UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Technology

Paola Avalos Conchas

Payload transportation system of a Learning Factory

Bachelor's Thesis (12 ECTS)

Curriculum Science and Technology

Supervisor(s):
Associate professor of robotics engineering Karl Kruusamäe
Lecturer of robotics technology Veiko Vunder

Tartu 2023

Payload transportation system of a Learning Factory

Abstract:

The transition to Industry 4.0 has highlighted the urgent need for a skilled workforce proficient in operating and maintaining advanced robotic and automation systems. To address this critical issue and prepare the next generation of industry professionals, this thesis focuses on the design and implementation of a payload transportation system for a Learning Factory.

The proposed system combines SLAM, AR Tracking, and ROS Navigation technologies to enable efficient navigation and obstacle avoidance within the Learning Factory environment. Additionally, the system facilitates seamless cooperation with a manipulator robot, enabling collaborative tasks and enhancing the overall efficiency of the system.

The outcome of this work is demonstrated in a book transportation scenario incorporating a multi-robot system that consists of two manipulator robots, and a mobile base. This work contributes to bridging the skills gap and equipping the next generation of industry professionals with practical robotics knowledge.

Keywords: Autonomous Navigation, Learning Factory, ROS, Mobile Robot, Multi-robot system, SLAM

CERCS: T125 Automation, robotics, control engineering , T120 Systems engineering, computer technology

Õppetehase kasuliku koorma transpordisüsteem

Lühikokkuvõte:

Tööstus 4.0-le üleminek on kaasa toonud üha suureneva vajaduse kvalifitseeritud tööjõu järele, kes oskaks kasutada ja hooldada kaasaegseid robot- ja automatiseerimissüsteeme. Järgmise põlvkonna spetsialistide ettevalmistamiseks on vaja kavandada ja rakendada transpordiülesandeid käsitlevaid õppetehaseid. Seda olulist probleemi antud lõputöö lahendabki.

Kavandatav süsteem ühendab SLAM-i, AR-jälgimise ja ROS-navigatsioonitehnoloogiad, et juhtida efektiivselt robotit õppetehase keskkonnas. Lisaks hõlbustab süsteem sujuvat koostööd manipulaatorrobotiga, võimaldades robotitevahelist koostööd ja tõsta süsteemi üldist efektiivsust.

Töö tulemust demonstreeritakse raamatute transportimise stsenaariumiga, milles on integreeritud robotsüsteem kahest manipulaatorrobotist ja ühest mobiilsest ratasplatvormist. See töö aitab ületada oskuste puudujääki ja pakkuda järgmise põlvkonna professionaalsetele robotiinseneridele praktilisi robootikaalaseid teadmisi.

Võtmesõnad:

Autonoomne navigeerimine, õppetehas, ROS, mobiilne robot, mitme roboti süsteem, SLAM

CERCS: T125 Automatiseerimine, robootika, control engineering , T120 Süsteemitehnoloogia, arvutitehnoloogia

ABBREVIATIONS

AMCL - Adaptive Monte Carlo Localization

AR TAG - Augmented Reality Tag

DHCP - Dynamic Host Configuration Protocol

IP Address - Internet Protocol address

LiDAR - Light Detection and Ranging

LF - Learning Factory

MRS - Multi-Robot System

ROS - Robot Operating System

SLAM - Simultaneous localization and mapping

URI - Uniform Resource Identifier

TABLE OF CONTENTS

1 . INTRODUCTION

In today's rapidly evolving world, robotics and automation have become integral to various aspects of our lives. From manufacturing industries to healthcare, transportation to agriculture, robots have proven to be valuable assets, enhancing efficiency, productivity, and safety. Their ability to perform repetitive tasks with precision and accuracy, coupled with advancements in artificial intelligence and machine learning, has unlocked new possibilities for innovation and progress.

However, despite the potential that robots hold, we are still far from fully using their capabilities on a global scale. There are several challenges and barriers that impede widespread adoption and integration of robotics and automation technologies, especially in the context of Industry 4.0. Often referred to as the fourth industrial revolution, it represents the convergence of automation, data exchange, and advanced manufacturing technologies to create smart factories.

The transition to Industry 4.0 necessitates a skilled workforce capable of operating and maintaining the advanced robotic and automation systems that drive this transformation. However, there is a significant skills gap that needs to be addressed. Many industries struggle to find qualified professionals with the necessary knowledge and expertise in robotics and automation. This gap is particularly evident among students and young professionals, as traditional educational approaches often fail to provide sufficient practical experience and hands-on training in these areas.

Recognizing the pressing need to bridge this gap and prepare the next generation of industry professionals, this thesis aims to address a crucial aspect of robotics education within the context of Industry 4.0. Specifically, it focuses on the design and implementation of a Learning Factory (LF)—a dynamic and interactive educational environment that enables students to acquire hands-on experience in robotics and automation.

By addressing this crucial aspect of robotics education, the thesis aims to help in preparing students as the next generation of industry professionals.

2. LITERATURE REVIEW

2.1 Learning Factories

With the rise of Industry 4.0, the digitization and automation of manufacturing processes are quickly becoming the norm [1]. The use of smart networked systems and IoT as well as integration of autonomous robots, has created interconnected production systems that require specialized knowledge and skills beyond traditional engineering disciplines [2]. The demand for skilled professionals who can design, operate, and manage these systems has increased rapidly, highlighting the need for innovative educational approaches that prepare students for a career in Industry 4.0 [2], [3] and Learning Factories have emerged as a potential solution to prepare students for a career in Industry 4.0.

Learning Factories are simulated production environments that are designed to provide students with practical experience in an environment designed and used primarily for the purpose of learning [4]. Additionally, they can act as a research factory for demonstrating and testing new technologies, enabling researchers to identify and implement innovative solutions [3].
The facility may be physical or virtual, or a blended combination. It generally involves multiple machines or operations which can be flexibly reconfigured, and can extend to include supply chains and customer services [5]. They have shown to be effective for developing theoretical and practical knowledge in a real production environment [6], which can push the change for the future generation of engineers.

2.1.1 Examples of LF

In this subsection, I will explore examples of LFs to illustrate their practical applications in various domains as well as the diverse focus they can have. One such example is LPS' Research Group "Lern- und Forschungsfabrik" (LFF) [7], which focuses on application-oriented research; they cooperate with industrial partners and conduct research in the fields of production automation, production management, production services and industrial robotics [7].

The LFF is equipped with a wide range of conventional lathes and milling machines, CNC cutting machines, automated robot cells, industrial robot systems and manual assembly lines as well as measuring machines (Figure 1) [7].

Figure 1. *LFF space [8]*

Another notable example is PoliFactory, the makerspace of Politecnico di Milano [9]. New design and digital manufacturing processes are experimented within this space, developing research on technologies and production/distribution models of circular transition [9].Since this LF is focused on digital design, their tools reflect that. Their equipment includes Laser and Vinyl Cutters, CNC Milling Machines, a wide range of 3D Printers, soldering stations, 3D Scanner and a diverse range of hand tools [9]. (Figure 2)

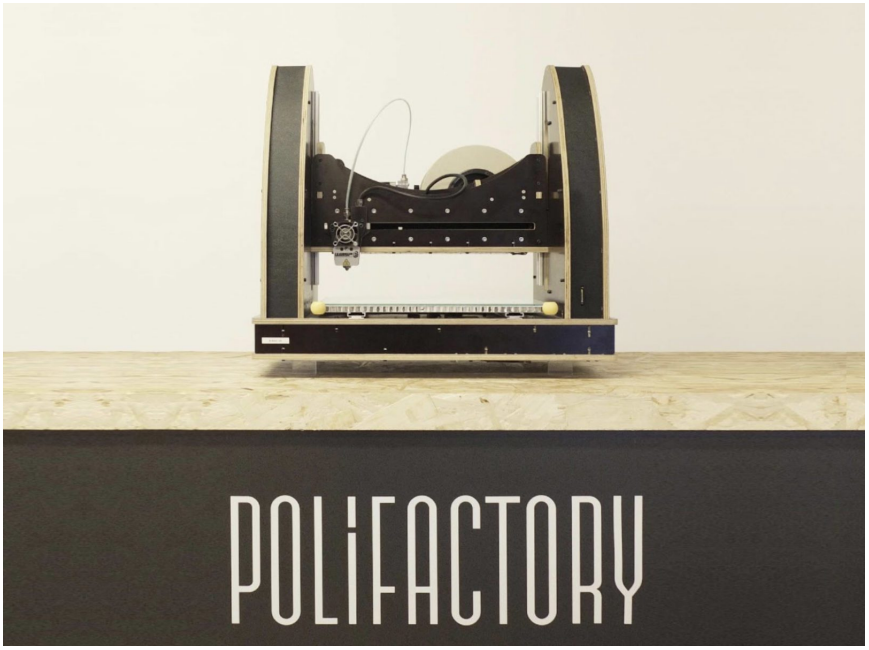

Figure 2. *3D printing equipment at PoliFactory [9].*

Lastly, there is the “mini-factory” laboratory at the Free University of Bolzano (Figure 3) that focuses on simulating a flexible, changeable and reconfigurable assembly line in which a product or multiple products can be produced [10].

Their equipment include: Virtual Reality Headsets, a wide range of robotic arms, a 4-axis Scara robot, an exoskeleton, mobile robots with cobots and other industry-grade equipment.

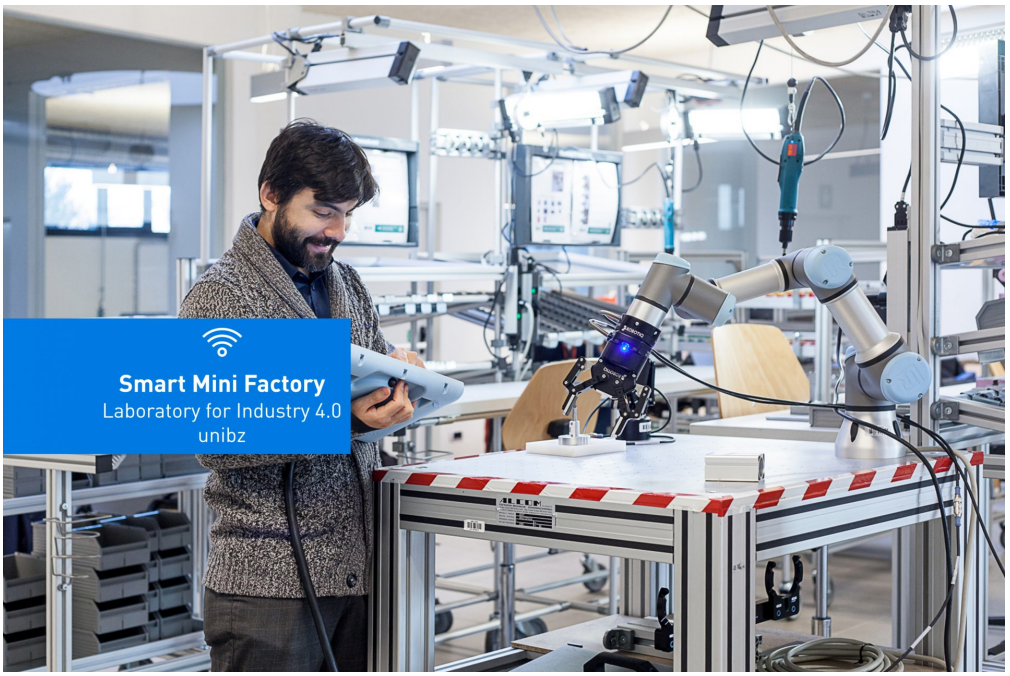

Figure 3. *LF of Free University of Bolzano [11]*

2.2 Multi-Robot Systems (MRS)

Multi-robot systems are commonly seen in the industry, especially in manufacturing and logistics sectors [12 , 13]. It is highly beneficial for a Learning Factory to implement a multi-robot system, as it provides students with hands-on experience and an understanding of the challenges of this approach such as collaborative automation, task allocation, communication constraints, and complex control algorithms.

A Multi-Robot System (MRS) is a group of two or more autonomous robots working together towards a shared goal. MRSs are typically built by combining basic robots in a modular manner to create effective, efficient, and dependable solutions that are cost-effective. While a single robot system may be powerful, certain tasks may be too difficult or even impossible for it to accomplish alone [14]. With the aid of other robots, these systems can efficiently complete tasks that are not distributed and relatively simple [15].

The main challenges or MRS are related to communication constraints, as a generalized mechanism for sharing information is required [16]. The assumption in most autonomous robot systems is the existence of a fixed wireless network over which the robots can communicate, as it is required for task completion [17]. Additionally, the implementation of MRS requires more complex architectures and control algorithms compared to single robot systems, which presents an additional layer of challenges.

There is a wide range of domains in which a multi-robot system can be applied and depending on the application, different tasks represent different requirements [18].

2.3 ROS (Robot Operating System)

When developing a MRS it is essential to combine functionalities of different robots and modularity becomes an important factor to keep in mind. This is where the Robot Operating System (ROS) comes in handy. ROS is an open source software framework for robots. It offers ready-made tools for commonly encountered problems in robotics. ROS provides features such as hardware abstraction, low-level device control and the primary goal of it is to facilitate code reuse in the field of robotics [19].

ROS is hardware agnostic, meaning that it is possible to implement code developed in ROS while maintaining hardware independence and properties of maintainable and flexible software [19].

ROS operates on a node-based architecture, where nodes are the smallest executable units of the system. Nodes exchange data through messages forming a cohesive program [20]. This unique design of ROS enables the integration of programs that were not initially developed to work with ROS, providing developers with greater flexibility in building robotic systems. ROS nodes communicate with each other via named buses called topics, which internally use unidirectional channels. (Figure 4).

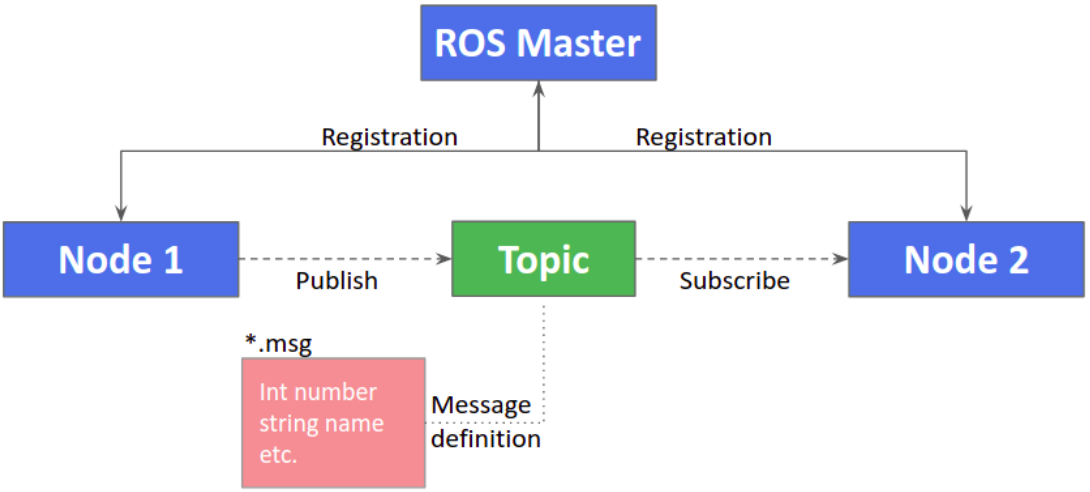

Figure 4. *Structure of communication with nodes in ROS* [21]

A node publishing to a topic means that it is sending information to it using a predefined message format. If a node subscribes to a topic it reads the messages that other nodes have published. Topics can have multiple publishers and subscribers. The overseer that handles all the communication between the nodes in a ROS application is called the ROS master, it enables nodes to locate each other and communicate peer-to-peer [22].

2.3.1 Developing a MRS network in ROS

In the context of MRS, ROS provides a robust platform for coordinating and controlling multiple robots simultaneously due to its modular architecture.

Even though communication in ROS is peer-to-peer, the ROS master acts as the central directory for nodes to look up their communication counterparts, and provides a global registry for configuration values [22, 23].

However, when multiple robots need to exchange information, managing the network becomes challenging. As the number of robots increases, the network traffic and the complexity of coordinating communication between nodes grow significantly. This can lead to bottlenecks, decreased performance, and increased maintenance efforts.

There exist mainly two possible solutions to address this challenge. The first solution is to create a single large ROS network managed by a single roscore node. However, it is important to note that in such a setup, the failure of one master node or ROS network can potentially disrupt the entire system leaving the running nodes headless. Therefore, the second solution is preferred, which involves creating a mechanism to allow information to be exchanged between ROS sub-systems. This approach offers superior scalability and more efficient communication [24].

Each subsystem can operate independently and communicate with others as needed, without relying on a single point of control. This approach offers flexibility, adaptability, and easier expansion as the number of robots or subsystems increases.

Moreover, this decentralized approach allows for the establishment of a single monitoring point for the entire system. With each subsystem having its own roscore node, a central monitoring mechanism can efficiently gather and analyze information from all subsystems, providing a comprehensive view of the entire network. This centralized monitoring simplifies system oversight and troubleshooting.

Additionally, multiple ROS systems can be run on the same network and even on the same machine on different ports. Each ROS system has its own master, but by default, they are unaware of each other and operate separately. This also becomes a problem of how to allow ROS systems to communicate with each other.

The ROS framework already provides a solution for such systems, via the *multimaster_fkie* package, which enables inter-master communication by establishing connections among ROS masters, allowing them to share topics, services, and parameters across different robots or systems [23] (Figure 5). It allows for managing multiple ROS masters and synchronizing their state, ensuring consistent communication and coordination among them [22].

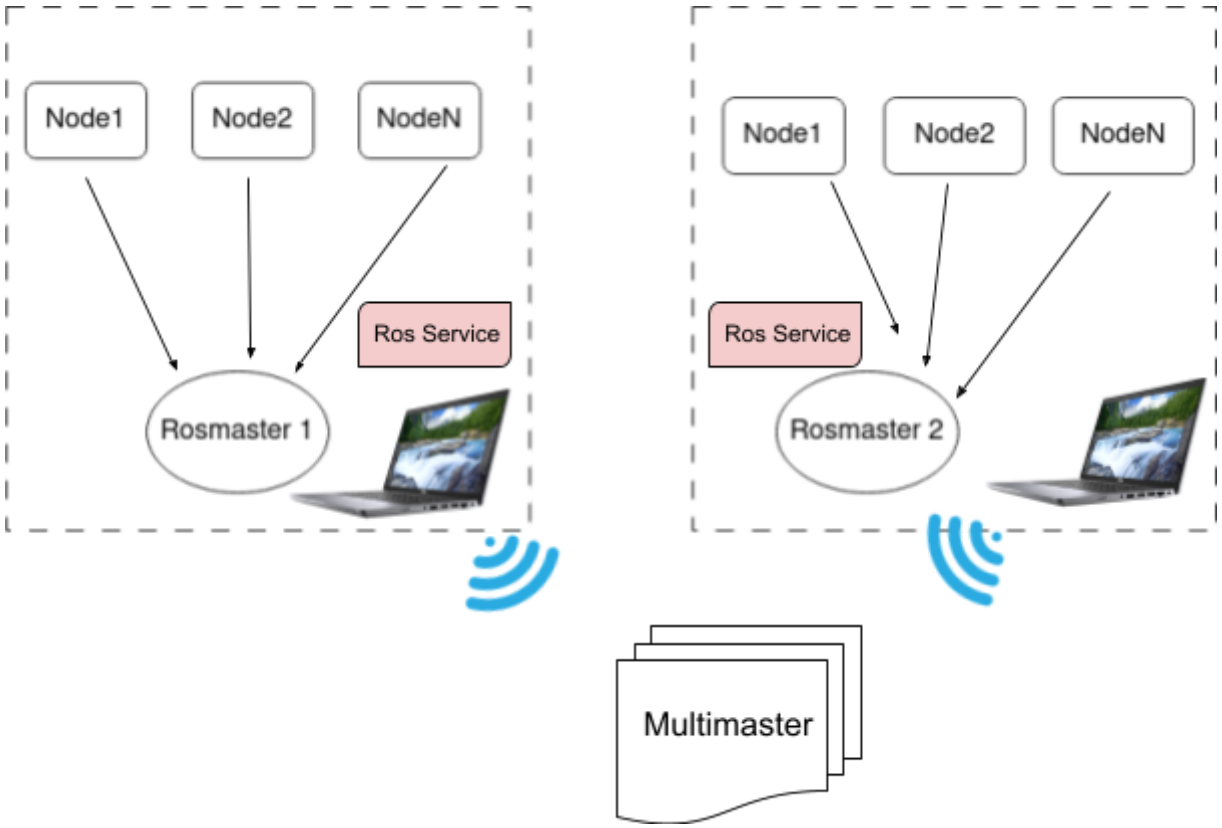

Figure 5. *Interconnection of 2 ROS Masters via Multimaster.*

The roscore nodes can communicate with each other over a common network using the ROS master URI and hostname. The ROS master URI is a unique identifier for each roscore node, and the hostname is the IP address or domain name of the computer running the roscore node [25].

Once the ROS networks are set up, multimaster_fkie can be used to establish connections among the roscore nodes. This is done by adding each master's URI and hostname to the other master's configuration files.

One significant advantage of using multimaster_fkie in a MRS is its ability to provide fault tolerance. This approach allows the system to continue functioning even if one or more masters fail. The remaining masters can continue to communicate and coordinate with each other, ensuring that the robots can continue to operate effectively.

2.4 Autonomous Navigation

In an industry setting, autonomous navigation is widely employed to enable robots to efficiently move around the production floor, navigate through workstations, and perform tasks such as transporting items within warehouses, and even conduct last-mile deliveries [26]. Siegwart *et al* [27] define 4 building blocks for successful navigation:

1. Perception- the robot must interpret its sensors to extract meaningful data
2. Localization- the robot must determine its position in the environment
3. Cognition- the robot must decide how to act to achieve its goals
4. Motion control- the robot must modulate its motor outputs to achieve the desired trajectory.

This section focuses on discussing the methods and tools used in ROS to accomplish these tasks.

2.4.1 Mapping and Localization

To enable effective navigation for a robot, a representation of the environment is essential, which is typically achieved through a map. Mapping is needed before localization, but to build a map, the robot needs to explore and locate itself first. This creates a dependency between the two tasks, which is commonly solved by SLAM [28]. While both can be executed independently, research has shown that performing them simultaneously results in better performance [23 , 24].

There are two primary methods for robot localization: odometry and sensor-based approaches. Odometry uses wheel encoders to estimate the robot's position based on the distance traveled, while sensor-based localization methods use external sensors such as depth cameras or LIDAR to determine the robot's position relative to known features in the environment [30].

Accurate and effective localization is crucial for proper path planning and execution. Without reliable localization, a robot may struggle to navigate complex environments and make errors that could result in collisions or other problems. In ROS, a commonly used tool for localization is AMCL (Adaptive Monte Carlo Localization), which implements a probabilistic localization algorithm known as particle filter-based localization [31 , 32]. AMCL uses sensor data, such as laser scans or depth camera data, to estimate the robot's pose relative to a pre-built map [32]. It detects any drift occurring in the pose estimate based on the odometry and compensates for it (Figure 6)

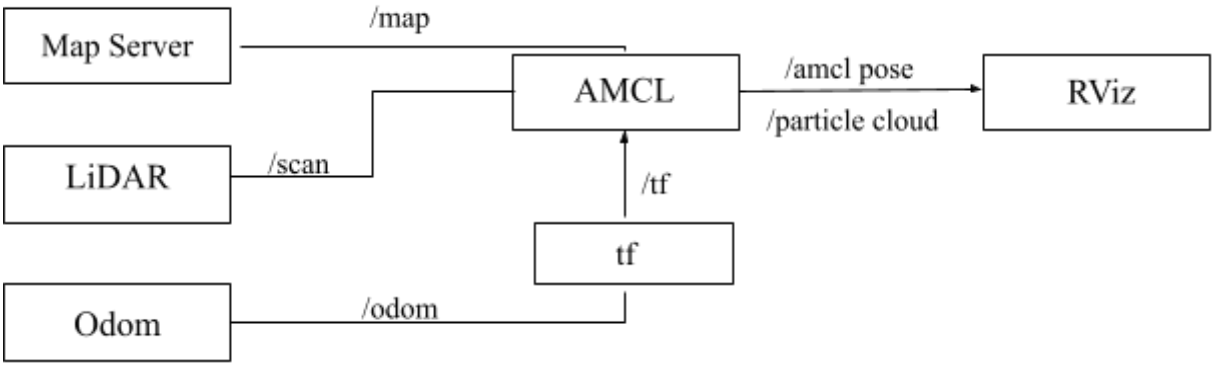

Figure 6. *System overview of the AMCL on ROS framework.*

2.4.2 Path planning and Navigation

Once a robot has localized itself in an environment, it needs to plan a path to reach its goal. This involves generating a feasible path or trajectory from the robot's current position to its desired goal while taking into account environmental constraints and the robot's kinematics. Algorithms like A* (A-star), Dijkstra's, or Dynamic Window Approach (DWA) are commonly used to search for an optimal or near-optimal path [33]. These algorithms can also adapt to changes in the environment and obstacles, enabling the robot to avoid blockages and find alternative paths [28]. Once a path has been planned, motion control is used to execute the path by generating control commands for the robot's actuators. This process considers the robot's dynamics, kinematics, and constraints and may involve feedback control mechanisms to ensure precise and smooth motion.

2.4.3 Autonomous Navigation in ROS

ROS Navigation is a framework that exposes functionalities of trajectory planners through a ROS based interface, using a collection of packages that enable mobile robots to move in the environment avoiding obstacles encountered along the way from its current position to a goal position. It is designed to be as general-purpose as possible primarily meant for differential drive (two separately driven wheels placed on either side of the robot body) and holonomic wheeled robots (robots can move in any direction) [34].

The core of the Navigation framework is the move_base block (Figure 7), which takes a goal pose as input and generates a trajectory for the robot to follow from its current position to the goal position [35]. The move_base block links the global and local planners to adjust the

behavior of the robot during path planning. The robot's position is estimated using odometry, which uses motion sensors like encoders, and the data is published on the /odom topic [35].

The output of the path planning is given in the ROS topic */cmd_vel* in the form of linear and angular velocity messages, the robot base controller then converts those values into the equivalent motor speed to follow the trajectory [36].

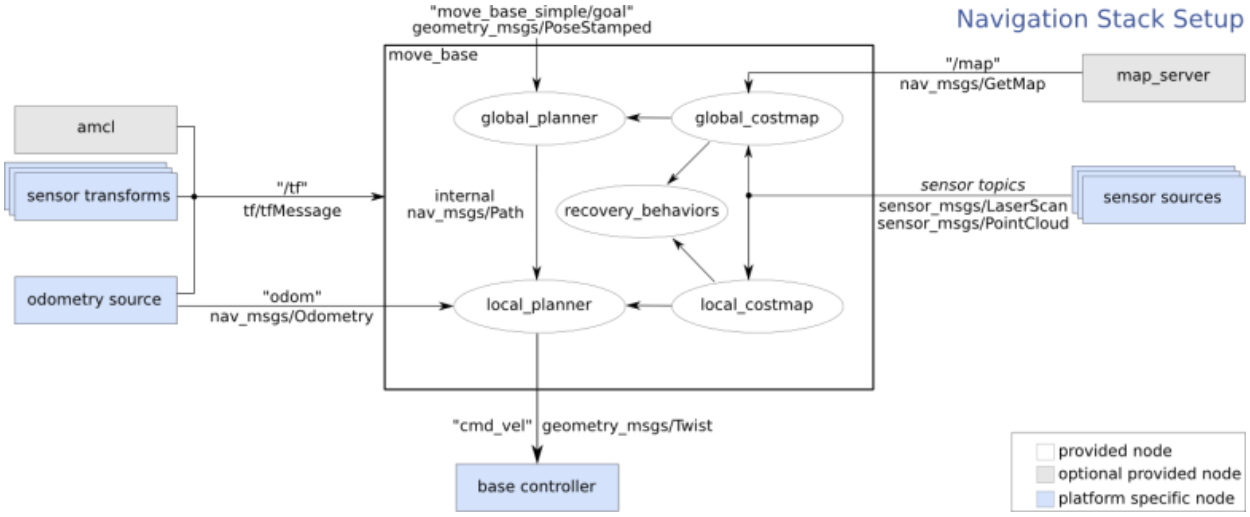

Figure 7. *Standard ROS Navigation Stack* [37]

In order to receive goal requests, ROS Navigation uses ActionServer for communication with an ActionClient. In ROS, services follow a client-server model. A node that wants to utilize a specific service acts as a client and sends a request to the node that provides the service, which acts as the server. The server processes the request and sends back a response to the client [38]. Move_base is an implementation of a SimpleActionServer, which has a single goal policy, subscribed to the topic *move_base_simple/goal*. A *SimpleActionClient* can send goal poses to the server, and move_base generates a trajectory making use of global and local planners [36].

3. REQUIREMENTS

This thesis is a part of a joint project that aims to create a Learning Factory for students. It is beneficial for the students at the University of Tartu to have a learning factory in which they can put their knowledge in practice and acquire skills needed for their future workplace. The resulting Learning Factory should:

- Make use of manipulator and mobile robots
- Allow coordination and communication between the robots
- Be developed in ROS

Given the previously laid out requirements, the first task was to develop a use case that fits the requirements and provides a real-life example of the application. As a result, a smart library system was designed in which a mobile robot navigates autonomously to the goal location to deliver books and communicates with the manipulator to start the shelving process. Once the process is complete, the manipulator will request that the mobile robot returns to its starting position for more books and remains on standby for further goals. See Figure 8 for a representation.

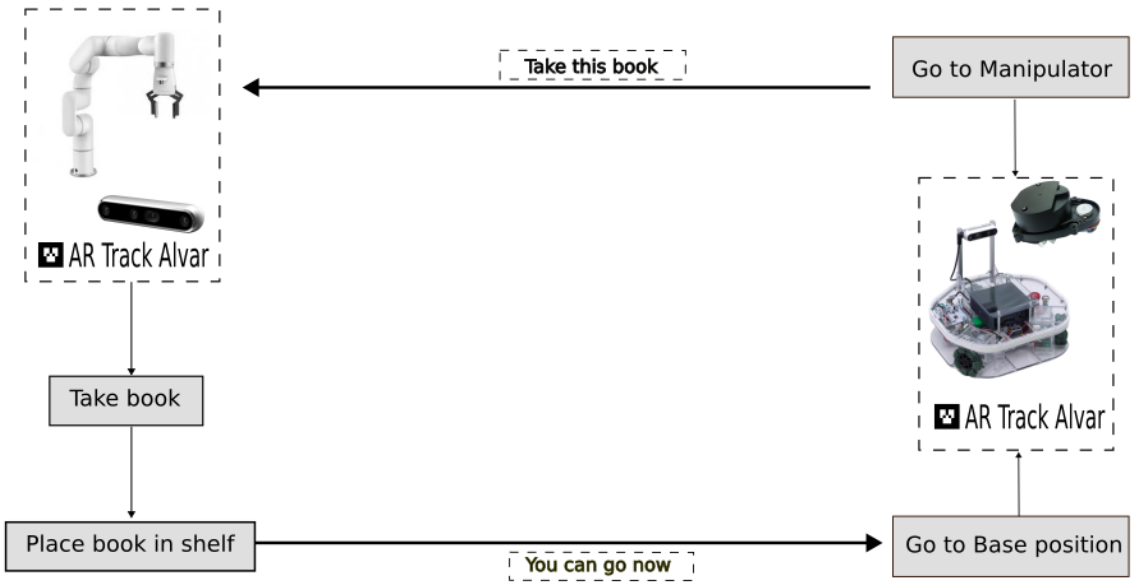

Figure 8. *Simple logic representation of the task designed.*

My thesis deals with the mobile robot component, i.e. the payload transportation task of this system and its implementation.

3.1 Functional Requirements

1) Allow users to modify the environment and goals without changing the source code.
2) Enables mobile robot to communicate with a manipulator robot.
3) Navigates the environment towards a goal while avoiding obstacles.
4) Allows for multiple mobile robots in the same network.

4. DESIGN AND IMPLEMENTATION

Taking into account the previously laid out requirements, the general architecture depicted in Figure 9 was developed in cooperation with a fellow Science and Technology student, Hurova, Iryna.

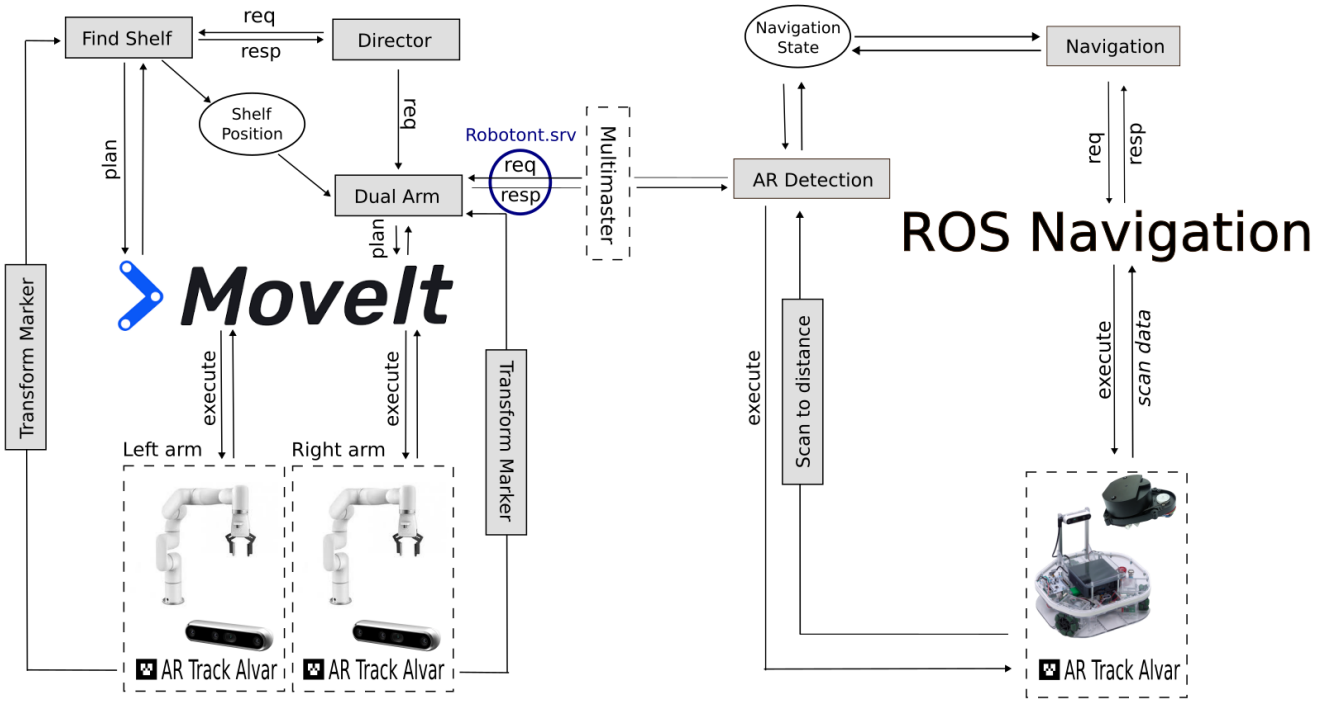

Figure 9. *System overview of implementation of kitting station and payload transportation system.*

For the payload transportation system, the navigation flow was thought out so that the robot receives a goal, transports the books to this goal and requests that the kitting station takes the payload. It then should go back to the starting position for more books to transport and the process is repeated. (Figure 10)

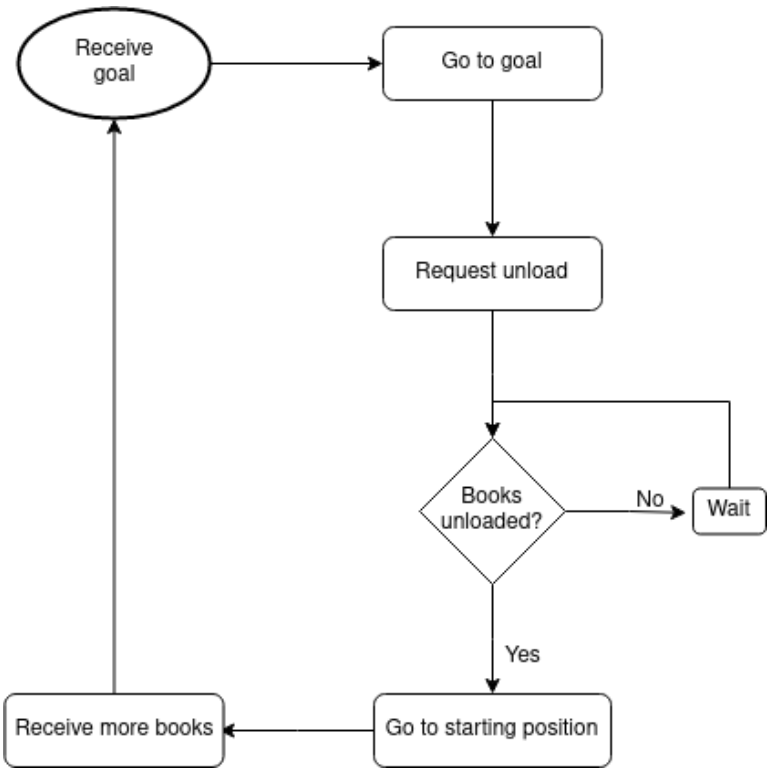

Figure 10. *Overview of the navigation flow of the payload transportation system.*

4.1 Navigating the environment

To tackle this challenge, the ROS Navigation was explored due to its capabilities. As explained in Section 2.5, this tool takes inputs such as a goal pose, odometry information, and data from various sensors to assess the robot's environment. It then generates a path plan and executes the necessary actions to reach the desired goal.

This section delves into the implementation of the Navigation package within the thesis, discussing both its utilization and the challenges encountered during the process.

4.1.1 Mapping the environment

Since the robot has to navigate in the space designated for the LF, a 2D map is provided using the *map_server*. The map can be created using pre-existing mapping packages such as gmapping or hector_slam which are available in ROS. Gmapping was chosen since it takes into account the odometry information to generate and update the map and the robot's pose [35]. Making it an ideal solution for a ground mobile robot.

Having a high-quality map is crucial for navigation. The map's quality varies depending on each case, but an inaccurate or cluttered map can make navigating the environment challenging and this was one of the initial issues. Since the map is a bitmap file, is it possible to modify it using Paint or similar programs [39]. To improve the map's quality, I chose to use Gimp to manually sharpen the corners and clean the map (Figure 11).

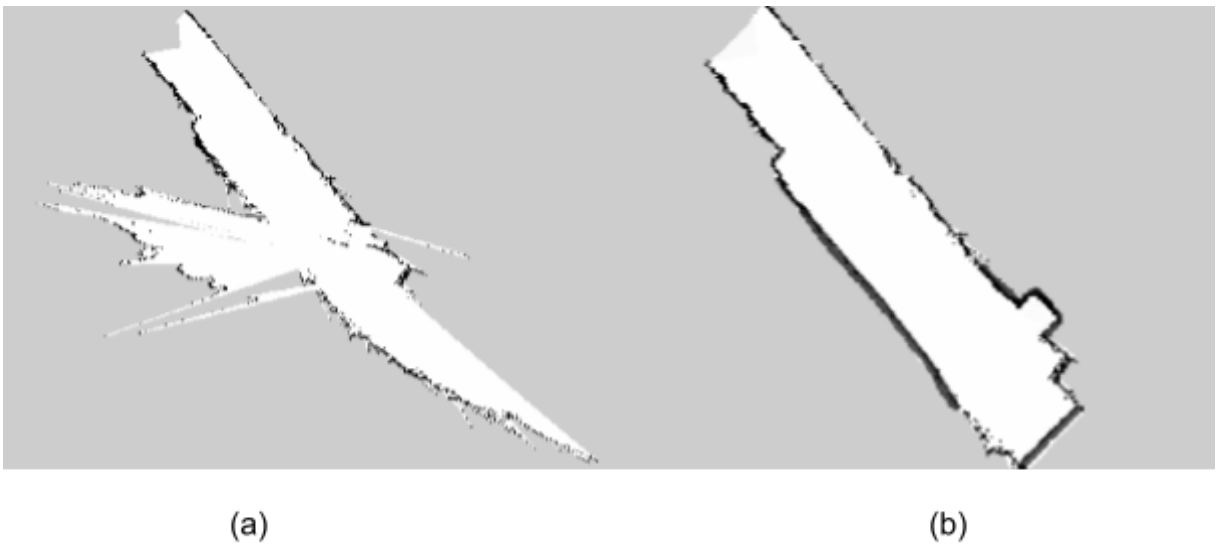

Figure 11. *Before and after cleaning the map using Gimp.*

Visualization of the map can be done using the *Monitoring* node of the thesis package. The node launches RViz, a popular visualization tool which allows the user to view the environment. This visualization node is set up so RViz will display the map, the robots position in the map and the planned paths to the goal (Figure 12).

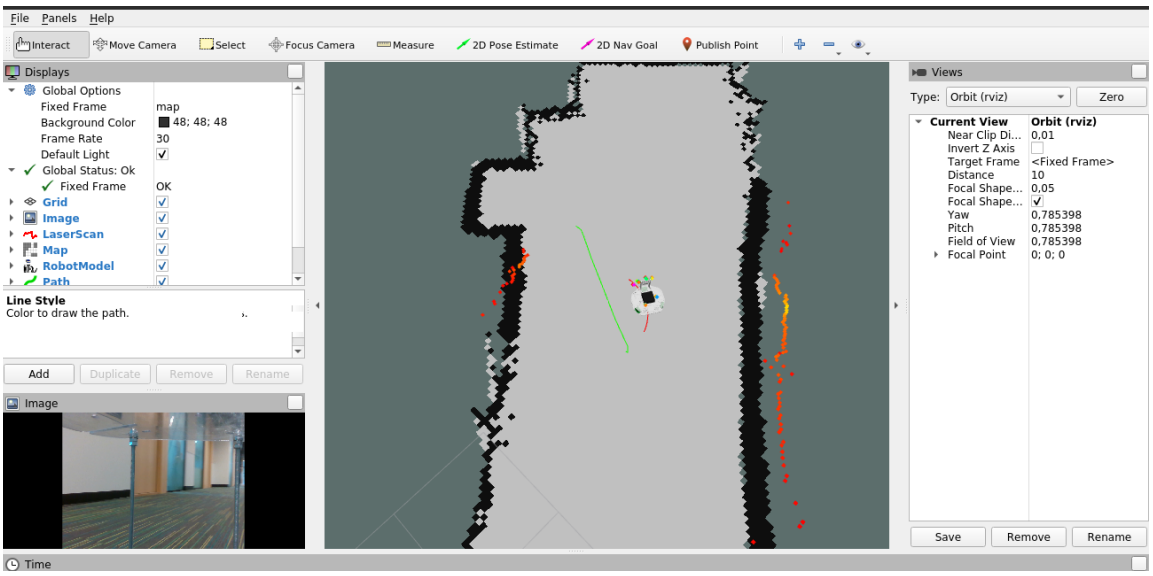

Figure 12. *The RViz window, showing the robot relative to the map, laserscan, camera feed, Global navigation plan (in green), and Local navigation plan (in red).*

4.1.2 AMCL Parameter Tuning

As mentioned in Chapter 2.4.1, Localization is a vital part of Navigation. The AMCL package is part of the Navigation package and provides the transforms needed for the localization and subsequent path planning.

Even though AMCL works out of the box, there were instances where the transforms were not correctly made (Figure 13a). This is a commonly seen issue and there are many review papers dedicated to understanding parameter tuning and its effect on the localization accuracy [32], [40]. This is why it is preferred to tune the parameters to each specific setup.

There are various parameters that were tuned based on the knowledge of the Robotont platform and LIDAR being used. Configuring these parameters increased the performance and accuracy of the AMCL package and decreased the recovery rotations that the robot carries out while carrying out navigation. See Table 1 for the parameters implemented.

Parameter	Default	Tuned
laser_max_beams	30	60
max_particles	5000	2000
resample_interval	2	1
transform _tolerance	0.1	0.5
update_min_a	0.5	0.2

Table 1. *The modified parameters of the AMCL package.*

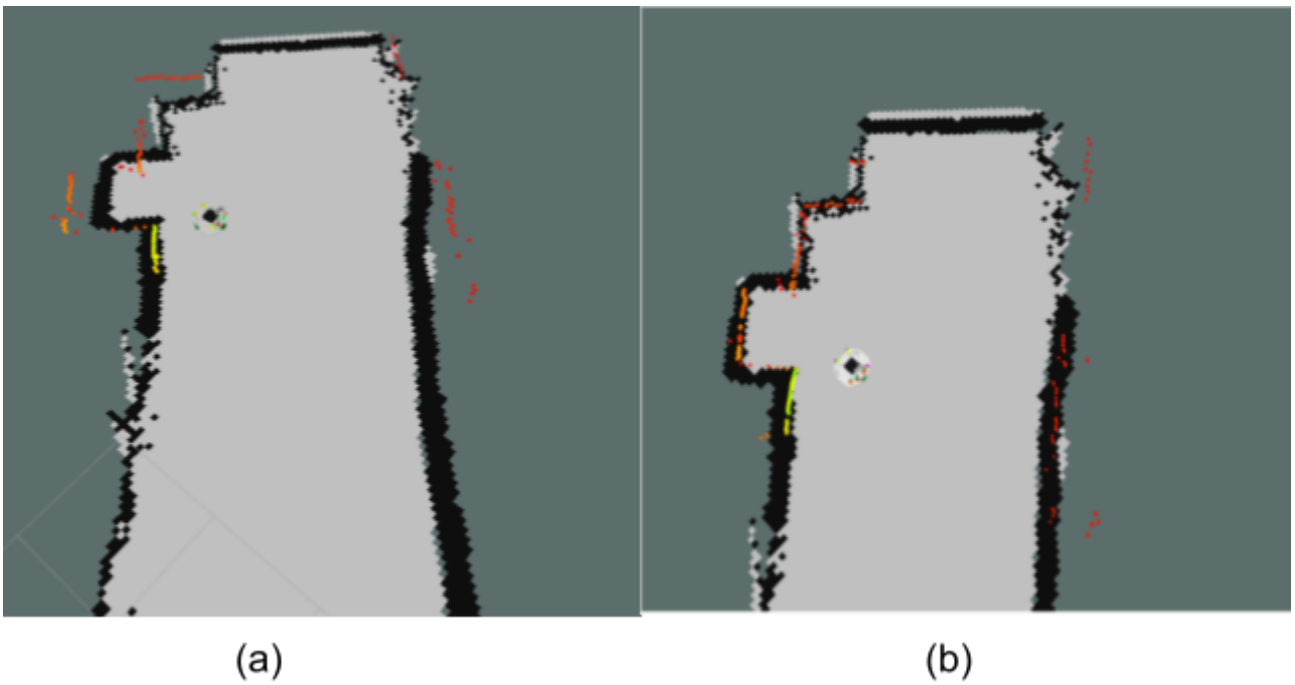

Figure 13. *Comparison of AMCL Omni Localization Results in RViz (a) The robot's estimated pose using the default parameter values, (b) Improved localization performance achieved by tuning the parameters.*

4.2 AR Detection

In order to enhance the accuracy of the final pose and ensure consistent positioning of the mobile robot, AR Tags were implemented as an additional component to navigation. These visual markers possess video tracking capabilities that calculate the camera's real-time position and orientation relative to physical markers in real time (Figure 14). The *ar_track_alvar* package is used to detect these markers.

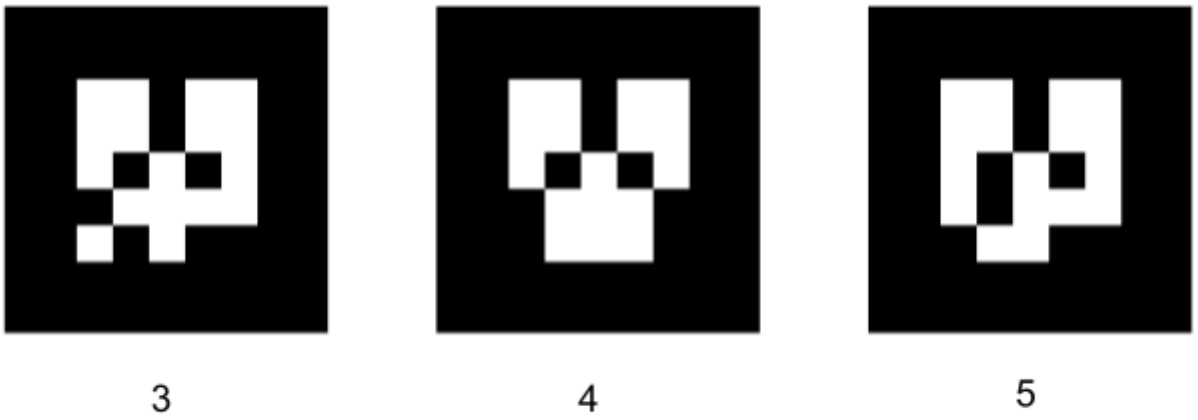

Figure 14: *AR Tag Visual markers with IDs 3,4,5 respectively.*

The implemented solution involves utilizing the pose information provided by the AR tags to establish the robot's position in relation to them. This entails adjusting the robot's position relative to the marker based on its orientation. If the robot deviates beyond the desired distance and angle tolerances, it adjusts its movements to align with the marker and moves closer to the target position using the *Twist* message. This serves as a fallback system in case the navigation alone encounters difficulties, ensuring consistent robot positioning. The system designed can be seen in Figure 15.

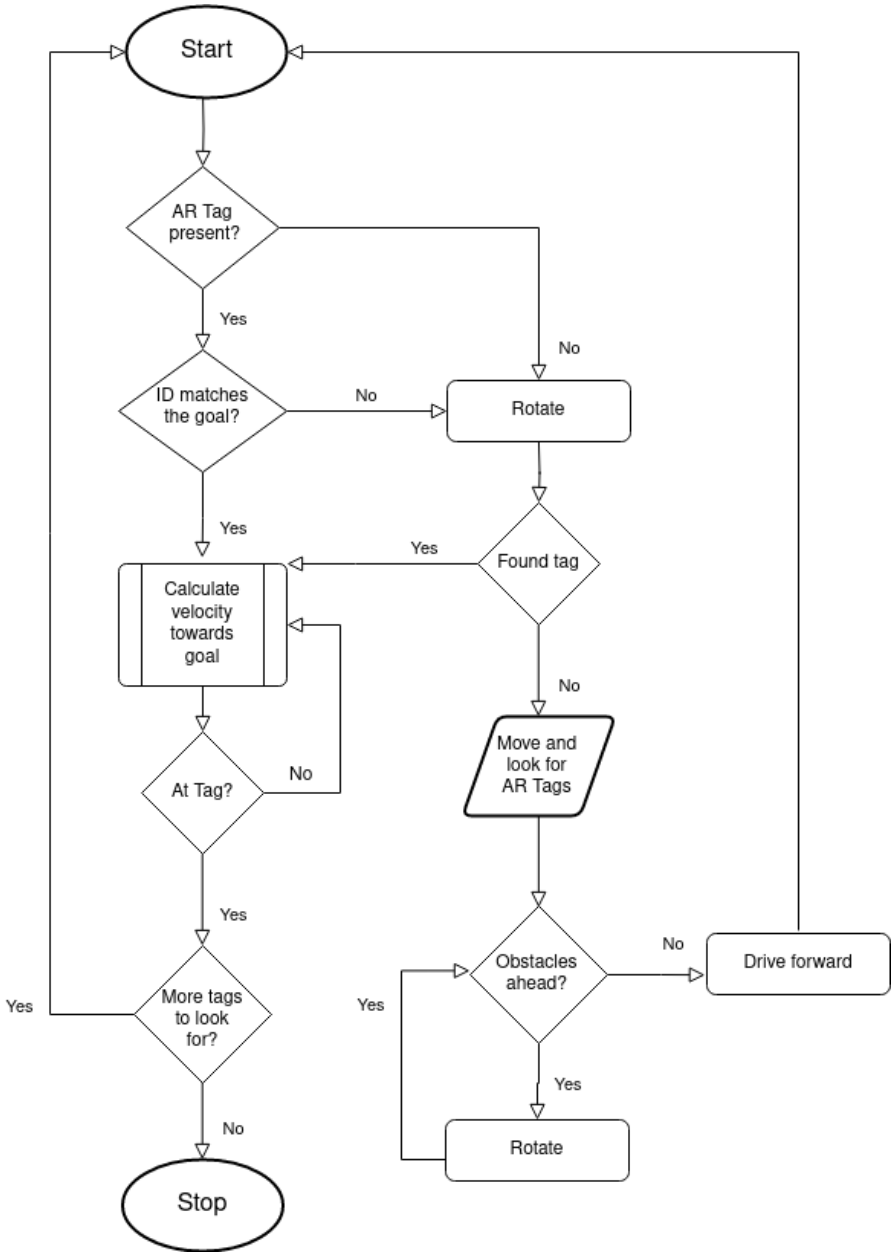

Figure 15. *Flowchart of the AR-based adjustment solution.*

This combination of methods allows the robot to utilize physical markers effectively for accurate positioning, ensuring reliable navigation capabilities in various environments. The integration of AR Tags as part of the system's functionality enhances the robot's ability to navigate with precision and adapt to different scenarios, making it a versatile and robust solution.

4.3 System Implementation

By employing the MoveBase and AR Detection navigation approaches, the robot's successful arrival at its destination is ensured, while also establishing a fallback plan in the event of MoveBase failure. Taking into account these methods, the general architecture depicted in Figure 16 was developed.

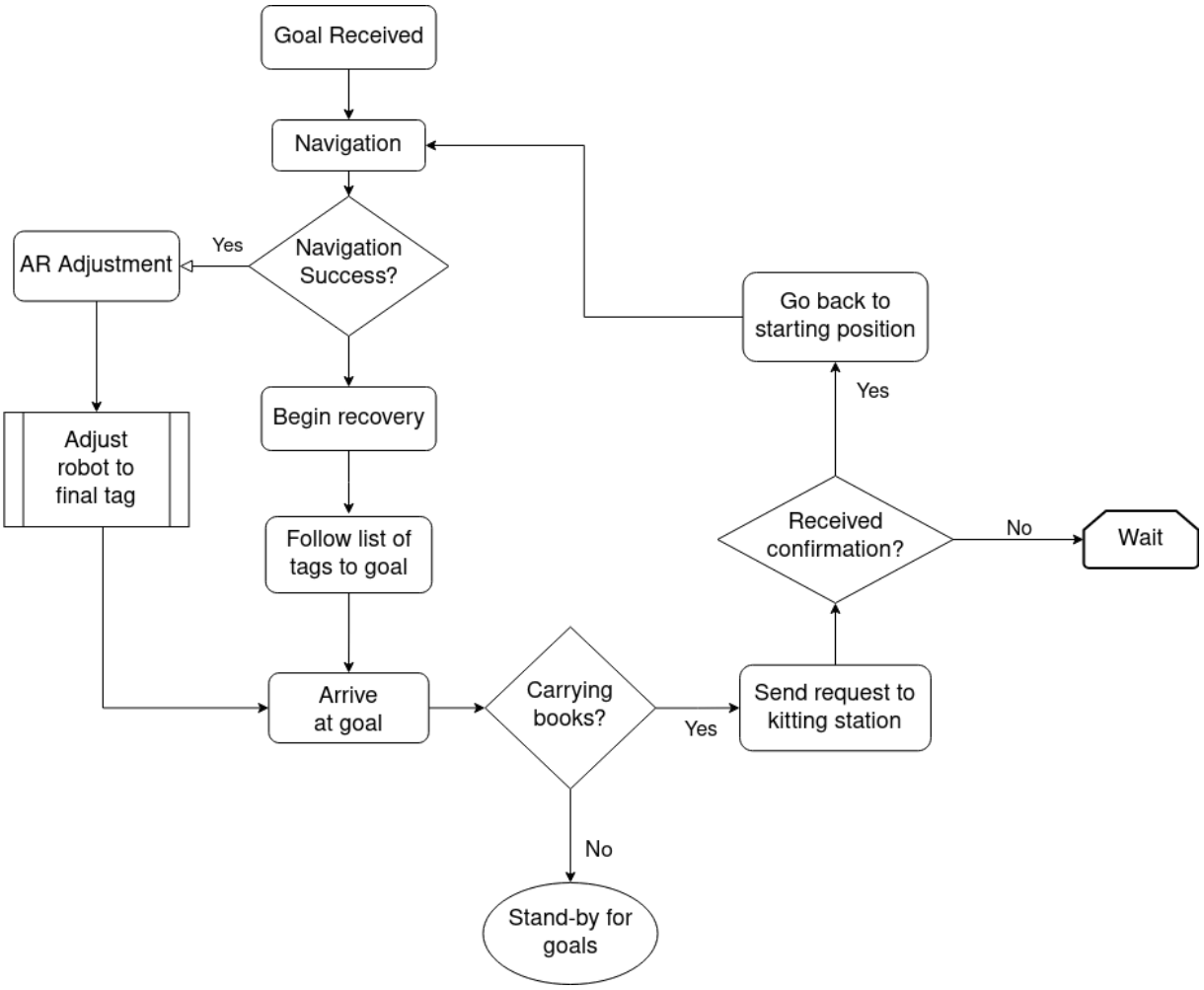

Figure 16. *Flow of payload transportation processes.*

Initially, the robot receives a user-defined goal pose, indicating the location of the manipulator within the map[1]. The navigation process is facilitated by MoveBase, which takes in this goal and creates a path plan. Upon confirmation of success from *MoveBaseServer* that the goal has been reached, the system initiates *AR Adjustment*. This step focuses on employing the previously discussed AR Detection implementation but focusing only on adjusting the robot's pose relative to the last tag on the list, which corresponds to the marker placed at the goal.

[1] When working in a new environment, the goal pose should be given. See Annex for Instructions.

This ensures that any minor orientation offsets or movements are promptly corrected, thereby getting the robot to the correct pose.

In the event of unsuccessful navigation, a recovery plan is implemented. If MoveBaseServer is not successful, *AR Recovery* is initiated and the robot begins searching for AR Tags specified in the list. Upon locating a tag, the robot adjusts its position to face the tag. If the tag is not the final one, the robot continues its search for the subsequent tag until reaching the end of the list. The strategic placement of tags in the environment serves as a guide for the robot, assisting it in navigating towards the manipulator's location.

Regardless of the method employed, reaching the goal location is achieved. Upon successfully arriving at the desired destination, the robot proceeds to send a request to the *Robotont* service hosted by the manipulator. This communication is made possible through the integration of Multimaster, which allows for the sharing of services among masters. The request comprises data related to the AR Tag assigned to a specific cell containing books and is transmitted using the *Robotont.srv* protocol (Figure 17).

Figure 17. *The mobile robot carrying cardboard boxes mimicking books marked by AR Tags.*

Upon receiving a response from the Robotont service that the process has finished, the robot initiates its return journey to the starting position. It employs the same navigation methods, MoveBase and AR Detection, to navigate back to the original location to await for another goal.

4.4 Multi-robot configuration

Expanding upon the established navigation system, the advantages of scaling and implementing multiple robots in the system were considered. By using the same software architecture and nodes across robots, a unified approach can be achieved.

Since each robot uses the same software architecture and has the same nodes, some problems arise in the overall network due to the same node name and conflicting communication. To resolve this issue, different namespaces are set for each robot, so all nodes under a robot will be separate from the ones under another robot. To achieve this, the system was designed such that there is propagation of the namespace specified in the system file "*robotont.service*" to all other launched files and nodes. This ensures that only nodes relevant to the robot are being taken into account.

Building a distributed ROS system across multiple machines is more convenient if the IP addresses of the hosts are fixed [41]. Therefore, a predefined client list was set in the router's DHCP (Dynamic Host Configuration Protocol) server to implement the network topology shown in Figure 18. With this framework, it is possible to control the movement of multiple robots via wireless communication network and using only one monitoring station to run several robots. This allows for greater flexibility of a system in which multiple robots could be in the same space and keep a continuous transportation flow.

Figure 18: *Network topology diagram of the configuration, each robot is automatically assigned a fixed IP address. They are all wirelessly connected to the same network and can be accessed via a single workstation.*

5. Validation

In order to demonstrate the functionalities of the payload transportation system, the following scenario was implemented: The task is to deliver the books placed on top of the robot to the manipulator facing the open area while avoiding obstacles. The mobile robot was able to deliver books from the starting location to the unloading station successfully.

5.1 Hardware Specifications

The omni-directional mobile robotics platform Robotont, shown in Figure 19, is used to develop and validate the thesis

The Robotont's hardware includes:

- Intel Core i5 (7th Gen) 7260U (2 cores, up to 3.4 GHz)

- Lidar: *HLS-LFCD* 360° 2D *LiDAR*

- Camera: Intel *RealSense* depth camera *D435i*

Figure 19.*The omni-directional Robotont, developed by the Intelligent Materials and Systems Lab of the University of Tartu* [42].

5.2 Software

ROS environment:

- Ubuntu 20.04
- ROS Noetic

6. DISCUSSION AND FUTURE WORK

The objectives of this thesis have been fulfilled, resulting in the development of an autonomous navigation system for a mobile robot that is capable of collaborating with a manipulator. However, in the broader scope of the project, there are opportunities for further improvements and the addition of new features.

6.1 Limitations

Mapping large areas will likely result in skewed maps due to the cumulative uncertainty of state estimation [43]. A more accurate map can be obtained by mapping a smaller area with distinct features. It is also worth mentioning that there are environmental constraints, such as having a well-lit room with solid walls (i.e avoiding large windows or glass panels).

The mobile robot used for the development of this thesis, the Robotont, is only able to drive on flat surfaces and is prone to slippage, therefore, environments with elevated surfaces or debris on the floor are not ideal for navigation.

6.2 Future Work

The current system has been developed using ROS Noetic, and it would be beneficial to conduct testing to determine its compatibility with other ROS distributions or the newer ROS2, since it would potentially simplify the network configuration as it is built on a data distribution system (DDS), which is able to connect nodes without the master.

While the system functions adequately, there is room for enhancing the dynamic obstacle avoidance capabilities provided by the Navigation Package. Several potential improvements could be achieved by different methods, such as optimizing trajectory planning algorithms and deep reinforcement learning techniques [44].

Given that ROS is hardware agnostic, the system developed using is not tied to a single robot model. While this thesis was developed using the Robotont, other mobile robots [2] can use the system with minor modifications to the source code. This represents a big advantage in the context of a Learning Factory since it allows for flexibility. This can also solve the environmental constraints and allow for use in a warehouse with different surfaces and potentially larger and heavier loads that the Robotont is not able to carry.

[2] Clearpath Robotics' TurtleBot, Jackal or Husky, to name a few

While multiple mobile robots are able to communicate and keep their autonomy under the same network, a better system architecture is necessary to provide a smooth user experience. At the moment, the foundations have been set to allow future collaboration between robots but it cannot yet act as a global navigation system.

REFERENCES

[1] "How Industry 4.0 technologies are changing manufacturing." https://www.ibm.com/topics/industry-4-0 (accessed Apr. 10, 2023).

[2] B. Motyl, G. Baronio, S. Uberti, D. Speranza, and S. Filippi, "How will Change the Future Engineers' Skills in the Industry 4.0 Framework? A Questionnaire Survey," *Procedia Manuf.*, vol. 11, pp. 1501–1509, 2017, doi: 10.1016/j.promfg.2017.07.282.

[3] Executive Agency for Small and Medium sized Enterprises. and PwC., *Skills for industry: skills for smart industrial specialisation and digital transformation.* LU: Publications Office, 2019. Accessed: May 01, 2023. [Online]. Available: https://data.europa.eu/doi/10.2826/69861

[4] "Learning Factories Networks," *Eit Manufactoring*. https://www.eitmanufacturing.eu/what-we-do/education/resources/learning-factory%E2%80%8B/ (accessed Apr. 20, 2023).

[5] S. Edwards, "'Learning Factories' - What they are and why their time has come," Jul. 31, 2020. https://www.fenews.co.uk/exclusive/learning-factories-what-they-are-and-why-their-time-has-come/ (accessed Apr. 15, 2023).

[6] F. Baena, A. Guarin, J. Mora, J. Sauza, and S. Retat, "Learning Factory: The Path to Industry 4.0," *Procedia Manuf.*, vol. 9, pp. 73–80, 2017, doi: 10.1016/j.promfg.2017.04.022.

[7] "LPS Learning and Research Factory (LFF)," *Advanced Technologies for Industry*. https://ati.ec.europa.eu/technology-centre/lehrstuhl-fur-produktionssysteme (accessed Apr. 20, 2023).

[8] "LPS Learning Factory, Ruhr-Universität Bochum, Germany," *International Association of Learning Factories*. https://ialf-online.net/index.php/component/content/article/17-members/64-los.html?Itemid=101 (accessed Apr. 16, 2023).

[9] "About PoliFactory." https://www.polifactory.polimi.it/en/about/ (accessed Apr. 10, 2023).

[10] D. T. Matt, E. Rauch, and P. Dallasega, "Mini-factory – A Learning Factory Concept for Students and Small and Medium Sized Enterprises," *Procedia CIRP*, vol. 17, pp. 178–183, 2014, doi: 10.1016/j.procir.2014.01.057.

[11] "Smart Mini Factory." https://smartminifactory.it/ (accessed Apr. 19, 2023).

[12] A. Farinelli, N. Boscolo, E. Zanotto, and E. Pagello, "Advanced approaches for multi-robot coordination in logistic scenarios," *Robot. Auton. Syst.*, vol. 90, pp. 34–44, Apr. 2017, doi: 10.1016/j.robot.2016.08.010.

[13] J. A. Marvel, R. Bostelman, and J. Falco, "Multi-Robot Assembly Strategies and Metrics," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 1–32, Jan. 2019, doi: 10.1145/3150225.

[14] Z. Yan, N. Jouandeau, and A. A. Cherif, "A Survey and Analysis of Multi-Robot Coordination," *Int. J. Adv. Robot. Syst.*, vol. 10, no. 12, p. 399, Dec. 2013, doi: 10.5772/57313.

[15] A. Gautam and S. Mohan, "A review of research in multi-robot systems," in *2012 IEEE 7th International Conference on Industrial and Information Systems (ICIIS)*, Chennai, India: IEEE, Aug. 2012, pp. 1–5. doi: 10.1109/ICIInfS.2012.6304778.

[16] M. M. Veloso and D. Nardi, "Special Issue on Multirobot Systems," *Proc. IEEE*, vol. 94, no. 7, pp. 1253–1253, Jul. 2006, doi: 10.1109/JPROC.2006.877080.

[17] J. Stephan, J. Fink, V. Kumar, and A. Ribeiro, "Concurrent Control of Mobility and Communication in Multirobot Systems," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1248–1254, Oct. 2017, doi: 10.1109/TRO.2017.2705119.

[18] R. N. Darmanin and M. K. Bugeja, “A review on multi-robot systems categorised by application domain,” in *2017 25th Mediterranean Conference on Control and Automation (MED)*, Valletta, Malta: IEEE, Jul. 2017, pp. 701–706. doi: 10.1109/MED.2017.7984200.
[19] “ROS - Introduction,” *ROS - Introduction*. https://wiki.ros.org/ROS/Introduction (accessed Apr. 10, 2023).
[20] W. Guan, S. Chen, S. Wen, Z. Tan, H. Song, and W. Hou, “High-Accuracy Robot Indoor Localization Scheme Based on Robot Operating System Using Visible Light Positioning,” *IEEE Photonics J.*, vol. 12, no. 2, pp. 1–16, Apr. 2020, doi: 10.1109/JPHOT.2020.2981485.
[21] “ROS Publisher and Subscriber.” https://robolabor.ee/homelab/en/ros/subscribepublish (accessed Apr. 15, 2023).
[22] “ROS Master,” Apr. 12, 2023. http://wiki.ros.org/Master
[23] A. Tiderko, “Multimaster-fkie.” http://wiki.ros.org/multimaster_fkie (accessed May 03, 2023).
[24] S. Hernandez Juan and F. Herrero Cotarelo, “Multi-master ROS systems,” Universitat Politècnica de Catalunya (UPC), IRI-TR-15-1, Jun. 2015. Accessed: Apr. 14, 2023. [Online]. Available: http://www.iri.upc.edu/files/scidoc/1607-Multi-master-ROS-systems.pdf
[25] “ROS Environment Variables.” http://wiki.ros.org/ROS/EnvironmentVariables (accessed May 16, 2023).
[26] H. Martínez-Barberá and D. Herrero-Pérez, “Autonomous navigation of an automated guided vehicle in industrial environments,” *Robot. Comput.-Integr. Manuf.*, vol. 26, no. 4, pp. 296–311, Aug. 2010, doi: 10.1016/j.rcim.2009.10.003.
[27] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*, 2nd ed. in Intelligent robotics and autonomous agents. Cambridge, Mass: MIT Press, 2011.
[28] Y. D. V. Yasuda, L. E. G. Martins, and F. A. M. Cappabianco, “Autonomous Visual Navigation for Mobile Robots: A Systematic Literature Review,” *ACM Comput. Surv.*, vol. 53, no. 1, pp. 1–34, Jan. 2021, doi: 10.1145/3368961.
[29] M. E. López, L. M. Bergasa, R. Barea, and M. S. Escudero, “A Navigation System for Assistant Robots Using Visually Augmented POMDPs,” *Auton. Robots*, vol. 19, no. 1, pp. 67–87, Jul. 2005, doi: 10.1007/s10514-005-0607-3.
[30] M. O. A. Aqel, M. H. Marhaban, M. I. Saripan, and N. Bt. Ismail, “Review of visual odometry: types, approaches, challenges, and applications,” *SpringerPlus*, vol. 5, no. 1, p. 1897, Dec. 2016, doi: 10.1186/s40064-016-3573-7.
[31] G. Peng *et al.*, “An Improved AMCL Algorithm Based on Laser Scanning Match in a Complex and Unstructured Environment,” *Complexity*, vol. 2018, p. 2327637, Dec. 2018, doi: 10.1155/2018/2327637.
[32] W. P. N. D. Reis, G. J. D. Silva, O. M. Junior, and K. C. T. Vivaldini, “An extended analysis on tuning the parameters of Adaptive Monte Carlo Localization ROS package in an automated guided vehicle,” *Int. J. Adv. Manuf. Technol.*, vol. 117, no. 5–6, pp. 1975–1995, Nov. 2021, doi: 10.1007/s00170-021-07437-0.
[33] S. Abdallaoui, E.-H. Aglzim, A. Chaibet, and A. Kribèche, “Thorough Review Analysis of Safe Control of Autonomous Vehicles: Path Planning and Navigation Techniques,” *Energies*, vol. 15, no. 4, p. 1358, Feb. 2022, doi: 10.3390/en15041358.
[34] “ROS Navigation.” http://wiki.ros.org/navigation (accessed May 16, 2023).
[35] R. L. Guimarães, A. S. De Oliveira, J. A. Fabro, T. Becker, and V. A. Brenner, “ROS Navigation: Concepts and Tutorial,” in *Robot Operating System (ROS)*, A. Koubaa, Ed., in Studies in Computational Intelligence, vol. 625. Cham: Springer International Publishing, 2016, pp. 121–160. doi: 10.1007/978-3-319-26054-9_6.

[36] F. E. Parra Gil, “Implementation of robot manager subsystem for TEMOTO software framework,” University of Tartu, 2020.
[37] “ROS Navigation Setup.” http://wiki.ros.org/navigation/Tutorials/RobotSetup (accessed May 16, 2023).
[38] “ROS Services.” http://wiki.ros.org/Services (accessed May 23, 2023).
[39] “How to edit a map generated with Gmapping,” *The Construct*. https://www.theconstructsim.com/ros-qa-136-how-to-edit-a-map-generated-with-gmapping/ (accessed May 12, 2023).
[40] A. Koubâa, Ed., *Robot Operating System (ROS): the complete reference. Volume 6*. in Studies in computational intelligence, no. 962. Cham: Springer, 2021.
[41] P. Anggraeni, M. Mrabet, M. Defoort, and M. Djemai, “Development of a wireless communication platform for multiple-mobile robots using ROS,” in *2018 6th International Conference on Control Engineering & Information Technology (CEIT)*, Istanbul, Turkey: IEEE, Oct. 2018, pp. 1–6. doi: 10.1109/CEIT.2018.8751845.
[42] “Robotont.” http://robotont.ut.ee/ (accessed May 12, 2023).
[43] R. Valner *et al.*, “Scalable and heterogenous mobile robot fleet-based task automation in crowded hospital environments—a field test,” *Front. Robot. AI*, vol. 9, p. 922835, Aug. 2022, doi: 10.3389/frobt.2022.922835.
[44] M. Everett, Y. F. Chen, and J. P. How, “Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning,” 2018, doi: 10.48550/ARXIV.1805.01956.

APPENDIX

1. Source code and Video demonstration

The source code as well as instructions and video demonstration can be found in the repository for this thesis.

https://github.com/PaoAvalos/Thesis_Bsc_2023_TransportationSystem

NON-EXCLUSIVE LICENCE TO REPRODUCE THESIS AND MAKE THESIS PUBLIC

I, Paola Avalos Conchas

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

Payload transportation system of a Learning Factory,

supervised by Prof. Karl Kruusamäe and PhD Veiko Vunder.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Paola Avalos Conchas

24/05/2023