the document

This document outlines the implementation of the "Edge-Controlled Automation with Mobile Manipulation" use case, advancing from the system design options explored in previous deliverables. The primary objective is to provide the hardware and software components required for initial trials at the WZL Testbed, enabling validation of mobile robotics using 5G connectivity features. The report focuses on real-time data aggregation, sensor integration, and the middleware development to support edge-based control of mobile manipulator. The implementation aims to enhance robot localization, path planning, and object recognition while leveraging 5G features to improve overall system performance, responsiveness, and safety in flexible assembly scenarios of manufacturing environments.

1.2 Structure of the document

The document is organized into five main sections. Section 1 introduces the objectives and structure of the deliverable, as well as its relation to other activities within the project. Section 2 provides an overview of the edge-controlled automation system, detailing the intended use case, changes made, and updated requirements for 5G integration. It also includes a description of the use case, its challenges, and the integration of positioning with 5G NR. Section 3 covers the system architecture and implementation at the WZL testbed, focusing on both hardware and software components, along with the integration of 5G features, including real-time positioning. Additionally, the section presents the communication protocols and interfaces, emphasizing how different components interact within the system. Finally, Section 4 concludes the document, summarizing the key findings and outlining the next steps for future work.

1.3 Relation to other activities

The content provided in this deliverable is closely related to other activities carried out in other WPs of the TARGET-X project. In particular, the initiatives related to establishing 5G network requirements and aligning use case requirements across TARGET-X verticals, led by WP1 (Methodological assessment framework) and WP6 (Technology evolution beyond 5G) as outlined in deliverable D1.1 and D2.2. These were essential in shaping the 5G setup of the deliverable D2.1 and D2.2 and the present document (D2.3) within WP2 (Manufacturing). Additionally, activities pertaining to the examination of the energy monitoring in manufacturing use cases from WP3 (Energy) has strong dependence on the development of the WP2 use cases, one example of this was the joint use case Demo *Meter-X-based process analytics, comparing the power profile of a process with different trajectories and different weights,* presented at the European Conference on Networks and Communications (EuCNC) & 6G Summit 2024 [11] and in the TARGET-X Open Day Event. The Demo integration is further detailed in deliverables D3.1, D3.2, and D3.4. Moreover, collaborative efforts between WP2 and WP6 resulted in the identification and definition of beyond 5G technologies for the specific use case outlined in WP2. These advancements are detailed in deliverables D6.1 and D6.3.

2 Edge-Controlled Automation with Mobile Manipulation

This section describes the use case initially presented in Deliverable D2.2, focusing on the updated requirements and system design. It highlights the rationale and impact of changes made since the previous deliverable, justifying key decisions. The chapter also details the current use case setup, including 5G requirements and how 5G New Radio (NR) features are expected to enhance positioning for mobile manipulation at WZL Testbed.

2.1 Background

Mobile robots such as AGVs and mobile manipulators are key to flexible assembly systems, offering significant versatility [3]. These robots navigate and interact with their environment autonomously, equipped with AI-driven cognitive capabilities and sensor technologies such as LiDAR. They require substantial computational power for tasks such as image recognition, object detection, motion planning, and LiDAR data processing, which onboard systems cannot fully support. Consequently, there is increasing adoption of edge computing to offload these complex tasks to nearby edge devices, enhancing performance and real-time decision-making in dynamic manufacturing environments [8].

However, a robust network infrastructure is expected for the effective deployment of edge-controlled mobile manipulators for flexible assembly. These systems rely heavily on real-time data processing and decision-making to navigate and perform complex tasks autonomously. The volume and complexity of sensor data, including high-resolution images and LiDAR point clouds, necessitate rapid and reliable transmission to the edge computing systems for processing. This uplink (from robot to edge) capacity is a bottleneck in traditional wireless networks, impacting the responsiveness and accuracy of mobile manipulators [12–14]. Real-time tasks like object detection and pose estimation need fast feedback between the robot and edge system to adjust actions dynamically. Delays can lead to errors, inefficiencies, and potential hazards on the production floor. Thus, low-latency, high-bandwidth communication is crucial for maintaining the precision and reliability of these robots.

While advanced wireless technologies such as 5G offer lower latency, higher bandwidth, and greater reliability, they still fall short in fully addressing the uplink challenge, especially with the massive data volumes from LiDAR sensors. The volume and speed of data generated require even more robust solutions. The uplink demands for real-time processing of sensor and camera data remain a significant hurdle, necessitating continuous innovation in network technologies to meet the demanded requirements of modern assembly systems in manufacturing environments.

These requirements and uplink challenges will be further discussed in Section 2.5.

2.2 Overview of Intended Use Case (from D2.2)

The initial use case for edge-controlled automation with mobile manipulation, as detailed in *Deliverable D2.2*, focused on leveraging 5G connectivity to enable flexible, real-time control of mobile robots in a manufacturing environment. This use case was centered around the integration of mobile manipulators within flexible assembly systems, emphasizing the need for high adaptability, real-time processing, and dynamic task allocation within industrial settings. Key technological requirements identified included adaptive control strategies, sensor integration for object detection, and path planning to optimize motion control, as illustrated in Figure 2.1.

Pick and place are a fundamental assembly process in many industrial settings, involving selecting an object from one location and placing it in another. This repetitive and precision-required task is ideal for automation, which can significantly enhance productivity, consistency, and efficiency while reducing costs and freeing human workers for more complex tasks. Mobile manipulators advance this automation by combining mobility with manipulation, enabling them to navigate a given industrial environment and perform the required task. Key technologies enabling the automation of pick and place include machine vision for object detection, advanced motion planning algorithms for navigation and manipulation, simulation pipelines, and real-time communication for outsourcing computational power.

The system design included four proposed design options, each addressing different technological aspects such as motion planning, real-time communication, and object recognition. The introduced use case architecture in D2.2 aimed to offload computationally intensive tasks, such as motion planning and perception, to edge computing systems (local servers). Furthermore, the expected network requirements for integrating 5G technologies, particularly in handling uplink data from sensors like LiDAR and cameras, were presented. Performance parameters, including latency, bandwidth, and reliability, were also identified to ensure seamless robot-to-edge communication streams. Details regarding the previously defined network requirements from the application side can be found in *D2.2*, Section 3.2. The updated requirements will be further discussed in Section 2.5 of this document.

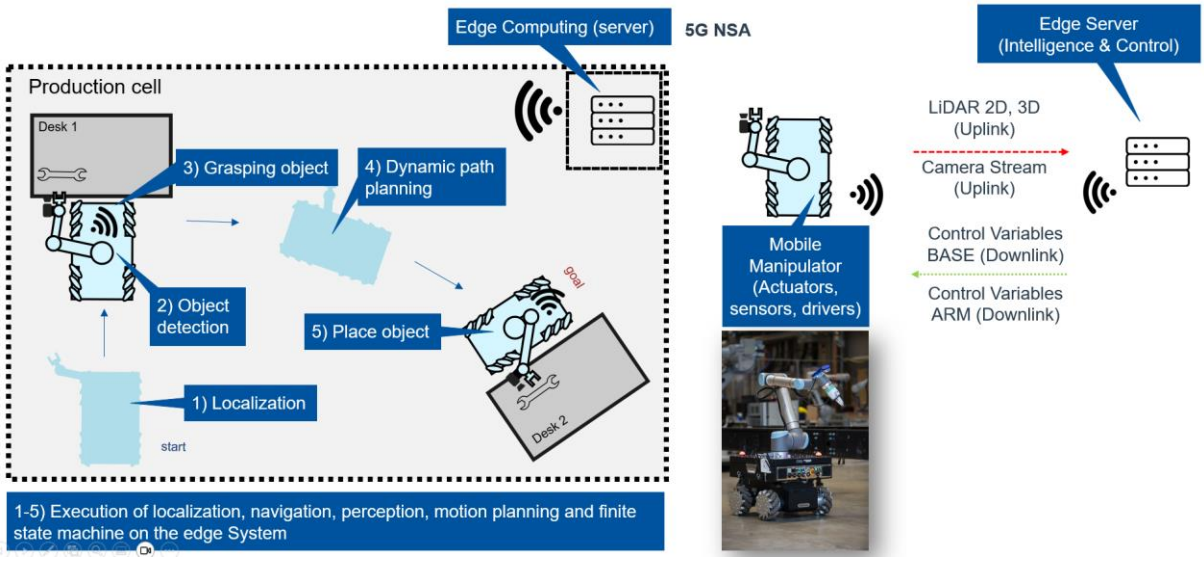

Figure 2.1: Execution of localization, navigation, perception, and motion planning with edge computing on the local server. Source: provided by WZL [9].

2.3 Summary of Changes Made

In *Deliverable D2.2*, four design options were presented for the system architecture of the edge-controlled mobile manipulation use case. Initially, **Design Option 3** was chosen due to its holistic reactive motion control approach, which offered greater agility and adaptability in dynamic environments. However, during the implementation phase, technical challenges arose, prompting a reevaluation. As a result, we have now opted for **Design Option 1**, which is more aligned with current industry practices and provides a simplified yet reliable sequential motion planning

approach. The decision to shift to Option 1 was driven by the practicalities of ensuring seamless integration into the existing industrial environment.

2.3.1 Rationale for Changes

In *D2.2*, four potential architectures were presented for the system design of mobile manipulators:

- **Design Option 1: Sequential Motion Planning** – This approach separates motion planning for the mobile base and manipulator, simplifying the overall system design. It avoids simultaneous movement, reducing complexity and making it more suitable for industrial applications.
- **Design Option 2: Whole-Body Motion Sampling-based Planning** – This architecture plans the entire robot's movement in a holistic manner, optimizing paths for both the base and the manipulator. While efficient, it adds complexity and demands higher computational power.
- **Design Option 3: Holistic Reactive Motion Control** – This approach integrates planning and reactivity, allowing the robot to respond dynamically to its environment without requiring explicit planning for each task. It was initially chosen for its potential to improve efficiency in dynamic, real-time scenarios.
- **Design Option 4: Whole-Body Motion Hybrid Planning with Collision Avoidance** – The most complex option, combining motion planning with obstacle avoidance. This design prioritizes optimal, collision-free motion but introduces significant implementation complexity.

Although **Design Option 3** was initially selected due to its ability to handle dynamic environments, technical challenges emerged during the implementation phase. These included difficulties in achieving real-time responsiveness and the complexity of integrating advanced reactive control algorithms into the existing system infrastructure. Given these challenges and the industry's preference for more standardized solutions, we opted for **Design Option 1**, which offers a simpler and more reliable framework for sequential motion planning, making it easier to deploy and maintain.

2.3.2 Impact of Changes on System Design

The shift from **Design Option 3** to **Design Option 1** does not impact the overall network architecture or the core requirements of the use case. The edge-controlled architecture remains consistent across both designs, ensuring that the network's ability to handle sensor data transmission and control communication streams is unaffected. The primary change lies in the internal planning logic of the mobile manipulator, moving from a holistic reactive approach to a more sequential, step-by-step motion control. This change simplifies the system design without compromising the use case's broader network demands, such as latency, bandwidth, and real-time processing requirements.

2.4 Updated use case description

The pick-and-place use case remains the core focus of the system, retaining the same physical layout and task sequence introduced in *Deliverable D2.2*. However, the integration of 5G NR positioning introduces a significant enhancement to the localization process, which now precedes the pick-and-

place task. This update reduces reliance on LiDAR sensors for mobile robot localization, providing a cost-effective and efficient alternative for global positioning.

Updated Layout and 5G NR Positioning Integration

LiDAR sensors, while highly precise, are expensive and add significant hardware costs to the system. In this use case, 5G NR positioning is employed as a means to reduce process times, cut hardware costs, and enhance safety by enabling the robot to localize itself globally without the need for additional sensors. The goal is to utilize 5G fetures for global localization by M26, reducing the reliance on LiDAR for tasks where high precision is not strictly necessary.

Before initiating the pick-and-place sequence, the mobile robot will use 5G NR positioning to localize itself within the shop floor. This approach allows the robot to navigate the environment without depending solely on LiDAR, reserving the LiDAR sensor for more precise actions during the manipulation task, such as object detection and handling at the assembly station. By integrating 5G NR positioning with other sensor inputs as part of a sensor fusion solution, the robot can achieve high accuracy in path planning and benchmark its performance against the SLAM (Simultaneous Localization and Mapping) algorithm.

The new layout, as depicted in the updated image in Figure 2.2, illustrates the integration of the 5G NR positioning system within the factory's top-down view. The mobile manipulator will localize itself near the center of the shop floor using 5G NR before moving to the assembly station located at the bottom right corner of the workspace. This setup promotes real-time map creation, allowing the robot to continuously update its environment map as conditions change and share the map with other robots operating in the same space.

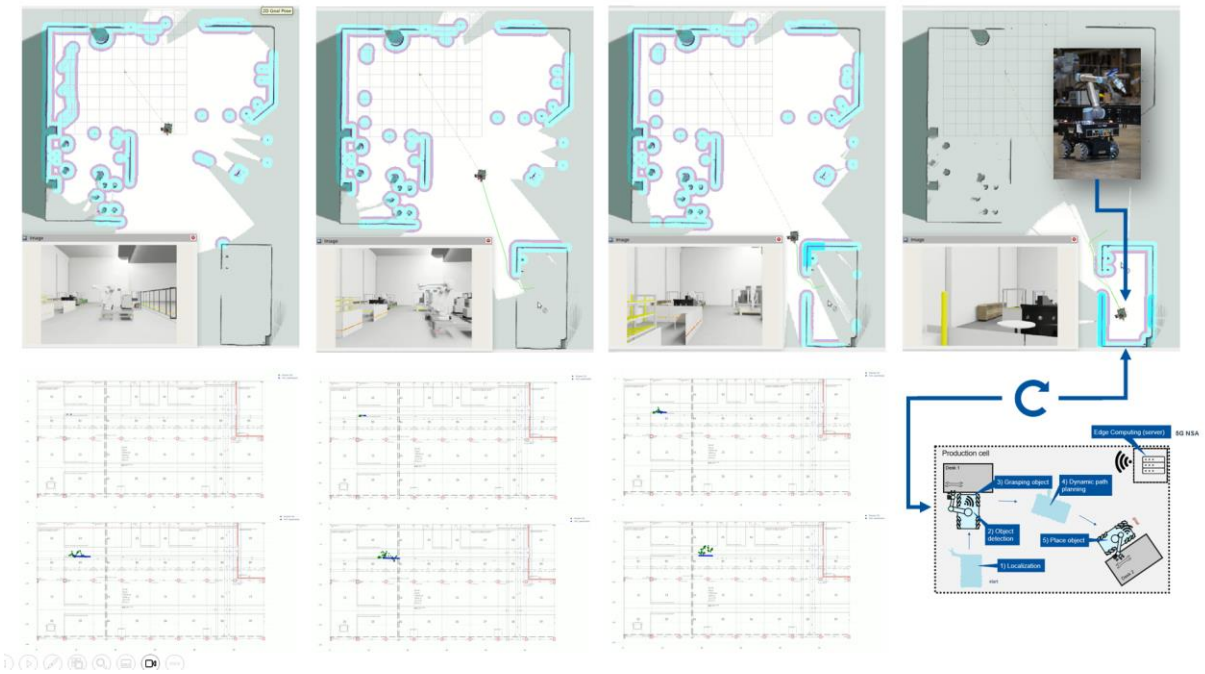

Figure 2.2: Overall layout of the use case, illustrating the robot's localization on the shop floor and the designated location for the pin-picking task. The bottom left section shows the benchmarking results of 5G positioning, displayed on the 5G dots map.

Task-Driven Use Case Description

The use case follows a task-driven approach, focusing on automating the pick-and-place task in a dynamic, edge-controlled environment. After localizationusing 5G NR positioning, the mobile manipulator will be sent to the designated assembly station to begin the pick-and-place sequence, as outlined in *Deliverable D2.2* and illustrated in Figure 2.1 of Section 2.2.

Upon reaching the assembly station, the robot will follow these steps:

1. Use the LiDAR sensor to localize itself within the map of the environment.
2. Use camera data to detect the object to be picked and its pose (position and orientation).
3. Use control variable to grasp the object with the mobile manipulators' end-effector.
4. Use LiDAR data to transport the object to the next station or location.
5. Use control variable to place the object in the designated spot with precision.

The system's real-time operation is controlled and monitored via the edge (local server), ensuring efficient, safe task execution. By reducing dependence on LiDAR for localization, 5G NR positioning enhances system autonomy, cuts down hardware costs, and improves overall safety and efficiency.

2.5 Revised requirements towards 5G

This section provides the updated network requirements for the "Edge-Controlled Automation with Mobile Manipulation" use case, reflecting the latest implementation and measurement results at the WZL testbed. The core network requirements remain consistent with those established in *Deliverable D2.2*; however, the values have been revised based on new data from recent trials. It's crucial to note that the criteria and terminology defining these requirements were established in WP1, ensuring a consistent portrayal of the use cases across the TARGET-X verticals and testbeds. Further details were given in D1.1 and D2.2.

The following tables present the updated values for each communication stream involved in the use case:

Table 1: Performance requirements for each data flow of the use case.

DATA FLOW		AVERAGE DATA RATES (Mbit/s)	LATENCY (ms)	AVAILABILITY (%)
CAMERA (Uplink)		13	4	≥ 99
LiDAR SENSOR (Uplink)	2D	2	20	≥ 99.9
	3D	127	20	≥ 99.9
CONTROL VARIABLES (Downlink)	BASE	0.01	20	≥ 99.9
	ARM	0.01	4	≥ 99.9

These revised values are based on updated measurements obtained via ROS commands during the current full-stack implementation, supplemented by expert estimations from WP6. Details on the methods used to collect and estimate these values can be found in *Deliverable D2.2*.

Table 2: Complementary requirements for each data flow of the use case.

DATA FLOW		MESSAGE SIZE (Bytes)	TRANSFER INTERVAL (ms)	UE VELOCITY (m/s)	# UE	SERVICE AREA (m^2)
CAMERA (Uplink)		70 k	40	0.45	1	20
LiDAR SENSOR (Uplink)	2D	35 k	120	0.45	1	20
	3D	640 k	100	0.45	1	20
CONTROL VARIABLES (Downlink)	BASE	50	500	0.45	1	20
	ARM	100	10	0.50	1	20

The data for the revised performance requirements were obtained by considering the specific latency needs of the application, which combines network Round-Trip Time (RTT) and computation time. For the base, which operates at a desired speed of 0.45 m/s with a minimum tolerance of 1 cm, the required application RTT was measured at 20 ms. This includes the time for sending data, processing it, and sending a response back. For the manipulator arm, with a desired speed of 0.5 m/s and a tolerance of 2 mm, a stricter RTT of 4 ms was calculated. The tests were conducted in an indoor environment using a private 5G NSA network. The system's reliability is critical, with the uplink (UL) for LiDAR data requiring greater than 99.9% reliability, reflecting industry best practices, and the downlink (DL) for control commands needing reliability equal to or greater than the uplink. The mobility requirements were defined with the desired operating speed for WZL is 0.45 m/s, although the maximum speed of 1.5 m/s.

These complementary parameters ensure that the data flows meet the use case's performance and reliability needs during execution.

Additionally, in *Deliverable D2.2*, it was determined that the 5G connectivity for the use case must meet key functional requirements. These include network configuration management for monitoring and managing the 5G testbed, ensuring the system can specify Quality of Service (QoS) demands and gather performance metrics. Mobility management support is also essential for the mobile manipulator, particularly in the confined spaces of a factory. Finally, end-to-end (E2E) QoS support is required to guarantee low latency, high reliability, and sufficient throughput for communication services, covering all segments of the 5G system.

2.6 Positioning with 5G NR

In the context of the "Edge-Controlled Automation with Mobile Manipulation" use case, positioning with 5G NR presents a compelling solution for industrial applications, particularly as a cost-effective alternative to expensive sensors like LiDAR. While other positioning technologies were explored, from an industrial point of view, 5G NR stands out as a potential solution to reduce hardware costs while maintaining the precision needed for path planning and safety in mobile robotics.

This approach centers around using 5G NR signals to estimate the position of mobile robots within the factory environment. By utilizing signals between the 5G User Equipment (UE) and 5G gNodeB (network nodes), the system aligns with the 3GPP Release 16 standards for indoor positioning. This method integrates 5G NR-derived position estimates with data from other onboard sensors, offering a sensor fusion approach that balances cost and performance. The goal is to leverage 5G NR positioning to complement and potentially replace more costly solutions like LiDAR, especially for global localization tasks, without compromising the accuracy needed for precise path planning.

5G NR positioning is particularly valuable for industrial environments because it reduces the reliance on expensive hardware, offering a more scalable and economically viable solution for mobile manipulation tasks. In the WZL testbed, 5G NR positioning is being implemented and validated in two phases: technology introduction and experimentation with use case-oriented applications. The results aim to demonstrate how 5G NR can effectively support mobile robot localization in dynamic industrial settings, ultimately paving the way for its integration into future 6G systems. In Figure 2.3, results of benchmarking and trials done by WP6 and WP2 at WZL Testbed. More details on the integration will be given in Section 3.3.4.

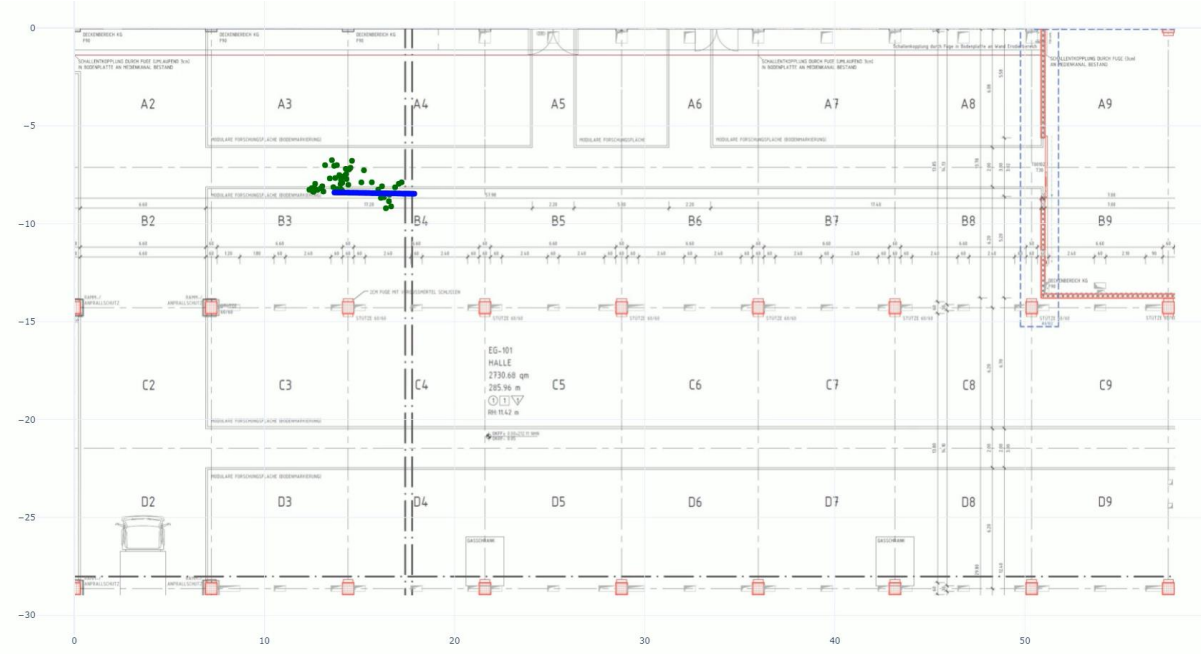

Figure 2.3: 5G NSA dots map at RWTH-WZL shopfloor. Benchmarking of Localization for Mobile Robots. In blue is the laser tracker, in green is the 5G NR Positioning.

3 System Architecture and Implementation at the WZL Testbed

3.1 Overview of System Architecture

The system architecture for the edge-controlled automation use case is built around a modular and scalable setup, as shown in Figure 3.1, designed to handle real-time operations in industrial environments. This architecture combines hardware components such as the mobile manipulator, sensors (LiDAR, cameras), and 5G-enabled communication infrastructure with a robust software stack running on both the robot and edge servers. The software components handle key functions such as localization, perception, and motion planning, leveraging the real-time data processing capabilities provided by the 5G network and the edge server. The integration of 5G features enhances the mobility and precision of the mobile manipulator by enabling the offloading of computationally intensive tasks—such as real-time perception and motion planning—to the edge server. This reduces the processing burden on the robot and ensures that data-heavy tasks are handled efficiently. The low-latency and high-bandwidth capabilities of 5G allow for seamless communication between the robot and the edge server, optimizing the overall system performance and supporting real-time decision-making in dynamic industrial environments.

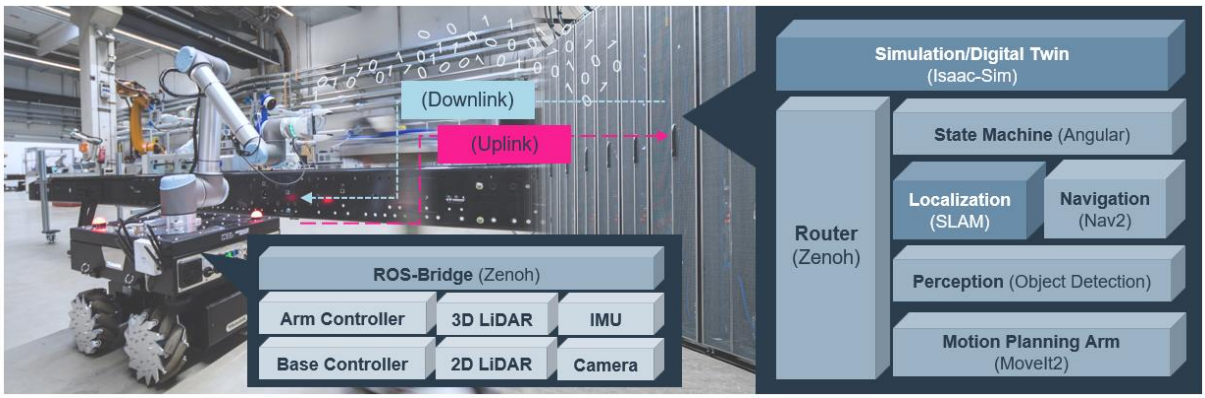

Figure 3.1: Depicted edge-controlled architecture for the mobile manipulation use case. The task-driven software components are distributed between on-board robot (left) and edge server (right) according to their computing demands.

3.2 Hardware Components and Setup

3.2.1 Components

As already presented in D2.2, the hardware used for the edge-controlled mobile manipulation use case includes:

Mobile Manipulator: A robotic arm mounted on a mobile base equipped with multiple sensors (LiDARs sensors, camera, IMU). The mobile manipulator **RB-Kairos+**, features an omnidirectional wheelbase and a UR10 robotic arm with six degrees of freedom (DOF). The UR10 is equipped with a two-finger gripper, a RealSense L515 end-effector camera (with depth and RGB sensors), and two SICK S300 2D LiDAR scanners for a full 360° field of view. Additional sensors include an inertial measurement unit (IMU) and hall effect encoders to ensure precise motion tracking. The robot is capable of speeds up to 1.5 m/s, while the arm's third wrist, where the camera holder is mounted,

moves at a speed of 0.2 m/s. The current configuration achieves a repeatability of approximately 5 mm between the commanded and actual robot pose. The 5G modem is mounted on the base of the RB-Kairos+, with future updates planned to include new 5G Release-16 devices provided by Fivecomm, further enhancing the robot's connectivity for real-time operations.

3D LiDAR Sensor (mounted on the Mobile Manipulator): The Ouster OS1 3D LiDAR, as shown in the picture below, used in the edge-controlled mobile manipulator, provides high-precision 3D mapping with a 360° horizontal field of view and 45° vertical coverage. It offers up to 120 meters of range, configurable vertical resolutions of 16, 32, or 64 channels, and a data rate of up to 2.62 million points per second. Designed for industrial environments, it operates with a Class 1 eye-safe laser and has an IP68/IP69K rating for water and dust resistance. Its robust design, along with Gigabit Ethernet connectivity, makes it ideal for applications requiring real-time data in dynamic environments.

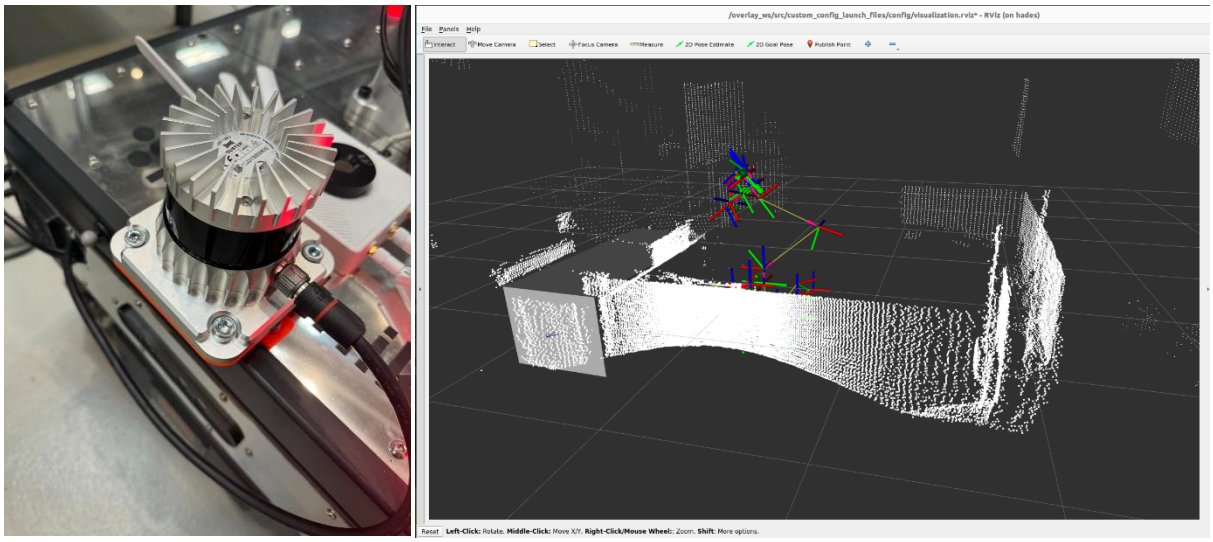

Figure 3.2: Ouster OS1 3D LiDAR mounted on the mobile manipulator, providing high-resolution 3D mapping capabilities for navigation and object detection. On the right is the visualization of the streamed Cloud points from the 3D LiDAR Sensor.

Edge server: a local server system serves as the central hub for computational tasks within the edge-controlled automation setup. This server processes computationally intensive operations, such as perception, motion planning, and sensor fusion. The server hardware is housed in the WZL trial site's factory cloud infrastructure. The configuration includes two AMD EPYC 75F3 processors, each with 32 CPU cores at 2.95 GHz, providing ample computing power. The system also includes 16 RDIMM memory modules (256 GB in total), SSD storage for high-speed data access, and an NVIDIA Ampere A10 GPU for accelerated processing, particularly for tasks involving computer vision and AI-based algorithms.

5G Modem: a 5G modem developed by Fivecomm was integrated with the mobile manipulator. The modem, which is 5G Release-16 compatible, is based on the Quectel RG520N module with Qualcomm's Snapdragon X62 chipset. The modem connects to a Raspberry Pi 4, which runs OpenWRT for network management, enabling reliable communication with the 5G infrastructure. Each modem is equipped with external antennas and supports Ethernet and USB-C connectivity.

These setups, illustrated in Figure 4.1 of *D2.2*, form the basis of the system architecture for the use case.

3.2.2 Integration of current 5G Features

One of the necessary steps towards the successful implementation and validation of the robotics use case was the integration of **5G modems** with the network. Two units of 5G modems designed and developed by Fivecomm were used.

The modems are 5G Release-16 compatible, since they integrate a Quectel RG520N module, which in turn includes the Qualcomm Snapdragon X62 chipset. The 5G modem is formed by a 5G board implementing the necessary interfaces, used as a hat on top of a Raspberry Pi 4. The Raspberry Pi 4 has been included so the modem implements an OpenWRT Linux distribution and supports Ethernet connection. The modems have been provided to the project with a plastic case that permits to plug external antennas via SMA connectors, provides access to both SIM and SD cards and includes 3 LED for monitoring the modem status. Power supply can be provided via 9-12V as well as through USB-C. The second alternative is considered important, as it permits the user to mount the 5G modems in moving devices when needed. Figure 3.3 shows how these modems look like.

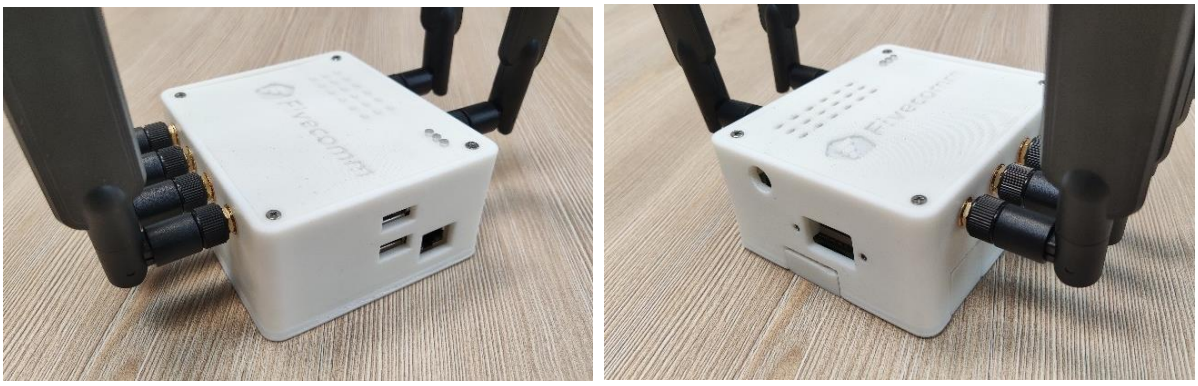

Figure 3.3: 5G modem integrated in the robotics use case.

Note that this model of 5G modem is different from the one used in the manufacturing use cases, as only the 5G hat part was needed for such a case. In addition, the GPIO PINs used between the hat and the Raspberry Pi 4 are different. Readers may refer to D2.4 for additional information.

Some of the main functionalities of these modems are the following:

- Non-standalone (NSA) and standalone (SA) compatible for Release 16 (3x, 3a and 2 architecture options).
- Network slicing support.
- Low- and mid-band frequencies (including n77, n78, and other main bands).
- Up to 6 external antennas supporting MIMO.
- USB and Ethernet connectivity support.
- Routing and port forwarding capabilities for external applications.
- Wi-Fi access point.

As mentioned above, the modems come with the firmware OpenWrt, which can be installed on devices such as the Raspberry Pi 4 included, to turn them into a network-oriented equipment. It

allows for extensive customization and supports a broad range of network functions. Apart from OpenWRT, two components of its base features are also included, i.e., the Quectel drivers and Quectel Connect Manager (quectel-CM).

Integration within the robotics use case

The 5G modem from Fivecomm was successfully integrated into the Kairos Mobile Manipulator. The modem operates using 5G Non-Standalone (NSA) at the 5G-Industry Campus Europe (5G-ICE) and is currently in the process of being fully integrated with the manipulator's power supply system. This setup ensures seamless communication between the robot and the edge server via the 5G network. Figure 3.4 illustrates the implementation process and the modem's integration within the mobile manipulator system.

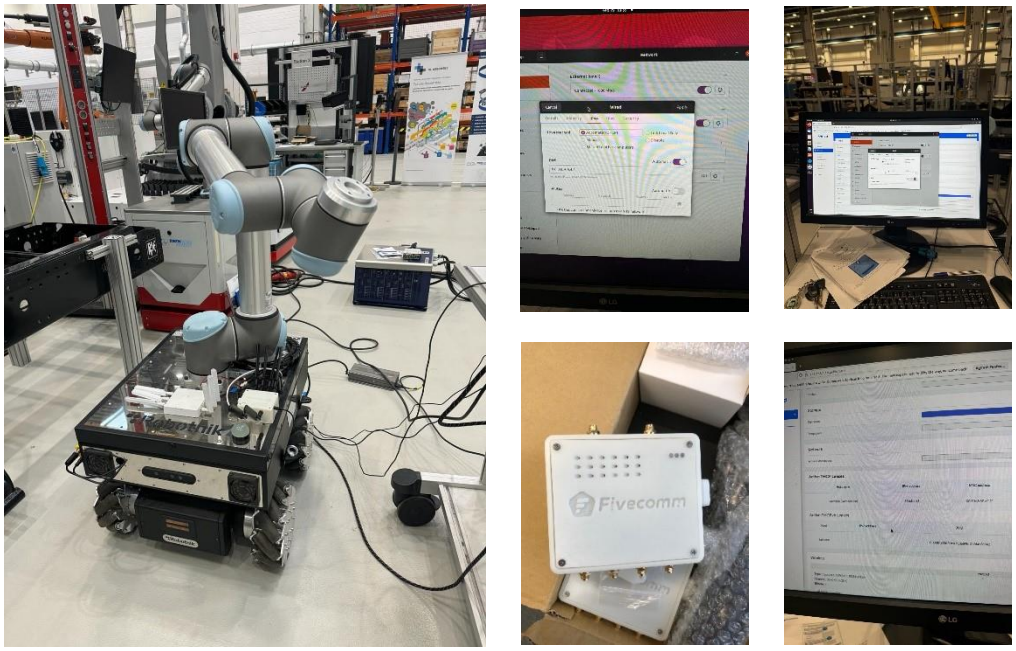

Figure 3.4: Integration of the 5G Modem at WZL Testbed.

3.2.3 Integration of Positioning with 5G NR

The localization system for mobile robots in the WZL Testbed leverages 5G NR signals and a sensor fusion approach to accurately determine the robots' positions within the factory environment. This setup includes the installation of 5G radio dots, precisely positioned using a laser tracker, to enhance signal coverage and accuracy. As shown in Figure 3.5 and Figure 3.6, installation of 3 additional 5G radio dots took place at the WZL Testbed.

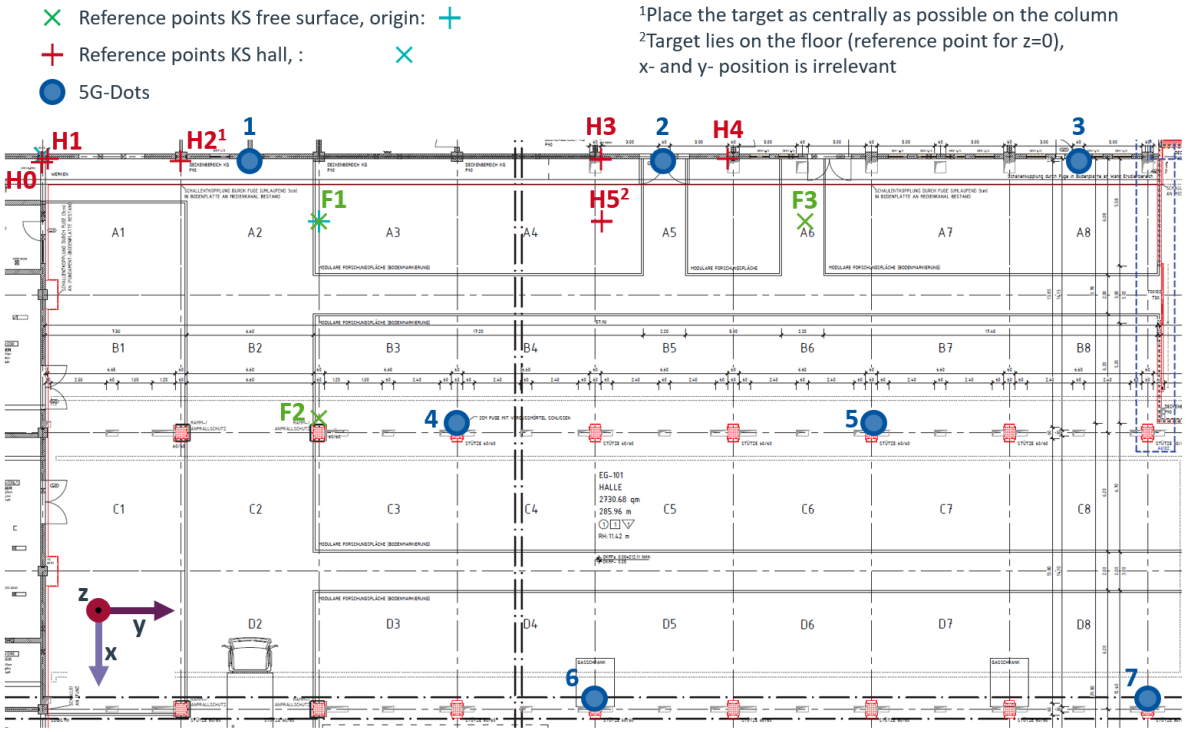

Figure 3.5: Installation of 5G-Dots at WZL Testbed in the Coordinate system(s) of RWTH-WZL-IQS.

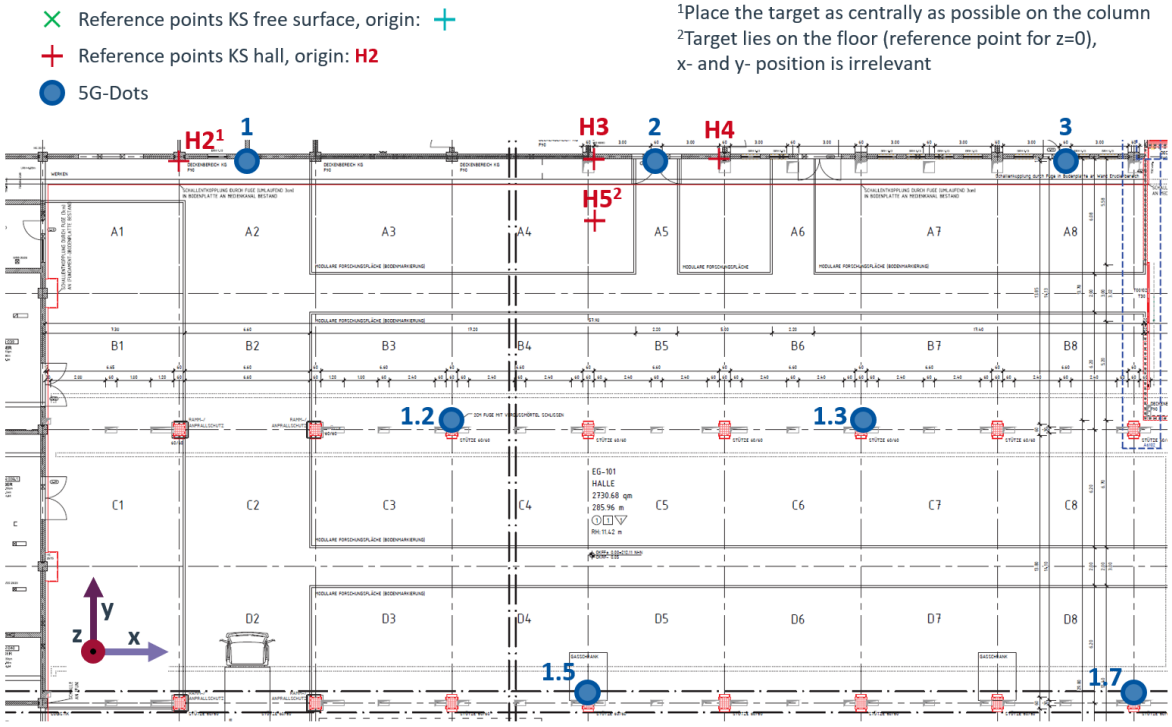

Figure 3.6: Installation of 5G-Dots at WZL Testbed in the Ericsson Coordinate system.

The system adheres to 3GPP Release 16 standards, utilizing signals between 5G User Equipment (UE) and 5G gNodeB nodes to estimate the robot's position, as shown in Figure 3.7. An industrial PC (IPC) processes the position data in real-time and sends the calculated positions to an MQTT broker, which is then used for benchmarking the accuracy of 5G NR-derived positions against reference measurements from a laser tracker.

Figure 3.7: Initial trials on Benchmarking of Localization via 5G NR with a Mobile Manipulator at WZL Testbed.

3.3 Software Components and Development

The figure below illustrates the complete software stack for mobile manipulation, including the 5G positioning pipeline. It highlights the various software components distributed across different devices, demonstrating the integration and coordination required for efficient system operation.

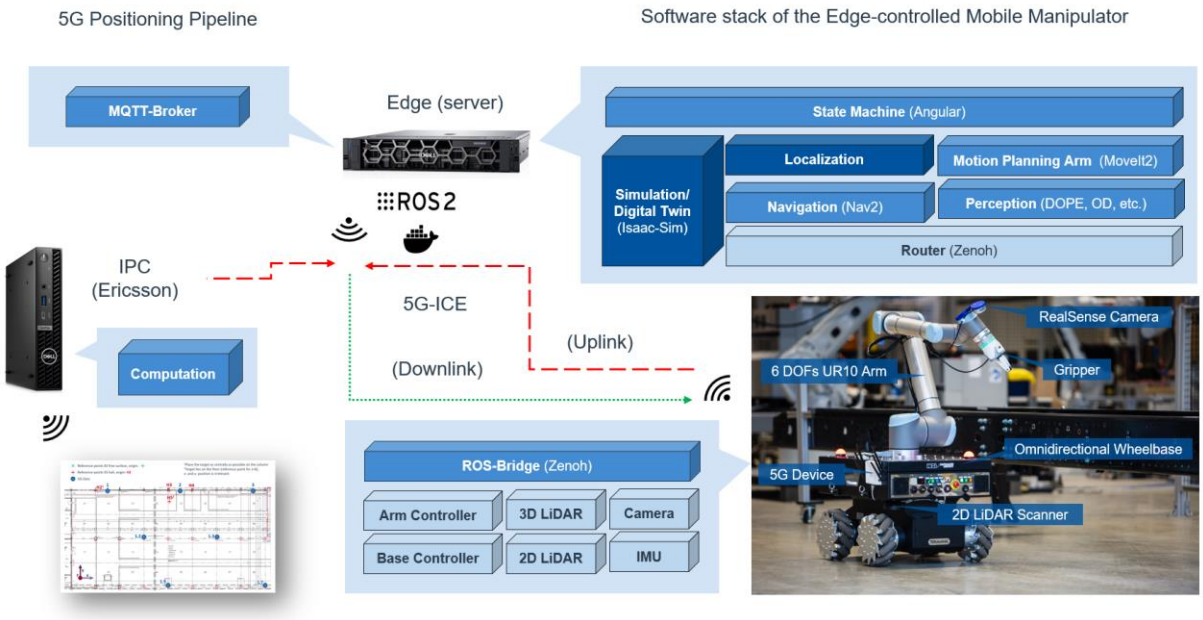

Figure 3.8: Architecture of the Edge-Controlled automation with mobile manipulation. On the left is the 5G Positioning Pipeline. On the right is the software stack for edge-controlled mobile manipulation.

3.3.1 Software Stack for Edge-Controlled Automation

The software architecture for the edge-controlled mobile manipulation is built around Robot Operating System (ROS) 2, which provides modularity and scalability for managing various robotic functions. The Stack includes:

- **Git repositories**: Git [15] is a widely used distributed version control system that enables multiple developers to collaborate on code while tracking changes and maintaining a history of modifications. One of its key advantages is the ability to branch and merge, allowing developers to work on new features, bug fixes, or experimental changes in isolation before integrating them into the main codebase. Every component of the software stack as shown in Figure 3.8 is tied to a specific Git repository (repo), which ensures traceability, modularity, and structured development. Each repository can have multiple branches, such as development, testing, and production, to facilitate smooth workflow transitions.
- **Containerization**: Docker [16] is used to containerize different components of the system, enabling efficient resource allocation, scalability, and easy deployment. In the context of ROS 2, Docker provides isolated environments (containers) where different ROS 2 nodes and services can run independently, without conflicting with other system dependencies or libraries. This is particularly useful in complex robotic systems, where various components such as perception, navigation, and control need to operate simultaneously. By containerizing ROS 2 nodes,

developers can ensure consistency across different environments, from development to testing and production, as each container holds its own dependencies and configurations. This modularity also simplifies debugging, scaling, and updating specific components without affecting the entire system. Additionally, Docker's integration with ROS 2 supports seamless orchestration of multiple containers, enabling distributed computing across edge devices and cloud infrastructure. As illustrated in Figure 3.9, Docker containers running ROS 2 nodes can easily be deployed on the edge server, ensuring robust and efficient communication between the robot and the server.

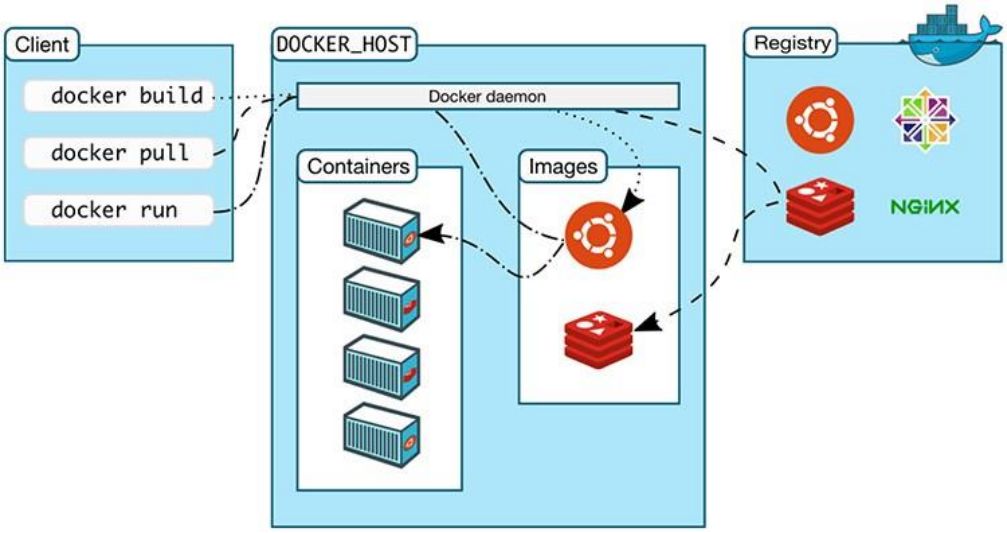

Figure 3.9: Docker Architecture. Source: [17].

- **Resource Allocation**: The edge server handles computationally intensive tasks such as perception and motion planning, while the robot handles real-time tasks like sensor data collection.
- **ROS 2**: an open-source framework designed for building and managing robotic systems, providing modularity, scalability, and real-time performance. Unlike ROS 1, it uses a decentralized architecture based on the Data Distribution Service (DDS), enabling direct communication between nodes without a central master, as shown in Figure 3.10. This improves real-time capabilities and scalability, crucial for distributed systems like mobile manipulation. ROS 2 supports seamless integration with various sensors, actuators, and control systems via flexible middleware, making it ideal for complex applications. Its advanced Quality of Service (QoS) policies allow for fine-tuning communication parameters, enhancing reliability and efficiency in edge-controlled environments. ROS 2 is also multi-platform and works well with containerization tools like Docker, simplifying deployment across edge servers and robots.

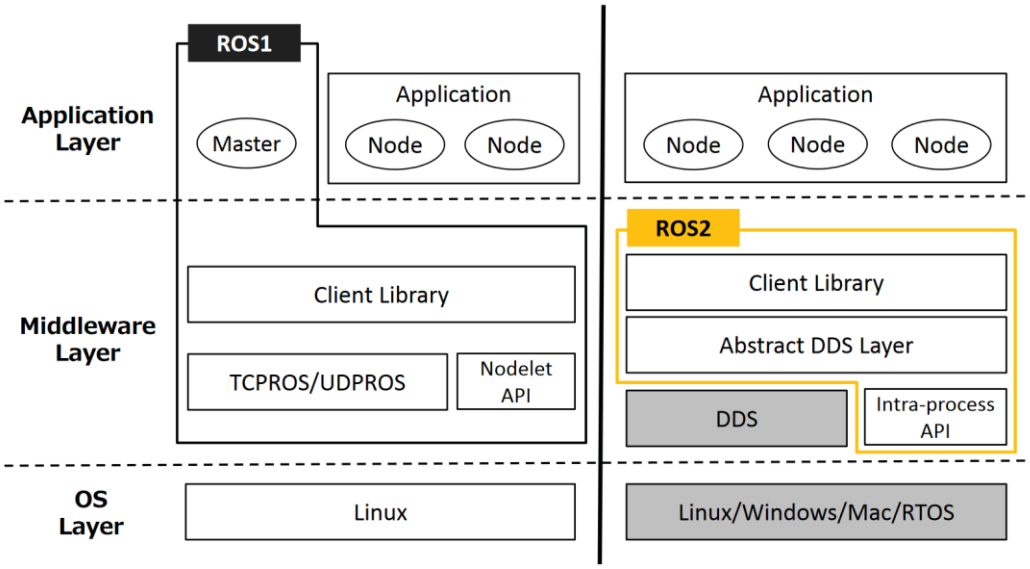

Figure 3.10: ROS1/ROS2 architecture. Source: [18].

3.3.2 Components Running on the Edge Server

The edge server runs several critical components as introduced in Figure 3.8 and in the figure below, arranged in the sequence of the data flow (from start to end):

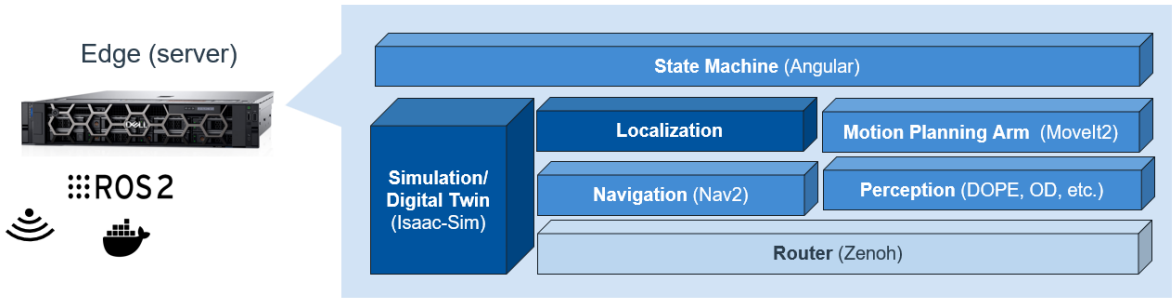

Figure 3.11: Software Components (repos) running on the Edge Server.

- **Localization**: This component processes data from the 2D and 3D LiDAR sensors, as well as the IMU, using the SLAM algorithm. SLAM enables the robot to build a map of an unknown environment while simultaneously tracking its position within that map. It operates by continuously scanning the surroundings with sensors, updating the map, and refining the robot's location on the map in real time. In this use case, SLAM is used to accurately track and estimate the robot's location within a previously recorded map, ensuring precise navigation and localization, even in dynamic or complex environments.
- **Navigation (Base)**: The navigation component handles path planning for the mobile base, including dynamic path adjustments. Once a goal pose is set—either by the operator or the state machine—the navigation stack plans the path, considering both fixed and dynamic obstacles

along the way. In robotics, "pose" refers to the combined position and orientation of a robot or object in a 3D space, describing both its location and the direction it faces.

- **Perception (Object Pose Estimation)**: When the mobile base reaches the goal pose, the perception component is activated (either by the operator or the state machine). It processes camera data to detect the object to be grasped and estimates its pose. In robotics, the "pose" refers to the combined position and orientation of an object in 3D space. Once the pose is detected, the necessary metadata is generated, and the grasping operation can be triggered.
- **Motion Planning (Arm)**: This component is responsible for planning the movement of the manipulator (arm). It uses inverse kinematics to calculate the necessary joint positions and velocities to move the arm from its current state to the goal pose (object pose). The output is a planned trajectory that can be executed immediately or triggered later by the operator or the state machine.
- **Router (*Zenoh*)**: The *Zenoh* router component manages communication between the robot and the edge server, coordinating the flow of data (topics) between ROS nodes. It also handles client-server configurations, ensuring efficient data traffic management.
- **State Machine**: The state machine oversees the execution of tasks and manages decision-making processes for the application. The state machine must be pre-scripted, and all other components must be enabled and operational for it to function. Alternatively, each component can be triggered individually by the operator without the need for the state machine.
- **Simulation/Digital Twin** (Isaac-Sim): Isaac-Sim, developed by NVIDIA, is a powerful simulation platform for testing and developing robotics applications in a virtual environment. At the WZL Testbed, a digital twin of the edge-controlled mobile manipulator replicates the entire software stack, including simulated sensors like cameras and LiDAR. This allows for comprehensive testing and debugging of the software components before deployment on physical hardware. Isaac-Sim can run independently, supporting continuous development, but it requires specific GPU capabilities to handle high-fidelity simulations and sensor data processing, making it essential for efficient software development.

3.3.3 Components Running on the Robot

Key software components running directly on the robot include, as shown in Figure 3.8and below:

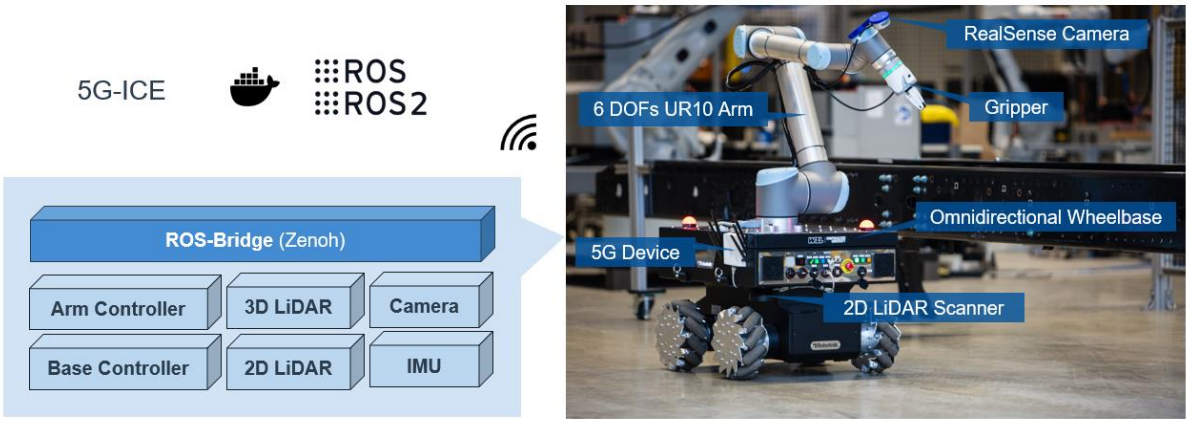

Figure 3.12: Software Components (repos) running on the Mobile Manipulator.

- **ROS-Bridge**: This component plays a crucial role in translating messages and topics between ROS 1 and ROS 2, as the Kairos robot still operates on ROS 1. The ROS bridge is a common solution for managing ROS 1/ROS 2 migration, particularly for robotic systems that are restricted to ROS 1 due to manufacturer limitations. In addition to handling message translation, this component also integrates *Zenoh*, which manages communication traffic between the robot and the edge server, ensuring continuous data exchange.
- **Base & Arm Controller**: both components manage the motion of the mobile base and the robotic arm. The ROS interfaces for these controllers are typically provided by the manufacturers, allowing the system to communicate with the hardware effectively. However, these interfaces need to be properly configured and wrapped as containerized components, tailored to the specific use case. This ensures that the controllers integrate seamlessly with the overall ROS-based architecture, enabling precise and coordinated movement of the robot's base and arm.
- **LiDAR Sensor**: This component manages the ROS interfaces for both 2D and 3D LiDAR sensors. These ROS interfaces are typically provided by the manufacturers but need to be properly configured for the specific use case. In this component, the configurations are containerized and wrapped as a distinct module to ensure smooth integration with the overall system.
- **Camera ROS**: This component handles the ROS interface for image capture and processing, enabling data to be sent to the Perception component. Like the LiDAR sensors, the ROS interfaces for cameras are usually supplied by the manufacturers, but they must be configured and containerized as a comp [19]onent tailored to the use case requirements.

3.3.4 5G NR Positioning Pipeline

The 5G NR positioning pipeline processes data from 5G signals and integrates it with other sensor data to provide accurate and real-time localization for the mobile robot. Following from the Figure 3.13, the pipeline begins by capturing 5G-based localization data in (A) using the 5G User Equipment (UE), which is measured in the Ericsson coordinate system (CS). Simultaneously, reference measurements in (B) are collected from systems like laser trackers, LiDAR sensor and OptiTrack for benchmarking purposes. These data sources are then synchronized using a script in (2) that aligns

the data across both the 5G localization and reference systems, transforming them into a common coordinate system (C) for consistency.

Next, the time-synchronized data (C) is transmitted via Message Queuing Telemetry Transport (MQTT) protocols to a middleware layer in (4), which facilitates communication between the mobile robot and edge server. The synchronized data, including positioning information from the 5G signals and reference systems, is stored in a database in (5) for further analysis. The middleware ensures smooth data flow, enabling fast and reliable communication.

Finally, this processed data is visualized in real-time using custom dashboards, with additional visualization through tools like Grafana in (6). These visualizations help track the robot's movement, compare 5G-based positioning with reference systems like laser trackers, and benchmark performance against LiDAR readings. This comparison ensures that 5G positioning is optimized for tasks like path planning and navigation within the dynamic industrial environment.

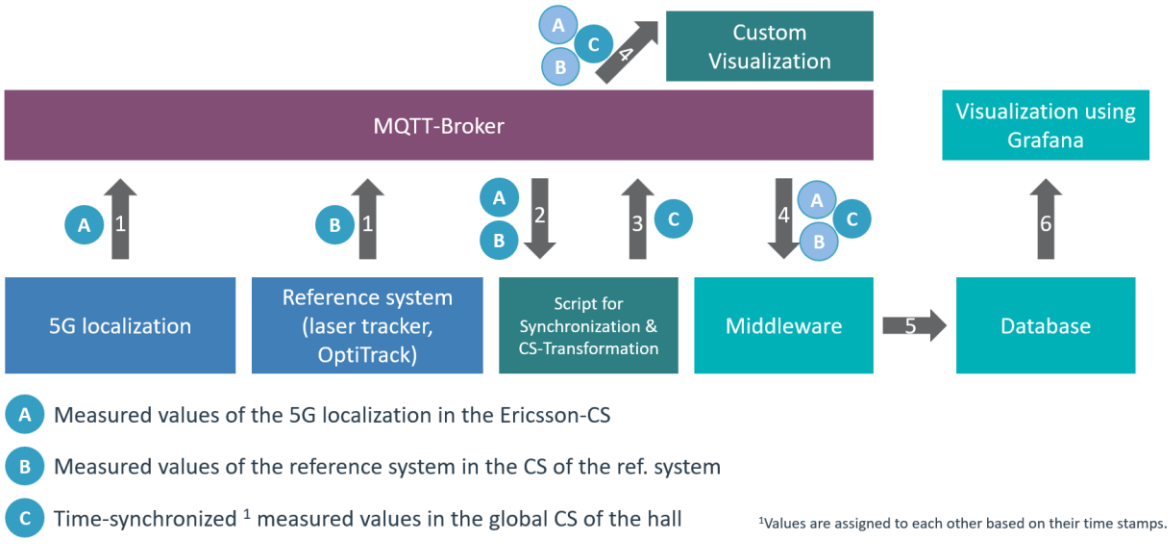

Figure 3.13: The 5G NR positioning pipeline, showing the flow from 5G localization and reference systems through synchronization, middleware, and MQTT for real-time visualization and benchmarking in Grafana.

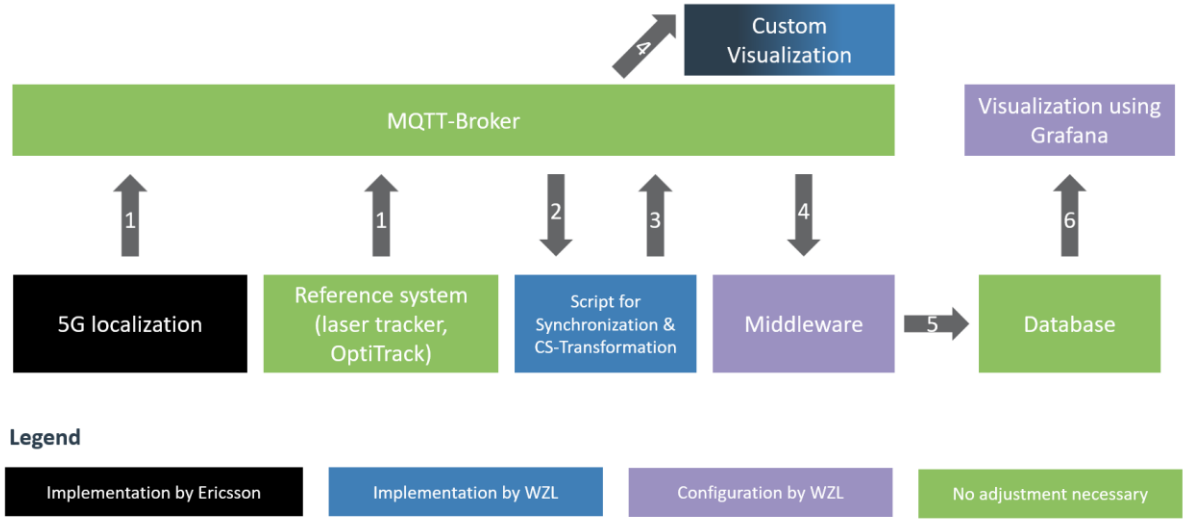

Figure 3.14: Overview of the 5G NR positioning pipeline implementation, highlighting responsibilities and data flow from localization and reference systems through synchronization, middleware, and visualization.

3.3.5 Software Implementation Process

The software implementation process for the edge-controlled mobile manipulation system involves the deployment of a robust ROS 2-based stack on both the edge server and the mobile manipulator robot, as shown in the Figure 3.15. The following steps detail how the software components are integrated and deployed across different devices, focusing on the use of containerization, communication protocols, and key development tools.

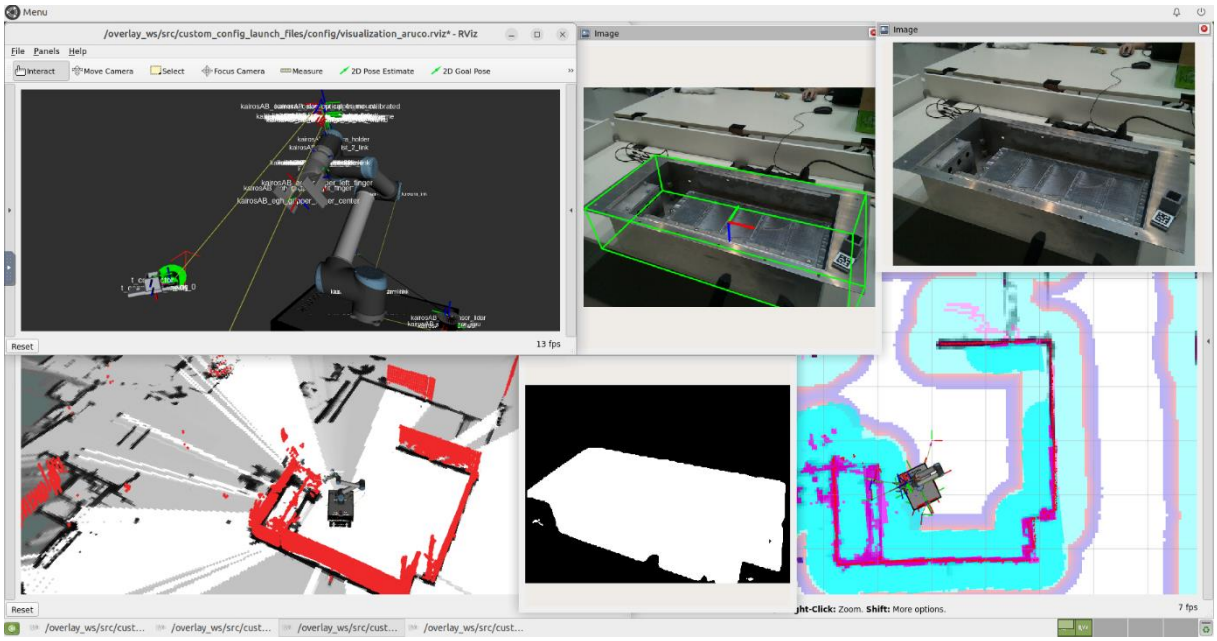

Figure 3.15: Visualization of the Full-Software-Stack of the Edge-Controlled Mobile Manipulation. Running process on the Edge Server.

Deployment Steps:

1. **Git repositories and development pipelines**:

In a Continuous Integration (CI) and Continuous Deployment (CD) pipeline setup, Git branches play an essential role, as the pipeline can be configured to run tests, builds, and deployments on specific branches. By attaching each software component to a designated Git repo and version, the system ensures that changes are isolated, tested, and can be easily rolled back if issues arise, maintaining the integrity and stability of the codebase across different environments. CI/CD pipelines automate the process of testing, building, and deploying code changes to production environments, ensuring that each modification is thoroughly validated before release. Typically, when new code is pushed to the Git repository, the CI pipeline is triggered to run automated tests, linting, and other validation processes. If the tests pass, the CD pipeline can then automatically deploy the code to staging or production environments, in this use case, the direct deployment happens on the edge-server and in the robot (mobile manipulator). This integration enhances development agility, reduces manual intervention, and ensures that software can be delivered more frequently and reliably. Figure 3.16 exemplifies the process.

2. **Containerization**:

Docker is employed to containerize different software components and ROS interfaces, making the deployment process more manageable and ensuring consistent performance across diverse environments. Each ROS (1 or 2) interface is first encapsulated within a Docker container and within a repository, allowing for modular and flexible deployment. This process applies for every sensor interface (Camera ROS, LiDAR, etc.) and software tool (Nav2, MoveIt2, etc.). This approach also simplifies resource allocation between the edge server and the mobile manipulator, enabling efficient scaling of resources.

As already exemplified in D2.2 and shown in the figure below, the development process follows the approach presented in the Figure bellow, where all the component (git repos) and their respective images are then deployed in the edge server and in the robot, without the need of physically (manually) building them at the machines (edge server and robot or other intended devices).

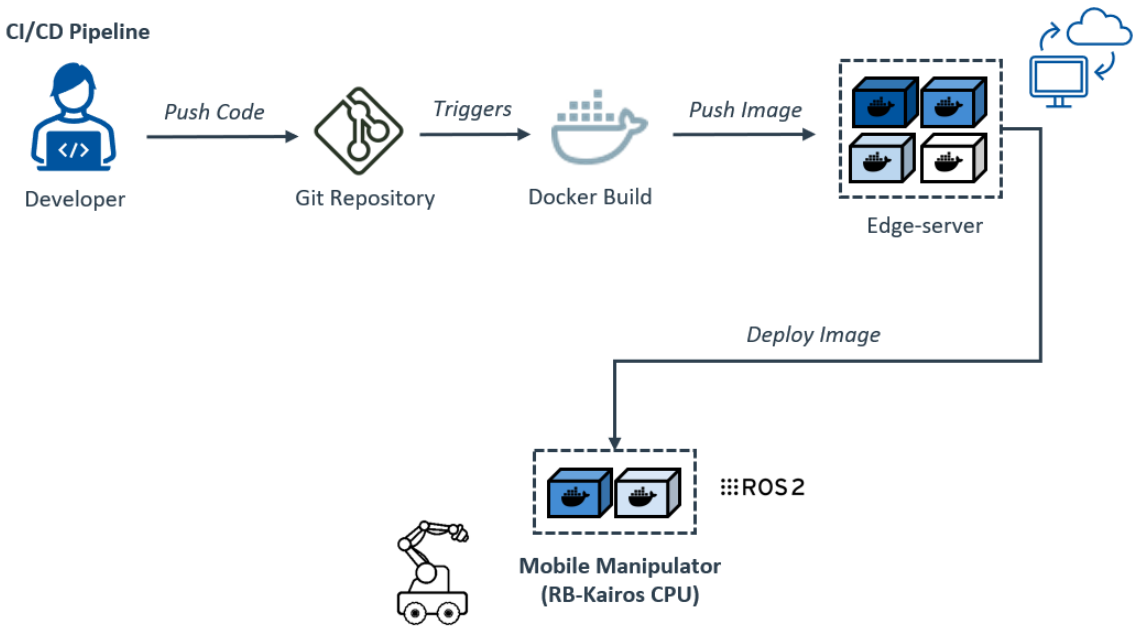

Figure 3.16: Process of the software development using CI/CD Pipelines and automatics deployment in the edge server and in the robots.

3. **Edge Server Setup**:

The edge server plays a crucial role in processing data-heavy tasks such as localization, path planning, and perception. Key software components deployed on the edge server include:

Localization: As mentioned before, the localization component within the ROS 2 stack utilizes the Cartographer SLAM algorithm, a robust tool for real-time 2D and 3D simultaneous localization and mapping across various platforms and sensor configurations. This implementation supports both mapping and localization modes—mapping mode is employed to create new environmental maps, while localization mode uses these maps to precisely position the robot within its surroundings. The localization process integrates data from Lidar and IMU sensors, significantly enhancing the robot's spatial awareness and navigation capabilities. This component is crucial for ensuring that the robot can accurately localize itself in dynamic and complex environments, enabling reliable navigation and task execution within the ROS2 framework [20]. The localization stack is displayed via the visualization tools with the Navigation stack in Figure 3.17.

Navigation (Nav2): The Navigation 2 (Nav2) package in the ROS 2 stack is a key component for delivering advanced navigation and path planning capabilities for mobile robots. Nav2 operates by coordinating a collection of nodes, each dedicated to specific functions such as cost map management, global and local path planning, behavior tree execution, and sensor data processing. Its modular architecture allows for the easy addition or removal of nodes to suit the specific needs of different applications. Communication between these nodes is handled through ROS 2's QoS settings, which provide detailed control over network performance, including aspects like reliability and latency [21].

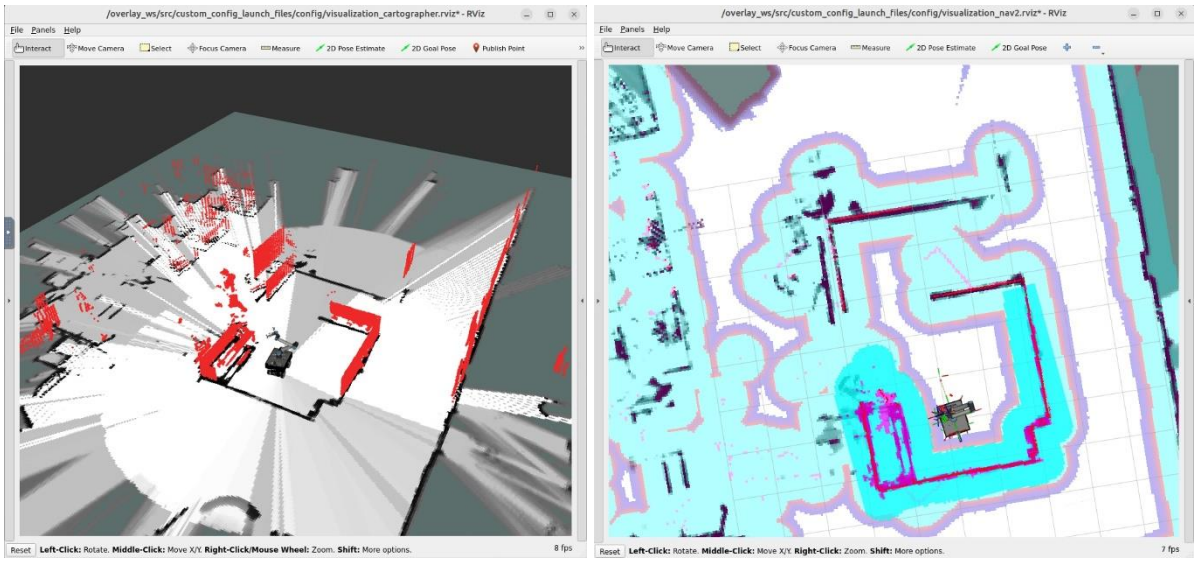

Figure 3.17: Visualization of the Localization and Navigation (NAV2) stack from the real robot during execution.

Perception: As previously mentioned, the perception component is responsible for detecting the object to be grasped using input from the vision system (camera sensor). The critical output required is the pose estimation (metadata) of the detected object, which is necessary for triggering motion planning and solving the inverse kinematics problem. The algorithm employed for this task is **FoundationPose**, an advanced AI-powered perception module specifically designed for object pose estimation using RGB-D (color and depth) images, as illustrated in Figure 3.18. FoundationPose leverages foundation models to provide precise 6-DoF (degrees of freedom) object pose estimations, making it particularly effective for robotic manipulation tasks, such as pick-and-place operations in dynamic and unstructured environments. By utilizing pre-trained deep learning models, FoundationPose efficiently performs object recognition and pose estimation, even in complex scenarios, enhancing the robot's ability to interact with and manipulate objects in real-world applications. In this use case, a CAD file of a battery pack was used to train the algorithm, with the training process relying solely on synthetic data generated via Isaac-Sim from NVIDIA Omniverse [22].

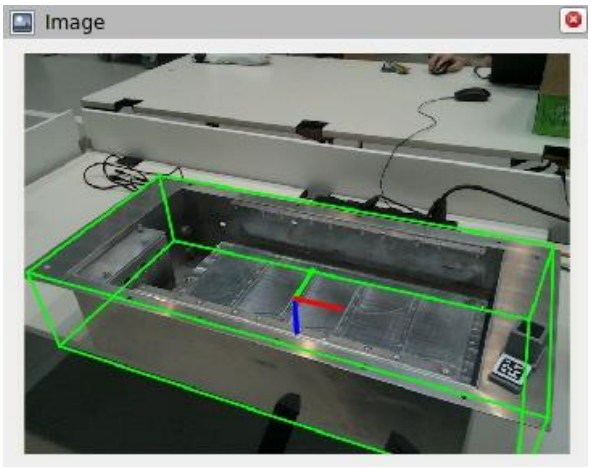

Figure 3.18: Pose estimation of the Battery pack as an object running in real-time.

Motion Planning (MoveIt2): MoveIt2 is a vital motion planning framework within the ROS2 ecosystem, designed to manage complex robotic tasks. Building on the original MoveIt framework, MoveIt2 integrates enhancements that leverage ROS2's advanced features, including real-time processing, multi-threading, and secure communication. One key component, the MoveIt Task Constructor (MTC), improves task-level motion planning by breaking down complex operations into smaller, manageable subtasks, which are then solved and combined into a comprehensive plan. This modular approach is particularly beneficial for intricate tasks like pick-and-place operations, providing flexible and efficient motion planning in dynamic environments. Once the goal pose is obtained from the perception algorithm, MoveIt2 tackles the **inverse kinematics (IK)** problem to compute the necessary joint configurations for the robotic arm to reach the goal pose accurately. Inverse kinematics ensures that the robot's joints are positioned correctly to achieve the desired end-effector position and orientation. Solving IK is critical in enabling the robot to execute complex manipulation tasks like grasping objects with high precision, while also avoiding collisions and ensuring smooth motion [23].

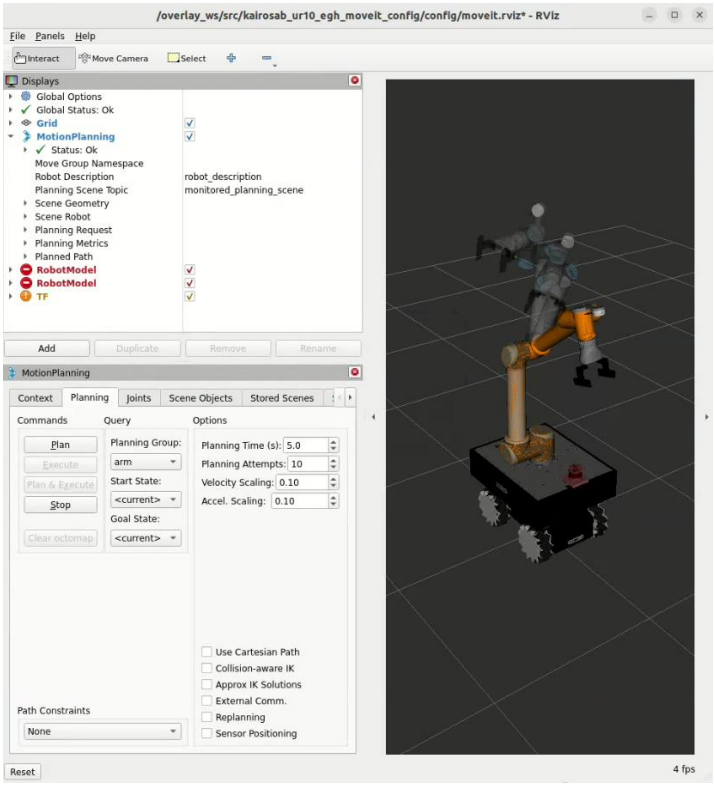

Figure 3.19: Motion Planning of the Arm. Integration of the URDF of the Kairos Mobile Manipulator with MoveIt2 Framework.

4. **Robot-Side Deployment**:

On the mobile manipulator, software components such as sensor drivers and controllers are installed to manage data acquisition and low-level control. This includes:

LiDAR Drivers: The ROS interface or LiDAR component is a ROS-based system essential for enabling high-resolution, real-time 3D mapping and perception. The integration of Ouster LiDAR sensors with ROS is achieved through the *ouster-ros* package, which provides drivers and tools to convert raw sensor data into ROS-compatible messages, such as *sensor_msgs/PointCloud2*. These messages are critical for tasks like navigation, SLAM, and obstacle detection, allowing robots to perceive and understand their environment in three dimensions [24]. In addition, the *point_cloud_transport* package enhances the management of point cloud data within ROS. Similar to *image_transport*, but specifically designed for 3D point cloud data, it offers various transport plugins that optimize data transmission over networks. The *point_cloud_transport_plugins* repository further extends these capabilities by providing additional plugins that support different compression algorithms, ensuring efficient and adaptable data transport based on network conditions and application needs [25]. Collected, these components enable robust and scalable 3D perception systems within ROS, which are crucial for autonomous navigation, mapping, and interaction with complex environments. By utilizing these tools, robots can effectively process and leverage Lidar data for a wide array of applications in dynamic and challenging settings.

Camera Drivers: The camera component in an ROS-based system is vital for enabling robots to interact with their environment through visual data. This is accomplished by integrating the camera

Intel RealSense, which provides advanced depth sensing and RGB imaging capabilities. The realsense-ros package serves as a ROS wrapper for RealSense devices, facilitating the streaming of depth and RGB data, which is essential for 3D perception tasks such as object detection and SLAM [26]. Additionally, the *image_pipeline* package in ROS supports various image processing tasks, including camera calibration, stereo image processing, and efficient image transport. These tools ensure accurate image representation and depth estimation, enabling robust real-time image capture and processing. Combined, these components are crucial for tasks that require visual input, empowering robots to navigate, recognize objects, and interact autonomously with their environment [27], [26].

Controllers: ROS 2 controllers for the mobile base and the arm manage the movement of the robot, translating high-level commands into precise motions.

In this architecture, the ROS-Bridge plays a pivotal role. Since the Kairos robot still operates on ROS 1, the ROS-Bridge component is responsible for translating messages and topics between ROS 1 and ROS 2 systems, ensuring that data flows seamlessly between them. This bridge is particularly useful in managing the migration from ROS 1 to ROS 2, especially when dealing with legacy systems or manufacturer-specific limitations that require ROS 1 compatibility. The ROS-Bridge ensures that all sensor data, such as LiDAR and camera inputs, as well as control commands for the mobile base and arm, are properly communicated between the ROS 1-based systems on the robot and the ROS 2-based edge server. Additionally, Zenoh is integrated to manage the communication traffic between the robot and the edge server, optimizing data exchange and ensuring low-latency operation even in a distributed setup.

5. **Middleware Configuration**:

ROS 2 uses DDS (Data Distribution Service) middleware for real-time communication between nodes. The CycloneDDS implementation is configured to enable low-latency, high-reliability communication between the robot and the edge server. Zenoh middleware is also integrated, allowing for efficient handling of discovery traffic and network management, particularly when managing communication between remote or distributed systems.

6. **Development Tools and Environment**:

The deployment process for the mobile manipulator relies on a range of ROS 2 tools and development environments to ensure efficient operation and accurate data handling. Key tools include:

- **RViz**: This tool is crucial for real-time visualization of the robot's sensor data, including LiDAR and camera inputs, path planning, and motion trajectories. RViz supports various plugins, such as Cartographer for SLAM and MoveIt2 for motion planning, allowing developers to monitor the state of the system in real time, facilitating debugging and optimization [28].

- **VNC** (Virtual Network Computing): VNC provides remote access to the ROS 2 environment, allowing developers to interact with the system from any location, whether for monitoring, debugging, or control. This is especially useful for large-scale deployments or geographically distributed teams, enabling real-time oversight of the system without physical presence at the testbed [29].
- **MQTT**: MQTT plays an important role in facilitating communication between the 5G positioning pipeline and the edge server. This protocol supports efficient data exchange and sensor fusion, which is critical for benchmarking 5G-based localization against other sensor inputs like LiDAR or visual data.

Robot Description: The Robot Description component in ROS 2 is fundamental for defining the robot's physical and kinematic properties using Xacro and URDF files. These files outline the robot's structure, specifying links, joints, sensors, and the transformations between coordinate frames. This structured description is critical for integration with key ROS components such as RViz for visualization, and for ensuring accurate performance of navigation, motion planning, and perception systems. Additionally, the robot description files are indispensable for simulation environments, enabling a precise digital twin of the robot, which is essential for testing and development [30].

Visualization: In ROS 2, RViz serves as the primary tool for visualizing and interacting with various components, including SLAM, navigation, and motion planning. RViz's compatibility with multiple plugins, such as Cartographer and MoveIt2, makes it an essential tool for monitoring real-time data and system states during operation. This visualization capability is particularly useful during demonstrations, system testing, and debugging [28].

To enhance accessibility, VNC is utilized to remotely access RViz and the broader ROS 2 environment. By providing real-time access via a web browser or VNC client, this tool allows developers to manage, monitor, and interact with the robot from any location, which is particularly beneficial in large-scale or distributed deployments [29].

The upper components described here are integral to the Kairos robot's operation within a production cell. As demonstrated in Figure 3.15, the workflow begins with the robot localizing itself within the environment using the localization module. This is followed by object detection, where the robot accurately identifies the location of the target object. After identifying the object, the robot proceeds with the grasping operation based on the pose estimation. Following this, the robot computes the optimal path to its goal location, using advanced path planning algorithms. Finally, the object is placed at the designated location, completing the task.

7. State Machine:

The state machine component plays a crucial role in managing the operational flow across different modules. It enables the definition of various states, transitions, and associated actions in a structured way. Implemented in Python, the state machine features a behavior-tree-like structure that allows for flexible management of state transitions. This flexibility is further enhanced by an Angular frontend, which provides a visual interface for dynamically managing and adjusting state transitions. This component is vital for ensuring smooth operation, effective error handling, and

adaptability in dynamic environments, contributing significantly to the system's overall reliability and performance [19].

8. Simulation/Digital Twin (Isaac-Sim):

As mentioned earlier, Isaac-Sim is an advanced simulation platform developed by NVIDIA, designed to create realistic environments for testing and developing robotics applications. It is especially valuable for validating software components in a controlled virtual environment before deployment on physical hardware. At the WZL Testbed, a digital twin of the edge-controlled mobile manipulator has been developed, replicating the entire software stack, including the components running on the robot. This digital twin integrates simulated sensor data, such as from cameras and LiDAR to closely mimic real-world scenarios.

This simulation environment is crucial for testing and debugging stack components in isolation, ensuring they perform correctly before being deployed in a physical setup. Furthermore, the simulation can be run independently of the physical system, allowing continuous development and testing of the software. However, Isaac-Sim requires specific GPU capabilities to handle the demanding tasks of rendering high-fidelity simulations and processing sensor data, making it a valuable tool for the efficient and reliable development of the mobile manipulator's software stack.

Figures 3.20 to 3.23 provide an illustration of the full software stack running in ROS2 within the NVIDIA Isaac-Sim environment, showcasing components such as localization, navigation, and motion planning. In Figure 3.20, the software stack is shown running with simulated camera and LiDAR data for localization and navigation, while Figure 3.21 presents a broader overview of the same stack. Figure 3.22 highlights the MoveIt2 motion planning stack in action within the digital twin framework. Lastly, Figure 3.23 demonstrates the mobile manipulator positioned for a grasping operation within the simulated environment.

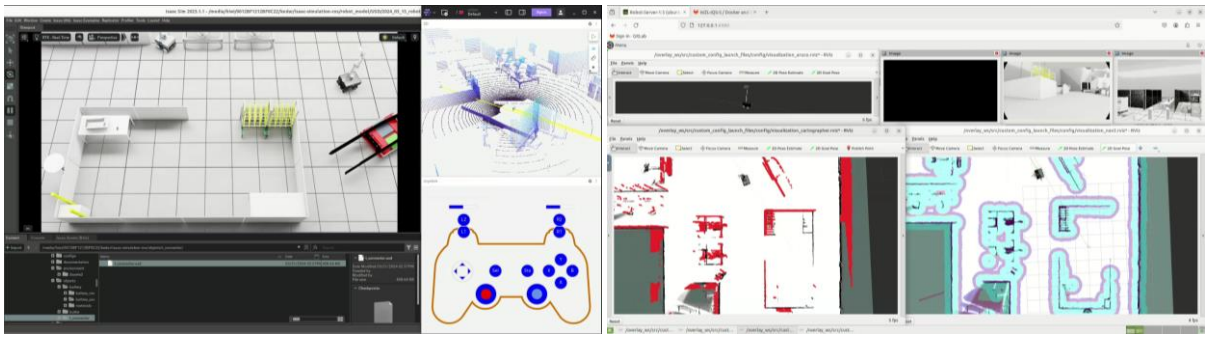

Figure 3.20: Full software stack in ROS2 running in the simulation environment of NVIDIA Isaac-Sim, including localization and navigation using simulated camera and LiDAR sensor data.

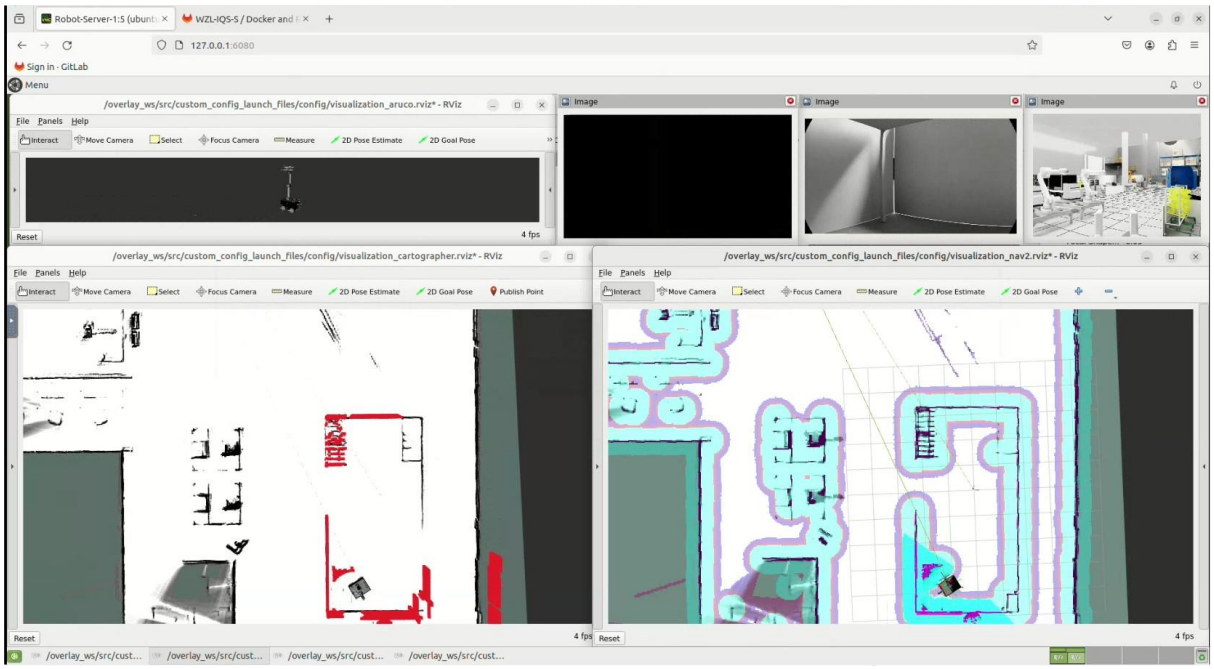

Figure 3.21: Full software stack in ROS2 running in the simulation environment of NVIDIA Isaac-Sim, demonstrating localization and navigation.

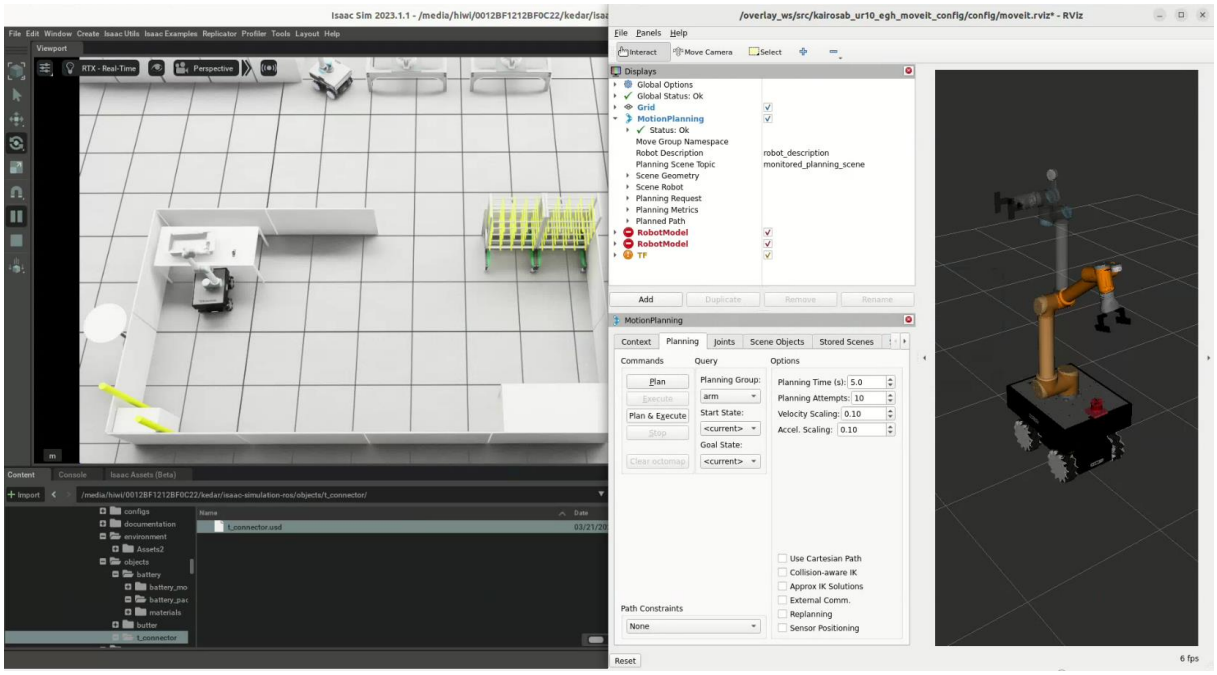

Figure 3.22: Motion Planning stack (MoveIt2) running on the Digital Twin Framework.

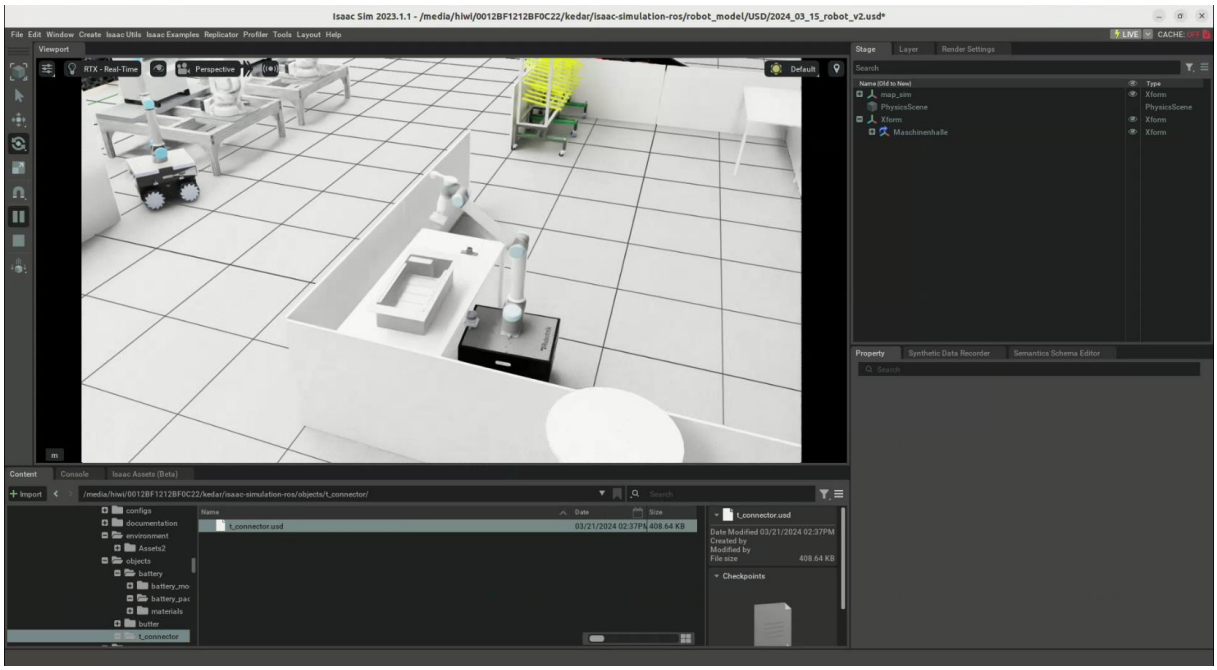

Figure 3.23: Digital Twin framework in NVIDIA Isaac-Sim, showing the mobile manipulator positioned for grasping.

By following this structured process, the edge-controlled mobile manipulation system can execute complex tasks such as localization, object detection, and motion planning in real-time. The system leverages containerization and advanced communication protocols like DDS and MQTT, ensuring seamless operation across the network and offloading computationally intensive processes to the edge server for optimized performance.

3.4 Communication and Interfaces

In this section, we describe the communication protocols and interfaces used in the edge-controlled mobile manipulation system, which ensure seamless interaction between hardware and software components.

The mobile manipulation use case leverages ROS 2 as the core middleware for communication between the robot's components and the edge server. ROS 2 provides a modular and scalable framework, which allows for real-time communication and message passing between various nodes within the system. The ROS 2 architecture supports decentralized communication, enabling distributed systems to exchange data directly, which is critical for applications like mobile robotics.

At the heart of ROS 2 is the DDS middleware, which facilitates reliable and efficient communication across the system. DDS enables peer-to-peer connections between the mobile manipulator and the edge server, without the need for a central master, enhancing the system's scalability. This system uses QoS policies to ensure that critical messages, such as localization updates and control commands, are delivered with the necessary reliability and latency. The communication is managed through ROS topics, services, and actions, allowing for both asynchronous and synchronous message passing.

The integration of Zenoh, a decentralized middleware, extends ROS 2's communication capabilities. Zenoh optimizes data sharing and synchronization across edge devices, cloud systems, and hybrid environments, offering a unified data plane that is more resilient to dynamic network conditions. The zenoh-bridge-ros2dds bridge ensures efficient communication between the ROS 2 system and Zenoh, reducing network overhead and managing discovery traffic.

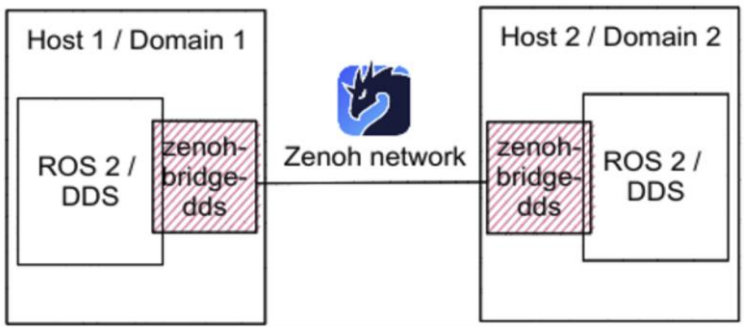

Figure 3.24: ROS 2 Middleware. DDS and Zenoh.

Data Flow Between Components

The flow of data starts from the mobile manipulator, where sensors like LiDAR and cameras generate raw data. These sensor streams are handled by ROS 2 nodes on the robot, which process the data locally or send it to the edge server for more computationally intensive tasks such as path planning and object detection.

- **Edge Server Communication**: The edge server manages critical tasks like localization, motion planning, and object perception. The processed data is then transmitted back to the robot using DDS and Zenoh protocols, allowing the robot to navigate and perform tasks in real-time.
- **5G Infrastructure**: The 5G modems integrated into the mobile robot provide low-latency, high-bandwidth communication between the robot and the edge server, ensuring that real-time data flows uninterrupted, even in dynamic industrial environments. The MQTT protocol is also employed for transmitting telemetry data, making it possible to benchmark robot positioning using data from both 5G signals and reference systems such as laser trackers or OptiTrack.

Together, these communication protocols enable the seamless exchange of data, ensuring real-time synchronization and coordination between the robot, the edge server, and the 5G infrastructure.

3.5 Currently executing of the Use case.

The entire software stack for the mobile manipulator is designed to be executed via the edge server. Once the operator gains access to the edge server and the necessary repositories are cloned from Git, the deployment process is straightforward. A single bash command, run.sh, launches the demo,

enabling seamless execution of the use case. This setup supports not only 5G networks but also Wi-Fi, offering flexibility depending on the network infrastructure available.

Figure 3.17 illustrates the private Git repository structure for the Kairos Stack, showing how each software component is neatly organized into submodules. This modular setup ensures ease of management and updates for each part of the system.

```
[MAIN_DIRECTORY]
├── kairos-stack
│   ├── scripts
│   │   └── run_XXX.sh
│   ├── docker
│   │   └── localization-ros
│   │       ├── docker_run
│   │       │   └── run_XXX.sh
│   │       └── docker_build.sh
│   ├── ...
│   ├── docker
│   │   └── nav2-ros
│   │       ├── docker_run
│   │       │   └── run_XXX.sh
│   │       └── docker_build.sh
├── ...
```

Figure 3.25: Git repo structure of the main software component Kairos Stack, in which all the software components mentioned before) are submodules.

As shown in Figure 3.18, the run script allows operators to easily enable or disable specific components of the software stack by commenting or uncommenting lines. Since the Docker images are already built through the CI/CD pipeline, no additional image building is required, streamlining the execution process.

```bash
#!/bin/bash

# Common config
export ROS_DOMAIN_ID=30
export ROS_DOMAIN_ID_ROBOT=42

export HOST_NAME=${USER}
export HOST_IP=localhost

export ROBOT_NAME=kairos
export ROBOT_IP=192.168.18.197
export ROBOT_PREFIX=kairosAB

SCRIPT="$(readlink -f $0)"
SCRIPT_DIR="$(dirname "$(dirname "$SCRIPT")")/scripts"
source ${SCRIPT_DIR}/common.sh

MAIN_FOLDER="$(dirname "$(dirname "$SCRIPT")")/docker"


COMPONENTS=(

    # Utils
    "zenoh","zenoh-setup/docker_run/run_two_hosts.sh" # Communication
    "visualization-ros","visualization-ros/docker_run/run_visualization.sh" # VNC visualization

    # Drivers
    "robot","bridge-ros/docker_run/run_bridge_and_kairos.sh" # Main driver + ROS1/ROS2 bridge
    #"foxglove-ros","foxglove-ros/docker_run/run_controller.sh" # Controll robot with ps4 controller
    "kairos-ur-ros","kairos-ur-ros/docker_run/run_kairos_ur.sh" # UR driver

    # Sensors
    "ouster-ros","ouster-ros/docker_run/run_ouster_driver.sh" # Driver 3D LIDAR
    "camera-ros","camera-ros/docker_run/run_camera_driver.sh" # Camera driver

    # Preprocessing
    "point-cloud-processing-ros","point-cloud-processing-ros/docker_run/run_point_cloud_processing.sh" # Preprocessing of point clouds

    # Robot specific software
    "robot-description-ros","robot-description-ros/docker_run/run_robot_description.sh" # Transform publisher
    #"moveit2-ros","moveit2-ros/docker_run/run_kairosAB_moveitgroup.sh" # Arm movement

    # Mapping/Localization/Navigation
    "localization-ros","localization-ros/docker_run/run_localization_ouster.sh" # SLAM
    "navigation-ros","nav2-ros/docker_run/run_navigation_ouster.sh" # Navigation

    # Perception
    "aruco-ros","aruco-ros2/docker_run/run_aruco_detecion.sh" # Aruco marker detection
    #"isaac-pose-estimation-ros","isaac-pose-estimation-ros/docker_run/run_pose_estimation_multi.sh" # AI pose estimation (Camera)
    "point-cloud-pose-estimation-ros","point-cloud-pose-estimation-ros/docker_run/run_inference.sh" # AI pose estimation (LIDAR)
    "foundationpose-ros","foundationpose-ros/docker_run/run_foundationpose.sh" # AI pose estimation (Camera)
    "nvblox-ros","nvblox-ros/docker_run/run_nvblox.sh" # 3D scene reconstruction

    # Control
    #"state-machine-ros","state-machine-ros/docker_run/run_state_machine_testing.sh" # State machine

    # Recording
    "rosbag-record-ros","rosbag-record-ros/docker_run/run_rosbag_record.sh" # Data recording

    "killer","$SCRIPT wait_for_user_and_kill"
)


# Check user input
if [ "$1" == "wait_for_user_and_kill" ]; then
    wait_for_user_and_kill "${COMPONENTS[@]}"
    exit 0
else
    start_components "$MAIN_FOLDER" "${COMPONENTS[@]}"
fi

```

Figure 3.26: Main run script of the Edge-controlled Mobile Manipulation Software stack.

Figures 3.19 and 3.20 showcase the successful demonstration of the use case during events and visits, including the TARGET-X Open Day Event in 2024, where the edge-controlled mobile manipulator was presented to a live audience.

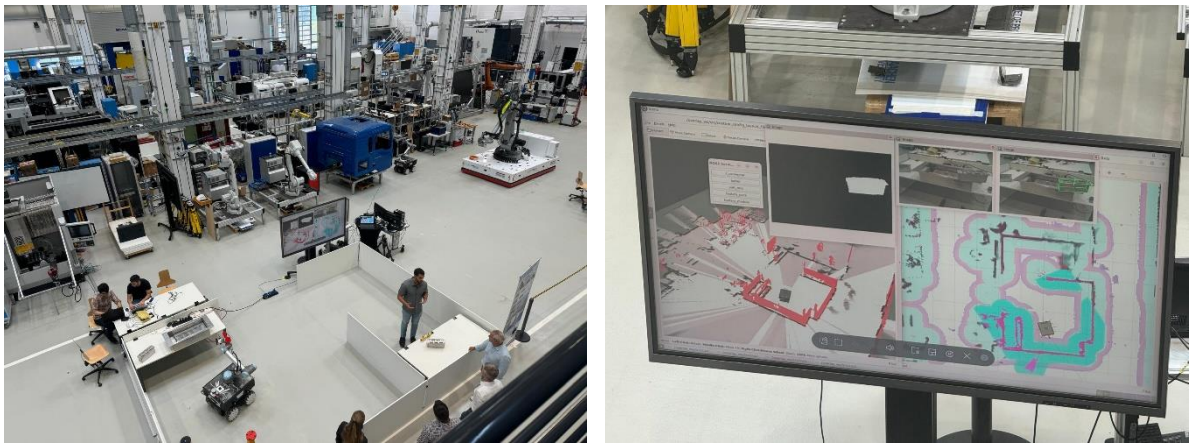

Figure 3.27: Presenting the Edge-controlled Mobile Manipulator.

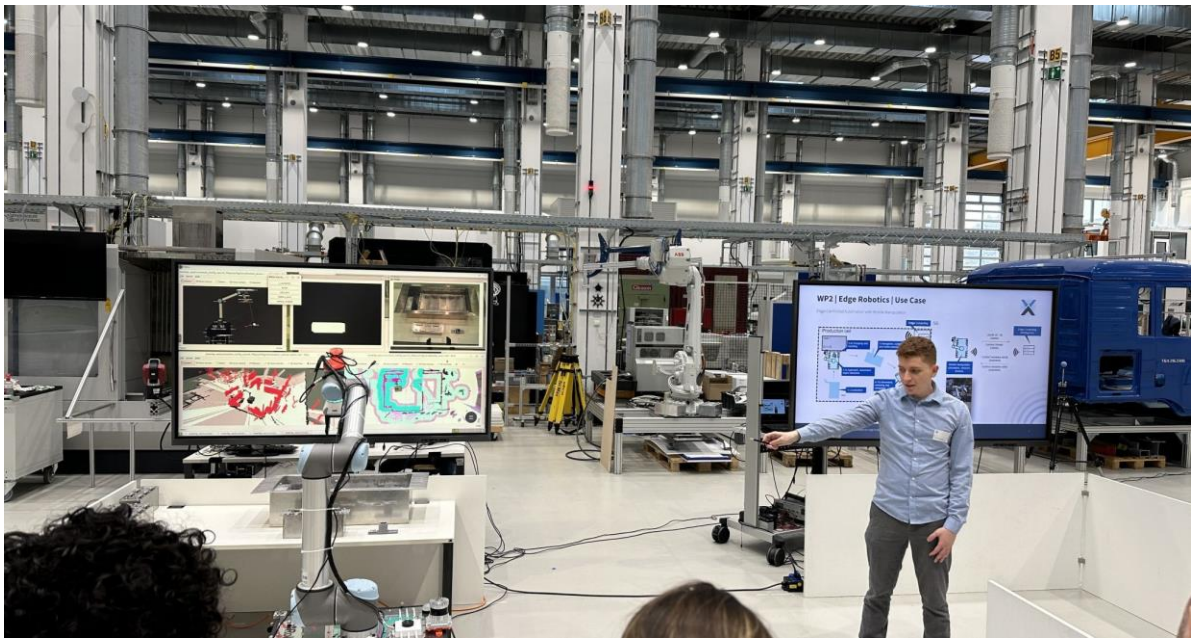

Figure 3.28: Successful presentation of the running Use Case during the TARGET-X Open Day Event 2024.

4 Conclusion

In this deliverable, we have successfully outlined the implementation of the edge-controlled automation use case with mobile manipulation, demonstrating the integration of 5G technology with a comprehensive ROS2-based software stack. The use of advanced technologies, such as NVIDIA's Isaac-Sim for simulation, real-time perception, and motion planning frameworks like MoveIt2, enabled a robust platform for flexible automation. Key achievements include the deployment of a modular and scalable system architecture, the utilization of containerization through Docker, and the seamless coordination of hardware components, including LiDAR and camera sensors.

The project has showcased the benefits of offloading computational tasks to the edge server, optimizing performance and ensuring real-time responsiveness for critical tasks such as localization, navigation, and object manipulation. Through 5G NR-based positioning, the system shows potential in mobile robot localization, complementing traditional sensors like LiDAR. These innovations could potentially reduce the cost of hardware while maintaining high precision and reliability in industrial settings.

However, challenges such as network bandwidth management and uplink capacity remain ongoing areas of focus, as the system's demand for fast and reliable communication continues to grow. Future work will focus on further validation of 5G NR localization, benchmarking performance against other sensor inputs, and refining the CI/CD pipeline to streamline deployment processes.
In terms of the next steps, the validation and benchmarking of the network requirements are expected to be presented in Deliverable D2.5. This will provide deeper insights into how the system performs under different network conditions, further optimizing the communication infrastructure for enhanced performance in future deployments.

In conclusion, this implementation marks a significant step toward the practical application of edge-controlled mobile manipulators in dynamic industrial environments, with 5G technology playing a crucial role in enhancing flexibility, scalability, and real-time control. As the project progresses, the system will continue to be benchmarked and optimized to meet the evolving demands of next-generation manufacturing processes.

5 References

References

[1] Hüttemann, G. *Modellbasierte a priori Bewertung von linienlosen mobilen Montagesystemen: Lehrstuhl für Fertigungsmesstechnik und Qualitätsmanagement / Werkzeugmaschinenlabor WZL der RWTH Aachen*.

[2] TARGET-X Project, WP2 – Manufacturing, 2023. *D2.2 Report on System Design Options and 5G/6G Setup for Edge Robotics*.

[3] Hüttemann, G., Buckhorst, A.F., Schmitt, R.H., 2019. Modelling and Assessing Line-less Mobile Assembly Systems *81*, p. 724.

[4] Buckhorst, A.F., Grahn, L., Schmitt, R.H., 2022. Decentralized Holonic Control System Model for Line-less Mobile Assembly Systems *75*, p. 102301.

[5] van Tran, L., Lin, H.-Y., 2021. Semantic Segmentation and 6DoF Pose Estimation using RGB-D Images and Deep Neural Networks, in *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*, IEEE, p. 1.

[6] Yuan, S., Ge, Z., Yang, L., 2023. Single-Camera Multi-View 6DoF pose estimation for robotic grasping *17*, p. 1136882.

[7] AWK23, Editor, 2023. *AWK'23 – Digital conference proceedings: Conference abstracts*.

[8] Bergs, L., Tréheux, G., Schäper, L., Manasses Pinheiro de Souza, Lucas, Münker, S., Göppert, A., Schmitt, R.H., Editors. *Edge Computing Framework for Enhanced Robotic Adaptivity in Line-Less Mobile Assembly Systems*. Elsevier B.V.

[9] WZL, Laboratory for Machine Tools and Production Engineering of RWTH Aachen. RWTH AACHEN UNIVERSITY Laboratory for Machine Tools and Production Engineering (WZL) of RWTH Aachen University - English. https://www.wzl.rwth-aachen.de/cms/~sijq/wzl/?lidx=1. Accessed 11 December 2023.

[10] TARGET X. TARGET-X. https://target-x.eu/. Accessed 11 December 2023.

[11] EuCNC, 10 April 2024. European Conference on Networks and Communications (EuCNC) & 6G Summit. https://www.eucnc.eu/. Accessed 14 October 2024.

[12] Groshev, M., Baldoni, G., Cominardi, L., La Oliva, A. de *et al.*, 2023. Edge robotics: are we ready? an experimental evaluation of current vision and future directions *9*, p. 166.

[13] NVIDIA Technical Blog. SoftBank Solves Key Mobile Edge Computing Challenges Using NVIDIA Maxine | NVIDIA Technical Blog. https://developer.nvidia.com/blog/softbank-solves-key-mobile-edge-computing-challenges-using-nvidia-maxine/. Accessed 31 October 2024.

[14] Kehl, P., Ansari, J., Jafari, M.H., Becker, P. *et al.*, 2022. Prototype of 5G Integrated with TSN for Edge-Controlled Mobile Robotics *11*, p. 1666.

[15] Git - What is Git? https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F. Accessed 31 October 2024.

[16] 10 May 2022. Docker: Accelerated Container Application Development. https://www.docker.com/. Accessed 31 October 2024.

[17] Kumar, M., 3 April 2022. Docker Architecture, Life Cycle of Docker Containers and Data Management. https://dev.to/docker/docker-architecture-life-cycle-of-docker-containers-and-data-management-1a9c. Accessed 18 October 2024.

[18] Maruyama, Y., Kato, S., Azumi, T., 2016. Exploring the performance of ROS2, in *Proceedings of the 13th International Conference on Embedded Software*, Association for Computing Machinery, New York, NY, USA.

[19] Bergs, L. WZL-IQS-S / Docker and ROS Tools / ROS2 / State Machine Angular · GitLab, 2024. https://git-ce.rwth-aachen.de/wzl-mq-ms/docker-ros/ros2/state-machine-angular. Accessed 2024.

[20] Cartographer. cartographer-project/cartographer: Cartographer is a system that provides real-time simultaneous localization and mapping (SLAM) in 2D and 3D across multiple platforms and sensor configurations, 2024. https://github.com/cartographer-project/cartographer. Accessed 2024.

[21] Nav. Nav2 -- Nav2 1.0.0 documentation, 2024. https://docs.nav2.org/. Accessed 2024.

[22] NVlabs. NVlabs/FoundationPose: [CVPR 2024 Highlight] FoundationPose: Unified 6D Pose Estimation and Tracking of Novel Objects, 2024. Accessed 2024.

[23] Moveit. moveit/moveit2:robot: MoveIt for ROS 2, 2024. https://github.com/moveit/moveit2. Accessed 2024.

[24] OUSTER. ouster-lidar/ouster-ros: Official ROS drivers for Ouster sensors (OS0, OS1, OS2, OSDome), 24. https://github.com/ouster-lidar/ouster-ros. Accessed 2024.

[25] point_cloud_transport. ros-perception/point_cloud_transport: Point Cloud Compression for ROS, 2024. https://github.com/ros-perception/point_cloud_transport. Accessed 2024.

[26] IntelRealSense. IntelRealSense/realsense-ros: ROS Wrapper for Intel(R) RealSense(TM) Cameras, 2024. https://github.com/IntelRealSense/realsense-ros. Accessed 2024.

[27] image-pipeline. ros-perception/image_pipeline: An image processing pipeline for ROS, 2024. https://github.com/ros-perception/image_pipeline. Accessed 2024.

[28] Rviz. ros2/rviz: ROS 3D Robot Visualizer, 2024. https://github.com/ros2/rviz. Accessed 2024.

[29] VNC NVIDIA Docs. VNC | NVIDIA Docs, 2023. https://developer.nvidia.com/docs/drive/drive-os/6.0.8.1/public/drive-os-linux-sdk/common/topics/sys_components/vnc.html. Accessed 2024.

[30] Document. About Different ROS 2 DDS/RTPS Vendors -- ROS 2 Documentation: Foxy Documentation, 2024. https://docs.ros.org/en/foxy/Concepts/About-Different-Middleware-Vendors.html. Accessed 11 July 2024.