Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

A management system for autonomous mobile robots

Bc. Filip Dvořák

Supervisor: Ing. Pavel Burget, PhD.
July 2023

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Dvořák Filip**	Personal ID number:	**406426**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Control Engineering**		
Study program:	**Cybernetics and Robotics**		

II. Master’s thesis details

Master’s thesis title in English:

A management system for autonomous mobile robots

Master’s thesis title in Czech:

Systém pro řízení autonomních mobilních robotů

Guidelines:

The goal of this thesis is to design and implement a fleet management system, which will be based on KUKA Navigation Server[1] for KUKA autonomous mobile robots KMP and KMR[2]. The fleet management system will become part of a multi-agent system[3] developed in Testbed for Industry 4.0 and will provide services of delivering loads (i.e. parts or products) of different types to required locations, mainly at stationary robots or assembly lines.
1) Design and implement an architecture of a fleet management system (FMS) that can connect to various navigation servers to allow managing various types of mobile robots. The FMS will also have interface to a warehouse management system and an independent localization system. The FMS must include an interface to the multi-agent system of Testbed.
2) Get acquianted with the KUKA mobile robots and their navigation solution (KUKA navigation server). Transfer KUKA navigation server from local computers in the robots to a central server that will serve for all mobile robots in Testbed.
3) Implement high-level functionality to allow delivering required load (e.g. set of parts for a stationary robot) by a mobile robot from a warehouse. This includes automatic selection of a best-suitable mobile robot, real-time trajectory adaptation, and in-time delivery.
4) Test the system in Testbed with at least two zones at stationary robotic lines.

Bibliography / sources:

[1] KUKA Aktiengesellschaft Germany. KUKA Navigation Solution [online]. [vid. 2021-12-01] Available at:
https://www.kuka.com/-/media/kuka-downloads/imported/9cb8e311bfd744b4b0eab25ca883f6d3/kuka_navigation_solution_en.pdf
[2] KUKA Aktiengesellschaft Germany. KUKA Product portfolio_01/2022 [online]. [vid. 2022-01-01] Available at:
https://www.kuka.com/-/media/kuka-downloads/imported/9cb8e311bfd744b4b0eab25ca883f6d3/kuka_rob_product-portfolio_en_screen.pdf
[3] Yoav Shoham and Kevin Leyton-Brown. Multiagent Systems: Algoritmic, Game-Theoretic and Logical Foundations. Cambridge University Press. June 2012. ISBN 0521899435

Name and workplace of master’s thesis supervisor:

Ing. Pavel Burget, Ph.D.   Testbed CIIRC

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **17.01.2023**   Deadline for master's thesis submission: **14.08.2023**

Assignment valid until:
by the end of summer semester 2023/2024

Ing. Pavel Burget, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgements

I want to thank to Ing. Pavel Burget, PhD. for his support during my writing of the thesis. Also for the opportunity to work in Testbed for Industry 4.0, where I had the chance to work with many different robots and other new technologies and hardware. I also want to thank to my family, girlfriend and friends for their patience they had to show during my journey through my studies.

Declaration

I declare that the submitted work was developed independently and that I stated all the available information sources in accordance with the Methodological instruction on the observance of ethical principles in the preparation of university theses.

Abstract

The goal of this thesis is to design and implement a fleet management system for mobile robots wihtin Testbed for Industry 4.0 at CIIRC CTU. This system will be built over existing navigation solution for mobile robots created by KUKA AG company. The Fleet management system should provide capability to deliver shelves and loads (i.e. parts or products) of different types to required locations within Testbed workplace. The implemented fleet management system will have an interface for communication within the multi-agent system developed in Testbed. Architecture of this system is designed in a way that it is possible to extend it for control of other robotic systems or other types and brands of mobile robots. Fleet manager also implements information from independent localisation system which allows optimisation in delivering required load by a mobile robot from the warehouse or in between the robotic lines. This optimisation includes automatic selection of a best-suitable mobile robot, real-time trajectory adaptation, and in-time delivery. Designed fleet management system is then tested for a set of transportation tasks. Tasks are given through multi-agent interface. We compare the processing time of the task in the case of embedded input from the real time location system and without it.

Keywords: mobile robotics, in-time delivery, multi-agent systems

Supervisor: Ing. Pavel Burget, PhD.

Abstrakt

Cieľom tejto diplomovej práce je implementovať Fleet management system, ktorý bude vychádzať z hotového riešenia KUKA Navserver, ktoré využíva mobilných robotov KUKA KMP a KMR. Tento navrhnutý systém by mal obsahovať rozhranie pre komunikáciu s multi-agentným systémom, ktorý je vyvíjaný v priestoroch Testbedu pre priemysel 4.0. Fleet management systém by mal umožniť prepravu tovaru, súčiastok alebo celý regálov do požadovaných lokácií, a to primárne medzi skladom a výrobnými linkami. Pri návrhu systému sa prihliada na možnosť neskoršieho zakomponovania mobilných robotov iných výrobcov. Systém by mal implementovať dáta z nezávislého lokalizačného systému. Tieto dáta systém využíva pre riadenie robotov s ohľadom na optimalizáciu času doručenia a predchádzanie stretom s prekážkami. Navrhnutý systém testujeme zadávaním úloh prepravy skrz mutliagentné rozhranie. Následne porovnávame časy splnenia úloh v prípade zakomponovania dát z lokalizačného systému s prípadom, kedy tieto dáta nie sú k dispozícii.

Klíčová slova: mobilní robotika, včasné doručení, multiagentní systémy

Překlad názvu: Systém pro řízení autonomních mobilních robotů

Contents

Figures

Tables

Chapter 1

Introduction

In today's world industry, more and more emphasis is being placed on automation and production efficiency. In the case of the industrial production, there is a large degree of robotization of the production workplaces, which leads to the marginalization and displacement of the human workforce. This trend is evident not only in the robotization of production lines but also in the development of automated warehouses aimed at expediting the movement, storage, and dispatch of goods. The implementation of automated goods transportation extends beyond warehouses, finding its way into the factory floor. Mobile robots play a crucial role in transporting goods, materials and parts from local factory warehouses to production lines, where these items are further processed. Then, of course, robots can also carry parts in between the production lines. Another example of mobile robots usage is in matrix production [1], where mobile robot carry the basis of the product across the factory by going through different production lines by step. The product is slowly built directly on the back of the mobile robot. This approach is used for example in automotive.

The term *Industry* 4.0 [2] together with the term *Flexible production* [3] refers to such a form of automation mentioned above and they are trying to make production and factories smarter. Both of these concepts are based on enhancing manufacturing agility and responsiveness. Both concepts capitalize on data-driven decision-making, real-time connectivity, and the seamless exchange of information. Industry 4.0 technologies provide the infrastructure necessary for flexible production, enabling manufacturers to monitor, analyze, and optimize their processes to accommodate changes in demand, design, and production methods.

In our work, we design a system that adopts and applies the concepts of Industry 4.0 and flexible production in a specific application. Specifically, we propose and implement system of smart management of material transportation at the workplace between the local warehouse and individual production lines. System should make this transportation as optimal as possible with respect to time. We extend the data given to system in a way so it not only incorporates data from existing robotic solution from KUKA, but also from independent real time localisation system. Given all the data, we develop system which can optimize and make the transfer of the material at the workplace more efficient. This kind of optimisation in real application leads to an increase in the production of the factory and thus to greater profits. The system should be easy to extend and allow management of robotic systems of various brands, according to standards like VDA5050 [4]. Thus, making it more accessible for real customers. The system also implements interface and connection to the existing multi-agent platform, which creates possibilities for further optimization and increase in efficiency of the production.

1.1 Aim and contribution of the work

The main goal and contribution of the work is in the design and implementation of a real application - Fleet management system (FMS). This system will be physically deployed in the Testbed Industry for 4.0 department at Czech Institute of Informatics, Robotics and Cybernetics of the Czech Technical University. In Testbed, which serves as place for research and development in industry 4.0 technologies, there are multiple production lines and local warehouse with shelves and parts intended for production. In order to not to use human workforce to move parts or whole shelves between production lines, a mobile robotic system from KUKA AG company was purchased. It is possible to install and program this system to do simple transport tasks from one location to another. But that would be considered as hard-coded application which does not meet the standards of Industry 4.0 or Flexible production.

In Testbed, research and development takes place and department functions as a place where the latest technologies of the industry can be seen. Projects that deal with flexible production and efforts to increase the efficiency of production in industry are developed and take place here. All this using the latest industrial technologies available on the market. The purpose of this work is to develop the capabilities of the KUKA robotic system in a way so it´s transport capabilities will be available through specific interface for some of these ongoing projects and applications which are being developed, that have are research or also of business nature. Specifically, it is, for example, the development of a multi-agent system taking care of the production process of the product in the factory. Another example is the development of a manufacturing execution system, which also procures and optimize the production process over several production lines.

The designed FMS system not only exposes the possibility of transporting goods for superior control systems, but also additionally processes data from the real time location system installed at the workplace. This means that FMS incorporates a certain optimal control of the motion of mobile robots, which is normally not available in the KUKA Navserver solution.

Testbed also serves as a showroom, where the latest technologies are presented for technological companies interested in automation and flexible manufacturing. It is the place which holds various workshops, presentations and conferences about the topics in industry. Mobile robotics and its demonstrations are not commonly seen in the Czech Republic. This means that the functional demonstration of the mobile warehouse is an attraction for potential customers and those interested in cooperation with the Czech Institute of Cybernetics and Robotics.

1.2 Structure of the thesis

The work consists of 4 chapters. At the beginning, in chaper *Problem description*, we describe Testbed workplace and all the used hardware and components which we worked with. We point out specific problems of the components we later address in the implementation part.

In chapter *Fleet management system design and implementation*, we describe the proposed architecture of the Fleet Management System. We describe all it´s components and how they are interconnected and what communication protocols they use. We describe multi-agent interface. We also describe how to install KUKA Navigation Server which is part of the assignment.

Then subsequently in chapter *Test of the system*, we test the implemented system with a set of tasks, which order transportation of goods with the Testbed workplace. For some tasks, we artificially introduced obstacles in specific locations of workplace. For those tasks, We test the system in two separate states. We observe behaviour of the system when there is information from real-time location tracking system available. Then we observe behaviour of the system when there is no information about the tracker location.

In the last chapter *Conclusion*, we conclude results of our work and present possible future work.

Chapter 2
Problem description

This chapter describes the workplace where we implement proposed application. Chapter contains description of used hardware in warehouse. There is also description of KUKA Navigation solution together with description of used KUKA mobile robots. We describe Simatic real-time location system from Siemens which is used as a source of location data for this application. Also multi-agent system concept proposed in Testbed is presented.

Some hardware shortcomings are also mentioned, which we must take into account during implementation of the system.

2.1 Testbed for Industry 4.0

The Testbed for Industry 4.0 is a department located on the ground floor of the Czech Institute of Informatics, Robotics and Cybernetics of the Czech Technical University building in Prague. In the next part of the work, we refer to it only as Testbed. Testbed was established in the summer of 2017 and serves as an experimental and development workplace where the latest technologies in the industry can be found, supplied by leading technology and industrial companies like KUKA, Siemens, ABB, Sick, Keyence and many more. The Testbed thus serves partly as a showroom. At the same time, it is possible for those interested from a number of technological and industrial companies, to collaborate on their solutions using available hardware and software with the assistance of researchers who work here.

In Testbed, there are several production lines that serve to demonstrate flexible production and the principles used in Industry 4.0. Floor plan of Testbed is shown in figure 2.1.

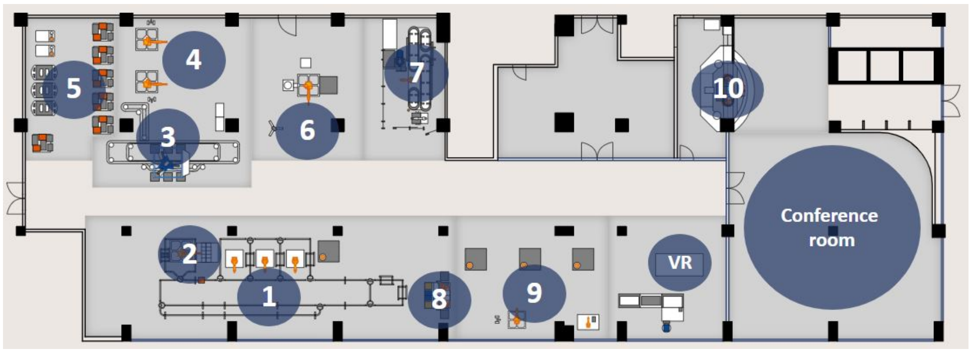


Figure 2.1: Testbed floor plan [5]

In the image, all the production lines are numbered and described as:

1. Montrac production line equipped with monorail conveyor system and 4 KUKA robots
2. Montrac automatic loading robotic station equipped with KUKA robot
3. Delta robot production line
4. Universal robotic cells equipped with 2 KUKA robots
5. Automated warehouse with a fleet of 5 KUKA mobile robots
6. Robotic multi-axis additive manufacturing cell which uses KUKA robot and Leica laser tracker
7. ABB assembly line for flexible fast production equipped with 4 ABB robots
8. Cell for assisted assembly with 2 KUKA collaborative robots
9. Robotic vision cells, each with KUKA robot
10. Robobar

Between these production lines, the warehouse serves as the primary tool for storing and transporting products, goods and materials. That is because it is equipped with KUKA mobile robots. In addition to the warehouse's mobile robots, Testbed also has a Bettaroe mobile robot used for testing the delivery of shipments from Robobar. The third mobile robot in Testbed is Festo Robotino, which works within the CP Factory production line, which is located near VR station. These two robots are not yet used within the warehouse system framework.

2.2 Warehouse hardware

Two types of KUKA mobile robots are used to move products within the Testbed workplace. The warehouse consists of parking spaces and charging stations for these robots. Also, warehouse is equipped with seven shelves which can contain different materials. Warehouse area is shown in figure 2.2.

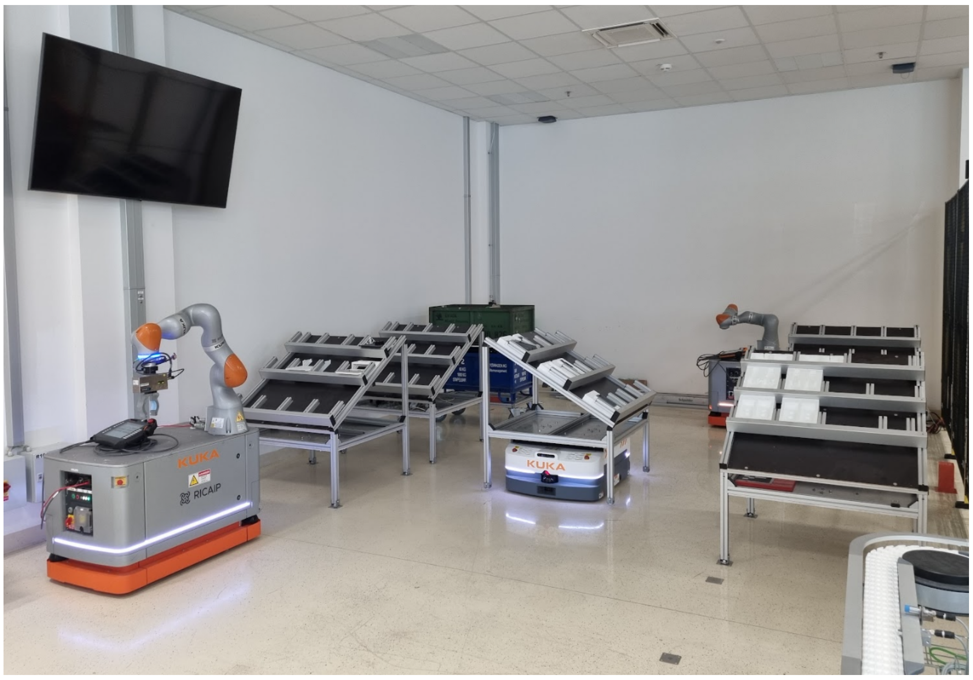


Figure 2.2: Testbed Warehouse area and equipment

2.2.1 KUKA mobile robots

Mobile robots, or otherwise called also AGVs, are robotic systems designed to move autonomously within defined environment. They are equipped with onboard computers and sensors using which they can locate themselves, navigate and perform specific tasks.

KUKA KMR IIWA

The first type of the robot present in Testbed warehouse is KUKA KMR IIWA. This robot consists from KMP200 mobile platform on which the KUKA LBR IIWA 14 R820 robotic arm is mounted [6]. Robot is shown in the figure 2.3. This robot is mainly used for transporting of small amount of work pieces around Testbed, using its gripping device.

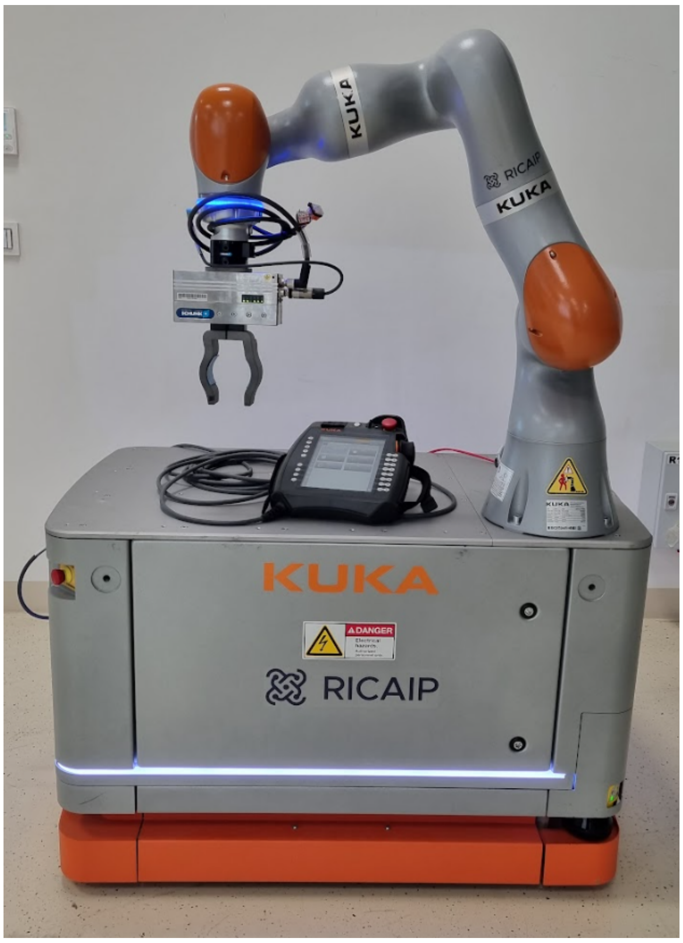


Figure 2.3: KUKA KMR IIWA robot

The platform KMP200 has omnimove technology, which allows it to move in x, y and rotational axis at the same time [8]. Even from standing start. This technology allows the robot to easily position itself in space. Also with high precision of ±3 mm. Robotic platform can carry up to 200 kg of total weight. The robotic platform is equipped with two laser scanners that provide a 360-degree view of the robot's surroundings. Scanners can evaluate obstacles at three different distances. Based on the data from the scanners, the robot adjusts its speed according to the distance from obstacles. At the same time, based on the data from the laser scanner and wheel odometry, it is positioned in space on map in which robot moves. Thus, the simultaneous localization and mapping process takes place. As a result, it can avoid obstacles and navigate in space independently.

Robotic arm has load capacity of 14 kg and a reach of 820 mm. Pose repeatability of this robot is ±0.15 mm. The IIWA robotic arm is equipped with force torque sensors in each of the seven individual joints. This means that this robot can work in collaborative mode, which makes human-robot collaboration possible. Reason for this is that robot can work in a mode when it sense extensive force applied to the body of the robotic arm. If the force exceeds certain threshold, robot will stop from safety reasons.

The robotic arm is for our application equipped with a Schunk EGI 80 electric parallel gripping device [9]. Gripper is mounted on flange of the KUKA IIWA robot as shown in figure 2.4. Gripper can be controlled for certain position and gripping force. His maximum gripping force is up to 80 N. The fingers of the gripper are printed on a 3D printer. The fingers are capable of gripping individual objects. The fingers can be changed manually depending on application and on what products the robot has to carry. It is also possible to change whole gripper. That is thanks to the manual gripper exchange mounting.

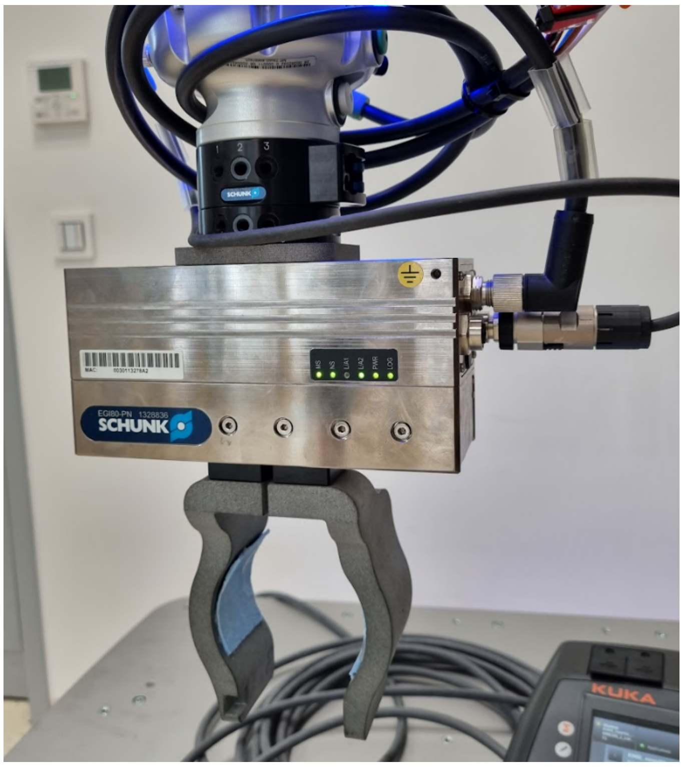


Figure 2.4: Schunk EGI 80 gripper with 3D printed fingers

Gripper is powered from robot battery. Gripper is connected to the robot controller through PROFINET as a slave device, thus it is possible to operate gripper from the robotic program.

KUKA KMP

The second type of robot is KMP600-s [7]. It is a robot equipped with two drive wheels and four support wheels. The robot can move in x direction and also rotate on spot. The robot can move at 8 different speeds depending on how far it is from the obstacle. Maximum acceleration is 1.25 meters per second and maximum speed is 2 meters per second. Positional accuracy without use of magnetic strip or any other assistant markings is ±10 mm. The mobile robot KMP is equipped with a lift. The lift can rise up by 60 mm in less than three seconds and it can lift up to 600 kg [7]. Robot battery has capacity to operate for eight hours and it can be charged under two hours. Charging is done through plug, because it is faster. Charging can be done autonomously. This type of robot is used to move entire shelves as it is shown in figure 2.5. Shelves are being moved between the warehouse and between individual production lines.

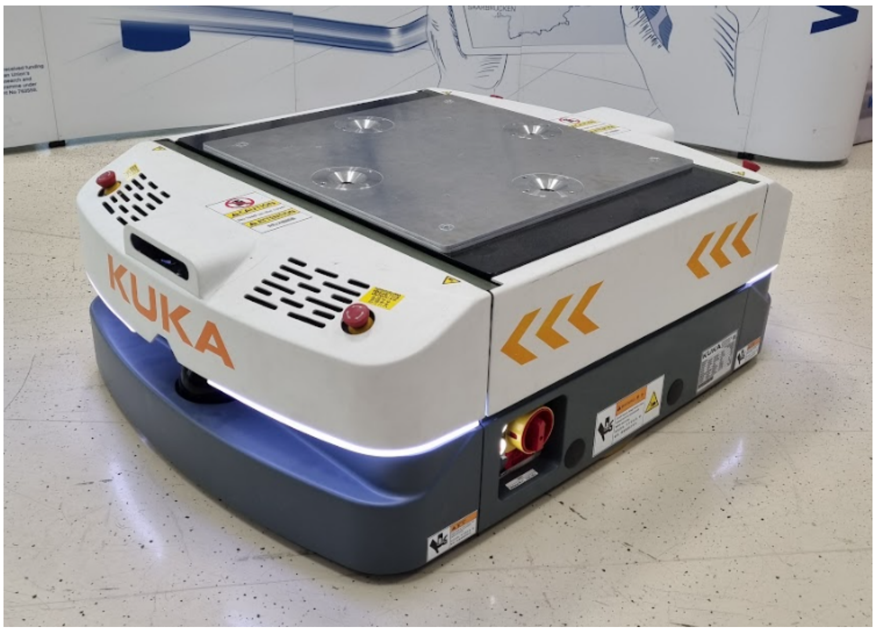


Figure 2.5: KUKA KMP600-s with additional positioning plate mounted on lift device

The positioning accuracy of the KMP robotic platform is not good enough for the precise placement of the shelves into the robotic lines. Thus we need to come up with some algorithm, which can increase precision of positioning of the robot. We also need to implement algorithm for automatic charging of KMP in case of low battery. Fleet Manager needs to check the battery level on each of the AGVs. Otherwise there is a risk that robot batteries will be excessively discharged and damaged.

2.2.2 Shelves

Shelves are used to store materials of various types. There is total of seven shelves in warehouse. The shelf has a square base with the length of the side 100 cm. The height of the shelf is 115 cm. On a plane inclined at 30 degrees, there are 3 levels which height can be adjusted according to the needs of a particular application or product dimensions. The shelf is equipped with a spherical element located on the underside of each of the legs. This feature allows for manual repositioning of the rack within the Testbed, for example by a human worker. Shelf is shown in the figure 2.6.

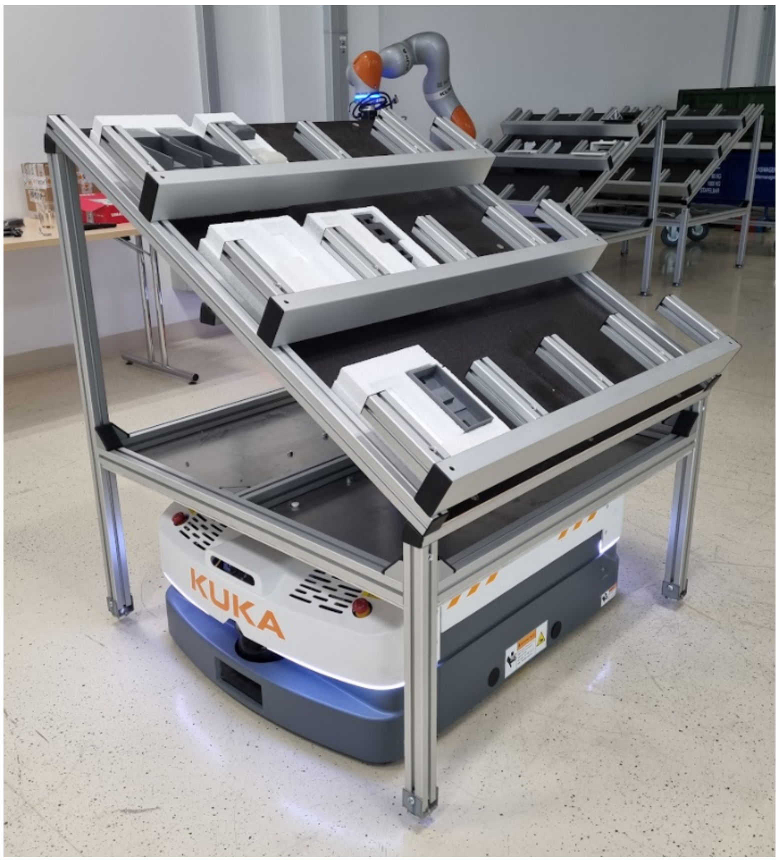
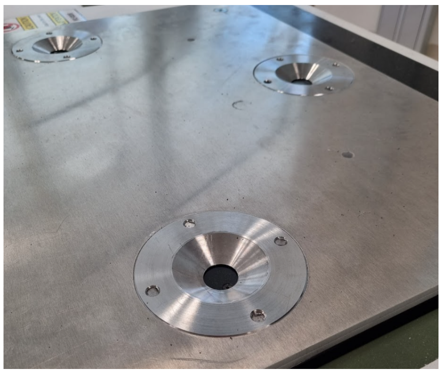


Figure 2.6: KUKA KMP with shelf on top of it

Precise positioning of the shelves is very important. Better to say, we have to ensure that the shelf is always placed in the same place within the production cell. If the shelf is not placed at the same position, production cell has to always detect trays of materials through some camera system or other position detection algorithm. For precise placement of the shelf in the production line, first we need to be able to pick it up repeatedly in a same manner. Fortunately, this precise picking up of the shelf problem was solved during design of shelves. Shelves are equipped with centering cones with a radius of 60 mm and a depth of 20 mm. Cones are placed on the bottom plate of the shelf. The centering cones together with their counterparts located on the AGV KMP mobile robot serve to eliminate or increase the tolerance of positioning inaccuracy of the KMP mobile robot when picking up the rack and placing it to the position on the top of the AGV. Centering cones system is shown in figure 2.7.


(a) : Opposite part for centering cone placed on KUKA KMP

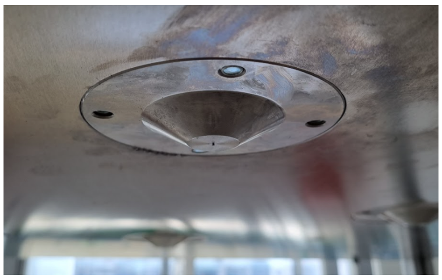

(b) : Centering cone placed on the bottom part of the shelf

Figure 2.7: Centering cones system

The shelves in warehouse are placed at precisely specified and numbered places. The places are defined by four centering pads fixed on the floor. Centering pod is shown in figure 2.8. Dimension of the pods is 80 mm by 80 mm. Hole in the middle has diameter 20 mm.

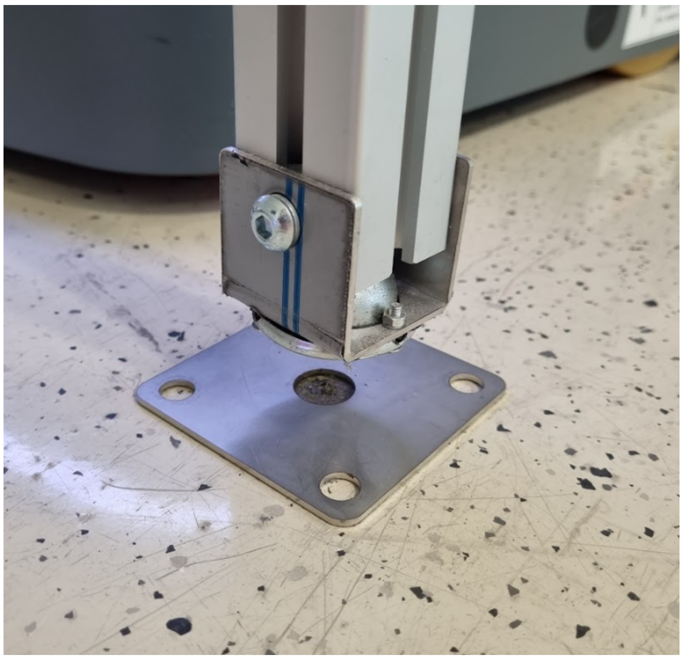


Figure 2.8: Plate which fix position of the shelf on the ground

This centering is also important when we want to precisely place the shelf to the certain location within the workplace. Thus centering metallic pods are placed in every location of the Testbed, where the delivery of the shelf is possible.

The accuracy of the KUKA KMP robot when positioning itself is ±10 mm.

When lifting the shelf, this accuracy is sufficient, because thanks to the cones we get a tolerance of ±30mm. The problem we have to solve is to ensure a fixed position of the rack after it is placed. The basic solution is the use of metal centering pads, see figure 4. This solution fixes the shelf, but it still has ±2mm freedom of movement and tolerance. This inaccuracy can have an adverse effect on the robotic application that subsequently removes or adds goods to the shelf. Also, we need to increase the precision of the KUKA KMP robot positioning. That is because the hole where we try to put the shelf provides tolerance ±10mm which is not sufficient. Thus we have to increase the accuracy of the robot in a different way, for example programmatically. Another possible issue to solve is, that it is hard to create shelves in a way when all of them have perfectly same dimensions. This leads to the impossibility of creating hard-coded approach to program which access the positions when picking and putting parts into the rack.

2.3 KUKA Navigation Solution

The KUKA Navigation Solution [10] takes care of the physical control of the movement of KUKA mobile robots. This solution enables the autonomous management of a fleet of KUKA mobile robots. It also enables the resolution of mutual robot collisions and functions as a tool for programming the logic of the robots movement in space. For navigation, it uses an offline map of the workplace and then data from laser scanners and the status of individual mobile robot. It does not require any cameras or other markings in the space such as magnetic strips. The system locates robots on the map based on data from scanners and odometry from sensors in the wheels. The system provides a JAVA API interface. Programming of the logic takes place in the JAVA programming language, within the eclipse-based Sunrise Workbench environment [11].

In order to be able to programmatically control and communicate with robots, we need to install the so-called KUKA Navserver application.

2.3.1 Navserver installation options

The basic and easiest option for operating this Navserver application is to install it on one of the computers located on the KUKA KMR robot. KMR is equipped with its own WLAN network and this computer is connected there as well. The disadvantage of this installation is that in order to communicate with the Navserver, all other robots must be in close proximity to the KMR robot where the Navserver is installed because they have to be able to connect to its WLAN network. The advantage of this approach is that in the case of using a single KMR for a application, this method is sufficient and much easier for installation. Architecture of this standalone setup is shown in figure 2.9.

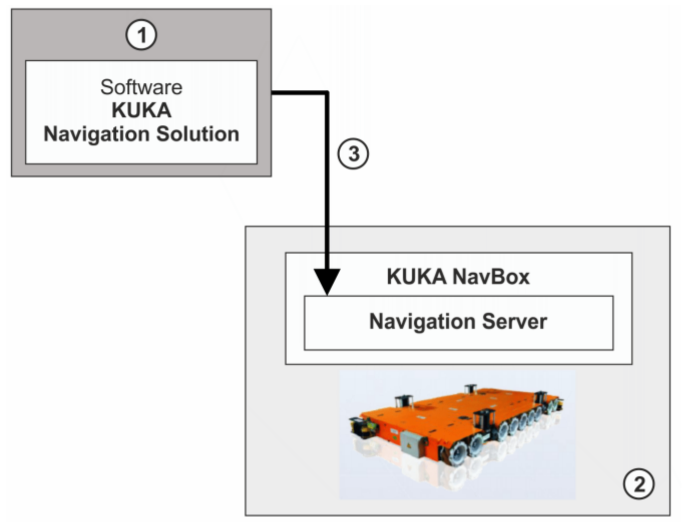


Figure 2.9: Architecture of standalone Navserver installation

Where specific components are:

1. Computer where Sunrise Workbench with Kuka Navigation Solution is installed. This computer is also used to change and program code for NavServer.

2. LAN or WLAN connection to virtual computer where NavServer is installed

3. Virtual computer with the NavServer installation

4. WLAN connection to the robot NavBox

5. NavBox of the KUKA KMR robot which collects and sends data to the Navserver. It serves as interface between Navserver and robot control computer.

6. NavBox of the KUKA KMP robot which collects and sends data to the Navserver. It serves as interface between Navserver and robot control computer.

The second and more advanced installation option is the installation of the Navserver on a virtual computer, or a server. In our case, virtual computer or server should have access to the WLAN network of the Testbed workplace. This approach is more complex to install and requires software changes in each AGV connected to the Navserver. But it provides the necessary flexibility in the case of the dimensions of the Testbed workplace. This means that this installation is necessary in our case. Architecture of this advanced setup is shown in figure 2.10.

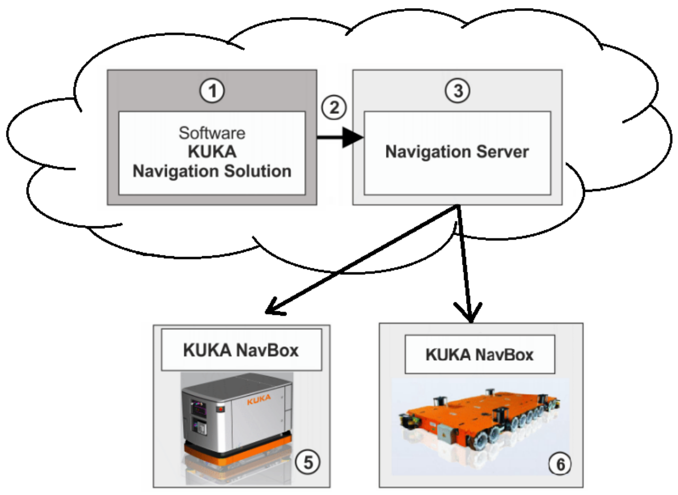


Figure 2.10: Architecture of WLAN Navserver installation

Where specific components of architecture are:

1. Computer where Sunrise Workbench with Kuka Navigation Solution is installed. This computer is also used to change and program code for NavServer.
2. NavServer is installed within same robot computer where the NavBox is installed.
3. LAN or WLAN connection from user computer to the robotic system.

The Sunrise Workbench software [11] is used for the NavServer installation. During the installation on remote computer, we need to provide enough computing power when creating virtual computer instance. Also, we need to address windows sharing problems, create PostgreSQL database and configure it. Process of installation is fairly well described in documentation, but some action or steps, which may be intuitive, are not mentioned. Thus, first installation of software may be difficult and time demanding.

2.3.2 Map creation

Navserver uses a static map to control robots. This map is created offline using Navserver and a KUKA robot equipped with at least one laser scanner. In our case, both KMR and KMP robots meet this condition. Due to the location of scanners on individual robots, it is more handy to use a KMR type robot for scanning the map, because KMP robot has bigger blind spots. But mainly because of KMRs ability to move in two directions at once thanks to OmniMove technology [8].

The map is then generated by driving the robot around the space and actively recording and storing data from laser scanners. The Navserver map creation algotithm then uses the iterative closest point algorithm to reconstruct the map based on data from the laser scanner data which are stored in XML file. Exported map from Navserver Map interface is in .omap2 format. Default map stored in the Navserver is shown in figure 2.11.

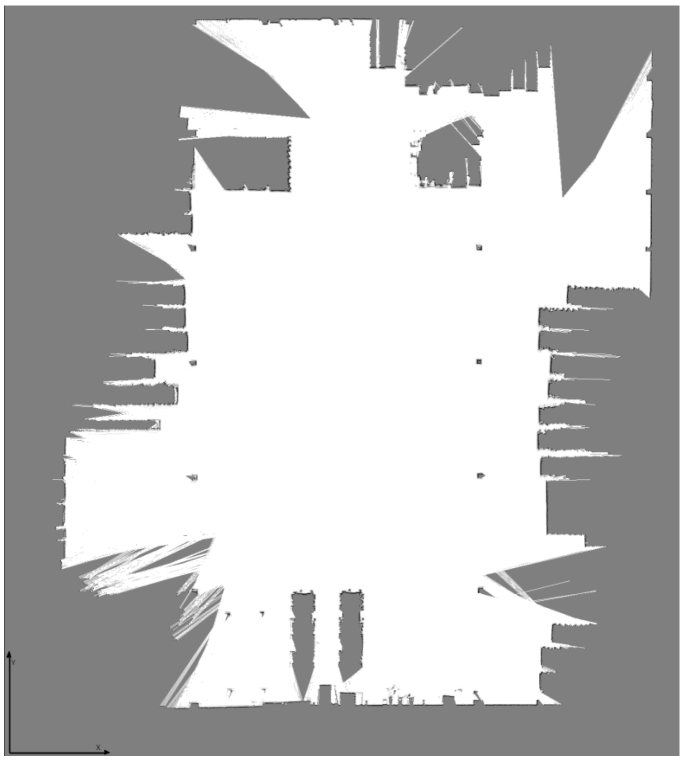


Figure 2.11: Default Navserver map

Black points in the image represent walls or obstacles. Gray points in map represent unknown points. Those are points which scanner did not reach during scan. That´s because of the wall or obstacle which blocked the laser beam, or because that space was not traversed during scan. White point represent free space. Map has a orientation which can be defined during map creation. In interface, map is shown with orientation as: X-axis points from left to right and Y-axis points from down to up. Parameters of each map are:

- Name of the map
- Resolution
- Width
- Height

We need to be precise during creation of the map. Especially around places where we want to achieve precise positioning. That means, that we should traverse that specific area from different directions. Preferably at low speed. Map represents state of the workplace as it was during the scan. When something important changes, is added or removed in the map, it is advised to scan the workplace from scratch and create new map. Map does not represent or show obstacles which are put there temporarily. When we try to navigate and control robots in non-actual map, it can lead to inability to achieve desired positions and in high inacuraccy in positioning of the robot.

2.3.3 Robot motion options

With the standard Sunrise Workbench software package, AGVs can move within a topology graph that is defined in map. In the case of using KMR, it is possible to program movement even along pre-set line segments. Again, thanks to the KUKA OmniMove technology. A sample of the graph created in the map can be seen in the image 2.12. The graph is created in the system through the Sunrise Workbench interface using the KUKA Map perspective window. For the possibility of creating a graph, there must be a map stored in the NavServer. During creation of the graph, its data is stored directly in the Navserver database, from where it is the loaded everytime some service needs it.

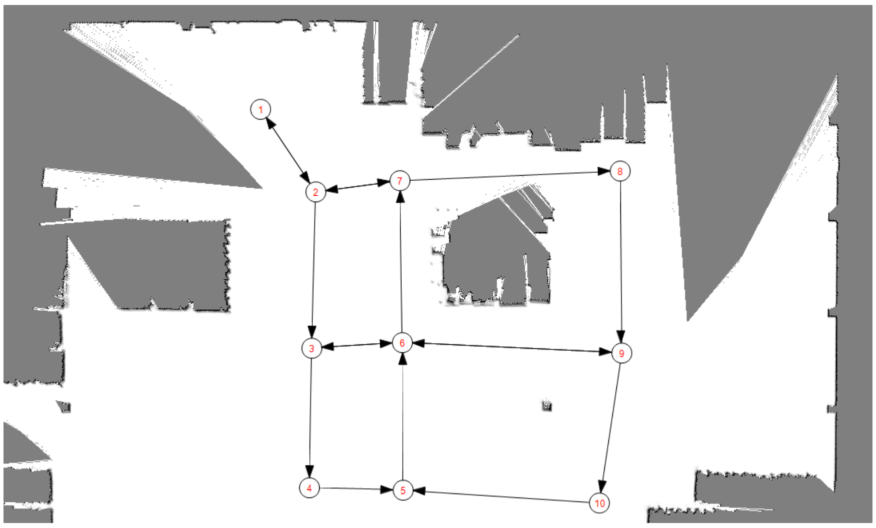


Figure 2.12: Example of graph created in default map

A topology graph consists of vertices and edges. The location of the vertices and their interconnection using edges is defined by the user. Each edge represents one-way connection between two nodes. Both, vertices and edges, have definable parameters, with the help of which it is possible to specify the type of movement, speed, orientation of the robot when moving along the edge and more. The specific parameters are as follows.

Some of the important parameters for the nodes setting:

- Accuracy - sets attribute for how precise positioning of the robot at this position should be

- Position - exact x and y coordinates of node in map
- Orientations to map - sets possible orientations of the robot with respect to the map which are allowed for the robot in this position
- Velocity and acceleration parameters - it is possible to set limits for maximum velocity and acceleration which will robot have when it will be moving through, from or into this node
- Restriction - changing this parameter, we can black-list specific node and prevent robots from visiting it
- Waiting zone - it is possible to set time which robot should wait at specific node

Available parameters for the edges setting:

- VirtualLineMotion - parameter specifies the motion and orientation which will robot use when traversing along edge
- Velocity and acceleration parameters - it is possible to set limits for maximum velocity and acceleration which will robot have when it will be moving along edge
- PathRecover - parameters sets how far can robot traverse from the edge during the motion
- Restriction - changing this parameter, we can black-list specific edge and prevent robots from visiting it
- Weight - it is possible to add weights to individual edges and thus influence the path planning algorithm

When there is a graph defined, it is possible to program an application which will control robot motion within this graph. In reality, fine tuning of position of nodes and edges is necessary. That is because each node represents specific point in real workplace.

Since the graph is stored in the NavServer database, it is possible to bypass its creation via the KUKA Sunrise Workbench and create it there directly. For example, via a program which will generate and import graph data to specific database table. This approach of dynamic graph creation could save a lot of time in case, we have a highly variable environment and we often need to do mapping and graph creation. Also it is possible to create graph grids which can cover whole map which can lead to higher flexibility in robot motion and path programming.

2.4 Real time location system

RTLS, short for real time location system, is a technology that is used to locate and track objects, people, objects or material in a specific area. RTLS is mostly used in indoor spaces, most often in factories, in logistics, but also in retail and healthcare. This form of tracking provides real-time position data that can be used to increase production efficiency, increase workplace safety, and more.

There can be many different technologies used for tracking. The most commonly used technologies in RTLS are based on active radio frequency identification, infrared, optical localization or bluetooth signals transmission and receiving[13]. RTLS system usually consists of a physical gate located at fixed position in area where the tracking and localisation should be done. Gateways act as signal receivers, or also as signal transmitters. Next, there are tags, transponders or trackers that are portable and are placed on objects or people that we want to track or locate. The algorithm on the RTLS server continuously processes data from tags that are received through the gateways. Signal of one tracking tag usually have to be received by more than one gate in order to determine its position. RTLS evaluates data from gateways and determines the position of the tags in workspace area in real time. Then RTLS server can visualize the data in the application, or enable access to data via some API interface.

One of the most common used technologies in RTLS is Ultra-wideband technology [15]. Ultra-wideband technology uses radio bandwidth to transfer data and communicate on short distances. It uses high frequency bandwidth - more than 500 MHz. One large commercial field of use of this technology is in smartphones and smart locators like Apple AirTag or Samsung SmartTag. These trackers can communicate with other devices of the same brand. Thus mobile phone or notebook acts as a gateway and smart tag acts as tracker. Since these brands are spread all over the world, we can track the tag practically anywhere where there are a large number of people with smartphones. The problem is that it is a closed ecosystem of one brand and it does not allow third parties to read data from this system. Also, it is not considered as industrial grade technology.

The SIMATIC RTLS system from Siemens [14] is implemented in the premises of the Testbed. In case of the SIMATIC RTLS, system provides a localization accuracy of 20 cm to 30 cm. But its required to have quite dense network of gates in area, where the localisation is carried out. Siemens SIMATIC RTLS consists of two main parts, the hardware part and the software part. Their architecture and interconnection is shown in the figure 2.13.

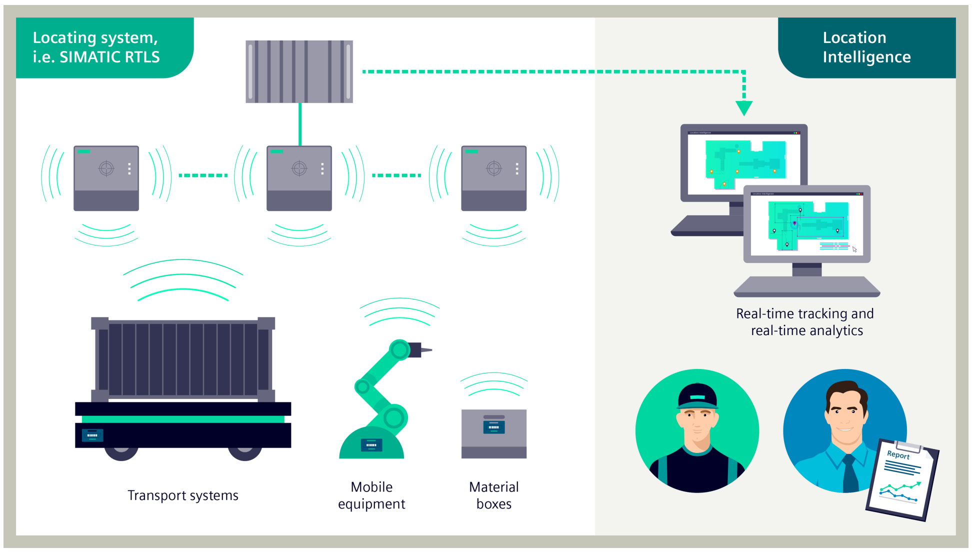


Figure 2.13: Architecture of Siemens Simatic RTLS [14]

On the left part of the image 2.13, there are the hardware parts of the system, specifically gateways and tracking tags placed on the devices. Gateways gather the data from tracking tags and send them into the Siemens Location Manager component. This is the software component of the RTLS solution which computes the real-time position of the tracking tags. This position information can then be cyclically sent into higher control systems or is available for program access through the API interface. Together, the software and hardware parts make up Location Intelligence [16].

2.4.1 SIMATIC RTLS transponders

There are two types of trackers used in Testbed. Trackers are shown in the figure 2.14.

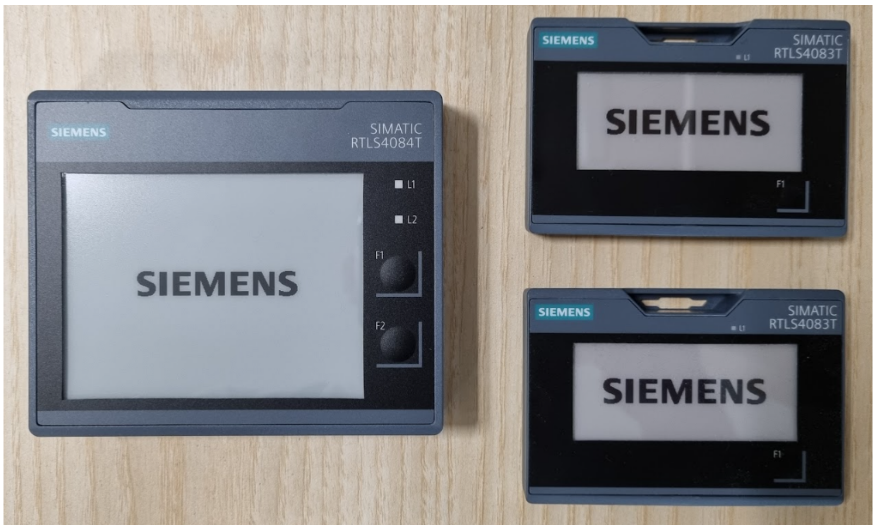
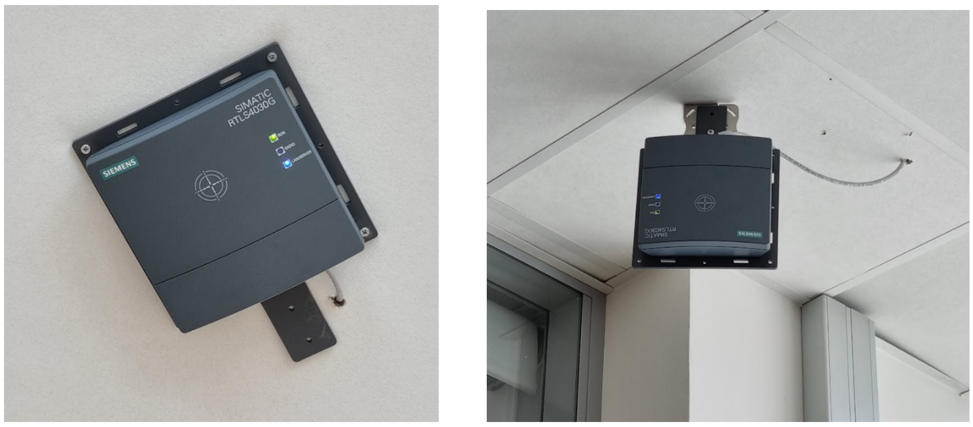


Figure 2.14: Simatic RTLS tracking tags RTLS4084T (left) and RTLS4083T (right)

There is thirty trackers of type RTLS4083T and ten trackers of type RTLS4084T available in Testbed. Main difference is in the size of the ePaper display. Other big difference is in battery. RTLS4084T transponder is powered by external battery pack and in standby mode can last up to 8 years or with 1 second localization cycle up to 18 months. RTLS4083T transponder has built-in rechargeable Li-ion battery. For one charge it can last 1 year in standby mode or with 1 second localization cycle up to 6 months. Both transpodners can show data on display. The more often is display information changed, the faster the battery is drained. Display, can show information such as object id, product number, volume information and tag address. Accuracy of localization of both transponders should be ± 0.1m.

2.4.2 SIMATIC RTLS gates

There are nineteen gates distributed in the Testbed in a way, so that they have the best signal in the places where AGVs move most often. The gates are placed on the ceiling in horizontal or vertical positions. Gates and their mounting is shown in figure 2.15. RTLS gateways SIMATIC RTLS4030G are configured to receive data from tags. At the same time, they are connected to Testbed's local network, where the Siemens Location Manager module is installed on the server. An API interface is exposed on this server, which allows access to location data and status of individual tags.



(a) : Horizontal mounting of the gate **(b) :** Vertical mounting of the gate

Figure 2.15: Mounting of the RTLS gates on the ceiling of the Testbed

Data is available through API. In the Fleet Manager system implementation, we need to read this data through API appropriately. At the same time, it is assumed that we have to filter this received data and approximate the position of the locator tag. The position received from the trackers will probably not be static even if the tracked object or person is not moving. This is based on the given accuracy of the trackers which is expected to oscillate around real value. A lower accuracy during tracking can occur due to wrong gateways configuration. This situation has occurred in the past and the setting of the gates have been tuned repeatedly.

2.5 Multi-agent system

Multi-agent systems (MAS) are composed of multiple entities known as agents. These systems can handle complex tasks by breaking them into smaller subtasks, which are then managed by individual agents. To solve problems, agents collaborate together within a shared environment, exchanging information and resources while making decisions in a decentralized manner. Notable traits of MAS systems include their decentralized decision-making, adaptability, robustness, and capacity to handle errors. They can solve complex tasks very effectively.

The application of MAS is particularly prominent in robotics, where an agent can represent a single robot in a swarm of robots. These robots then can collaborate to achieve goals such as exploring uncharted terrain or conducting search and rescue operations. Another significant area for MAS is in manufacturing and industry area. Here, agents can represent distinct production cells, individual machines or even whole factories. With MAS system, optimizing production efficiency, enhancing workplace safety, preventing machine breakdowns, and supervising warehouse robots becomes feasible.

At Testbed, we use the MAS concept for demonstration and experiments with distributed production. The main demonstration of MAS capabilities lies in smart factory and smart production, which can manufacture a product taking into account the optimal production time, manufacturing quality, etc. Recently, there have been tests and experiments on an RC car model made from 3D printed parts. The model contains soldered board designed by colleagues at Testbed. Car is powered by a battery and thus it is a realistically functional model that demonstrates the principle of distributed production. The multi-agent system can currently manage production within one robotic cell. The plan is to make it possible to manage production across all production lines in Testbed using the principle of the MAS.

The core components of the Testbed multi-agent platform are device/machine agents. Each device is represented by its agent in the group of cooperating programs. Agents have the ability to activate particular machine actions or operations by using the machine interface. The agentsmain structure follows the Enterprise Model-View-Controller (MVC) pattern [18]. Through the Advanced Message Queuing Protocol (AMQP) controller[19], the agents can send, receive, and manage messages. Each agent is dockerized. Architecture of the agents and their communication through AMQP is shown in figure 2.16.

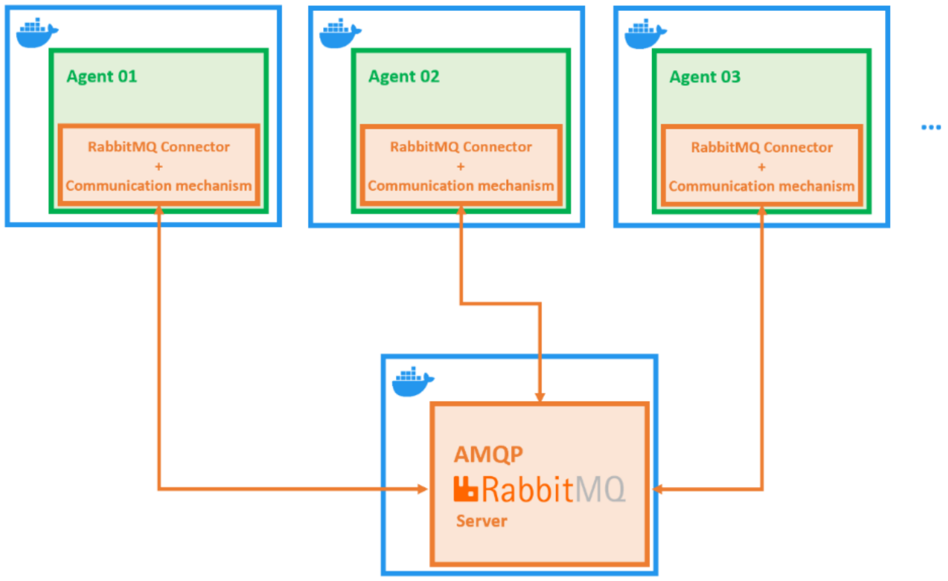


Figure 2.16: Dockerized multi-agent communication architecture

These messages are then managed in a way where they're processed independently by agent´s state machines. State machines are called strategies. The decision-making process operates without needing to remember past decisions. All the information related to the current state and conditions is continuously stored and extracted from a database. Architecture of single agent instance is shown on image 2.17.

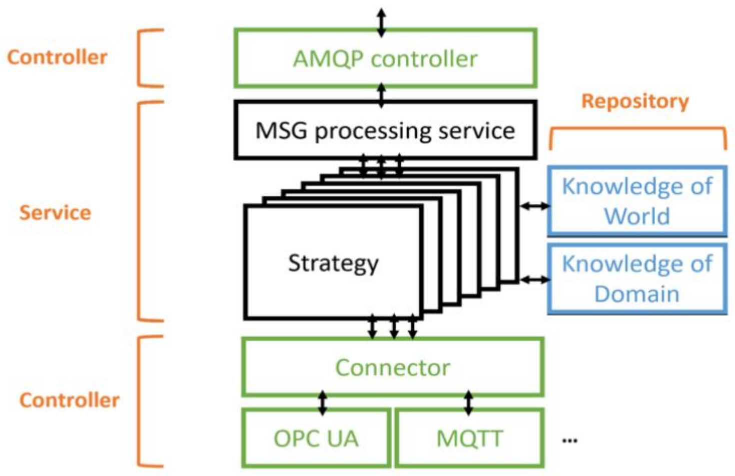


Figure 2.17: Architecture of single agent

Every agent is enrolled in the agent registry known as the directory service. Besides its own identity, an agent also enlists the abilities it intends to provide to others. This registry for agents operates as a ZooKeeper instance [20]. Representing a product order, there is a product agent. This agent employs the Plan-Commit-Execute protocol to collaboratively determine the optimal conditions for assembling the product. The agent continuously communicates with the physical device using OPC UA interface [21]. There is also option to communicate through alternative options, like MQTT interface. The entire communication among the multi-agent setup is supervised through the GrayLog [22] logging platform.

The warehouse system forms an important link that provides the transportation of material between production lines. Therefore, it is necessary to create an agent that would represent the warehouse system in existing multi-agent ecosystem. That means, agent must be able to communicate transportation capabilities of warehouse system to other agents and at the same time it must be able to provide transportation management using AGVs to other agents who need it. In this work, we deal with the design and implementation of this agent.

Chapter 3

Fleet Management System design and implementation

This chapter is about specifics of design and implementation of Fleet Management System which incorporates all the Testbed hardware, software and principles mentioned in previous chapter.

Fleet Management System is supposed to ensure transportation functionality within the Testbed. Transport should be possible between the local warehouse and between the production lines. FMS system should be able to connect to Kuka Navserver and control it in order to fulfill transporation task given from superior control system. Therefore, when designing the system, we must take into account its easy expandability and flexibility. When designing the architecture, we must consider the creation of a multi-agent interface that can also forward orders for the transport of shelves and goods. The system therefore contains both an interface for AMQP communication and an interface for communication through OPC UA.

When implementing transportation function in the Testbed premises, we consider transport options between the warehouse, the Montrac line, the Delta line and the ABB line. Each of these lines has its own characteristic place for docking of the shelf into the line. This means that it is necessary to design the docking and undocking process for each production line separately. This is due to the safety measures installed on individual lines.

Then we further expand the created FMS with data input from the RTLS system installed in Testbed. This will add the possibility of optimizing the shelf transport system beyond capabilities of Kuka Navigation Solution alone. This is done by enabling the manual marking of obstacles in the Testbed workplace by RTLS transponders. Then information about the obstacle is transmitted to the FMS system which can work with it and propagate it to the Navserver, which then plan collision free routes for the AGVs.

But first of all, we need to install Kuka Navserver.

3.1 KUKA NavServer installation

In order to use the Kuka Navigation Solution, we must first install the Kuka Navserver. We install Navserver on a virtual enviroment - computer instance - initialized in local VMWare in Testbed. This virtual computer must meet the following requirements:

- Windows 10 LTSC version
- 4 cores
- 8 GB RAM
- HDD size 50 GB and more

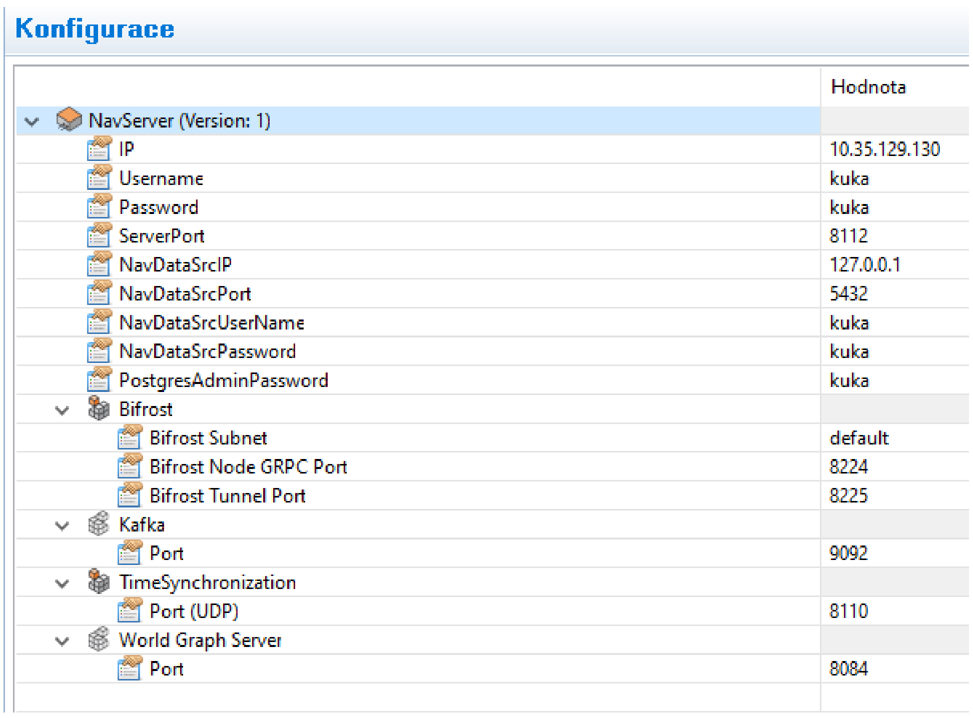
We pick IP address of the computer as 10.35.129.130. Computer has to be connected into the Testbed WLAN with name CIIRC Testbed Lab. We select the name of computer as kuka-navserver17.

Then we follow installation instructions given in KUKA KMR documentation [?]. Steps of installation of additional software needed to run Navserver consists of:

1. Installation of Java version 1.8
2. Installation of PostgreSQL database version 9.6
3. Installation of Visual C++ Redistribuable 32-bit version 2015 or newer

Then we need to create kuka user for the PostgreSQL database, because system is set up in a way, where it uses this username to access database. We change the config file of the database in a way so only specific ip addresess can access the database. From security reasons. We also create Kuka user in the Windows system, because during installation of Kuka Navserver project, the project is installed and files are copied from work computer onto the server using windows sharing option. And when trying to share, Kuka Sunrise Workbench is configured to use Kuka User on the target computer where project is being installed. We create NAV folder on drive C:/, where the Navserver project will be installed.

After the Navserver computer is ready, we connect to the same network with our work computer where we have KUKA Sunrise Workbench 1.17.0.5 installed. Using our workbench, we create project for Navserver and configure it. Configuration is shown in figure 3.1.



Konfigurace

	Hodnota
NavServer (Version: 1)	
IP	10.35.129.130
Username	kuka
Password	kuka
ServerPort	8112
NavDataSrcIP	127.0.0.1
NavDataSrcPort	5432
NavDataSrcUserName	kuka
NavDataSrcPassword	kuka
PostgresAdminPassword	kuka
Bifrost	
Bifrost Subnet	default
Bifrost Node GRPC Port	8224
Bifrost Tunnel Port	8225
Kafka	
Port	9092
TimeSynchronization	
Port (UDP)	8110
World Graph Server	
Port	8084

Figure 3.1: Navserver Sunrise project settings

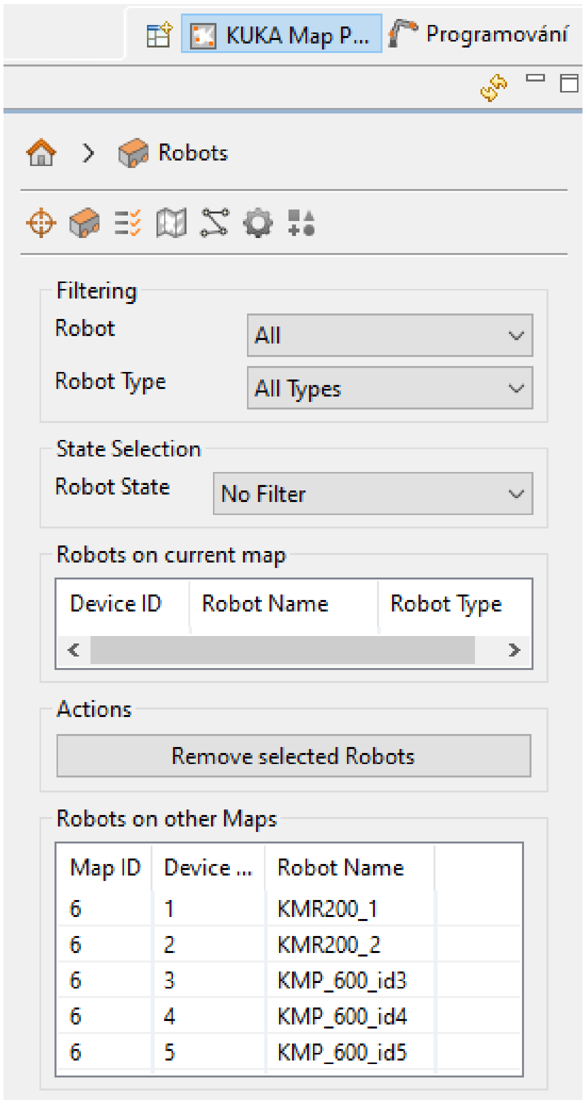
After configuration of Navserver, we need to reinstall every KMR and KMP project in a way so they connect to Navserver on IP address 10.35.129.130 after their startup. For this task, we again use KUKA Sunrise Workbench 1.17.0.5. We reinstall KMRs and KMPs and change their IP addresses and instance IDs in their project settings as it is given in table 3.1.

AGV Name	IP address	Instance ID	AGV Type
KMR200_1	10.35.129.5	1	KMR
KMR200_2	10.35.129.6	2	KMR
KMP_600_id3	10.35.129.7	3	KMP
KMP_600_id4	10.35.129.8	4	KMP
KMP_600_id5	10.35.129.9	5	KMP

Table 3.1: Table of KMR and KMP IP addresses.

In case of KMRs, we also need to reconfigure their Scalance Switches to the client mode. We also need to change IP address of the device based on table 3.1. For the both types of robots, we also have to enter and set up WLAN credentials and password of the network, where Navserver is installed. That´s because we need switches of every robot to connect to the same WLAN network where Navserver is, so they can communicate.

After installation of NavServer project and making changes in switches, we can restart Navserver and robots. After the restart, all the robots are connected to the Navserver. We can check it in the Kuka Sunrise Workbench, more precisely in the Map Perspective window in the *Robots on other Maps* section. We see it there, as it is shown in figure 3.2, so the installation of Navserver and reconfiguration of robots is successful.



Map ID	Device ...	Robot Name
6	1	KMR200_1
6	2	KMR200_2
6	3	KMP_600_id3
6	4	KMP_600_id4
6	5	KMP_600_id5

Figure 3.2: All the robots are connected to the Navserver

3.2 Testbed map

In order to be able to move and control robots within Testbed workplace, we need to add map of the actual enviroment into the Navserver. We will do it using work computer with Sunrise installed as it was in previous section during Navserver installation.

For scanning of the Testbed map, we use KMR mobile robot for reasons mentioned above in subsection Kuka KMR. In Sunrise we go to Map Perspective window where we lock the KMR200_1 robot and move it to the default map. Then we start the mapping proccess. We drive robot around Testbed workplace, slowly. We need to be carefull to allways change our position relative to the robot. Otherwise, during multiple of scans, it could see our feet in the same position as in previous scans, thus mapping algorithm would consider our feet as an obstacle in the area. We do more traversing around docking areas of Montrac, Delta and ABB line. Whole process takes around 15 minutes, because Testbed is relatively large workplace. In the end, we birng the robot back to its starting position and we stop the mapping process. After that, data from scanners are saved internally and we can download it as XML file, or we can directly generate map. We generate the map from data and we got the map shown in figure 3.3.

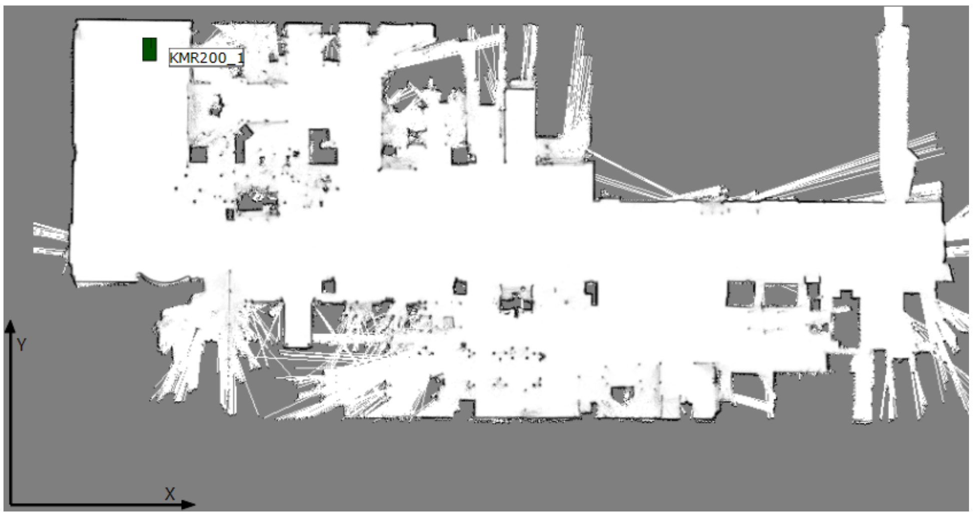


Figure 3.3: Map of the Testbed

Paramters of the map are:

- Name: 2023_05_23_TestbedAll
- Resolution: 0.O5
- Width: 1291
- Height: 631

We need these data to create transformation from a map which is used by RTLS.

In this map we create topology graph. This graph is used druing the transportation task of shelves from two precise location in warehouse or within production lines. Graph is also used for testing of the RTLS data effect onto the navigation of the robots. Used graph is shown in figure .

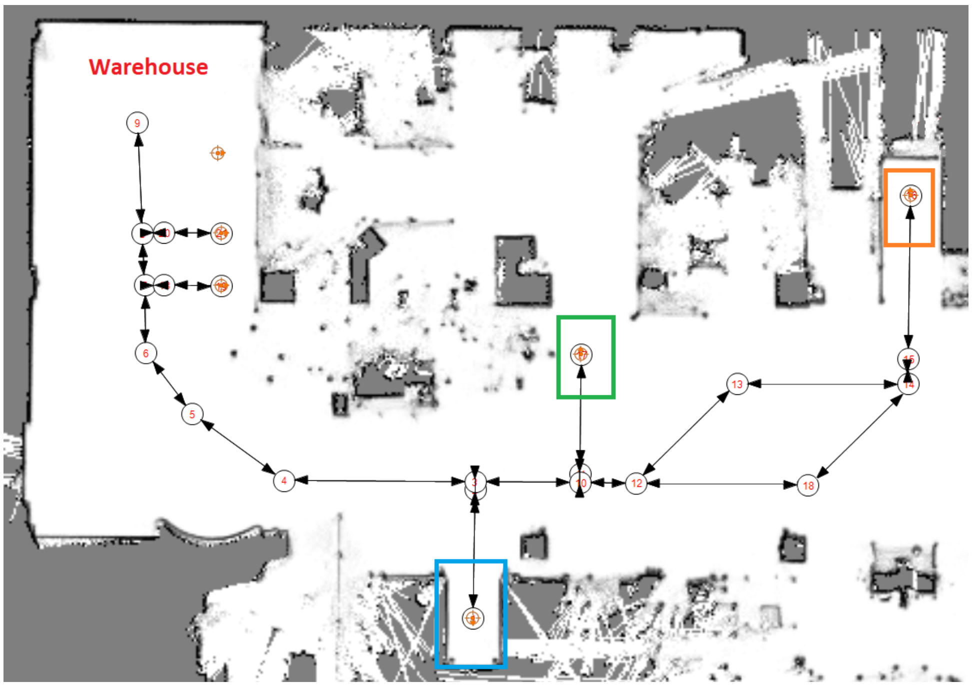


Figure 3.4: Map of the Testbed with specified graph

In the figure, there are also marked positions of docking stations. Blue one corresponds to the Montrac docking station, green one corresponds to the Delta docking station and orange one corresponds to the ABB docking station.

3.3 Shelves precise positioning

As it was mentioned before in the part containing shelf description, we need to solve the problem of inaccurate positioning of KMP robot in docking station. We can find out how imprecisely the robot is at the specified location using laser scanners. It is possible to create a location element within the map. This element shares most of the parameters that characterize node. However, it is possible to create FineLocalization data for this location element. This means that at a given location it is possible to create an accurate record of data from the robot's lasers about the surroundings of this location. This process requires recording of data from scanners, which lasts approximately 20 seconds. This location data is then stored in the Navserver database. Subsequently, as part of the program, we can navigate the robot to the node with same position as this location. That with low positioning accuracy. After reaching the position, robot can measure the current data from the lasers. Then the current data is programmatically compared with those that were learned and saved previously with high precision. The offset is expressed from their difference. Offset tells us how inaccurate the current KMP position is compared to the desired correct position. Offset expresses the inaccuracy in the X-axis, Y-axis and rotation angle. Then we subsequently correct this offset appropriately using algorithm implemented by us. Algorith uses basic relative robot motions, readings from laserscanners and saved FineLocalization data. Description of algorithm follows.

Considering that the KMP can only move in the x-axis forward and backward and turn, for the correction in the Y-axis we have to apply a movement similar to longitudinal parking. This means that the process of minimizing the Y-axis offset is iterative. Due to the small spaces of the docking stations, the robot can move a maximum of 5 cm in one iteration in the Y-axis. Movement in the Y-axis consists of going back, turning, moving forward and turning again by the same value of the angle, just in the opposite direction. After one iteration of correcting of Y-axis, measurement of offset is processed again. We iterate through this unti Y-axis offset is within desired standard. Then algorithm corrects X-axis offset by just moving forward or backward. At last, it will correct its angle offset. In this way, we can achieve the accuracy of placing the shelf in the range of ± 3 mm.

At warehouse, there is a long wall without any sharp edges. To be to teach FineLocalization data, we need to add some contrast edges. For that, we can use metalic block mounted to the ground as it is shown in figure 3.5. This block are used also in the Montrac docking line.

Last thing how we increase the precision of shelves on the ground is by adding 3D printed element to the centering pads on the ground. The centering element has the task of eliminating the inaccuracy of the positioning of the robot when parking at the place of placing or picking up the rack. Centering pad with mounted 3D printed element is shown in figure 3.6. This element was discussed and modelled by a colleague at Testbed.

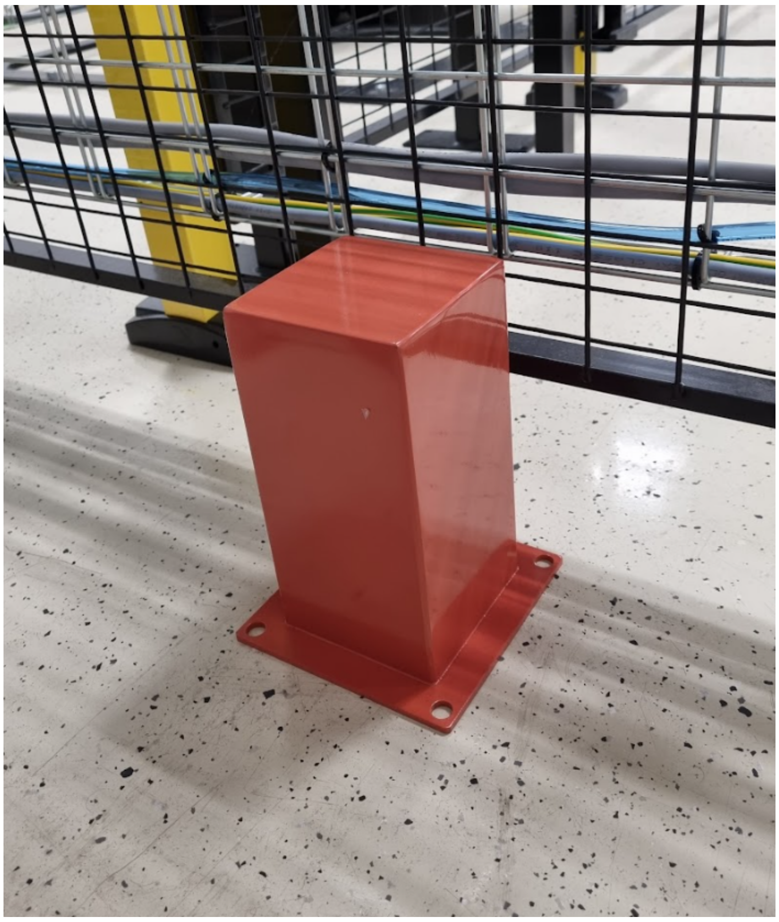


Figure 3.5: Map of the Testbed with specified graph

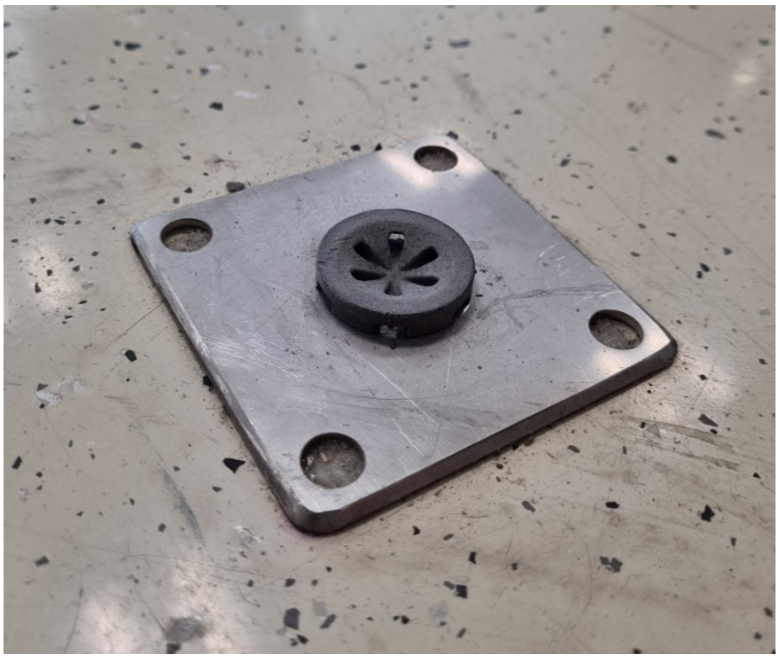


Figure 3.6: 3D printed part in centering pad

Using all those methods and improvements above, we get 100% success rate during picking and placing shelves to desired positions within Testbed workplace. Only thing that could affect this precision is sudden changes in docking positions for the shelves.

3.4 Docking to production line

In this section we explained implemented docking process on example of Montrac production line. Docking to ABB and Delta line is practically identical.

We describe the procedure and communication during the docking process. Docking means the process when the NavServer (the control system that manages the AGV - in our case specifically the KUKA KMP) navigates the specific AGV/KMP physically into to the docking station on the Montrac production line. Docking operation si possible with or without shelf. Montrac acts as a OPC UA Server with a given structure of directories and objects. The docking process itself is managed by the NavServer, which connects to the server as a OPC UA client. This is because the NavServer receives a request from its agent or superior system to deliver the goods/shelf to a specific station. The NavServer verifies its own delivery capabilities and, if resources are available, evokes the process of moving to the Montrac and the docking process itself. NavServer then successively calls methods on Montrac Server. The server processes the methods calls and provides output of methods to the NavServer. NavServer then processes the output of the individual called methods and, based on that, either continues with the next step of the process or terminates the process. In this variant of the communication design, we expect that the scanners at the entrance to the docking station will be able to distinguish the presence of a shelf in the given station. This means they can detect the legs of the shelf. Alternatively, they should be able to recognize the fact that there is another, unknown object in the station.

Communication Brief description of communication during docking and undocking. More detailed requirements for methods and data processing are described under individual methods. Docking Symbol -> describes direction of communication as "from" -> "to": 1) NavServer -> Montrac: determines the status of the station – NavServer calls the Montrac OPCUA server method RequestStationStatus 2) Montrac -> NavServer: returns the status of the station or information about the docking option through the RequestStationStatus method 3) NavServer: If docking is possible, it sends the AGV (or picks up the shelf and sends the AGV loaded with the shelf) to the predock position. If docking is not possible, it returns this information to the client/agent. 4) NavServer -> Montrac: The moment the AGV reaches the predock position, it waits and requests the docking - it calls RequestDocking with specific parameters. 5) Montrac -> NavServer: through the output of the RequestDocking method, it returns permission or denial of docking. 6) NavServer –> Montrac: requests a safety connection via RequestSafetyConnection. 7) Montrac -> NavServer: Montrac establishes a safety connection with the given AGV.

The connection status is returned by the output of the RequestSafetyConnection method. 8) NavServer –> Montrac: requests to switch the field of Montrac safety scanners for KMP entrance via RequestSafetyFieldEntrance. 9) Montrac -> NavServer: Montrac switches the array of scanners in case of correct contour detection. The status of switching scanners is returned by the output of the RequestSafetyFieldEntrance method. 10) NavServer: If the safety connection is successful and the array of Montrac scanners is switched, the NavServer sends the AGV to the station. If he has a shelf, he puts it down. 11) NavServer-> Montrac: confirms the placement of the shelf by calling RequestDockingCheck. 12) Montrac -> NavServer: Montrac scans the RFID and checks the correct placement of the rack with the scanners. Corresponds to NavServer. 13) Montrac – if RequestDockingCheck is TRUE, it detects the contour of the parked KMP and switches the array of scanners (cancels entry) 14) Montrac – checks the switching of the scanner array, if OK, and the RFID rack sensor is TRUE, it allows the movement of the robots 15) NavServer unlocks KMP on the NavServer side.

Undocking 1. NavServer -> Montrac: NavServer requests undocking of a specific type. Request Undocking with or without a rack by calling the RequestUndock method. The method must evaluate the state of the production line and thus verify the possibilities of undocking with or without a rack. If, for example, material is being removed from the rack, it is possible to dock the KMP without the rack (during the departure of the KMP (field of Montrac scanners for the departure of the KMP), however, the robots must always be parked and safely stopped). 2. Montrac -> NavServer: through the output of the method called in the previous point. Montrac will allow or deny the requested undocking. 3. NavServer –> Montrac: requests to switch the field of Montrac safety scanners for KMP departure via method RequestSafetyFieldExit. 4. Montrac -> NavServer: Montrac switches the array of scanners in case of correct contour detection. The state of active laserscanner field is returned as the output of the RequestSafetyFieldExit method. By switching the field, the security of the Agilus robots will be disabled. 5. NavServer: if the required type of undocking is enabled, the NavServer will send the AGV/KMP to the predock position (either with rack or without). 6. NavServer -> Montrac: When the AGV reaches the predock position, the NavServer calls the RequestSafetyDisconnection method, which requests disconnection of the Safety connection. 7. Montrac -> NavServer: Montrac disconnects the safety and switches the array of scanners (closes the passage). In case of successful disconnection of safety-connection between Montrac and a specific AGV and switching of scanner fields, Montrac will return information to the NavServer through the output of the RequestSafetyDisconnection method called in the previous point. Montrac will enable the security and movement of Agilus and Cybertec robots. 8. NavServer: after successfully disconnecting the safety connection, the NavServer unlocks the given AGV on the NavServer side.

3.5 Fleet management system architecture

Fleet management system is designed in a way so it can communicate with other

The system is designed to form a communication interface between superior control systems as MES or Multi-agent system. Those systems provides data inputs and task which are processed by Fleet managements system. FMS then delegate tasks to subordinate robotic systems. The connection of the modules is shown in the figure3.7.

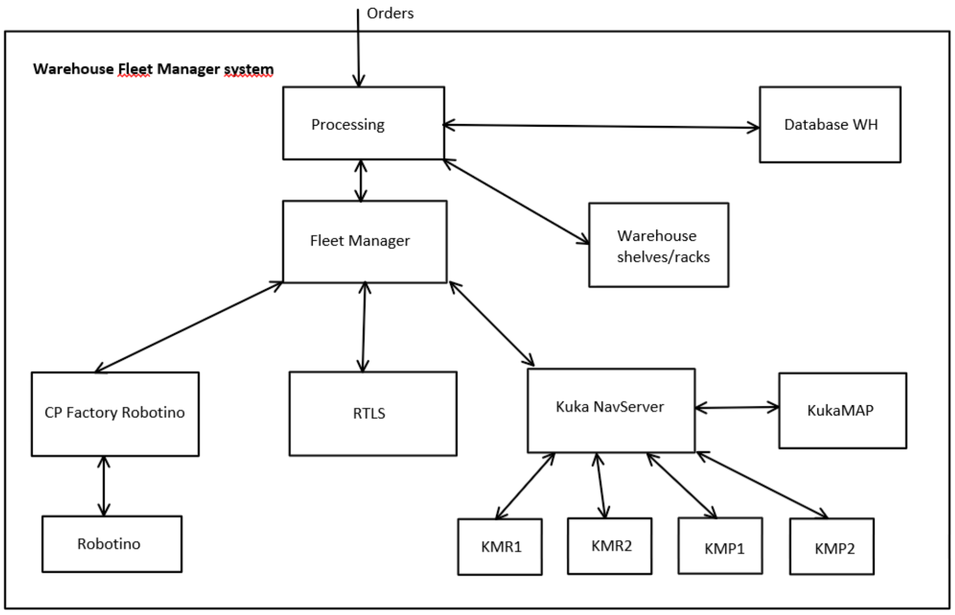


Figure 3.7: Architecture of the Fleet Manager System

System contains OPC UA client and server for communication with MES and also for communication with production lines during docking process. System contains AMQP RabbitMQ message broker for communication with NavServer and also for communication with Multi-agent interface. It contains MQTT client for acquisition of data from RTLS system. Communication architecture is shown in figure 3.8.

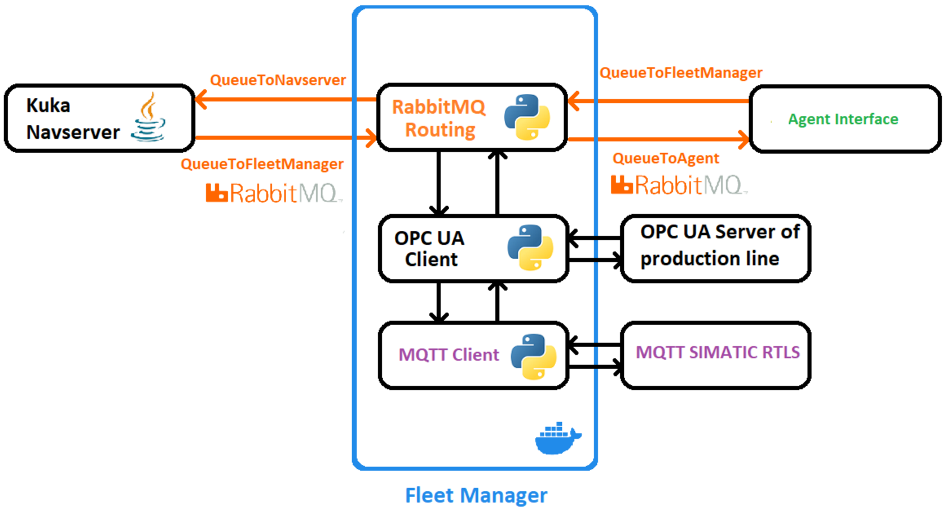


Figure 3.8: Architecture of the communication within Fleet Manager System

The system is designed in Python version 3.7 programming language. For this version of python, the OPC-UA libraries version 0.98.13 are available and primarily used for the initialization of the OPC UA client and server. The pika python amqp version 1.3.2 library is also used. System is dockerized.

3.5.1 Multi-agent interface

There is a instance of agent created for warehouse. It exists within its own docker instance. It communicate with Testbed Multi-agent platform as it was explained in section 2.5. This instance is modified so it translate received data and sends them into the Fleet Manager using AMQP qeueue named QueueToFleetManager. Agent is designed with fixed set of operations which are shown outside to the other agents. For experiments, these capabilities where exposed:

- Pick
- Place
- Transport

Based on this we can specify pick location for the KMP, or the location from which the KMP will carry the shelf. Place location is also specified. Transport has a boolean character, meaning that those operations could be used to command KMP just to move from one position to another. Values inside pick and place are allowed as:

- Warehouse

- Montrac
- Delta
- ABB

We can explore nodes in ZooKeeper manager. We can see there a list of capabilities of all agents connected to the AMQP server paired with a specific agent, who provides this capability. Also there is a registry of all connected agents. Agent ids are unique. List of capabilities is defined based on specific abilities of agents which are currently connected to the MAS. It may looks like this:

- / (ZooKeeper root)
  - Capabilities
    - Pick
      - Agent_Id1
      - Agent_Id2
    - Place
      - Agent_Id1
    - Transport
      - Agent_Id1
      - Agent_Id2
  - Registry
    - Agent_Id1
    - Agent_Id2

That means that Agent_Id1 have pick, place and transport capabilities and Agent_Id2 has only pick and transport capability. Further design and implementation is needed to extend this functionality.

3.6 RTLS interface

We studied Siemens Intelligence system. Interface of this system is shown in image 3.9.

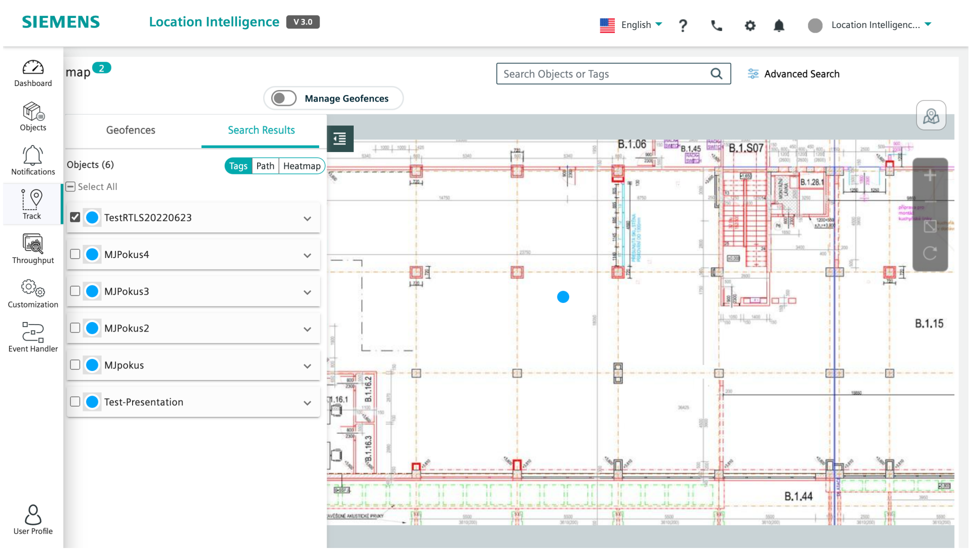


Figure 3.9: RTLS Simatic intelligence manager

Interface is mainly for application settings which we are not using. There is a lot of options to create forbidden zones, check batteries of tags but mainly check the position of the tags. We were observing how the position of one tag was changing in period of one minute. Result is shown in figure 3.10.

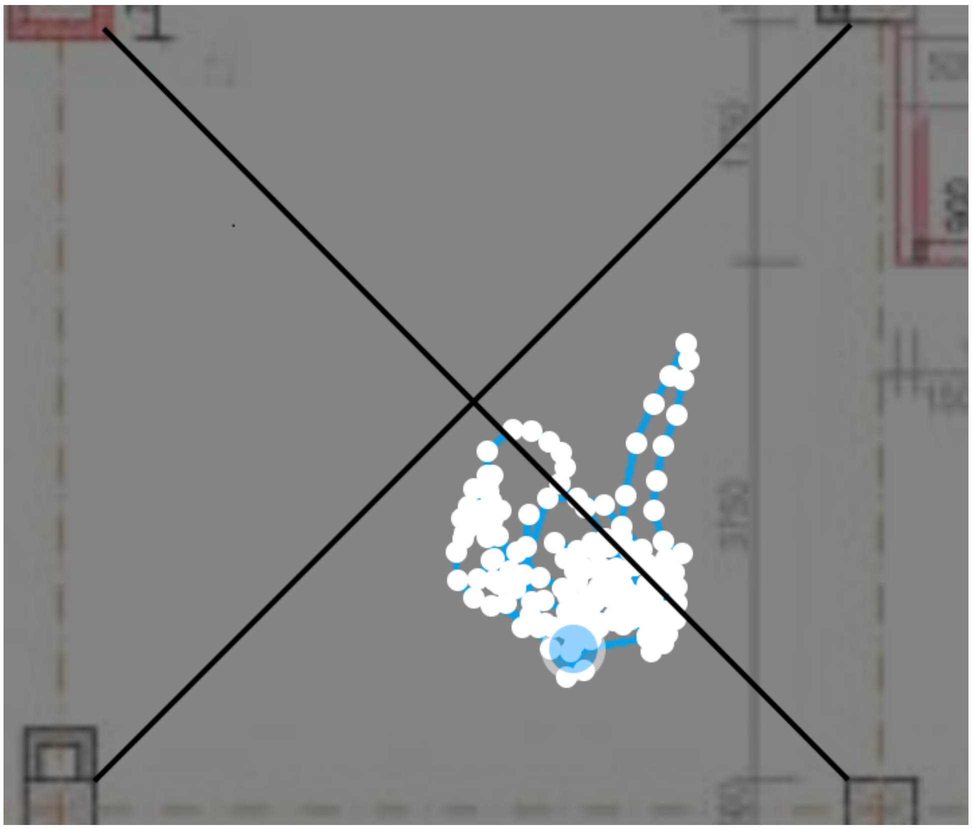


Figure 3.10: RTLS tag position

Based on this in Fleet Manager we collects 5 data sequences within 100 ms and average the position of the tag.

3.7 Trajectory and time optimisation

Since we have a simple graph where we are testing the motions within Testbed, we can optimise trajectory based on data from RTLS in one segment of the graph, near the ABB line.

We use RTLS tag for marking the artifically placed obstacle on the graph path in the Navserver map. Robot cannot move outside of this graph, so if it would hit the obstacle robot would stop and also the program execution would be timed-out. But based on approximated data from RTLS system which can provide data about obstacle position, we can propagate this information into the graph. As soon as we find the position of tag which mark obstacle, we set the node parameter to forbidden. If there is some plan execution on Navserver, we replan it. Any other new plan will plan path without considering this node whish was marked forbidden. This part we test in next chapter.

Chapter 4

Test of the system

We test the communication and data flow from agent to navserver and robot. Then we test RTLS system.

4.0.1 Communication test for task assignment

The warehouse receives a request for the delivery of specific product for the Montrac production line. This product is located on a specific shelf in the warehouse. We can transfer the shelf to the Montrac station using KUKA KMP mobile robots. These are directly controlled by the KUKA Navserver which is controlled by Fleet Manager system. Fleet Manager system waits for data on AMQP queue from agent. Queue is called QueueToFleetManager. As soon as there are some data, Fleet Manager process them and sends instructions for target and source destination through AMQP queue QueueToNavserver. In this example, shelf has pick location Warehouse and place location Montrac.

1. NavServer (Java) sends message addressed to Montrac OPC UA Server via RabbitMQ. The message contains the name of the method we want to call, the parameters for the method (stationName, robotId, dockingType) and the name of the queue, specifically QueueToMontrac.
2. RabbitMQ receiving and routing script (Python) reacts to this message and, considering its parameters and addressing for Montrac OPC UA, calls a method on the Server using the OPC UA Client (Python).
3. Montrac OPC UA Server runs the programmed method and provides output result.
4. OPC UA Client gets the result of the called method and through the RabbitMQ receiving and routing script sends this result to the Navserver (using queue QueueToNavserver)
5. NavServer evaluates the response of the called method and continues with further execution of the code (such as robot movement, or further method calls)

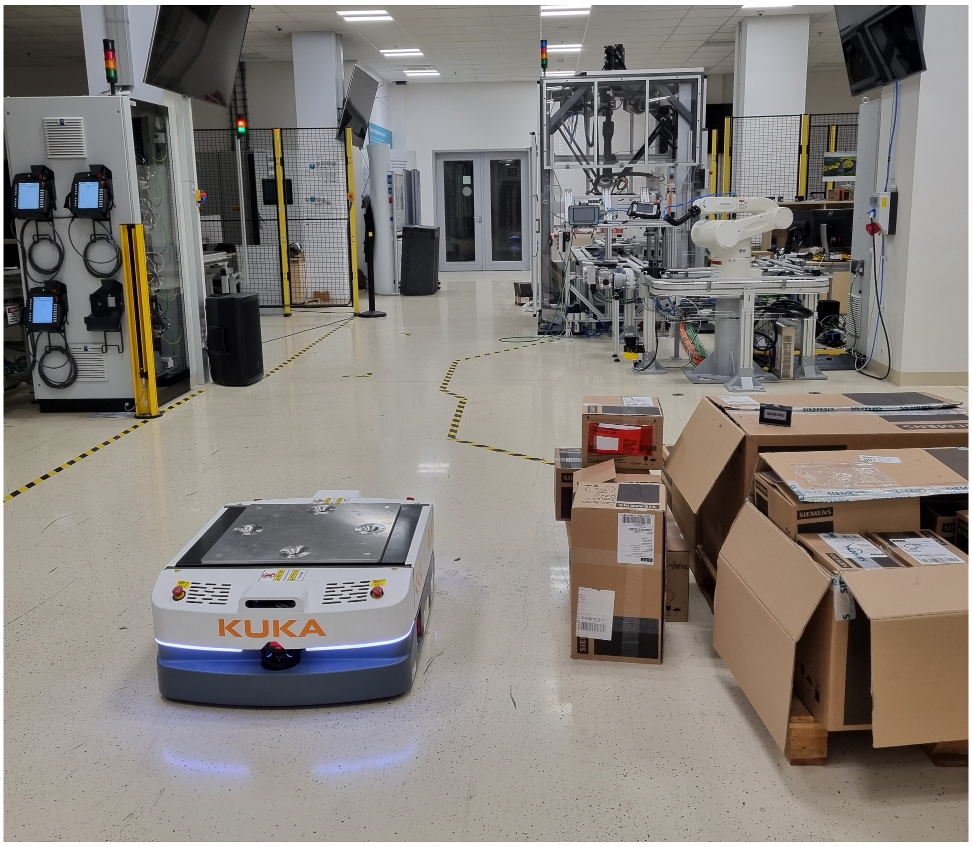


Figure 4.1: RTLS test setup

4.0.2 Test of RTLS

We place the boxes on the path in graph which is normally used on a way to ABB line, because it is shorter 3.4. We command the robot to go to ABB line. Wihtout the data from the RTLS system, the robot will stop in front of the boxes with red sensor indicators. Robot would not move until program reset and manual guidance of the robot to the distance from the boxes.

In next run, we put the SIMATIC RTLS4083T tracker on the pile of boxes. Within few seconds, fleet manager receives information about position if this tag which he consider internally as an obstacle. Fleet manager within 5 seconds of detection changes the parameter of the closest node to the forbidden. Then KMP motion is executed and path is planned. Final path leads through node 18 on the graph 3.4, which is also visible on the picture 4.1. On this picture, the whole setup of the obstacle is shown.

Chapter 5
Conclusion

In this work we successfully installed Kuka Navigation Solution from the local KMR Navbox installation to the Testbed WLAN network. Next we got acquainted with the Kuka Navserver and KUKA mobile robots. Based on this Kuka solution we proposed and implemented Fleet Manager System. This system creates interface between superior control systems as is for example multi-agent system. This MAS system is also slowly developed in Testbed, so its capabilities are not wide. Thus we implemented simple agent interface for this FMS system. Then we put our hands on RTLS Simatic system which can provide location data of the tags placed in Testbed. System is unfortunately not that precise as it was promised, but we were able to filter data and if the nodes were distant enough from each other and if the obstacle was clearly put on one node, then we could quite precisely change parameters of correct node. That led to real time optimisation in trajectory when delivering shelves around Testbed workplace. During implementation of Kuka system, testing the positioning of robot and increasing the precision, we were able to solve a lot of other practical problems which were outside of the scope of this work. Thus working on this thesis was much more beneficial for me than I was expecting in the beginning.

Bibliography

[1] KUKA Aktiengesellschaft Germany. *Matrix production: an example for Industry 4.0* [online] Available from: `https://www.KUKA.com/en-de/industries/solutions-database/2016/10/matrix-production`

[2] Saurabh Vaidya, Prashant Ambad, Santosh Bhosle. *Industry 4.0 – A Glimpse* [online] Procedia Manufacturing Volume 20, 2018, Pages 233-238. Available from: `https://www.sciencedirect.com/science/article/pii/S2351978918300672`

[3] MANU. G, VIJAY KUMAR. M, NAGESH. H, JAGADEESH. D, GOWTHAM. M. B. *FLEXIBLE MANUFACTURING SYSTEMS (FMS): A REVIEW* [online] International Journal of Mechanical and Production Engineering Research and Development 8(2):323-336. Available from: `https://www.researchgate.net/publication/324841870_Flexible_Manufacturing_Systems_FMS_A_Review`

[4] German Association of the Automotive Industry. *VDA 5050 AGV Communication Interface* [online] Tech. Rep. Version 2.0, Jan. 2022. Available from: `https://github.com/vda5050/vda5050`

[5] Czech Institute of Informatics, Robotics, and Cybernetics at CTU. *Groundfloor Testbed* [online] Available from: `https://testbed.ciirc.cvut.cz/labs/testbed/`

[6] KUKA Aktiengesellschaft Germany. *KMR IIWA* [online] Available from: `https://www.kuka.com/en-gb/products/mobility/mobile-robots/kmr-iiwa`

[7] KUKA Aktiengesellschaft Germany. *KMP 600-S diffDrive* [online] Available from: `https://www.kuka.com/en-us/products/mobility/mobile-platforms/kmp-600-s-diffdrive`

[8] KUKA Aktiengesellschaft Germany. *KUKA omniMove AGVs* [online] Available from: `https://www.kuka.com/en-de/products/mobility/mobile-platforms/kuka-omnimove`

[9] SCHUNK SE & Co. KG. *EGI 080-EC* [online] Available from: `https://schunk.com/us/en/gripping-systems/parallel-gripper/egi/egi-080-ec/p/000000000001474387`

[10] KUKA Aktiengesellschaft Germany. *KUKA Navigation Solution* [online] Available from: `https://www.KUKA.com/-/media/KUKA-downloads/imported/9cb8e311bfd744b4b0eab25ca883f6d3/KUKA_navigation_solution_en.pdf`

[11] KUKA Aktiengesellschaft Germany. *KUKA Sunrise.OS* [online] Available from: `https://www.kuka.com/en-de/products/robot-systems/software/system-software/sunriseos`

[12] KUKA Product portfolio 01/2022 *KUKA Navigation Solution* [online] Available from: `https://www.KUKA.com/-/media/KUKA-downloads/imported/9cb8e311bfd744b4b0eab25ca883f6d3/KUKA_rob_product-portfolio_en_screen.pdf`

[13] Malik, Ajay. *RTLS For Dummies.* Wiley. p. 336. ISBN 978-0-470-39868-5. 2009

[14] Siemens *Real-time locating with RTLS.* [online] Available from: `https://www.siemens.com/global/en/products/automation/identification-and-locating/simatic-rtls.html`

[15] Yusnita Rahayu, Tharek Abd. Rahman, Razali Ngah, P.S. Hall. *Ultra Wideband Technology and Its Applications.* [online] Wireless and Optical Communications Networks, 2008. 5th IFIP International Conference. Available from: `https://www.researchgate.net/publication/4340245_Ultra_wideband_technology_and_its_applications`

[16] Siemens *Location Intelligence: Optimizing production and logistic workflows.* [online] Available from: `https://assets.new.siemens.com/siemens/assets/api/uuid:f712ed9f-55ae-45ab-a74f-d63708b3dd04/dics-b10059-01-7600locationintelligence-144_original.pdf`

[17] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algoritmic, Game-Theoretic and Logical Foundations.* [online] Cambridge University Press. June 2012. ISBN 0521899435.

[18] Trygve Reenskaug. *The Model-View-Controller (MVC) Its Past and Present.* [online] University of Oslo 2003. Available from: `https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=4ef90a7b9c1b1cd02acf273694e4059a70c7d198`

[19] RabbitMQ Tm. *AMQP 0-9-1 Model Explained.* [online] Available from: `https://www.rabbitmq.com/tutorials/amqp-concepts.html`

[20] The APACHE software foundation. *ZooKeeper 3.5 Documentation* [online] Available from: `https://zookeeper.apache.org/doc/r3.5.1-alpha/zookeeperOver.html`

[21] OPC FOUNDATION - The Industrial Interoperability Standard. *OPC Technologies - Unified Architecture* [online] Available from: `https://opcfoundation.org/about/opc-technologies/opc-ua/`

[22] Graylog Inc. *What is Graylog?* [online] Available from: `https://go2docs.graylog.org/5-1/what_is_graylog/what_is_graylog.htm`