Deliverable D2.2

5G-BASED TESTBED FOR TRIALS WITH INDUSTRIAL ROBOTICS

The 5G-SMART project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 857008.

Deliverable D2.2 5G-Based Testbed for Trials with Industrial Robotics

Grant agreement number:	857008
Project title:	5G Smart Manufacturing
Project acronym:	5G-SMART
Project website:	www.5gsmart.eu
Programme:	H2020-ICT-2018-3

Deliverable type:	Report (R)
Deliverable reference number:	D7
Contributing workpackages:	WP2
Dissemination level:	Public
Due date:	January 31, 2022
Actual submission date:	January 31, 2022
Responsible organization:	ABB
Editor(s):	Ognjen Dobrijevic (ABB)
Version number:	V1.0
Status:	Final
Short abstract:	This deliverable reports on realizing the 5G-based testbed for industrial robotics at the Kista trial site. It provides design and implementation details for the three use cases to be trialed, namely around vision-assisted robot collaboration, real-time human-robot interaction, and advanced visualization of robot status. The deliverable also outlines the 5G network solution on the trial site which is used for running the industrial robotics use cases.
Keywords:	Lead-through robot motion teaching, mobile robot navigation, machine vision, mobile robot docking, robot pick and place, augmented reality visualization, 5G system, mmWave radio, edge cloud

Contributor(s):	Aftab X. Ahmad (ABB) Ognjen Dobrijevic (ABB) Pietro Falco (ABB) Polychronis Kontaxakis (ABB) Jon Tjerngren (ABB) Vicknesan Ayadurai (Ericsson) Peter de Bruin (Ericsson) Igor Ruffini (Ericsson) Karam Takieddine (Ericsson)

Disclaimer

Executive summary

This document reports on realizing the 5G-based testbed for industrial robotics at the Kista trial site. It provides design and implementation details for the three use cases to be trialed: vision-assisted robot collaboration, real-time human-robot interaction, and advanced visualization of robot status. Functional specifics of the key testbed components are presented, focusing on the needed aspects of robot control as well as on data-exchange interactions among the related functional blocks. The most crucial implementation parts of software, which prototype the robot control and other auxiliary functions, are described. The deliverable also outlines the 5G network solution on the trial site used to run the industrial robotics use cases. The deployment of the 5G network is summarized, including also how its mmWave radio connectivity is integrated with the implementation of the use cases.

Contents

1 Introduction

As with many other industry domains, the manufacturing sector is going through a fundamental change by introducing and exploiting digital technologies. One of the key determinants of the ongoing transformation is "smartness". By digitizing manufacturing processes, new means of interactions between human workers, machines, products, and other factory-floor assets are facilitated, maximizing productivity, decreasing overall cost, and improving safety [CCO19] [ER20]. Industrial robots will continue to have the main role in such a manufacturing ecosystem. Since responsibilities of human workers are envisaged to move more towards supervision of production processes, industrial robots and other machines will need to coordinate more autonomously, but also to allow improved ways for the humans to access them [EGG+20]. Other requirements on the future manufacturing include a greater flexibility to deploy, commission, and re-configure production processes.

Wireless connectivity is seen as one of the primary enablers for digitalization of the manufacturing sector. Besides facilitating the production flexibility and new ways of human-robot interactions, wireless communication is also needed to support mobility on the factory floor. This pertains both to human workers equipped with advanced user interfaces, such as augmented reality (AR)-based headsets, and to autonomous mobile robots, which are used, e.g., to transport production materials and goods across the factory floor. Another enabler for digitalization is the utilization of cloud and edge paradigms, which offer service deployment flexibility, a scalable allocation of computing resources, and resiliency to faults. In the context of industrial robotics, for both stationary and mobile robots, the edge computing paradigm has significant potential. Offloading parts of control from robot units, such as for motion planning, could simplify their hardware configuration and overall footprint on the factory floor. All these incentives, coupled with communication requirements on high availability and bounded latency for wireless-based robot control, make 5G a leading wireless-technology candidate [SL21].

1.1 Objective of the document

The document reports on the realization of the 5G-based testbed for industrial robotics at the Kista trial site. It builds upon the previously published 5G-SMART Deliverables D1.1 [5GS20-D11] and D2.1 [5GS20-D21]. Deliverable D1.1 describes the "storyline" for the three use cases from the industrial robotics domain, trialed at the Kista site, as well as their general functional and non-functional requirements. The three use cases are considered representative examples of smart manufacturing, since they encompass an autonomous, vision-assisted robot collaboration, real-time human-robot interaction for robot commissioning, and AR-based access to industrial robots and their operational status. Deliverable D2.1, on the other hand, specifies the overall design for the 5G-based testbed, also defining main robotics-related functional blocks and their interactions.

This document presents the main design and implementation specifics for the industrial robotics use cases. A functional view on the developed testbed components is given, focusing on the needed aspects of robot control and supervision, and data-exchange interactions among the related functional blocks. Developed software, which prototypes the robot control and supervision as well as other auxiliary functions, is described, also documenting how it is used in the 5G-based testbed. The deliverable also outlines the 5G network solution on the trial site used to run the use cases implementation. The deployment of the 5G network is summarized, including also how its mmWave radio connectivity is integrated into robotics-related equipment.

1.2 Structure of the document

The rest of this document is structured as follows. Section 2 gives a system-level overview of the 5G-based testbed for industrial robotics and focuses on explaining how the main robot control and supervision components interact to achieve features of the three use cases. Functional design requirements on the prototyped software are outlined in Section 3, while implementation and usage of the software are reported in Section 4. In Section 5, the 5G solution for the Kista trial site is outlined, followed by conclusions of the work in Section 6.

2 System-level overview

This section presents a system-level overview of the 5G-based testbed for industrial robotics, focusing on a functional perspective and interactions among the key robot control and supervision components to realize features of the use cases to be trialed.

The three use cases are described in 5G-SMART Deliverable D1.1 [5GS20-D11] and here we provide a brief summary of them for completeness of this document:

- *Use case 1 on 5G-connected robots and their collaboration*. The use case envisages a mobile robot transferring an object, navigating the mobile robot between workstations with stationary robots, and having the stationary robots grasp and move the object. That represents a smart factory scenario of autonomously transferring materials between production lines, based on a collaboration between different types of robots. Two distinctive features of the use case instantiation are machine vision assisted docking of the mobile robot to a precise location next to the robot workstation and vision-assisted execution of the object pick-and-place operation by each of the stationary robots.
- *Use case 2 on vision-supported real-time human-robot interaction.* This use case is instantiated by a human worker (e.g., commissioning engineer) programming a stationary robot to autonomously execute the object pick-and-place operation through the means of demonstration. Instead of writing a robot program, the worker mimics grasping and moving the object through application on a mobile device (e.g., smartphone), which the stationary robot then executes. Such a scenario illustrates the potential of advanced means of human-robot interaction on the factory floor. Another distinctive feature of the use case is machine vision support for detecting and tracking objects.
- *Use case 3 on advanced visualization for the factory floor*. The use case is characterized by employing AR-based means to remotely manipulate robot motion and to visualize operational robot information in an efficient way. It also illustrates how novel technologies can be exploited to create new ways for human workers to access factory-floor machinery. Like in the other two use cases, machine vision support is used to detect and distinguish between different objects.

2.1 Architecture

Figure 1 shows an overview of the reference testbed setup at the Kista trial site for the three use cases. Such a setup is inspired by the vision of 5G-empowered industrial robotics, which is then prototyped by developing the associated use cases for the 5G-based testbed in Kista. The testbed setup incorporates the following hardware components, related to the use cases:

- two YuMi© stationary robots [YuMi] to realize autonomous pick and place of an object,
- a mobile robot platform [Ridgeback] with Light Detection and Ranging (LiDAR) sensors, used for implementing functional aspects of material transfer handling,
- a smartphone to demonstrate new ways of robot programming based on human-robot interaction,
- video cameras [Kinect] [RealSense] to implement machine vision features of object detection and localization, and

- an AR headset [MagicLeap] to demonstrate advanced features of visualization for the factory floor.

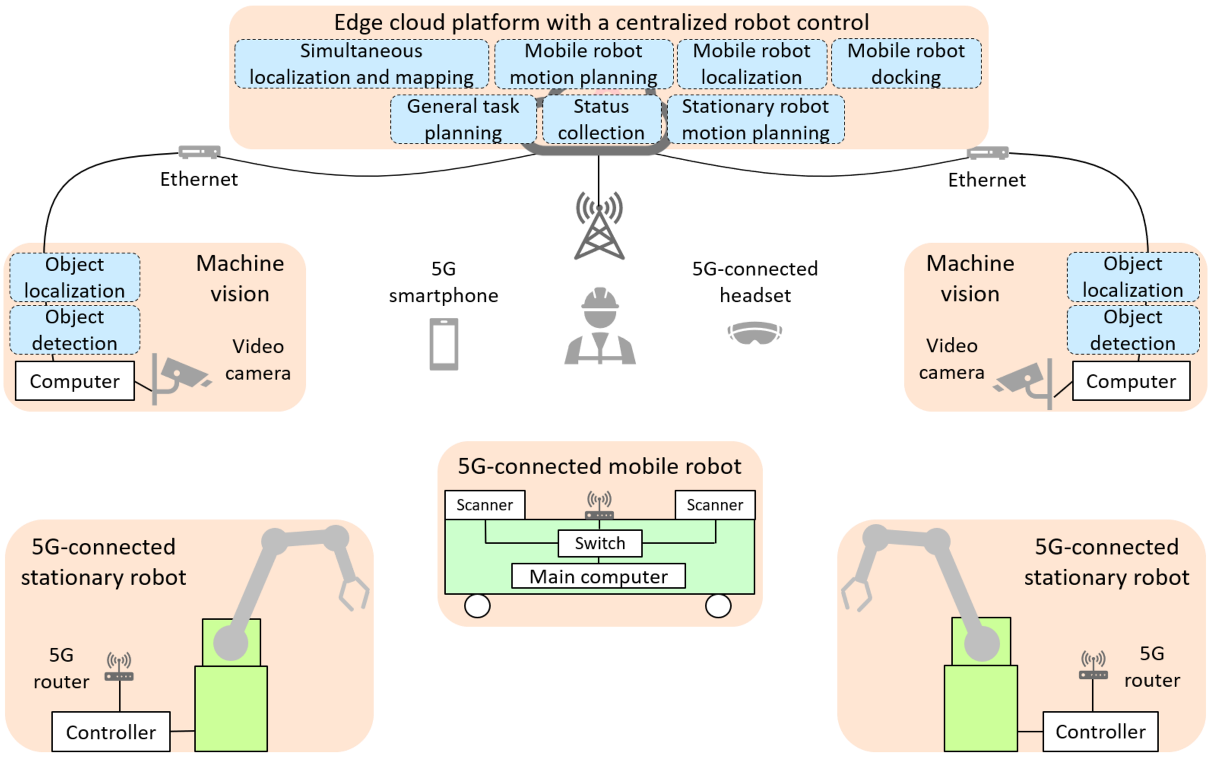


Figure 1: Reference industrial robotics setup for the 5G-based testbed at the Kista trial site

At the heart of the 5G-empowered industrial robotics vision there is “off-loading” of a part of the robot control and supervision functionality, such as Simultaneous Localization and Mapping (SLAM), general task planning and robot motion planning, from specialized, on-board robot hardware into edge cloud platforms. Having in mind expectations that 5G will provide a wireless-based communication service with bounded latency and high availability, such a design approach could, for instance, increase the deployment flexibility but also decrease service commissioning time. Another distinctive feature of the setup is a machine vision system with video cameras that “overlooks” the testbed area and is used to assist autonomous mobile robot docking as well as execution of stationary robot pick-and-place. More details about the overall testbed design can be found in 5G-SMART Deliverable D2.1 [5GS20-D21].

2.2 General workflow

The rest of this section illustrates the main functions of the testbed components as well as their interactions to realize the main features of the three use cases. Our intention with this is to give a high-level overview and understanding of the developed components, before dwelling into details of their design and implementation. The focus is on a use-case level, i.e., explanations on 5G-related hardware and software are purposely omitted.

For illustrative purposes, Figure 2 shows a workflow that combines the operation of the testbed components in a common demo. That workflow consists of two, independent sequences of operations: mobile robot transfer of an object between production workstations with stationary robots and inspection of stationary robot status by human worker equipped with AR headset (noted in the figure with the blue and green coloring, respectively). The two sequences of operations relate to the features of *Use case 1* and *Use case 3*, respectively. Prior to running such a demo, the features of *Use case 2* around human-robot interaction are utilized by human worker to teach stationary robots how to execute pick-and-place operation.

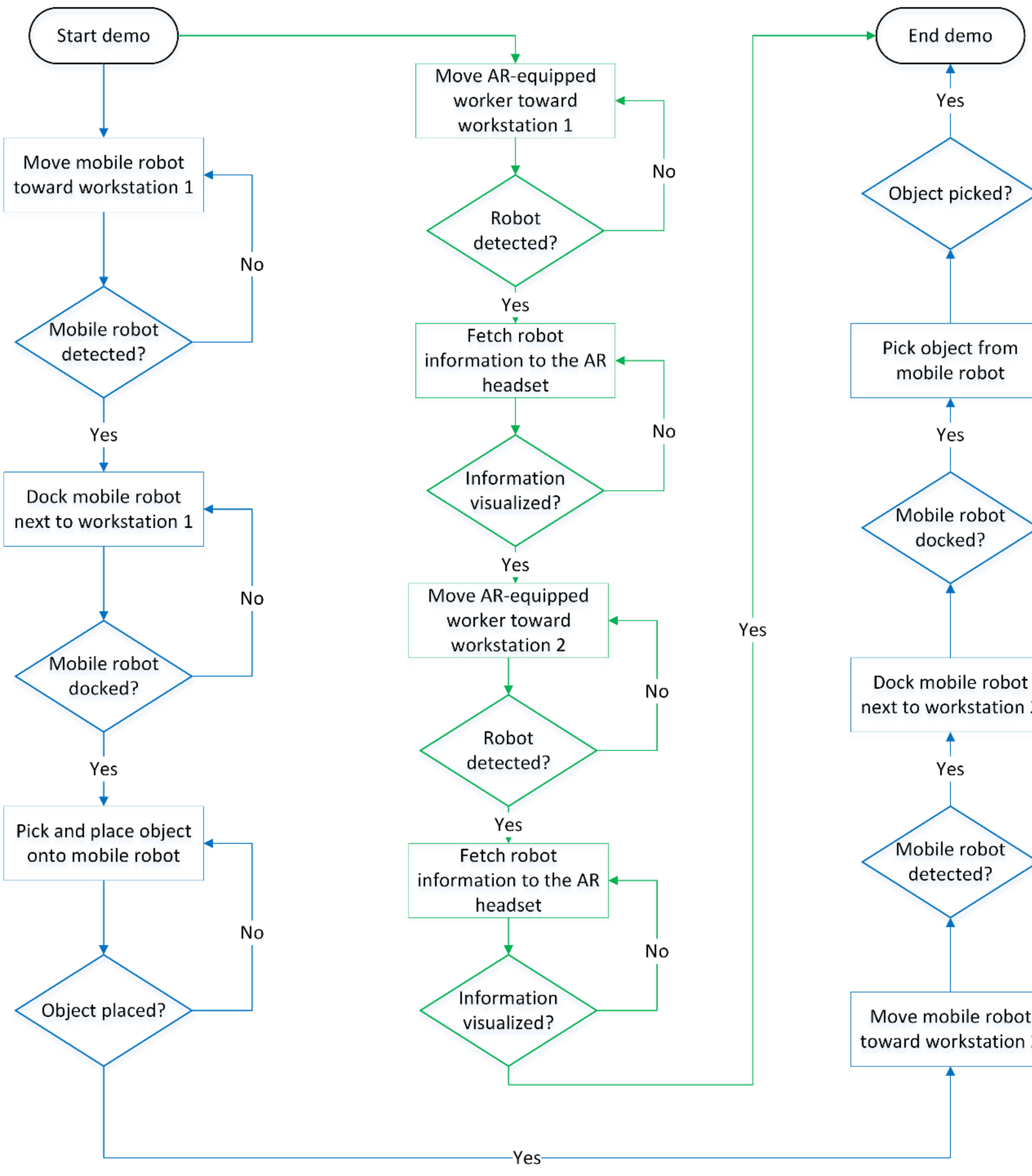

Start demo
Move mobile robot toward workstation 1
Mobile robot detected?
No
Yes
Dock mobile robot next to workstation 1
Mobile robot docked?
No
Yes
Pick and place object onto mobile robot
Object placed?
No
Yes
Move AR-equipped worker toward workstation 1
Robot detected?
No
Yes
Fetch robot information to the AR headset
Information visualized?
No
Yes
Move AR-equipped worker toward workstation 2
Robot detected?
No
Yes
Fetch robot information to the AR headset
Information visualized?
No
Yes
End demo
Yes
Object picked?
No
Pick object from mobile robot
Yes
Mobile robot docked?
No
Dock mobile robot next to workstation 2
Yes
Mobile robot detected?
No
Move mobile robot toward workstation 2

Figure 2 A common demo workflow for the testbed components

In the object transfer part of the workflow, mobile robot is first tasked to move toward a robot workstation which is pre-defined in demo configuration. The use case feature of planning and executing motion for the mobile robot is referred to as mobile robot navigation and is described in more detail in subsection 2.2.1. After reaching the robot workstation, machine vision system is tasked to detect the mobile robot, which in turn triggers moving the mobile robot to a precise location next to the workstation. That use case feature of planning and controlling motion of the mobile robot, which is assisted by the machine vision, is referred to as mobile robot docking (subsection 2.2.2). After docking is finished, continuation of the typical demo workflow assumes stationary robot picking an object from its holder and placing it onto the mobile robot. That operation is also assisted by the machine vision system and is presented in more detail in the pick and place subsection (2.2.3). A very similar sequence of operations is performed when the mobile robot is tasked to transfer the received object to the other robot workstation.

In parallel, a human worker equipped with AR headset is tasked to inspect stationary robots (Figure 2, sequence noted in green). The human worker is moving toward any of the robot workstations. As soon as her/his AR headset detects a stationary robot, the latest information on robot status and health is fetched and then visualized by using advanced graphical elements (e.g., panels). A functional view on the AR-based visualization is outlined in subsection 2.2.4.

2.2.1 Mobile robot navigation

Figure 3 gives an overview of the interaction between different robot control functions to achieve mobile robot navigation. This figure illustrates a typical workflow that is executed to move mobile robot to a targeted position inside the physical environment, for which a virtual map is created beforehand in the map creation process. Key types of data being exchanged between the functions are pointed out, also noting which of the data is being transmitted over a 5G radio link. In addition, it is noted which of the functions runs in edge cloud.

The robot control functions employed for mobile robot navigation have the following responsibilities:

- *SLAM* – creates a virtual map of the physical environment through which mobile robot is to be navigated, containing obstacles learned of during the map creation process,
- *General task planning* – triggers the overall process of navigating the mobile robot within the physical environment,
- *Mobile robot motion planning* – determines path for the mobile robot to take to reach the targeted position, also avoiding obstacles which may appear on the robot's way, and
- *Mobile robot localization* – tracks movement of mobile robot relative to the virtual map.

The workflow in Figure 3 is depicted to start with the initial step of loading the produced virtual map of the mobile robot's environment into *Mobile robot motion planning* and *Mobile robot localization*. While that step is explicitly noted in the workflow, it is assumed to be a part of the overall testbed commissioning phase and needs to be in place for other operational features of the use cases as well. *Demo configurator* is an auxiliary entity that provides input regarding configurable parameters of a desired demo, for instance which stationary robot should mobile robot be first moved to. The latter aspect is denoted with targeted position and can also be considered a part of the commissioning phase.

After the demo is started, *General task planning* initiates motion planning for the mobile robot by conveying information on its targeted position (*2. Initiate motion planning* in Figure 3). *Mobile robot motion planning* first calculates a shortest path from the robot's current position to the targeted one, followed by instructing the mobile robot on how to move in the physical environment and reach the target. The latter instructions are continuously transferred to the mobile robot in the form of motion velocities (*3. Send motion velocities*). As the mobile robot moves through physical space, it periodically "scans" the environment with on-board sensors to detect possible objects and distances to them. These are conveyed from the mobile robot as "point clouds" (*3. Send point clouds and odometry data*). In addition to the point clouds, the mobile robot approximates a change of its position over time and forwards that as "odometry data". Both latter types of data are delivered to *Mobile robot localization*, which then estimates mobile robot position and orientation in the environment for *Mobile robot motion planning* (*3. Send robot position and orientation*). The interactions between *Mobile robot motion planning*, *Mobile robot localization*, and the mobile robot may be seen as a continuously running "control loop" (though not in the classical, control-theory sense).

Mobile robot motion planning and *Mobile robot localization* are navigating the mobile robot until it reaches the targeted position (*4. Mobile robot reached targeted position*). *Mobile robot motion planning* also encompasses functionalities for the mobile robot to avoid obstacles on its path. It is worth mentioning that the machine vision support is not used for mobile robot navigation and that the navigation relies on LiDAR sensors on-board mobile robot platform to perceive the surrounding environment.

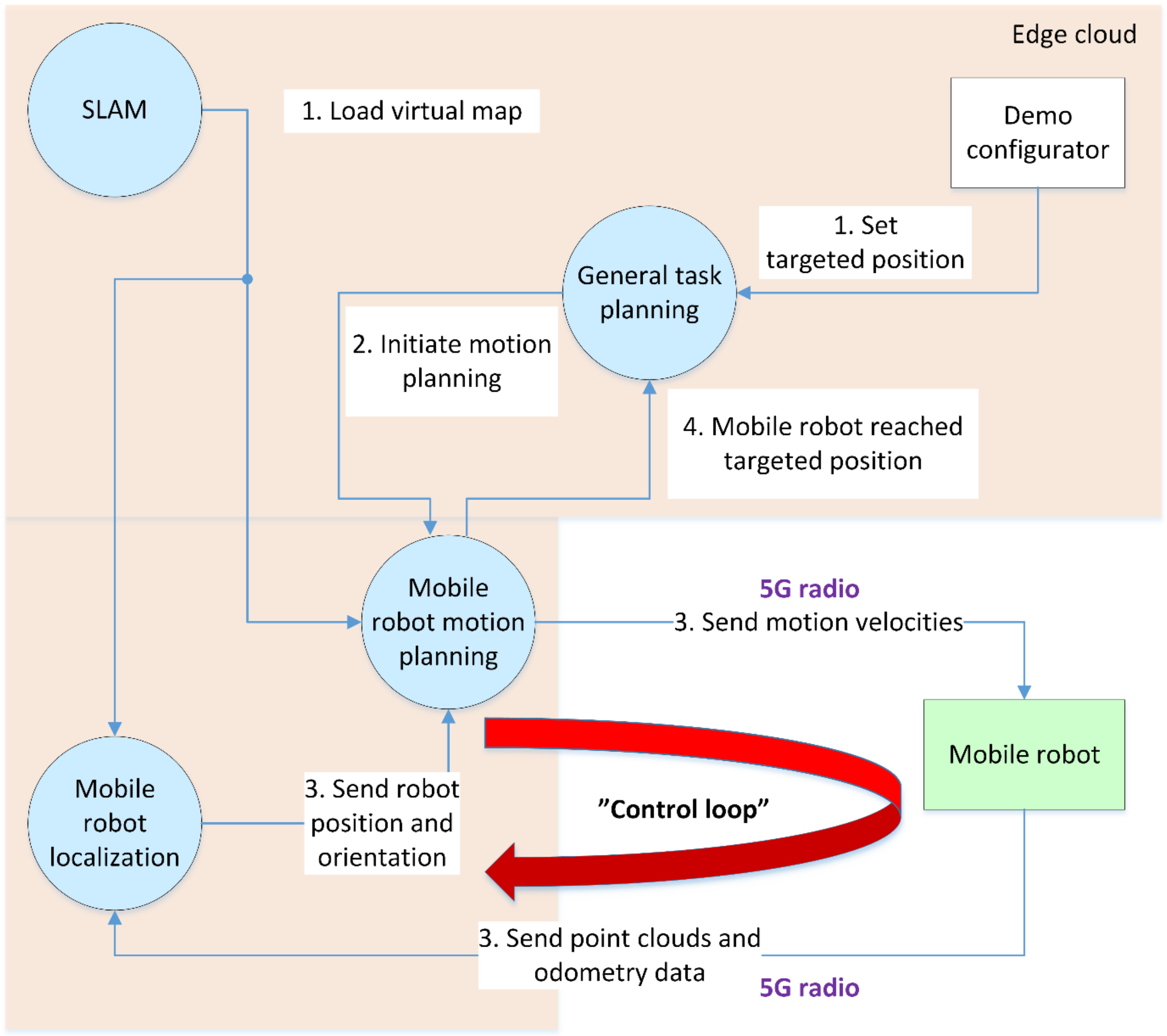

Edge cloud
SLAM
1. Load virtual map
Demo configurator
1. Set targeted position
General task planning
2. Initiate motion planning
4. Mobile robot reached targeted position
Mobile robot motion planning
5G radio
3. Send motion velocities
Mobile robot
"Control loop"
Mobile robot localization
3. Send robot position and orientation
3. Send point clouds and odometry data
5G radio

Figure 3 Interaction of robotics-related functions for mobile robot navigation

2.2.2 Mobile robot docking

Figure 4 provides an overview of the interaction between robot control functions to realize mobile robot docking. The figure shows a typical workflow where *Mobile robot motion planning* navigates mobile robot to a targeted position and then *Mobile robot docking* moves the robot to a more precise position, e.g., next to a stationary robot for pick and place operation. The main flow of data between the functions is depicted, to illustrate general operation. *Object detection* and *Object localization* are highlighted with another color to emphasize that they are not collocated with other robot control functions in edge cloud, but rather run in a separate computing node which is directly attached to video cameras. In a generalized deployment scenario, these two machine vision functions could be executed in the same edge cloud.

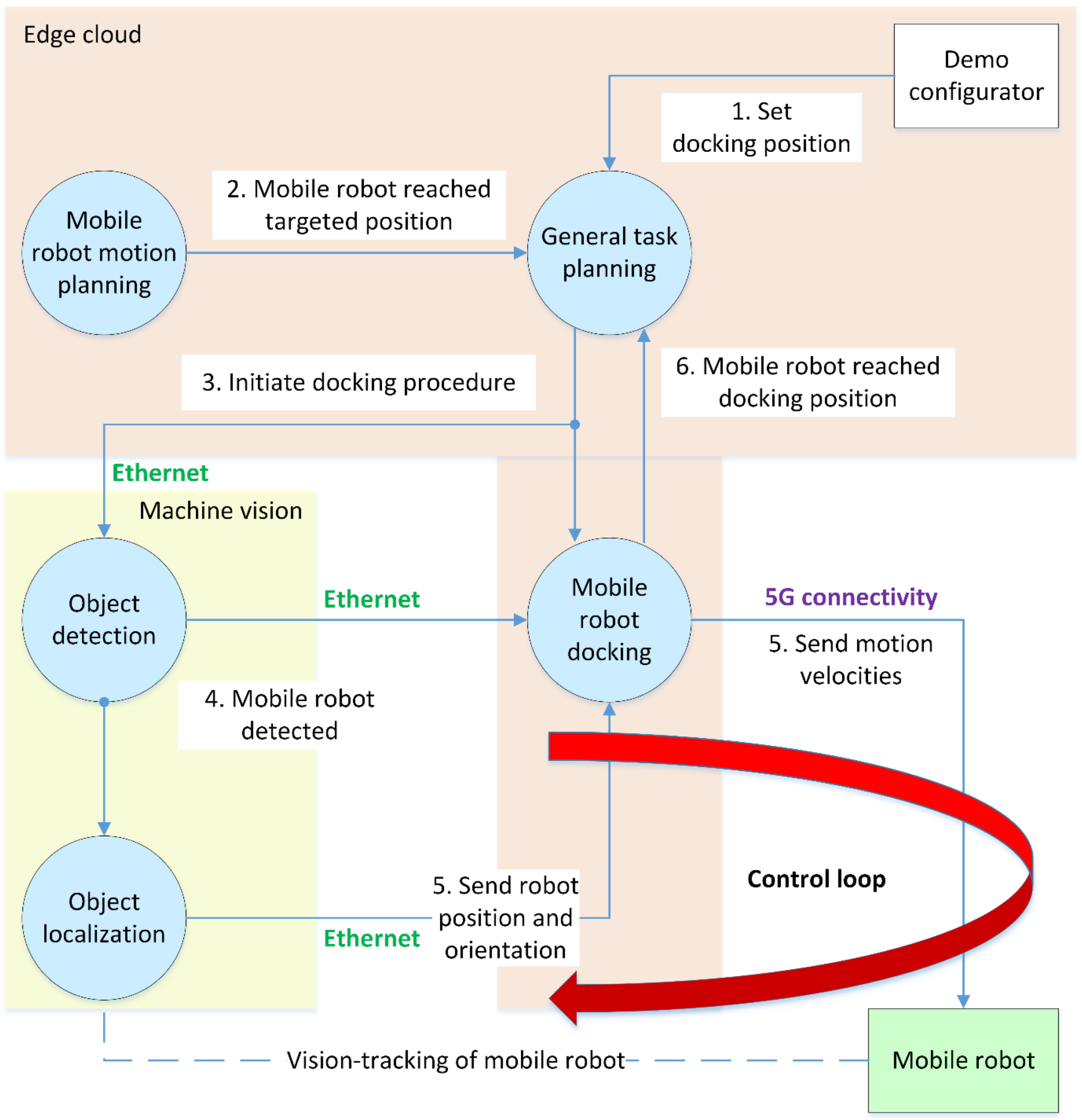

Edge cloud
Demo configurator
1. Set docking position
Mobile robot motion planning
2. Mobile robot reached targeted position
General task planning
3. Initiate docking procedure
6. Mobile robot reached docking position
Ethernet
Machine vision
Object detection
Ethernet
Mobile robot docking
5G connectivity
5. Send motion velocities
4. Mobile robot detected
Control loop
Object localization
5. Send robot position and orientation
Ethernet
Vision-tracking of mobile robot
Mobile robot

Figure 4 Interaction of robotics-related functions for mobile robot docking

Additional robot control functions, which support realization of this use cases' feature, encompass the roles as follows:

- *Object detection* – recognizes mobile robot, among different possible objects, via a marker that is placed on it,
- *Object localization* – tracks mobile robot via its marker, and
- *Mobile robot docking* – moves mobile robot as close as possible to (destined) docking position.

The workflow in Figure 4 is shown to start with the initial step of defining the destined docking position by *Demo configurator*. Similar to mobile robot navigation, that step is considered a part of the overall testbed commissioning. Since a typical demo workflow for the summarized use cases on industrial robotics involves mobile robot docking to follow mobile robot navigation, the figure further depicts *Mobile robot motion planning* notifying *General task planning* of mobile robot reaching targeted position. Such a position can also be pre-defined by *Demo configurator*, to indicate when a more precise docking procedure should be invoked.

After *General task planning* triggers the docking procedure (*3. Initiate docking procedure*), *Object detection* is tasked to recognize the mobile robot by utilizing the video camera subsystem. After that is achieved (*4. Mobile robot detected*), *Object localization* starts to continuously track the mobile robot relying on the same video camera subsystem and the robot's marker (noted with the dashed line *Vision-tracking of mobile robot* in Figure 4). The goal is to estimate the mobile robot's position and orientation in the environment. That data is periodically conveyed to *Mobile robot docking* (*5. Send robot position and orientation*), which then compares the actual position and orientation with the targeted ones. *Mobile robot docking* then tries to minimize the difference between the actual and targeted values by moving mobile robot. The latter control is achieved by instructing the mobile robot on how to change its position in the environment (*5. Send motion velocities*).

When the control difference falls in value below a pre-defined threshold, then the docking procedure is terminated. The interactions between *Mobile robot docking*, *Object localization*, and mobile robot are an example of closed-loop control.

2.2.3 Stationary robot pick and place operation

Figure 5 shows the interaction between the robot control functions to execute the operation of pick and place (P&P) by each of the stationary robots. The figure shows a typical workflow of *Mobile robot docking* moving the mobile robot to a precise position next to the targeted stationary robot, followed by P&P operation controlled by *Stationary robot motion planning*. The main interactions between the functions are depicted, also including machine vision support by *Object detection* and *Object localization*.

Like workflows presented in previous subsections, Figure 5 starts with a commissioning step. If there are multiple objects which can be selected for the P&P operation in a demo, then *Demo configurator* defines a picking scenario. Since a common demo workflow for the three use cases on industrial robotics encompasses P&P execution to follow after mobile robot docking, the workflow shows *Mobile robot docking* notifying *General task planning* of mobile robot reaching the docking position.

General task planning invokes the P&P operation (*3. Initiate pick and place (P&P) operation*), first causing *Object detection* to recognize the object to be picked up by the targeted stationary robot and placed on the docked mobile robot. After that (*4. P&P object detected*), *Object localization* periodically tracks the P&P object (noted with the dashed line *Vision-tracking of P&P object*) and estimates its position and orientation in the physical environment. The position and orientation are periodically transferred to *Stationary robot motion planning* (*5. Send position and orientation of P&P object*), which then compares them with the position and orientation of the mobile robot. The latter comparison is done to determine the motion that the stationary robot needs to perform to place the P&P object on the mobile robot (*5. Send motion positions and speeds*).

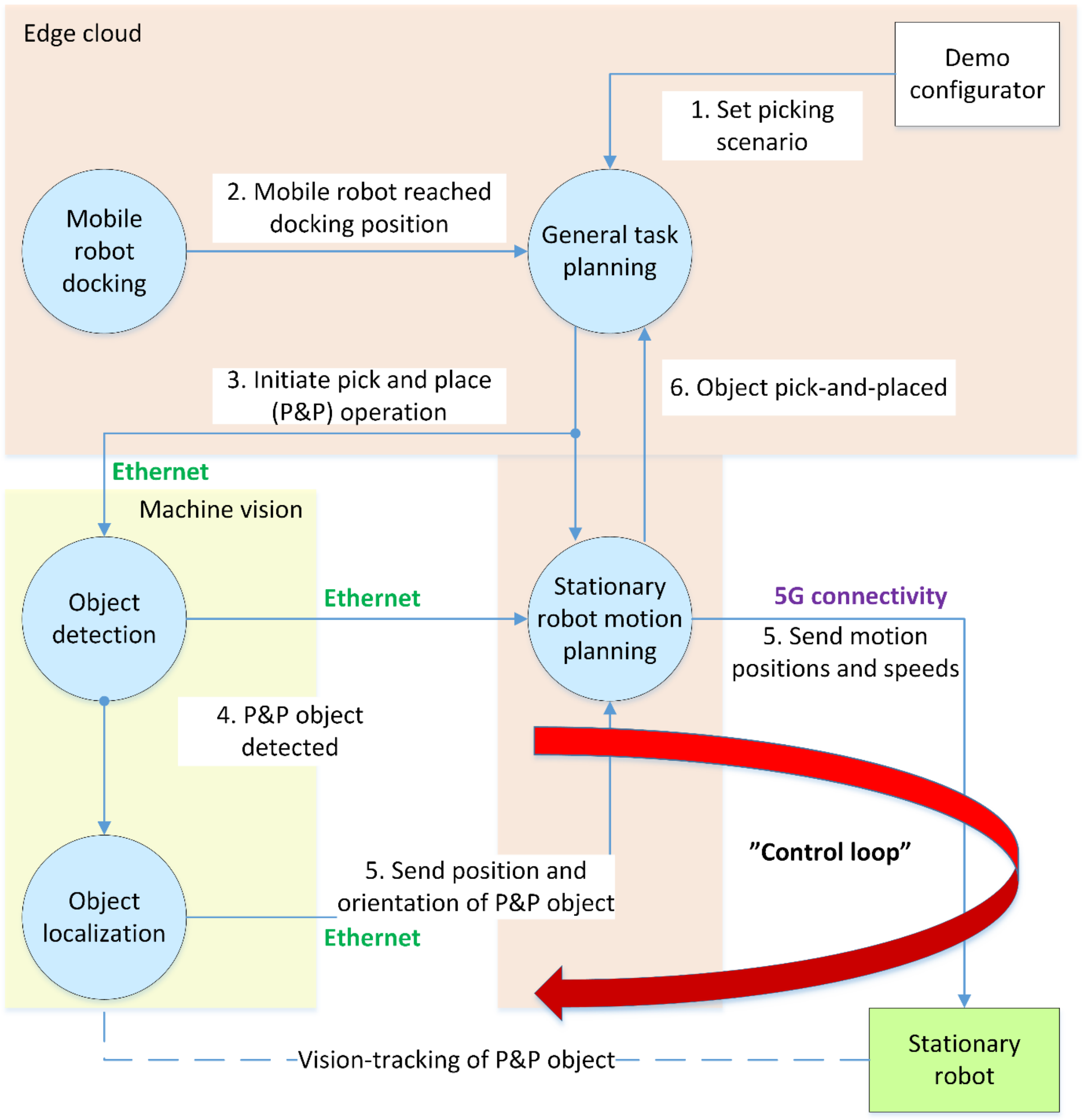

Edge cloud
Demo configurator
1. Set picking scenario
Mobile robot docking
2. Mobile robot reached docking position
General task planning
3. Initiate pick and place (P&P) operation
6. Object pick-and-placed
Ethernet
Machine vision
Object detection
Ethernet
Stationary robot motion planning
5G connectivity
5. Send motion positions and speeds
4. P&P object detected
"Control loop"
Object localization
5. Send position and orientation of P&P object
Ethernet
Vision-tracking of P&P object
Stationary robot

Figure 5 Interaction of robotics-related functions for pick and place execution

Similar to mobile robot navigation, the interactions between *Stationary robot motion planning*, *Object localization*, and stationary robot may be seen as a continuously running a control loop.

2.2.4 AR visualization of robot status

Figure 6 shows the interaction between robotics-related functions to retrieve status from stationary robot for the purposes of the AR visualization.

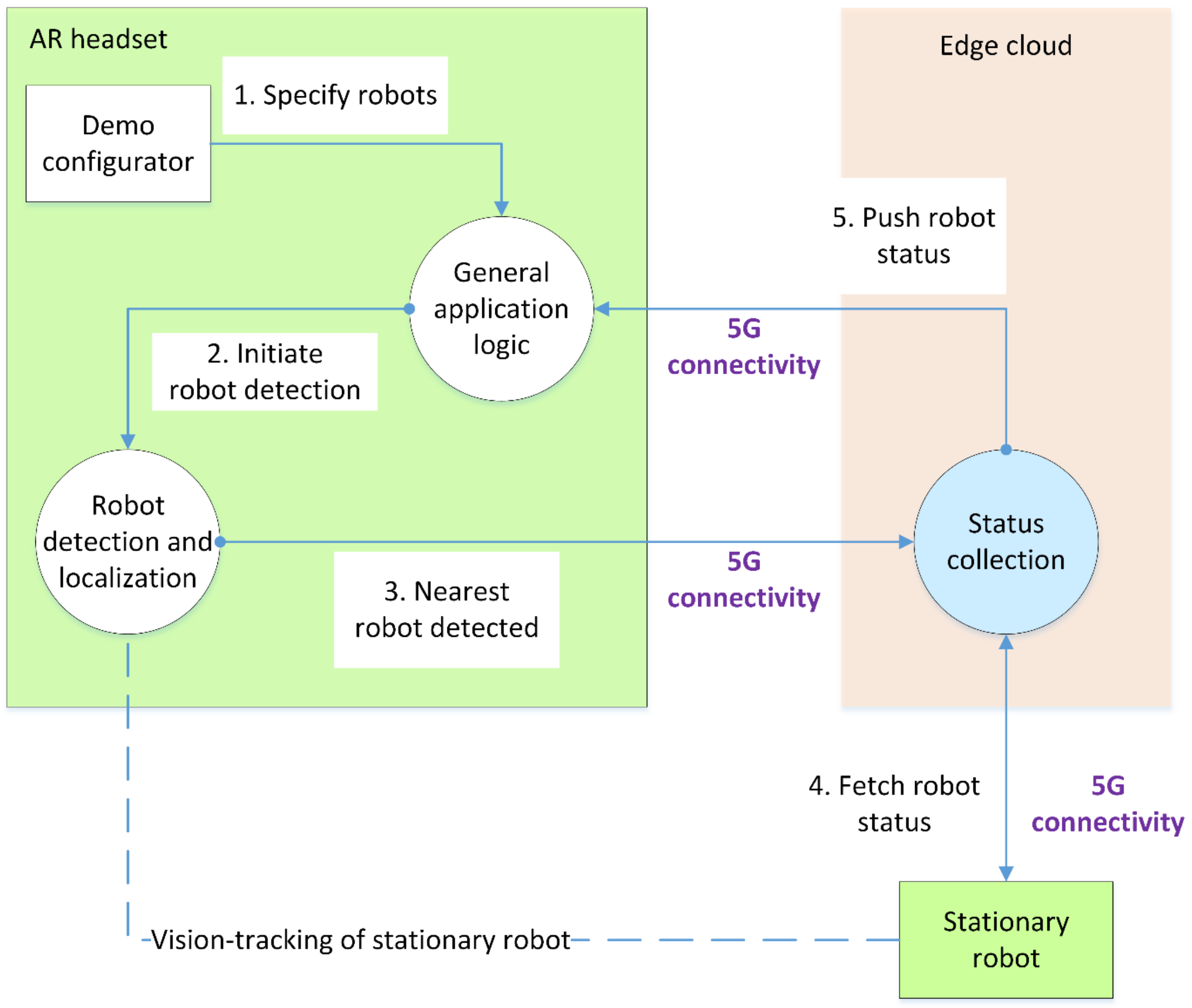


Figure 6 Interaction of robotics-related functions for AR visualization

Like workflows presented in previous subsections, Figure 6 starts with a commissioning step. If there are multiple robots which can be selected for the status collection operation in a demo, then *Demo configurator* defines which of them will be used in the current scenario and conveys that information to *General application logic*.

The workflow shows the following robotics-related functions, which support realization of this use case's features:

- *General application logic* – serves as the core function of the AR visualization application and coordinates other related functions.
- *Robot detection and localization* – The workflow starts when *General application logic* triggers this function to detect stationary robots (*2. Initiate robot detection*), by reading a unique marker tag which is assigned to and fixed onto each of the robots. In a general case, more than one such robot can be detected simultaneously. The robot that is nearest to human worker wearing the AR headset is selected (*3. Nearest robot detected*), by taking into

account the relative positions of all detected robots to the AR headset and then calculating which of them is the closest.

- *Status collection* – The previously determined nearest stationary robot has its status and health data fetched (*4. Fetch robot status*) and then conveyed to the AR headset (*5. Push robot status*) to be displayed to the AR user. After that trigger, the robot status is periodically updated.

3 Functional design of key software components and prototypes

This section summarizes functional requirements and, where needed, a high-level design of the main software components that are prototyped for the industrial robotics use cases.

3.1 Lead-through teaching of robot pick and place

The lead-through teaching describes the jogging (or manipulation) and programming of an industrial robot in a contactless way, by using a mobile device (e.g., smartphone). The mobile device for such a purpose needs to comprise an inertial measurement unit (IMU), combining accelerometer and gyroscope, and a red, green, and blue (RGB) sensor. In essence, the arm of the stationary YuMi© robot is equipped with a tag, which can be detected by the mobile device's camera. The mobile device can then track the tag and use it to estimate the current pose of the robot's arm, i.e., of its robot gripper for picking an object.

A sensor fusion algorithm is required for the mobile device to estimate the current pose of the robot gripper, which is sent to the robot controller as a Cartesian reference, enabling the human teacher to jog the robot arm in different configurations. By moving the mobile device in physical space, the teacher mimics motion she/he wants the robot arm to perform and also continuously estimates gripper's pose via the mobile device.

The key requirements for contactless lead-through teaching are:

1. An a-priori known tag for the robot arm (e.g., a company logo or an ArUco marker [ArUco]), which means to know the size, shape, features, colors, etc. to track pose of the tag. The tracking will exploit 6D pose estimation methods based on RGB camera images [NBM+16].
2. A mobile device which comprises an IMU and an RGB sensor.
3. A sensor fusion algorithm to estimate pose of the mobile device with respect to the stationary robot base frame.
4. A communication and control interface toward the YuMi© robot controller, such as External Guided Motion (EGM) [EGM], which is used to execute the thought arm motion from edge cloud.
5. A user interface to allow different teaching sequences and jog modalities.

3.2 General planning of robot tasks

The task planning concept is composed of three main components, where each one takes care of individual aspects of the planning and the execution of the developed use case features. These components allow autonomous coordination between, e.g., mobile robot docking and P&P execution.

3.2.1 Behavior engine

The core component is a behavior engine that exposes a set of known capabilities of the overall robotics system, via a single uniform Robot Operating System (ROS) [ROS] action interface. Each capability is exposed as a behavior, which a higher-level component (e.g., a fleet management system for mobile robots) can request for execution. For example, behaviors can be:

- navigate a mobile robot platform to a location inside a map,

- dock a mobile robot platform, at an offset, based on a marker detected via a machine vision system,
- perform pick and place operation by activating and moving a robot arm.

Each behavior takes care of activating and communicating with lower-level system functions, such as mobile robot docking or stationary robot motion planning, as well as making sure that they are accessed in the correct order, and aborting execution/notifying the system's user if something goes wrong (e.g., the navigation location is unreachable, or no marker detected for the docking).

3.2.2 Task processor

The next component is a task processor, which takes a list of tasks to perform, where each task consists of the behavior(s) that should be requested along with the necessary parameters for each behavior. For example, a task list can be formulated as follows:

- task 1: navigate mobile robot platform to location L_1, perform docking with offset O_1, run pick and place for robot R_1,
- task 2: navigate mobile robot platform to location L_2, perform docking with offset O_1, run pick and place for robot R_2,
- etc.

The task processor goes through the whole task list and converts each task into messages for the behavior engine, as well as requesting the behavior engine to execute the behavior(s) in the correct order.

3.2.3 Task specifier

The final component creates a task list by specifying tasks with the behaviors to request, as well as setting the necessary parameters. The task list is then sent to the task processor for execution.

This component is manually specified/edited by the user who sets up the testbed demoing scenario.

3.3 Mobile robot navigation and docking

The key requirements for realizing mobile robot navigation and docking, as outlined in subsections 2.2.1 and 2.2.2, are:

1. A mobile robot platform with on-board LiDAR sensors for detecting obstacles, which also exposes a ROS-based communication interface [ROS] to receive motion velocities, such as Ridgeback [Ridgeback].
2. State-of-the-art algorithms for planning motion of mobile robot platform as well as obstacle avoidance (as offered by the ROS Navigation stack [ROSnavstck]).
3. An a-priori known tag for the mobile robot platform (e.g., an AprilTag marker [ATag]), which is used for tracking the mobile robot based on RGB camera images.
4. Video camera with a wide field-of-view for tracking the mobile robot platform.
5. An object detection and localization algorithm to estimate pose of the mobile robot platform with respect to the stationary robot base frame.
6. Task processor for mobile robot docking and navigation.

3.4 Stationary robot pick and place

The stationary robot pick and place (P&P) includes object detection, object pose estimation, and P&P decisions based on the object placement, robot arm trajectory planning, and trajectory execution. The key enablers for stationary robot P&P are:

1. An RGB camera with a depth sensor to generate point clouds of the object to be picked and placed and to do 6 degrees of freedom (DOF) pose estimation of the object.
2. An ArUco marker for the object pose estimation.
3. A communication and control interface toward the stationary robot controller, such as EGM, to execute the pick and place motion from the edge cloud.
4. Task processor for the P&P execution and for moving arm of the stationary robot to the home position after the execution ends.

3.5 AR visualization of robot status

The application AR visualization of robot status shows the potential of AR technology for supervision of industrial robots and their every-day operation, to increase work efficiency of human technicians on the factory floor. The overall functional design of the AR application prototype is summarized as follows:

1. Interface to collect operational status information from different industrial robots through their provided APIs, e.g., a robot identifier, robot health, and time in operation. The same interface will be used to also update on status information periodically for each such robot to match the robot's most actual state.
2. A marker tag fixed onto each of the robots to read their operational status.
3. AR buttons for a human technician to press and, consequentially, control robot's pre-thought movement. The Start, Stop, and Motion Change commands will be supported.
4. Graphical elements to visualize human view augmentations as well as displays with robot status information, e.g., 3D model of the robot and its parts, overlay panels with robot-specific information, and an instructional video about the robot.

There are different scenarios that are explored by combining AR technologies with industrial robots. One of them is remote robot monitoring for manufacturing industry. It provides the AR visualization of production-related information to any headset-equipped human worker on the factory floor in real-time. Other scenarios, like remote training and remote support, can also be explored.

4 Implementation of the software components and prototypes

This section documents implementation specifics and configuration details for the main prototyped software, following a component-by-component description organization in the previous section. It also reports on hardware components, software libraries, and open-source projects used for the implementation.

4.1 Lead-through teaching of robot pick and place

This section describes the lead-through teaching, where the stationary robot learns a behavior from observing the motion executed by a human worker. During demonstration the robot needs to learn two tasks, trajectory generation and object pick/drop. In the trajectory generation task, the human worker manipulates the robot by contactless lead through using the smartphone. The pick/drop task is taught by opening and closing the robot gripper. A brief description of teaching is first given, followed by explaining the user interface for teaching and manipulation.

4.1.1 Introduction

The following method description is provided for a generic manipulator (e.g., a robot arm), however both a mobile device and the method could be applied to a mobile robot platform. Let us assume using a generic manipulator with N DoF and equipped with a typical robot hand (i.e., gripper). The proposed user interface in the form of a mobile application for the teaching integrates both an RGB camera and an IMU. Sensor fusion algorithms are used to estimate the pose of the mobile phone. Moreover, the robot arm is equipped with an a-priori known AprilTag marker, which allows to estimate its pose by using 6D-tracking methods based on RGB camera image processing (Figure 7). An AprilTag is selected for the robot arm over, e.g., an ArUco marker as it generally provides better object detection resiliency, which is required for this real-time interaction. A computer vision algorithm based on OpenCV is used to compute the teaching device's pose with respect to the robot base frame and send it to the robot controller as the Cartesian reference

$$ {}^{B}T_{D} = {}^{B}T_{E}\, {}^{E}T_{D} \tag{1} $$

where subscripts B, E and D indicate frames of robot base, robot arm and teaching device (smartphone), respectively. The transformation ${}^{B}T_{E}$ is given by the forward kinematics (EE pose estimation), whereas ${}^{E}T_{D}$ is given by the camera/tag tracker. An additional homogeneous transformation DT offset is used to avoid the collision between robot arm and the teaching device.

$$ {}^{B}T_{offset} = {}^{B}T_{E}\, {}^{E}T_{D}\, {}^{D}T_{offset} \tag{2} $$

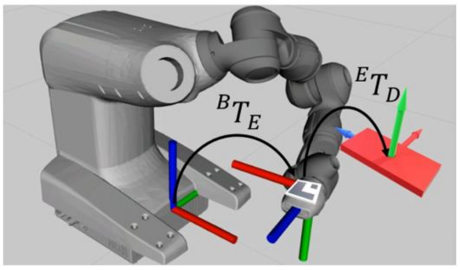


Figure 7 The device pose is retrieved by composing two homogeneous transformations $^{B}T_{E}$ and $^{E}T_{D}$, which are given by the forward kinematics and the camera/tag tracker, respectively.

A sensor fusion between the camera/tag tracker (with an operating frequency of 20 Hz) and the IMU (100 Hz) is used in order to improve the estimate of $^{E}T_{D}$, thus the performance of the proposed method. Figure 8 depicts the sensor fusion scheme. The teaching device's orientation is estimated by using a Madgwick filter in which the angular velocities (measured by gyroscope) are used to estimate sensor orientation, whereas static accelerations and the orientation retrieved by the camera/tag tracker are used during the correction phase. The teaching device's position is estimated by using a Kalman filter in which the dynamic accelerations are used to estimate the pose, whereas the position retrieved by the camera/tag tracker is used during the correction phase.

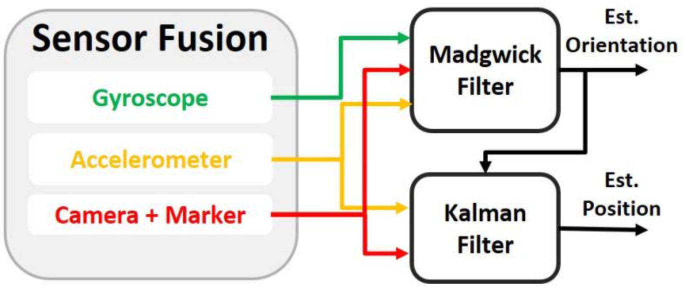


Figure 8 The picture depicts the sensor fusion between an IMU and camera/marker tracker. The orientation is estimated by using a Madgwick filter, whereas the position is given by a Kalman filter.

4.1.2 Solution architecture

Figure 9 depicts the software architecture for the P&P teaching and execution, illustrating flows of ROS messages between different nodes. The implementation is based on ROS1, which is used to build a system of ROS nodes that enables exchange of data for the teaching and execution. One ROS node is located in the mobile device for teaching, another one is collocated with machine vision algorithms for object tracking (e.g., during the P&P execution), while other ROS nodes reside in the edge cloud platform. Regarding the latter nodes, there is a ROS node collocated with the motion planning algorithm for stationary robots, which generates EGM messages for executing P&P operation.

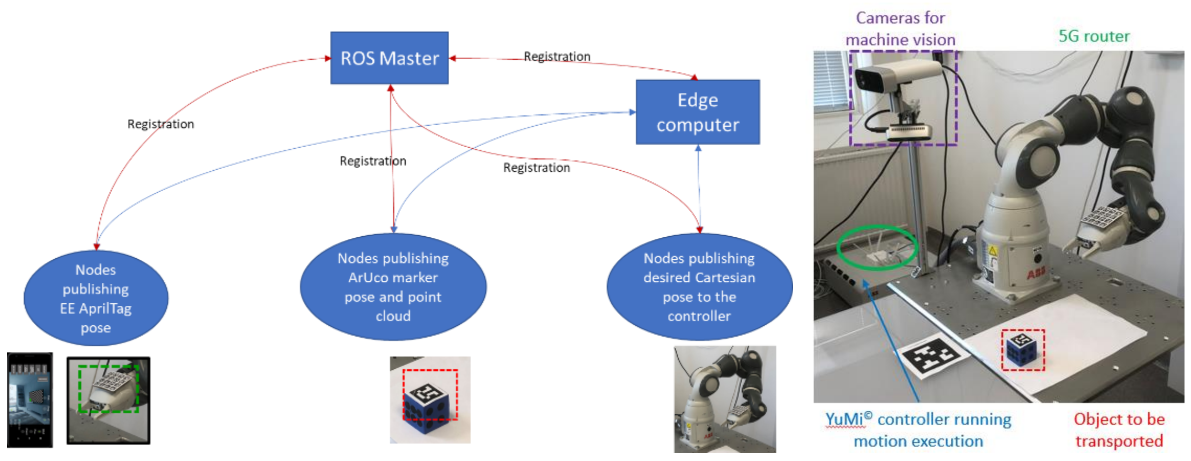
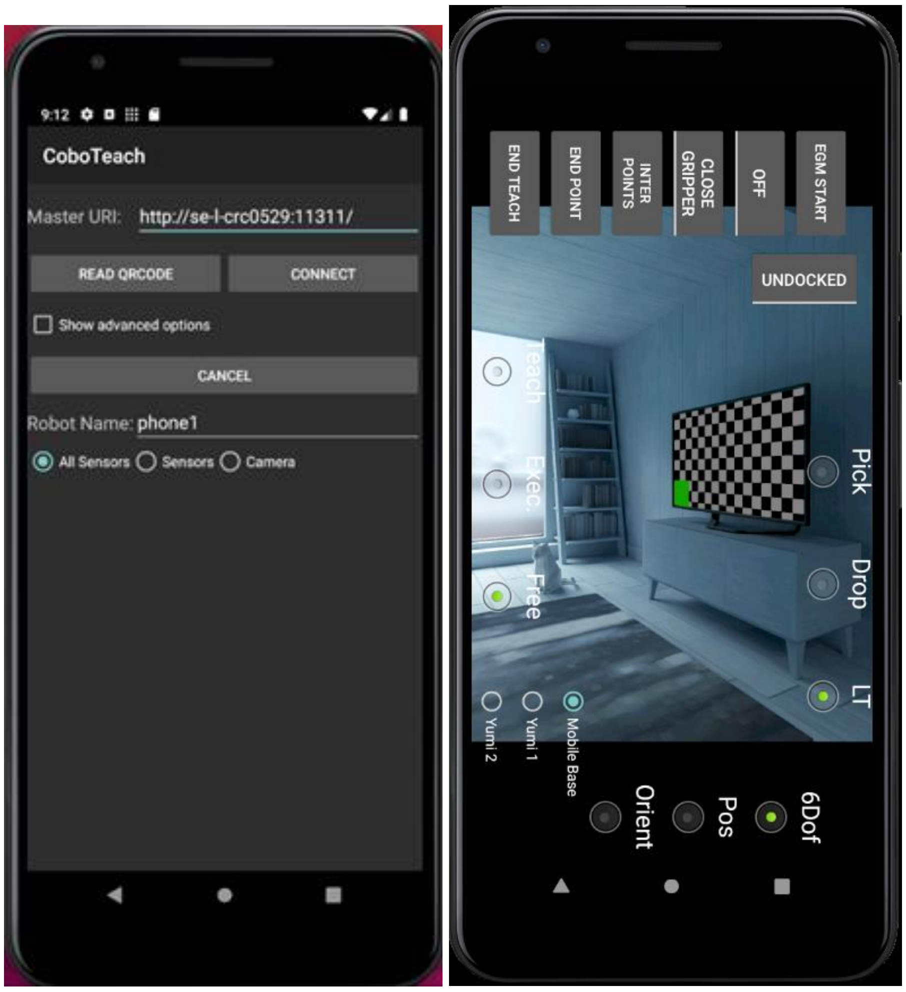

ROS Master
Registration
Edge computer
Registration
Registration
Registration
Nodes publishing EE AprilTag pose
Nodes publishing ArUco marker pose and point cloud
Nodes publishing desired Cartesian pose to the controller
Cameras for machine vision
5G router
YuMi© controller running motion execution
Object to be transported

Figure 9 Solution architecture for P&P teaching and execution (source: ABB)

4.1.3 User interface

The teaching mobile application consists of two pages (Figure 10). First one configures the teaching environment, e.g., it connects the ROS node to the ROS system for the communication purposes. The Robot Name is set to represent the name of the teaching device (“phone1” in Figure 10), while both video camera and IMU in the teaching device are used (“All Sensors” selected).


9:12

CoboTeach

Master URI: http://se-l-crc0529:11311/

☐ Show advanced options

Robot Name: phone1

[✓ All Sensors] Sensors Camera

Figure 10 User interface for the teaching developed for an Android-based smartphone (source: ABB)

The second page provides the user interface for jogging (manipulation) modes, robot selection, mode of operation in the teaching process, type of teaching, and a teaching sequence. An explanation of main options in the user interface is given below (Table 1).

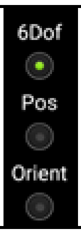
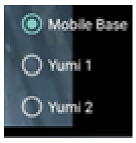
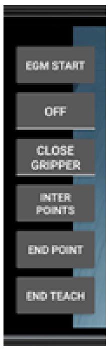
Jogging/manipulation mode: 1. **6Dof:** position + orientation 2. **Pos:** only position, orientation locked 3. **Orient:** only orientation, position locked	
Robot selection: 1. **Mobile Base:** to choose a mobile robot platform for the teaching/execution 2. **Yumi1:** to choose a single-arm robot arm for the teaching/execution, e.g. one of the two YuMi© robots 3. **Yumi2:** to choose another single-arm robot arm for the teaching/execution	
Mode of operation: 1. **Teach:** To start a teaching session 2. **Free:** To move the robot (lead-through jogging) 3. **Exec:** To start an autonomous execution session (the robot performs in autonomous mode what it learned during the teaching)	
Type of the teaching (only for testing) • **Pick:** To pick an object from mobile robot platform • **Drop:** To drop an object from robot workstation onto the mobile robot platform • **LT:** Test teaching (for debugging a test scenario)	
Teaching sequence • **EGM START:** To start EGM communication, if EGM connection between robot controller and the edge cloud platform (with the motion planning algorithm) timed out • **ON/OFF:** To enable/disable the lead-through teaching • **CLOSE GRIPPER:** To close and open the robot gripper • **INTER POINTS:** To store inter-points for creating a trajectory for teaching • **END POINT:** To store the sequence of inter-points and robot gripper status • **END TEACH:** To finish the teaching	 EGM START OFF CLOSE GRIPPER INTER POINTS END POINT END TEACH

Table 1 A summary of options in the user interface for lead-through teaching

4.1.4 Manipulation

The method has been implemented on an Android smartphone and demoed for a single-arm YuMi© robot. In the initial phase, the user who wants to teach moves the smartphone to the tag placed on the robot arm. When the user is satisfied with the distance between the camera and the tag (i.e., having a clear image of the tag marker), she/he places a finger above the ON/OFF button. At that moment the robot controller saves the afore-mentioned distance and uses it as the ${}^{D}T_{offset}$. Finally, the homogeneous transformation ${}^{B}T_{offset}$ resulting from (2) is used as a Cartesian reference by the robot controller until the teacher's finger is placed above the ON/OFF Button. In addition, the user interface (Figure 10) allows to close/open the gripper, select different jog modalities (free jog, lock the robot arm orientation, lock the robot arm position), and record points or trajectories in the robot workspace.

4.1.5 Teaching and jogging

To teach and jog the single-arm YuMi©, first the marker attached on the robot arm is detected by the teaching application. After that, the ON/OFF button in the application is used to enable lead-through teaching, followed by teaching and jogging the robot arm.

The following steps are performed for the teaching.

1. Place an object to be picked and placed in the machine vision camera's field of view.
2. Select one of the YuMi© robots (Yumi1/Yumi2) in the application.
3. Choose Teach in the application.
4. Use the option (Pick/drop), if you want to pick or drop the object: pick when the object is picked from mobile robot platform and dropped onto the YuMi© workstation and drop when the object is picked from the robot workstation and dropped onto mobile robot platform.
5. Move the robot arm by using the contactless lead-through (with the ON/OFF button). When needed, push INTER POINT, that point will be saved in frame of the object for P&P. The teacher needs to save multiple inter-points which make the intended trajectory of a robot arm for the P&P operation.
6. When the teacher finishes with the object for P&P (e.g., grasping a cube), push CLOSE GRIPPER. After that, select END POINT, because that is the last action related to the object.
7. Push END TEACH.

4.1.6 Execution test

To test a learned task, place the P&P object in the camera's field of view, either on mobile robot platform or fixed robot workstation, and push the EXEC button in the teaching application. The cube is picked/dropped based on the location, if the object is first on the fixed workstation, it will be dropped onto the mobile robot platform, and vice versa. To run the execution within a demo with other robotics features, there is no need to press the execution button. In such a case, the execution is called by a service from the general task planning component. That component calls two services from the behavior state machine, the first task is to pick and drop the object, and the second task is to bring the single-arm YuMi© into a home position after the P&P execution.

4.2 General planning of robot tasks

The task planning concept is composed of three main components (Figure 11), where each one takes care of individual aspects of the planning and the execution of the demoing scenario(s).

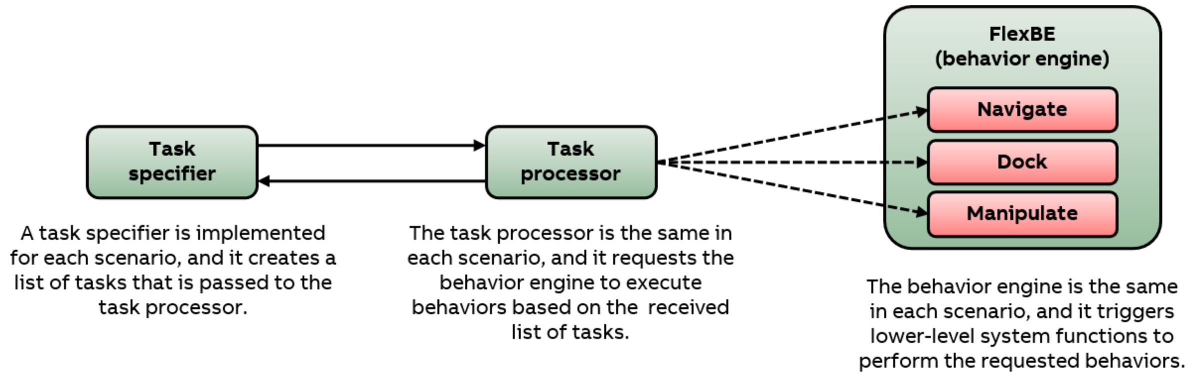

FlexBE
(behavior engine)
Navigate
Dock
Manipulate
Task specifier
Task processor
A task specifier is implemented for each scenario, and it creates a list of tasks that is passed to the task processor.
The task processor is the same in each scenario, and it requests the behavior engine to execute behaviors based on the received list of tasks.
The behavior engine is the same in each scenario, and it triggers lower-level system functions to perform the requested behaviors.

Figure 11 Sketch of the three main general planning components and their interactions.

4.2.1 Behavior engine

The core component is a behavior engine, and it is based on the third-party project called [ATag] University of Michigan, “AprilTag”, last access: December 30, 2021. Available at: https://april.eecs.umich.edu/software/apriltag

[FlexBE [FlexBE]. FlexBE provides ROS packages with ROS nodes for running the behavior engine, as well as the infrastructure, Python modules and APIs for creating custom behaviors. Three custom behaviors have been implemented for this testbed and this has been done in Python code that utilizes the FlexBE structure and APIs. The behavior engine exposes a ROS action server for requesting behaviors to be executed.

The first behavior is called “Navigate”, and it requests the navigation component to move the mobile robot platform to a location within the used map. For this, the behavior first requires input in the form of the frame of reference of the map, and the location (position and orientation) to navigate to within the specified frame of reference. Then, the behavior calls the navigation ROS node’s ROS action for performing the navigation. If the action request is accepted, then feedback is continuously received (with the platform’s current location) while the platform is moving and until the navigation is finished. After that, it reports the result of the action. The behavior finally reports back the result to the component that requested the behavior to be executed.

The second behavior is called “Dock”, and it requests the docking component to move the mobile robot platform based on a marker detected by the machine vision system. For this, the behavior first requires input in the form of which marker to use during the docking, the robot frame (specifying a part of the platform) to dock against the marker and the offset (desired relative position and orientation between the marker and the robot frame) specifying how to dock. Then the behavior calls the docking ROS node’s ROS action for performing the docking. If the action request is accepted, then feedback is continuously received (with the current control error) while the platform is moving and until the docking is finished. After that, it reports the result of the action. The behavior finally reports back the result to the component that requested the behavior to be executed. This behavior is also able to undock the mobile robot platform.

The third behavior is called “Manipulate”, and it requests the manipulation component to move the robot arm in question for the pick and place operation. For this, the behavior first requires input in the form of which robot to use during manipulation. Then the behavior calls the manipulation ROS node's ROS service to perform the manipulation. If the service request is accepted, then the behavior waits for the result of the service. The behavior finally reports back the result to the component that requested the behavior to be executed.

4.2.2 Task processor

The task processor is a ROS node implemented in C++, and it exposes a ROS action server for processing a list of tasks. When the task processor accepts an action request, it converts each task in the provided task list into action requests for the behavior engine. Each task's subtask specifies the desired behavior and the required inputs.

The task processor goes thought the whole task list and each task's subtasks and requests the behavior engine's action server to execute corresponding behavior. If the behavior engine accepts the request, then it provides continuous feedback until the behavior is finished, and it reports back the result of the action.

4.2.3 Task specifier

The task specifier component is a simple ROS node implemented in Python, that creates a task list specifying what to do, and then requests the task processor to execute that list via its action server. A user creates a use case specific implementation of this task specifier component and makes sure that correct data is specified.

The behavior engine and task processor ROS nodes are started as background processes, which then wait for the user to start a task specifier component.

The task specifier is a test and demo scenario specific component, and it can be a simple ROS node implemented in Python, which creates a task list with demo specific data and requests the task processor to execute the task list. For example, a task list can be:

- task 1:
  - navigate the mobile robot platform to a location A in front of robot workstation 1,
  - dock the mobile robot platform towards robot workstation 1, using offset O,
  - perform pick and place for robot 1,
- task 2:
  - navigate the mobile robot platform to a location B in front of robot workstation 2,
  - dock the mobile robot platform towards robot workstation 2, using offset P,
  - perform pick and place for robot 2,
- task 3: navigate mobile robot to a home location C.

4.3 Mobile robot navigation and docking

4.3.1 Introduction

This subsection describes the navigation and docking components utilized in this project for autonomous mobile robot motion and collision avoidance. The used mobile robot platform, Ridgeback [Ridgeback], is an omnidirectional platform and equipped with two 2D LiDAR sensors, one located in front and one in the back of the platform. These sensors provide a 360° perception allowing the mobile robot to move freely in any direction. Exploiting the ROS framework, for autonomous navigation open-source software packages, such as the Gmapping [Gmap] and ROS Navigation stack [ROSnavstck], are used. Gmapping is only used as a pre-requisite to ROS Navigation stack and provides a 2D map of the mobile robot's environment, based on a decoupled SLAM algorithm.

4.3.2 ROS Navigation stack

The Navigation Stack is simple on a conceptual level. It takes in information from odometry and sensor streams and outputs velocity commands to send to a mobile robot platform. The use of the Navigation Stack on an arbitrary robot, however, is a bit more complicated. As a pre-requisite for the navigation stack usage is that the robot must be running ROS, have a "tf" transform tree (description of all movable robot links) in place, and publish sensor data using the correct ROS message types. Also, the Navigation stack needs to be configured for the shape and dynamics of a robot to perform at a high level. To help with this process, this description is meant to serve as a guide to typical Navigation stack setup and configuration.

Overall Architecture

The ROS Navigation stack provides a modular system for integrating several modules related to mobile robot localization, path planning and perception under a common framework. The navigation stack architecture is illustrated in Figure 12. The architecture is built around a main module called "move_base". This core module integrates a 2D perception system (sub-module "Perception (2D)"), global and local path planning algorithms (sub-module "Path Planning (2D)"), local and global maps (dedicated to local and global path planners, respectively), and pose estimation inside a map system, i.e., global and local maps (sub-module "Localization (2D)"). The framework is modular enough to allow a developer to substitute any of the above sub-modules with a customized version. In this project we are utilizing the default open-source implementations included in the navigation stack package.

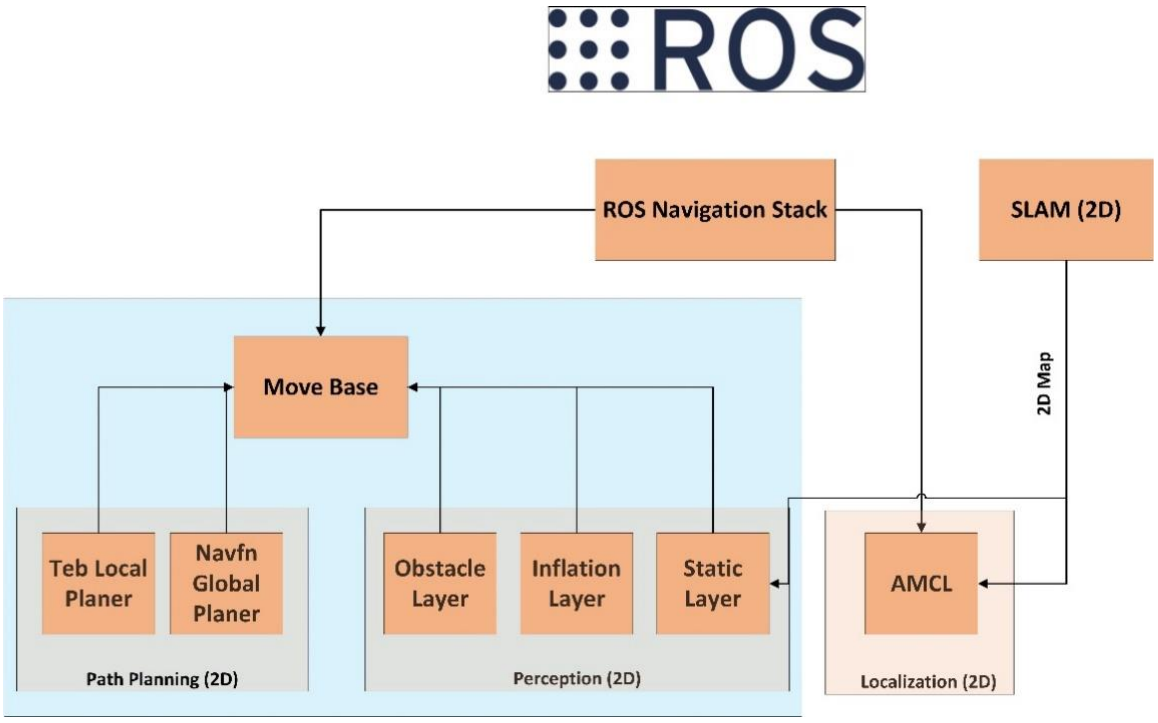


Figure 12 The ROS Navigation Stack modules [ROSnavstck]

The “move_base” module provides a ROS interface for configuring, running, and interacting with the navigation stack on a robot. A high-level view of the “move_base” module and its interaction with other components is shown in Figure 13. In that figure, the components highlighted by blue color vary based on the robot platform, the gray components are optional but are provided for all mobile robot systems, and the white components are required for all such systems.

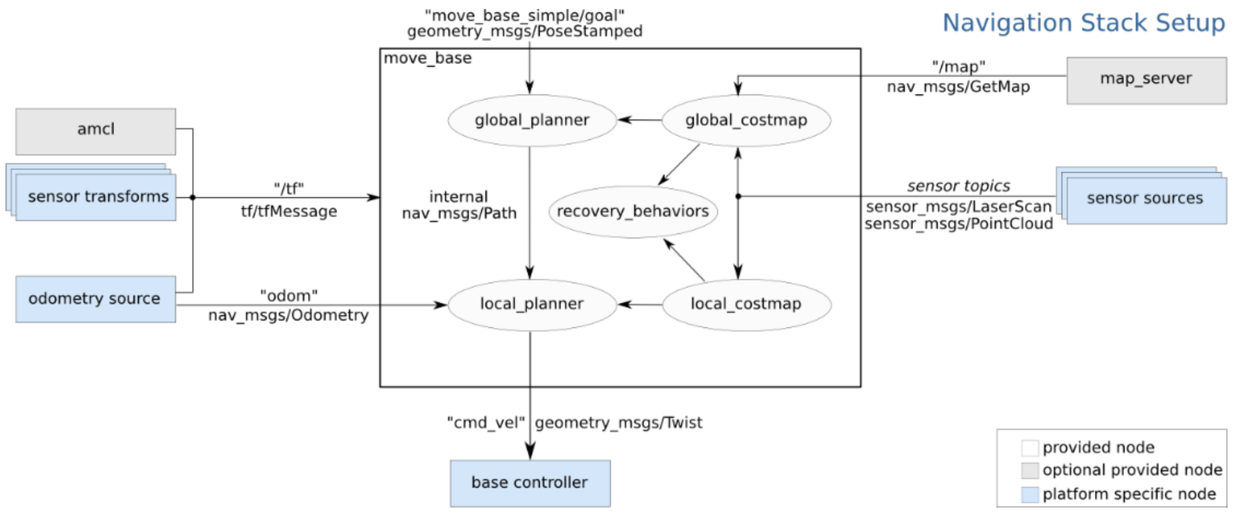


Figure 13 ROS Navigation Stack Node structure [ROSnavstck]

The Perception sub-module

Functionalities of this sub-module are organized into so-called layers. The 2D perception layers used here for optimal navigation and collision avoidance are divided into three main categories: a) static, b) dynamic, and c) complementary. In the static category, there is the "Static Layer", which represents the 2D map made available by the SLAM module and it depicts the robot's environment. The dynamic category includes the "Obstacle Layer", which is directly updated from the two 2D LiDAR measurements and represents the robot's environment at each time instant, including dynamically moving objects in the robot's environment (e.g., human workers). Finally, we have the complementary category and the "Inflation Layer". This layer extends (inflates) the size of detected obstacles according to a size parameter. The "Inflation Layer" is useful for generating optimal path plans at a safe distance from the detected obstacles. The 2D perception layers are depicted in Figure 12 and visualized "in action" in Figure 14. In the latter figure, the black line-dots depict the 2D map generated by the SLAM algorithm, the red line-dots are the LiDAR measurements, and the light cyan color indicates the inflated area around the detected objects.

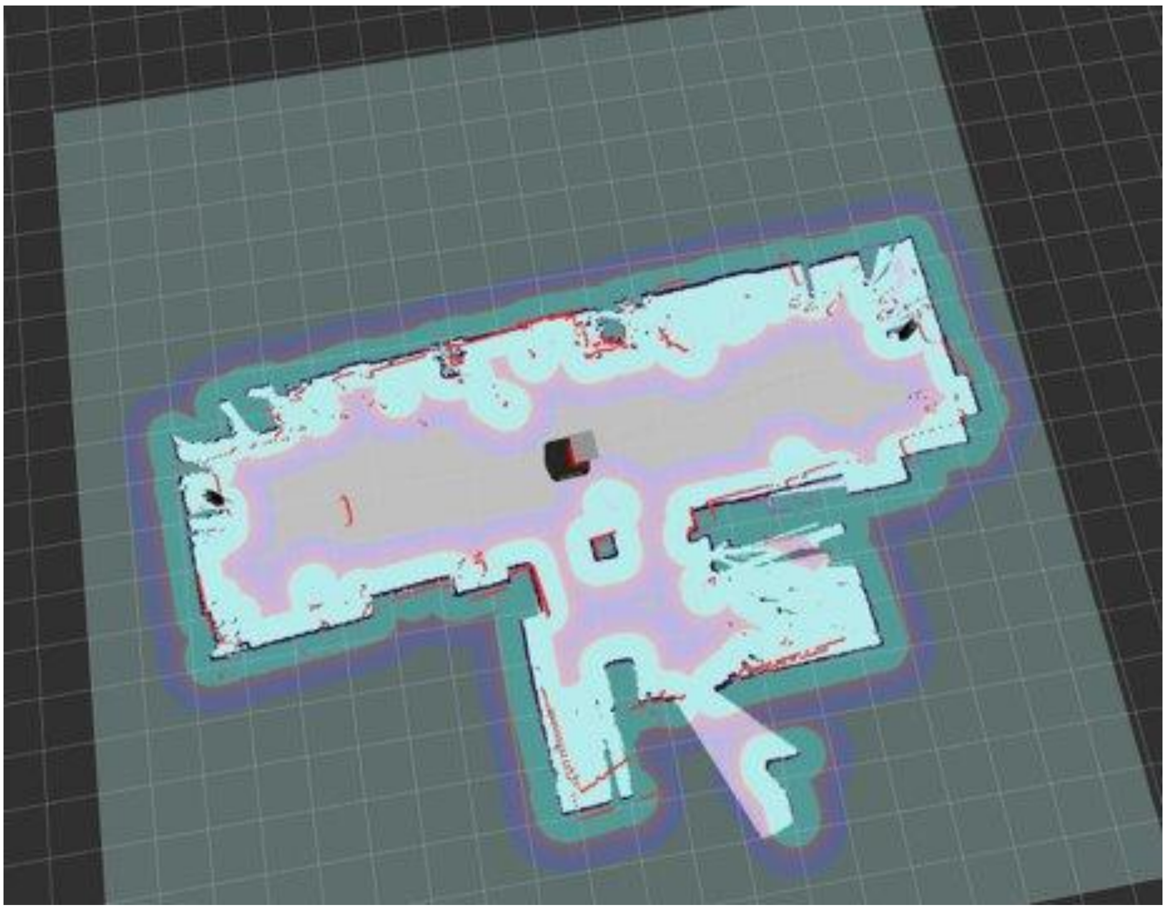


Figure 14 Perception and mapping with the ROS Navigation stack

4.3.3 Path planners

To effectively navigate, a robot needs an appropriate path plan and steering control. For reaching a target pose (position and orientation), a mobile robot platform uses a global and a local planner.

Before the robot starts to move, the global planner creates a path from its starting location (usually current robot pose) to the destination point, based on an acquired global map, which combines all the layers described in Figure 12. Then, the local planner subscribes to the global path plan, taking into account information about robot kinematics (e.g., size of the platform and drive system type, such as holonomic/non-holonomic). It also considers sensor readings, which build a local map of its surroundings and calculates relevant steering commands to achieve the goal. To use the "move_base" node in the navigation stack, we need to have a global planner and a local planner.

Global Planner

"navfn" uses Dijkstra's algorithm to find a global path with minimum cost between the starting point and the destination point. Global planner is built as a more flexible replacement of "navfn" with more options. These options include (1) support for algorithm A*, (2) toggling quadratic approximation, and (3) toggling grid path.

Local Planner

For the local planner, we use a method called "Timed Elastic Band" [RFW+13]. It implements an online optimal local trajectory planner for navigation and control of mobile robots as a plugin for the ROS Navigation stack. The initial trajectory generated by a global planner is optimized during runtime, by minimizing the trajectory execution time (time-optimal objective), separation from obstacles and complying with kino-dynamic constraints, such as satisfying maximum velocities and accelerations.

The current implementation complies with the kinematics of non-holonomic robots (differential drive and car-like robots). Support for holonomic robots is included since the Kinetic edition for ROS.

The optimal trajectory is efficiently obtained by solving a sparse scalarized multi-objective optimization problem. The user can provide weights to the optimization problem in order to specify the behavior in case of conflicting objectives. Since local planners such as the "Timed Elastic Band" get often stuck in a locally optimal trajectory, as they are unable to transit across obstacles, an extension was implemented with respect to its initial form. A subset of admissible trajectories of distinctive topologies is optimized in parallel. The local planner can switch to the current globally optimal trajectory among the candidate set. Distinctive topologies are obtained by utilizing the concept of homology / homotopy classes [RHB17].

The Localization sub-module

"AMCL" (adaptive Monte Carlo localization) is a probabilistic localization module for a mobile robot moving in 2D. It implements the AMCL approach, which uses a particle filter to track the position and orientation of a mobile robot in a given map.

To use adaptive particle filter for localization, we start with a map of the mobile robot's environment and we can either set the mobile robot to some position, in which case we are manually localizing it before starting to move, or we could very well make the robot start with no initial estimate of its position. As the robot moves forward, we generate new pose samples that predict the mobile robot's position (based on odometry) after the motion command. LiDAR readings are incorporated by re-weighting these samples and normalizing the weights. Generally, it is good to add a few random uniformly distributed samples as it helps the mobile robot to recover itself in cases where it has lost track of its position. In those cases, without these random samples, the robot will keep on re-sampling from an incorrect distribution and will never recover. The reason why it filters multiple sensor readings to converge is that, within a map, we might have ambiguities due to symmetry in the map (e.g., inside a featureless corridor), which is what gives us a multi-modal posterior belief.

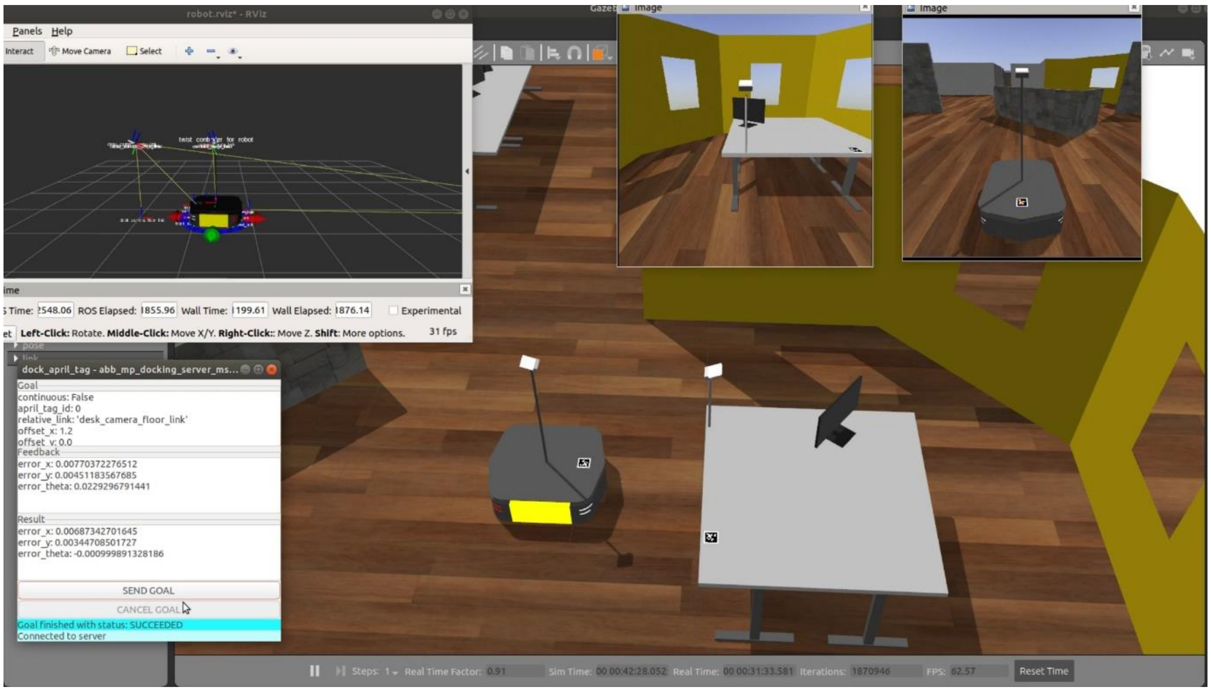
At the conceptual level, the "AMCL" package maintains a probability distribution over the set of all possible mobile robot poses, and updates this distribution using data from odometry and laser rangefinders.

The package also requires a predefined map of the environment (as noted in Figure 12) against which to compare observed sensor values. At the implementation level, the AMCL package represents the probability distribution using a particle filter. The filter is "adaptive", because it dynamically adjusts the number of particles in the filter: when the mobile robot's pose is highly uncertain, the number of particles is increased; when the mobile robot's pose is well determined, the number of particles is decreased. This enables the mobile robot to make a trade-off between processing speed and localization accuracy.

Even though the "AMCL" package works fine out of the box, there are various parameters which one can tune based on their knowledge of the platform and sensors being used. Configuring these parameters can increase the performance and accuracy of the AMCL package and decrease the recovery movements that the robot carries out while carrying out navigation.

4.3.4 The docking module

Navigation for the Rigdeback mobile robot between the YuMi© workstations is performed by using the ROS Navigation stack as described in previous subsections. However, the current "AMCL" localization algorithm has an absolute trajectory error accuracy of about 5 to 6 cm [RST+12]. Therefore, to have the mobile robot precisely approach each of the robot workstations, a docking system with less-than-centimeter accuracy is required. For that purpose, relative docking module was created based on the AprilTag [ATag] marker detection. The AprilTag visual fiducial detection algorithm [WO16] is used to detect strategically placed marker tags on the mobile robot, with a video camera fixed on each of the two robot workstations. When the video camera detects the AprilTag on-board the mobile robot platform, it calculates the relative distance and bearing angle between the tag and the workstation base. This distance is then fed into a proportional–integral–derivative (PID) controller that drives the mobile robot towards a targeted relative distance and bearing, which is millimeters away from the workstation. Figure 15 and Figure 16 illustrate usage of the mobile robot docking within a simulation and a real lab environment, respectively. The docking simulation figure contains screenshots of the used Gazebo software environment [Gazebo] and several simulation windows, such as execution of the docking control algorithm or views provided by virtual cameras.



```
dock_april_tag - abb_mp_docking_server_ms...
Goal
continuous: False
april_tag_id: 0
relative_link: 'desk_camera_floor_link'
offset_x: 1.2
offset_y: 0.0
Feedback
error_x: 0.00770372276512
error_y: 0.00451183567685
error_theta: 0.0229296791441

Result
error_x: 0.00687342701645
error_y: 0.00344708501727
error_theta: -0.000999891328186

Goal finished with status: SUCCEEDED
Connected to server
```

Figure 15 Gazebo simulated environment for development and testing of the docking module

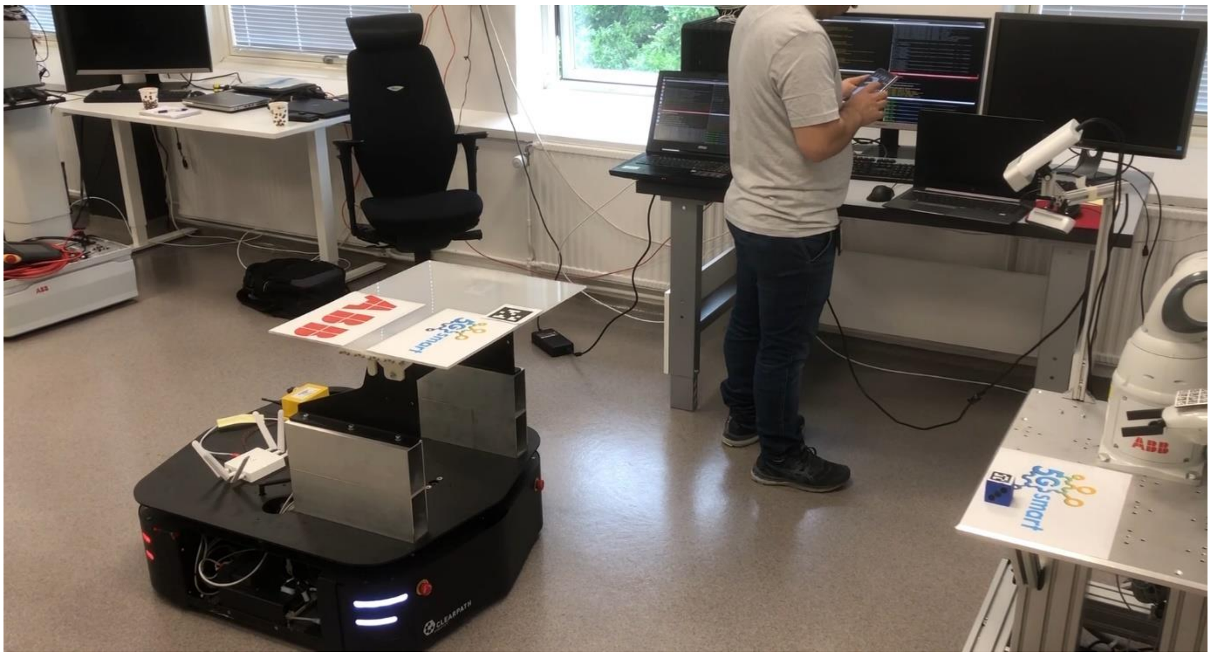


Figure 16 Mobile robot assumes a near position relative to robot workstation and is ready for the docking procedure to start (source: ABB)

4.4 Stationary robot pick and place

This section summarizes execution for the P&P operation of stationary robot, following the carried-out learning procedure (subsection 4.1.5). This execution is trigged by the FlexBE component as a service, after the mobile docking action is completed. The FlexBE calls two services from the behavior state-machine. For this report, the first service is to pick the object (e.g., a cube) from the mobile robot platform, while the second service is to bring the stationary robot arm into home position. The FlexBE calls the execution service, where the P&P object is recognized both with point clouds and the ArUco tag marker to estimate the position and orientation of the object. An AruCo marker is used for the P&P object specifically over, e.g., an AprilTag as its usage is somewhat simpler to configure, while still sufficing the pose estimation. The execution service produces a linear interpolated trajectory to pick the object according to the learned task. The execution process is depicted in Figure 17 for a cube. After the object is placed onto the robot workstation, the second service is called by FlexBE to bring the stationary robot arm to the home position.

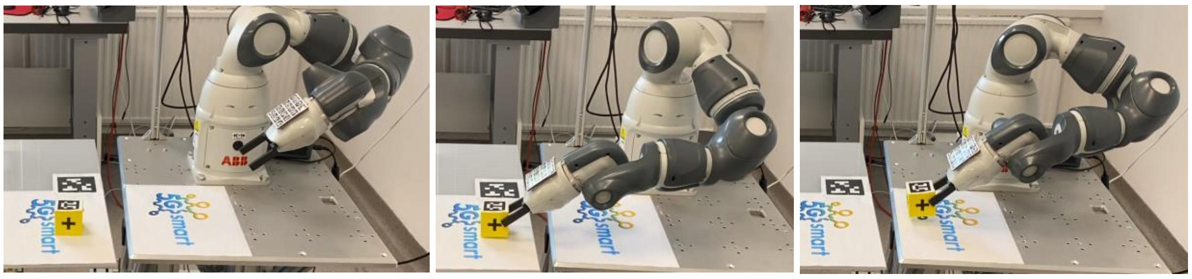


Figure 17 Execution service for the pick action on the cube, from left to right: (a) home position, cube's pose estimated, (b) the cube picked from the mobile robot platform, (c) the cube dropped onto the robot workstation (source: ABB)

After the first pick action is completed by placing the object onto the robot workstation, the mobile robot platform docks to the other workstation. Then, the execution service is called again, another cube is then picked from the robot workstation and dropped onto the mobile robot platform. That execution sequence is shown in Figure 18.

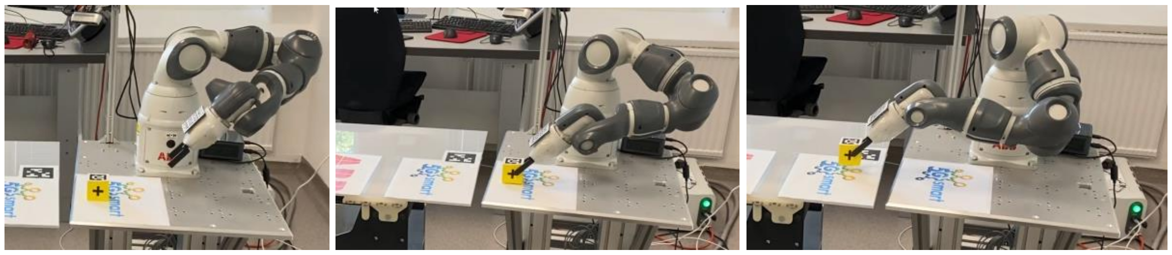


Figure 18 Execution service for the drop action on the cube, from left to right: (a) home position, cube's pose estimated, (b) the cube picked from the workstation, (c) cube dropped onto the mobile robot platform (source: ABB)

4.5 AR visualization of robot status

The solution for AR visualization covers the following functionalities (see Figure 19 for an overview):

1) AR-assisted robot monitoring using panel overlays over real-world industrial robots, to render robot status in real-time.
2) AR-assisted robot motion control, in which starting / stopping / changing execution of a pre-defined robot arm motion can be triggered by pressing AR-based buttons with the user's hands.
3) Visualization of a 3D robot hologram, robot technical information, and videos to enhance remote training.
4) The AR-assisted monitoring and motion control over a 5G network.

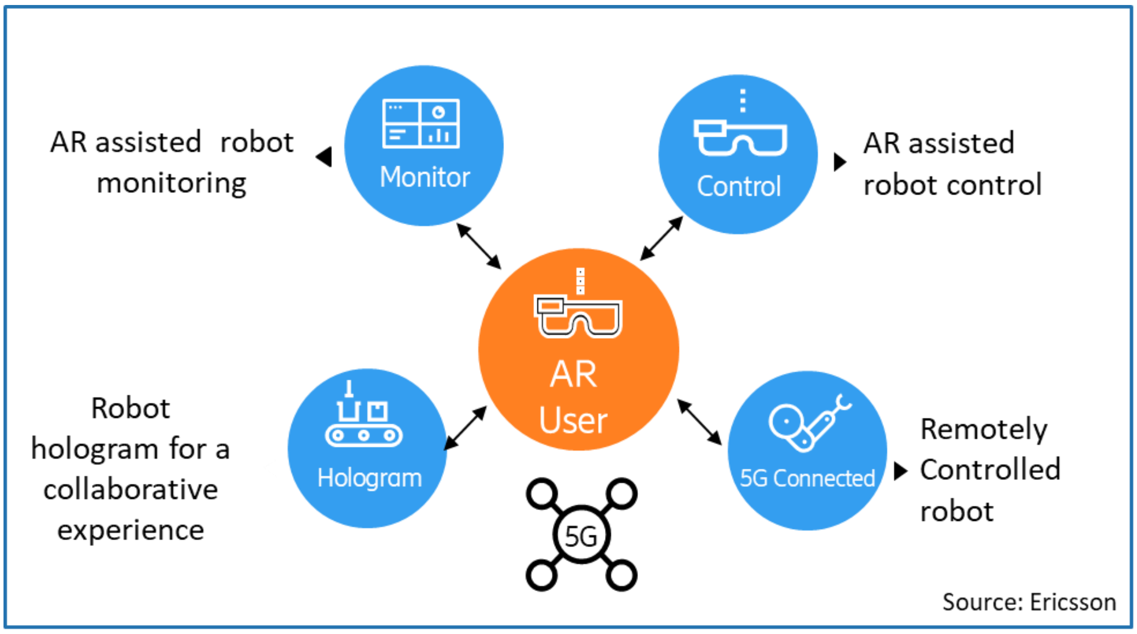

AR assisted robot monitoring
Monitor
Control
AR assisted robot control
AR User
Robot hologram for a collaborative experience
Hologram
5G Connected
Remotely Controlled robot
5G
Source: Ericsson

Figure 19 Overview of the AR visualization solution's functionalities

The main hardware and software components used for implementing and demoing the AR application are listed below.

Hardware:

1) Magic Leap One kit[1]: AR headset in which the application user can visualize and interact with information overlays and panels and control industrial robots using hand gestures.
2) YuMi© robot [YuMi]: stationary robot used to test and demo the AR application prototype.
3) High-end notebook (*minimum specification: Quad-core Intel or AMD, 2.5GHz, 2GB RAM, NVIDIA Geforce or AMD Radeon graphics, minimum resolution of 1280x1024 expected, Windows 8 64-bit or OS X El Capitan 10.11; recommended specification: 8th generation Intel i7 with 6 cores, 12 threads with hyperthreading, 8GB RAM, NVIDIA 1070 GTX or later with recent GeForce drivers for PC or NVIDIA GeForce or AMD Radeon GPU for Mac, Windows 10 64-bit or Apple OS X 10.11 to macOS 10.14; PC graphics cards being Open GL 4.3 conformant and Mac graphics cards being Open GL 4.1 conformant*) – a station where the AR application

[1] https://www.magicleap.com/en-us/magic-leap-1

prototype will run, being responsible for rendering graphical elements, sending commands to the stationary robot, retrieving robot status information, etc.

Software and other components:

1) Unity[2]: platform used to develop the AR application prototype.
2) The Lab – Lumin[3]: application that enables the execution of the AR application prototype on the Magic Leap One headset.

The software architecture of the AR application prototype is shown in Figure 20. The architecture is implemented by the following components:

1) Unity 3D system is an engine developed by Unity Technologies to design 3D applications, including those for AR. The entire AR application was built on it, which includes the HTTP protocols to interact with the robot's controller.
2) ABB Robot Studio allows design of robotic systems by furnishing ready-to-use virtual versions of the entire catalogue of ABB robots. It allows the setup of a simulated workspace for the robots and provides straightforward planning for them. It runs in a robot's controller, allowing it to control the robot.
3) Robot Web Services: ABB provides a set of APIs exposed in each of their low-level robot controllers, called Robot Web Services (RWS). This RWS interface uses HTTP to allow access to various robot controller information and data (via RAPID services) by clients external to the ABB Robot Studio. The APIs used are the GETs /rw/system/robottype, /rw/rapid/execution, /rw/motionsystem/errorstate, /rw/system/energy, /ctrl/clock, /ctrl/clock/timezone and POSTs /rw/rapid/execution/start/?mastership=implicit and /rw/rapid/execution/stop/?mastership=implicit.
4) RAPID[4] contains the Instructions, Functions and Data Types information (e.g., CPU and battery data) that can be read from the robot.

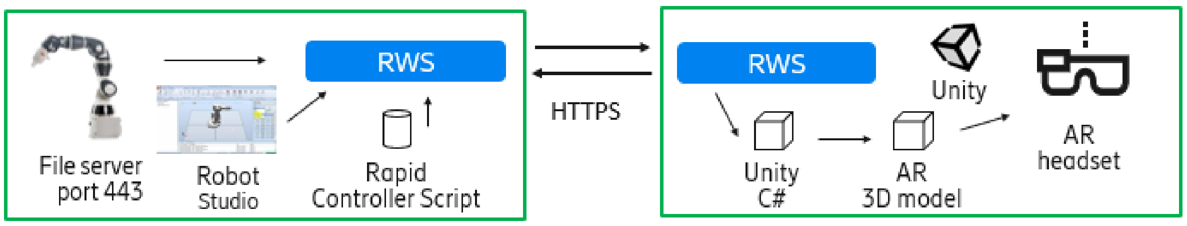


Figure 20 Robot Studio – Unity 3D integration for the AR visualization

The envisaged deployment scenario is shown in Figure 21. It contains two separate locations inside the same factory premises, where an AR-headset-equipped human technician is inspecting the ABB

[2] https://unity.com/
[3] https://developer.magicleap.com/en-us/learn/guides/develop-setup
[4] https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf

robot arm from a distance. The network layer supports a secure connection between the AR user and the robot controller, using communication protocol WireGuard[5] adopted by the RWS.

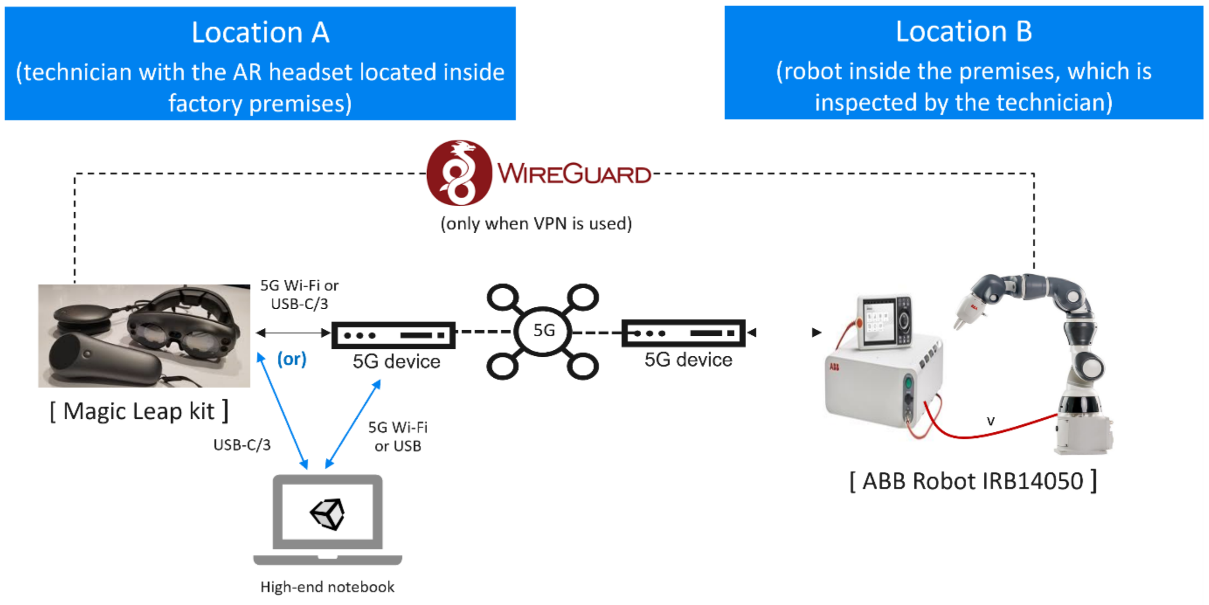


Figure 21 The deployment scenario for the AR-based use case implementation

A typical workflow of using the AR application prototype is described in Figure 22. The workflow covers AR visualization of robot status, remote robot motion control, robot monitoring, and robot troubleshooting.

[5] https://www.wireguard.com/

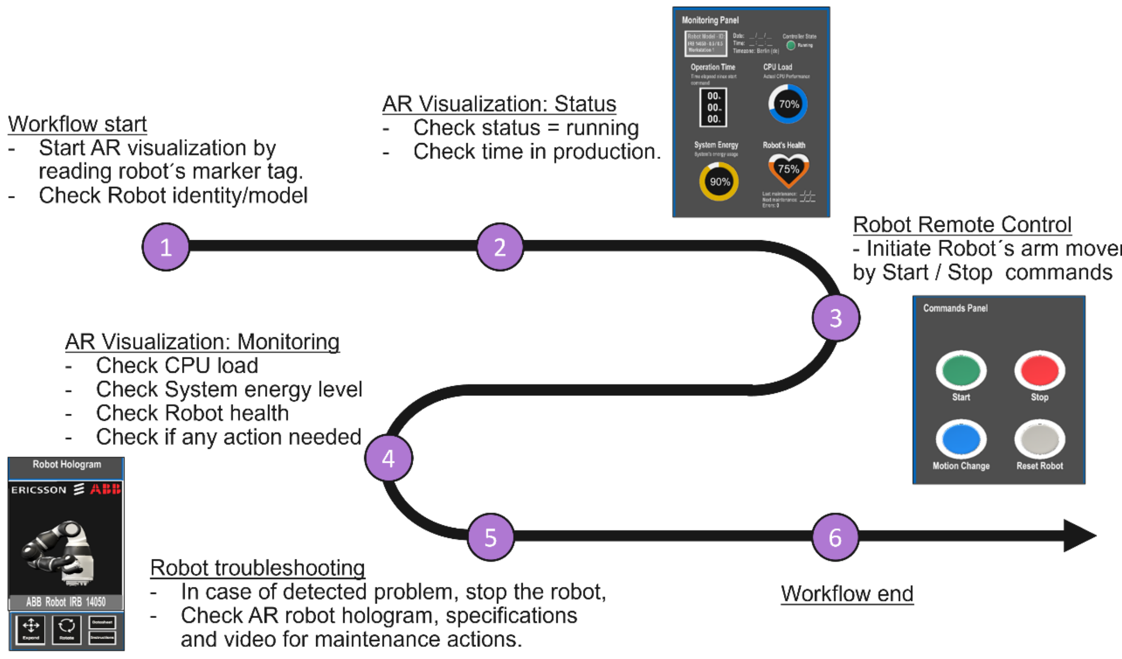

Workflow start
- Start AR visualization by reading robot´s marker tag.
- Check Robot identity/model
AR Visualization: Status
- Check status = running
- Check time in production.
Robot Remote Control
- Initiate Robot´s arm movements by Start / Stop commands
AR Visualization: Monitoring
- Check CPU load
- Check System energy level
- Check Robot health
- Check if any action needed
Robot troubleshooting
- In case of detected problem, stop the robot,
- Check AR robot hologram, specifications and video for maintenance actions.
Workflow end
1
2
3
4
5
6
Monitoring Panel
Commands Panel
Start
Stop
Motion Change
Reset Robot
Robot Hologram
ABB Robot IRB 14050

Figure 22 A typical workflow of using the AR application prototype

The AR application prototype consists of 3 panels as shown in Figure 23: Monitoring, Command, and Robot Hologram panels. These panels are described below.

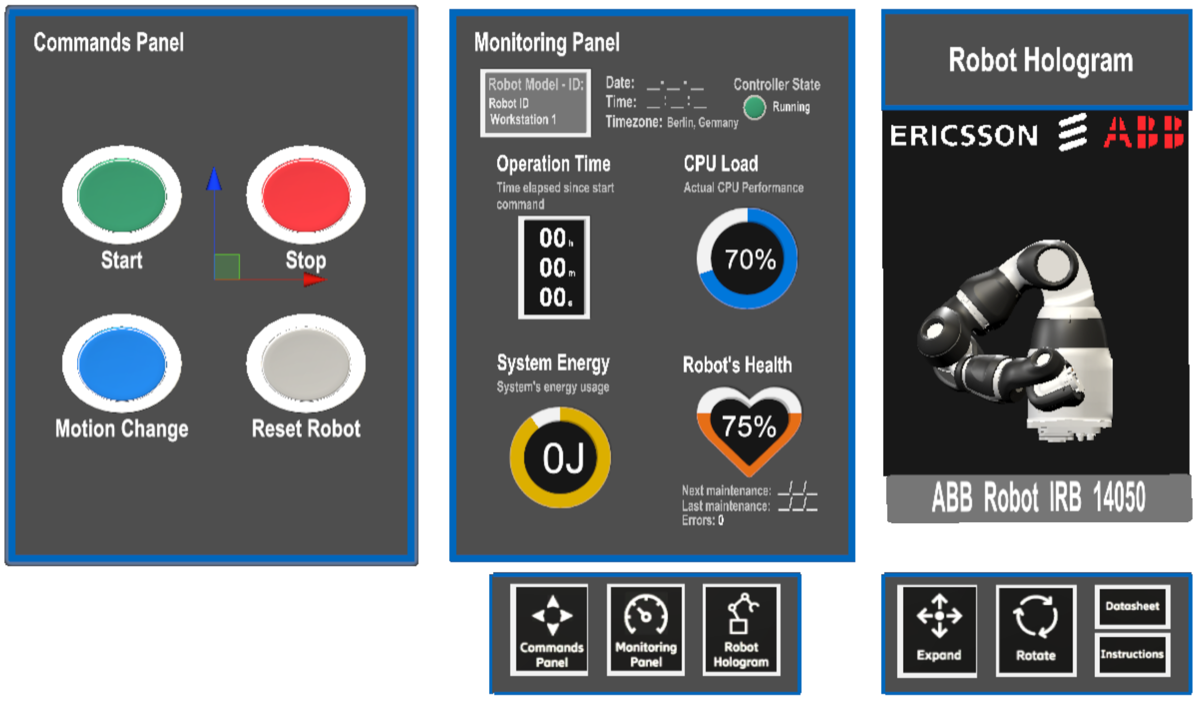


Figure 23 Panels of the prototype AR application

The Monitoring panel consists of robot data displayed in the form of AR overlays as follows:

1. Robot Model / Identity: static information consumed from the RWS API with model and a unique identifier of the robot.
2. Date / Time / Time zone: current date, time, and time-zone of the robot, consumed from the RWS API.
3. Controller State: dynamic information that represents the status of the robot control, e.g., running, or not running.
4. Operation Time: dynamic information that is counted when the robot starts to move (the Start command).
5. System Energy: average system energy consumption per hour.
6. CPU Load: dynamic information that is estimated based on robot system energy consumption.
7. Robot Health: dynamic information that is estimated as follows:
    - Health = 100% - 75% (green): current date is close to the last maintenance date.
    - Health = 75% - 50% (orange): current date is in the middle between the last and the next maintenance dates.
    - Health = 50% (red): current date is close to the next maintenance date.

    Note: for the demonstration purposes, the last and next maintenance dates are hard-coded, defined as 1 month before and after the current date, respectively.

The Command panel consists of AR-assisted commands to control a pre-defined movement of the robot arm. Four commands are available as described below: Start, Stop, Motion Change, and Reset Robot. The panel can be hidden by a closed fist gesture and displayed by a single-handed thumb up gesture at any moment:

1. Start: starts the robot movement.
2. Stop: stops the robot movement.
3. Motion change: changes robot motion path.
4. Reset robot: resets the robot system and stops the movement.

The Robot Hologram panel consists of the visualization of a 3D robot hologram. The AR application enables to:

1. Expand and rotate the stationary robot hologram.
2. Handle the stationary robot's base, joints and gripper with the AR user's hands.
3. Read, in form of an AR overlay, the robot's technical specification (e.g., data sheet) and watch an instructional video on how to handle the robot (being possible to play/pause and rewind the video).

5 5G infrastructure and integration

This section summarizes the 5G network solution at the Kista trial site, which provides connectivity to operate the implemented use cases. Deployment of the 5G network is outlined as well as how its mmWave radio connectivity is integrated into the robotics-related equipment.

5.1 The Kista trial site

For the 5G New Radio (NR) network installation at the Kista trial site, 5G is deployed on mmWave spectrum, which is also referred to as high band (HB) or frequency range 2 (FR2). One reason for such a design decision is the belief that HB could be an interesting option for indoor industrial deployments and connectivity solutions. Its short wavelength does not penetrate walls very well, so the network could be kept well isolated, and the available bandwidth on these frequencies is relatively wide, to also support future applications with (very) high requirements on throughput and capacity. Another reason is the local spectrum availability on a test license for HB. 5G NR, and other 3GPP technologies, rely on licensed spectrum to deliver a controlled radio environment and reliable performance. Spectrum in the range of 27.5 – 27.94 GHz (part of the so called 28 GHz band, or formally n257, a time-division duplexing (TDD) band in the range of 26.50 – 29.50 GHz) is licensed for testing at the Kista trial site. The network deployed is a so called 5G non-standalone (NSA) solution, which also requires a 4G LTE connectivity leg for the control plane communication. This is deployed on the so called 1800 MHz band (or formally B3, a frequency-division duplexing (FDD) band with uplink on 1710 – 1785 MHz and downlink on 1805 – 1880 MHz).

TDD is used for the user plane data traffic over the 5G NR connectivity leg (Figure 24). This means the spectrum is time-shared between uplink (UL) and downlink (DL), according to a pre-defined pattern. The 3GPP NR specification allows rather flexible TDD frame structures. At the base station (BS) side, the number of uplink and downlink slots may be almost arbitrarily configured within the TDD frame periodicity. A guard period, when switching from downlink transmission to uplink transmission, is implemented in the special slots, which are configured with a combination of suitable uplink and downlink symbols, interjected with a flexible number of silent symbols (or guard symbols). In practice, however, deployment of specific TDD frame structures in a network needs to be implemented and verified by interoperability testing between chipset and device providers, and infrastructure providers. The tradition of multi-vendor networks stipulates the need for, at least some level of interoperability verification through testing/trials between any network-UE combination, to secure network-wide support and cross-border roaming. The TDD pattern selected for the Kista network is the so called DDDSU[6] pattern, also referred to as 4:1 (four slots DL and 1 slot UL), where D/U indicates slots in which downlink/uplink-only symbols are transmitted, and S is the special slot. The special slot consists of 14 symbols and can be described as D:G:U, indicating the first D symbols are downlink, the following G are silent guard symbols, and the last U symbols are uplink. In the Kista deployment, the special slot was 10:2:2 (10 DL symbols : 2 guard symbols : 2 UL symbols).

[6] At present, commercial 5G networks are primarily used for mobile broadband (MBB) services. MBB traffic is DL heavy (most data is going from the network to the UE), so currently available TDD frame structures are designed to support this traffic. DDDSU is supported both by the network and the selected UEs.

The HB network is using a subcarrier spacing (SCS) of 120 kHz, compared to 15 kHz for LTE and 30 kHz for Sub-6GHz NR. This means that slot length is reduced, so that the transmission time interval (TTI) is 125 µs, and with the selected TDD frame structure DDDSU, the cycle time of the pattern is 0.625 ms. Compared to LTE and Sub-6GHz NR, the parts of the latency performance relating to radio transmissions are thus significantly improved. Further configurations to ensure best possible latency performance are to inactivate the so-called discontinuous reception functionality that brings the 5G user equipment (UE) into a sleep mode to save battery (which is impacting latency negatively when the UE wakes up again) and to activate prescheduling of UL radio resources (avoiding the step of first requesting UL transmission resources).

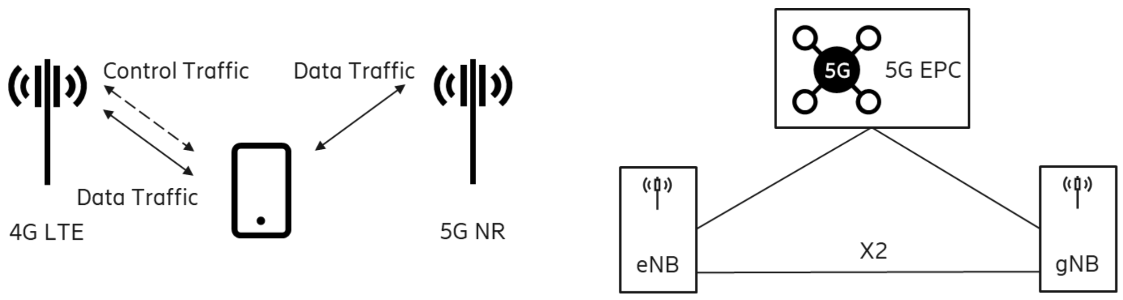


Figure 24: Illustration of the 5G Non-standalone architecture (source: Ericsson)

5.2 5G network solution

As previously described in 5G-SMART Deliverable D2.1 [5GS20-D21], the Kista trial site deployment consists of the Ericsson Radio Dot System for the 4G LTE leg and the 5G NR leg is implemented with a small-footprint radio in the HB spectrum. 200 MHz of the licensed bandwidth is utilized for the 5G NR part. Both LTE and NR radios are commercially available hardware, and the spectrum bands are both standardized and licensed. Other network equipment, such as Baseband Unit, core network (CN), switches, firewall, local edge cloud computing resources, etc., is installed in a 19" cabinet. Figure 25 provides a schematic overview of the 5G network solution.

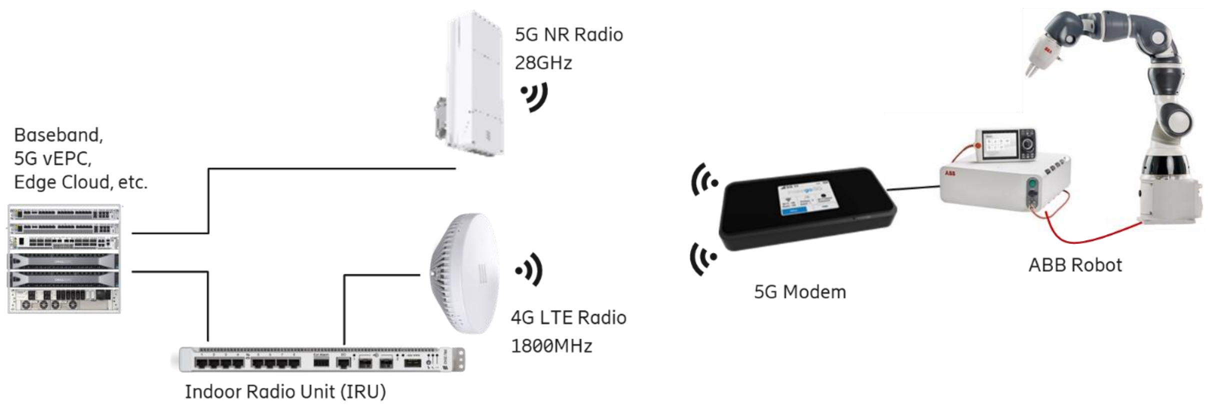


Figure 25: An example overview of the 5G communication network solution

The 5G network for the Kista trial site is installed to cover the complete testing area, where the robotics-related equipment is deployed. The two base stations (NR and LTE) are connected to a local core network, which is co-located with the edge cloud platform and installed nearby (labeled as "5G-SMART NW" in Figure 26) the testing area and the radios.

The 5G RAN and CN software used is generally and commercially available, and not modified in any way for the trials. The software is upgraded continuously and several times over the test period, to always have access to the ever-growing feature and functionality selection. All applications run over the 5G network and their associated traffic are served in a best-effort manner by the underlying 5G infrastructure, and due to expectedly good latency performance and plentiful spectrum availability, in combination with the small number of robots and other end-devices connected, the capacity and network capability in the factory can be seen as resource over-provisioning.

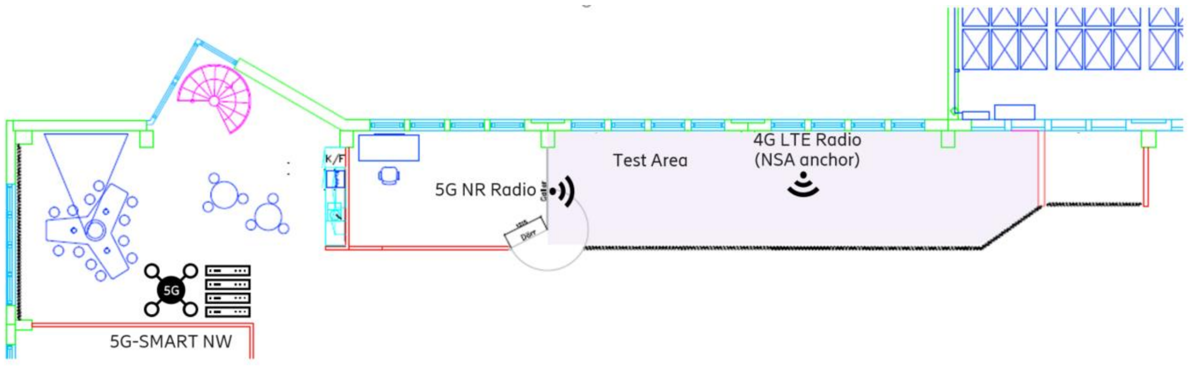


Figure 26: The Kista trial site 5G network deployment

Different 5G UEs are used for the different use cases, which display different needs for 5G integration and supporting communication functionality.

- Inseego Moretti is a so-called Pocket Router, which allows for the integration with different end-devices over a cabled USB-C interface. This UE type is used to integrate all the different robot units, both mobile and stationary, to 5G wireless communication, as well as the AR headset. For the AR headset specifically, 5G is integrated also using an intermediate laptop but it is expected that future applications will see 5G supported "natively".
- Sony Xperia Pro is a regular smartphone that is running the lead-through teaching application.

Figure 27 shows an overview of 5G interconnectivity at the Kista trial site. The network solution comprises two parts: a wired Ethernet portion and a wireless-connectivity part with the 5G NSA network using mmWave NR. The edge cloud platform with a centralized robot control and supervision is located in the wired portion, along with the two infrastructure computers that host machine vision algorithms as well as a communication gateway toward the 5G NSA network. All other equipment units communicate with the edge cloud platform, or with each other, over 5G.

The lead-through teaching and the AR visualization each utilize two 5G communication links when in operation. In the former case, information is exchanged between the teaching application on the 5G-enabled smartphone and the edge cloud platform, and then between the edge cloud platform and the YuMi© robot controller via the controller's 5G router. For the latter case there are two 5G connectivity legs, i.e., information is conveyed between the AR headset and the YuMi© controller via the headset's 5G router and via the controller's 5G router.

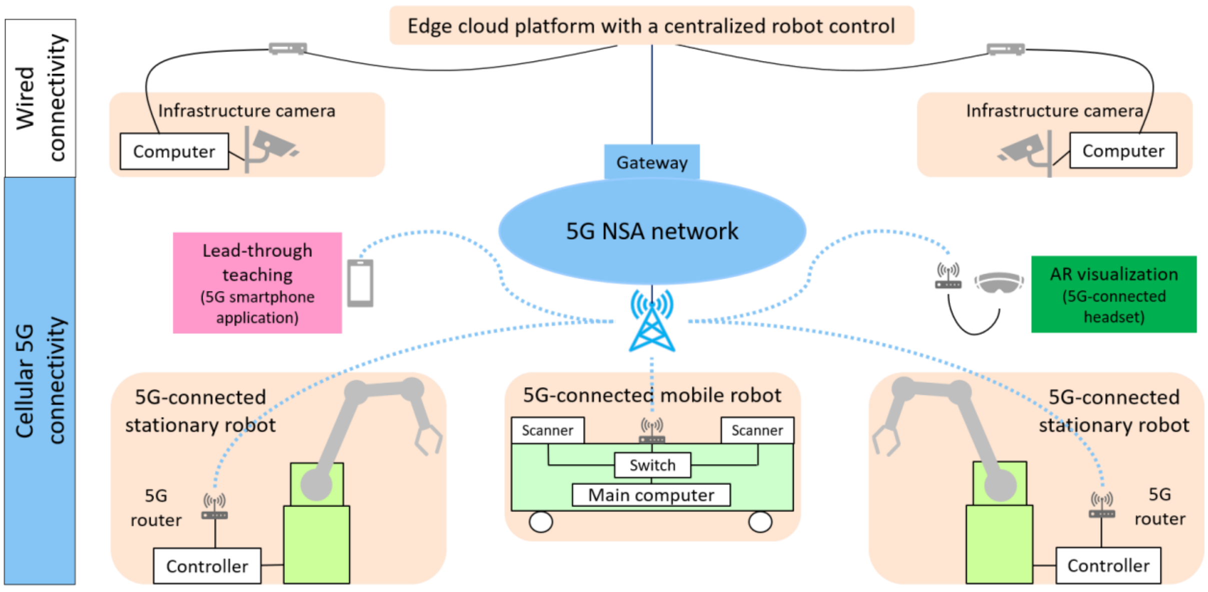

Wired connectivity
Cellular 5G connectivity
Edge cloud platform with a centralized robot control
Infrastructure camera
Computer
Infrastructure camera
Computer
Gateway
5G NSA network
Lead-through teaching (5G smartphone application)
AR visualization (5G-connected headset)
5G-connected stationary robot
5G router
Controller
5G-connected mobile robot
Scanner
Scanner
Switch
Main computer
5G-connected stationary robot
5G router
Controller

Figure 27 Overview of 5G interconnectivity at the Kista trial site

For mobile robot navigation and stationary robot pick and place, different 5G links are employed throughout different phases of their execution. For instance, LiDAR data is continuously delivered uplink to the edge cloud platform via the mobile robot's 5G router. In response, the edge cloud platform sends downlink motion commands to the mobile robot over its 5G router.

6 Conclusions

This deliverable has reported on the realization of the 5G-based testbed for industrial robotics at the Kista trial site. The main achievements in that activity are documented, explaining functions of the developed testbed components and their interactions, describing functional design and implementation of the software that is prototyped for delivering features of the industrial robotics use cases, and illustrating how these components are used for demonstrations. In addition, a 5G solution for the testbed is outlined, also depicting inter-connectivity of the components over 5G.

All envisaged features of the three use cases have been prototyped and demonstrated, namely mobile robot navigation from edge cloud with collision avoidance, vision-assisted mobile robot docking from edge cloud, lead-through teaching of stationary robot pick and place through human demonstration, vision-assisted execution of the object pick-and-place operation from edge cloud, remote manipulation of stationary robot motion, and AR-based visualization of operational robot information. Related hardware and software have been integrated with the 5G network at the Kista trial site and tested, demonstrating that the whole 5G-based testbed is fully operational and ready for the 5G validation phase.

7 References

[CCO19] I. Castelo-Branco, F. Cruz-Jesus, and T. Oliveira, "Assessing Industry 4.0 readiness in manufacturing: Evidence for the European Union", Computers in Industry, vol. 107, pp. 22-32, May 2019.

[ER20] B. Eschermann and R. Ramachandran, "Digitalization is making automation safer and greener", ABB Review, no. 4, pp. 8-15, October 2020. Available at: https://library.abb.com/d/9AKK107991A3781

[EGG+20] L. D. Evjemo, T. Gjerstad, E. I. Grøtli, and G. Sziebig, "Trends in Smart Manufacturing: Role of Humans and Industrial Robots in Smart Factories", Current Robotics Reports, vol. 1, pp. 35-41, April 2020.

[SL21] J. Sachs and K. Landernäs, "Review of 5G capabilities for smart manufacturing", 2021 17th International Symposium on Wireless Communication Systems (ISWCS), pp. 1-6, September 2021.

[5GS20-D11] 5G-SMART, Deliverable 1.1 "Forward looking smart manufacturing use cases, requirements and KPIs", June 2020.

[5GS20-D21] 5G-SMART, Deliverable 2.1 "Design of 5G-Based Testbed for Industrial Robotics", May 2020.

[YuMi] ABB, "Single-arm YuMi Collaborative Robot | ABB Robotics – Collaborative Robots | ABB Robotics", last access: December 30, 2021. Available at: https://new.abb.com/products/robotics/collaborative-robots/irb-14050-single-arm-yumi

[Ridgeback] Clearpath, "Ridgeback – Omnidirectional mobile manipulation robot", last access: December 30, 2021. Available at: https://clearpathrobotics.com/ridgeback-indoor-robot-platform/

[Kinect] Microsoft, "Azure Kinect DK – Develop AI Models | Microsoft Azure", last access: December 30, 2021. Available at: https://azure.microsoft.com/en-us/services/kinect-dk/

[RealSense] Intel, "Intel® RealSense™ Technology", last access: December 30, 2021. Available at: https://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html

[MagicLeap] Magic Leap, "Augmented reality platform for Enterprise | Magic Leap", last access: December 30, 2021. Available at: https://www.magicleap.com/en-us

[ArUco] OpenCV, "OpenCV: ArUco marker detection (aruco module)", last access: December 30, 2021. Available at: https://docs.opencv.org/4.x/d9/d6d/tutorial_table_of_content_aruco.html

[NBM+16] C. Nissler, S. Büttner, Z. -C. Marton, L. Beckmann, and U. Thomasy, "Evaluation and improvement of global pose estimation with multiple AprilTags for industrial manipulators", 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1-8, September 2016.

[EGM] ABB, "Externally Guided Motion", Application manual, 2021.

[ROS] Open Robotics, "ROS: Home", last access: December 30, 2021. Available at: https://www.ros.org/

[ROSnavstck] Open Robotics, "navigation – ROS Wiki", last access: December 30, 2021. Available at: http://wiki.ros.org/navigation

[ATag] University of Michigan, "AprilTag", last access: December 30, 2021. Available at: https://april.eecs.umich.edu/software/apriltag

[FlexBE] Philipp Schillinger, "FlexBE Behavior Engine", last access: December 30, 2021. Available at: http://philserver.bplaced.net/fbe/

[Gmap] Open Robotics, "gmapping – ROS Wiki", last access: December 30, 2021. Available at: http://wiki.ros.org/gmapping

[RFW+13] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Efficient trajectory optimization using a sparse model", 2013 European Conference on Mobile Robots, pp. 138-143, September 2013.

[RHB17] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies", Robotics and Autonomous Systems, vol. 88, pp. 142-153, February 2017.

[RST+12] J. Röwekämper, C. Sprunk, G. D. Tipaldi, C. Stachniss, P. Pfaff and W. Burgard, "On the position accuracy of mobile robot localization based on particle filters combined with scan matching", 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3158-3164, October 2012.

[WO16] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection", 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4193-4198, October 2016.

[Gazebo] Open Source Robotics Foundation, "Gazebo", last access: December 30, 2021. Available at: http://gazebosim.org/

Appendix

List of abbreviations

3GPP	The 3rd Generation Partnership Project
4G	The Fourth Generation (of cellular network technologies)
5G	The Fifth Generation (of cellular network technologies)
API	Application programming interface
AMCL	Adaptive Monte Carlo localization
AR	Augmented Reality
BS	Base station
CN	Core network
DL	Downlink
DOF	Degrees of freedom
EGM	External Guided Motion
FDD	Frequency-division duplexing
FR2	Frequency range 2
HB	High band
IMU	Inertial measurement unit
LiDAR	Light Detection and Ranging
LTE	Long-Term Evolution
MBB	Mobile broadband
mmWave	Millimeter Wave
NR	New Radio
NSA	Non-standalone Architecture
OS	Operating System
PID	Proportional–integral–derivative (controller)
P&P	Pick and place (operation)
RAN	Radio Access Network
RGB	Red, green, and blue (color model)
ROS	Robot Operating System
RWS	Robot Web Services
SCS	Subcarrier spacing
SLAM	Simultaneous Localization and Mapping
TDD	Time-division duplexing
UE	User Equipment
UL	Uplink

Table 2: List of abbreviations