FLEET MANAGEMENT SYSTEM IMPROVED WITH VDA5050 STANDARD FOR AUTONOMOUS FLEET ROBOTS

BEKİR BOSTANCI

Thesis for the Master’s Program in Electrical and Electronics Engineering

Graduate School
Izmir University of Economics
Izmir
2024

FLEET MANAGEMENT SYSTEM IMPROVED WITH VDA5050 STANDARD FOR AUTONOMOUS FLEET ROBOTS

BEKİR BOSTANCI

THESIS ADVISOR: ASSOC. PROF. DR. PINAR OĞUZ EKİM

A Master's Thesis

Submitted to

The Graduate School of Izmir University of Economics

The Department of Electrical and Electronics Engineering

İzmir

2024

ETHICAL DECLARATION

I hereby declare that I am the sole author of this thesis and that I have conducted my work in accordance with academic rules and ethical behavior at every stage from the planning of the thesis to its defense. I confirm that I have cited all ideas, information and findings that are not specific to my study, as required by the code of ethical behavior, and that all statements not cited are my own.

Name, Surname: Bekir Bostancı

Date: 08.01.2024

ABSTRACT

FLEET MANAGEMENT SYSTEM IMPROVED WITH VDA5050 STANDARD FOR AUTONOMOUS FLEET ROBOTS

Bostancı, Bekir

Master’s Program in Electrical and Electronics Engineering

Advisor: Assoc. Prof. Dr. Pınar OĞUZ EKİM

January, 2024

VDA5050 is a robot fleet management system interface developed for the effective management and interoperability of autonomous robots. This interface is designed to overcome the challenges posed by the increasing use of autonomous mobile robots. It provides solutions for coordinated operation of multiple robots, efficient distribution of tasks and integration of different brands of robots under the same management system. Users can incorporate robots of different brands and purposes directly into the fleet through the VDA5050. This increases the flexibility of businesses, allowing them to respond more quickly and adaptively to various operational needs. In this thesis, the fleet management system has been meticulously designed with clear requirements, a modular structure and the integration of the VDA5050 interface in accordance with traffic management system standards. The system includes basic modules such as Vehicle Management, Driver, Maintenance, Battery, Inventory, Reporting, Alerts and Security. A custom visualization page was also created for the VDA5050 interface. The development incorporates today's software technologies to provide a user-friendly interface, simulation sections and seamless functionality, providing a comprehensive solution for efficient fleet management. As a result, this thesis provides a flexible and efficient VDA 5050 compliant fleet management system interface that facilitates the interoperability of autonomous robots and enables the integration of different makes and models, making the use of autonomous robots in industrial and commercial applications more accessible and efficient. In this way, businesses will be able to take full advantage of the advantages offered by robot technology.

Keywords: VDA5050, Fleet Management System, AGV, AMR, ROS

ÖZET

OTONOM FİLO ROBOTLAR İÇİN VDA5050 STANDARDI İLE GELİŞTİRİLMİŞ FİLO YÖNETİM SİSTEMİ

Bostancı, Bekir

Elektrik Elektronik Mühendisliği Yüksek Lisans Programı

Tez Danışmanı: Doç. Dr. Pınar OĞUZ EKİM

Ocak, 2024

VDA5050, otonom robotların etkin yönetimi ve birlikte çalışabilirliği için geliştirilmiş bir robot filosu yönetim sistemi arayüzüdür. Bu arayüz, otonom mobil robotların kullanımının artmasıyla ortaya çıkan zorlukların üstesinden gelmek için tasarlanmıştır. Birden fazla robotun koordineli çalışması, görevlerin verimli bir şekilde dağıtılması ve farklı marka robotların aynı yönetim sistemi altında entegrasyonu için çözümler sunar. Kullanıcılar farklı marka ve amaçlardaki robotları VDA5050 aracılığıyla doğrudan filoya dahil edebilirler. Bu, işletmelerin esnekliğini artırarak çeşitli operasyonel ihtiyaçlara daha hızlı ve uyarlanabilir bir şekilde yanıt vermelerini sağlar. Bu tezde filo yönetim sistemi, net gereksinimler, modüler bir yapı ve trafik yönetim sistemi standartlarına uygun VDA5050 arayüzünün entegrasyonu ile titizlikle tasarlanmıştır. Sistem Araç Yönetimi, Sürücü, Bakım, Batarya, Envanter, Raporlama, Uyarılar ve Güvenlik gibi temel modülleri içermektedir. Ayrıca VDA5050 arayüzü için özel bir görselleştirme sayfası oluşturuldu. Geliştirme, kullanıcı dostu bir arayüz, simülasyon bölümleri ve sorunsuz işlevsellik sağlamak için günümüz yazılım teknolojilerini içermektedir ve verimli filo yönetimi için kapsamlı bir çözüm sunmaktadır. Sonuç olarak bu tezle birlikte, otonom robotların birlikte çalışmasını kolaylaştıran ve farklı marka ve modellerin entegre edilebilmesini sağlayan esnek ve verimli VDA 5050 ile uyumlu filo yönetim sistemi arayüzü sunularak endüstriyel ve ticari uygulamalarda otonom robotların kullanımı daha erişilebilir ve verimli hale getirilmiştir. Bu sayede işletmeler robot teknolojisinin sunduğu avantajlardan tam olarak yararlanabileceklerdir.

Anahtar Kelimeler: VDA5050, Filo Yönetim Sistemi, AGV, AMR, ROS

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to Assoc. Prof. Dr. Pınar Oğuz Ekim for her guidance and insight throughout the research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

VDA: Verband der Automobilindustrie

VDA5050: Fleet Manager Interface

AMR: Autonomous Mobile Robot

AGV: Autonomous Guided Vehicle

NFC: Near Field Communication

LiDAR: Light Detection And Ranging

IMU: Inertial Measurement Unit

VDMA: Verband Deutscher Maschinen- und Anlagenbau

WMS: Warehouse Management System

MES: Manufacturing Execution System

HTTP: Hypertext Transfer Protocol

URL: Uniform Resource Locator

MQTT: Message Queuing Telemetry Transport

ERP: Enterprise Resource Planning

MAPF: Multi Agent Path Finding

FMS: Fleet Management System

ROS: Robot Operating System

ROS2: Robot Operating System 2

SLAM: Simultaneous Localization and Mapping

DDS: Data Distribution Service

Nav2: ROS Default Navigation Package for ROS2

RMF: Robot Middleware Framework

DTS: Driverless Transport Systems

JSON: JavaScript Object Notation

UTF: Unicode Transformation Format

CAD: Computer-Aided Design

MC: Master Controller

NURBS: Non-Uniform Rational B-Spline

API: Application Programming Interface

ORM: Object-Relational Mapping

Wi-Fi: Wireless Fidelity

CHAPTER 1: INTRODUCTION

It has been observed that autonomous mobile robots have a tendency to function in fleets, particularly in various sectors such as logistics, production sites, electronics, the pharmaceutical industry, and the metal industry (Wesselhöft, Hinckeldeyn and Kreutzfeldt, 2022). Within these sectors, autonomous robots possess significant potential to carry out routine, monotonous, and power-intensive tasks (Aubin, Gorissen and Milana, 2022). Specifically, the utilization of autonomous robot fleets in lieu of human drivers engaged in long-distance and repetitive tasks can enhance work efficiency and minimize errors (René de Koster et al., 2007).

Among the primary challenges encountered by autonomous robots operating in fleets are the following:

1. Traffic Management: The presence of multiple robots operating in the same vicinity necessitates effective traffic management in order to prevent conflicts (VDA, 2022).
2. Energy Management: Due to extended operational durations and the constant need for movement, effective energy management becomes critical (VDA, 2022).
3. Safety: In areas where autonomous robots operate, safety must be of utmost concern. Interactions with humans and adherence to safety protocols are imperative (VDA, 2022).

The VDA5050 protocol serves as a standard means of managing and controlling these fleets of autonomous robots. The key features of VDA5050 are as follows:

Functionality: VDA5050 operates as a protocol designed to control the movement, tasks, and various other parameters of autonomous robots.

Operation: It offers a standardized communication protocol that facilitates the exchange of information among robots, task synchronization, and centralized control. Communication

Method: Typically, VDA5050 employs ethernet as its primary means of communication, enabling information exchange between robots and the central control unit.

An additional noteworthy characteristic of VDA5050 is its distinction from the Master Controller. While a Master Controller serves as the primary computer or control unit responsible for overseeing a single robot, VDA5050 functions as a communication protocol capable of simultaneously controlling multiple robots.

The advantages provided by VDA5050 are as follows:

Multi-Brand Compatibility: VDA5050 can effectively operate autonomous robots from various brands within the same system, ensuring harmonious functionality (VDA, 2022).

Prepared Solutions: As a meticulously prepared standard, VDA5050 already offers solutions to many potential future challenges. By facilitating the more efficient and widespread utilization of autonomous mobile robot technology, this standard fosters a safer and integrated working environment within industrial applications.

1.1 Problem Statement

The number of autonomous mobile robots (AMR) is increasing rapidly day by day. The market for AGV and AMR has demonstrated a resilient expansion in the year 2022, in spite of obstacles such as reductions in capital expenditure, disruptions in the supply chain, inflationary pressures, and uncertainties of a geopolitical nature. Nevertheless, the industry managed to achieve sales totaling approximately $5 billion during that particular year. Projections indicate that there will be an estimated shipment of about 670,000 mobile robots (AGVs & AMRs) for the year 2028, accompanied by an installed base that will reach 2.7 million (AGV and AMR Market, 2023). In order for autonomous mobile robots to contribute to the workforce and make a difference, they need to work in fleets. Robots working in fleets bring many problems with them. The main ones are traffic management, accidents and work optimisation (Singhal et al. 2017). Fleet management system software is developed for mobile robots to eliminate these errors and find solutions. Each company develops these software to manage the robots it develops and to control multiple robots. However, if companies want to use different brands of robots together, a chaos arises, different brands of robots and different fleet management software do not support each other. VDA5050 is a fleet management system communication interface that enables robots of different brands and models to be used in the same environment and a single software to be sufficient for these operations. VDA5050 transmits the operations to be performed by autonomous mobile robots to the fleet management system with a specific structure, and transmits the commands requested by the users and the fleet management system to the robots. Thanks to this standardized interface, the integration of robots is much shorter and effortless without the need for a special process (VDA, 2022).

This thesis covers how autonomous mobile robots work, the problems that may arise from the operation of multiple autonomous mobile robots in the same environment, how the fleet management system will solve these problems, the advantages and disadvantages of VDA5050 and how a VDA5050 supported fleet management system can be developed.

The aim of this thesis is to determine the tasks of fleet management systems, to create a simple VDA5050 supported fleet management system within the framework of the determined tasks and to assign tasks to robots with this fleet management system.

1.2 Outline of the Thesis

The structure of the thesis is given below.

Chapter 2 describes the working principles of autonomous robots in general, what is a fleet management system and what are its general tasks, the general principles of autonomous mobile robots with ROS and the work in fleet management systems.

Chapter 3 how the VDA5050 interface should be used for mobile robots, the message structure in the interface is explained in detail.

Chapter 4 the functions of the fleet management system, existing fleet management systems and their differences, communication protocols and user interfaces in fleet management systems are explained.

Chapter 5 development of a fleet management system as a prototype and its testing and explanation by a VDA5050 supported robot simulation.

Chapter 6 discusses the benefits of a VDA5050-supported fleet management system and how it will progress in this field in the future.

CHAPTER 2: LITERATURE SURVEY

2.1 Autonomous Robots and Fundamental Working Principles

Autonomous robots can be analyzed under 2 headings as robots working in indoor and outdoor areas. The main purpose of autonomous robots working in the indoor area is to move safely from their location to the desired goal. Autonomous mobile robots operating in indoor areas are widely broadcast as 2 types. Autonomous Guided Vehicle (AGV) and Autonomous Mobile Robot (AMR). While general different AGV robots follow the route determined with the help of a line sensor, AMR robots continue on their way with natural navigation, avoiding obstacles in the environment with the help of sensors such as Light Detection and Ranging (LiDAR) and camera (Siegwart, Nourbakhsh and Scaramuzza, 2011).

AGV robots are defined as the first generation mobile robots. They are simpler and robust in terms of software as they move on a fixed route and stop at the obstacles in front of them and wait for the obstacle to be removed from the route. In addition, AGVs do not need to have a localisation algorithm because they move while following the line. Since AGV positions are made only by magnetic line tracking, they are not directly affected by changes in dynamic interior spaces (Wang, 2015). Thanks to the Near Field Communication (NFC) sensors placed underground, the last location information of the robot can be found. Some AGV robots follow their routes using fixed reflectors instead of magnetic line tracking. Positioning algorithms are used in these robots (Lynch et al., 2018).

Autonomous mobile robots consist of much more complex systems than autonomous guided vehicles. The general difference of AMR is that they have a natural navigation system (Včelák et al., 2005). Natural navigation is called robots that perceive the environment and plan their own route without the need for any guidance. For this, they need to have AMR localisation and navigation software and these software process the data from the sensors using different algorithms. The most important issue for navigation in dynamic environments is the error-free implementation of localisation and obstacle avoidance processes. Here, the quality of the sensors and the tuning of the algorithms according to the situation have a great influence. The biggest advantage of AMR over AGV is that they have natural navigation, but this advantage also causes AMR to be complex structures.

Hardware autonomous robots are provided to move with the help of a computer, usually with 2 electric motors, additional sensors (LiDAR, Inertial Measurement Unit (imu), camera, etc.). However, there are many different movement methods (omni directional, ackermann drive, differential drive) (Alatise and Hancke, 2020).

2.2 Fleet Management Systems

Fleet management system is the general name of the software that enables multiple robots to work in the same environment at the same time. The general purpose of fleet management systems, which are divided into many different types, is to enable mobile robots to perform their tasks safely. Fleet management systems enable users to perform many tasks automatically. Fleet manager systems automate operations such as the battery status of robots, which robot should be assigned to which task, in which order the tasks should be performed (Chatzisavvas et al., 2022). In addition, the fleet manager is responsible for storing historical information about the robots by recording the operations on the system in the database. Users can analyze how efficient the robots and their operations are in line with this data. Fleet management systems should also be able to manage multiple robot brands within themselves and send tasks to robots according to the job descriptions of robots.

Fleet management systems generally consist of 4 different sections. Robot - fleet management system communication, fleet management system - frontend communication, data processing and data storage. These four main functions are the basic components that all fleet management systems must have (Ortiz et al., 2021).

2.3 VDA5050 Standard

VDA5050 is an open source fleet manager robot interface developed by the Verband Deutscher Maschinen- und Anlagenbau (VDMA) organization is an open source fleet manager robot interface. The first version of VDA5050 was released in November

2020. The fact that the VDMA organization is involved in the automotive industry has made VDA5050 popular for the needs of AMR and AGVs. VDA5050 has major and minor versions within itself. With each new version, it becomes a structure that covers more robots and situations. It has 2 major versions so far and the biggest difference between the two versions is the robot identification structure called "factsheet".

VDA5050 stated that communication was tested using the message queuing telemetry transport (MQTT) communication protocol. When designing the interface, it was stated that at least 1000 vehicles, vehicles with different driving dynamics can be used and commands should be provided to robots.

Although VDA5050 was initially thought to be developed only for autonomous mobile vehicles developed for the automotive sector, it is thought to be used in robots in different sectors in the future with its widespread use. The reason for this is that a different organization where different brands of robots can be combined under one roof has not been done so far. It is expected that robots using VDA5050 will become widespread as fleet robot usage needs arise in public areas such as hospitals (Valner et al., 2022), schools and shopping centers.

Many robot fleet management software developers have started to include the VDA5050 interface in their systems. Companies such as Node Robotics, Synaos, Kinexon provide VDA5050 support for their fleet managers. In addition, some software companies (Flexus is an example) integrate the VDA5050 interface into their systems, including warehouse management system (WMS) or manufacturing execution system (MES) such as SAP. This means that for many warehouses or factories, they can continue their order processes without changing anything in the WMS or MES they already use (Flexus, 2020).

There are many advantages of using the VDA5050 interface. First of all, since VDA5050 is an open source interface, it is an environment that everyone can use. Without paying an extra cost, robot manufacturers can integrate their robots with VDA5050 and get one step ahead of their competitors in the autonomous robot market. Another advantage is that many automotive companies within the VDMA organization are already using autonomous robots and using their experience in VDA5050. Perhaps the biggest advantage of VDA5050 is that different brands and models of robots can be used at the same time. Normally, in order to integrate the interfaces developed by robotics manufacturers for their own robots into WMS systems, each condition must be re-coded. In addition, a traffic management system must also be coded for the operation of different brands of robots. In addition, since the system that will emerge as a result of these processes will not be tested before, its efficiency will be questionable. However, a robot managed with the VDA5050 interface will be able to introduce itself to the fleet manager with major version 2 and start the task. In addition, errors and requests can be expressed with the community formed under the VDA5050 interface shared on Github (VDA, 2022). Thus, with the further growth of the community in the future, a testable structure that can be developed by everyone far from a single hand is put forward.

However, VDA5050 also has some disadvantages at the moment. With the rapid development of technology, different concepts of robots are produced by robot manufacturers. However, VDA5050 cannot offer solutions to some concept robots with its current versions. Another important disadvantage is that VDA5050, which was originally developed for AGVs, is also used by AMRs today. In addition, with VDA5050, a robot can reuse the node it has traveled over and the start and end nodes only when it leaves them. This is a disadvantage that will cause other robots to wait for nodes that are far from each other. Another issue is the simultaneous localization and mapping (SLAM) issue. As mentioned before, VDA5050 is an interfacing originally developed for AGVs and AGVs do not use SLAM systems. However, this is not the case for AMRs. Therefore, with version 3, it was decided to create a section on how to transfer the maps that will be created in the SLAM case to the robots. Another disadvantage is the map zones. Many robotics companies implement different zones in their robots. The most well-known of these are bidirectional zone, restricted zone, speed limit zone. However, these features are not included in the VDA5050 interface. Although the date announced for the release of VDA5050 major version 3 was determined as the first quarter of 2023, there is still no information on the subject in the fourth quarter of 2023, which is a disadvantage that shows that VDA5050 is progressing slowly. In addition, the fact that there are not many resources on the subject and that no sample code is published by VDMA is a disadvantage for developers.

2.4 ROS (Robot Operating System) and Application Areas

ROS, called Robot Operating System, is a top level operating system prepared to realize robotic applications under one roof. However, ROS is not a real operating system that has a kernel running on the processor. It is defined as a structure that contains certain structures (sub-pub logic, service, parameters, node, package etc...) that are included in robotic applications and should be included in almost every robot (Macenski et al., 2022).

Since the robot operating system was not prepared for real-time applications when it was first prepared and was not sufficient for industrial applications, a higher version, ROS2, was published. The real-time feature that comes with ROS2 has started to be used more for industrial applications with the transfer of data through a data distribution service (DDS) rather than a centralized data (Maruyama, Kato and Azumi, 2016).

The biggest contribution of the ROS ecosystem is that the well-known algorithms already used in the field of robotics have been made available under one roof as open source, open to everyone. Robot companies can develop their robots much more effectively by combining basic software such as navigation and localization with their own concepts. For example, the ROS implementation of the adaptive monte carlo localization algorithm, a very well-known positioning algorithm, is already shared as open source. This increases efficiency considerably (ROS, 2021).

Although the most realized projects on ROS seem to be robot arms and autonomous mobile robots, many different applications can also be realized with ROS. Surface and underwater vehicle applications, drones and advanced robotic systems can also be created using ROS infrastructure. Today, many robotics companies develop their systems either completely using the ROS framework or use ROS for prototyping.

Navigation stacks and localization algorithms developed for mobile robots, as well as ROS-compatible simulation environments, provide an environment where software developers can develop robot software even on their own. The Nav2 package, which is one of the most comprehensive packages in ROS2, offers different global planner algorithms for many different autonomous mobile robots, different local planner algorithms, as well as adaptive systems for different navigation systems (omni directional and ackerman) (Cheng et al., 2022).

The advantages of ROS for the robotics ecosystem are that the applications developed are gathered under a single roof. Just like an operating system, it is ensured that developers can use the application developed by a different software developer on a robot without the need to modify the software to try it on the robot. Another advantage is the creation of an environment where the ROS community can share problems in the field of robotics with each other. Thus, it provides a platform where people experienced in robotics and inexperienced people can come together and discuss problems.

2.5 Current Studies and Gaps

Nowadays, there is a steady increase in the quantity of systems dedicated to managing fleets of robots. This rise is directly proportional to the growing demand for such systems due to the increasing number of autonomous mobile robots in operation. Numerous organizations and companies, both open source and closed source, are actively engaged in addressing this matter. These entities are striving to employ these software solutions in warehouses and production facilities, incorporating features such as route optimization software, task distribution, and intelligent charging capabilities that are tailored to the specific requirements (Singhal et al., 2017). Simultaneously, open source initiatives like Robot Middleware Framework (RMF) and OpenTCS, which serve as non-profit applications, offer an exemplary model for developers.

Collision avoidance is of great importance in fleet management systems. All fleet management systems are developing different algorithms for deadlock prevention and collision avoidance (Hazik et al., 2022). Another issue is energy efficiency. Fleet management systems can greatly increase energy efficiency by directing tasks to robots. At the same time, since battery management systems are under the responsibility of the fleet management system, this is another issue that is being worked on. Since fleet management systems know which robots have how much workload and how far they travel, they can organize regular maintenance intervals and studies are being carried out on this subject (Atik et al., 2023). Another topic is swarm robotics. There are concepts where robots need to act in large units. In this concept, robots move at the same time at the same speed and fleet management software is working on these concepts.

On the other hand, communication standards are one of the most important issues that are open and need to be worked on more in this field. Efforts are being made to enable different brands and models of robots to work together and to implement these systems quickly without customization. The resolution of this issue is accomplished to the implementation of the V5050 interface (van Duijkeren. et al., 2023). However, studies on this subject need to be increased. Another important issue is the unstable behavior of robots in dynamic environments. Most fleet management systems create tasks assuming that the environment is stable and the routes are suitable. However, in real life, this situation is actually much different. The intensity of e-commerce sites on certain weeks and days also occurs in warehouses, and warehouse dynamics change in an instant. Fleet management systems need to improve themselves in dynamic environments. Another issue is the applications of large fleets of robots. VDA5050 sets this number at 1000. However, many fleet management applications increase deadlocks and have problems as the number of square meters per robot decreases. Safety is one of the most critical issues. If a large number of robots in warehouses are hacked by hackers, critical issues such as data privacy may arise, as well as physical accidents can be created by hackers (Wan et al., 2016). The financial damage that may arise in this regard is one of the issues that many fleet management applications do not work on much at the moment.

CHAPTER 3: VDA5050 for AUTONOMOUS MOBILE ROBOTS

3.1 Introduction of VDA5050 Standard

VDA5050 is an interface for data exchange between the central fleet management system and driverless transportation system. Figure 1. shows the communication between the robot and the fleet management system. The interface was prepared by Verband der Automobilindustrie (VDA) and Verband Deutscher Maschinen-und Anlagenbau (VDMA). The aim of both parties in preparing this interface is to create a universal interface. Their aim is to achieve the following objectives by using the existing AGVs and existing Fleet Management Systems VDA5050 interface.

- Description of a standard for communication between AGV and master control and thus a basis for the integration of transport systems into a continuous process automation using co-operating transport vehicles.
- Increase in flexibility through, among other things, increased vehicle autonomy, process modules and interface, and preferably the separation of a rigid sequence of event-controlled command chains.
- Reduction of implementation time due to high "Plug & Play" capability, as required information (e.g. order information) are provided by central services and are generally valid. Vehicles should be able to be put into operation independently of the manufacturer with the same implementation effort taking into account the requirements of occupational safety.
- Complexity reduction and increase of the "Plug & Play" capability of the systems through the use of uniform, overarching coordination with the corresponding logic for all transport vehicles, vehicle models and manufacturers.
- Increase in manufacturers independence using common interfaces between vehicle control and coordination level.
- Integration of proprietary driverless transport systems (DTS) inventory systems by implementing vertical communication between the proprietary master control and the superordinate master control.

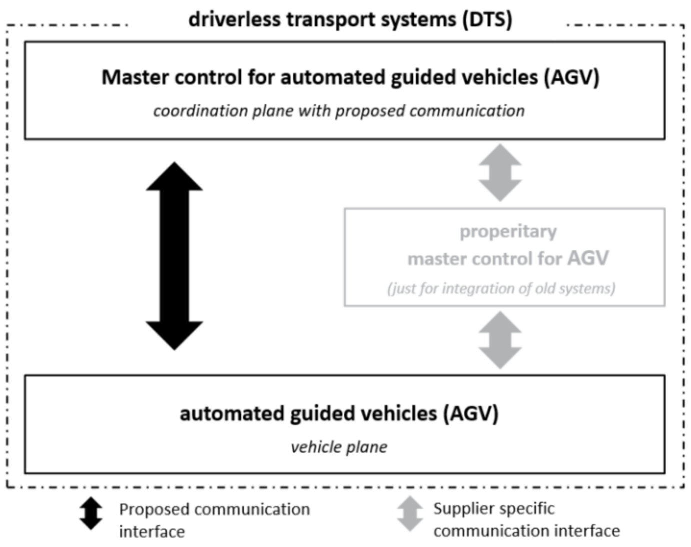

driverless transport systems (DTS)
Master control for automated guided vehicles (AGV)
coordination plane with proposed communication
properitary
master control for AGV
(just for integration of old systems)
automated guided vehicles (AGV)
vehicle plane
Proposed communication interface
Supplier specific communication interface

Figure 1. Integration of DTS inventory systems (Source : VDA, 2022)

In addition, this interface is tailored to the needs of production and plant logistics in the automotive industry. VDA5050 can also be used by AGV, autonomous forklift underrun tractor and AGV with free navigation. VDA5050 interface currently has 2 major versions. The biggest difference between the last major version and the first one is the "factsheet" section. In the future major versions, it is aimed to provide a structure that can control more robots with new features.

3.2 VDA5050 Interface Requirements

There are some requirements for the use of the VDA5050 system. These requirements are listed under different sections in the VDA5050 document. In general, it is expected that the conditions on the usage and communication side are met. If these conditions are not met, errors and accidents may occur in VDA5050 supported fleet management software. Therefore, meeting the requirements is of critical importance. The VDA5050 interface must be designed according to the following requirements.

- Must be able to control at least 1000 robots
- Be able to control vehicles with different driving dynamics, for example omni directional and differential directional robots
- Must be able to make decisions, for example whether to stop or pass at an intersection

- Minimum MQTT version 3.1.1 must be used
- JavaScript Object Notation (JSON) structure used
- Parameters are defined in English so that VDA5050 can be used everywhere
- Some parameters in messages are optional. If it is optional, this message should not be sent to the robot unless it is specified in the factsheet.
  For example, if the AGV is using a trajectory message, it must specify it in the factsheet.
- All communication is Unicode Transformation Format (UTF)-8 encoded.
- Enumerations must be capitalized.
- Where possible, JSON data types must be used. NOT FALSE, TRUE, it must be "false", "true"
- VDA5050 needs to transfer data using MQTT.

3.3 VDA5050 Process and Content of Communication

VDA5050 communication consists of two phases. The first stage implementation VDA5050 interface states that paths must be defined first. Although it is imported with Computer-Aided Design (CAD) file as an example in the document, this method can also be done using the user interface via WebSocket. Afterwards, the charging station, loading and unloading areas should be placed on the route. Finally, AGV information should be given to the Master Controller. This information can also be sent by the robot via factsheet.

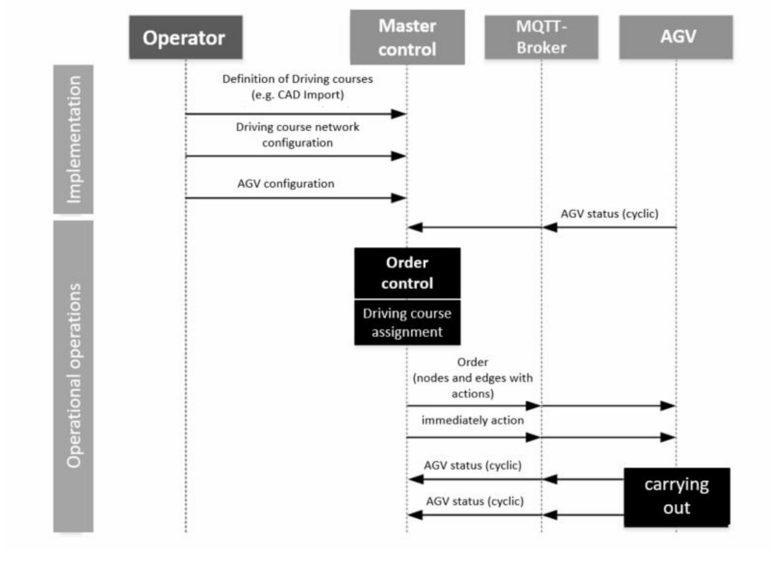


Figure 2. Structure of the Information Flow (Source : VDA, 2022)

In the second stage, the robot sends the status information to the Master Controller via MQTT status topic. If there is a required action in the Master Controller, it transmits this information to the robot via order or instantActions topics.

The tasks of the robot and the master controller are separated from each other. The tasks of the Master Controller are as follows.

- Assigning orders to robots
- Determining the routes that the robots will go according to the configurations of the robots
- Prevent robots from getting stuck
- Sending robots to the charging station by monitoring the charging status of the robots
- Determine the nodes that the robots will use to wait or activate them
- Interacting with and activating other devices in the environment, such as elevators and automatic doors
- Detecting and reporting communication failures
- The tasks of the robot are as follows.
- Localization
- Navigate using the specified nodes
- Continuously updating the status of the robot to the Master Controller

It is also recommended to order topic names in the MQTT broker in this way.
interfaceName/majorVersion/manufacturer/serialNumber/topic
uagv/v1/DEV/robot001/state

Table 1. VD5050 MQTT Message URL Struct Explanation (Source : VDA, 2022)

MQTT Topic Level	Data type	Description
interfaceName	string	Name of the used interface
majorVersion	string	Major version number, preceded by "v"
manufacturer	string	Manufacturer of the AGV (e.g., RobotCompany)
serialNumber	string	Unique AGV Serial Number consisting of the following characters: A-Z a-z 0-9 _ ,. ,:, -,
topic	string	Topic (e.g. Order or System State) see Cap. 6.5

In addition, each message starts with a header and this header is as follows

Table 2. VD5050 MQTT Message Header (Source : VDA, 2022)

Object structure/ Identifier	Data type	Description
headerId	uint32	header ID of the message. The headerId is defined per topic and incremented by 1 with each sent (but not necessarily received) message.
timestamp	string	Timestamp (ISO 8601, UTC); YYYY-MM-DDTHH:mm:ss.ssZ (e.g.“2017-04-15T11:40:03.12Z”)
version	string	Version of the protocol [Major].[Minor].[Patch] (e.g. 1.3.2)
manufacturer	string	Manufacturer of the AGV
serialNumber	string	Serial number of the AGV

3.4 VDA5050 Subtopics Communication

There are 6 different topics in VDA5050. 2 of them go from Master Controller to robot and 4 of them go from robot to Master Controller. It is mandatory to publish all other topics except Visualization.

Table 3. VD5050 Topics (Source : VDA, 2022)

Subtopic name	Published by	Subscribed by	Used for	Implementation	Schema
order	master control	AGV	Communication of driving orders from master control to the AGV	mandatory	order.schema
instantActions	master control	AGV	Communication of the actions that are to be executed immediately	mandatory	instantActions.schema
state	AGV	master control	Communication of the AGV state	mandatory	state.schema
visualization	AGV	visualization systems	Higher frequency of position topic for visualization purposes only	optional	visualization.schema
connection	Broker/ AGV	master control	Indicates when AGV connection is lost, not to be used by master control for checking the vehicle health, added for an MQTT protocol level check of connection	mandatory	connection.schema
factsheet	AGV	master control	Setup of AGV in master control	mandatory	factsheet.schema

3.4.1 Order Topic

Before explaining how the VDA5050 order system works, a few terms need to be explained.

- node: a position on the path that the robot has to take, which also indicates direction.
- edge: connects nodes together
- released: each node and each edge has a released parameters. The robot cannot move on a node or edge with false released parameters. When a robot passes through a node and continues its movement on an edge, the released parameters of the start and end nodes are false for other robots. However, after the robot passes the end node, the parameters are changed to true.
- base: is the name given to the part of the order in the order from the start node to the first released=false parameters node or edge.
- horizon: the part of the order excluding the base. In other words, it is the name given to the part of the order that needs to go but is currently occupied.
- The logic in VDA5050 order is as listed below
- Each node and edge has a sequenceId. The sequenceId of a node is always an even number, while the sequenceId of an edge is always an odd number.
- There must be at least one node in an order. The number of edges is always one less than the number of nodes.
- The first node is the node that the robot is on or assumed to be on. How much this tolerance will be is specified on the factsheet.
- If a node on the order is not released, all subsequent ones must be unreleased.
- Orders sent without horizon are also valid.
- It is not mandatory to send all of an order at once. Order can be updated.

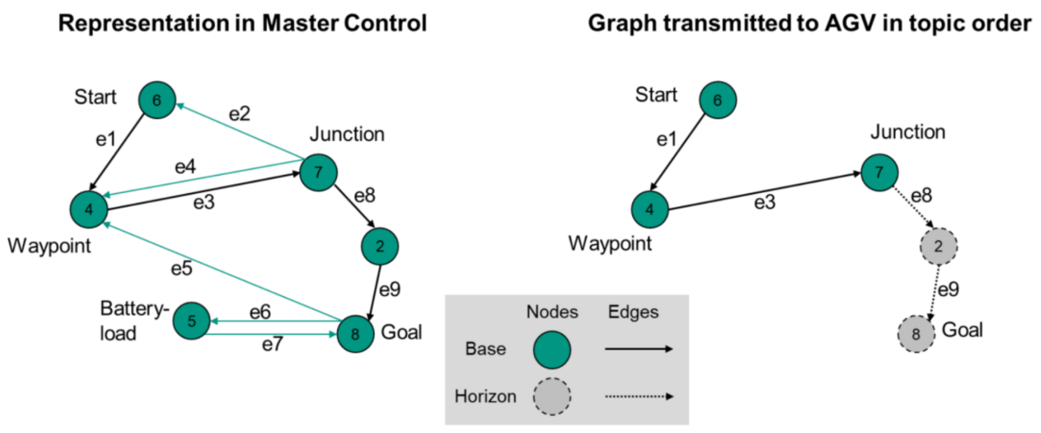


Figure 3. Graph representation in Master Control and graph transmitted in orders (Source : VDA, 2022)

The left side of the figure shows all nodes and edges in the Master Controller as well as the junction points. However, the order node id ends at 8. Fleet manager creates an order by looking at the released parameters of the nodes. On the right side, the nodes and connections are seen in the order. Starting with node 6, the order will first go to node 4 on the e1 edge, then to node 7 with e3, but at node 7 the robot will stop until the order update comes and the e8 edge is released. As shown in the diagram, the part up to node "7" is called Base while the rest is called Horizon.

Order updates can happen for different reasons. If there is a traffic jam on the destination point, the Master Controller can change the path of the robot. While doing this, the orderId must remain constant but the orderUpdateId must be higher than the previous order.

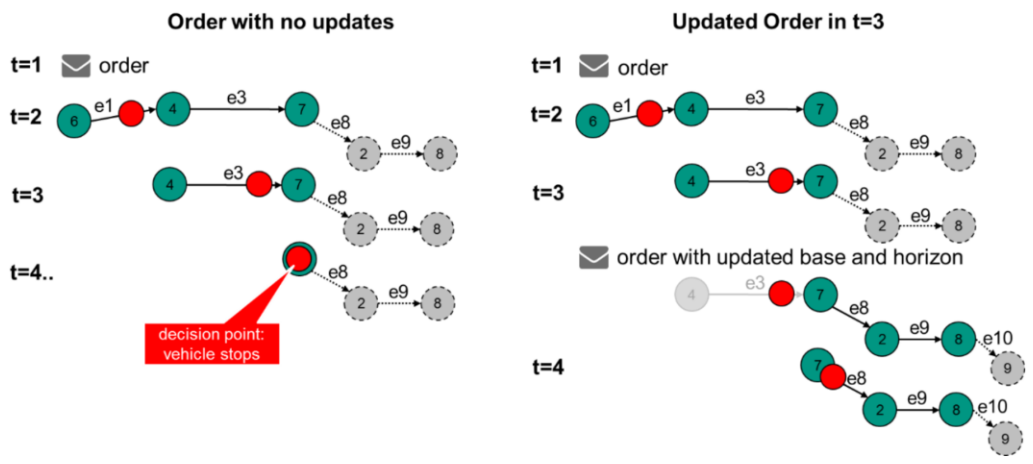


Figure 4. Procedure for changing the driving route "Horizon" (Source : VDA, 2022)

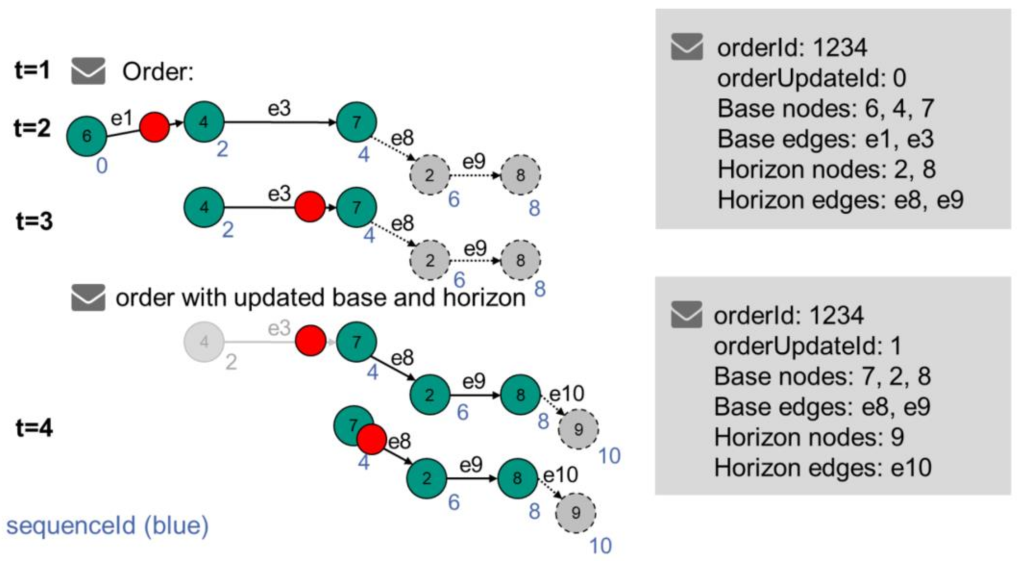


Figure 5. Regular update process - order extension (Source : VDA, 2022)

As in this example, the robot will go to node 7 without receiving any update, but at node 7 the robot will stop and with the update, the goal is updated from node 8 to node 9. The important point here is that the orderId remains the same and the orderUpdateId is a larger value. If orderUpdateId was a smaller value, the robot would not update the order. Or if the orderId was a different value before the order was finished, the new order will be rejected by the robot.

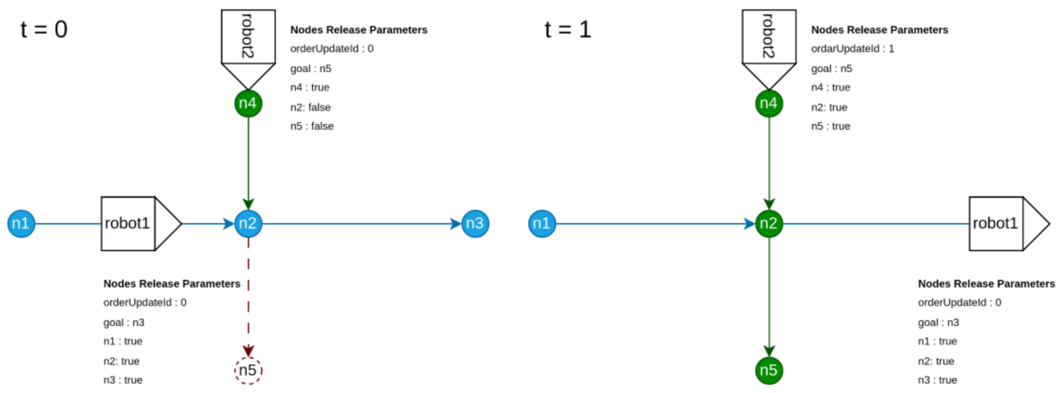
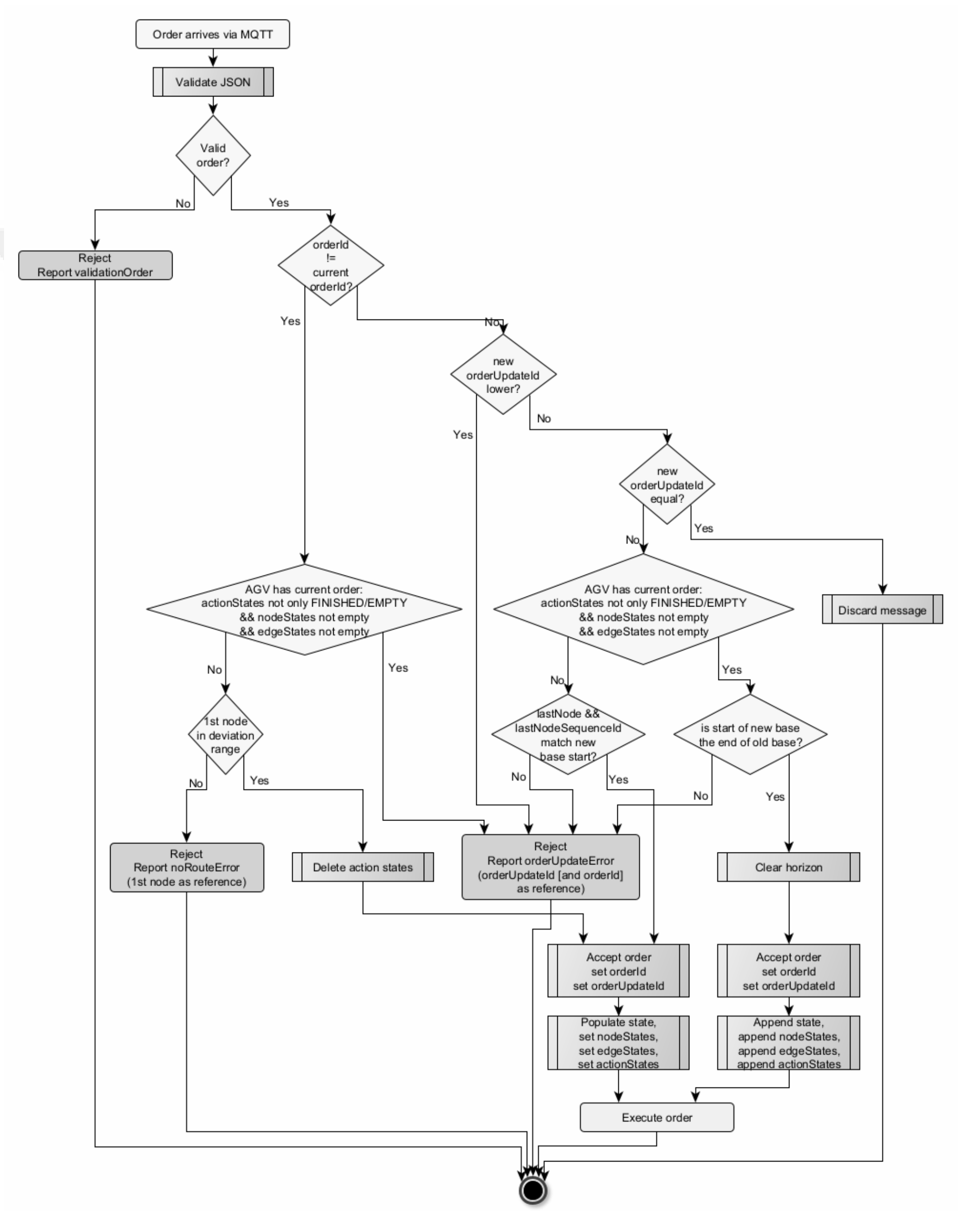


Figure 6. Multi Robot Order Processing with VDA5050

In this example it can be seen that the Master Controller does not allow robot2 to go to node n2 because robot1 is using node n2. However, when robot1 arrives at the destination node, the orderUpdate for robot2 has arrived and the robot can move again. There are 2 different methods to cancel an order. Order rejection and order cancel.

Order rejection is the rejection of the order going to the robot before it starts. There are 3 different types of order rejection: validationError, orderError, orderUpdateError.

- validationError : occurs if the submitted order structure is not correct
- orderError : Occurs if it is a task that the robot cannot perform.
- orderUpdateError : Occurs if a message is received with the same orderId and a lower orderUpdateId.


Order arrives via MQTT
Validate JSON
Valid order?
No
Yes
Reject Report validationOrder
orderId != current orderId?
Yes
No
new orderUpdateId lower?
Yes
No
new orderUpdateId equal?
No
Yes
Discard message
AGV has current order: actionStates not only FINISHED/EMPTY && nodeStates not empty && edgeStates not empty
No
Yes
AGV has current order: actionStates not only FINISHED/EMPTY && nodeStates not empty && edgeStates not empty
No
Yes
1st node in deviation range
No
Yes
lastNode && lastNodeSequenceId match new base start?
No
Yes
is start of new base the end of old base?
No
Yes
Reject Report noRouteError (1st node as reference)
Delete action states
Reject Report orderUpdateError (orderUpdateId [and orderId] as reference)
Clear horizon
Accept order set orderId set orderUpdateId
Accept order set orderId set orderUpdateId
Populate state, set nodeStates, set edgeStates, set actionStates
Append state, append nodeStates, append edgeStates, append actionStates
Execute order
Figure 7. The process of accepting an order or orderUpdate (Source : VDA, 2022)

Order cancel is canceling the order that the robot is executing by the master controller. In order for the user to give a different order to the robot in an emergency, he must first send the cancelOrder instant action to the robot. After the cancelOrder instantAction reaches the robot, the robot will either stay where it is or continue its movement until the next node and stop there. If there is an action that needs to be done, it should be reported as failed in the actionState section in the state topic, and if there is an action already in progress, it should be canceled and reported as failed. If the action cannot be stopped, the cancelOrder action should wait for the action to finish in running status. After all actions are finished and the robot stops, cancelOrder action status is updated as finished.

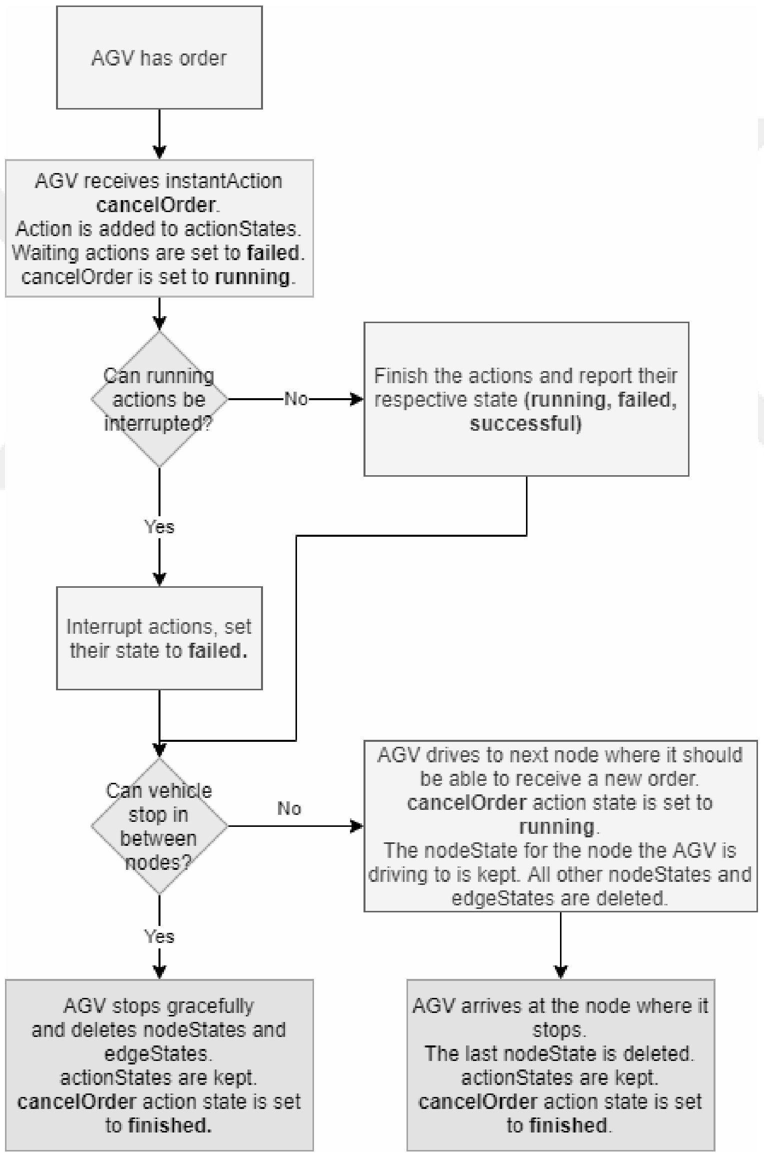

AGV has order
AGV receives instantAction cancelOrder. Action is added to actionStates. Waiting actions are set to failed. cancelOrder is set to running.
Can running actions be interrupted?
No
Finish the actions and report their respective state (running, failed, successful)
Yes
Interrupt actions, set their state to failed.
Can vehicle stop in between nodes?
No
AGV drives to next node where it should be able to receive a new order. cancelOrder action state is set to running. The nodeState for the node the AGV is driving to is kept. All other nodeStates and edgeStates are deleted.
Yes
AGV stops gracefully and deletes nodeStates and edgeStates. actionStates are kept. cancelOrder action state is set to finished.
AGV arrives at the node where it stops. The last nodeState is deleted. actionStates are kept. cancelOrder action state is set to finished.

Figure 8. Expected behavior after a cancelOrder (Source : VDA, 2022)

3.4.1.1 Order Topic Implementation

Every VDA5050 message, the order message has to contain a header. In addition to the orderId, orderUpdateId, zoneSetId parameters that come after the header, there are also nodes and edges arrays. orderId must be unique for each order and ensures that the robot does not confuse the new order with the old orders. orderUpdateId is the parameter that shows whether the same order has been updated. zoneSetId is an optional parameter. If the robot does not specifically request this data, the Master Controller should not send this data. The sections containing the actual contents of the order are the nodes and edges arrays. In these arrays, the path that the robot should take is specified in detail.

Table 4. VD5050 MQTT Message Struct Explanation (Source : VDA, 2022)

Object structure	Data type	Description
headerId	uint32	Header ID of the message. The headerId is defined per topic and incremented by 1 with each sent (but not necessarily received) message.
timestamp	string	Timestamp (ISO 8601, UTC); YYYY-MM-DDTHH:mm:ss.ssZ (e.g."2017-04- 15T11:40:03.12Z")
version	string	Version of the protocol [Major].[Minor].[Patch] (e.g. 1.3.2)
manufacturer	string	Manufacturer of the AGV
serialNumber	string	Serial number of the AGV
orderId	string	Order identification. This is to be used to identify multiple order messages that belong to the same order.
orderUpdateId	uint32	Order update identification. Is unique per orderId. If an order update is rejected, this field is to be passed in the rejection message
zoneSetId	string	Unique identifier of the zone set, that the AGV has to use for navigation or that was used by master control for planning. Optional: Some master control systems do not use zones. Some AGV do not understand zones. Do not add to message, if no zones are used.
nodes [node]	array	Array of nodes objects to be traversed for fulfilling the order. One node is enough for a valid order. Leave edge list empty for that case.
edges [edge]	array	Array of edge objects to be traversed for fulfilling the order. One node is enough for a valid order. Leave edge list empty for that case.

A node is defined as follows. A unique nodeId, a squenceId, this id should be incremental and always consists of even numbers for the node, nodeDescription is an optional parameter and can be added if there is an additional description about the node, the released parameter specifies whether the node is located in the base or in the horizon. In addition, each node has a nodePosition JSON object and an action array. nodePosition object contains the x,y and theta values of the robot, while allowedDeviationXY and allowedDeviationTheta specify how far the robot can be from this node. Each node has a mapId parameter and specifies on which map the node the robot will go to is located. Actions array can be empty or contain multiple actions. A more detailed explanation about actions is in the instantActions section.

An edge contains many different parameters. Like node parameters, edges have a unique edgeId. sequenceId must always be an odd number. Optionally there is an edgeDescription parameter. Released parameter determines whether it is horizon or base, just like nodes. In addition, edges must necessarily have the start and end nodeId on the edge. It also has an action array to run actions on the edge. If the action is still in running state when the edge ends, the action will be stopped.

3.4.2 Instant Actions Topic

Action is the name given to the activities that robots can run. To run an action, either the action must be defined on any node or edge in the order or it must be sent to the robot with the instantAction topic. The instantAction topic and the actions array in the order use the same Json Object. If an action is defined on the edge, the action must be terminated when it exits the edge, but actions defined on the node can run as long as they need to run. There are certain actions that the VDA5050 interface defines as predefined. In addition to these, robot manufacturers can develop custom actions according to the characteristics of the robots. Actions consist of certain statuses: initializing, running, paused, finished, failed. In predefined actions, some actions need to be initialized, while some predefined actions can start the action directly from the running status. Pick and drop actions must be initialized before they are started. However, startPause action can start directly from running state. Also some actions have opposite actions, for example startPause and stopPause. These opposite actions are responsible for doing the opposite of what one of them is doing or stopping it.

Predefined actions are: startPause, stopPause, startCharging, stopCharging, initPosition, stateRequest, logReport, pick, drop, detectObject, finePositioning, waitForTrigger, *cancelOrder*, factsheetRequest.

The instantActions topic has to contain a header like every VDA5050 topic. In addition to the header, only the actions array is included.

Table 5. VD5050 MQTT Instant Actions Structure (Source : VDA, 2022)

Object structure	Data type	Description
headerId	uint32	header ID of the message. The headerId is defined per topic and incremented by 1 with each sent (but not necessarily received) message.
timestamp	string	Timestamp (ISO 8601, UTC); YYYY-MM-DDTHH:mm:ss.ssZ (e.g., “2017-04-15T11:40:03.12Z”)
version	string	Version of the protocol [Major].[Minor].[Patch] (e.g., 1.3.2).
manufacturer	string	Manufacturer of the AGV.
serialNumber	string	Serial number of the AGV.
actions [action]	array	Array of actions that need to be performed immediately and are not part of the regular order (see chapter 6.7).

3.4.3 State Topic

State topic is the VDA5050 interface topic that goes from the robot to the master controller and contains all kinds of information on the robot. In this topic, many detailed data such as the order on the robot, which action is running, which node or edge of the order is sent. The state topic should go to the master controller at least every 30 seconds. However, in some cases, even if 30 seconds do not pass, the message needs to be shared to update the status. These situations are

- When an order is received
- When an order update message is received
- In cases where the load status changes (load on or off)

- When any Error or Warning message is added
- When a node is traversed
- When Operating Mode changes
- When the "Driving" parameter changes
- When any of the nodeStates, edgeStates or actionStates changes

Since another purpose of VDA5050 is to support the simultaneous operation of at least 1000 robots, bandwidth should be reduced as much as possible and unnecessary communication should be avoided. For this, 2 simultaneous states should be sent in a single message, not in different messages.

The robot shares developments on nodeState and edgeState. If the robot calculates its own pathing (all free navigation vehicles must be included in this), they must send the path they calculate in Non-uniform rational B-spline (NURBS) format. If the nodes are sent as released, the robot should not plan again and should apply the path sent.

If the robot determines that there are not enough base nodes left, it can change the newBaseRequest flag to true. The robot can also use the information topic for visualization and debugging. However, this information should not be used for decision making. For example, to update the order path by specifying the obstacles detected by the robot.

The state topic is also responsible for reporting errors to the Master Controller. Errors have two different levels: warning and fatal. If the error is fatal, the robot becomes operational again with human intervention. Detailed information about errors can be passed to the errorReferences array.

3.4.4 Visualization Topic

Visualization topic has the same structure as agvPosition in the state topic. However, the difference is that there is a lot of data in the state topic, and if it is shared at high frequency, high bandwidth will be needed, so it is a topic that only contains robot position information for visualization.

Table 6. VD5050 MQTT agvPosition Struct (Source : VDA, 2022)

Object structure	Unit	Data type	Description
agvPosition {		JSON-object	Defines the position on a map in world coordinates. Each floor has its own map.
positionInitialized		boolean	“true”: position is initialized. “false”: position is not initialized.
localizationScore		float64	Range: [0.0 1.0] Describes the quality of the localization and therefore, can be used, e.g. by SLAM-AGV to describe, how accurate the current position information is. 0.0: position unknown 1.0: position known Optional for vehicles, that cannot estimate their localization score. Only for logging and visualization purposes.
deviationRange	m	float64	Value for the deviation range of the position in meters. Optional for vehicles that cannot estimate their deviation e.g. grid- based localization. Only for logging and visualization purposes.
x	m	float64	X-position on the map in reference to the map coordinate system. Precision is up to the specific implementation.
y	m	float64	Y-position on the map in reference to the map coordinate system. Precision is up to the specific implementation.
theta		float64	Range: [-Pi ... Pi] Orientation of the AGV.
mapId		string	Unique identification of the map in which the position is referenced.
mapDescription }		string	Additional information on the map.

3.4.5 Connection Topic

When the robot connects to the network, it informs the fleet manager. In this topic, the publisher is not only the robot but also the broker can send data on the topic. The reason for this is that when the robot disconnects from the network, it cannot send information to the fleet manager, so this data is given by the MQTT broker. As in every message, there is a header. VDA5050 indicates that there are 3 different connection states in robots. ONLINE, OFFLINE, CONNECTIONBROKEN. Here connection broken indicates that the robot's connection is broken, unlike offlined. However, in offline state, it is understood that the robot is disconnected appropriately. The most important point here is that robots cannot directly change their connectionStates to ONLINE while connecting to the fleet manager. The robot first changes its state to CONNECTIONBROKEN and then it can change it to ONLINE.

Table 7. VD5050 MQTT Connection Topic Struct (Source : VDA, 2022)

Identifier	Data type	Description
headerId	uint32	Header ID of the message. The headerId is defined per topic and incremented by 1 with each sent (but not necessarily received) message.
timestamp	string	Timestamp (ISO8601, UTC); YYYY-MM-DDTHH:mm:ss.ssZ(e.g.“2017-04-15T11:40:03.12Z”).
version	string	Version of the protocol [Major].[Minor].[Patch] (e.g. 1.3.2).
manufacturer	string	Manufacturer of the AGV.
serialNumber	string	Serial number of the AGV.
connectionState	string	Enum {ONLINE, OFFLINE, CONNECTIONBROKEN} ONLINE: connection between AGV and broker is active. OFFLINE: connection between AGV and broker has gone offline in a coordinated way. CONNECTIONBROKEN: The connection between AGV and broker has unexpectedly ended.

3.4.6 Factsheet Topic

Factsheet topic is the topic that allows the robot's information and capabilities to be transmitted to the fleet manager. This topic is the biggest change that came with major version 2 and allows different robots to be transferred to fleet managers. Since this topic was not included in the first major version of VDA5050, it is not possible to transfer the differences of the robots (for example: driving dynamics, dimensions, capabilities, maximum load capacities, etc.). For this reason, fleet managers with VDA5050 major version 1 support have to perform these operations with additional communication with the robots. However, this has been improved with version 2. Also factsheet data is only sent when factsheetRequest instantAction is called.

Factsheet topper is created by combining 7 different JSON objects excluding the header section. Each object contains different objects within itself. The master controller needs to use this data to give the robots tasks that the robot can do.

Table 8. VD5050 MQTT Factsheet Topic Struct (Source : VDA, 2022)

Field	Data Type	Description
headerId	uint32	Header ID of the message. The headerId is defined per topic and incremented by 1 with each sent (but not necessarily received) message.
timestamp	string	Timestamp (ISO8601, UTC); YYYY-MM-DDTHH:mm:ss.ssZ(e.g.“2017-04-15T11:40:03.12Z”).
version	string	Version of the protocol [Major].[Minor].[Patch] (e.g. 1.3.2).
manufacturer	string	Manufacturer of the AGV.
serialNumber	string	Serial number of the AGV.
typeSpecification	JSON-object	These parameters generally specify the class and the capabilities of the AGV.
physicalParameters	JSON-object	These parameters specify the basic physical properties of the AGV.
protocolLimits	JSON-object	Limits for length of identifiers, arrays, strings and similar in MQTT communication.
protocolFeatures	JSON-object	Supported features of VDA5050 protocol.
agvGeometry	JSON-object	Detailed definition of AGV geometry.
loadSpecification	JSON-object	Abstract specification of load capabilities.
localizationParameters	JSON-object	Detailed specification of localization.

CHAPTER 4: FLEET MANAGEMENT SYSTEM DESIGN

4.1 Fleet Management System

Fleet Management System is the software that enables autonomous robots to perform predefined tasks in the same environment and time by multiple users. While autonomous robots fulfill the tasks they receive in their environment, they should understand the obstacles in the external environment and transmit this information to the Fleet Manager and follow the information and orders from the Fleet Manager. In indoor areas, fleet robots are mostly used in production sites or warehouses (Cupek et al., 2020). Autonomous mobile robots are generally used to carry loads, transfer products and materials. However, a number of problems arise when these tasks are to be performed in parallel in the same environment. For example, optimally distributing tasks to robots, organizing traffic between robots, monitoring the charging status of robots and directing robots to empty charging stations.

Fleet management system consists of 4 different parts. Robot to fleet management system communication, fleet management system to robot communication, operations, data storage. In the operations section, there are operations such as defining tasks to robots, determining the paths of robots to tasks, preventing deadlocks, tracking the charging status of robots, traffic management, as well as fleet manager communication with other services, and writing data to the database.

Fleet management systems are divided into different types. In general, these can be divided into Centralized, Decentralized, Hybrid, Cloud-Based, Edge-Based. In Centralized systems, all robots are connected to an application via a network and all decisions are decided by this manager application. However, in Decentralized systems, on the contrary, each robot has the ability to make its own decision and in these systems, robots are connected directly to the database, not to an intermediary application because of that they must solve deadlock by themselves (Sauer et al., 2022). And hybrid systems are systems where both are used together. For example: In Centralized systems, the Fleet Manager works on the main computer, not on the robot, and as tasks are uploaded to the system, it distributes these tasks to the robots appropriately. However, in Decentralized systems, robots connect directly to the database and if there is an idle task, they communicate with other robots to do it. The most appropriate robot takes on the task and continues the system. Cloud-Based systems are more suitable for outdoor use. Since they are not in a specific area, they always need to be connected to the internet via a mobile line and receive their tasks through remote servers. Edge-Based systems are mostly used for indoor areas, especially industrial areas. Considering the situations where the internet can be interrupted, a system positioned on the local network is a great factor to keep the system up and running at all times. In addition, Edge-Based systems are more preferred due to issues such as privacy and security in the industry (Wan et al., 2016). Another advantage of Edge-Based systems is the latency that may occur in the system. Normally, in Cloud-Based systems, the transfer of data causes a certain time delay since the data transfer will be made over a remote server, while this situation is much lower in Edge-Based systems (Wan et al., 2016).

In Fleet Manager systems, there are also some communication protocols that enable robots to communicate with the Fleet Manager or among themselves. These are ROS (ROS2 does not have this feature), MQTT, RESTful Application programming interface (API), WebSocket are the most commonly used ones. Although users can access all robots on the local network thanks to the communication system in ROS1, it brings great security vulnerabilities as there is no encryption on the communication line. MQTT, on the other hand, has become the most popular communication protocol due to its lightweight and already used in IoT projects and possibly approved by the industry (Atmoko and Yang, 2018). RESTful API and WebSocket are among the communication protocols used for robots, although they are mostly used in websites. While RESTFul api is currently created for one-time requests, WebSocket is more suitable for real-time data transfer. The biggest disadvantage of these systems is that they are lightweight and their efficiency decreases when the internet connection is not very good and reconnecting to the network is slower than other systems.

Fleet management systems (FMS) can also be classified in 2 different ways: open and closed source. Many robot software companies develop fleet management system applications. However, some foundations (open robotics), including the community, also develop open source fleet management applications. To give examples of these; closed source fleet management applications : Synaos, Kinexon, Ds Automation, Node

Robotics, Arculus and open source fleet management applications : RMF, openTCS (RMF, 2022).

Fleet Manager Robot Interface enables the transfer of data using a common structure using the communication protocol (e.g. ROS, MQTT etc...) between the robot and the fleet management system. VDA5050 and Mass Robotics Interop interfaces are widely used. Each fleet manager vendor company has their own interfaces designed to harmonize robots with the fleet manager, but many robotics software companies do not release them as open source. However, VDA5050 is published by Verband Deutscher Maschinen- und Anlagenbau e. V. (VDMA), it can be called the most popular robot fleet manager interface today because it is the product of an organization, not a private company, and because it was developed in line with the needs of the automotive industry (VDA, 2022).

In addition, fleet management systems can be connected with warehouse management systems or manufacturing execution systems(3rd party software) (Crosby et al., 2017). Warehouse management systems used in warehouses or production facilities cannot directly transfer tasks to robots. With the help of APIs provided by fleet management systems, tasks from WMS can first be transferred to the fleet management system and then tasks can be transferred to robots through the fleet management system. In this field, which is used by many production facilities and warehouses, data transmission is provided by software plug-ins provided by SAP software. Thus, the integration process can be realized without the need for users to change the systems they are already using.

4.2 Fleet Manager System Functions

The fleet management system needs to fulfill different tasks. These tasks are listed as follows. Order system, battery control, traffic management, data analysis, visualization, error management, data storage (RMF, 2022).

In every fleet management system, a suitable system design should be made for sending orders to robots. However, users do not prefer to change the task assignment systems. Therefore, fleet managers sometimes need to receive tasks as raw data in WMS or Enterprise resource planning (ERP) systems (Crosby et al., 2017). However, the order in which these tasks will be performed and which robot will perform which task is provided by the fleet management application. Different algorithms are used for this process and are generally called multi agent path finding (MAPF) algorithms. Since the problem of the order in which the orders are to be carried by the robots is a nondeterministic polynomial problem, the computation time of the algorithms for this problem is not known. Therefore, fleet manager companies claim that the algorithms they develop work more efficiently.

In battery control systems, two issues are important. The data flow is regular and accurate and the average completion time of the orders is estimated. Many mobile robots use lithium-based batteries and the voltage and charge values in lithium batteries are not linear. For this reason, it is difficult to accurately calculate the charge percentage of the battery pack. Another factor is the calculation of the order estimated completion time. Robots are generally sent to the charging station with a certain threshold. The threshold value is determined by the companies with a reliable percentage, if the threshold value is high, the battery potential of the robot will not be used and if the value is low, it will cause the order to be unfinished with a long order.

Traffic management is the most difficult problem to be executed by the fleet manager. With the diversity of robots and free navigation, fleet managers need to perform traffic management not only on nodes and edges but also in large areas. However, this is not a necessity but an additional feature. Although fleet managers use different methods for traffic management, the logic specified in the VDA5050 order system includes a simple traffic management. Robots going to the horizon and waiting for order update is actually a feature for traffic management. For VDA5050, base and horizon can be calculated by MAPF algorithms and sent to robots (Ma, 2022). Another application is to prevent accidents by using the trajectories and footprints of robots. In this method, a certain part of the robot's trajectory, which is directly proportional to the robot speed, is processed in discrete, regularly spaced positions. By placing a robot footprint at these points, a comparison is made before accidents occur and the accident situation is determined. In the event of a collision, one of the robots waits and the other robot's path is changed if possible to prevent accidents and deadlocks.

Error management is another feature provided by fleet managers for users and technicians to understand the status of robots. With this feature, robot diagnostic data is transferred from the robot to the fleet manager and then to the database and user interface. Users can then re-analyze the errors or warnings saved in the database. This will prevent errors and shorten repair times (Atik, et al., 2023).

A lot of information sent by the robot is analyzed by the fleet manager and stored in the database. This information can be presented to the user as a report in the user interface according to the user's request. In general, this information can be data such as Wireless Fidelity (Wi-Fi) signal strength by location, routes used by robots, average order time.

A user interface where the user can access the fleet manager is another part of the fleet management system. Through this interface, users can see the location of the robots, information such as errors and warnings, upload tasks to the robots and perform all other operations through the interface.

Data storage is the structure that every fleet management system must have. The operations defined by the user and all data from the robot are stored in the database. Maps, map semantics (roads, charging stations, zones etc...), robots and orders are the general sections in the database. Non-relational type databases are more preferred for robotics. The reason for this is that the data sizes are large and they are not related to each other (Ortiz et al., 2021).

4.3 Fleet Management System Communication Systems and Protocols

The fleet management system needs to communicate with the robot and the user interface. These two communication protocols can be different or the same. In addition to these two data transfers, the fleet management system may also need to communicate with third party software. The communication protocol for third party software is determined by the software to be connected (Quadrini, Negri and Fumagalli, 2020).

User interfaces can be web-based projects or desktop or mobile applications. Common data communication protocols, such as HTTP APIs and RESTful APIs, are utilized for web-based applications in user interfaces written in various languages. Moreover, in specific contexts, other protocols like WebSocket for real-time communication or GraphQL for efficient data queries may be employed. The design of RESTful APIs adheres to the principles of Representational State Transfer (REST). The widely used rest-api, with the support of libraries developed for many software languages, creates an interface for different software to communicate with each other. However, rest-api may not always be sufficient for the whole project. Since the HTTP protocol only works on the eventbase, it will return only one response to a request created by the client. In cases where data needs to be transmitted continuously within the application, such as showing the robot position in real time on the user interface, showing the robot charging status, notifying the user of errors and warnings, the HTTP protocol will not be sufficient. Websocket protocol is an alternative for such applications. With websocket applications, two communication methods are possible. Data can be sent continuously from the server to the client in response to a request. In addition, graphql is another architecture design that is more suitable for large data today as opposed to rest-api. In general, the difference between graphql and rest-api is that only the desired parts of the object are retrieved in a request, not the entire object. This gives speed to software developers during the development phase. In very rare cases the user interface can also use MQTT data. However, modern browsers cannot subscribe directly to the MQTT protocol or publish data. Therefore, the websocket feature must be turned on in the broker configurations. In addition, since MQTT does not have a request response system, it will not be a suitable method to forward many user transactions to the server (Kayal and Perros, 2017).

Communication protocols with the user interface can be used for robots to communicate with the fleet management system. However, the communication requirements between the robot and the fleet manager are not the same as the communication requirements between the user interface and the fleet manager. The biggest difference is that robots need to stay connected to the fleet management system as much as possible. With the increase in the number of robots in the fleet, the bandwidth requirement will increase and it will be a great advantage for the communication protocol to use as little data as possible. Considering these aspects, MQTT is the most suitable communication protocol for this process. With its low data usage, high connection rate and low hardware requirements, it is much more advantageous than protocols such as HTTP and Websocket. In addition, the VDA5050 interface states in the document that it should be used together with MQTT in this regard (Yokotani and Sasaki, 2016).

In order for the fleet manager to communicate with 3rd party softwares, it should be learned whether the other software has a communication protocol that can already be used. However, since HTTP is the most common communication protocol today, many software support HTTP protocol. Then, the fleet management system should be informed about the architecture used and the fleet management system should communicate with the software. In addition, there are API documentation of fleet management systems developed by many software companies. By using API documentation, data from 3rd party softwares can also be sent to the fleet management system. The dilemma here is that both the fleet management system can communicate with other software, and other software can communicate with the fleet management system. In an example scenario, a fleet management system needs to be integrated with the SAP WMS system in a warehouse. In this process, the fleet management system can send a request to WMS at certain intervals and ask if there is an IDLE order waiting to be made. If there is a pending order, it can process it within itself and send it to an idle robot. In the opposite scenario, WMS can use the API of the fleet management system to notify the system when there is a pending order. The fleet management system can direct this order to a waiting robot.

Table 9. Compersion Communication Protocols

Feature	HTTP	MQTT	WebSockets	ROS2 DDS
Communication Type	Request-Response	Publish & Subscribe	Full-duplex	Publish & Subscribe
Protocol	Application Layer	Application Layer	Transport Layer	DDS (Middleware Layer)
Message Format	Text (HTML, JSON, XML)	Binary or Text	Binary or Text	Binary
Header Overhead	High	Low	Low	Low
Connection Type	Stateless	Persistent	Full-duplex Persistent	Persistent
Push Capability	No	Yes	Yes	Yes
Use Cases	Web Browsing, REST APIs	IoT, Real-time Messaging	Real-time Applications, Games	Robotics, Real-time Control
Quality of Service	Not built-in	Built-in (QoS levels)	Not built-in (custom implementation)	Built-in (QoS levels)
Security	SSL/TLS for encryption	SSL/TLS for encryption	SSL/TLS for encryption	DDS Security
Overhead	Relatively High	Low	Low	Low
Connection Speed	Slower (multiple requests)	Faster (persistent)	Faster (full-duplex)	Faster (persistent)
Scalability	Scales with more servers	Scales well with brokers	Scales well with connections	Scales well with connections

4.4 User Interface and Accessibility

The user interface is another part of the fleet management system. It allows the users to observe the robots and all data to be entered into the fleet management system by the users. Fleet management system user interfaces, which are usually created with a web-based system, can sometimes be desktop applications. In addition, mobile applications or web applications that support mobile devices are also developed to enable users to use the system from mobile devices. Generally, they can be examined under 3 main headings where maps, robots and orders are seen. Analysis and reports are also presented to users by the user interface. In advanced fleet management systems, the maps of the robots are made through the user interface, thanks to this feature, end users can perform the slam operation without additional support. In addition, the user interface is of great importance for designing the areas that robots can use, determining the paths, and creating restricted areas. Otherwise, users need to make the necessary measurements on the map themselves and send it to the robot with the help of a command. User interfaces also play an important role in defining tasks and orders. Another issue is the initialization of the robots. The initial positions of the robots need to be sent to the robots manually. A special predefined action is defined for this process in VDA5050. The "initPosition" action sends the current position of the robot to the robot. If this feature is not provided by the user interface, the user must do this manually.

As mentioned before, user interfaces can be developed for different platforms. However, the high performance of modern browsers, the fact that web-based applications can be used by mobile phones, and the fact that they can be accessed from all operating systems without depending on the operating system provide great advantages over web-based user interfaces. Another advantage is that the user can directly access the fleet management system with any browser without the need for an additional application (Bal, 2013). However, different companies also develop user interfaces as desktop applications for security reasons. In addition, some service robots display their user interfaces to the user via tablets on their own. In these cases, the most preferred platform is Android. In web-based applications, javascript and typescript are highly preferred because they are popular and contain many libraries. Since desktop applications are generally developed for windows, c# and java languages are preferred. Through the user interface, users should be able to perform all the permitted actions they want to perform on robots without depending on the manufacturer. These can be briefly listed as map creation, task definition, robot error and warning tracking, battery control operations (RMF, 2022). Advanced fleet management systems include many more details. For example, changing robot configuration values, detailed analysis pages, different types of map semantics (restricted zone, charging area, parking area, speed zones etc...).

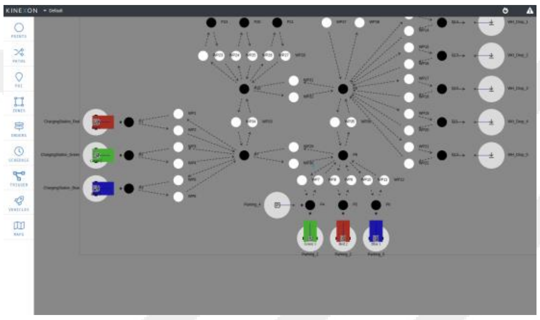


Figure 9. Kinexon User Interface

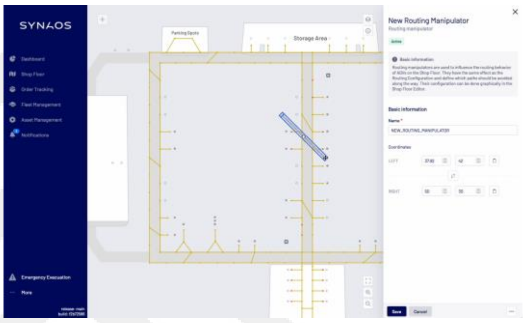


Figure 10. Synaos User Interface

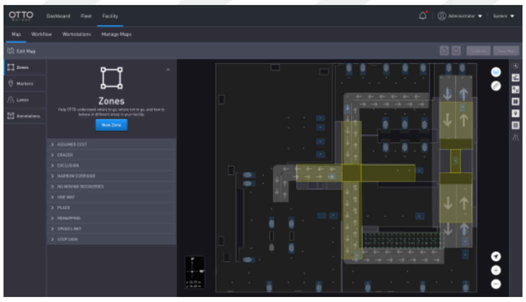


Figure 11. Otto Motors User Interface

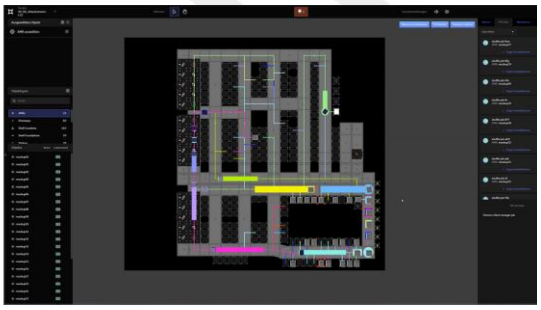


Figure 12. Arclus User Interface

CHAPTER 5: IMPLEMENTATION AND RESULTS

5.1 Prototype Development

The fleet manager application to be developed must be supported by the VDA5050 interface. Central fleet management systems must have a backend and a frontend software in the application (Ortiz et al., 2021). There is also a need for a VDA5050 supported robot for testing.

The structure that will form the center of the fleet management system is the backend software. The backend should contain the fleet management system and the VDA5050 master control. It also needs a database to save the data. In order to realize the fleet management system software, it is necessary to choose a platform on the backend side. Languages such as Java, Python or Javascript/Typescript have many backend frameworks. The appropriate language and framework should be chosen according to the needs of the system to be created. On the Python side, there are very popular backend frameworks such as Django and Flask. However, it is known that most fleet management applications use Javascript. The biggest reasons for this are that many different libraries are supported by Javascript, the NodeJs working environment is reliable, it can be written in the same language as the frontend software to be developed, and it supports asynchronous processes.

Using object-relational mapping (ORM) on the database side will bring many advantages. ORM is used to avoid being dependent on any database. In addition, no custom code or syntax needs to be written for any database operation with ORM. Therefore, it is appropriate to use ORM within the project. TypeORM is a popular library that is widely used with typescript (Zhang and Ritter, 2001).

On the frontend side, there are many different languages and frameworks as on the backend side. There are different frameworks such as Flutter, Vue, React, Angular for a web-based user interface. An application created with Flutter will be a great advantage when creating a native mobile application in the future. In addition, user interface customizations Javascript or Typescript. Vue, React and Angular are Javascript based frameworks. Angular is older and more cumbersome than Vue and

React. Although React is popular today and has community support, it is predicted that Vue will be more supported by the community in the future because it is more up-to-date and intuitive. In addition, when vue and other frameworks are compared, it is stated that vue is more efficient in terms of performance. For these reasons, vue will be preferred in the project (Au-Yeung, 2021).

In the system to be created, the robot is simply asked to go to certain points on the paths given to the robot and collect and return to the storage area at the last point and perform the drop-off action. For this scenario, a map and the nodes on the map are needed. The connections and relations of the nodes with each other will be defined in the database with vertex and edges, and then the shortest path for the robot to reach the target will be calculated with a path finding algorithm. The order to be transmitted to the robot will be delivered to the robot with the MQTT protocol in VDA5050 standards, then the robot will go to the location or locations specified in the order and after the pick operations are performed, it will go to the drop point and the order will be terminated with the drop action.

In order for VDA5050 messages to arrive, an MQTT broker must also run on the backend side. The broker does the task of receiving MQTT messages from publishers and sending them to subscribers. While doing this, the broker can be on the same network or in the cloud, but brokers running in the cloud require constant internet connectivity and will also cause a delay. There are many open source brokers and each one has its advantages and disadvantages. Mosquitto will be preferred in the project because it is developed in C++, easy and fast to install and open source (Hillar, 2017).

Incoming MQTT messages should be checked whether they match VDA5050 objects. Different types of messages should be rejected by the fleet manager and only valid messages should be accepted. The topics that are published on fleet manager side are order and instantActions topics. In addition, the fleet manager should also discover other topics in the MQTT interface name and subscribe to the topics it finds. MQTT.js library will be used for these operations.

In addition, a VDA5050 based robot is needed to test the created orders. For this, a ROS based turtlebot and VDA5050 adapter will be used. The robot will be made to move by using the VDA5050 apter, which is available on Github (VDA, 2022) and developed as open source.

5.2 Implementation of the Prototype

First of all, the backend side needs to be created for fleet manager and VDA5050 implementation. The backend should be created using the libraries and frameworks mentioned earlier. Since the fleet manager system backend will be built using the NestJS framework, the NestJS framework must first be downloaded to a specific location. At the same time, the system should be installed with the @nestjs/cli library. Thus, errors can be prevented. Then the backend installation is completed with "nest new fleet-manager". When this command is executed in the terminal, the appropriate package manager will need to be selected. After this command, basic files will be created in the file. Then TypeORM should be installed with the "npm i typeorm" command. After that, NestJS cli can be used again for the modules to be created. For this, "nest g module module_name" command should be used. Currently, the fleet management system needs to have robot, map, order and adapter modules in our project. Thanks to the modular system, the structure will not depend on each other. Changes that may occur in the future, for example, the development of a different adapter instead of VDA5050 or the creation of a new version of VDA5050 will ensure that the existing structure is used without changing. After creating the modular system here, the structure will be in the form of Figure 13.

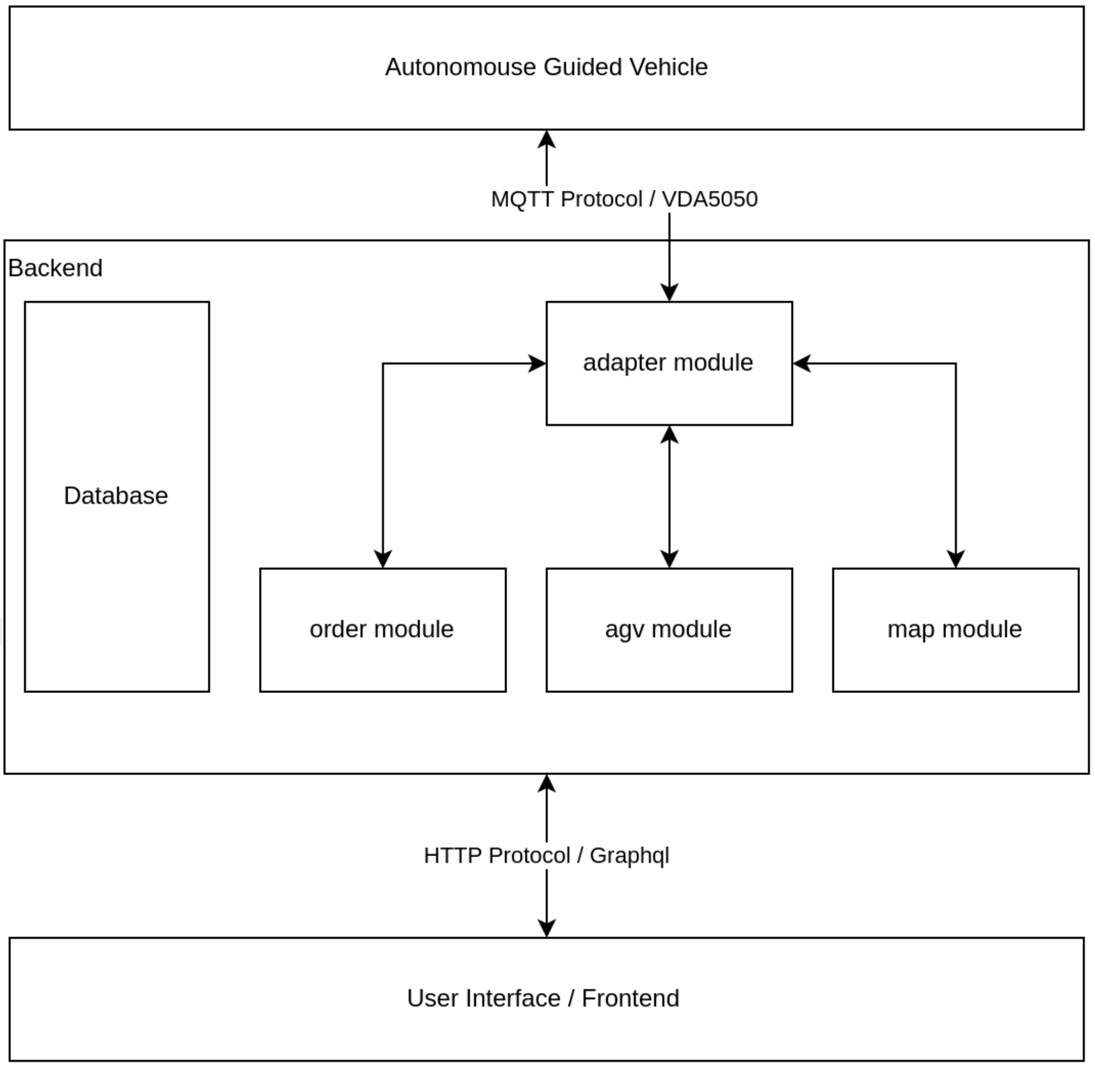


Figure 13. Fleet Management System Module Structure

First of all, robots need to have a map before creating a task. Therefore, the fleet management system needs to include a type that contains the properties required for the map. The map object should contain a unique id, map name and dimensions to display the map. It should also contain a nodes array that will hold the routes that will take place in the map. Each node in the nodes array must contain a unique node id, node location and the nodes it is connected to. These operations should take place in the NestJS service class. TypeOrm should be used to save these transactions to the database.

Secondly, a robot must be created. Each robot must have a unique serial number. The manufacturer information under VDA5050 agvId can be added. All the information obtained with Factsheet should be included under the robot data. Robot position information should be updated with the information obtained from the visualization topic. In addition, the information in the VDA5050 state topic should be saved in the database.

Thirdly, order classes should be created. VDA5050 has made all scenarios free in terms of order. However, in the current scenario, the robot needs a pick and a drop node. Each order should have a unique id. Also the orders need to have a state. Thus, afterwards, analyses such as how many successful and how many unsuccessful orders can be analyzed. Also, data such as order creation time, start time and finish time can be stored in the database for analysis.

Finally, the adapter design will be the part where the fleet management system is designed. The master controller in VDA5050 will be represented here.

HTTP protocol and REST API will be used to transfer data and requests from the user interface to the fleet management system. For this, controllers should be created for each module. In addition, data that needs to be transferred in real-time should be transferred using websocket. When the user completes a new order, it will first come to the backend, i.e. fleet manager, via the request API. After the order module here creates the order, it will create a VDA5050 order in the VDA5050 adapter in the adapter module. Before the order is created, the shortest path will be calculated with the dijkstra algorithm using previously created nodes and vertices. The nodes and vertices on the calculated shortest path will be created and transmitted to the robot via MQTT.

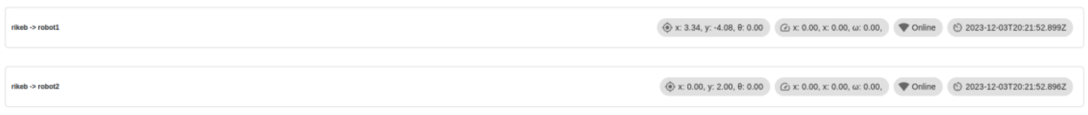
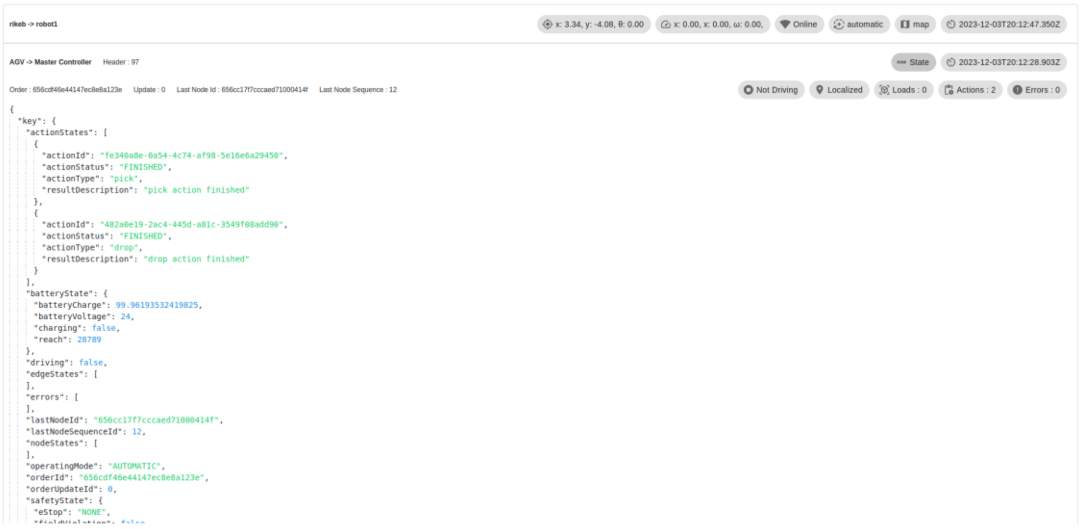

```
rikeb -> robot1    x: 3.34, y: -4.08, θ: 0.00    x: 0.00, x: 0.00, ω: 0.00,    Online    2023-12-03T20:21:52.899Z
rikeb -> robot2    x: 0.00, y: 2.00, θ: 0.00     x: 0.00, x: 0.00, ω: 0.00,    Online    2023-12-03T20:21:52.896Z
```

Figure 14. Fleet Management System VDA5050 Visualizer



rikeb -> robot1   x: 3.34, y: -4.08, θ: 0.00   x: 0.00, x: 0.00, ω: 0.00,   Online   automatic   map   2023-12-03T20:12:47.350Z

AGV -> Master Controller   Header : 97   State   2023-12-03T20:12:28.903Z

Order : 656cdf46e44147ec8e8a123e   Update : 0   Last Node Id : 656cc17f7cccaed71000414f   Last Node Sequence : 12   Not Driving   Localized   Loads : 0   Actions : 2   Errors : 0

```
{
  "key": {
    "actionStates": [
      {
        "actionId": "fe340a8e-0a54-4c74-af98-5e16e6a29450",
        "actionStatus": "FINISHED",
        "actionType": "pick",
        "resultDescription": "pick action finished"
      },
      {
        "actionId": "482a0e19-2ac4-445d-a81c-3549f08add90",
        "actionStatus": "FINISHED",
        "actionType": "drop",
        "resultDescription": "drop action finished"
      }
    ],
    "batteryState": {
      "batteryCharge": 99.96193532419825,
      "batteryVoltage": 24,
      "charging": false,
      "reach": 28789
    },
    "driving": false,
    "edgeStates": [
    ],
    "errors": [
    ],
    "lastNodeId": "656cc17f7cccaed71000414f",
    "lastNodeSequenceId": 12,
    "nodeStates": [
    ],
    "operatingMode": "AUTOMATIC",
    "orderId": "656cdf46e44147ec8e8a123e",
    "orderUpdateId": 0,
    "safetyState": {
      "eStop": "NONE",
      "fieldViolation": false
```

Figure 15. Fleet Management System VDA5050 Visualizer State Topic

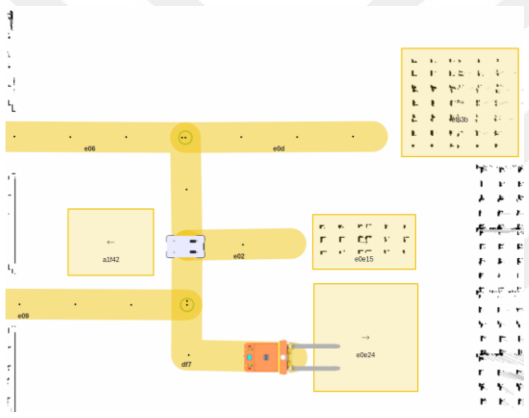


Figure 16. Fleet Management System User Interface Multi Robot

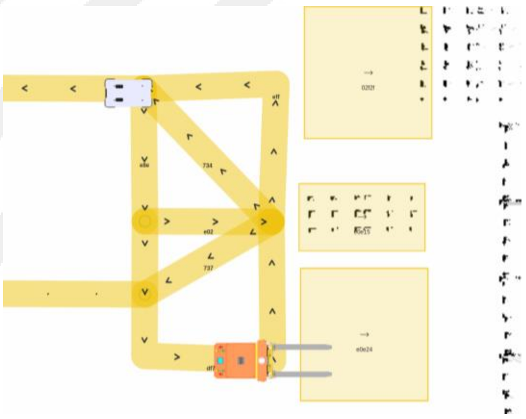
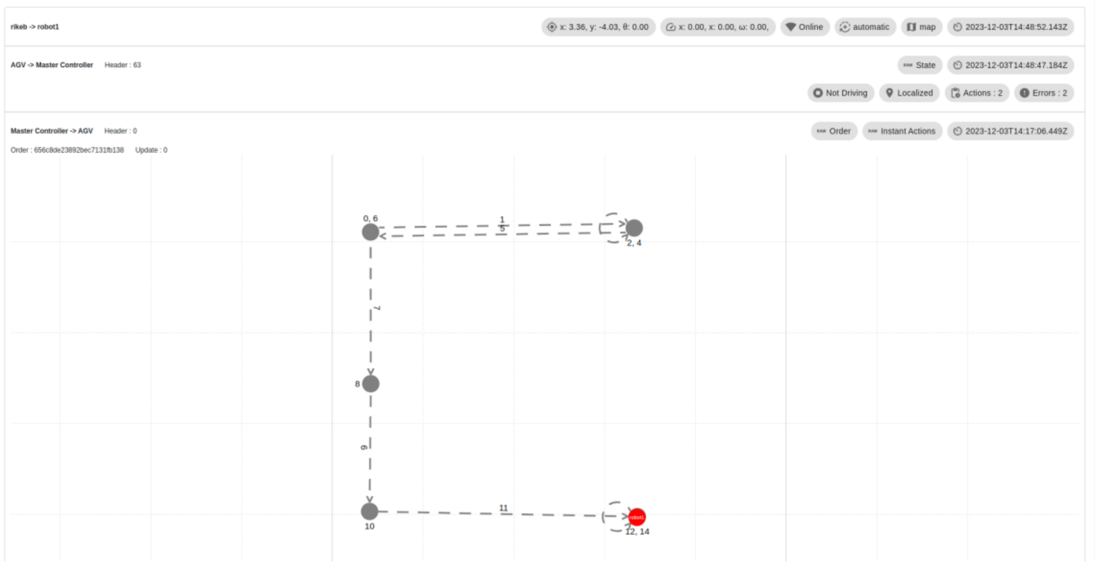


Figure 17. Fleet Management System User Interface Multi Robot

After the data is transmitted to the VDA5050-powered robot, the robot will share the state message at regular intervals. This message should be shared at maximum 30 seconds intervals. The state of the order is understood by considering the state message from the robot. Here, for the order to be finished, the actions in the order must be in Finished or Failed state. Also, nodeState and edgeState arrays must be empty. If any of the actions has a failed flag, it can be said that the order was not successful. Fleet management systems can create recovery states for such situations, but these are not the actions specified by the VDA5050 standard.

5.3 Performance Evaluation and Results

In the example scenario, the order reached the robot at time t=0 in Figure 19. The robot needs to pick up the pallet on sequenceId 2 node at time t=1 in Figure 20. In Figures 21 and 22, the robot follows the path towards the final node it wants to reach. In the 23rd Figure, the robot ends the order with a drop action. There are pick and drop actions in the order at Figure 18 it shows. The yellow lines show the roads. The points where the roads intersect, start or end show the nodes. The sequenceIds of the nodes are specified as seen in Figure 18. It is seen that the robot first goes to node 2 and then comes back to the same place. This is because after the pick operation, the robot returns to the node where it was before the pick operation. Then the robot goes to node 12 by following the nodes again and the drop action starts. Again, one more node is added to ensure that the robot is in the same position after the action ends.


rikeb -> robot1 | x: 3.36, y: -4.03, θ: 0.00 | x: 0.00, x: 0.00, ω: 0.00, | Online | automatic | map | 2023-12-03T14:48:52.143Z

AGV -> Master Controller | Header : 63 | State | 2023-12-03T14:48:47.184Z

Not Driving | Localized | Actions : 2 | Errors : 2

Master Controller -> AGV | Header : 0 | Order | Instant Actions | 2023-12-03T14:17:06.449Z

Order : 656c8de23892bec7131fb138 | Update : 0

Figure 18. Fleet Management System VDA5050 Visualizer Order Graph

As different robots support the VDA5050 interface, new robots will work directly with the same fleet management system. The short integration time and the fact that they are already tested systems have greatly reduced the installation time for the first transport.

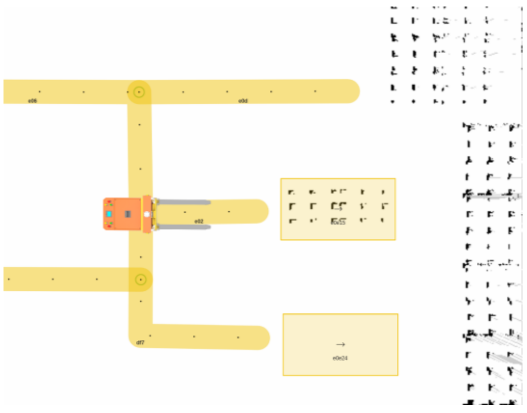


Figure 19. FMS Order t=0

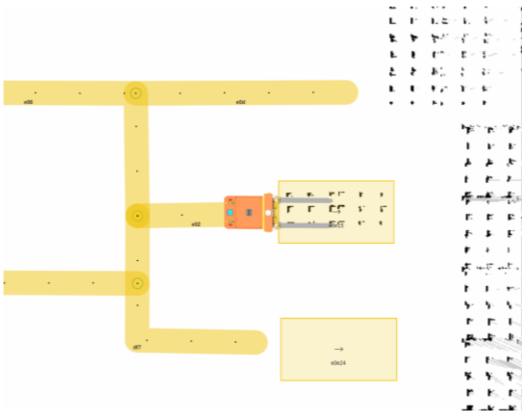


Figure 20. FMS Order t=1

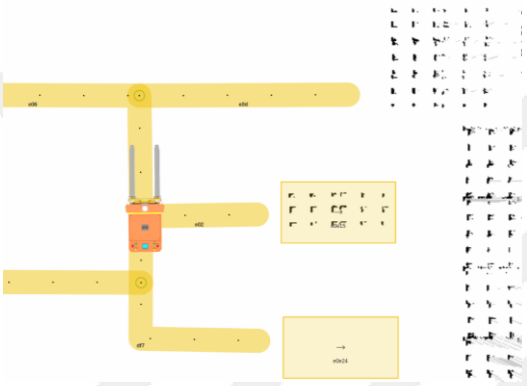


Figure 21. FMS Order t=2

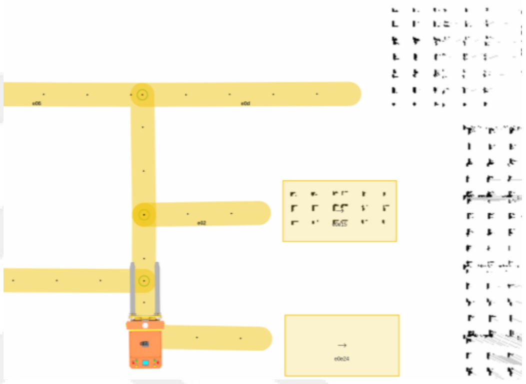


Figure 22. FMS Order t=3

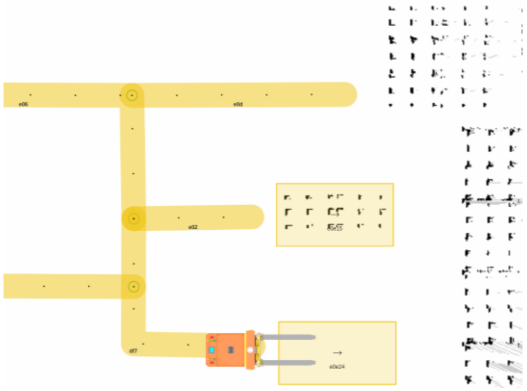


Figure 23. FMS Order t=4

As shown in Figure 24, many features of the VDA5050 can be visualized with the VDA5050 visualization page. This section consists of 3 different areas. The header area at the top shows general information about the robot. This visualization process is provided with information from the VDA5050 visualizer topic and the state topic. Robot manufacturer, robot name, map name and operating mode information are provided using the state topic. Position and speed information is provided using the visualization topic. In the second section, the data coming from the Master Controller to the fleet management system is visualized. The chips here also act as buttons. 24. The tasks of the chips are written on Figure 24. When the State chip is pressed, the last JSON object on the State topic is shown. At the same time, this data is changed as current. In addition, the order information (order update, last node id, last node sequence) information in the state topic is also visualized using the information from the state topic. In addition, the state information of the robot such as initial position, loads actions and errors are also shown in this section. In the 3rd area, the information coming from the fleet management system to the robot is shown. This information comes from the order and instantActions topics. In addition, thanks to the graph in this area, the nodes and edges that the order has are visualized.

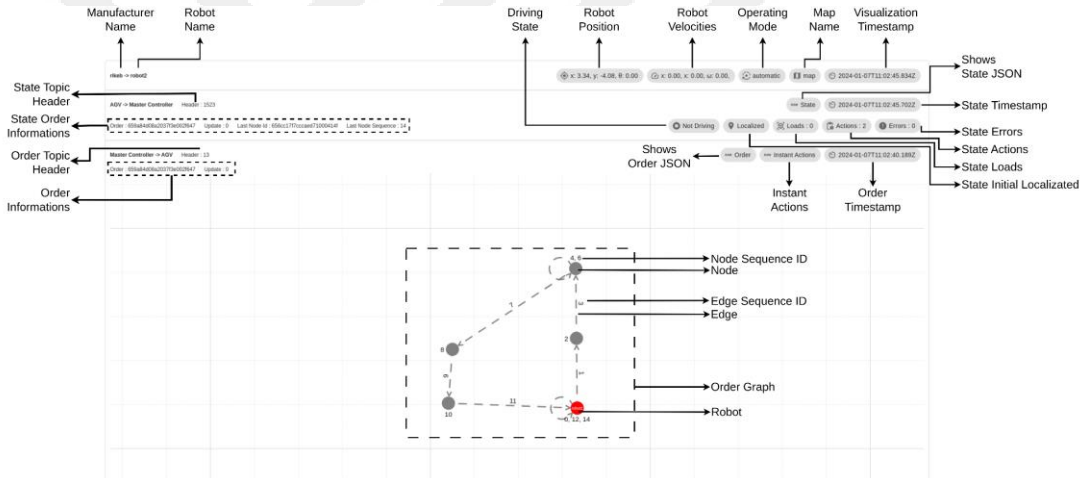


Figure 24. VDA5050 Visualizer Diagram

5.4 Potential Uses and Benefits of the Application

Although the general usage area of the application is warehouse and production facilities, it provides great benefits for every area where multiple robots will be used. While the standardization of many things in the world provides benefits to users, it reveals that this standardization is an important requirement for autonomous mobile robots. In addition to warehouses and production facilities, it is possible to use VDA5050 as a standard in the future in systems such as public spaces such as schools, hospitals and containers stored outdoors.

When focusing on the benefits offered by the system, the quick and easy integration process seems to be the biggest advantage. It is also an important factor in preventing robot accidents and reducing material losses. In addition to this, another important factor is that robots move more efficiently thanks to the optimization of order processes. This means that work processes can be carried out in a more organized, faster and error-free manner, which increases overall productivity.

The wide range of uses of VDA5050 increases its applicability for businesses in different industries. Adoption of standards allows industrial automation to be adopted by a wider range of users. This contributes to the development of industrial processes and makes it possible to manage business processes more effectively in various sectors.

CHAPTER 6: CONCLUSION AND FUTURE WORK

The use of mobile robots in fleets, the tasks of the fleet management system and the advantages of robots using the VDA5050 interface are described in detail in this thesis. The tasks that arise when multiple robots are used at the same time and in the same area are performed by the fleet management system. Today, many different types of fleet management systems are realized by different companies. The aim of these fleet management systems is to increase efficiency and ensure that robots perform their tasks safely. However, the fleet management systems developed by companies usually only enable the control of their own robots, and in environments where different brands of robots are used, large integration costs and time losses arise. Thanks to the VDA5050 interface developed by the VDMA organization, an open source fleet management system interface that can be developed by anyone is offered to manufacturers and users. Thanks to this interface, manufacturers or developers can include the VDA5050 interface in their robots or fleet management systems and it can be used by supported robots.

Although the VDA5050 system is an interface prepared with many situations in mind, it also contains many shortcomings for different fields. In future studies, first of all, the mapping system (delivery of maps to robots) should be improved in the VDA5050 system. Also, in case of a slam, the VDA5050 interface should create an option to transfer map data to the user. Another issue is traffic management. The fleet management system implemented in this thesis does not include an advanced traffic management system. The VDA5050 order logic system contains vulnerabilities (nodes that are too close to each other or robots that are too large) that create the possibility of collisions with each other. VDA5050 interface developed by VDMA was initially developed with only warehouse and production facilities in mind, with the increasing number of autonomous robots, the need for fleet management systems will increase. In addition, transportation problems will occur not only in warehouses and production facilities but also in different areas and in the future, autonomous mobile robots will be used in public areas such as hospitals, schools, shopping malls, hotels and entertainment centers. Therefore, standardization in autonomous mobile robots will be a need.

REFERENCES

AGV and AMR Market (2023) *Mobile Robots Market Opportunity worth ~$20B by 2028 with an installed base of 2.7 Million Robots - Driven by Logistics & Manufacturing, Market Forecast till 2028* [Online]. Available at: https://www.researchandmarkets.com/reports/5398204/agv-and-amr-market-mobile-robots-market. (Accessed: 4 January 2024)

Alatise, M. and Hancke, G. (2020) *A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods*, IEEE Access, Vol. 8, pp. 39830-39846.

Atik, S. T., Chavan, A. S., Grosu, D. and Brocanelli M. (2023) *A Maintenance-Aware Approach for Sustainable Autonomous Mobile Robot Fleet Management, IEEE Transactions on Mobile Computing*, pp. 1-14.

Atmoko, R. A. and Yang, D. (2018) *Online Monitoring & Controlling Industrial Arm Robot Using MQTT Protocol, IEEE International Conference on Robotics, Biomimetics and Intelligent Computational Systems (Robionetics)*, Bandung, Indonesia, pp. 12-16.

Au-Yeung, J. (2021) *Vue. js 3 By Example: Blueprints to learn Vue web development, full-stack development, and cross-platform development quickly*: Packt Publishing Ltd.

Aubin, C. A., Gorissen, B. and Milana, E. (2022) *Towards enduring autonomous robots via embodied energy*, Nature, Vol. 602, pp. 393-402.

Bal, S. N. (2013) Mobile web—Enterprise application advantages, International Journal of Computer Science and Mobile Computing. Citeseer, Vol. 2(2), pp. 36-40.

Chatzisavvas, A., Chatzitoulousis, P., Ziouzios, D. and Dasygenis, M. (2022) *A Routing and Task-Allocation Algorithm for Robotic Groups in Warehouse Environments, Inf.*, Vol. 13, p. 288.

Cheng, P. D. C., Indri, M., Sibona, F., De Rose, M. and Prato, G. (2022) ‘Dynamic Path Planning of a mobile robot adopting a costmap layer approach in ROS2’, in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1-8.

Crosby, M., Petrick, R., Rovida, F. and Krueger, V. (2017) *Integrating Mission and Task Planning in an Industrial Robotics Framework, Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 27(1), pp. 471-479.

Cupek, R., Drewniak, M., Fojcik, M., Kyrkjebø, E., Lin, J. C. W., Mrozek, D., Øvsthus, K. and Ziebinski, A. (2020) *Autonomous Guided Vehicles for Smart Industries -- The State-of-the-Art and Research Challenges*, in Krzhizhanovskaya, V. V. and Valeria, V. (eds) Computational Science -- ICCS 2020. Cham: Springer International Publishing, pp. 330-343.

Flexus (2020) *SAP VDA5050 integration* [Online]. Available at: https://www.flexus.de/en/glossary/vda-5050/. (Accessed: 24 December 2023)

Hazik, J., Dekan, M., Beno, P. and Duchon, F. (2022) *Fleet Management System for an Industry Environment*, *Journal of Robotics and Control (JRC)*, Vol. 3(6), pp. 779-789.

Hillar, G. C. (2017) *MQTT Essentials-A lightweight IoT protocol*. Packt Publishing Ltd.

Kayal, P. and Perros, H. (2017) A comparison of IoT application layer protocols through a smart parking implementation, in *2017 20th Conference on Innovations in Clouds,* Internet and Networks *(ICIN),* pp. 331-336.

Lynch, L., Newe, T., Clifford, J., Coleman, J., Walsh, J. and Toal, D. (2018) *Automated Ground Vehicle (AGV) and Sensor Technologies- A Review, 2018 12th International Conference on Sensing Technology (ICST)*, Limerick, Ireland, pp. 347-352.

Ma, H. (2022) *Graph-Based Multi-Robot Path Finding and Planning*, Current Robotics Reports. Springer Science and Business Media LLC, Vol. 3(3), pp. 77-84.

Macenski, S., Foote, T., Gerkey, B., Lalancette, C. and Woodall, W. (2022) *Robot Operating System 2: Design, architecture, and uses in the wild*, Science Robotics, Vol. 7.

Maruyama, Y., Kato, S. and Azumi, T. (2016) *Exploring the performance of ROS2, 2016 International Conference on Embedded Software (EMSOFT)*, pp. 1-10.

Ortiz, E., Andrés, B., Fraile, F., Poler, R. and Bas, Á. (2021) *Fleet management system for mobile robots in healthcare environments*, Journal of Industrial Engineering and Management, Vol. 14, pp. 55-71.

Quadrini, W., Negri, E. and Fumagalli, L. (2020) *Open Interfaces for Connecting Automated Guided Vehicles to a Fleet Management System*, Procedia Manufacturing, Vol. 42, pp. 406-413.

René de Koster, Tho Le-Duc and Kees Jan Roodbergen (2007) *Design and control of warehouse order picking: A literature review, European Journal of Operational Research*, Vol. 182(2), pp. 481-501.

Robot Middleware Framework (2022) *RMF - Programming multiple robots with ROS 2* [Online]. Available at: https://osrf.github.io/ros2multirobotbook/. (Accessed: 24 December 2023)

Robot Operating System. (2021) *ROS - Robot Operating System* [Online]. Available at: https://www.ros.org/. (Accessed: 24 December 2023)

Sauer, M., Dachsberger, A., Giglhuber, L. and Zalewski, L. (2022) *Decentralized Deadlock Prevention for Self-Organizing Industrial Mobile Robot Fleets, 2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, pp. 1-6.

Siegwart, R., Nourbakhsh, I. R. and Scaramuzza, D. (2011) *Introduction to Autonomous Mobile Robots*. MIT press.

Singhal, A., Kejriwal, N., Pallav, P., Choudhury, S., Sinha, R. and Kumar, S. (2017) *Managing a fleet of autonomous mobile robots (AMR) using cloud robotics platform, 2017 European Conference on Mobile Robots (ECMR)*, pp. 1-6.

Valner, R., Masnavi, H., Rybalskii, I., Põlluäär, R., Kõiv, E., Aabloo, A., Kruusamäe, K. and Singh, A. (2022) *Scalable and heterogeneous mobile robot fleet-based task automation in crowded hospital environments—a field test*, Frontiers in Robotics and AI, p. 9.

van Duijkeren, N., Palmieri, L., Lange, R. and Kleiner, A. (2023) *An Industrial Perspective on Multi-Agent Decision Making for Interoperable Robot Navigation following the VDA5050 Standard* [Online]. Available at: http://arxiv.org/abs/2311.14615. (Accessed: 24 December 2023)

Včelák, J., Ripka, P., Kubik, J., Platil, A. and Kašpar, P. (2005) *AMR navigation systems and methods of their calibration, Sensors and Actuators A:* Physical, Vol. 123, pp. 122-128.

VDA (German Association of the Automotive Industry) (2022) *VDA5050: Interface for Navigation of Autonomous Transport Systems*, VDA AG, Berlin, Germany. Available at: https://github.com/VDA5050/VDA5050/blob/main/VDA5050_EN.md (Accessed: 24 December 2023)

Wan, J., Tang, S., Yan, H., Li, D., Wang, S. and Vasilakos, A. V. (2016) *Cloud robotics: Current status and open issues*, IEEE Access, Vol. 4, pp. 2797-2807.

Wang, W. (2015) *AGV Magnetic Sensor Data Acquisition System, Applied Mechanics and Materials*, Vol. 713-715, pp. 1056-1060.

Wesselhöft, M., Hinckeldeyn, J. and Kreutzfeldt, J. (2022) *Controlling Fleets of Autonomous Mobile Robots with Reinforcement Learning: A Brief Survey*, Robotics, Vol. 11(5), pp. 85.

Yokotani, T. and Sasaki, Y. (2016) Comparison with HTTP and MQTT on required network resources for IoT, in *2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, pp. 1-6.

Zhang, W. and Ritter, N. (2001) *The real benefits of object-relational db-technology for object-oriented software development*, in British National Conference on Databases. Springer, pp. 89-104.