RO47007 Multidisciplinary Project

Ahold Delhaize

Team 20

Henk Jekel	5609593
Ernst Cancrinus	4915577
Kenny Brandon	4461428
Marijn de Rijk	4888871
Stan Vijverberg	4975553

Monday 19th June, 2023

Case: The Ahold Delhaize company case 2023

Contents

Abstract

This project report presents the outcomes of Project Team 20's efforts to enhance the capabilities of a mobile manipulator robot provided by AIRLab for a demo in collaboration with Ahold Delhaize. The primary goal of the project was to develop a realistic demo in which the robot autonomously collects items from a supermarket for online customers while simultaneously assisting in-store shoppers. Key areas of focus included navigation and mapping, perception, human-robot interaction, and motion control. By addressing these aspects, the team aimed to ensure efficient item picking while maintaining a positive user experience for customers in the supermarket environment.

First, we defined our project goal as the "implementation of the provided robot, Albert, in the existing traditional supermarkets to fulfill orders for customers." In the second chapter, a SWOT analysis was performed, and operational scenarios and nodes were defined. Then, in chapter three, we determined the extended scenarios, operational needs, functional requirements, and explained the functional hierarchy tree to provide insight into the required functions for the implemented solution. An activity diagram was also created to depict the flow of activities within the implemented system.

Chapter four discusses the FlexBE system, showcases the recorded robot behaviors, and demonstrates the capabilities of the entire system in simulation. Chapter five provides a summary of the results and real-world testing. Chapter six includes individual reflections from all team members regarding this project. Finally, the conclusion states the contents of the developed robot solution.

MEET THE TEAM

We called our team: 'SmartShelf Engineers', because we focus on finding smart solutions to problems in automating supermarket tasks.

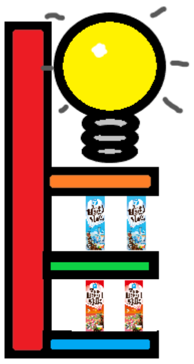
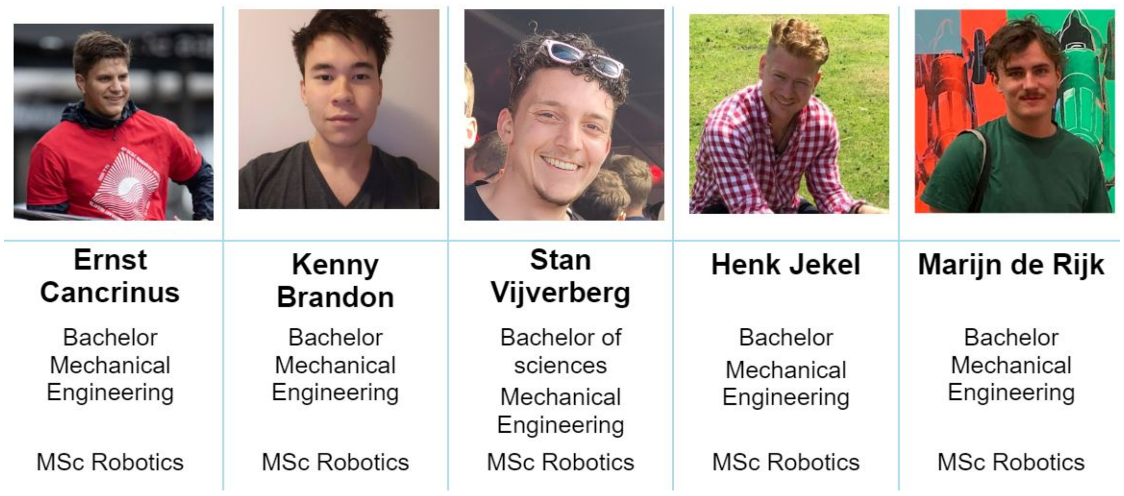


Figure 1.1: SmartShelf engineers logo

1.1 Team Introduction

The project team consists of five master students from the Robotics Department of the Technical University of Delft.



Ernst Cancrinus	Kenny Brandon	Stan Vijverberg	Henk Jekel	Marijn de Rijk
Bachelor Mechanical Engineering	Bachelor Mechanical Engineering	Bachelor of sciences Mechanical Engineering	Bachelor Mechanical Engineering	Bachelor Mechanical Engineering
MSc Robotics	MSc Robotics	MSc Robotics	MSc Robotics	MSc Robotics

Figure 1.2: Team Introduction

1.2 Team Role Division

In order to effectively manage our multidisciplinary project, each team member has been assigned a specific role. This allows everyone to focus on their respective tasks and utilize their expertise. However, there may be some overlap between roles, and students will assist each other as needed. Additionally, we have designated project team roles that rotate throughout the project, ensuring smooth workflow and overall accountability. This structure promotes collaboration and leverages everyone's skills for a successful outcome.

Ernst Cancrinus	Planning & Navigation	Project team member
Stan Vijverberg	Planning & Navigation	Report manager
Henk Jekel	Perception	Project manager
Marijn de Rijk	Human Robot Interaction	Simulation manager
Kenny Brandon	Motion Control	Project team member

Table 1.1: Team Specializations and Role Division

1.3 Important Agreements

The scope and objectives of the project, as well as the robot's intended capabilities and functions, will be clearly defined by the team. Each member of the team will be assigned specific responsibilities based on their role, with individuals responsible for planning and navigation, perception, human-robot interaction, and motion control.

Communication protocols will be established and agreed upon by the team, including the frequency and mode of communication, as well as the process for reporting progress and addressing any issues encountered during the project.

Furthermore, the team will establish code management procedures, outlining how the code will be stored and versioned, as well as who will have access to it. Additionally, the team will determine how changes made by multiple members will be merged together.

1.3.1 Communication protocols

The team has agreed to hold two meeting sessions per week, with individual work completed remotely between sessions. In the initial stages of the project, the team will prioritize working together in person to re-familiarize themselves with ROS and gain a thorough understanding of the simulation environment.

At the start of each session, a round of updates will be conducted to provide a status report from each team member regarding their progress since the previous meeting. Additionally, each session will conclude with the assignment of tasks to each team member, with the aim of completing these tasks prior to the next scheduled meeting.

1.3.2 Documentation

The team must reach a consensus on the approach for documenting the design, programming, and testing of the robot. This includes determining the appropriate tools to use and establishing the frequency at which the documentation will be updated.

It is recommended that, at the end of each sprint, the `README` files for all implemented nodes are updated. This is not only important for external viewers to understand the code, but also for the team members to be able to quickly reference the structure and design choices of a specific package or implementation when working on making all of the software function together. The team planned to use Ganttproject to track the tasks to be completed, but alternative workflows were also explored.

1.3.3 Version Control

For code version control, the team has decided to utilize a GitLab repository. To facilitate collaborative work, separate branches will be created for each role involved in the project, such as planning, perception, HRI, and motion control. These branches will then be merged or rebased into the main branch, enabling the integration of all written code at the end of a sprint using pull requests. The team has determined that merging will be utilized when it is necessary to maintain a clear record of changes and preserve the history of both branches. In contrast, rebasing will be used when maintaining a clean and linear history is paramount and it is important to avoid generating superfluous merge commits.

1.3.4 Conflict management

Conflicts could occur during every team project, especially when the pressure rises. If conflicts occur between team members during this project, one should always act professionally. Together we should be able to find the reason for the conflict, talk about it and find a solution. If we are not able to do this, for whatever reason, we will make an appointment with a MDP staff member.

PROJECT GOAL

In this chapter, the purpose and organization of the project are outlined. The team will collaborate with AIRLab and utilize their mobile manipulator Albert for the project while following the course manual [1]. AIRLab is a supermarket-like testing environment in Robohouse, Delft. Various robots can be tested here in a simulated supermarket environment. Albert is one of these robots, consisting of a differential drive base with a 7 DoF arm on top of it. The objective of the project is to demonstrate the robot's capabilities in a realistic supermarket environment. Specifically, the focus will be on order picking, wherein an online customer places an order and the robot independently collects the requested items from the store. The robot will also be able to interact with and service in-store customers, extending its abilities beyond its original use case.

The team faces the challenge of devising a demonstration that can be integrated with the existing AIRLab demo.

2.1 Problem Definition

Ahold Delhaize [2] faces stiff competition from supermarkets that provide home delivery services. In an effort to satisfy customers' desire for prompt delivery, establishing a central distribution centre in the Netherlands poses logistical challenges due to the high cost of delivery routes to remote areas of the country.

The objective of this project is to implement the provided robot, Albert, in the already existing traditional supermarkets to fulfill orders for customers. By packing and dispatching orders from local stores instead of distribution centres, the delivery process can be expedited and made more cost-effective. Achieving this goal requires the development of an automated order-picking system that does not necessitate store layout modifications and can collaborate seamlessly with store personnel and customers without disrupting their shopping experience.

2.2 Project Plan

The Gantt Chart and the Work Breakdown Structure are included in Appendix B. To create the Work Breakdown Structure, we used Miroboard, an online whiteboard platform that allows for the addition of various content types, including text, images, maps, and diagrams, and facilitates teamwork via the use of visual templates.

On the other hand, a Gantt chart is a type of bar chart used to visualize project schedules. It presents a list of tasks on the vertical axis and time intervals on the horizontal axis.

The relevant parts of our project are the deadlines of the four sprints. The project is divided into four parts with a final report for all of them. These reports show the work done in the past sprint(s). All the work for the deliverables is strictly planned in the days before the deadline, to make sure there is time to put everything nicely together.

2.3 Vision

The vision for the solution implementation is formed during a brainstorming session with the team. In this vision we see robots working in the supermarket during the night, and during the day, if it's not busy. These robots fulfill various tasks: shelf stocking, cleaning, order picking and assisting customers. In this project, we will focus on order picking and customer assistance. The autonomous collection of products in local supermarkets makes distribution centres unnecessary. Fewer buildings and transportation are needed, which makes this vision sustainable and future-proof. Intuitive customer interaction for assistance will increase the shopping experience.

2.3.1 SWOT Analysis

A SWOT analysis is conducted from the perspective of Albert Heijn (Ahold Delhaize). This tool helps to analyze what the company does best now and to devise a successful strategy for the future.

Strengths	Weaknesses
Numerous supermarkets over The Netherlands	Vulnerable public image
Overall good public image	Large personnel costs
Progressive image	Large transportation costs
Large revenues	
Developed grocery delivery	
Opportunities	**Threats**
Large budget for innovation	Customer safety
Upcoming robotic sector	Deterioration of public image
Making grocery delivery more efficient	Increasing maintenance costs

Table 2.1: SWOT Analysis

2.3.2 Simple operational scenario

This story describes a day in the life of Albert, the order-picking robot.

> At 8:45 am, I arrived at work and found Albert, our store's order-picking robot, already hard at work. The first five online orders of the day were neatly organized in baskets in a corner of the supermarket, ready for delivery within a 1km radius of the store. Albert had gathered and arranged the necessary food items before the first store worker arrived.
>
> Throughout the morning, Albert diligently worked on collecting customer orders that came in online. After completing each order, my colleagues and I were notified to go to Albert's base station in order to swap out the robot's product basket.
>
> At 1:03 pm, an elderly woman on a mobility scooter entered the store. She had difficulty standing and reaching some items. Thankfully, Albert was there to lend a helping hand. As she drove up to Albert, who was busy picking a box of cereal from the shelf, she spoke the phrase "Hello Albert," notifying the robot that she needed assistance. After placing the cereal in its basket, Albert responded, "Hello there, how can I help you?"
>
> "I am looking for some baking products," the lady asked. "Can you please specify which baking product you are looking for? We have sugar, baking soda, and other items in stock," the robot replied. "Baking soda, please," said the lady. "Do you require picking assistance?" Albert asked. Confirming that she did, the lady followed Albert to the baking section, where the robot picked the baking soda from the shelf and extended its arm to hand it to her. The rest of Albert's day consisted of picking more online orders and assisting in-store customers in finding and selecting their desired groceries.

This short story highlights the convenience and assistance that a grocery assistant robot like Albert can provide for both customers and employees.

2.4 Operational scenarios

The scenarios that the robot may encounter are defined in Table 2.2. These scenarios can be classified into two subclasses: deployment and operational. The deployment scenarios are relevant when the robot is first introduced to a specific supermarket, while the operational scenarios are encountered each time the robot is put to use. During our development, the main focus lies on the marked (*) operational scenarios.

Deployment phase	
S1 Robot placement	The robot is placed by the operator at the charging station.
S2 Load product database *	The product location database is updated when the robot is first deployed or when the store layout changes.
Operational phase	
S3 System start	After being shut down, the robot arm undergoes joint calibration and the base performs self-localization. The operator then places an empty basket on top of the robot frame in preparation for operation.
S4 Manage order *	The system receives an order request consisting of one or more product types and generates a fastest route ordered list for picking the products. If the order does not fit in one basket, it is split into multiple sub-orders.
S5 Navigate to product *	The mobile base navigates to the designated storage location of the next listed product, avoiding obstacles such as customers and store workers.
S6 Pick product *	The robot moves the gripper to the next listed product in the order and grasps it.
S7 Place product in basket *	The robot places the product in the basket.
S8 Replace basket on robot *	After completing a (sub)order, the robot returns to the charging station. When arrived at the charging station, the operator is requested to replace the basket.
S9 Safety stop	A physical safety stop is pressed by a human to shut stop all systems of the robot and resets after the safety stop.
S10 Shut down and charge	When there are no more orders for the robot, it returns to the charging station and requests the operator to connect the charger if the battery is low.
S11 Providing voice interacted assistance *	The customer asks the robot where certain products are and then shows the customer to the designated shelf.
S12 Presenting product to customer *	The robot presents the picked product from a shelf to a customer who asks the robot for assistance.

Table 2.2: Operational Scenarios

2.5 Nodes

To implement the solution, a set of ROS nodes that can function together in a system is required. The table below shows a list of functionalities which are currently to be necessary for the solution. Some software is initially given, so the functions which need editing or creation are marked (*). The initial repository contains a fully functioning simulation environment and a move to 2D location function, which has some obstacles avoidance in it. Arm movements are possible as well. The main challenge is to let everything work together, including our own functionalities. The functionalities that work closely together are taken together into one node. This will keep the code clear. This is the provisional node layout, it could change throughout the project.

Node #	Function #	Description of the function
1	1	System configuration and calibration.
1	2*	Configuration of products by updating the product location in the system's database or when the layout changes.
2	3	Starting up the subsystem by calibrating the joints and self-localization of the base.
3	4*	Managing the order by receiving an order list and calculating the fastest route for picking the products.
3	5	Navigating the base of the robot to the designated shelf with the targeted products.
4	6*	The robot detects the product on the shelf, while it moves its gripper to pick the product from the shelf.
4	7*	The robot moves its gripper with the product in it to the basket it is holding.
5	8*	The system requests a new basket when the order is completed or requires another basket to complete the order.
1	9	Safety stop for human-robot interaction to ensure the robot can operate safely by avoiding any potential hazards to humans.
1	10	Shutting the system down by going to its initial base and starting recharging.
5	11*	Responding to a customer's requests by voice interaction who asks for a product.
5	12*	Presenting the grabbed product to customers who requested assistance.

Table 2.3: Provisional list of Nodes

FUNCTIONAL ARCHITECTURE

In this chapter, the system engineering approach is employed to explicate the functional architecture of the solution within the Albert Heijn store. This is done by extending the operational scenarios based on which the deployment/operational needs are determined. The extended scenarios that are taken into account are the ones at the beginning of the operation, like starting the system, until the end of the operations like shutting down and charging the robot. From each of these operational scenarios, needs and functional requirements are formulated. Based on these requirements, functional specifications are established, which are then used to construct a functional hierarchy tree. This functional hierarchy tree gives a representation of how the order-picking system of the robot is broken down into parent-and children functions. Finally, the activity diagram gives a first model of the system, which is separated into three categories. These categories represent the types of technology in the activity diagram, which have their own functions that are implemented for the solution.

3.1 Extended scenarios

In this section, we expand upon the operational scenarios identified in chapter 2. Instead of only broadly defining scenarios in which the robot operates, decomposition is done into concrete, specific scenarios having pre- and post-conditions as well as triggering events. Extended operational requirements are necessary to ensure that a system or solution meets all the necessary criteria for successful operation in its intended environment. The increased specificity allows for easier identification of needs and functional requirements, which will be done in section 3.2
When the robot is first introduced to the store it needs to be placed at the charging station manually. This requirement is addressed by the extended operational scenario S1, which outlines the process for robot placement as presented in Table 3.1.

Table 3.1: Extended deployment Scenario S1 - Robot placement

Deployment Scenario	*S1 - Robot placement*
Preconditions	• The presence of a robot within the supermarket premises • Availability of employees for oversight and management
Triggering event	The management of the supermarket chooses to deploy the robot for operational purposes.
Description	The robot operator initiates the deployment phase by placing the robot at the designated charging station.
Postconditions	• The robot is successfully stationed at the charging station for operational readiness.

Subsequent to positioning the robot at the designated charging station, the robot must be equipped with information pertaining to the product location database. This knowledge is indispensable for the robot's ability to pick customer orders. As such, the extended operational scenario S2, detailed in Table 3.2, outlines the process for uploading the store map onto the robot.

Table 3.2: Extended deployment scenario S2 - Load product database

Deployment scenario	*S2 - Load product database*
Preconditions	• Availability of a map of the store and the product locations • The robot's computer must be turned on
Triggering event	Completion of S1 robot placement
Description	The operator uploads the product database of the supermarket into the robot's memory to facilitate its comprehension of the products and their locations.
Postconditions	• The robot possesses an understanding of the environmental layout of the supermarket.

To get out of the shutdown state, the robotic arm undergoes joint calibration, while the base initiates self-localization. Subsequently, the operator positions an empty basket atop the robot frame in preparation for operational deployment. This sequence of actions is defined by the extended operational scenario S3, which is outlined in Table 3.3.

Table 3.3: Extended operational Scenario S3 - System start

Operational Scenario	*S3 - System start*
Preconditions	• Database and store layout are up to date. • Battery level is sufficient. • The user interface is accessible to the human operator. • The robot is connected to the user interface.
Triggering event	A customer order comes in while the robot is not operational
Description	In order to begin system operations, the operator utilizes a touchscreen interface located on the robot to initiate system start. Following this, the robot performs a calibration procedure to establish the joint configuration of its robotic arm. After completing the calibration, the ROS navigation stack performs localization while traversing to a goal destination, and it is initialized based on a given pose. Then the operator adds an empty basket atop the robot.
Postconditions	• The joints are calibrated. • The base is self-localized. • All sensors are turned on. • There is an empty basket atop the robot frame.

Upon receipt of an order request comprising one or multiple product types, the system processes the request and generates a list of the fastest routes for picking the requested products. If the order quantity exceeds the basket's capacity, it is subdivided into multiple sub-orders. The extended operational scenario S4, as presented in Table 3.4, defines the process for managing the order fulfillment process.

Table 3.4: Extended operational Scenario S4 - Manage order

Operational Scenario	*S4 - Manage order*
Preconditions	• The system has started. • All products within the order are present in the store. • There is an order that has not been fully processed or completed on the list of customer orders. • The robot is equipped with a database regarding the location and corresponding April tag id of the products present in the supermarket. • The robot has either completed the previous order or needs to resume picking the previous order after a low battery occurrence.
Triggering event	The robot receives an order request or encounters a low battery situation during the picking process and needs to resume picking after recharging.
Description	The robot generates an optimized picking route for the next order in the order list, which may be split into multiple lists to accommodate large orders. The route is designed to ensure the fastest possible collection of all required products.
Postconditions	• The robot has a list of product locations in the store it needs to navigate to, in order to complete the order list. • The robot has the fastest route to pick the products of the (sub)order.

The mobile base proceeds to the designated storage location of the following product listed for retrieval, utilizing obstacle avoidance techniques to circumvent static obstacles such as aisles and dynamic obstacles such as customers and store workers. The extended operational scenario S5, outlined in Table 3.5, defines the process for navigating to the designated product storage location.

Table 3.5: Extended operational Scenario S5 - Navigate to product

Operational Scenario	*S5 - Navigate to product*
Preconditions	• The robot has started. • The system receives a location it needs to navigate to according to the product store database. • There are no obstacles clashing with the base of the robot.
Triggering event	The order has just been managed and then receives a product location from the order list.
Description	The robot follows the predetermined fastest route to navigate to the location of the (next) product. It employs obstacle avoidance mechanisms to circumvent both static and dynamic obstacles along its trajectory.
Postconditions	The robot is in front of the shelf of the product from the order list at the optimal distance for picking the product.

The robot manipulates the gripper to approach the subsequent product indicated in the order and securely grasps it. This process is defined by the extended operational scenario S6, presented in Table 3.6, which outlines the steps for picking up a product.

Table 3.6: Extended operational scenario S6 - Pick product

Operational Scenario	*S6 - Pick product*
Preconditions	• The robot has started. • The order has been managed • The robot's base stands in front of the shelf with the targeted product.
Triggering event	Robot has completed navigation to the next listed product.
Description	The robot's arm/gripper picks the next listed product from the shelf.
Postconditions	• The robot is holding a product. • The gripper of the robot makes only contact with the product and not the shelves.

Subsequently, the robot deposits the product into the basket. This process is defined by the extended operational scenario S7, as presented in Table 3.7, which outlines the steps for placing a product in the basket.

Table 3.7: Extended operational scenario S7 - Place product in basket

Operational Scenario	*S7 - Place product in basket*
Preconditions	• The robot holds a product in its gripper.
Triggering event	The robot successfully picks up a product from the shelf
Description	After the robot successfully picks up the product, it needs to place the product in the basket.
Postconditions	• The product has been placed in the basket. • The product can be checked from the managed order list.

Upon completion of a sub-order or when the battery level is low, the robot navigates back to the charging station. In the event of order completion, the operator is notified to replace the basket. The extended operational scenario S8, as outlined in Table 3.8, defines the process for the robot's return to the charging station.

Table 3.8: Extended operational scenario S8 - Replace basket on the robot

Operational Scenario	*S8 - Replace basket on the robot*
Preconditions	• (sub)order completed • Store worker is available.
Triggering event	The robot has completed picking the current (sub)order.
Description	The robot autonomously navigates back to the designated charging station and initiates a request for the replacement of the basket carrying the (sub)order with an empty one.
Postconditions	• The robot's basket is empty • The robot is at the charging station.

A physical safety button is activated by a human operator to halt all robotic systems. This process is defined by the extended operational scenario S9, as presented in Table 3.9, which outlines the steps for performing a safety stop. Once an operator has deemed the situation safe, the robot may resume operation.

Table 3.9: Extended operational scenario S9 - Safety stop

Operational Scenario	*S9 - Safety stop*
Preconditions	• The system has started. • An anomaly occurs.
Triggering event	A physical safety button is activated
Description	This safety scenario can be triggered by any person within range who perceives a risk posed by the robot. Upon activation, the robot will become compliant. The safety stop can only be deactivated by authorized personnel to reinitialize the robot.
Postconditions	• The robot becomes compliant • The robot remains stationary.

In the event that the robot returns to the charging station in the absence of any further orders and due to low battery, the operator is promptly notified to plug it into the charger. This process is defined by the extended operational scenario S10, as presented in Table 3.10, which outlines the steps for shutting down the robot and charging it.

Table 3.10: Extended operational scenario S10 - Shut down and charge

Operational Scenario	*S10 - Shut down and charge*
Preconditions	• No more (sub)orders are left in the order list. • The robot has started.
Triggering event	The battery of the robot gets below 10%.
Description	When the robot's battery is getting too low it navigates back to the charging station, shuts down and an operator plugs it into the power supply for charging.
Postconditions	The robot is turned off and is being charged at the charging station.

Customers can interact with the robot and request help in finding or picking products in the store. This process is defined by the extended operational scenario S11, as presented in Table 3.11, which outlines the steps for interacting with the customer through voice assistance.

Table 3.11: Extended operational scenario S11 - Providing voice interacted assistance

Operational Scenario	*S11 - Providing voice interacted assistance*
Preconditions	• The robot has started. • The robot is not busy with picking a product from the shelf.
Triggering event	The customer asks for assistance in finding a product by saying: ”Albert”.
Description	A customer asks where to find a certain product. The robot adds this task to the order list and then navigates to the store location of the requested product. Optionally, the robot offers to pick it up from the shelf.
Postconditions	• The robot identifies the requested product and navigates the customer to it. • The robot picked the product for the in-store customer.

When a robot picks a product for a customer, it will extend its arm and present the product to the customer, who can pull it off of the robot's vacuum gripper. This process is defined by the extended operational scenario S12, as presented in Table 3.12, which outlines the steps for presenting a product to the customer.

Table 3.12: Extended operational scenario S12 - Presenting product to customer

Operational Scenario	*S12 - Presenting product to customer*
Preconditions	• The robot has successfully picked a product with its gripper. • The robot has started.
Triggering event	The customer asks for assistance in picking a product.
Description	After picking a product it presents the product in the direction towards the requesting customer.
Postconditions	• The customer takes the presented product from the gripper.

3.2 Needs and functional requirements

In order to facilitate the implementation of the complex system, it is customary for systems engineers to define stakeholder needs and functional requirements. The functional requirements are derived from the stakeholder needs, which can be further categorized into deployment and operational needs. These needs are rooted in Albert Heijn's mission to enhance customer satisfaction, optimize order fulfillment processes, and improve efficiency within the stores.

The system requirements serve as the foundation for the functional hierarchy tree, as defined in Section 3.3. The functional requirements are designed to ensure a seamless flow of the robot through the entire process, starting from receiving an order and efficiently picking products to fulfill that order for customers, all while assisting in-store customers with their product selection.

Table 3.13: Deployment needs and functional requirements for S1 - System configuration and calibration

Deployment needs	*S1 - Robot placement*
DN_S1_1	The robot has a standard neutral state
DN_S1_2	The robot has a charging station which can be considered as an initial base point.
Functional requirements	*S1 - Robot placement*
FR_S1_1	The robot must have the ability to be shifted into a neutral state.
FR_S1_2	The robot must possess the capability to be manually navigated.

Table 3.14: Deployment needs and functional requirements for S2 - Load product database

Deployment needs	*S2 - Load product database*
DN_2_1	The system needs to link each product to a specific location in the store.
DN_2_2	The product database contains the name of the product, its dimensions and coordinates.
Functional requirements	*S2 - Load product database*
FR_S2_1	The system shall be able to read which specific shelf belongs to the product's location.
FR_S2_2	The system shall be able to read databases which contain the same type of data, but with other products.

Table 3.15: Operational needs and functional requirements for S3 - System start

Operational needs	*S3 - System start*
ON_3_1	A computer with access to all actuators is needed, to let this computer start all components.
ON_3_2	Optionally move the base a bit(± 1 meter) to localize itself in its internal map.
ON_3_3	The lidar sensor is not blocked by an object which is not part of the map.
ON_3_4	The touchscreen is accessible.
ON_3_5	The robot shall perform the requested movements accurately.
Functional requirements	*S3 - System start*
FR_S3_1	The robot should possess the capability to handle all startup operations by simply pressing the startup button on the user interface.
FR_S3_2	The robot shall be able to perform self-localization to determine the precise robot base location on the map upon startup.
FR_S3_3	The robot shall be able to perform a calibration procedure of its joint angle encoders to establish the configuration of its robotic arm upon startup.
FR_S3_4	In instances where the robot's battery level is below full capacity, the robot shall detect this and cancel the startup procedure accordingly.
FR_S3_5	The robot shall request the operator to place an empty basket atop the robot upon startup.

Table 3.16: Operational needs and functional requirements for S4 - Manage order

Operational needs	*S4 - Manage order*
ON_4_1	The robot is able to receive multiple order requests remotely.
ON_4_2	The robot is able to find the fastest possible route through the store to pick all products in the order request.
ON_4_3	If an order request contains a number of products above a set threshold (for example 5 products), The robot is able to split the order request into multiple separate order requests and make several separate routes, between which it returns to the base station to replace its crate with products.
ON_4_4	The robot shall be able to receive messages from the order system by subscribing to a topic that receives order messages.
Functional requirements	*S4 - Manage order*
FR_S4_1	The system shall process the order request and decompose it into pick and place operations for the products in the order.
FR_S4_2	The robot shall be able to split the order into suborders if the order does not fit in one basket.
FR_S4_3	The robot shall be able to perform the fastest route path planning for the locations of the products in the (sub)order.
FR_S4_4	The robot shall be able to reorder the products in the (sub)order according to the path planning algorithm.
FR_S4_5	The robot shall be able to receive messages from the order system by subscribing to a topic that receives order messages.
FR_S4_6	The robot shall be able to provide a 6-DOF orientation of the robot base to navigate towards such that the gripper is capable of picking the product.

Table 3.17: Operational needs and functional requirements for S5 - Navigate to product

Operational needs	*S5 - Navigate to product*
ON_5_1	The robot is able to avoid moving and stationary obstacles and drive around them to reach the goal of its route.
ON_5_2	The robot is able to use the route which has been created in S4 to navigate to products.
ON_5_3	The robot is able to stop its movement in front of the shelf of the product it is supposed to pick at a set distance (for example 50 cm) which is optimal for picking products.
ON_5_4	The robot shall be able to determine the next location of the robot base.
Functional requirements	*S5 - Navigate to product*
FR_S5_1	The robot shall be able to avoid dynamic obstacles.
FR_S5_2	The robot shall be able to avoid static obstacles.
FR_S5_3	When a new location for the robot base is defined, the robot shall plan the fastest route based on the static map provided.
FR_S5_4	When the fastest route based on the static map has been computed the robot shall move towards the defined robot base location avoiding static and dynamic obstacles along the path in real-time.

Table 3.18: Operational needs and functional requirements for S6 - Pick product

Operational needs	*S6 - Pick product*
ON_6_1	The robot shall be able to pick products of a cylinder or box shape, with a maximum dimension of 20 cm, offering a top flat surface available for suction and a maximum weight of 1 kg.
ON_6_2	The robot shall pick the products from shelves that have multiple rows stacked on top of each other, without letting the gripper clash with the shelves.
ON_6_3	The robot shall pick the products while its base is orientated with the front facing the product it needs to pick from the shelf.
ON_6_4	After standing in front of the right shelves, the robot shall identify the product that is in front of it, with its stereo camera.
ON_6_5	After identifying the product on the shelf, the robot shall move its gripper towards the product according to the depth of the stereo camera.
ON_6_6	The robot's gripper stops moving towards the product upon making contact. Then the product needs to stick to the gripper by the suction cup.
ON_6_7	The robot needs to be able to view the product that needs to be picked using the camera.
Functional requirements	*S6 - Pick product*
FR_S6_1	The robot shall grasp a product from the shelf of the type requested.
FR_S6_2	The robot shall be able to send a message to the stock system that the number of items in the stock of the picked type has decreased by one.
FR_S6_3	The robot shall not make any contact with other products or shelves during the picking.
FR_S6_4	The robot shall point the camera at the gripper at the product such that the product is visible on the image plane.

Table 3.19: Operational needs and functional requirements for S7 - Place product in basket

Operational needs	*S7 - Place product in basket*
ON_7_1	The robot will place the picked product within the boundaries of the basket's dimensions, without dropping the product outside the basket.
ON_7_2	During the placement of the product in the basket the suction cup at the gripper stays active until the product is at the correct orientation inside the basket.
ON_7_3	The robot gripper will move the picked product inside the basket without any collision with the outsides of the basket. So the product only has contact with the gripper.
Functional requirements	*S7 - Place product in basket*
FR_S7_1	The robot shall be able to place the picked product inside the basket.
FR_S7_2	The robot shall be able to check the specific product from the order list.
FR_S7_3	The robot shall be able to place the product without any collision or dropping it.

Table 3.20: Operational needs and functional requirements for S8 - Replace basket on robot

Operational needs	*S8 - Replace basket on robot*
ON_8_1	The robot shall be given an initial, unobstructed position at which the charging station is located.
ON_8_2	The robot shall be capable of navigating to the charging station avoiding static and dynamic obstacles.
Functional requirements	*S8 - Replace basket on robot*
FR_S8_1	When moving to the charging station, the robot shall be able to plan a collision-free trajectory avoiding both static and dynamic obstacles in real-time.
FR_S8_2	Before initiating the navigation towards the charging station, the robot shall move its arm to the move configuration, minimizing its volume in 3d space.
FR_S8_3	When arriving at the charging station, the robot shall be waiting for operators to replace the basket that contains a full basket or finished (sub)order.
FR_S8_4	The robot shall provide the success state of the action.

Table 3.21: Operational needs and functional requirements for S9 - Safety stop

Operational needs	*S9 - Safety stop*
ON_9_1	The safety stop button is reachable for a standing adult from every side of the robot
ON_9_2	Sensors keep active after safety stop activation, to keep location data.
ON_9_3	Any human close to the robot must be able the safely escape from the robot.
Functional requirements	*S9 - Safety stop*
FR_S9_1	The robot shall become compliant when the safety stop is initiated.
FR_S9_2	The robot arm actuators will not move until the safety stop is off.
FR_S9_3	The robot will be able to start up and reinitialize again when the safety stop is off.

Table 3.22: Operational needs and functional requirements for S10 - Shut down and charge

Operational needs	*S10 - Shut down and charge*
ON_10_1	Moving to the charging station requires static obstacle avoidance.
ON_10_2	Moving to the charging station requires dynamic obstacle avoidance.
ON_10_3	The robot shall know its battery level at all times.
ON_10_4	The robot needs to be able to be charged at the charging station.
Functional requirements	*S10 - Shut down and charge*
FR_S10_1	The robot shall have a shutdown functionality.
FR_S10_2	The robot shall have a charging functionality.
FR_S10_3	The robot shall be able to avoid static obstacles.
FR_S10_4	The robot shall be able to avoid dynamic obstacles.
FR_S10_5	The robot shall be able to be plugged into the charger by an operator.

Table 3.23: Operational needs and functional requirements for S11 - Safety stop

Operational needs	*S11 - Providing voice interacted assistance*
ON_11_1	The system of the robot must be able to use a voice interactive API.
ON_11_2	The system must be able to respond when the customer says: "Albert" .
ON_11_3	The system shall be able to respond back to the customers as confirmation of their responses.
Functional requirements	*S11 - Providing voice interacted assistance*
FR_S11_1	The system shall be able to receive the customer's voice as input.
FR_S11_2	The system shall be able to process and interpret the customer's request as a (sub)order for the order list.
FR_S11_3	The robot shall be able to extract individual products from the customer's list and generates a plan for collecting them.

Table 3.24: Operational needs and functional requirements for S12 - Presenting product to customer

Operational needs	*S12 - Presenting product to customer*
ON_12_1	The robot will present the picked product to the customer's direction without dropping it.
ON_12_2	During presenting the product to the customer the suction cup at the gripper stays active until the product is taken by the requesting customer.
ON_12_3	The robot gripper will move the picked product towards the customer without any collisions. So the product only has contact with the gripper.
Functional requirements	*S12 - Presenting product to customer*
FR_S12_1	The robot shall be able to present the picked product towards the requesting customer.
FR_S12_2	The robot shall be able to present the product without any collisions or dropping it.
FR_S12_3	The robot shall be able to check the customer's requested task from the order list.

3.3 Functional hierarchy tree

In order to facilitate the implementation of the needs and functional requirements specified in section 3.2, specific functions were defined to enable the autonomous collection of items from the supermarket for online customers while simultaneously assisting in-store shoppers.

The functional hierarchy tree is utilized as a flow diagram to depict the various functions involved in the picking robot system. It clearly identifies four main functions: base navigation, robot arm control, voice interaction provision, and product database management.

For each main function and sub-function within the functional hierarchy tree, a comprehensive functional description is presented in a tabular format. This description encompasses the following elements: label, description, inputs, outputs, parent function, and children functions. The tabular representation significantly enhances the comprehension of the functions' roles and relationships within the system.

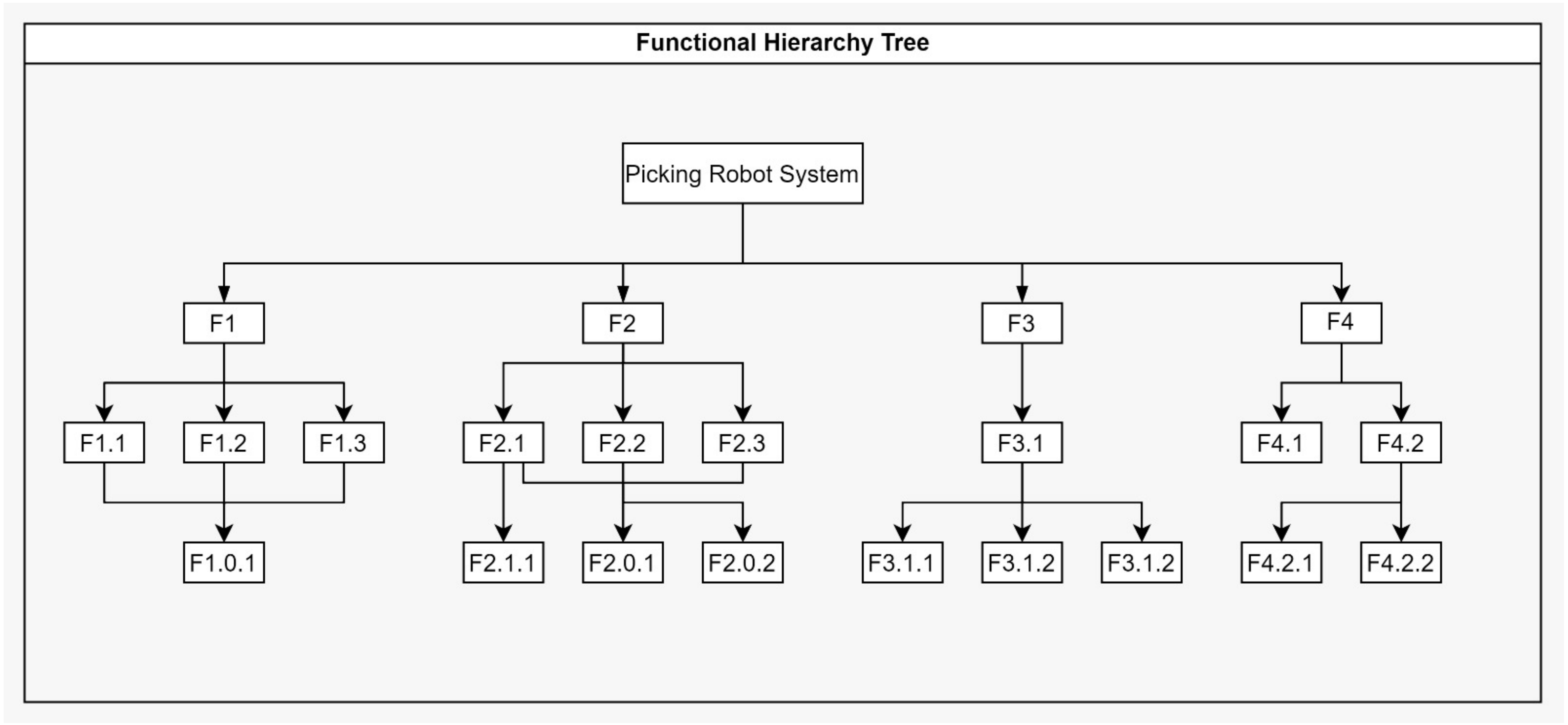


Figure 3.1: Functional Hierarchy Tree

Table 3.25: Function description F1

Function name:	Navigating robot base.
Label:	F1
Description:	Navigating the base of the robot around the store in order to get to shelved products and the base station.
Inputs:	Destination coordinates in x, y and yaw.
Outputs:	Path planning and motor control command to navigate.
Parent function:	
Children functions:	F1.1 F1.2 F1.3

Table 3.26: Function description F1.1

Function name:	Navigate to product.
Label:	F1.1
Description:	Navigating the base of the robot to the location of the corresponding products on the order list.
Inputs:	Destination coordinates of the shelved product in x, y and yaw.
Outputs:	Path planning and motor control commands.
Parent function:	F1
Children functions:	F1.0.1

Table 3.27: Function description F1.2

Function name:	Replace basket on robot.
Label:	F1.2
Description:	When the robot finished a (sub)order or has a full crate it needs to be replaced by an empty one, by navigating to the charging station.
Inputs:	Destination coordinates of the charging station in x, y and yaw.
Outputs:	Path planning and motor control commands.
Parent function:	F1
Children functions:	F1.0.1

Table 3.28: Function description F1.3

Function name:	Recharge battery
Label:	F1.3
Description:	When the battery is below 10% the robot needs to navigate back to the initial location which is the charging station of the robot.
Inputs:	Destination coordinates of the charging station in x, y and yaw.
Outputs:	Path planning and motor control commands.
Parent function:	F1
Children functions:	F1.0.1

Table 3.29: Function description F1.0.1

Function name:	Avoid collision.
Label:	F1.0.1
Description:	Avoid possible collision with static and dynamic obstacles, like customers and refill carts, while it is navigating to specific locations.
Inputs:	Coordinates of the point cloud from the lidar and radar sensors.
Outputs:	Path planning and motor control commands.
Parent function:	F1 F2 F3
Children functions:	

Table 3.30: Function description F2

Function name:	Controlling robot arm.
Label:	F2
Description:	This subsystem enables the robot's arm and suction gripper to interact with products on the shelves and pick them for customers.
Inputs:	Object perception data, customer order list and arm control commands.
Outputs:	Arm movement commands and gripper control commands.
Parent function:	
Children functions:	F2.1 F2.2 F2.3

Table 3.31: Function description F2.1

Function name:	Pick product
Label:	F2.1
Description:	When the base of the robot is facing towards the shelves, it uses the arm to navigate the gripper from the initial state to a state where it makes contact with the product.
Inputs:	Detected product coordinates on the shelf.
Outputs:	Arm position and arm movement command.
Parent function:	F2
Children functions:	F2.1.1 F2.0.1 F2.0.2

Table 3.32: Function description F2.2

Function name:	Place in basket
Label:	F2.2
Description:	After picking the product, the robot arm needs to move the gripper in a certain configuration to the top of the basket and drop it.
Inputs:	Arm dropoff position values, basket collision box.,
Outputs:	Arm movement commands.
Parent function:	F2
Children functions:	F2.0.1 F2.0.2

Table 3.33: Function description F2.3

Function name:	Present product to customer.
Label:	F2.3
Description:	After picking a product it presents the product in the direction towards the requesting customer.
Inputs:	Sensor data from camera and lidar.
Outputs:	Arm movement commands
Parent function:	F2
Children functions:	F2.0.1 F2.0.2

Table 3.34: Function description F2.1.1

Function name:	Detecting products on shelf.
Label:	F2.1.1
Description:	While the picking action is initiated the arm uses the stereo camera to detect the product on the shelf and also estimate the depth to pick it.
Inputs:	RGB-D camera feed.
Outputs:	The coordinate frames of detected products.
Parent function:	F2.1
Children functions:	

Table 3.35: Function description F2.0.1

Function name:	Avoiding collision between arm/gripped product and obstacles
Label:	F2.0.1
Description:	While the arm performs the pick and place movements it must take into account that the gripper and/or picked product do not collide with the shelves or outsides of the basket.
Inputs:	Coordinates of the collision objects.
Outputs:	Arm movement commands.
Parent function:	F2.1 F2.2 F2.3
Children functions:	

Table 3.36: Function description F2.0.2

Function name:	Controlling the suction gripper
Label:	F2.0.2
Description:	While the gripper makes contact with the product it needs to stay attached by activating the suction in the gripper. And when the arm moves the product to the basket, the suction on the gripper deactivates.
Inputs:	Gripper control commands.
Outputs:	Gripper state updates.
Parent function:	F2.1 F2.2 F2.3
Children functions:	

Table 3.37: Function description F3

Function name:	Providing voice interaction assistance.
Label:	F3
Description:	The system of the robot facilitates communication between the robot and the customer through simple voice interaction.
Inputs:	Speech input.
Outputs:	Robot movement commands and customer interaction responses.
Parent function:	
Children functions:	F3.1

Table 3.38: Function description F3.1

Function name:	Listening to customer's request.
Label:	F3.1
Description:	By listening to the words the customer uses when requesting for a certain product it recognizes it from its product database.
Inputs:	Speech input (requested product).
Outputs:	Location of the requested product.
Parent function:	F3
Children functions:	F3.1.1 F3.1.2 F3.1.3

Table 3.39: Function description F3.1.1

Function name:	Convert spoken command to text.
Label:	F3.1.1
Description:	When the robot gets a vocal request from a customer it converts spoken commands from the customer into text.
Inputs:	Voice input.
Outputs:	Recognized text.
Parent function:	F3
Children functions:	

Table 3.40: Function description F3.1.2

Function name:	Process and interpret customer's request.
Label:	F3.1.2
Description:	The system of the robot extracts individual products from the customer's request and generates a plan for navigating and optionally collecting them.
Inputs:	Interpreted voice commands.
Outputs:	Extracted products from the product database.
Parent function:	F3
Children functions:	

Table 3.41: Function description F3.1.3

Function name:	Respond to customer's request.
Label:	F3.1.3
Description:	While the system of the robot gets a speech input from the customer it needs to give a response back as confirmation that it heard the customer.
Inputs:	Extracted individual product.
Outputs:	Robot movement commands and vocal response to the customer.
Parent function:	F3
Children functions:	

Table 3.42: Function description F4

Function name:	Manage product system database.
Label:	F4
Description:	The system needs to operate with a database of products which correspond to the received orders it needs to pick for the customers.
Inputs:	Product location data and store layout.
Outputs:	Assembled database.
Parent function:	
Children functions:	F4.1 F4.2

Table 3.43: Function description F4.1

Function name:	Load product database.
Label:	F4.1
Description:	Whenever new databases or changes about products their name, picking location or corresponding April tag are being made, it needs to be updated in the database.
Inputs:	Changes for the product database.
Outputs:	Updated database.
Parent function:	F4
Children functions:	

Table 3.44: Function description F4.2

Function name:	Manage order.
Label:	F4.2
Description:	The system of the robot receives an order from the customer and divides it into (sub)orders if it requires more than one crate.
Inputs:	Received order list.
Outputs:	List of (sub)orders.
Parent function:	F4
Children functions:	F4.2.1 F4.2.2

Table 3.45: Function description F4.2.1

Function name:	Plan picking sequence.
Label:	F4.2.1
Description:	Plan optimal picking sequence by chronologically sorting the picking locations from the order list, which is derived from the product database.
Inputs:	List of (sub)orders.
Outputs:	List of picking locations.
Parent function:	F4.2
Children functions:	

Table 3.46: Function description F4.2.2

Function name:	Update order list.
Label:	F4.2.2
Description:	When a product from the order list is placed in the basket it needs to update it, so it can be confirmed when the order list is complete.
Inputs:	Completing function F2.2.
Outputs:	Updated order list.
Parent function:	F4.2
Children functions:	

3.4 Activity diagram

As discussed in section 3.3, the functional hierarchy tree facilitates the implementation of the autonomous collection of items from the supermarket for online customers while simultaneously assisting in-store shoppers. To ensure the consistency of interconnections between functions, an activity diagram is employed to provide an overview. The activity diagram also contains functions derived from the unmarked operational scenarios in 2.2. The functions derived from the unmarked operational scenarios are still important to visualize the total operational flow of the robot's activity diagram even when we do not mainly focus on developing these functions.

The activity diagram is categorized into three sections: database, robot, and system. Each category encompasses specific tasks performed within it. For instance, functions related to order management and determining when the robot should return to the base are visible within the system category. On the other hand, functions such as scanning products on shelves, picking items, and placing them are performed physically by the robot.

All these functions converge to enable the robot's capability to fulfill orders from online customers while also providing assistance to in-store customers.

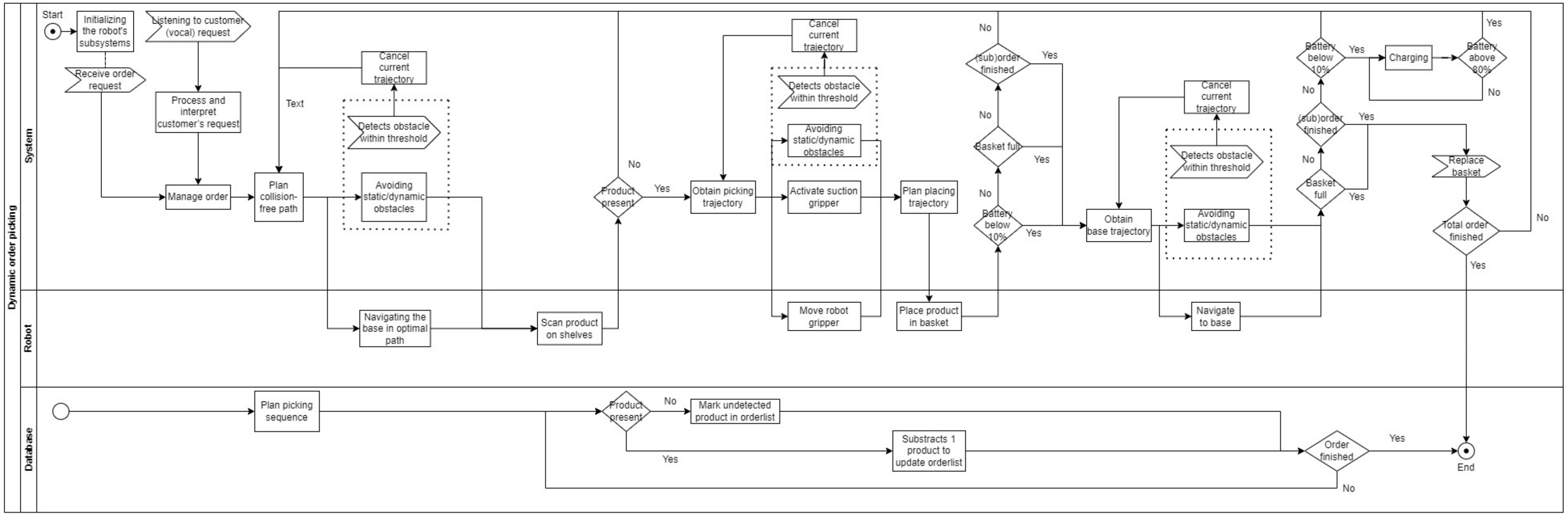


Figure 3.2: Activity diagram

FULL FUNCTIONAL DESCRIPTION OF THE ROBOT SOFTWARE

This chapter aims to enhance the understanding of the functional architecture discussed in the previous section by providing a comprehensive overview of the software components and their collective impact on the order-picking robot's overall behavior.

One difference with the previous section to mention is that battery management is not taken into account anymore, because this is not simulated and is not accessible in real-world Albert.

Subsection 4.1 utilizes an RQT graph to visually depict the implemented nodes and their interrelations. Furthermore, it details the specific contributions made by individual team members to these components.

To present an overview of the robot's behavior within the solution, subsection 4.2 shows the state machine implemented in FlexBE.

Moreover, subsection 4.3 focuses on the analysis of the recorded behaviors. The testing phase involved the use of ROS in a simulated environment, employing Rviz and Gazebo. These simulations served as a preliminary assessment to evaluate the functionality of the isolated behaviors before their implementation in the physical world.

During these simulations, the primary objectives were to evaluate the robot's ability to successfully pick up products and place them in the basket, as well as to navigate accurately to the designated shelves and base station.

4.1 Functional graph

The `rqt_graph` utility in ROS is used to show all nodes, topics, actions, and services, which constitute the main elements of the software architecture. Figure 4.1 shows a filtered and annotated `rqt_graph` of the robot behavior. Filtering has been applied to include the most critical nodes and topics present while making sure that the figure remains readable and fits on a page. See table 4.1 for information about the colour coding used.

4.1.1 `actionlib` and `FlexBE`

The core functionality of our solution depends on two ROS libraries: `actionlib` and `FlexBE`. The `actionlib` library allows for the definition of "action servers", and ros nodes which can be called by an "action client" in order to execute a specific behavior while providing feedback and result information. The `FlexBE` library allows us to create an FSM (Finite State Machine) based representation of the robot behavior. The states that the robot can be in must be defined. In our case, many defined states act as action clients, which call a specific action server on execution. States are connected together to form a final robot behavior. One of the great powers of FlexBE is that the connection between states can be updated live during the run time!

When looking at the `rqt_graph` in figure 4.1, the before mentioned architecture is clearly visible: we can see that the `/behavior` node, which encapsulates the `FlexBE` behavior, sends goals to each of the action server nodes via their respective topics, and receives results back.

The next section will dive deeper into our `FlexBE` behavior.

Table 4.1: Each of the colors in figure 4.1 corresponds to a discipline. The group members that implemented each functionality are listed.

	Type	Contributors
🟥	Order Management (Planning and Navigation)	Stan Vijverberg, Marijn de Rijk
🟦	Customer Voice Interaction (HRI)	Marijn de Rijk
🟩	Robot Skills (Motion Control)	Marijn de Rijk, Kenny Brandon
🟨	Customer Localization (Perception)	Henk Jekel
🟪	FlexBE (High-Level planning)	Ernst Cancrinus, Marijn de Rijk
⬜	Provided Functionality	AIRLab

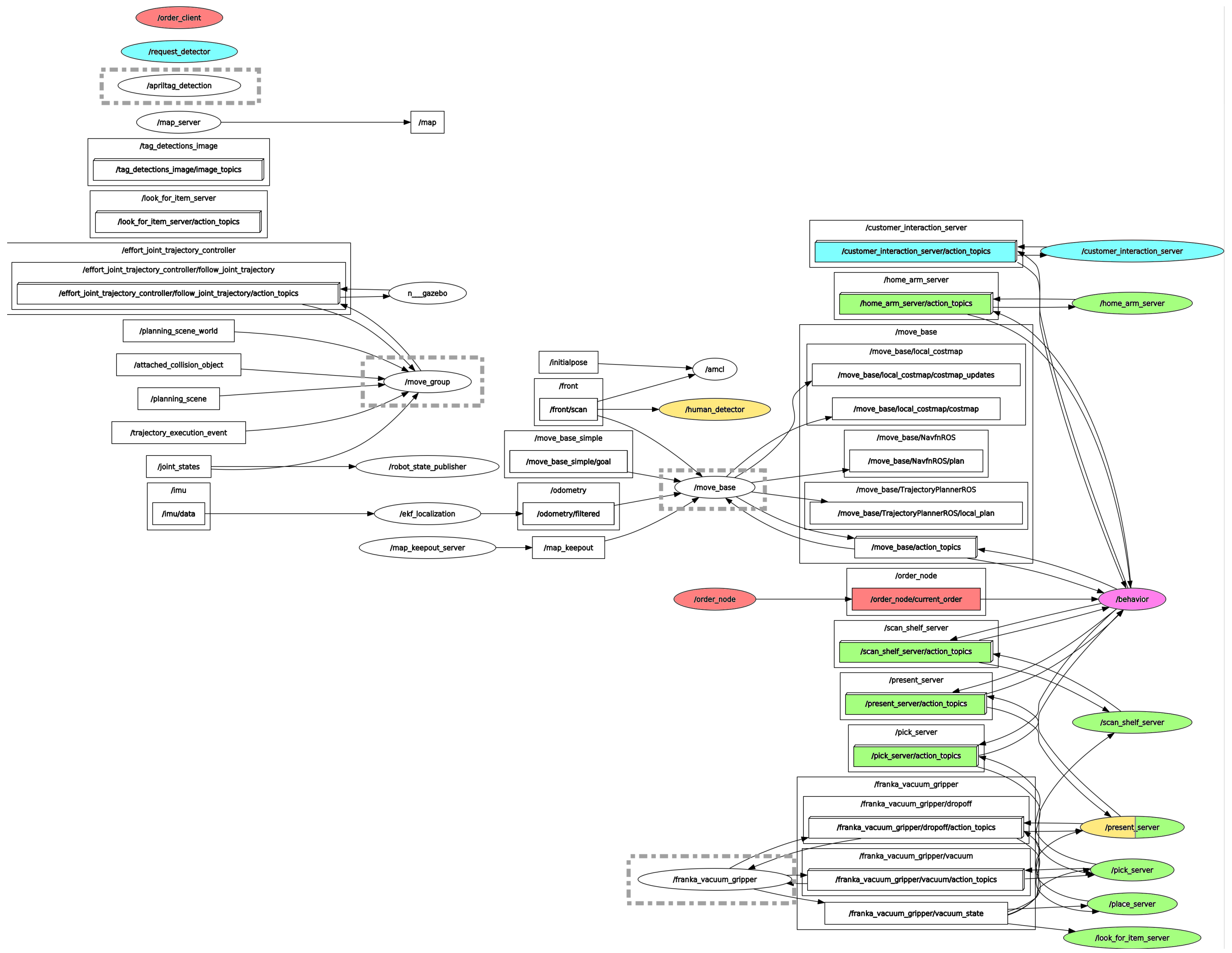

/order_client
/request_detector
/apriltag_detection
/map_server
/map
/tag_detections_image
/tag_detections_image/image_topics
/look_for_item_server
/look_for_item_server/action_topics
/effort_joint_trajectory_controller
/effort_joint_trajectory_controller/follow_joint_trajectory
/effort_joint_trajectory_controller/follow_joint_trajectory/action_topics
n__gazebo
/planning_scene_world
/attached_collision_object
/move_group
/planning_scene
/trajectory_execution_event
/joint_states
/robot_state_publisher
/imu
/imu/data
/ekf_localization
/map_keepout_server
/map_keepout
/initialpose
/amcl
/front
/front/scan
/human_detector
/move_base_simple
/move_base_simple/goal
/odometry
/odometry/filtered
/move_base
/customer_interaction_server
/customer_interaction_server/action_topics
/customer_interaction_server
/home_arm_server
/home_arm_server/action_topics
/home_arm_server
/move_base
/move_base/local_costmap
/move_base/local_costmap/costmap_updates
/move_base/local_costmap/costmap
/move_base/NavfnROS
/move_base/NavfnROS/plan
/move_base/TrajectoryPlannerROS
/move_base/TrajectoryPlannerROS/local_plan
/move_base/action_topics
/order_node
/order_node
/order_node/current_order
/behavior
/scan_shelf_server
/scan_shelf_server/action_topics
/present_server
/present_server/action_topics
/scan_shelf_server
/pick_server
/pick_server/action_topics
/franka_vacuum_gripper
/franka_vacuum_gripper/dropoff
/franka_vacuum_gripper/dropoff/action_topics
/present_server
/franka_vacuum_gripper/vacuum
/franka_vacuum_gripper
/franka_vacuum_gripper/vacuum/action_topics
/pick_server
/place_server
/franka_vacuum_gripper/vacuum_state
/look_for_item_server

Figure 4.1: simplified `rqt_graph` of the robot, colour coded to showcase the functionality that has been added

4.2 Implemented robot behavior

Figure 4.7 shows our `FlexBE` robot behavior which is used to model the full behavior of the robot. Below follows an explanation of the most important states. The FlexBE states were almost all created by us and the code follows a specific structure in order for the FlexBE system to be able to use them. A benefit of the FlexBE behavior engine is that existing states can be easily integrated into different behavior models and that even during execution changes can be made to the state machine in order to try different behavior variants. During the execution of the FlexBE behavior model, the FlexBE application will show which state is currently active and display outcomes of states and errors that might be encountered. This was very useful when designing the behavior.
When executing and designing a behavior, the FlexBE app is used. This app provides visual information about the behavior during execution which is very useful. An overview of the behavior running in this app during different simulation stages is also shown.

`order_master`

This state receives the current order, which is published by the `order_node`. If there is no order available, the state simply waits for one. When an order is received, the state makes sure that the following states receive the correct information in order to navigate to and pick the needed product.

`customer_interaction`

Just like Siri or Alexa, the Albert robot is always listening for a "trigger word" (in our case the word is "Albert"). When the trigger word is heard, the robot responds with "I will be with you in a minute". Once the word has been said, the robot will transition to the `customer_interaction` state in order to try and fulfill a customer's request. Due to the sequential nature of FSMs, some of the states in the behavior include a transition to `customer_interaction` state, which will be activated when the trigger word has been said.

Note that not all states can transition to the `customer_interaction` state: if the robot is busy picking a product, for instance, it is important to complete this action first before helping a customer.

`complete_order`

The `complete_order` state is activated when the robot has successfully completed the task that was received from the `order_master`. Note how multiple transitions to the `complete_order` state are possible: this is due to the possibility of a customer asking for help finding a product but not picking a product. In this specific situation, the robot's task would already be completed after having moved the base to the product location.

The full behavior

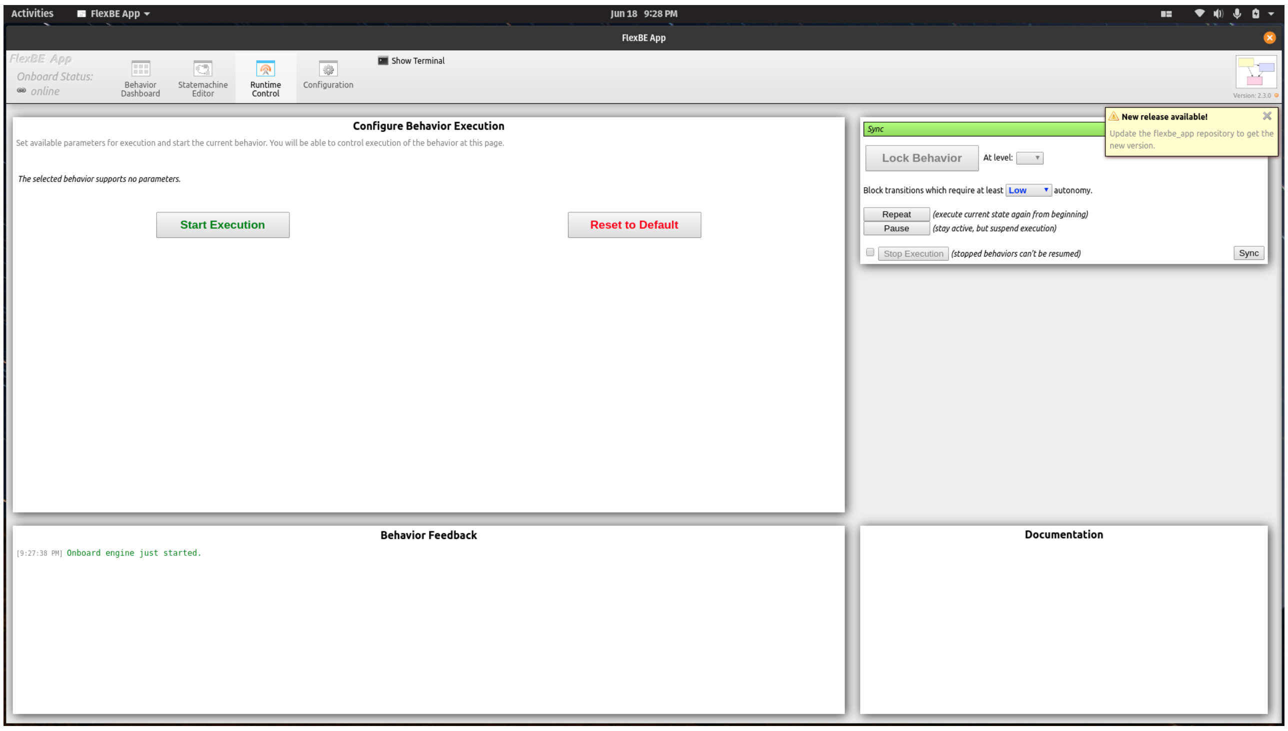
An example of the robot in action executing the designed `FlexBE` behavior would be:

The robot starts at its initial base station in its home position, where it waits for an incoming order. The order handling system creates an order list based on the orders it receives through customer interaction or through the user interface. The first order in the order list is then sent to the Flexbe state called `order_master`. The `order_master` state then sends the necessary information for completing the action to the other states, after which the behavior would then continue to the next state, called `move_base`. In this state, the robot moves to the front of the shelf where the product is located. If the location is reached, the behavior will move to the next state called `scan_shelf`. The panda arm will scan for April tags of the corresponding products. The robot can be interrupted at almost any time by an in-store customer. A customer could ask where a certain product is and if the robot could pick it up for him. These kinds of requests have priority over the regular order. Back to the scanned April tags, the robot moves its arm to the correct product and picks them. Now the product can be handed to the in-store customer or placed in the basket on the robot. Now it goes on to the next product in its list by going back to the order master state, after which it goes through all the states again. This process repeats until the entire order is complete. The robot will move to its initial base if the (sub)order is complete. At the base station, the basket will be replaced with an empty one.

Overview of the behavior in the FlexBE app during different stages of the simulation

In this section, an overview will be given of what the FlexBE app displays about the behavior during simulation. These images will show step-by-step what executing a behavior in the FlexBE app looks like.

figure 4.2 shows the runtime control screen in the FlexBE app. In this screen, the behavior execution is started with the press of a button.



Configure Behavior Execution

Set available parameters for execution and start the current behavior. You will be able to control execution of the behavior at this page.

The selected behavior supports no parameters.

New release available!

Update the flexbe_app repository to get the new version.

Block transitions which require at least Low autonomy.

(execute current state again from beginning)

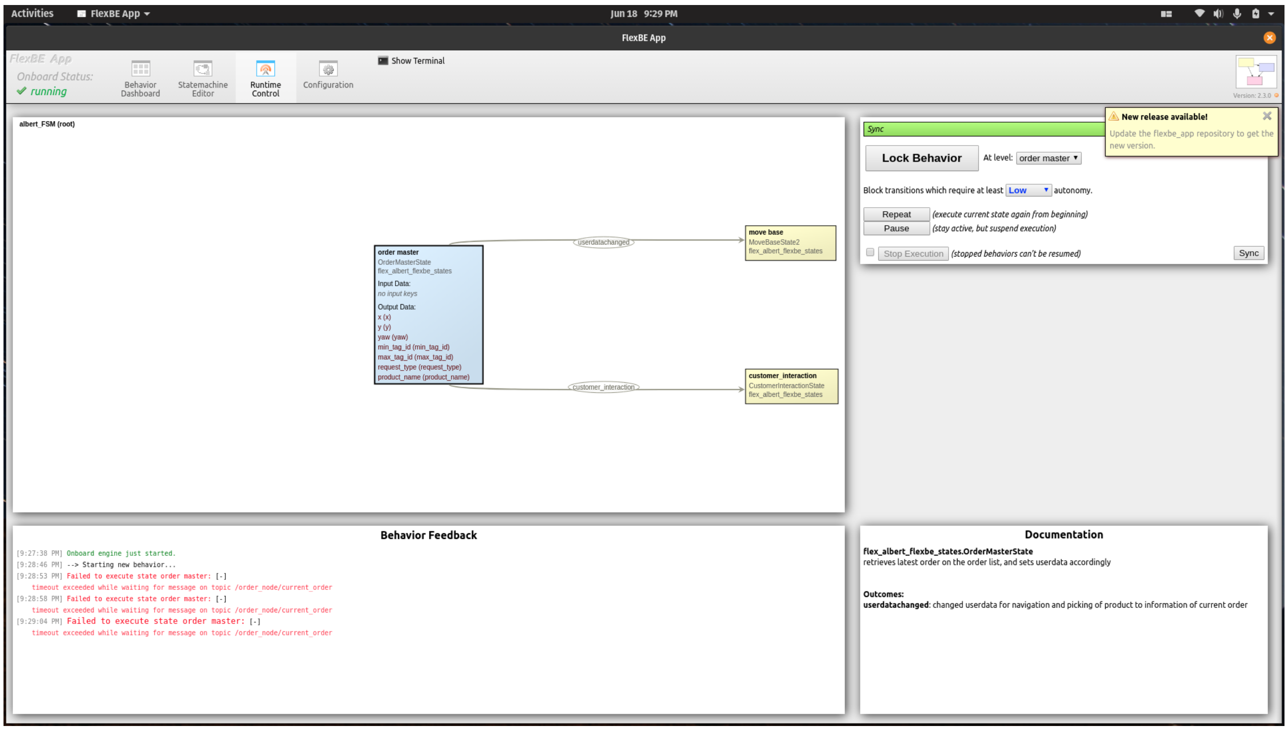
(stay active, but suspend execution)

(stopped behaviors can't be resumed)

```
[9:27:38 PM] Onboard engine just started.
```

Figure 4.2: overview of the runtime control screen of the flexbe app

After pressing the 'start execution' button, the first state the behavior enters is shown (Figure 4.3). In our case this is the order master state. Because the state has not received a current order from the order node yet, the behavior is waiting in this state. 2 possible outcomes can be seen: customer_interaction and 'userdatachanged'. The state has the outcome customer_interaction if the keyword for the voice detector is triggered ('Albert'). The state has the outcome 'userdatachanged' if an order request has been received.



New release available!
Update the flexbe_app repository to get the new version.

At level: order master

Block transitions which require at least Low autonomy.

(execute current state again from beginning)
(stay active, but suspend execution)
(stopped behaviors can't be resumed)

Behavior Feedback

```
[9:27:38 PM] Onboard engine just started.
[9:28:46 PM] --> Starting new behavior...
[9:28:53 PM] Failed to execute state order master: [-]
    timeout exceeded while waiting for message on topic /order_node/current_order
[9:28:58 PM] Failed to execute state order master: [-]
    timeout exceeded while waiting for message on topic /order_node/current_order
[9:29:04 PM] Failed to execute state order master: [-]
    timeout exceeded while waiting for message on topic /order_node/current_order
```

Documentation

flex_albert_flexbe_states.OrderMasterState
retrieves latest order on the order list, and sets userdata accordingly

Outcomes:
userdatachanged: changed userdata for navigation and picking of product to information of current order

Figure 4.3: view of the behavior when in the order_master state. Until an order has been received, the behavior will wait in this state.

After entering an order request through the user interface, The order master has the outcome 'userdatachanged' and the next state 'move base' is entered (Figure 4.4). This state moves the robot base to the target shelf location. As can be seen, there are 3 possible outcomes: customer_interaction, 'arrived' and 'task_completed'. The state has the task completed outcome if the customer has only requested to be led towards the product. After arriving at the target location, the behavior will go to the 'scan shelf' state which can be seen in figure 4.5.

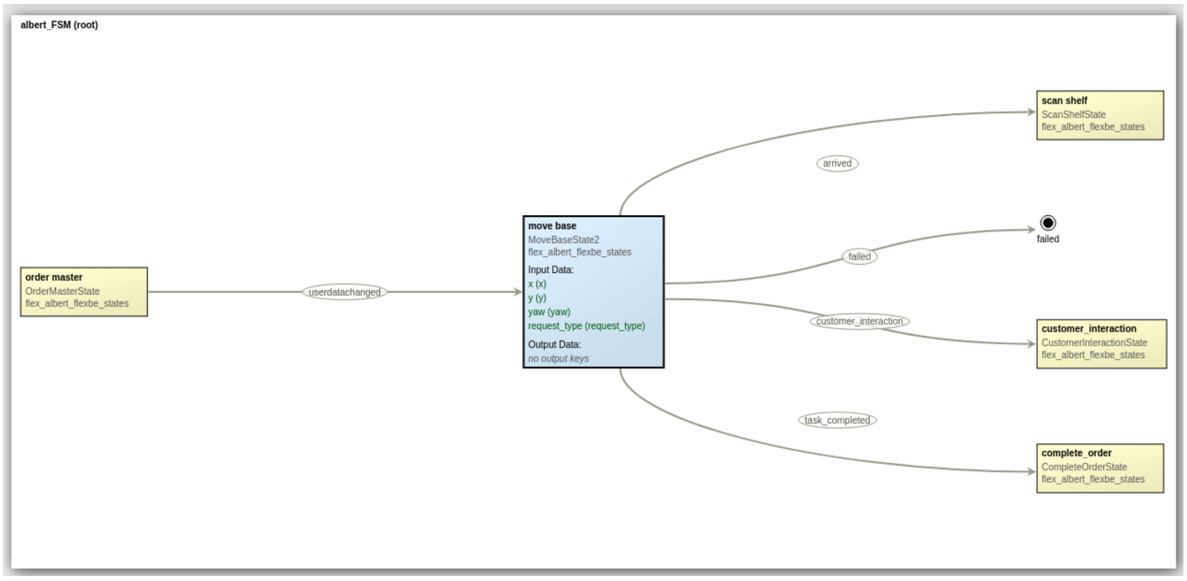


Figure 4.4: view of the behavior when in the move_base state

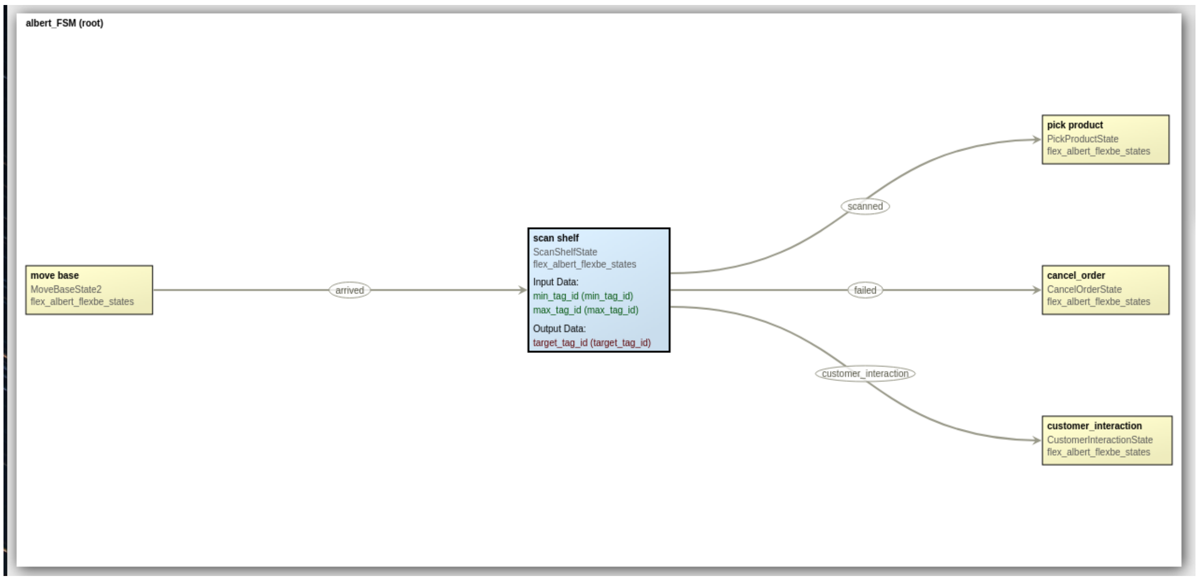


Figure 4.5: view of the behavior when in the scan_shelf state

Figure 4.6 shows the FlexBE behavior in action during the simulation. On the laptop screen, the runtime control overview can be seen while the behavior is being executed.

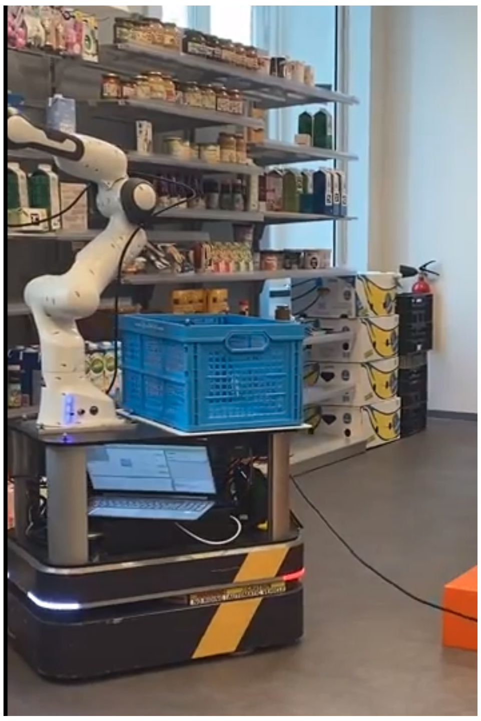


Figure 4.6: The FlexBE behavior in action during the robot simulation (the laptop screen shows the runtime control)

Figure 4.7 shows the complete behavior of the robot while the behavior is being executed. As can be seen, The current state the behavior has entered is displayed in blue.

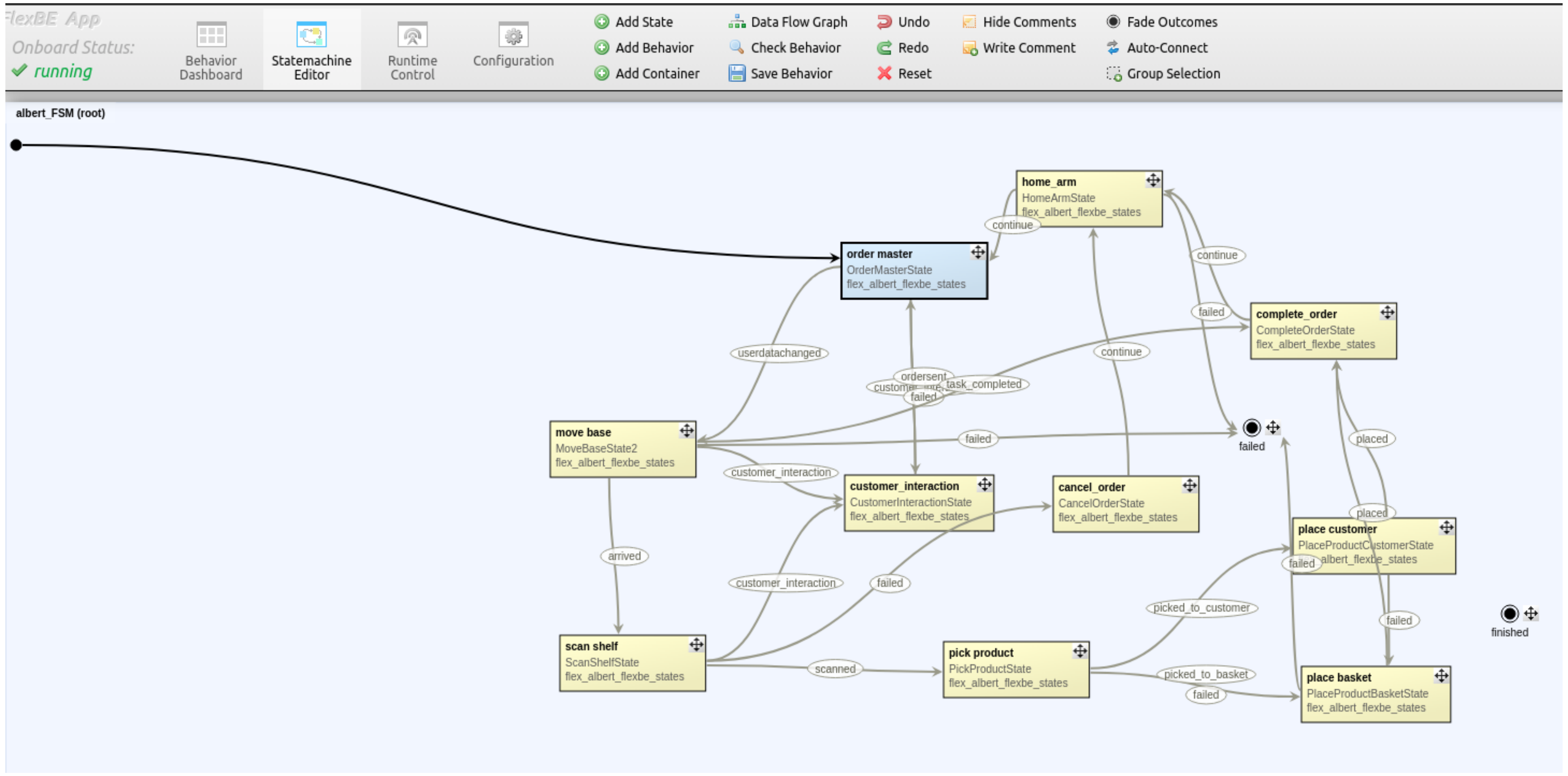


Figure 4.7: The FlexBE behavior in action during the robot simulation (the laptop screen shows the runtime control)

4.3 Recorded behaviors

This section gives an overview of all possible robot behaviors, supported by screenshots in Gazebo. The initial goal of these simulations is to check if every individual action works correctly. Each behavior is created by one of the specializations and the planning & navigation specialization basically connects all the behaviors. The recorded behaviors are connected with the FlexBE state machine discussed in 4.2.

4.3.1 Home arm

This behavior of homing the arm is part of the implemented motion control specialization. This behavior is implemented to make sure that the arm is in a natural position. It is folded up above its base, with the camera pointing forwards. The camera is orientated forwards to make obstacle detection possible. It turned out that the pose was not safe, so we changed it for real-world testing.

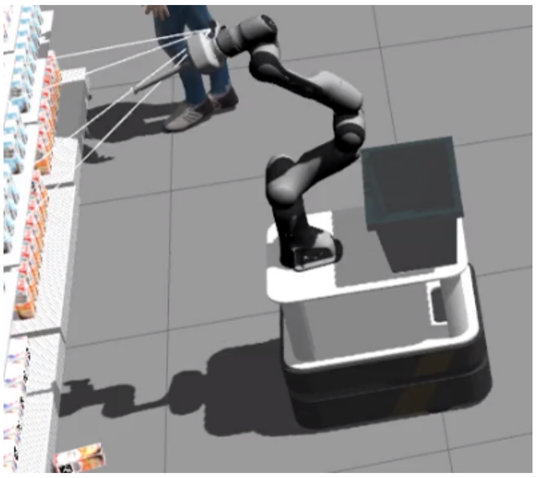
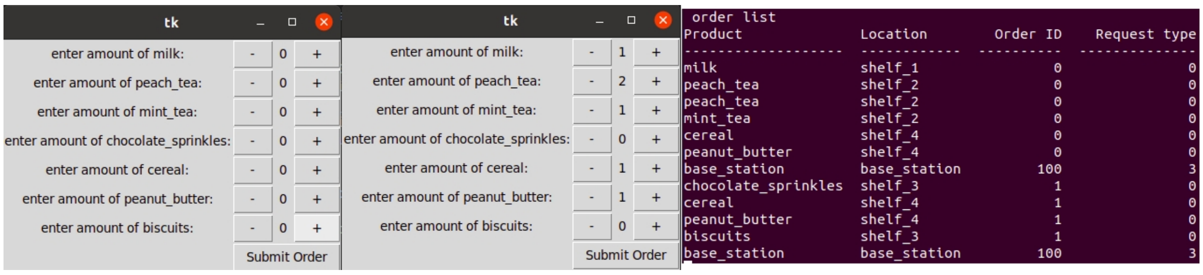


Figure 4.8: The home position of the robot arm

4.3.2 Order interface

We include a graphical order interface, to simulate orders arriving from customers online. The interface is part of the planning & navigation specializations, as well as falling slightly under human-robot interaction. The order interface pops up on the computer where the nodes are running from. The window shows all products present in the database and allows to add them to the customer's order list in an intuitive way. After filling in an order, it is sent to the `order_node`, which will add it to the list of pending orders.



tk

enter amount of milk:	-	0	+
enter amount of peach_tea:	-	0	+
enter amount of mint_tea:	-	0	+
enter amount of chocolate_sprinkles:	-	0	+
enter amount of cereal:	-	0	+
enter amount of peanut_butter:	-	0	+
enter amount of biscuits:	-	0	+

Submit Order

tk

enter amount of milk:	-	1	+
enter amount of peach_tea:	-	2	+
enter amount of mint_tea:	-	1	+
enter amount of chocolate_sprinkles:	-	0	+
enter amount of cereal:	-	1	+
enter amount of peanut_butter:	-	1	+
enter amount of biscuits:	-	0	+

Submit Order

```
 order list
Product              Location        Order ID    Request type
-------------------  ------------  ----------  --------------
milk                 shelf_1                0               0
peach_tea            shelf_2                0               0
peach_tea            shelf_2                0               0
mint_tea             shelf_2                0               0
cereal               shelf_4                0               0
peanut_butter        shelf_4                0               0
base_station         base_station         100               3
chocolate_sprinkles  shelf_3                1               0
cereal               shelf_4                1               0
peanut_butter        shelf_4                1               0
biscuits             shelf_3                1               0
base_station         base_station         100               3
```

Figure 4.9: Order interface initial situation, order interface right before submission, resulting order list after 2 orders

4.3.3 Move base

During this state, the robot moves the base to the waypoint which is next on the order list. This could be a product location or base station. Each location has an orientation. The behavior makes sure that the arm is in the home position. Moving the base is done using the `move_base`, a popular ROS package for navigation tasks.

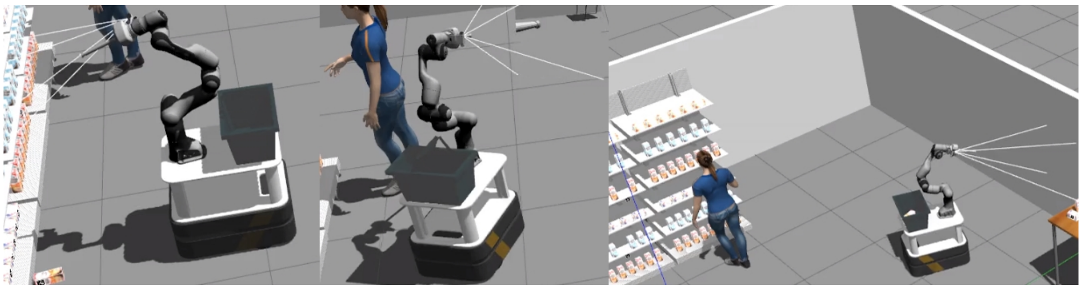


Figure 4.10: Move base behavior with 15 seconds between photos 1 and 3

4.3.4 Scan shelf

This state consists of scanning the shelves to find the wanted products. The products are marked by an April tag, which can be scanned using the `apriltag_ros` package provided by AprilRobotics. The range of April tags to look for has been passed to the state by the `order_master` state. If wanted April tags are found, the action server returns the id of the closest one, which can be used as a goal in the picking state.

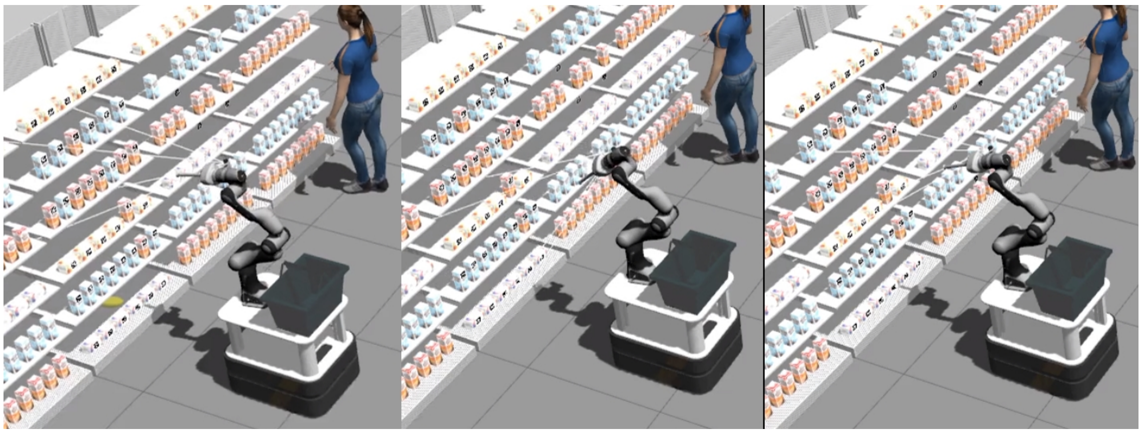


Figure 4.11: Scan shelf behavior with 2 seconds between each photo

4.3.5 Customer interaction

Customer interaction is part of the human-robot interaction specialization. While in operation, a user interaction can be initiated by saying the "Albert" trigger word. When an interaction is pending, the robot will complete its current task and then goes into the customer interaction state. The interaction is done by voice. Speech recognition is used to pick up the customer's voice and text-to-speech generation is used to give a response. The software uses the Chatgpt-API, which is connected to the product database, to be able to understand and respond to the human. The goal of the customer interaction state is to determine which product the customer wants, and whether they require picking assistance or not. Upon success, the state will send an order request to the `order_node`, just like when an online order arrives.

4.3.6 Pick product

The pick state is part of the motion control specialisation. The robot attempts to pick a product marked by a specific April tag id that has been found during the scan shelf state. After localising the product, the robot arm moves towards the product with the suction gripper activated so it immediately attaches the product on contact (in simulation it uses 'closest link attacher'-node) Then it holds the product in a position from where it decides to present it to a customer or place it in the basket.

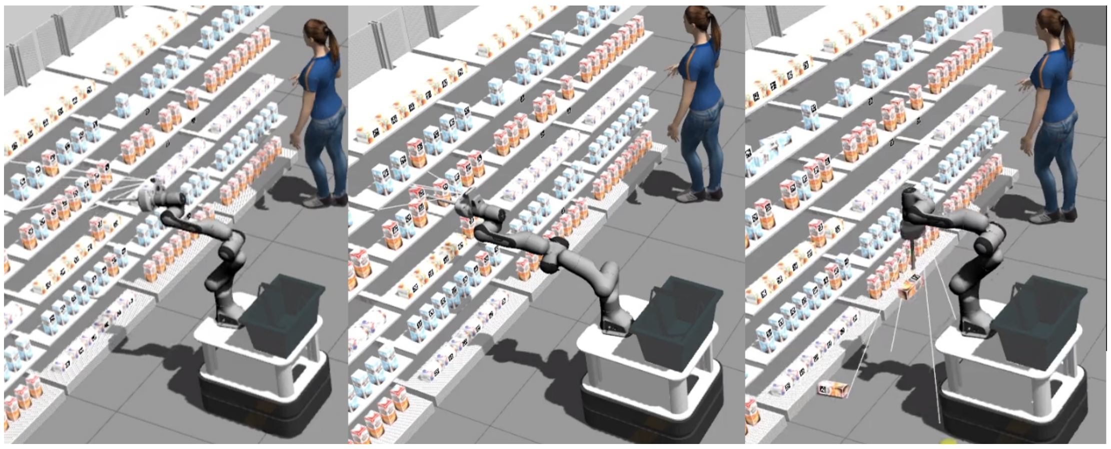


Figure 4.12: Picking products from the shelf.

4.3.7 Present Customer

The current action involving product-to-customer interaction falls within the realm of perception specialization. When a customer communicates their request for a specific product to the robot through voice interaction, the robot takes the initiative to guide and retrieve the requested item for the customer. By employing lidar and camera sensors, the robot accurately determines the customer's location and presents the product in their direction. An example scenario is displayed in Figure 4.13.

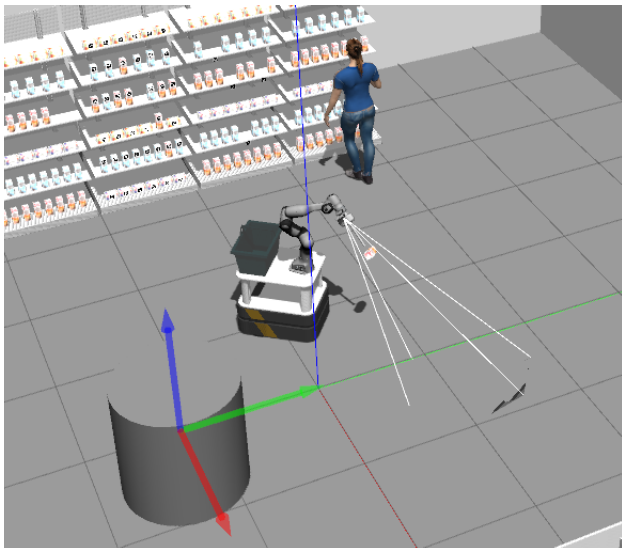


Figure 4.13: Scenario with 2 customer-like objects.

During the product presentation to the customer, Albert utilizes both the lidar sensor on the Boxer base and the camera mounted on the end effector of the Panda robot arm to locate the customer. The detection process begins by analyzing the lidar scan data, which captures a 270-degree angle view, excluding the region behind the robot. The responsible node called "scan_processor" processes laser scan data and generates TF (Transform) messages for detected clusters in the environment. The node converts the laser scan data to a PointCloud2 message and applies the DBSCAN clustering algorithm from the sklearn. cluster module to identify clusters. The detected clusters that satisfy certain width and depth criteria are filtered and their cluster centres are calculated. Finally, the node publishes TF messages for the closest clusters. The published frames are labelled based on their distance from the robot, with the nearest cluster labelled as "cluster_frame_1." Figure 4.14 shows the detected clusters. In this case, the cylinder is detected as the closest cluster, and the customer as the second closest cluster.

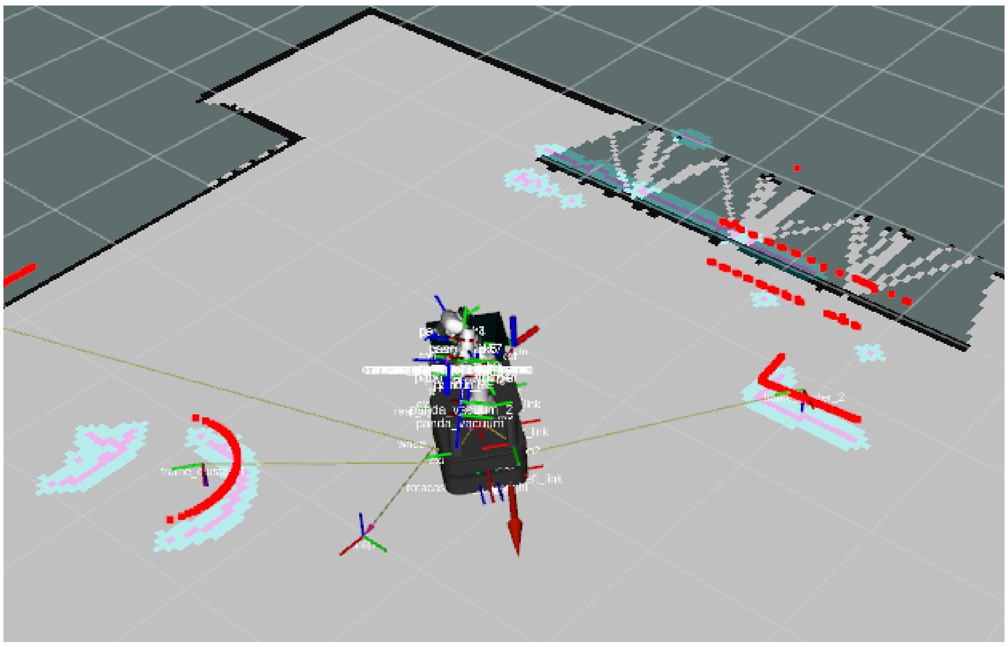


Figure 4.14: The lidar_cluster node detects both the cylinder and the customer.

Subsequently, the robot's arm rotates towards the closest cluster, and the last link of the Panda arm extends to align the camera with the potential customer. To determine whether the camera frame contains a customer, we employ a HOG (Histogram of Oriented Gradients) detector. This detector analyzes the visual features in the camera frame and determines the presence of a customer. If the identified cluster does not correspond to a customer, the robot proceeds to check the next closest cluster. If this cluster also does not represent a customer, the product presentation is conducted at a predefined standard location on the right side of the robot. This standard location serves as a fallback option when no customer is detected. In this case, the robot first checks the cylinder and the customer_detection node returns the boolean False to indicate that the image does not contain the customer, as shown in Figure 4.15.

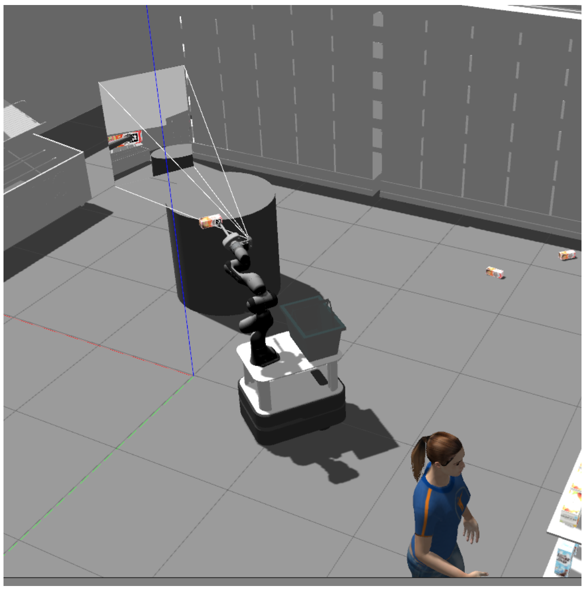


Figure 4.15: The customer_detector node verifies that the closest customer-like cluster is not the customer.

The robot arm then rotates towards the customer and the customer_detection node returns the boolean True to confirm that the camera image indeed contains the customer. The robot continues to present the product to the customer, waiting for the customer to pick the product off the vacuum gripper, as can be seen in Figure 4.16.

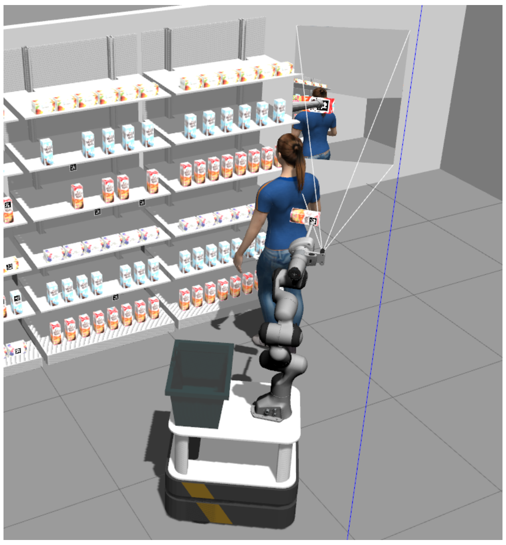


Figure 4.16: The customer_detector node verifies that the second-closest cluster is indeed the customer.

4.3.8 Place basket

This behavior is part of the motion control specialisation as well. When the robot successfully picked the product from the shelf, which is for the order list, it places the product just a little above the basket it is carrying and then deactivated the suction gripper (closest link attacher). The product then drops safely into the basket.

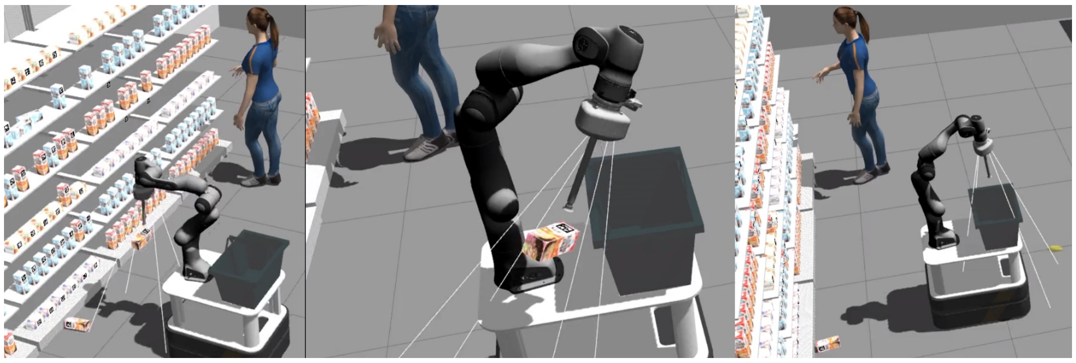


Figure 4.17: Placing the picked product into the basket

4.3.9 Complete order

This action is part of the implemented human-robot interaction and planning & navigation specializations. After completing the (sub)order it will navigate back to the base station, where its basket can be replaced by an empty one. Then the robot is ready for the next (sub)order.

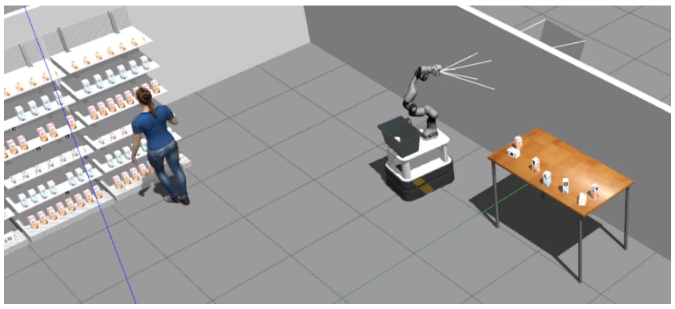


Figure 4.18: Robot returns to its base station and requests a basket change

SUMMARY OF REAL ROBOT RESULTS

This chapter is about the workings of the real robot in a full sequence and comparing results with the proposed behaviour. The real-world testing session is provided by Airlab, which included Albert and a supermarket testing environment with test products. Some initial changes were needed to make the test session work, these are mentioned in 5.1. Paragraph 5.2 describes the robot's behaviour during the full sequence. During running the full sequence, we encountered several bugs, which is the reason we included a debugging report in paragraph 5.3.

We used a test session to see if the separated behaviours operate as intended in the real world. This is also necessary to test the reliability. During these sessions, it was most important to see if the robot was able to pick products in the store from an customer's online order list and from an in-store customer's voice request.

5.1 Functional graph

First of all, there are some differences in hardware compared to the simulation environment. The real Albert is equipped with safety stops, one for the arm and one for the base, for example. The arm stop has a long wire so it can be pressed from a distance, one difficulty was this wire which hang before the lidar sometimes, which influenced the robot's behaviour badly. Secondly, we manually stuck April tags on the products we added to our database and placed them on shelves. Then, we manually drove Albert to locations before the shelf to get the raw location of the products. These locations are then linked to the products in the database.
Software-wise, there are a few minor changes made for the test session compared to the RQT graph 4.1:

- `/apriltag_detection` is updated with the actual April tag size. For the testing session, we printed 36h11 April tags with size 40mm. This size needed to be updated in the code, for the product locations to be correctly estimated by this node.
- `/map` is changed with the map of the Airlab testing environment. This is needed for the self-localisation of Albert to be correct.
- `/order_node` is not necessarily changed, but the output of this node is different in comparison with the simulation. The reasons for this are a different product database with products and April tags which differ from the ones in the simulation. Related to this are the different shelf locations, as the map has changed. Finally, incoming orders are randomly given to Albert, which obviously influences the behaviour.
- `/home_arm_server` is changed to another home pose in the launch file, because the original pose was not safe. The suction gripper protruded more than anticipated. The new pose points the gripper towards the basket, so nothing protruded.

5.2 Recorded behaviours

The recorded behaviours stated in 4.3 are tested in the full sequence during the final test session. This allowed us to test the reliability of the system. The results are visualised with photos and screenshots from videos. Successes are identified and eventual differences are analysed.

5.2.1 Home arm

The real-world application of the robot's homing arm behavior was successful. By folding the arm up above its base and orienting the camera downward, the robot ensured safety during navigation. It avoided knocking over products with the suction gripper and minimized the risk of contact with nearby humans. Overall, the implementation of this behavior resulted in a smooth and accident-free real-world operation.

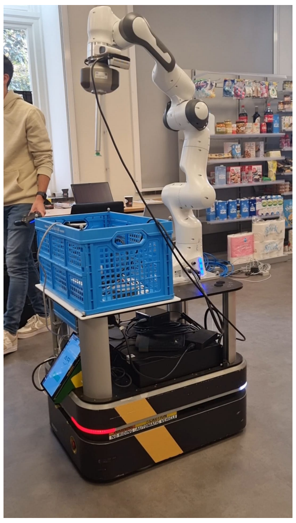
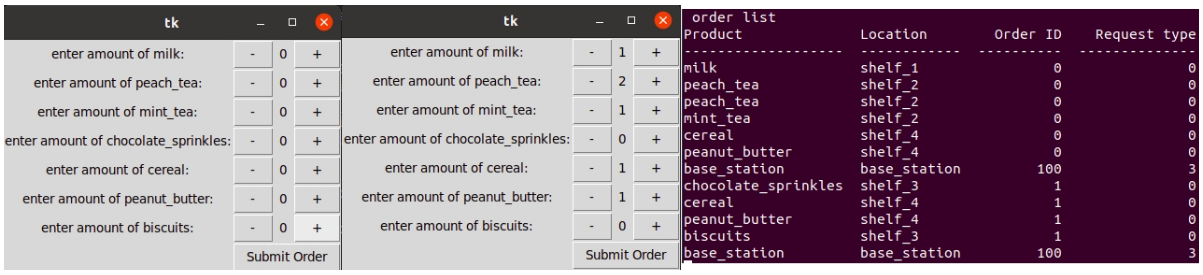


Figure 5.1: The home position of the robot arm

5.2.2 Order interface

The real-world application of the graphical order interface was successful. Integrated into the planning, navigation, and human-robot interaction specializations, the interface appeared on the computer running the nodes. It provided a window displaying all products in the database and enabled intuitive selection for the customer's order list. Once an order was completed, it was sent to the "order_node" and added to the list of pending orders. This streamlined process facilitated efficient order management and enhanced the overall functionality of the system.



enter amount of milk:	-	0	+
enter amount of peach_tea:	-	0	+
enter amount of mint_tea:	-	0	+
enter amount of chocolate_sprinkles:	-	0	+
enter amount of cereal:	-	0	+
enter amount of peanut_butter:	-	0	+
enter amount of biscuits:	-	0	+

enter amount of milk:	-	1	+
enter amount of peach_tea:	-	2	+
enter amount of mint_tea:	-	1	+
enter amount of chocolate_sprinkles:	-	0	+
enter amount of cereal:	-	1	+
enter amount of peanut_butter:	-	1	+
enter amount of biscuits:	-	0	+

```
 order list
Product             Location      Order ID    Request type
------------------  ------------  ----------  --------------
milk                shelf_1              0               0
peach_tea           shelf_2              0               0
peach_tea           shelf_2              0               0
mint_tea            shelf_2              0               0
cereal              shelf_4              0               0
peanut_butter       shelf_4              0               0
base_station        base_station       100               3
chocolate_sprinkles shelf_3              1               0
cereal              shelf_4              1               0
peanut_butter       shelf_4              1               0
biscuits            shelf_3              1               0
base_station        base_station       100               3
```

Figure 5.2: Order interface initial situation, order interface right before submission, resulting order list after 2 orders

5.2.3 Move base

The real-world application of the robot's base movement behavior was successful. In this state, the robot accurately navigated to the specified waypoint where the desired product was located. As part of the behavior, the robot ensured that the arm was in the home position, maintaining safety and efficiency. The base movement was accomplished using the widely used ROS package called "move_base," specifically designed for navigation tasks. This implementation allowed the robot to swiftly and reliably reach the target location, contributing to the overall effectiveness of the system.

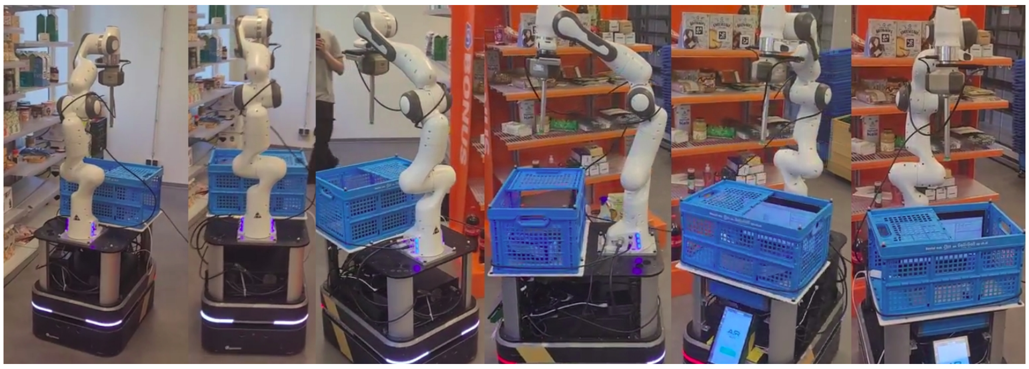


Figure 5.3: Move base behaviour with 15 seconds between photos 1 and 6

5.2.4 Scan shelf

The real-world application of the shelf scanning state was successful. During this state, the robot effectively scanned the shelves to locate the desired products. The products were identified using April tags, which were easily detected using the "apriltag_ros" package provided by AprilRobotics. The range of April tags to search for was communicated to the state by the "order_master" state. When the desired April tags were found, the action server returned the ID of the closest tag, which could then be used as a goal in the subsequent picking state. This streamlined process of scanning and identifying products contributed to the overall efficiency and accuracy of the robot's operations.

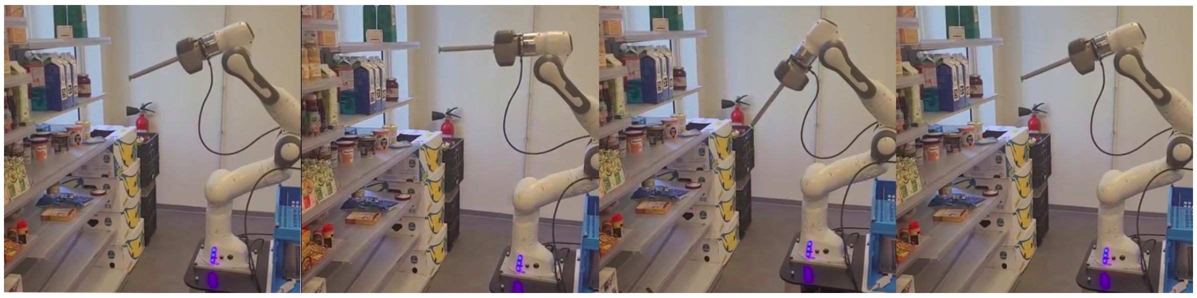


Figure 5.4: Scan shelf behaviour with 2 seconds between each photo

5.2.5 Customer interaction

The real-world application of the customer interaction state, which falls under the human-robot interaction specialization, was successful. During operation, the user interaction could be initiated by using the trigger word "Albert." When an interaction was pending, the robot completed its current task before entering the customer interaction state. This state facilitated voice-based communication between the robot and the customer. Speech recognition technology enabled the robot to understand the customer's voice, while text-to-speech generation allowed the robot to respond verbally. The software utilized the Chatgpt-API, which was connected to the product database, enabling the robot to comprehend and respond to human queries effectively.

The goal of the customer interaction state was to determine the specific product the customer wanted and whether they required any picking assistance. Upon successful interaction, the state sent an order request to the "order_node," similar to when an online order arrived. This seamless integration of voice-based customer interaction and order processing contributed to the smooth and efficient operation of the robot system.

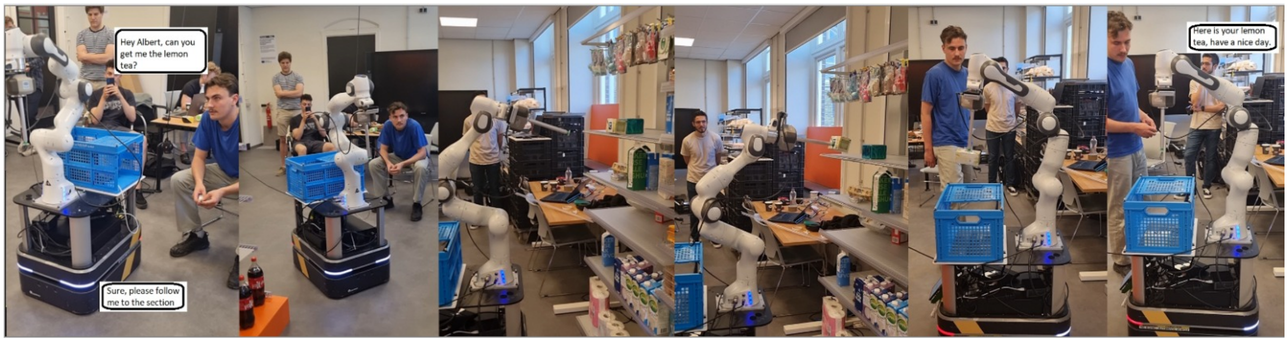

Hey Albert, can you get me the lemon tea?
Sure, please follow me to the section
Here is your lemon tea, have a nice day.

Figure 5.5: The customer interaction with the Albert robot.

5.2.6 Pick product

The real-world application of the pick state, which is part of the motion control specialization, was successful. In this state, the robot aimed to pick a product identified by a specific April tag ID that was previously detected during the shelf scanning state. Upon localizing the product, the robot arm smoothly moved towards it while activating the suction gripper. As soon as contact was made, the gripper was securely attached to the product. Figure 5.6 presents the successful application.

The robot then held the product in a position from which it determined whether to present it to a customer or place it in the basket. This decision was likely based on pre-defined criteria or instructions provided by the system. The effective coordination of the robot arm, suction gripper, and decision-making process ensured efficient product handling during the pick state. Overall, the successful execution of this behavior contributed to the seamless operation of the robot system and its ability to handle products accurately and effectively.

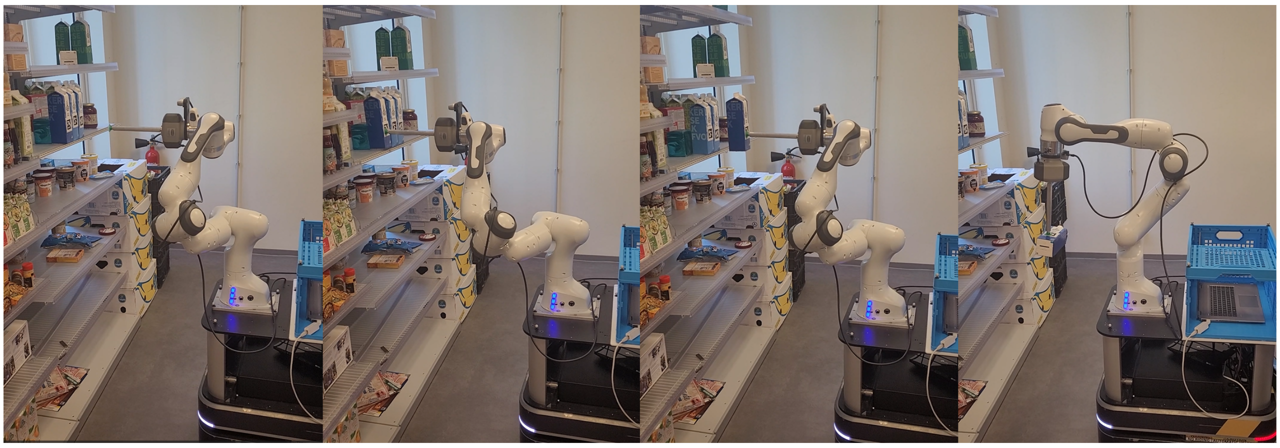


Figure 5.6: Picking products from the shelf.

5.2.7 Present Customer

The real-world application of the product-to-customer interaction action encountered some challenges during the test session. The scan_processor node experienced unexpected errors that the present server was unable to handle effectively. As a result, the robot resorted to presenting the product at the default location, as shown in Figure 5.7. Despite this setback, the robot successfully completed the presentation of the product to the customer, awaiting their interaction with the vacuum gripper.

It is worth noting that the limitations faced during the test session impacted the robot's ability to accurately detect and locate the customer using lidar and camera sensors. However, the robot persisted in presenting the product, demonstrating its resilience and adaptability to handle unforeseen challenges in real-world scenarios.

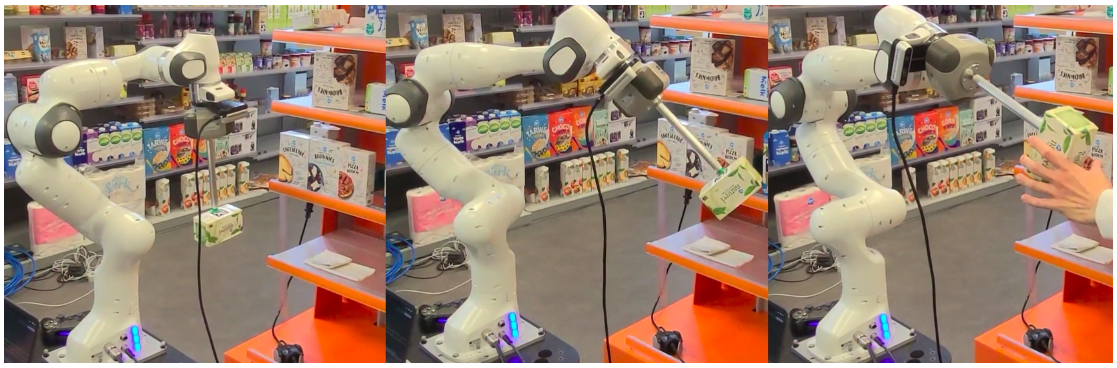


Figure 5.7: Real-life demonstration of presenting the product to the customer.

5.2.8 Place basket

Place product in the basket was successful as can be seen in figure 5.8. Once the robot successfully picked the product from the shelf, it positioned the product slightly above the basket it was carrying. The suction gripper was then deactivated, allowing the product to safely drop into the basket. This precise motion control ensured the product was securely placed in the designated location, contributing to the overall efficiency and accuracy of the system. The successful execution of this behavior facilitated seamless product handling and fulfillment of the order list.

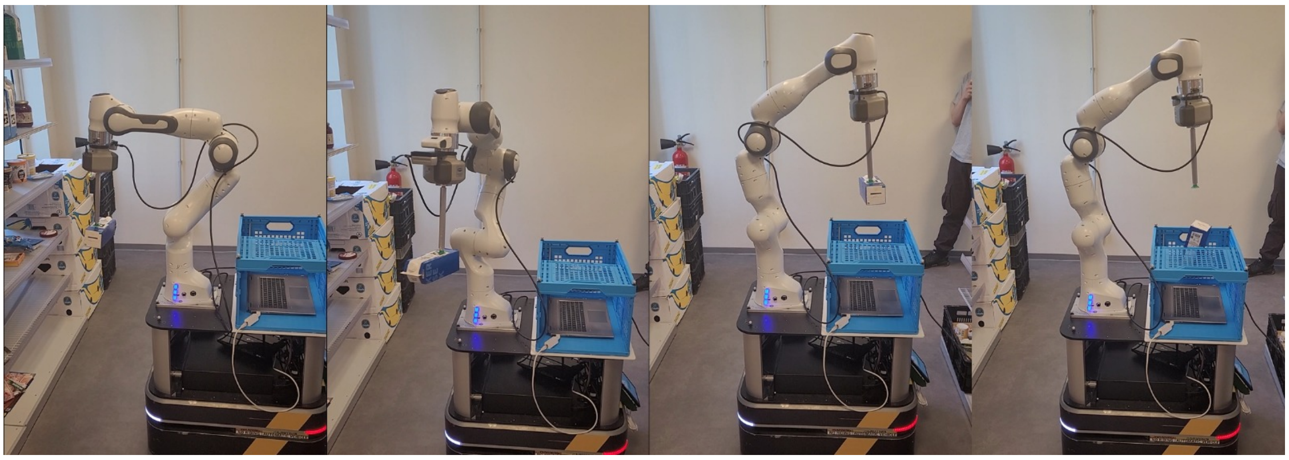


Figure 5.8: Placing the picked product into the basket

5.2.9 Complete order

The real-world application of the action involving returning to the charging station and replacing the basket was successful. Figure 5.9 confirms the proper real-world execution of this action. After completing a suborder or order, the robot effectively navigated back to the charging station. Once there, the basket was replaced with an empty one, ensuring the robot was ready for the next task.

By seamlessly returning to the charging station and replacing the basket, the robot demonstrated its ability to autonomously manage its energy levels and replenish its resources. This efficient workflow allowed for uninterrupted operation and optimized utilization of the robot's capabilities. Overall, the successful execution of this action contributed to the overall effectiveness and productivity of the robot system.

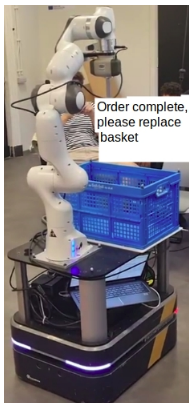


Figure 5.9: Robot returns to its base station and requests a basket change

5.3 Debugging report

During the final testing session, we encountered several bugs, which are listed below. This list should give our client a comprehensive insight into the issues we encountered. This information could help with faster troubleshooting in future work.

5.3.1 Locations where not reachable

Description: The simulation environment had deep shelves on the lidar-scanning height, which did not activate the collision safety bound for the base movement. The real-world testing environment did have products on scanning height, which caused strange robot movement because of an unreachable location.
Solution: There were two solutions for this problem, which were lowering the safety margin, or changing the waypoint a bit further away from the shelf. We choose the second option as this one was easier and keeps the safety margin.

5.3.2 Lidar_cluster_node

Description: The lidar_cluster_node generated an error during the real-life test session. As explained in section 5.2.7, this node should cluster the lidar points received by the lidar an nearby customer to hand the picked product. However, during the final test session, this node failed, which made handing the product to the customer impossible as his location is unknown.
Solution: The fast solution for this bug was to implement a default in the FlexBE behaviour tree of the place-in-basket action. This way we could finish a full sequence of the voice request scenario.

5.3.3 Products bump into shelf

Description: The arm sometimes hid the shelf with the picked product. The reason for this is that we did not add a collision box around the picked product. (We removed the product's collision box last test session because of unexpected behaviour.)
Solution: The solution to this problem was to adjust the picking movement by going 5cm further backwards, before rotating towards its next position. Further benefits could be had by introducing intermediate joint value positions, as the arm controller has much less difficulty planning to a joint position compared to an endpoint pose. Also, a welcome addition would be to introduce a collision plane on the shelf itself, preventing the arm from passing through the shelf during the picking motion.

5.3.4 Picking action using weird movements

Description: While picking products in the simulation, the arm would make sub-optimal movements and have difficulty picking products. The reason for this seemed to lay in the planning scene object we had placed on the robot's basket. This object was used to prevent the end effect or from hitting the robot's basket during picking and placing, which works well. The unforeseen problem with this was that the

arm sometimes planned unpredictable trajectories to avoid any of its joints coming into contact with the collision obstacle.
Solution: Adding a "home-arm" node which returns the arm to a base position from which it can more successfully and reliably pick new products. Also, during testing, we removed the basket collision obstacle, which helped mitigate some of the undesired behaviors.

5.3.5 Customer Interaction failed

Description: The customer interaction node failed during the final test session because the prompt was changed to be more specific, which caused the input character limit to be exceeded. This led the chatGPT agent to give its answer in the wrong format, meaning it could not be parsed correctly by the interpreting code. While products could still be ordered from the GUI, voice interaction orders failed during the start of the test session.
Solution: The solution was to revert the voice recognition file back to a previous, less detailed version. This caused the chatGPT API to give answers in the right format again.

5.3.6 Nodes not working

Description: We had problems with nodes shutting off after the Panda arm failed to reach a goal position. This was due to the Franka arm automatically closing certain nodes running on the robot's internal computer. This then caused the whole system to lose connection to the robot.
Solution: The solution was to stop all nodes, restart the robot and then start all nodes again. If possible, AIRLab may want to investigate why the panda arm automatically shuts down certain nodes, and if the arm could be restarted independently from the base.

5.3.7 Panda arm crashing

Description: After the placing in basket action, the first joint of the robotic arm was right on its limit. When we started to move to the next location, the arm sometimes hit its limit due to vibrations. This caused the panda arm to automatically go into a safety stop, which was unnoticed by us until we expected the next arm movement, which is the scan shelf action.
Solution: The solution for this issue was to change the drop-off location a few degrees off the basket's centre. This made sure that the robot could not hit its limits during the base movement.

INDIVIDUAL REFLECTION

6.1 Ernst

6.1.1 Learning Goals

- Specific: learn more about programming robot behaviors.
- Measurable: create a working behavior system for a robot that is able to complete the project goal.
- Achievable: Continuously add at least one node to the FlexBE system each week.
- Relevant: Adding in a working FlexBE behavior system will allow the robot to complete complicated tasks independently.
- Time-bound: have a working robot behavior system before the last live testing session in the Airlab.

6.1.2 Peer Feedback Received and Peer Feedback Given

There were two rounds of peer review during the project.

During the first round of peer review, the feedback that I got consisted of several points. Positively, I got feedback that I had a good overview of the project and that I was working hard. I got mixed feedback about the progress updates I was giving. Another point was that it was unfortunate that I cannot always be present during the group sessions because of my internship.

My experience regarding this feedback is that it is mostly valid. I could indeed be present physically during the group sessions less often than other group members. I also believe I could improve my communication skills by improving the progress updates I was giving. I spent a lot of time working on the project on my own, and therefore believe that the positive feedback I got regarding the work I was doing is valid.

During the second round of peer review, the feedback I received was more positive. I got positive feedback regarding the work I did on the FlexBE behavior system. There was almost no negative feedback.

I think the reason the feedback was more positive during this round was that the FlexBE system was more finished and its capabilities for controlling the robot could now be monitored more clearly. I also think the feedback during the second round of peer review was more positive because we as a group were more adapted to working with each other and this caused improved understanding for each others situations.

What I have learned from this feedback is that I should spent more time trying to communicate with group members what I am working on. I also learned that it is important to be present physically, as this leads to much better communication in comparison to online communication.

Regarding the feedback I gave to other group members, It was mostly that I think that everyone should try to stay in the loop regarding the entire simulation and code and not just focus on their specific part of the project. I think as the project progressed everyone was in the loop on the code and simulation and tried their best to stay up-to-date. I do not know if this was because of my feedback or just because everyone started spending more time on the project as it progressed.

6.1.3 Transferable Skills

the following skills will be assessed in this section: teamwork, leadership and entrepreneurial thinking. Of these skills, I think I am more adept at certain ones and that improvement would be welcome for others. I think my teamwork skills are reasonable. I am not certain about the level of my leadership skills. regarding entrepreneurial thinking, I think this area needs a lot of improvement. Getting new ideas is very important and can add a lot of value to projects. At this moment, I believe becoming more adept in thinking of new ideas or strategies would be very beneficial.

6.1.4 Agile, Specialist Roles, and Other Team Roles

I think we were able to apply Agile project management succesfully because were able to follow the sprints and deliver the report parts on time. It helped that because of this the workload of the project was divided over all weeks and not just near the end of the project.

6.2 Henk

6.2.1 Learning Goals

- Specific: Gain a strong understanding of robotics simulation and improve proficiency in ROS (Robot Operating System).
- Measurable: Successfully complete project deliverables, demonstrate practical skills in robotics simulation and ROS, and actively participate in team discussions and activities.
- Achievable: Attend all course sessions, engage in hands-on practice with robotics simulation and ROS, and actively collaborate with team members.
- Relevant: The learning goals align with the objectives of the course and contribute to enhancing knowledge and skills in the field of robotics.
- Time-bound: Complete project deliverables within the specified timeline, actively participate in team activities throughout the duration of the course, and demonstrate progress and improvement over time.

Learning goals: My learning goals for this course were to develop a strong understanding of robotics simulation, improve my skills in ROS (Robot Operating System), and enhance my ability to collaborate effectively in a team project.

Evolution of goals: Over time, my goals evolved as I gained more knowledge and experience in the course. Initially, my focus was on understanding the basics of robotics simulation and ROS. As the course progressed, I aimed to apply my knowledge by actively participating in the project and taking on specific responsibilities.

Achievement of goals: I believe I managed to reach my learning goals to a significant extent. I worked diligently on the project, consistently attended sessions, and made an effort to keep my team members informed about my progress. I dedicated considerable time to working on the report deliverables and implementing the required skills for the project.

Feedback from peers: The feedback from my peers indicates that they recognized my strengths in areas such as working on the report, communication, and independent work. They appreciated my efforts in finding solutions and my commitment to the project.

Areas for improvement: The feedback also highlighted some areas where I could improve. Suggestions included getting more involved in the simulation part of the project, gathering more information about the current state of the robotic simulation, and familiarizing myself with the simulation environment. Additionally, it was suggested that I try to keep a better overview of the entire project and be early with deadlines if teammates depend on them.

Agreement with feedback: Overall, I agree with the feedback provided by my peers. Their suggestions align with my own reflections on areas where I can further develop my skills and contribute more effectively to the project. I appreciate their input and will strive to implement these suggestions in future collaborations.

6.2.2 Peer Feedback Received and Peer Feedback Given

Regarding the peer feedback I received, I find it valuable and insightful. It highlights both my strengths and areas for improvement, which is crucial for personal growth and development. I generally agree with the feedback provided as it aligns with my own self-assessment and observations during the course.

From the feedback, I have learned that I have been diligent in working on the project deliverables and keeping my team members informed about my progress. However, there is room for improvement in terms of actively involving myself in the simulation part of the project, gaining more familiarity with the simulation environment, and expanding my knowledge in ROS.

As for the peer feedback I gave to other students, their reactions varied. Some students were receptive and acknowledged the feedback, while others may have been defensive or disagreed to some extent. It is natural for individuals to have different perspectives and reactions to feedback.

To improve the way I give feedback to my peers, I can focus on being more constructive and specific in my comments, highlighting both their strengths and areas for improvement. Additionally, I can ensure that I communicate my feedback in a respectful and supportive manner, emphasizing the importance of continuous improvement and learning from each other's perspectives.

6.2.3 Transferable Skills

During the course, I have had the opportunity to reflect on and develop several key skills, including teamwork, leadership, entrepreneurial thinking, and strategic multidisciplinary problem-solving.

In terms of teamwork, I have strengthened my ability to collaborate effectively with team members, communicate ideas, and actively contribute to group discussions. I have learned the importance of listening to different perspectives, resolving conflicts constructively, and leveraging the diverse skills and strengths of team members to achieve common goals.

Regarding leadership, I have had the chance to take on leadership roles within the team, such as coordinating tasks, setting deadlines, and ensuring progress. I have developed my skills in delegating responsibilities, providing guidance and support to team members, and fostering a positive and productive team dynamic.

In terms of entrepreneurial thinking, I have learned to think creatively, identify opportunities, and propose innovative solutions. I have explored ways to approach problems from an entrepreneurial mindset, considering factors such as feasibility, market potential, and value creation. This has helped me broaden my perspective and think beyond conventional solutions.

Strategic multidisciplinary problem-solving has been a central aspect of the course. I have learned to analyze complex problems from different angles, integrate knowledge from various disciplines, and develop strategic approaches to tackle them. I have honed my ability to consider both short-term and long-term implications, evaluate risks and benefits, and make informed decisions.

In terms of personal interests, I have found a strong inclination towards strategic multidisciplinary problem-solving. I enjoy exploring complex problems, breaking them down into manageable components, and applying a holistic approach to find effective solutions. I find the process intellectually stimulating and rewarding.

Through the course, I have identified both new strengths and weaknesses. My strengths lie in effective communication, collaboration, and the ability to analyze problems from multiple perspectives. I have also developed a good understanding of entrepreneurial thinking and its application. However, I have identified a need for further improvement in areas such as delegation, time management, and enhancing my knowledge in certain technical domains.

Overall, the course has provided me with a valuable opportunity to develop and reflect on these skills, allowing me to gain self-awareness and identify areas for further growth and improvement.

6.2.4 Agile, Specialist Roles, and Other Team Roles

Yes, I was able to apply Agile project management successfully during the course. I found the Agile methodology to be highly effective in managing the project and promoting collaboration and adaptability within the team.

What I particularly liked about Agile project management was the emphasis on iterative development, continuous feedback, and frequent communication. It allowed us to respond quickly to changes, adjust our plans accordingly, and deliver incremental results. The regular meetings, such as daily stand-ups and sprint reviews, helped to keep everyone aligned and informed about the project's progress.

To improve my proficiency in Agile project management, I can focus on a few key areas. First, I can deepen my understanding of different Agile frameworks and methodologies, such as Scrum or Kanban, to expand my knowledge and adopt best practices. This can involve studying relevant literature, attending workshops or training sessions, and actively seeking opportunities to apply Agile principles in real-world projects.

Second, I can work on refining my skills in facilitating Agile ceremonies and managing team dynamics. This includes ensuring effective communication, encouraging collaboration, and fostering a supportive and productive team environment. I can also strive to improve my ability to identify and address potential bottlenecks or impediments that may arise during the project.

In terms of responsibility in my role, I took my assigned tasks seriously and actively contributed to the team's objectives. I fulfilled my responsibilities within the Agile framework, such as updating task boards, participating in planning sessions, and providing regular progress updates. I also supported my team members when needed, offering assistance and seeking opportunities to contribute beyond my own tasks.

Having a specific role within the project allowed me to discover certain aspects about myself. For example, being responsible for a particular area of the project helped me develop a deeper expertise in that domain. It also highlighted the importance of taking ownership of my assigned tasks, meeting deadlines, and communicating effectively with my team members. I learned to manage my time efficiently and prioritize tasks to ensure successful completion.

Overall, the experience of applying Agile project management was positive, and I recognize its value in fostering collaboration, adaptability, and iterative development. By continuing to expand my knowledge, refine my facilitation skills, and embrace responsibility in my role, I can further enhance my proficiency in Agile project management and contribute effectively to future projects.

6.2.5 Other Key Reflection Conclusions

During this course, I learned a great deal about myself beyond the specific skills and knowledge related to the subject matter. I discovered that I thrive in collaborative environments and enjoy working as part of a team towards a common goal. I realized the importance of effective communication, active listening, and being open to different perspectives and ideas. I also recognized the value of taking initiative and being proactive in seeking opportunities for growth and learning.

Moreover, I learned that I am capable of adapting to new challenges and unfamiliar situations. I embraced the iterative nature of the project and was able to adjust my approach based on feedback and changing requirements. I became more comfortable with uncertainty and ambiguity, understanding that they are often part of complex problem-solving processes.

Additionally, I discovered the significance of self-reflection and continuous improvement. By regularly assessing my strengths, weaknesses, and areas for growth, I was able to make adjustments and strive for personal and professional development. I recognized that learning is a lifelong journey and that seeking feedback and actively incorporating it into my practice is crucial for growth.

Overall, this course provided me with valuable insights into my capabilities, preferences, and areas for improvement. It reinforced the importance of teamwork, adaptability, and self-reflection in my personal and professional development. I am now more aware of my strengths and weaknesses, and I am motivated to continue learning and growing in order to become a more effective and well-rounded individual.

6.3 Marijn

6.3.1 Learning Goals

- Specific: Learn how to efficiently collaborate on a software project with a team.
- Measurable: High quality of documentation, robust code and good communication regarding software functionality.
- Achievable: The goals are achievable by keeping the learning goals in mind while working on the software of the project.

- Relevant: In later work environments, I want to ensure that I am a good colleague to work with.
- Time-bound: Have working and documented software ready before the final test session.

6.3.2 Peer Feedback Received and Peer Feedback Given

I was very happy with most of the feedback I received, and it seemd to reflect the way I was feeling during the project. During the starting weeks I was working more on the coding side of the project. The feedback I received was positive: good energy and good coding skills. That is the kind of feedback I was looking for from my group mates. As the project continued I put in many hours of coding. Although I enjoyed being responsible for this part of the project, I sometimes felt that I had maybe done a bit too much. The fact that many of the packages were written by me meant that there was more communication needed to update the other group members on the things I had implemented, and this communication did not always go as well as I hoped. The feeling of disconnect I had at that time also influenced my attitude: sometimes I was left feeling stressed, or like I was the only one that understood exactly what had to be implemented and how in order for the project to be a success on the coding side of things. This is seen in my feedback for the second buddycheck as well: an excellent score on performance but slightly lower scores on communication and attitude. All in all, I have learned that my peers do really appreciate it when I put in hard work. Still, progress has to be made in maintaining good communication and a positive attitude at all times.

Giving feedback went badly, as I failed to submit my feedback for the second buddycheck on time. This is of course a pretty bad error om my side, and has something to doe with my chaotic nature and questionable planning skills. For the first feedback round I feel my feedback could have been a bit more specific, but it did very honestly reflect the feelings I had about the group at that time.

6.3.3 Transferable Skills

I really tried to become a better and more efficient team player during the course of this project. My personal interests lay mostly on the specific coding side of things, more specifically the workflow and version management of working together on the same codebase. I developed new strengths in this area during the course, but sadly it did not often feel like teamwork on the coding side. This was because some members had made their own files and packages, and the understanding of the specific workings of each-others code was not that present.

Leadership in the project was also a bit of an issue. At times, we felt a bit down or lost, meaning someone needed to step up in order to ”take the steering wheel”. I tried being the leader at times, but cannot say that I see myself as having grown in this role. Maintaining a good overview of all of the deadlines and deliverables is a daunting task, and I would often lose track of these things while spitting through code on my laptop all day. All in all, I would like to improve my overview on the leadership side of projects, so that I can perform as an engineer as well as more of a leader.

6.3.4 Agile, Specialist Roles, and Other Team Roles

Agile project management was applied in order to finish all of the sprint goals on time. However, to me the project felt like one entire giant stressful sprint, with new deadlines peeking around the corner seemingly each day. Of course, expecting a team that has never worked with agile to perfectly implement it in their workflow first-time is a bold wish, but I can see what the method holds in store for teams.

My role in the team was the HRI specialist, which was initially the lowest on my list of choices. This was probably due to the fact that I perceive the HRI discipline to be less technical and require less coding. Looking back, I am very happy with the role I received, as the HRI component of the project had the ability to really distinguish our solution from other groups. Implementing chat GPT in a voice assistant was something that I came up with during the first test session. It took some time crafting the perfect prompt in order to get the responses just right, but the enthusiastic responses from the client and supervisors made it well worth it and gave me a great sense of success.

6.3.5 Other Key Reflection Conclusions

This may sound a bit pessimistic, but I think it must be said: during this project I got to know a slightly darker version of myself which i didn't know existed. The hard work I put in to the Robotics master this year made me a bit worn out during this last quarter. I really felt this in terms of attitude: if one or two small things went wrong during the project, I would be left feeling very frustrated and at times even angry. I was even contemplating stopping the MDP project and continuing it next year, as I doubted our groups ability to produce a satisfactory result. Luckily, I had nice discussions with Karin van Tongeren as well as the project supervisor Martijn Wisse who encouraged me to continue, something which I am glad I did.

I am sure that I have learned a lot about myself and my role in a project. Mostly that I should not fall into despair when a project is not going exactly how I wish it to go. I hope that I can take these learned lessons into my future projects and job.

6.4 Kenny

6.4.1 Learning Goals

- Specific: Gain more experience in switching between programmable tasks and non-programmable tasks within a project.
- Measurable: Spend time on both types of tasks equally.
- Achievable: Achievable, which is also depending on the time management, assuming that programming and working on the report require an equal amount of time.
- Relevant: Yes, for decision skills on whether which tasks have more priority.
- Time-bound: Have a working robot system before the test session and completed reports before the given deadlines. Other words complete the project deliverables on time.

6.4.2 Peer Feedback Received and Peer Feedback Given

The received peer feedback, gave a lot of insight about my personal effort into the project. I also agree with the received peer feedback, which indicates that I spend a lot of time on the project and always try to be involved into the discussions and updates. I consistently focus on the deliverables for the project. Also, a point I need to work on, is to equally divide my effort between the report and the software. I need to watch out that I do not slack with my software skills while focusing priorities only on the reports.

For the peer feedback that I gave to my other project members, was there little to no contradiction. The characteristics that I acknowledged as their strengths, were also the things they were aware off themselves. And for the things that could be improved by the project members, there were not always things to point out. This is probably because of the fact that I was already satisfied by the working dynamic of the team, once we were nearing the test sessions.

6.4.3 Transferable Skills

- Teamwork: At the beginning of the project, we noticed that it was hard getting of the ground. But I saw big improvements in my own teamwork skills as my communication skills have grown during this project by keeping my project members up to date about my work. Also, listening to other peoples ideas was also something that I learned to do more often during this project.
- Leadership: Regarding leadership, I unexpectedly felt like the steering wheel was sometimes in my lap. This also showed that I improved when it comes to leadership, especially in situations when deadlines for the project were coming up. Then I started communicating more with the team in order to focus together on the same deliverables.
- Entrepreneurial thinking: In entrepreneurial thinking it was quite hard to notice my improvements. However, I did learn from other members in my team what their form of improvements were, regarding this project. The most remarkable thing was that when you implement a solution in a project, it matters the most that you come up with something that is necessary and also not approached by other project groups.
- Strategic multidisciplinary problem-solving: This can be considered as a key aspect of the whole project. The remarkable improvements I made during this project within the aspects of HRI, perception, planning & navigation and motion control was mapping the actions the robot needed

to accomplish. By noticing which goals you need to implement at an early stage, it can significantly improve your efficiency.

6.4.4 Agile, Specialist Roles, and Other Team Roles

Agile project management was quite applicable during this project by breaking down the development process into smaller sprints or tasks, which becomes easier to manage and respond to changes in requirements after receiving feedback.
The things about Agile project management that I liked were the effective communication among team members, ensuring everyone is aligned with the project goals and objectives.
To improve the use of Agile, the project team can focus on enhancing certain areas. These could include strengthening the sprint planning process for deliverables and ensuring clear and well-defined success-criteria for our implemented solution.
Also every group member of the project fulfilled their corresponding roll, by working hard on their part. From my own perspective I feel like I should also have been more involved in assembling the software parts together, but circumstances lead to other deadlines which also shifted priorities from time to time. What I discovered about myself having a specific role, is that it can give a feeling of great responsibility. And also that communicating about your specific tasks is important in your team dynamics and does not necessary mean you are on your own for that role.

6.5 Stan

6.5.1 Learning Goals

At the start of this project, I had two main learning goals, which are improving my programming skills and improving my capability to work together with multidisciplinary people. These learning goals are formulated below in the SMART formulation.

- Specific: Improve my programming skills by completing an order package.
- Measurable: Track progress by the completion of the package and its complexity.
- Achievable: Dedicate at least two hours every day to coding and following online tutorials.
- Relevant: Programming skills are essential for my career growth in software development.
- Time-bound: Complete the package before the final testing moment.

This learning goal did not evolve over time and it was reached in time. The peer feedback I received about this topic was fairly positive. In buddy check 1 someone said I was handy with ROS.

- Specific: Enhance collaboration and communication skills to effectively work with multidisciplinary people in my team.
- Measurable: Increase the frequency of successful collaborative interactions with team members from different disciplines.
- Achievable: Actively listen to and consider different perspectives.
- Relevant: Building strong collaborative skills is crucial for achieving project goals and promoting innovation in a multidisciplinary work environment.
- Time-bound: Show noticeable improvement in collaborative interactions within the 8-week project by consistently applying the acquired skills and seeking feedback from team members.

The second SMART-formulated learning goal was harder to track during the project, but I would say that it is reached. In buddy check 1 I received comments that implied that my communication and cooperation skills are good, but I could take more leadership and initiative in giving people feedback.

6.5.2 Peer Feedback Received and Peer Feedback Given

The received feedback reflects my personality, which is reactive. I work hard and communicate well, but sometimes there is some initiative missing. I agree with this feedback and I think that it is not new. I think that it is important to stay close to yourself, but a slight change could improve the project dynamics. So, I learned that it is sometimes needed to step out of your comfort zone. I found that this is possible with a 'low-stress vibe' as well.

6.5.3 Transferable Skills

- Teamwork: Throughout the project, our team developed strong teamwork skills through open communication, leveraging individual strengths, leading to the successful completion of our Albert robot project
- Leadership: Over the course of the project, our leadership skills experienced significant growth. Initially, we struggled with establishing clear roles and responsibilities, which resulted in a lack of direction and inefficiencies. However, as the project progressed, we recognized the importance of effective leadership. We used the suggestions from sprint 1 to define roles and set goals.
- Entrepreneurial thinking: Initially, we focused solely on executing predefined tasks, but we quickly realised that there is a lot of freedom in this project. We encouraged idea generation by thinking outside the box and seeking alternative solutions. This is how we came up with the idea of order picking in local supermarkets instead of a distant distribution centre.
- Strategic: Throughout the project, our team experienced notable growth in our strategic skills. Initially, we were focusing on immediate tasks and short-term goals, as these goals had deadlines. However, we found out that a broader view and strategic thinking were needed to connect all the parts.

6.5.4 Agile, Specialist Roles, and Other Team Roles

We were able to apply Agile successfully because we passed all sprint reports. I liked how the project was divided into manageable sprints. The intermediate feedback allowed us to reflect on our progress. The close collaboration within our cross-functional team fostered open communication, enabling quick decision-making. I found myself being easy on deliverables. So, I could improve my skills in using Agile by being more critical of the deliverables. Some unnecessary uncertainties were present in sprint reports.

6.5.5 Other Key Reflection Conclusions

To summarize, I had two main learning goals at the start of the project: improving programming skills and enhancing collaboration with multidisciplinary team members. Both goals were formulated using the SMART framework and were successfully achieved. Peer feedback acknowledged my proficiency in programming and collaboration skills, but suggested taking more initiative in leadership and providing feedback. I recognized my reactive nature and the need to step out of my comfort zone for improved project dynamics. Transferable skills included teamwork, leadership, entrepreneurial thinking, and strategic thinking. The successful application of Agile methodologies was noted, but I aim to be more critical of deliverables for further improvement.

Conclusion

This project aimed to create a comprehensive system for Albert, enabling order picking and assistance in supermarkets. The objective was to equip the robot with the ability to handle orders from both online and in-store customers by selecting the necessary products for them. The system encompasses various capabilities.

To interact with customers, the robot utilizes an advanced voice interaction system powered by ChatGPT. This enables the robot to understand and fulfill product requests from in-store customers through voice commands. A user interface has also been implemented in order to select items for orders, meant to simulate orders coming in from online customers. During operation, the robot autonomously identifies, picks, and places products, utilizing the FlexBE system as a state machine to govern its behavior. After completing an order, the robot returns to a base station. Additionally, the robot employs sensors such as lidar, radar, and stereo cameras to detect customers and potential obstacles.

By testing the enhanced capabilities of the Albert robot in simulations and real-world scenarios, we have found that the solution can be implemented with a reasonable level of reliability. The robot has successfully demonstrated its ability to pick orders from online customers within a store environment while also providing assistance to in-store customers in a similar manner. However, it is important to acknowledge that there are still numerous capabilities and limitations that require further testing and consideration before this system can be reliably implemented in future supermarkets.

References

[1] TU Delft. *course manual.* 2023. URL: `https://brightspace.tudelft.nl/d2l/le/content/500953/viewContent/3140637/View` (visited on 05/15/2023).

[2] Ahold Delhaize. *Ahold Delhaize.* 2023. URL: `https://www.aholddelhaize.com/` (visited on 05/15/2023).

Appendix A Change Log

Date	Changes	By
18 April 2023	New version for 2022-2023	G.N. Saunders based on original by A. van der Niet
2 May 2023	Work on chapter 1 and 2	The whole team group 20
3 May 2023	Added Work Breakdown Structure	Kenny
4 May 2023	Added SWOT analysis and Ganttchart	The whole team group 20
9 May 2023	Work on chapter 2 and 3	The whole team group 20
15 May 2023	Processed feedback report 1	Stan
31 May 2023	Processed feedback report 2	Kenny and Stan
31 May 2023	Start Chapter 4	Stan
7 June 2023	Finish processing feedback	Kenny
9 June 2023	Finish Chapter 4	Henk, Kenny, Marijn, Stan
13 June 2023	Start Chapter 5	Stan
18 June 2023	Individual reflections and conclusion	The whole team group 20
19 June 2023	Finishing the report	The whole team group 20

Appendix B Project Goal

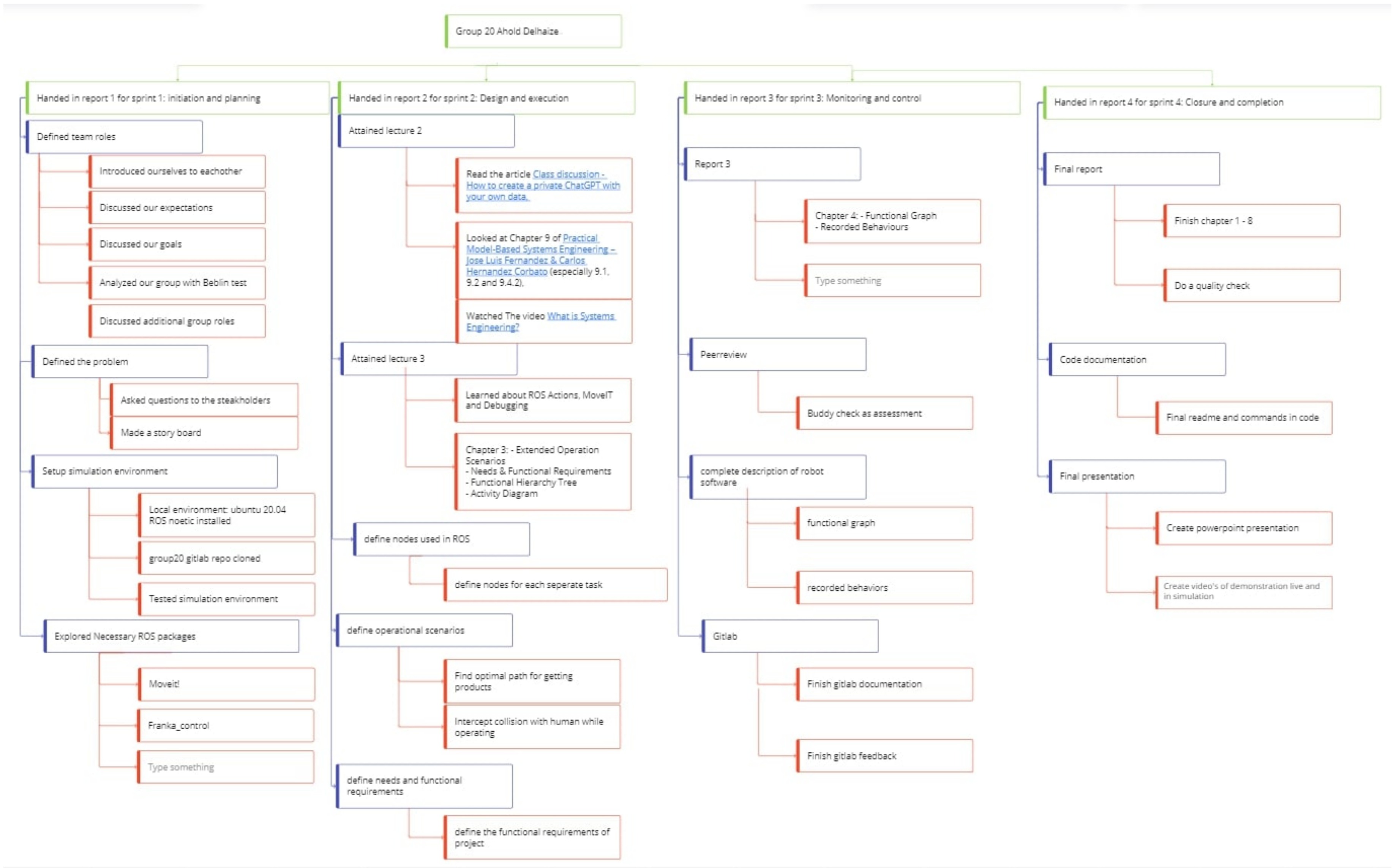
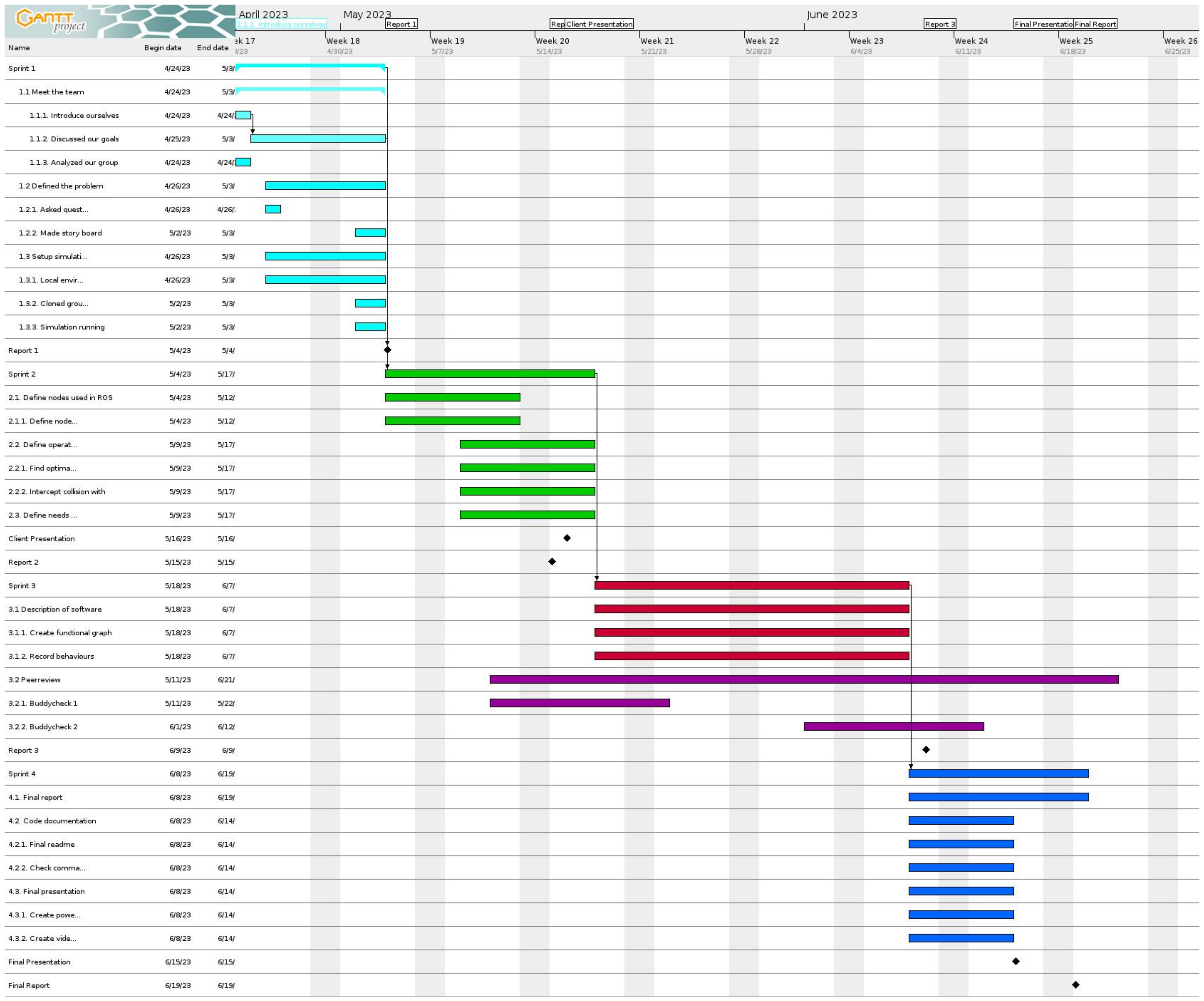

Group 20 Ahold Delhaize
Handed in report 1 for sprint 1: initiation and planning
Defined team roles
Introduced ourselves to eachother
Discussed our expectations
Discussed our goals
Analyzed our group with Beblin test
Discussed additional group roles
Defined the problem
Asked questions to the steakholders
Made a story board
Setup simulation environment
Local environment: ubuntu 20.04 ROS noetic installed
group20 gitlab repo cloned
Tested simulation environment
Explored Necessary ROS packages
Moveit!
Franka_control
Type something
Handed in report 2 for sprint 2: Design and execution
Attained lecture 2
Read the article Class discussion - How to create a private ChatGPT with your own data.
Looked at Chapter 9 of Practical Model-Based Systems Engineering - Jose Luis Fernandez & Carlos Hernandez Corbato (especially 9.1, 9.2 and 9.4.2).
Watched The video What is Systems Engineering?
Attained lecture 3
Learned about ROS Actions, MoveIT and Debugging
Chapter 3: - Extended Operation Scenarios - Needs & Functional Requirements - Functional Hierarchy Tree - Activity Diagram
define nodes used in ROS
define nodes for each seperate task
define operational scenarios
Find optimal path for getting products
Intercept collision with human while operating
define needs and functional requirements
define the functional requirements of project
Handed in report 3 for sprint 3: Monitoring and control
Report 3
Chapter 4: - Functional Graph - Recorded Behaviours
Type something
Peerreview
Buddy check as assessment
complete description of robot software
functional graph
recorded behaviors
Gitlab
Finish gitlab documentation
Finish gitlab feedback
Handed in report 4 for sprint 4: Closure and completion
Final report
Finish chapter 1 - 8
Do a quality check
Code documentation
Final readme and commands in code
Final presentation
Create powerpoint presentation
Create video's of demonstration live and in simulation

Figure B.1: Work Breakdown Structure



Name	Begin date	End date
Sprint 1	4/24/23	5/3/
1.1 Meet the team	4/24/23	5/3/
1.1.1. Introduce ourselves	4/24/23	4/24/
1.1.2. Discussed our goals	4/25/23	5/3/
1.1.3. Analyzed our group	4/24/23	4/24/
1.2 Defined the problem	4/26/23	5/3/
1.2.1. Asked quest...	4/26/23	4/26/
1.2.2. Made story board	5/2/23	5/3/
1.3 Setup simulati...	4/26/23	5/3/
1.3.1. Local envir...	4/26/23	5/3/
1.3.2. Cloned grou...	5/2/23	5/3/
1.3.3. Simulation running	5/2/23	5/3/
Report 1	5/4/23	5/4/
Sprint 2	5/4/23	5/17/
2.1. Define nodes used in ROS	5/4/23	5/12/
2.1.1. Define node...	5/4/23	5/12/
2.2. Define operat...	5/9/23	5/17/
2.2.1. Find optima...	5/9/23	5/17/
2.2.2. Intercept collision with	5/9/23	5/17/
2.3. Define needs ...	5/9/23	5/17/
Client Presentation	5/16/23	5/16/
Report 2	5/15/23	5/15/
Sprint 3	5/18/23	6/7/
3.1 Description of software	5/18/23	6/7/
3.1.1. Create functional graph	5/18/23	6/7/
3.1.2. Record behaviours	5/18/23	6/7/
3.2 Peerreview	5/11/23	6/21/
3.2.1. Buddycheck 1	5/11/23	5/22/
3.2.2. Buddycheck 2	6/1/23	6/12/
Report 3	6/9/23	6/9/
Sprint 4	6/8/23	6/19/
4.1. Final report	6/8/23	6/19/
4.2. Code documentation	6/8/23	6/14/
4.2.1. Final readme	6/8/23	6/14/
4.2.2. Check comma...	6/8/23	6/14/
4.3. Final presentation	6/8/23	6/14/
4.3.1. Create powe...	6/8/23	6/14/
4.3.2. Create vide...	6/8/23	6/14/
Final Presentation	6/15/23	6/15/
Final Report	6/19/23	6/19/

Figure B.2: Provisional ganttchart