CRANFIELD UNIVERSITY

Mikel Bueno Viso

AUTOMATED KITTING STATION
AUTOMATED AGS DISPATCH

School of Aerospace, Transport and Manufacturing
Robotics MSc

MSc Award
Academic Year: 2020 - 2021

Supervisor: Dr. Gilbert Tang
Associate Supervisor: Prof. Phil Webb, Dr. Marco Chacin

August 2021

CRANFIELD UNIVERSITY

School of Aerospace, Transport and Manufacturing
Robotics MSc

MSc Award

Academic Year: 2020 - 2021

Mikel Bueno Viso

AUTOMATED KITTING STATION
AUTOMATED AGS DISPATCH

Supervisor: Dr. Gilbert Tang
Associate Supervisor: Prof. Phil Webb, Dr. Marco Chacin
August 2021

This thesis is submitted in partial fulfilment of the requirements for the degree of Robotics MSc

ABSTRACT

Kitting is a critical process in every single industrial system which involves mass production, assembly and complex logistics, and it can be considered as a shipping and warehouse technique that has become popular with the development of more complex manufacturing and production systems. Kitting is manually done by an operator, which involves wasting time and money while undertaking a task which is not value-adding for the final product. This work presents the concept model and operations of an Automated Kitting Station, which automates the kitting process completely and allows for future integration within an Automated Production Process.

The 3D-Model, layout, process and implementation have been assessed and developed, and the Automated Kitting System and its operations have been validated through a ROS and Gazebo based simulation environment. The kitting task is done by a Robot Manipulator, which is capable of undertaking the materials pick and place tasks in real time as soon as kitting requests are done.

The results demonstrate that the manual kitting process can effectively be transformed and improved by implementing automation to it. Higher efficiency and accuracy are obtained, the material flow is better monitored and a complete control of the kitting process is obtained. The simulation shows an accurate performance of the robot and a quick and effective response to an external kitting request.

Keywords:
Robotics, Automation, Logistics, Manufacturing, Production, Simulation.

ACKNOWLEDGEMENTS

It has been an absolute privilege to work on my Individual Research Project, supervised by experienced and committed professionals. I would like to personally extend my sincere gratitude to Dr. Gilbert Tang, Dr. Marco Chacin, Prof. Phil Webb and Mr. José Angel Gonzalez Domingo for their continuous support, your dedication and keen interest on my work and progress have been mainly responsible for the results of this project.

I am extremely thankful to my family, especially *Aita eta Ama*, *Eider, Ane, Aitite eta Amama(-k), Izeko Mertxe, eta beste guztiak,* even though I am far away from home, I have always felt your support and strength coming from the Basque Country directly to my room in Stringfellow Hall. Cynthia, thank you for helping me being the person I am, I promise you all our hard work and mutual commitment will pay off!

And of course, this journey would not have been the best without the support and warmth of my dear friends in Cranfield: Pol, Dani, Marc, Xavi, Enara, Tim, Sergio, Javi, Quique and Víctor, thank you from the bottom of my heart!

And finally, *Aitite*, thanks for introducing my dad, uncle, myself and my whole family to the engineering world, I hope you are proud of your grandson wherever you are.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AGS	Aerospace General Standards
AGV	Automated Guided Vehicle
AI	Artificial Intelligence
AKS	Automated Kitting Station
AL	Assembly Line
APP	Automated Production Process
AT	Assortment Tray
CNN	Convolutional Neural Network
CV	Computer Vision
DOF	Degrees Of Freedom
ERP	Enterprise Resource Planner
JIT	Just In Time
KS	Kitting Station
LS	Labelling Station
MDU	Material Delivery Unit
ML	Machine Learning
ROS	Robot Operating System
SKU	Stock Keeping Unit
3PL	Third Party Logistics

1 INTRODUCTION

Automation is the future. Step by step, the world's commitment to the transition from old methodologies to brand new automated and efficient technologies is arising, and this brings new opportunities to front line industrial and logistics companies to improve and evolve their systems and processes.

Manufacturing and production companies are continuously seeking to transform and upgrade their industrial processes, and the application of cutting-edge Robotics and Automation technologies is becoming more and more popular among new implementations and innovations.

In this *MSc Thesis Project*, a potential improvement and transformation for a section of an industrial production process has been identified, and a solution is proposed by applying Robotics and Automation technologies.

1.1 Problem Definition

Kitting is an essential part of every single assembly/production line manufacturing process, and it involves supplying materials and parts to the operator. The kitting system affects to all other activities performed as well in the assembly line, and it is practiced as a method of materials/parts feeding among others such as continuous supply, batching and sequencing.

Briefly explained, kitting is the act of compiling multiple products into a single "kit", which is shipped either to a customer or directly to an assembly/production line or warehouse. Its implementation is justified due to its numerous benefits, which directly improve the operation in a production process:

- Kitting is usually undertaken in Kitting Stations, which are located at the beginning of production/assembly lines. Due to this attribute, space is saved in work stations and assembly lines are better organised.
- Increased warehousing efficiency: "*Search Time*" is completely removed. Using kitting as the main material feeding system reduces the time handlers and labour costs.

- The material flow is better controlled, since the authority over the shop floor is higher. Having a better control over the inventory is key to achieving an efficient workflow and minimizing quality issues.

Nonetheless, kitting is a **manual process** which is always done by an **operator**, which means improving the efficiency of any production process but at the cost of some relevant drawbacks:

- The Kitting Process is time consuming.
- The Kitting Process is a repetitive task.
- There is a chance of having human error.
- The fact of having to prepare the kit manually requires some time and effort (equal to money) for the operator which is not value adding either for the final product or the company.

The kitting process improves the flow and organisation of every single production process, yet the way it is implemented nowadays is not efficient enough, and it requires making a further step towards the adaptation to new technologies. Thus, a gap for improvement has been identified in the way kitting is implemented in production processes nowadays, and this project will focus on finding new and automated solutions to improve its efficiency and operation.

1.2 Proposed Solution

An **Automated Kitting Demonstrator** is proposed, which will be implemented and integrated within an Automated Production Process (APP), demonstrating a real transition from old-fashioned, manual and undervalued methodologies to cutting-edge, automated and efficient applications in production and manufacturing processes. The system will automate the kitting process completely, and integrate within an automated production process which will provide the **kit** directly to the operator in the assembly line.

Implementing **automation** to the Kitting Process brings a great amount of benefits:

- Time and money are saved.

- Human error is avoided.
- Higher accuracy is obtained.
- The process is completely controlled by a software, which is open to connect and communicate with external sources. The material flow is better monitored.
- A robot/computer can continuously be working. For instance, a great number of material packs can be sorted, stored and prepared to be delivered (to assembly line) during non-working hours.

The Automated Kitting Station proposed will improve the actual kitting process, and will demonstrate how Automation and Robotics can be implemented nowadays in order to upgrade manufacturing and production processes.

1.3 Main Objectives and Steps

The **aim** of the project is to design and develop an Automated Robotic Kitting System demonstrator and its operation. The system, its capabilities, operation, application and potential implementations will be validated using a ROS/Gazebo-based simulation environment.

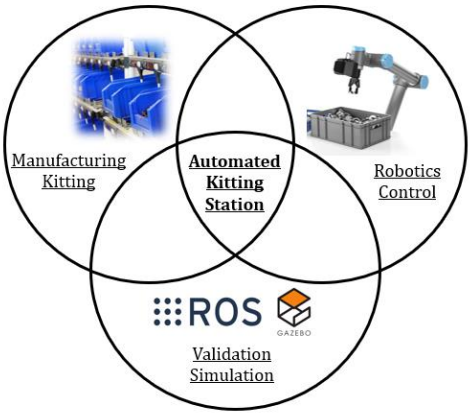


Figure 1. Automated Kitting Station. Aim of the IRP

As it can be observed in ***Figure 1***, Manufacturing & Logistics, Robotics Control & Automation and Simulation will be combined to undertake this project.

The following objectives have been defined, which will be considered as the guidebook in order to research, design, develop and implement the Automated Kitting Station:

a) Assess and research the kitting processes used in industry nowadays.
b) Research about Automated Kitting processes and methods.
c) Identify the key elements that form an Automated Kitting Process.
d) Decide which elements will form the System.
e) Design a System Concept of an Automated Kitting Demonstrator.
f) Design the 3D-Model of the Automated Kitting Station.
g) Design the Concept of Operations of the Automated Kitting Station.
h) Simulate and validate the process in a 3D-Simulation environment.
i) Assess, analyse and demonstrate the potential implementation and integration of the AKS within an Automated Production Process.

After having designed and validated the AKS, another objective will be targeted, which involves assessing, analysing and demonstrating the potential implementation and integration of the AKS within real Production Process.

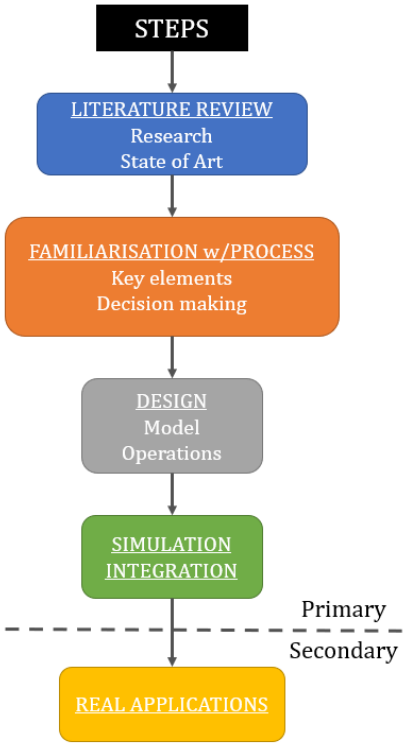


Figure 2. Main objectives and steps

The steps illustrated in ***Figure 2*** will be followed in order to meet all the objectives established throughout the project.

2 LITERATURE REVIEW

2.1 The Kitting Process

2.1.1 What is Kitting?

Kitting is a critical process in every single industrial system which involves mass production, assembly and complex logistics. It can be considered as a shipping and warehouse technique that has become popular with the development of more complex manufacturing and production systems, and it is particularly popular with businesses like Amazon, large retail chains or huge warehouses.

Different definitions of Kitting given by different logistics and industry companies are provided in the list below:

- Kitting is an inventory technique where individual, but related, products are packaged and shipped together as a single bundle [1]:
    - ~ Kitting in logistics is the act of compiling multiple products into a single “kit”, which is shipped either to a customer or directly to an assembly/production line or warehouse.
    - ~ Kiting in a warehouse is the physical act of finding multiple SKUs, bundling them into a single package, and creating a new SKU for that package before shipping.
    - ~ Kitting services refer to kitting that is performed by third-party fulfilment companies (3PL). It is also used for manufacturing processes where the third party assembles the product for the customer.
    - ~ Kitting operations is the full scope of activities involved in the product kitting process. This includes receiving the order(s), picking the products, assembling/organising the kit, packing the kit and shipping.
- Kitting is a material management strategy used to create pre-packaged and pre-labelled components. A kit includes different parts and components that are always used or assembled together. Kits simplify supply chains by taking several SKUs and combining them into one kit

(hence, a single SKU). Implementing a kitting process can move production towards JIT [2].

- Kitting or bundling means compiling individual items that are usually used or consumed together into a single SKU (a single kit or bundle). It is used by manufacturers and distributors alike, yet often in a different way and for different purposes [3].
- Kitting is the act of taking the individual parts of a product, compiling them together in a “kit”, and then delivering that kit to the production operation for assembly. With kitting services, the operators receive neatly-arranged kits containing all of the component parts required for a particular stage in the assembly process [4].

2.1.2 How is Kitting implemented?

The Kitting process involves combining a specific number of components, parts or products to a kit and assigning them a single SKU/label. This requires a few steps [1]:

1. Receive an order or material list. This new list will have a new SKU assigned, and each individual component of the list can be identified with an individual SKU.
2. Collect each individual item, and place them all together into a single tray/kit.
3. Assign the new SKU to the new kit created.
4. Ship. The kit will be shipped to the assembly line.

2.1.3 Why is Kitting used? Major benefits of Kitting

Due to the nature of the product and factory layout, the kitting process is a necessity in some industries. In an ideal world, a product would have a relatively low number of components and would be designed in such a way that makes it simple to assemble; nevertheless, this is rarely the case in industry nowadays. Therefore, kitting is essential and can be used to solve the following issues [5]:

- Lack of space: A common drawback of line stocking/production is the amount of space required to store the large material containers for each

component type next to the assembly line. Kitting removes the need to have material pallets and containers located right beside the assembly line by only providing the operator with the exact amount of material required to assemble one product/assembly at a time, thus freeing up space on the shop floor.

- Quality: Kitting can considerably contribute to enhanced quality, since the operator does not need to focus on what parts to assemble, and can instead focus all the energy and efforts on the assembly task.
- Flexibility: Kitting is considered to be a more flexible materials supply method than directly line stocking. For instance, providing materials in kits rather than large storage containers facilitates the making of product variants at any workstation.
- Materials handling: In the particular case of kitting, the total materials handling time can be seen as the sum of materials handling by the operator/assembler, materials handling during kit preparation, and internal transportation to get the parts to the assembly line. The use of kitting is believed to save time as the need for the operator to move around, search through containers and collect each component individually is eliminated.
- Learning: Kitting could aid the operator by positioning the material in a container (tray) in such a way that it acts as a work instruction (e.g., showing the assembly order).

2.1.4 Major drawbacks of Kitting

As mentioned in previous sections, Kitting is a really effective and beneficial process which brings a huge number of opportunities to improve production and manufacturing systems. Nonetheless, traditional kitting presents a variety of challenges and drawbacks [6,7]:

- Manual kitting is **time-consuming**. Manually storing, picking and packing can take days or weeks.
- **Errors** are more likely to occur. Inventory control is difficult and errors can result, even with multiple and frequent component counts.

- Processing **large orders** quickly is difficult. Manually picking, packing and organising requires a lot of resources.
- Preparing the kits requires some time and effort which is a **non-value adding** activity.
- Temporary **shortage of parts** will decrease the overall efficiency of kitting.
- When different kits contain **common parts**, an assignment of available parts to kits needs to be done.
- Spare parts might be needed at the assembly line in the case that a part in the kit is **wrong or defected**, otherwise the production will be disrupted.

2.1.5 Automated Kitting

Most of the drawbacks introduced in section 1.4 are related to the fact of undertaking the Kitting process manually. Thus, that issue presents a huge gap for improvement which involves a translation from Manual Kitting to Automated Kitting. Automating the Kitting process could bring great benefits to manufacturing/production companies by implementing a better control of the material flow and reducing workload and errors considerably. Here is a look at some of the top advantages of Automated Kitting [7]:

- Respond quickly in real-time to market/assembly demands: Rather than waiting days or weeks for workers to check inventory and then manually locate and pack order components, an automated process eliminates search time and promotes fast order preparation. If applied to Kitting before an assembly process, it would receive the order, check item availability and prepare the kit immediately.
- Save on warehouse space and racks: Kitting automation enables to operate with a smaller storage space footprint. Fewer racks, labels and lifts are needed.
- Reduce labour costs: Manual kitting is costly. In addition to employee time and effort, training time, work stations, active inventory management systems and lift equipment are needed. Automating the Kitting process enables the operator to fully focus on the task at hand.

- Increased order accuracy: Manual kitting involves having to spend the majority of the day searching for components, packing them, and creating a bill of materials, which could sometimes lead to human error. Automation platforms are designed to minimize those tasks, and therefore improve the accuracy and effectiveness of the Kitting process.
- Faster component restocking: The real-time monitoring capabilities of an Automated Kitting process enables inventory control operators to respond immediately when inventory begins to run low.
- A safer work environment: With less time spent driving, searching, lifting, and operating forklifts, employees are less likely to sustain slips, falls, back pain, and other injuries commonly experienced by highly mobile warehouse staff. Moreover, a reduction in foot traffic helps to prevent employee congestion and collisions.

In a nutshell, automation brings time, effort and cost savings, safety, efficiency, a precise monitoring of the material flow and a better organised and integrated Kitting process.

2.2 Automated Kitting Solutions

Different proved Automated Kitting Solutions applied within the industry will be introduced and described in this section. All demonstrations come from Robotics, Warehousing & Logistics, Manufacturing and Mass Production companies.

2.2.1 Robominds GmbH. Robobrain

Robominds GmbH [8] is a Robotics start-up based in Munich, developing the autonomous mobile Kitbot shown in ***Figure 3***.

Figure 3. Robominds GmbH. Autonomous Mobile Kitbot

Also called Robobrain, the product consists of an AGV + Industrial Robot system which combines AI, Robotics Control and CV to perform both simple and complex Human-Robot Collaboration tasks, being Automated Kitting one of those functions.

The robot operates and undertakes the kitting tasks in the Kitting Station shown in ***Figure 4***. The station is divided in 3 main parts:

- A **MDU** located in both left and right sides of the robot. It contains simple boxes full of different components, located on different levels and always accessible for the manipulator. In this case, the design of the rack is essential in order to allow the robot to reach all the boxes.
- A **Mobile Robot**, which consists of:
  - ~ UR5 robotic arm with a multi-gripper.
  - ~ AGV, which permits the movement of the demonstrator within the workspace, thus augmenting both the reachability and flexibility.
  - ~ A box where the materials (being part of a kit) are placed.
  - ~ A small conveyor belt.
- A **conveyor belt**, which connects to the mobile robot platform's belt and works as the main transportation method for full (kits) and empty boxes. This element avoids the neediness of using a specific gripper to pick and place the boxes.

Figure 4. Robominds GmbH. Automated Kitting Station

The manipulator applies a ***Bin Picking*** methodology in order to pick the spares from the boxes. The camera mounted on the robot arm or above the station recognizes any objects to be gripped, untrained and independent of coordinates, and identifies optimal gripping points. Different types of grippers can be used, yet a multi-gripper formed by a vacuum gripper, 2-fingered gripper and mounted camera seems to be the best option.

The Robobrain station is a good example of how Automated Kitting can be implemented in a real fast-paced industrial process, since it perfectly meets the requirements of an industrial kitting process: The material stored in a MDU (input) is processed and converted into kits (output) automatedly and ready to be carried to an assembly line.

2.2.2 KittingBot

KittingBot [9] is an Automated Kitting demonstrator developed by the University of Bonn. A mobile manipulation robotic system for autonomous kitting is proposed, formed by a Kuka Miiwa platform (omnidirectional base), a 7 DOF collaborative Kuka iiwa manipulator, cameras and distance sensors. The system is designed for collaborative kitting (in combination with manual kitting), and it is evaluated for autonomous kitting, part variant recognition and obstacle avoidance demonstrations.

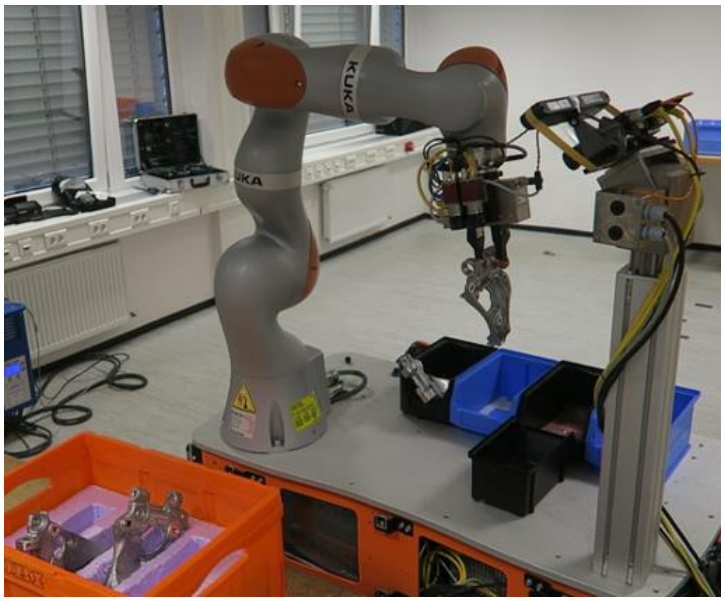


Figure 5. KittingBot. Automated Kitting Demonstrator

Figure 5 illustrates the Kitting Station. The process consists of picking the spares from the boxes located within the workspace (4 boxes, nearby and at the same level of the robot base) and placing them inside the kitting tray/box. As it can be observed, the camera is fixed on top of a vertical support, which allows an extra space/slot for a gripper since it frees up space in the end-effector. This feature is possible in this case, as the camera covers the whole workspace (the 4 boxes), yet would not be feasible for other real industrial cases (i.e., Robobrain).

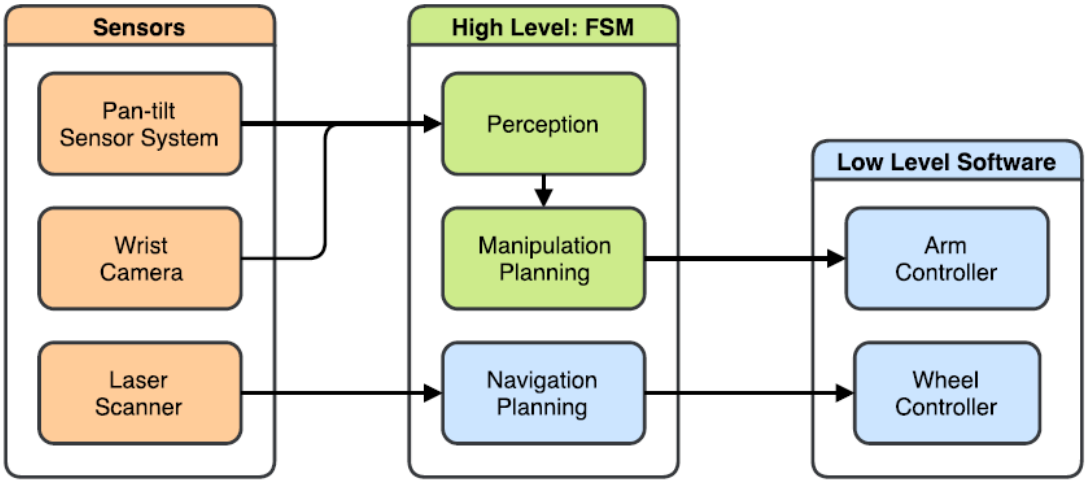


Figure 6. KittingBot. Software Architecture

The software architecture of the system can be appreciated in ***Figure 6***, and it can be divided mainly in Bin Picking (in green, own-developed), and Robotic Arm control and trajectory planning (in blue, developed by Kuka). Therefore, this enhances the relevance of ***Bin Picking applications*** for Automated Kitting purposes, and highlights the neediness of developing specific software applications such as:

- Detection and pose estimation of transport boxes.
- Part segmentation in those containers.
- Recognition of part variants.
- Grasp detection.
- Arm trajectory optimization.

KittingBot is a good example of a simple Automated Kitting application with an effective Bin Picking implementation, yet it must be further developed in order to implement it in a real industrial process:

- A real Kitting Process can handle 50+ object types and variations.
- A bigger workspace is needed.
- The camera must cover the whole workspace, if fixed.

2.2.3 Robotic General Part Feeder

The automatic parts feeding of multiple objects is an unsolved problem in the manufacturing industry. In this paper [10], the parts feeding problem is tackled by proposing a multi-robot system, which is directly linked to Automated Kitting. The system comprises three sub-components which perform bin-picking, regrasping and kitting, as shown in ***Figure 7***:

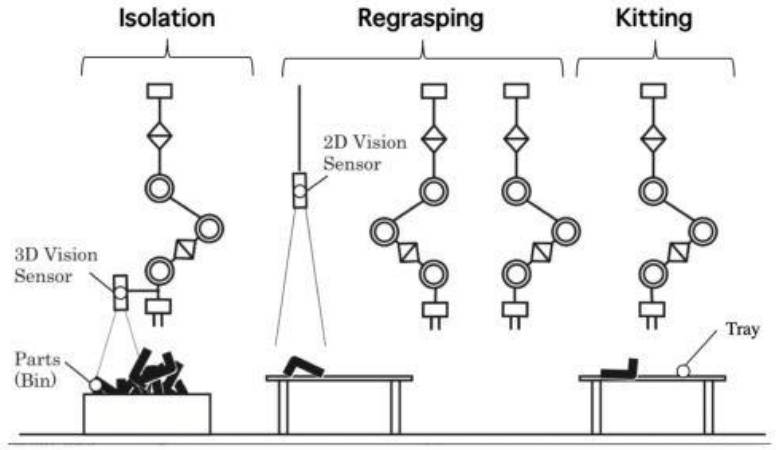


Figure 7. Parts feeding system: Bin Picking, Regrasping and Kitting

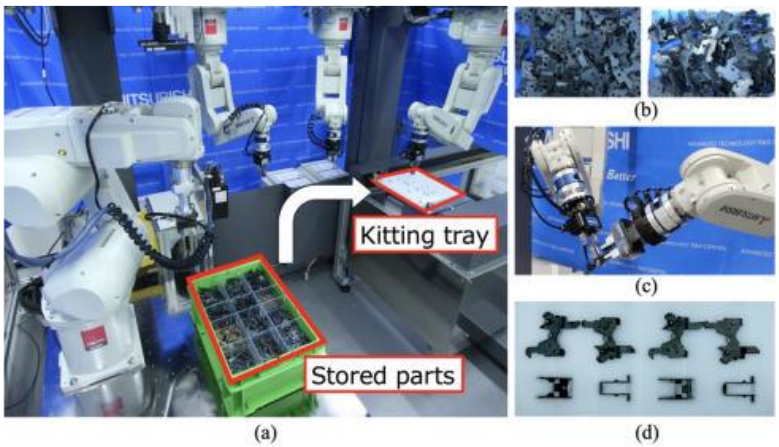


Figure 8. Parts feeding system. Workspace

Figure 8 illustrates the architecture/layout of the proposed parts feeding system. In the beginning, multiple types of parts are stored in supply bins on the left (with random poses), and the goal of the system is to kit the parts to a kitting tray.

Multiple robot arms are configured into a pipeline to play roles for each sub-component of the system:

- The first robot (left-most one) picks up the object and isolates it, by placing it on top of the table. Bin Picking using a 3D-Vision sensor is performed to undertake this task.
- Robot 2 and 3 are used to *Re-Grasp* the object placed on top of the table. Once its pose is recognized with the 2D-Camera, the object is regrasped by both robots using a specific algorithm, in order to re-allocate it to its future kitting pose.
- Once the object is placed with the correct pose, the last robot performs a simple pick and place task which places the object in its correct slot on top of the kitting tray.

The results show that the Mean Picks Per Hour (MPPH) of the proposed system is 351 with **eleven** various-shaped industrial parts, which is faster than the state-of-the-art robotic bin-picking systems.

The fact of using 4 collaborative robots could be considered as the main drawback of the system, yet it demonstrates how difficult it is to pick and place spares from random poses to specific and appropriate poses.

2.2.4 Automated Kitting Solutions. Main Conclusions

Having analysed 3 different effective and re-usable Automated Kitting solutions, some clear conclusions arise, especially about how Kitting should be automated and implemented within the AGS dispatch system:

- All systems pick the materials/spares from fixed boxes which are located within the workspace, and accessible to the robot manipulator. Therefore, there is no necessity of adding an automated process to the material storage, and this can be accomplished by designing or using a specific table or MDU.
- The Kitting process is undertaken by 5-6-7 DOF collaborative robots, which seem to be the most adequate to pick and place objects like spares or kitting materials. The presence of multi-grippers must be mentioned as

well, since, depending on the application, it could be required to design a specific end-effector with different grippers and a camera on top of it.

- **Bin Picking** is the most critical process of Automated Kitting. Kitting involves having to pick up a single object from a box/tray where a great number of components of the same type are mixed, increasing the difficulty of the task remarkably. Therefore, further review about Bin Picking fundamentals and how it is implemented for kitting must be done.
- Regarding the *Software Development* of the kitting process, it is completely feasible to implement software (for both Robot Control and Bin Picking) which is already developed and available in the market, having to nothing but adapt it to the specific industrial process and coordinate it with the ERP software.

The following ***Table 1*** compares some of the most relevant and critical features of the Automated Kitting stations introduced before:

	Robobrain	**KittingBot**	**RGPF**
Number of Robots	1	1	4
Robot model	UR5	Kuka iiwa	Mitsubishi Electric
Robot - DOF	6	7	6
Gripper type	Multi-gripper: - Vacuum - Parallel	Parallel gripper	Parallel gripper
Number of cameras	1	1	2
Camera position	End-effector	Fixed	Fixed
Perform Bin-Picking?	Yes	Yes	Yes
Valid for an Industrial Application?	Yes	No	No
Prepare kits?	Yes	No	Yes

Table 1. Comparison between different Automated Kitting solutions

2.3 Bin Picking

Bin Picking is a core problem in Computer Vision and Robotics nowadays. It involves picking and placing known objects with random poses out of a bin using a robot end-effector or gripper. As demonstrated in S*ection 2.2 (Automated Kitting Solutions)*, it can be considered as one of the most critical operations of the Kitting process, since it involves picking the AGS items from their boxes and placing them in the corresponding slot of the Kitting tray.

Different Bin Picking techniques and applications will be introduced in this section, which demonstrate how Bin Picking is used in order to fulfil different Kitting requirements.

2.3.1 Bin Picking and Placing of objects using CNN-s

In this paper [12], *M. Hanafy* and *S. Ingemarsson* propose a Bin Picking problem for different geometric materials using Convolutional Neural Networks. CNN-s are trained for multiple objects, to find the best grasping point on the component. A stereo camera is used for that specific purpose (determine the best grasping point), combined with depth calculation (calibrated with triangulation using AprilTags). Therefore, the position and orientation of the robot end-effector is calculated (output) by processing the image provided by the 3D-stereo camera (input).

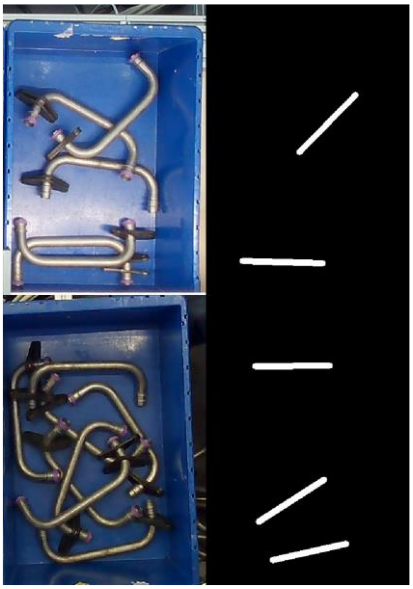


Figure 9. Bin Picking and Placing using CNN-s. Trained model example

As an example, 3200 different pictures for the *u-shaped pipe* (different materials were also tested) were used to train the model (with many different orientations, positions, object number and lighting conditions), where corresponding training targets were defined. A good example of how this process is undertaken is shown in ***Figure 9***, where all pixels which are potential picking points are set to 1, and the rest are set to 0.

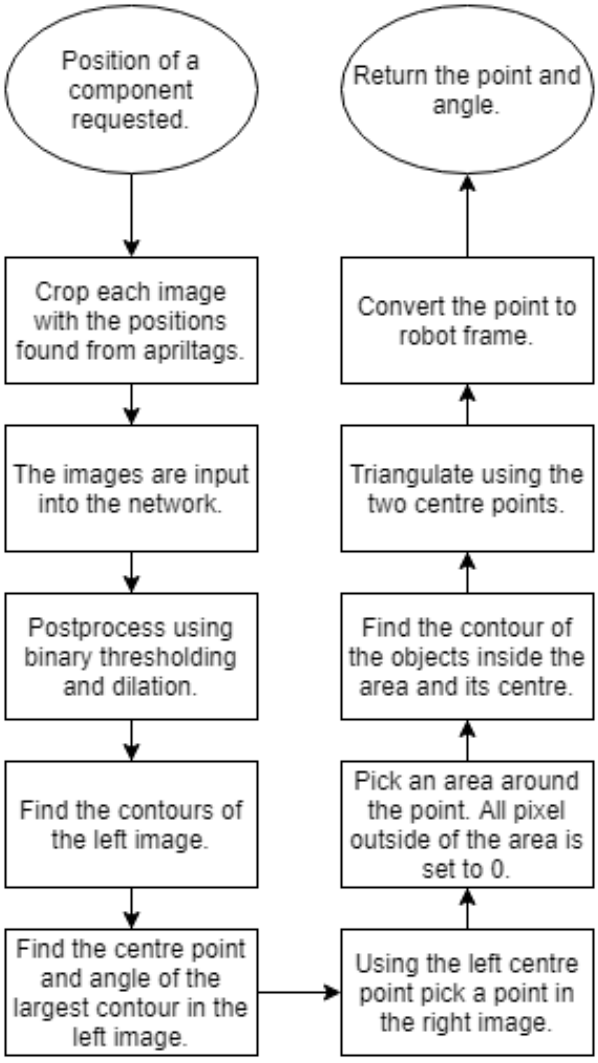


Figure 10. Bin Picking and Placing using CNN-s. Process flowchart

Figure 10 illustrates the whole image processing and gripping pose estimation process. Once the CNN model is trained, and the camera input is correctly formatted, it is used to make a prediction about the best possible gripping point, which will subsequently be calculated and converted to a robot pose by applying different CV procedures.

2.3.2 DexNet by Berkeley Automation

Berkeley Automation introduces *The Dexterity Network (Dex-Net)* [13], a research project including code, datasets and algorithms used to generate datasets of synthetic point clouds, robot parallel-gripper grasps and metrics of grasp robustness based on physics for thousands of 3D object models to train ML based methods to plan robot grasps. The main objective of Dex-Net is to develop highly reliable robot grasping across a wide variety of objects such as tools, spares, household items and industrial parts.

Different versions of Dex-Net have been uploaded over the years, which have complemented/improved the latest version of the Network:

- *Dex-Net 1.0* was designed to calculate robot grasps across datasets of over 10,000 3D mesh models.
- *Dex-Net 2.0* was created to generate training datasets to learn *Grasp Quality Convolutional Neural Network (GQ-CNN)* models which predict the probability of success of parallel-grasps based on point clouds. Dex-Net 2.1 added dynamic simulation with *pybullet* (which allowed the model to perform Bin Picking).
- *Dex-Net 3.0* added support for suction-based end-effectors.
- *Dex-Net 4.0* is the latest version, which unifies the reward metric across multiple grippers to efficiently train “ambidextrous” grasping policies that can decide which gripper is best for a specific object.

In a nutshell, Bin Picking is effectively performed due to a CNN trained on a large dataset containing more than 10,000 different objects with successful grasps. Nonetheless, the time cost of collecting physical data (for such number of objects) makes it difficult to collect clean and sufficiently large datasets to train different robots in different environments, and that is why Dex-Net focuses on training on synthetic datasets of grasps and point clouds labelled using geometric conditions related to grasp stability and *antipodality*[1]. They create a model with a discrete-time Partially Observable Markov Decision Process, which specifies states of the

[1] *Antipodal points* of a sphere are those diametrically opposite to each other.

heap, point cloud observations and rewards. The Dex-Net software is *Open Source*, and the package can be installed as a standalone Python module or a ROS package.

2.3.3 Random Bin Picking by Pickit3D

Pickit3D proposes another solution for Bin Picking [14], where a robot picks parts from bulk and loads any production line using an own-developed 3D vision system (see ***Figure 11***).

Figure 11. Pickit3D. Vision System for Bin Picking

The Bin Picking process in this case is the following:

1. The Picking 3D Vision System detects the part.
2. The best picking point of the part (without) collision is determined. This is achieved thanks to multi-pick points and tool modelling features.
3. The object is picked and placed by the manipulator.

Unlike both applications introduced before, Pickit3D is a recently developed, tested and implemented software which can perfectly handle Bin Picking operations with different objects such as:

- Axles, shafts and billets.
- Other cylinders: Crews, pipes, rods, bottles, sticks, drills, caps.
- Sheet metals.

The software supports 15 different models of collaborative robots, which means high flexibility and versatility, being these 2 essential for the design or improvement of every single fast-paced production/assembly line.

2.3.4 Bin Picking. Main Conclusions

Some conclusions arise after having analysed how Bin Picking is applied, and how different methods such as Dex-Net were created and implemented:

- AI and ML must be implemented in a Bin Picking process.
- Computer Vision techniques are essential in order to pre- and post-process the data handled by the trained models.
- The AGS Kitting process could involve 20-30 different types of objects (with different variations/size, of course). Thus, the pre-defined CNN models could be trained using the physical station, where the spares (and their poses) would be inputs and the robot pose (grasping pose) would be the output.
- Considering how challenging designing a whole Bin-Picking system is, and with the main objective of saving time and effort and focusing on the simulation and logistics of the AGS Kitting Station, an approach-based Bin Picking implementation could be an effective solution. Therefore, some assumptions would be made in order to simplify the implementation of Bin Picking process, yet it would be considered that the process is being undertaken effectively within the station (***Figure 12***) or could be implemented in the future.

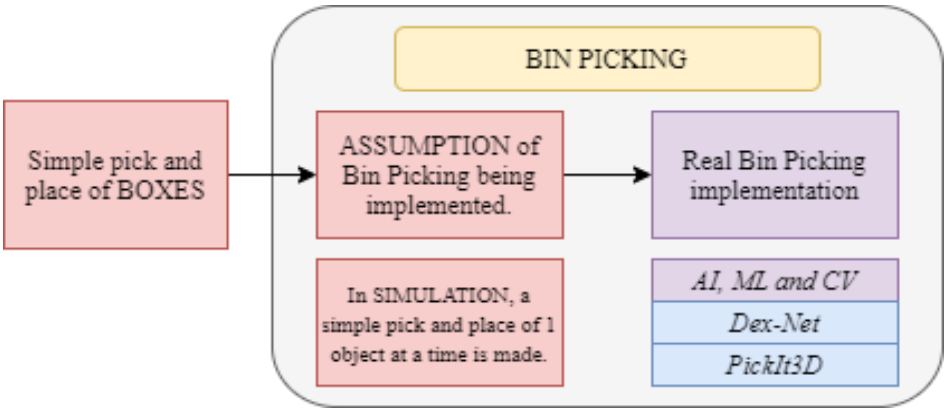


Figure 12. Approach-based Bin Picking implementation in Automated Kitting

3 METHODOLOGY

The concept design, concept of operations and implementation of the Automated Kitting Station will be introduced in this section. The design process has been built upon a well-defined and organised operation, and based on the complete automation and control of the kitting process. Therefore, some operational and functional assumptions have been made, in order to guarantee that all the requirements of the system are met and to ease the design and implementation of the simulation process which will be explained in *Section 4*.

The 3D-Model of the Automated Kitting Station is shown in ***Figure 13***:

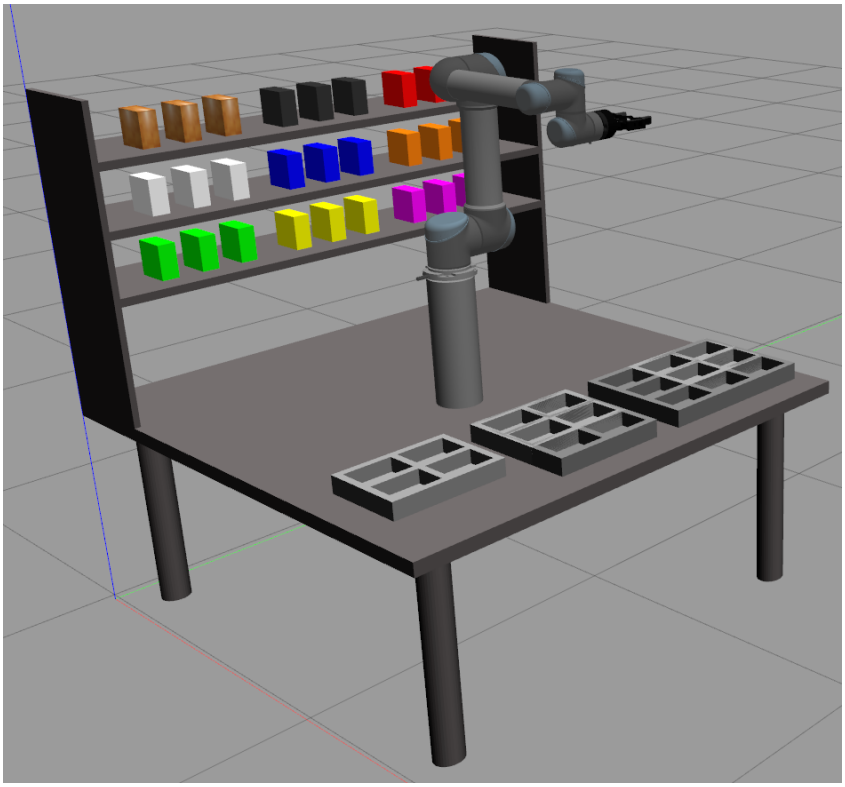


Figure 13. Automated Kitting Station. 3D Model

3.1 Concept Design

As explained in *Section 2.1 (Literature Review),* Kitting consists in creating pre-packaged and pre-labelled components (kits), which include different parts and components that are always used or assembled together. An Automated Kitting Station model diagram (top-view) is introduced in ***Figure 14***, which represents the layout of all different components that set up the system.

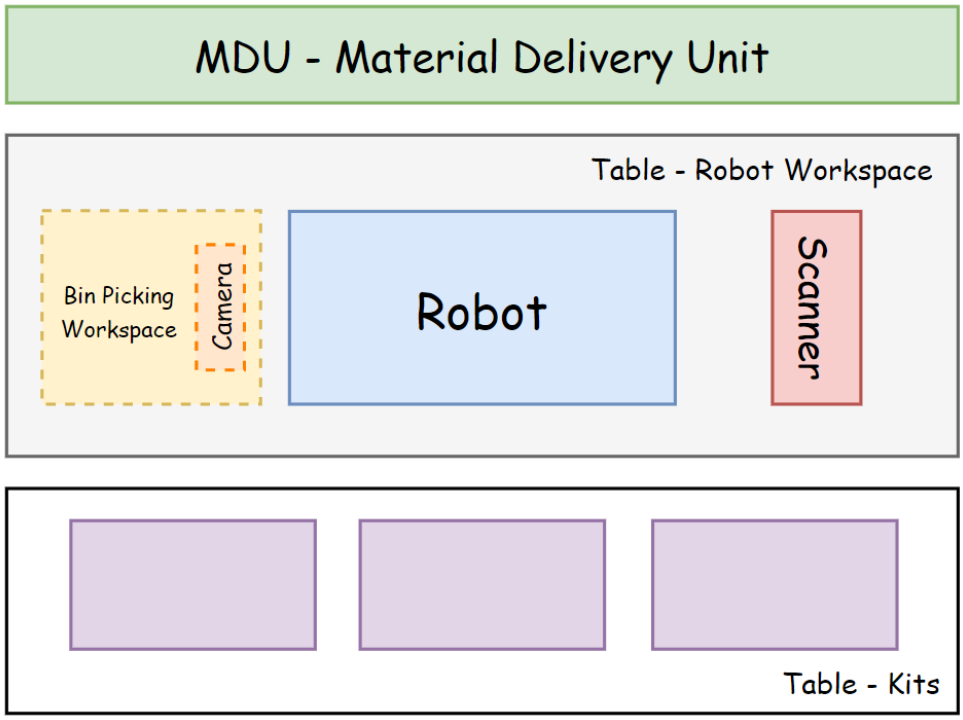


Figure 14. Automated Kitting Station. Concept Diagram

As previously explained in *Section 2.2 (Literature Review),* an Automated Kitting Station must be formed by a materials input, robot manipulator and a kitting output.

Materials Input: A MDU (Material Delivery Unit) will be used for the storage of the boxes before these are placed into kits. The MDU consists of a 3-level rack with 27 different boxes located in front of the robot manipulator, with open space in both front and rear sides in order to ease the materials pick and place operations.

Robot Manipulator: A Universal Robots UR5 robot with a Robotiq 2f-85 gripper will be used for the pick, scan and place tasks of the station. A scanner is also located in front of the robot, which main function is to scan the material-box labels in order to ensure that the right component is being picked and placed into the kit. Enough space is also saved on a side of the table for those cases where *Bin Picking* is needed. The robot would place the corresponding box on top of the workspace shown in ***Figure 14*** before undertaking the Bin Picking task.

Materials Output: Simple trays or assortment trays will be used as the main tools to generate and transport the kits. These will be located within the reach of the robot, and can be placed on top of 3 different surfaces, which election does not

affect the Automated Kitting process but it does directly affect the subsequent kit-to-assembly line transportation:

- The kits can be placed on top of the same table where the robot is located.
- The kits can be placed on top of an independent table.
- The kits can be directly placed on top of AGV-s.

This decision depends on the whole production process, and on how the Automated Kitting Station is implemented in it. In this specific case, and as shown in ***Figure 13***, the 1st option has been used in simulation.

The System Architecture demonstrates a simple but effective Kitting Station concept, with all crucial elements needed to undertake the kitting process inside an automated production environment. All requirements and design criteria for each individual component will be introduced and analysed in *Section 3.1.2 (System Architecture)* and *Section 3.1.3 (Operational Modules),* while the Kitting operation and module interconnectivity will be further explained in *Section 3.2 (Concept of Operations)*.

3.1.1 Main Assumptions

The first and main assumption of the process is that the whole process and operation is based on the **box-based** pick and place, as shown in ***Figure 15***:

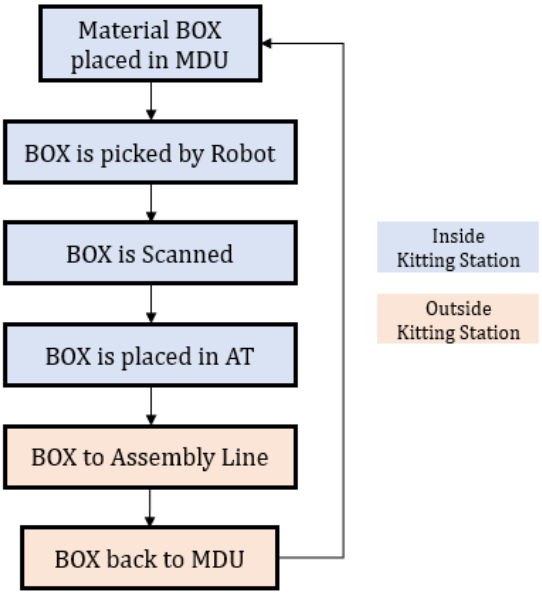


Figure 15. Box-based Kitting Process

The process is the following:

1. The box, which is full of spares, is placed on top of a shelf in the MDU.
2. If a material inside a box is required in a kit, that box will be picked, scanned and placed by the robot on an assortment tray.
3. The kit (thus, the assortment tray) will be transported to the assembly line.
4. The operator in charge of the assembly process will manually pick the exact necessary number of spares from inside each box.
5. The box will be returned to the process and placed again on top of the MDU if it still contains the required amount of spares (a threshold value will be defined), or it will be manually removed from the system if the amount of spares inside is less than the minimum required.

In order to effectively design, implement, simulate and validate the Automated Kitting Process, other 4 relevant assumptions must be considered:

- The **material flow** is monitored all the time throughout the whole automated process. Thus, the system knows all the time where the boxes are, and the exact amount of spares inside each box.
- The operation of the **scanner** is assumed. The scanner has not been implemented in simulation, yet it is assumed that the scanning and verification process is correctly conducted during the kitting process.
- The **box positions and poses** are pre-defined and known by the robot.
- The **Bin Picking** process will not be implemented, and it will be assumed that the exact amount of spares is taken out from each box by the operator in the assembly line. Nonetheless, bin picking will be considered as a future implementation and will be further explained in *Section 6 (Future Work)*.

The main objective of applying the assumptions is to focus the simulation and validation process on the *Automated Kitting*, its operation and logistics. The system proposed is a starting point for a future design and implementation of an Automated Production Process, which would consider and implement all features mentioned above and will be further explained in *Section 6 (Future Work).*

3.1.2 System Architecture

Apart from a well-defined, effective and strong operation, a proper software implementation which coordinates and moves all the data effectively is needed for such a complex logistics system. The Automated Kitting System is operated by different components that can drive and work independently, thus coordination and effective data management is essential in order to undertake the process and implement it within a real automated production process.

The System Architecture for the AKS software is proposed in ***Figure 16***, which coordinates, divides and undertakes all the tasks which will be described in the *Concept of Operations* section. The architecture is based on a **modular design**, which divides the main tasks between the different components that build the system. All the modules will be directly connected to the APP software, while this will be in charge of distributing the data in real time between modules.

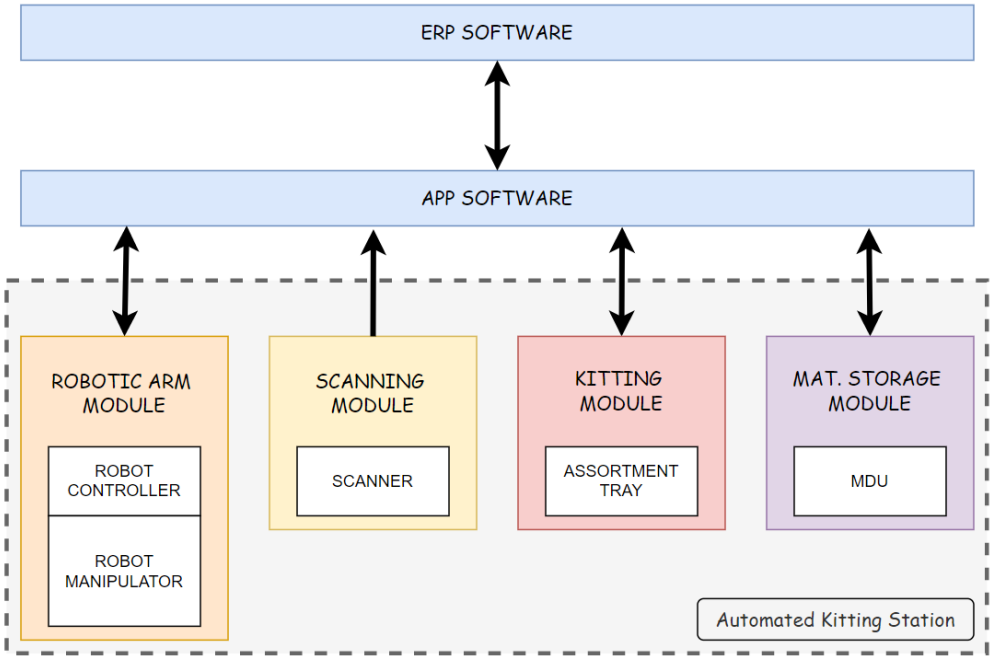


Figure 16. System Architecture

An efficient software architecture is proposed, which is capable of running different processes in parallel and relies on the APP software for all the coordination and data exchange operations. The APP software will play the role of a main program, while the rest of the modules will act as procedures or functions to be called and executed by the APP software.

3.1.3 Operational Modules

3.1.3.1 Robotic Arm Module

The Robotic Arm module will be in charge of undertaking the materials pick and place tasks, and therefore controlling the robot manipulator. It is formed by the robotic arm and its controller. The data shared by this module and the APP software is shown in ***Table 2***:

Data INPUT APP Software → Robotic Arm Module	Position of material boxes in MDU. Position of slots in Kitting Tray.
Data OUTPUT Robotic Arm Module → APP Software	Notification (Pick & place finished correctly)

Table 2. Robotic Arm Module. Data exchange

Each time a BoM (therefore, a Kitting request) is submitted, the Robotic Arm Module will receive the values of the pick and place positions from the APP software, convert them to waypoints, plan the trajectory between them and execute the process. After completing the pick and place and verifying that the operation has been made correctly, an “OK” notification signal will be sent to the APP software.

3.1.3.2 Scanning Module

The Scanning Module will be in charge of the scanning processes within the workspace. It will scan, identify and verify the material boxes picked and placed by the Robotic Arm. ***Table 3*** shows the data exchanged between the module and the APP software:

Data INPUT APP Software → Scanning Module	Activate Scanner
Data OUTPUT Scanning Module → APP Software	Material Box - Code

Table 3. Scanning Module. Data exchange

In a nutshell, the APP software will activate the Scanning Module each time a verification or scanning process is required during the process. It will request the code of the element to be scanned, and a code will be returned.

3.1.3.3 Kitting Module

The Kitting Module will be responsible for the organisation and allocation of the slots of the Assortment (Kitting) Trays. It will first verify if any of the assortment trays available meets the requirements for the ML requested (it will compare the number of boxes requested with the number of slots available in the ATs), and then will allocate the slots to all different material boxes. It will be activated by the APP software each time a ML is requested, and it will share the data shown in ***Table 4***:

<u>Data INPUT</u> APP Software → Kitting Module	Materials List List of available Assortment Trays
<u>Data OUTPUT</u> Kitting Module → APP Software	Kitting Tray – Code Slots assigned in AT for each box in the ML

Table 4. Kitting Module. Data exchange

3.1.3.4 Material Storage Module

The Material Storage Module will be in charge of the management of the material stored in the MDU. Considering the material/spares will be organised and stockpiled inside boxes within the different shelves of the MDU, ***Table 5*** is the best option to manage and share the data with the APP software:

3.1 N Y/N	3.2 N Y/N	3.3 N Y/N	3.4 N Y/N	3.5 N Y/N	3.6 N Y/N	3.7 N Y/N	3.8 N Y/N	3.9 N Y/N
M2B1	M2B2	M2B3	M3B1	M3B2	M3B3	M4B1	M4B2	M4B3
2.1 N Y/N	2.2 N Y/N	2.3 N Y/N	2.4 N Y/N	2.5 N Y/N	2.6 N Y/N	2.7 N Y/N	2.8 N Y/N	2.9 N Y/N
M2N1	M2N2	M2N3	M3N1	M3N2	M3N3	M4N1	M4N2	M4N3
1.1 N Y/N	1.2 N Y/N	1.3 N Y/N	1.4 N Y/N	1.5 N Y/N	1.6 N Y/N	1.7 N Y/N	1.8 N Y/N	1.9 N Y/N
M2W1	M2W2	M2W3	M3W1	M3W2	M3W3	M4W1	M4W2	M4W3

Table 5. Material Storage Module. Data Exchange

One of the main attributes of the system is that the material is monitored all the time, and that involves having to store and check the data in real time. The material type, the slot/position in the MDU, the amount of spares and the availability are the features that must be monitored during the process. Thus, the table consists of different columns and rows which represent:

- In **blue**, the slot position in the MDU. In the Automated Kitting Station, the MDU is a 3-level rack with 9 different slots in each shelf. In this specific

case, 3 boxes of each type of material have been included: M2, M3 and M4 Bolts; M2, M3 and M4 Nuts; and M2, M3 and M4 Washers.

- In **green**, the number of spares left inside the box.
- In **grey**, the code of the specific material box.
- In **red**, the availability in the MDU of the specific material box.

The APP software will directly access it when necessary in order to exchange the data in the following situations:

- Data will be read each time a new ML request is generated, in order to check the availability and position of each material (Module Output → APP software), and send the allocated position/pose of each box to the robot.
- Data will be written each time a new unit is placed on top of a shelf in the MDU (APP software → Module Input), or a kitting process is finished in order to update the data.

3.2 Concept of Operations

The design and development of the Concept of Operations of the Automated Kitting Station has been based on two different principles:

- The **logistics** and data management behind the automated kitting and material flow, which focused on the data exchange and coordination between all the modules described in *Section 3.1*.
- The **Kitting process**, which involves the physical task for picking, scanning and placing the boxes using a Robot Manipulator.

The operation of the Kitting Station starts when a *Kitting Request* with a Material List is made by the ERP software, and it finishes when the kit is ready to be transported to the assembly line and filled with the materials included in the ML.

3.2.1 Logistics

As shown in ***Figure 16***, the system architecture is based on a **modular design**, which divides the main tasks between the different components that build the system. All modules are directly connected to the APP (Automated Production Process) software , being this the ***brain*** of the logistics and operation behind the

process, which will be in charge of distributing the data in real time between the *operational modules* and coordinating the process.

An example of the operation of an Automated Kitting process will be provided, and the process will be explained step by step by using the diagram given in ***Figure 16*** in order to enhance the explanations, indicate how the data is managed and show how the modules are inter-connected:

1. (***Figure 17***). A new product must be assembled in the assembly/production line. Therefore, a **Kitting Request** is created in the ERP software, and the APP Software is notified. The **Material List** with all the materials is sent, including the following:
    - Material type: Code of each box.
    - Amount of spares.

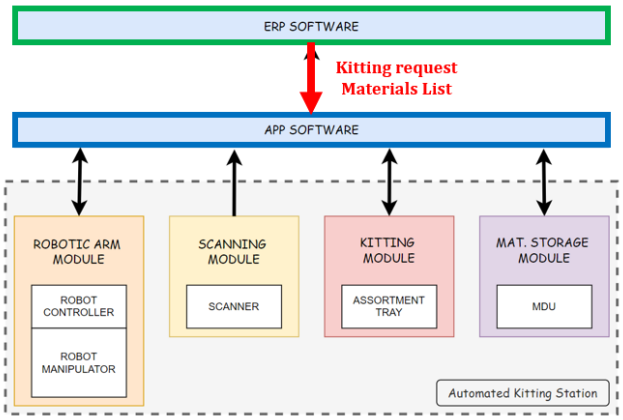


Figure 17. Concept of Operations. Logistics (I)

2. (***Figure 18***). The APP Software sends the data of the ML to both Kitting and Material Storage Modules.

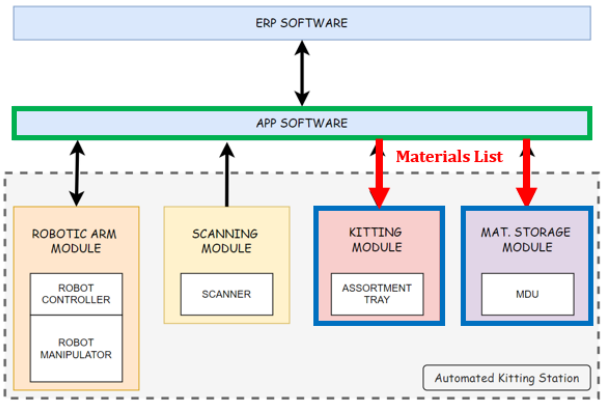


Figure 18. Concept of Operations. Logistics (II)

3. The availability of the materials included in the ML is checked, and the feasibility of undertaking the specific kitting process requested is verified:
    - ~ The Kitting Module verifies the availability of the assortment trays and assigns slots for each material in the AT selected.
    - ~ The Material Storage Module verifies that the material included in the BoM is available in the MDU and prepares the data with all the box positions in the MDU that will be used by the Robot to undertake the pick and place tasks.
4. (***Figure 19***). Both modules return the allocated slots of all materials included in the Material List in the selected Assortment Tray and MDU racks.

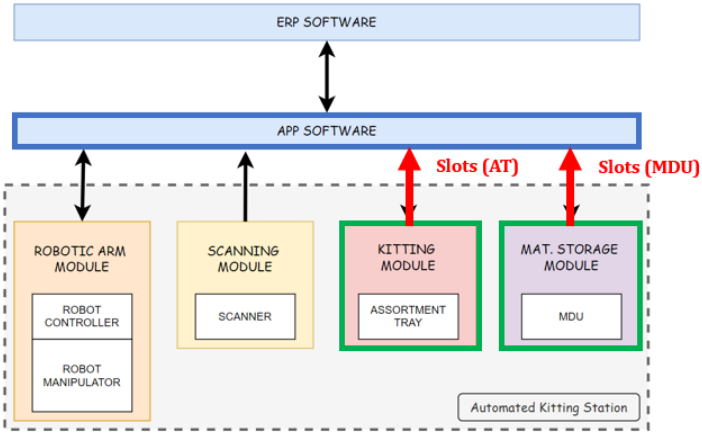


Figure 19. Concept of Operations. Logistics (III)

5. (***Figure 20***). The APP Software notifies the Robotic Arm and Scanning modules that the Kitting Process must start, and sends all the data related to pick & place poses to the Robotic Arm Module.

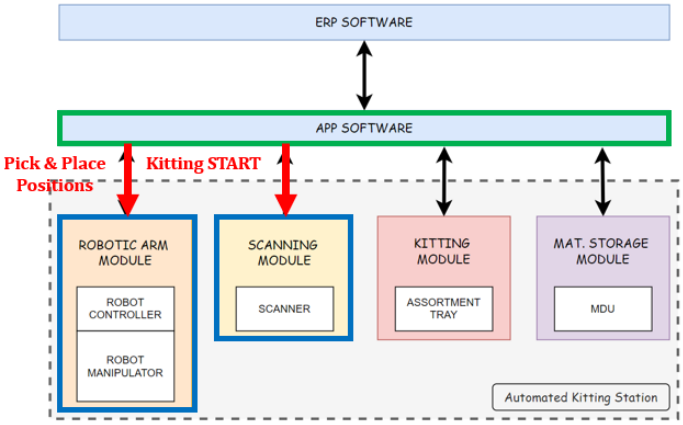


Figure 20. Concept of Operations. Logistics (IV)

6. The Robotic Arm Module and Scanning Module undertake the Kitting Process (box pick and place process), which will be further explained in *Section 3.2.2*.
7. (***Figure 21***). The Robotic Arm Module notifies the Software that the Kitting Process has finished successfully, and the Kit is ready to be transported.

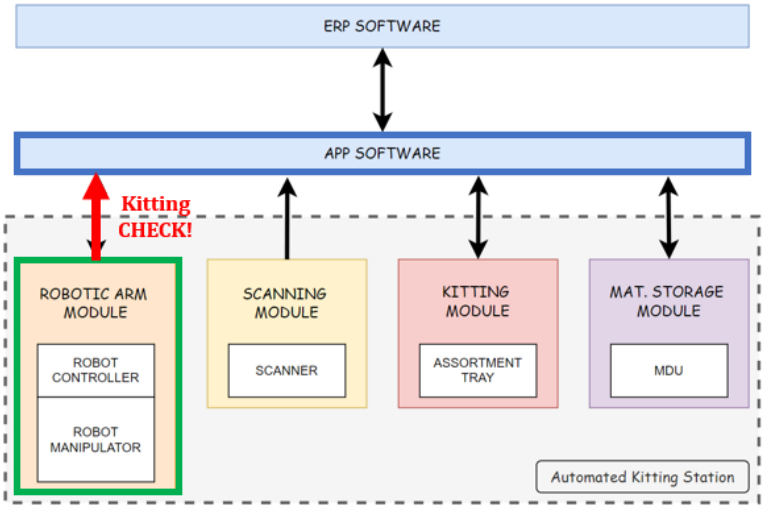


Figure 21. Concept of Operations. Logistics (V)

8. (Figure 22). The APP Software notifies the Kitting and Material Storage Modules that the process has finished, and their status must be updated:
   - ~ The Assortment Tray used will be temporarily unavailable.
   - ~ The boxes placed in the kit will be temporarily unavailable, and once those boxes return to the MDU, the amount of spares inside will be updated.

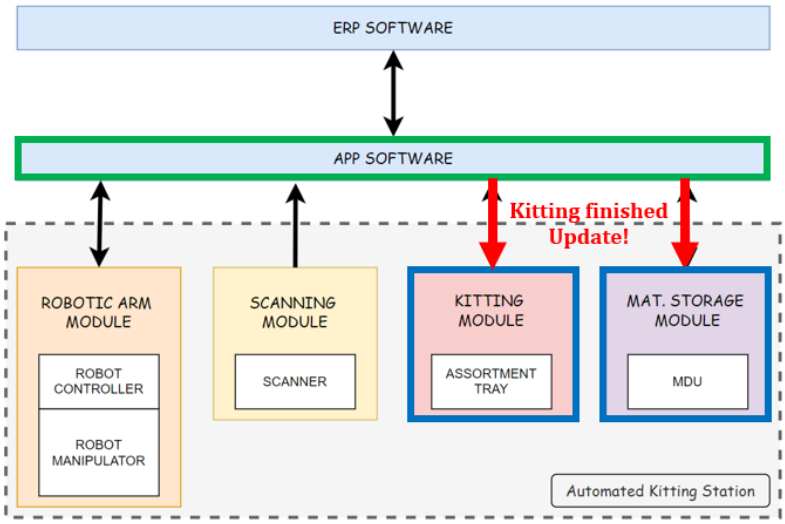


Figure 22. Concept of Operations. Logistics (VI)

3.2.2 Box-Based Kitting

The main operation principle of the Automated Kitting Station is that the kitting process is done by a robot manipulator instead of a human operator. Thus, the whole operation of the Robotic Arm and the pick & place procedure must be properly defined. ***Figure 23*** shows how the process has been designed and implemented for the Automated Kitting Station:

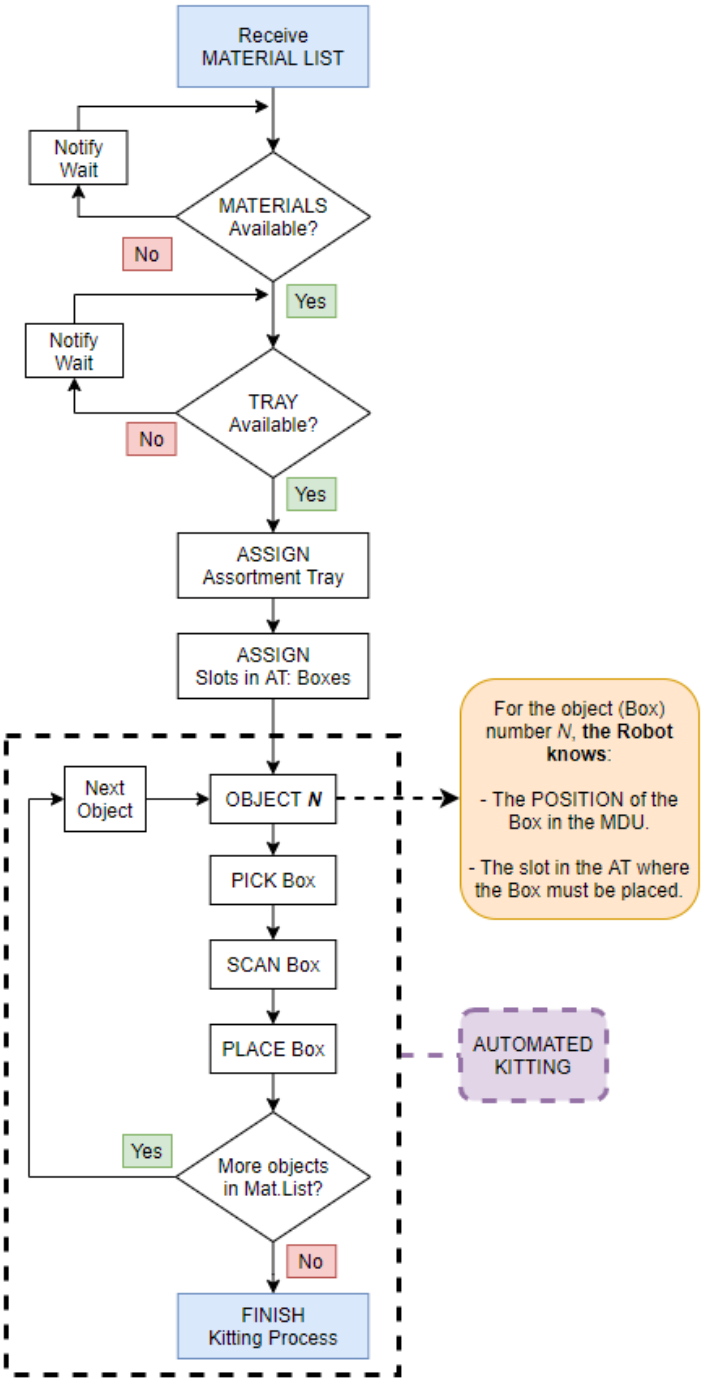


Figure 23. Concept of Operations. Automated Kitting

The first half of the process involves the verification and assignment operations described in steps 3 and 4 in *Section 3.2.1*. Thanks to this pre-operation, the robot is prepared to pick, scan and place each box, since it perfectly knows all the positions and slots which are directly transformed into waypoints and trajectories.

The Robot picks, scans and places every single box in the corresponding slot of the tray using the gripper, and the process is finished when every material included in the ML has been properly placed.

Figure 23 has been used as the main reference to generate the programming code in charge of controlling the robot. The code and its implementation will be further explained in *Section 4.2*.

3.3 Implementation. Automated Production Process

The Automated Kitting Station is integrated within an **Automated Production Process (APP)**, which concept design is provided in ***Figure 24***:

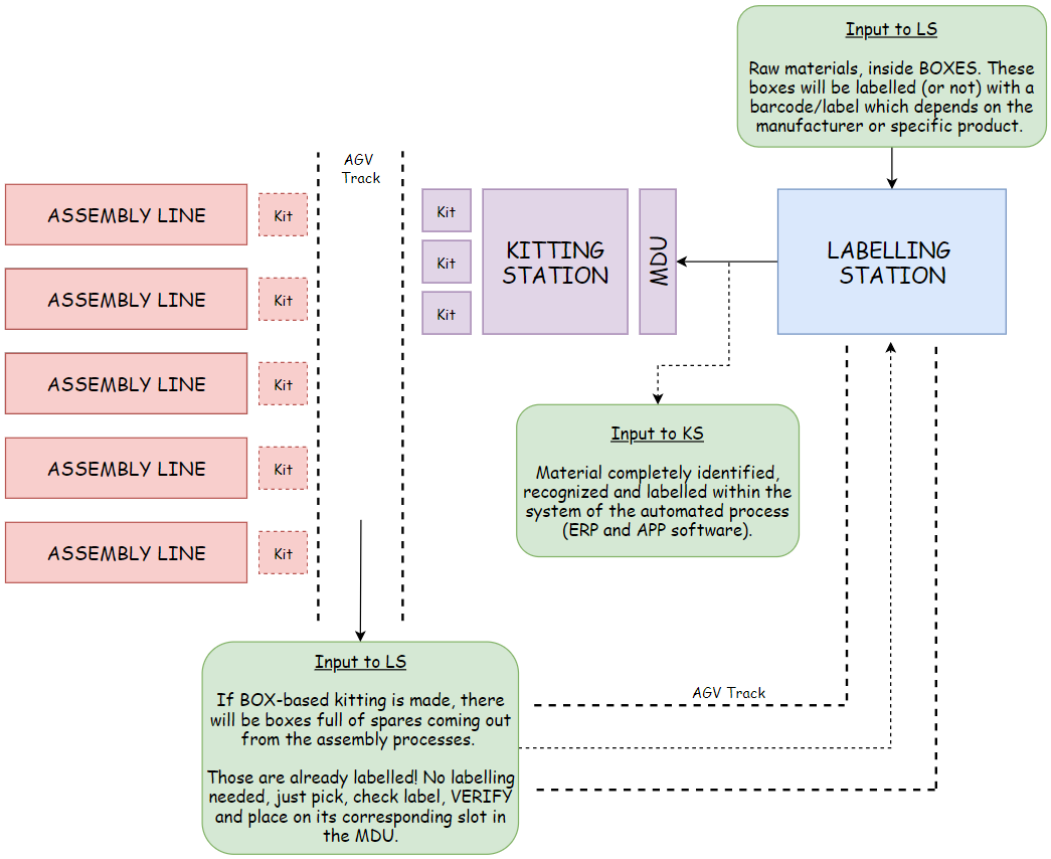


Figure 24. Automated Production Process. Concept Design

The Automated Production Process relies on the coordination between 4 different modules which operate independently and automatedly with the main objective of transforming a manual industrial process to a semi-automated production process. The whole system is formed by the following:

- Automated Kitting Station.
- Labelling Station: Automated system in charge of identifying, verifying and labelling all the material boxes that enter the system. It is formed by a vision system, a scanner, a labelling mechanism and a robotic manipulator. After labelling the boxes, its main function is to place the labelled boxes in their corresponding slots in the MDU. This process is done by the LS since both KS and LS are located next to each other.
- Automated Guided Vehicles (AGVs): The AGVs are in charge of transporting the material boxes throughout the system. There are 2 different trajectories where the AGVs operate:
    - ~ Transportation of the **kits** from the Kitting Station to their corresponding Assembly Line.
    - ~ Transportation of the **boxes** from the Assembly Lines to the Labelling Station.

  In *Section 3.1*, 3 different methods to pick and place the Assortment Trays are proposed; these could be directly placed on top of the AGVs, or placed on top of a table and then moved to the top surface of the AGVs using the Robot Manipulator or a mechanism. Due to the high complexity of designing such a mechanism and operation, it is assumed that the AGVs undertake the following processes during the Automated Production Process:
    - ~ The Assortment Trays which are in the Kitting Station are transported to the Assembly Lines as soon as the kitting operation has finished.
    - ~ The remaining boxes of the Assembly Lines are placed on top of the AGV (by the operator) as soon as the necessary amount of spares have been taken, and they are transported to the Labelling Station.

The System Architecture ***(Figure 25)*** of the whole process follows the same pattern of the Kitting Station. It is based on a modular design, and the whole process is controlled by the APP Software, which will be in charge of connecting and coordinating the different modules.

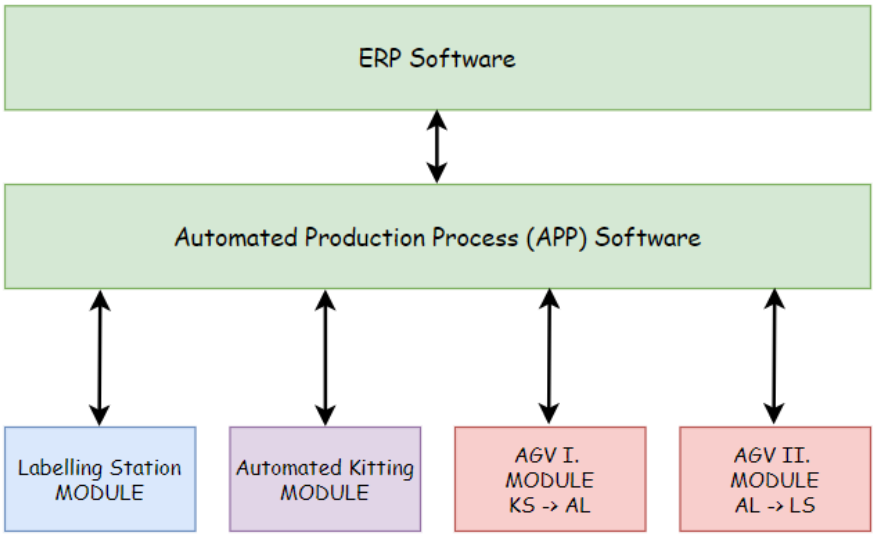


Figure 25. Automated Production Process. System Architecture

Figure 26 illustrates the whole progress and flow of a box inside the Automated Production Process from start (new Box entering the system) to finish (Box coming out from the Assembly Line and back to the Labelling Station):

1. The Labelling Station will scan and identify the new boxes entering the process:
   - ~ If the box is new, it will be properly labelled according to the labels/codes used throughout the process, and a slot in the MDU will be assigned.
   - ~ If the box comes from the assembly line, it will have a label and a MDU slot assigned to it.
2. The Robotic Arm of the LS will place the box in its corresponding slot of the MDU.
3. If the box is included in the ML of a kitting request, that box will be placed on top of an Assortment Tray in the Kitting Station.
4. The Assortment Tray will be transported to the Assembly Line.
5. In the Assembly Line, the operator will take the necessary number of spares from the box.

6. If the number of spares inside the box is still bigger than a threshold value, that box will be placed on top of another AGV, that will transport the box back to the Labelling Station.

   If the number of spares inside the box is smaller than the threshold value, the box will be taken out from the process. This involves having to request a new box of that material type for the process (it will depend on the logistics of the company and the production), and that its slot in the MDU will be temporarily free after that moment.

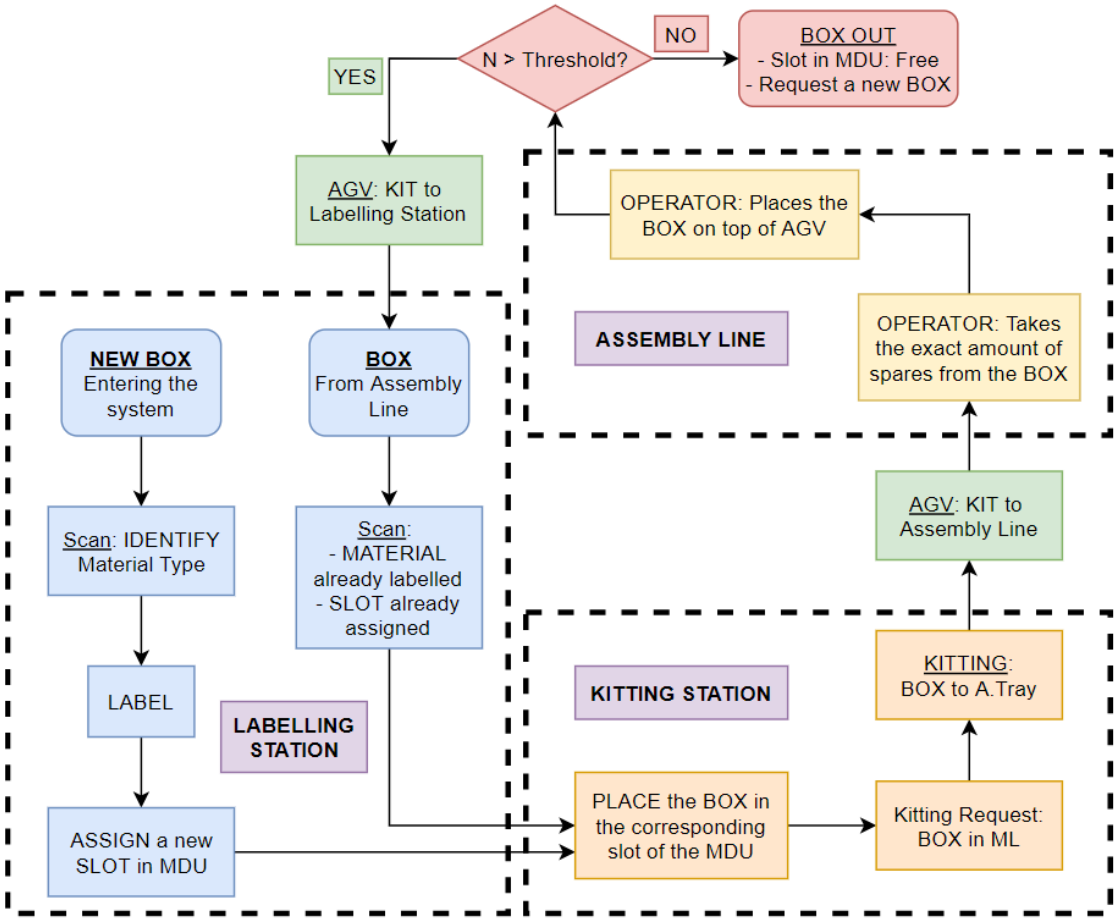


Figure 26. Life cycle of a Box in the APP

The whole Automated Production Process is based on the interactivity between modules, and the interaction between the Kitting and Labelling Stations is one of the most critical. As mentioned before, the APP Software is in charge of coordinating the different modules, and in this case, cooperation is needed when having to allocate a labelled box inside the Material Delivery Unit of the Kitting

Station, and ***Figure 27*** is a good example of how this process and interaction is done. The process is the following:

1. The Labelling Station notifies the APP Software that a box has been identified/labelled, and it sends the code.
2. The APP Software notifies the Material Storage Module that a new box has to be placed on the MDU. The box code is sent.
3. The Material Storage Module returns the allocated MDU slot of that specific box.
4. The APP Software notifies the Labelling Station to place the box on the slot returned by the Material Storage Module.
5. The Robotic Arm of the Labelling Station undertakes the pick and place task. The box is placed in the MDU.
6. Finally, the APP Software notifies the Material Storage Module that the task has finished and ***Table 5*** is updated.

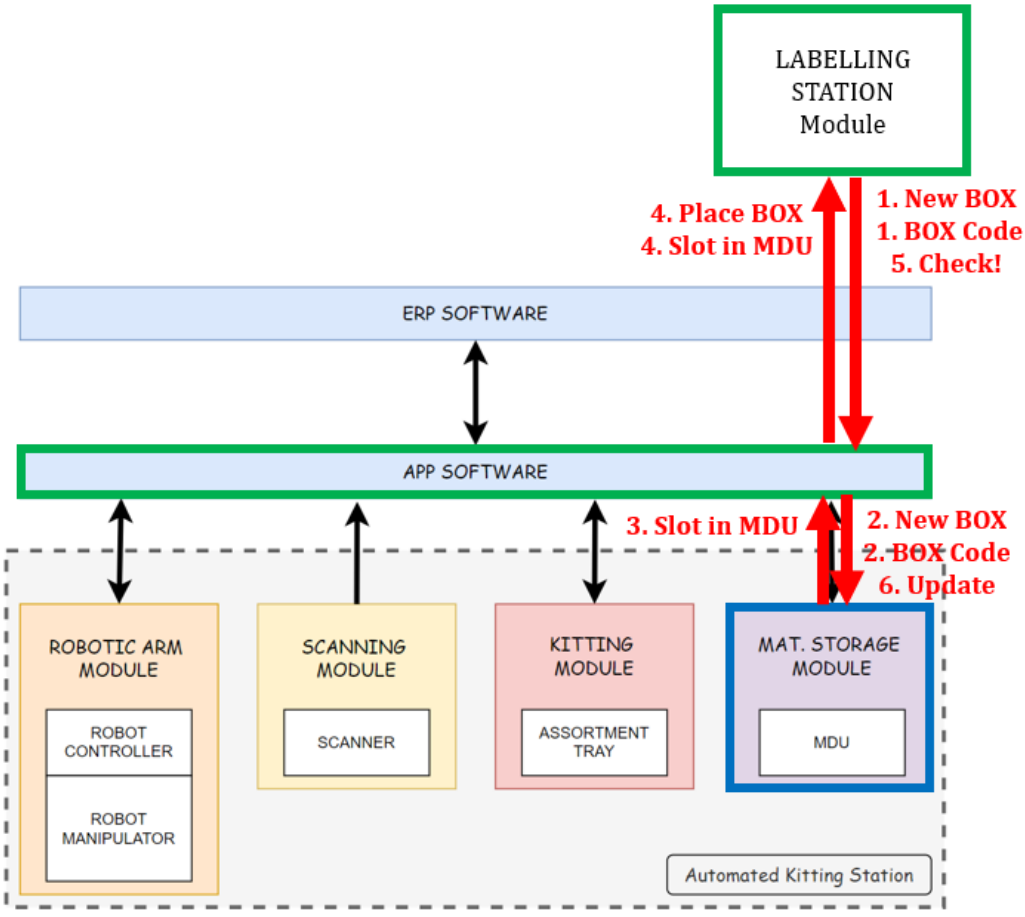


Figure 27. Interaction between Labelling and Kitting Stations

4 SIMULATION

Apart from designing and developing a concept model and process for the Automated Kitting Station, the system and its operation have been validated through simulation.

In this section, the software used for simulation will be introduced. Its features, environment and functionalities will be explained, and all the implementation process will be described step by step. The programming code used to control and operate the station will be analysed as well, and the simulation results will be given as evidence of its performance.

4.1 ROS & Gazebo Environment

Ubuntu 18.04 with **ROS Melodic** is the Operating System used to undertake the system simulation and validation tasks. ***Table 6*** below shows the technical specifications of the system used to develop and run the simulation:

Element	Specifications
Operating System	Ubuntu 18.04
ROS Version	Melodic
PC Laptop	MSI GL65 9SDK
CPU	Intel® Core™ i7-9750H
GPU	NVIDIA GeForce GTX 1660 Ti
RAM	16GB
Memory (Ubuntu Partition)	100GB
Simulation Software	Gazebo

Table 6. Simulation. System Specifications

4.1.1 What is ROS?

ROS (***Robot Operating System***) is not an actual operating system, but a framework and set of tools that provide functionality of an operating system on a heterogeneous computer cluster. Its usefulness is not limited to robots, but the majority of tools are focused on working with peripheral hardware [15].

ROS provides functionality for hardware abstraction, device drivers, tools for testing and visualization, communication between processes over multiple

machines, and much more. Its key feature is the way the software is run and the way it communicates, allowing the user to design complex software which can connect a network of processes (called **nodes**) with a central hub. Nodes can be run on multiple devices or modules, and they connect to that hub in various ways.

The main features and functionalities of ROS are [16]:

- Communications Infrastructure: At the lowest level, ROS offers a message passing interface that provides inter-process communication and is commonly referred to as a middleware. The ROS middleware provides these facilities:
  - ~ Publish/subscribe anonymous message passing.
  - ~ Recording and playback of **messages**.
  - ~ Request/response remote procedure calls.
  - ~ Distributed parameter system.
- Message passing: A communication system is often one of the first needs to arise when implementing a new robot application. ROS's built-in and well-tested messaging system saves the user's time by managing the details of communication between distributed nodes via the anonymous **publish/subscribe mechanism**.
- Robot-Specific features: In addition to the core middleware components, ROS provides common robot-specific libraries and tools that will get any robot up and running quickly. Here are just a few of the robot-specific capabilities that ROS provides:
  - ~ Standard Message Definitions for Robots.
  - ~ Robot Geometry Library.
  - ~ Robot Description Language.
  - ~ Preemptable Remote Procedure Calls.
  - ~ Diagnostics.
  - ~ Pose Estimation.
  - ~ Localization
  - ~ Mapping
  - ~ Navigation

- Tools: One of the strongest features of ROS is the powerful development toolset. These tools support introspecting, debugging, plotting, and visualizing (thus, **simulating**) the state of the system being developed. The underlying publish/subscribe mechanism allows you to spontaneously introspect the data flowing through the system, making it easy to comprehend and debug issues as they occur.

In a nutshell, ROS is an open-source framework that helps researchers and developers build and reuse code between robotics applications, and its multiple tools make it useful to simulate new models and operations realistically. It allows to reuse, install and implement technologies which have already been tested.

4.1.2 Simulation Environment

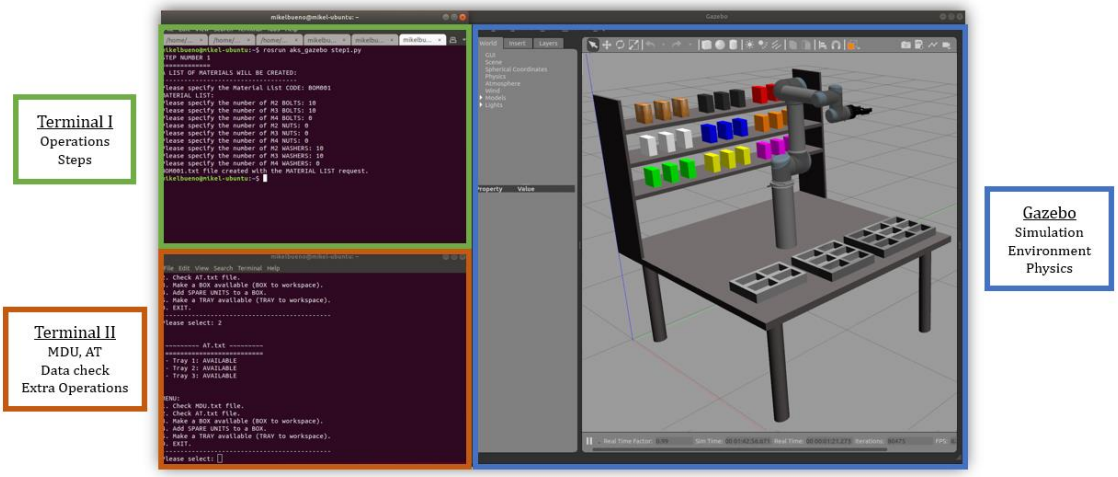


Figure 28. Simulation Environment

The Simulation Environment used to implement and execute the simulation process is shown in ***Figure 28***, and it is mainly formed by 2 different interfaces:

- Gazebo is the simulation software used to simulate the Automated Kitting Process. The workspace, 3D models, physics, Robot movement and operations are simulated and shown using this ROS Tool.
- The Terminal, which is used to execute all the programs (functions and procedures) that control the whole system in ROS. As mentioned in the *Methodology Section*, the Automated Kitting System needs an APP Software which coordinates all the modules and executes all the

operations, and that APP Software will be simulated by manually calling all the functions and procedures using the Terminal.

4.1.2.1 Gazebo Simulator

Gazebo [17] ***(Figure 29)*** is a 3D dynamic simulator with the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. While similar to game engines, Gazebo offers physics simulation at a much higher degree of fidelity, a suite of sensors, and interfaces for both users and programs.

Typical uses of Gazebo include:

- Testing Robotics algorithms and applications.
- Designing Robots and robotic processes.
- Performing regression testing with realistic scenarios.

A few key features of Gazebo include:

- Multiple physics engines.
- A rich library of robot models and environments.
- A wide variety of sensors.
- Convenient programmatic and graphical interfaces.

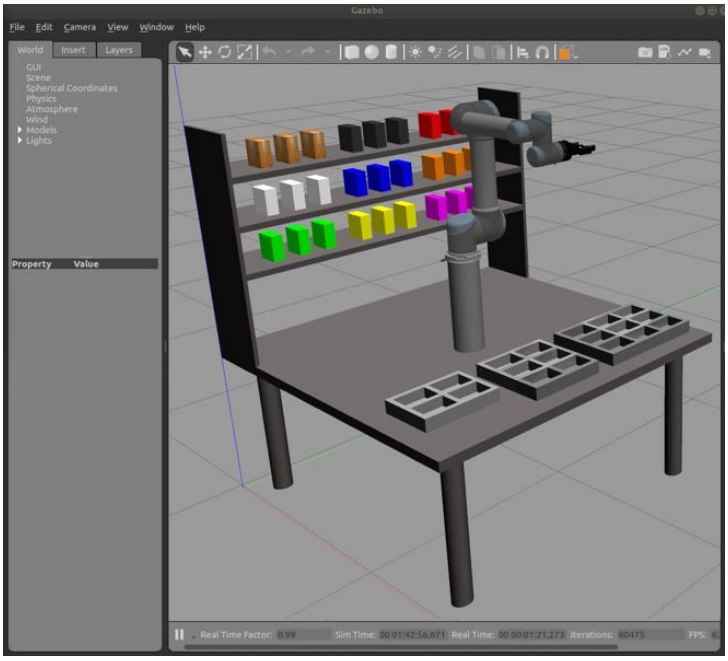


Figure 29. Gazebo Simulation Environment

4.2 Implementation Process

The whole software implementation process will be covered in this section. ROS is a really powerful tool for Robotics applications, yet it requires having to undergo through an accurate and complex process of implementation, installation and execution of different packages, plugins and features. Thus, the process will be explained step by step, by introducing all the packages, functionalities and elements included in each stage.

4.2.1 Create a ROS package and Workspace

After having installed ROS Melodic following the instructions in *ROS Wiki* [18], the ***aks_gazebo*** package has been created, which contains all the data and files related to the Automated Kitting Station, its workspace, 3D-Models, simulation procedure and programming code.

4.2.1.1 3D-Models

The 3D-Models of the workspace, boxes and Assortment Trays are created using *SolidWorks*. Nevertheless, Gazebo is familiar only with certain file formats, thus some conversion work is needed in order to import those models to the simulation environment:

1. 3D-Models are created in *SolidWorks*, and exported as ***.stl*** files.
2. *Meshlab* software is used to convert the ***.stl*** files to ***.dae*** files.
3. *Blender* software is used to open the ***.dae*** files and configure features as mass, inertia, colour or size.

The .***dae*** *(Digital Asset Exchange)* file is a 3D interchange file used for exchanging digital assets between a variety of graphics programs. It usually contains an image, textures or a 3D-Model. It is based on a *COLLADA XML* schema, which is supported in Gazebo.

4.2.1.2 Gazebo *World* and *Launch* files

Once the 3D-Models are ready to be included in Gazebo, some steps must be followed in order to spawn these objects into the Gazebo environment:

1. The ***aks_workspace.world*** file has been created. This file consists of an empty *Gazebo World* file, with all the 3D-Models included in it.
2. The ***aks_workspace.launch*** file is created, which function is to launch the Gazebo simulator with the *aks_workspace.world* workspace in it.
3. *aks_workspace.launch* file is executed using the *roslaunch* command, and the simulation environment shown in ***Figure 30*** is spawned.

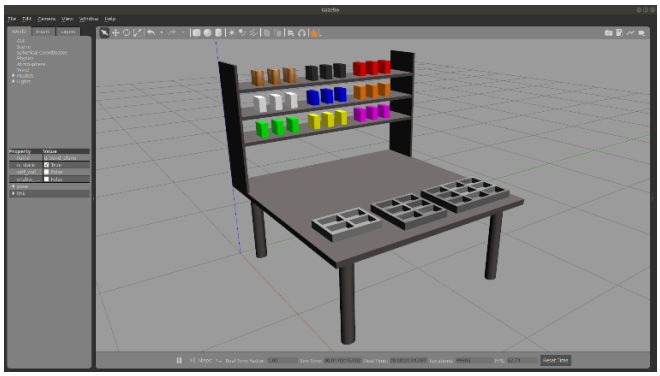


Figure 30. Gazebo World. AKS Workspace

4.2.1.3 Software Architecture

Figure 31 illustrates how the Software Architecture looks like after implementing this 1st step:

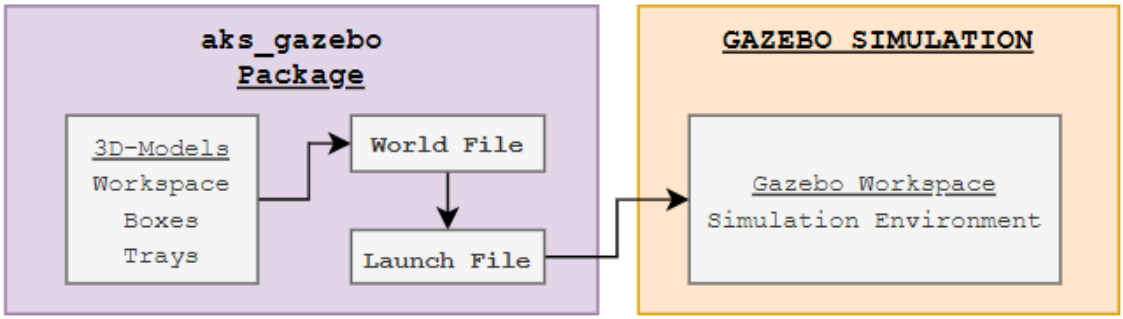


Figure 31. Software Architecture (I)

4.2.2 Import the Robot and Gripper 3D-Models

The pick and place task is done by a UR5 Robot with a Robotiq gripper attached to its end-effector. Both elements have their own ROS package, which made the implementation step simpler and more effective.

4.2.2.1 UR5 Robot and Robotiq 2f-85 Gripper

The ***universal_robot*** package [19] has been installed and added to the ROS Workspace in order to import the UR5 Robot to the AKS Gazebo workspace. The

package is part of the *ROS Industrial* repository, and contains all the 3D-Models, physical features, kinematics and controllers for the UR Robots.

The ***robotiq*** package [20] has been installed and added to the ROS Workspace in order to import the Robotiq 2f-85 Gripper to the AKS Gazebo workspace. The package is part of the *ROS Industrial* repository as well, and it contains all the features and characteristics to import the gripper model to the simulation environment.

The 3D-Models of both the Robot and the Gripper are given in the ***.urdf format***. The Unified Robotic Description Format (URDF) is an XML file format used in ROS to describe all elements of a robot.

After having cloned both repositories, all the data is ready to be imported to Gazebo. First of all, a new ***.urdf*** file is created, where both the Robot and Gripper models are merged. This new file refers to both *.urdf* files (3D-Models) which are located in the *universal_robot* and *robotiq* packages. Subsequently, a ***spawn_robot.launch*** file is created in the *aks_gazebo* package, which main objective is to spawn the new *.urdf* model into the *Gazebo World*. The *.launch* file contains the *.urdf* file created before, all the **controllers** which manage the performance of both the Robot and Gripper, and the (x,y,z) pose where the 3D-Model is spawned.

4.2.2.2 Software Architecture

Figure 32 illustrates how the Software Architecture looks like after implementing this 2nd step:

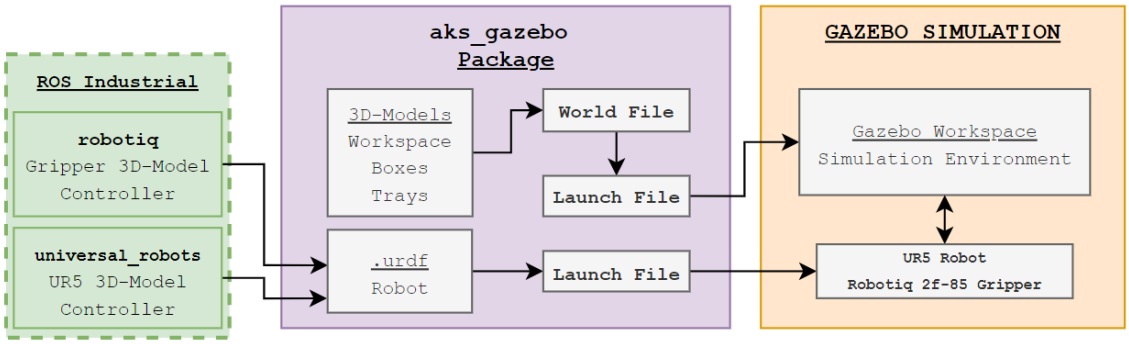


Figure 32. Software Architecture (II)

4.2.3 Create a MoveIt! package for Robot Programming

MoveIt! is a set of tools for mobile manipulation in ROS. The main *MoveIt!* web page [21] contains documentation, tutorials, and installation instructions as well as example demonstrations with several robotic arms (or robots) that use *MoveIt!* for manipulation tasks, such as grasping, picking and placing, or simple motion planning with inverse kinematics.

4.2.3.1 *MoveIt!* package

Following the instructions and using the *MoveIt! Setup Assistant* (***Figure 33***), the ***aks_moveit*** package has been created in the ROS workspace. The main function of this package is to control the operation of both the Robot and Gripper, and to generate the waypoints and trajectories that are used in simulation. *MoveIt!* directly subscribes and publishes to and from the nodes and messages which were previously generated when the ***.urdf*** file was spawned (*Section 4.2.2.1*). Once this package is properly installed, the robot operation (pick and place tasks) can be programmed using the *Python Code* which will be introduced in *Section 4.2.5* and described in *Section 4.3*.

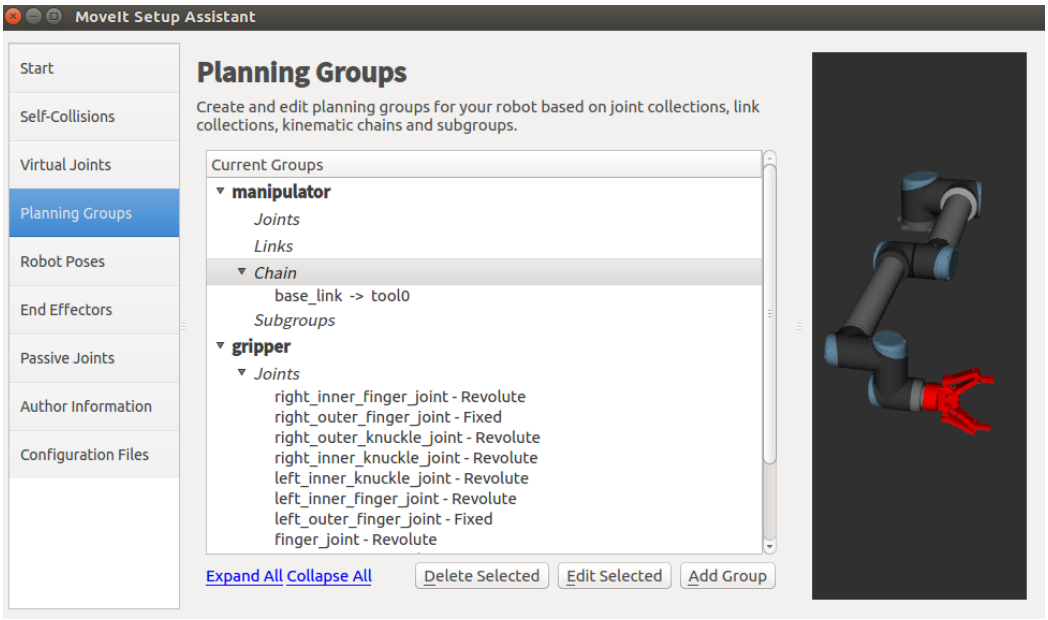


Figure 33. MoveIt! Setup Assistant

4.2.3.2 Software Architecture

Figure 34 illustrates how the Software Architecture looks like after implementing this 3rd step:

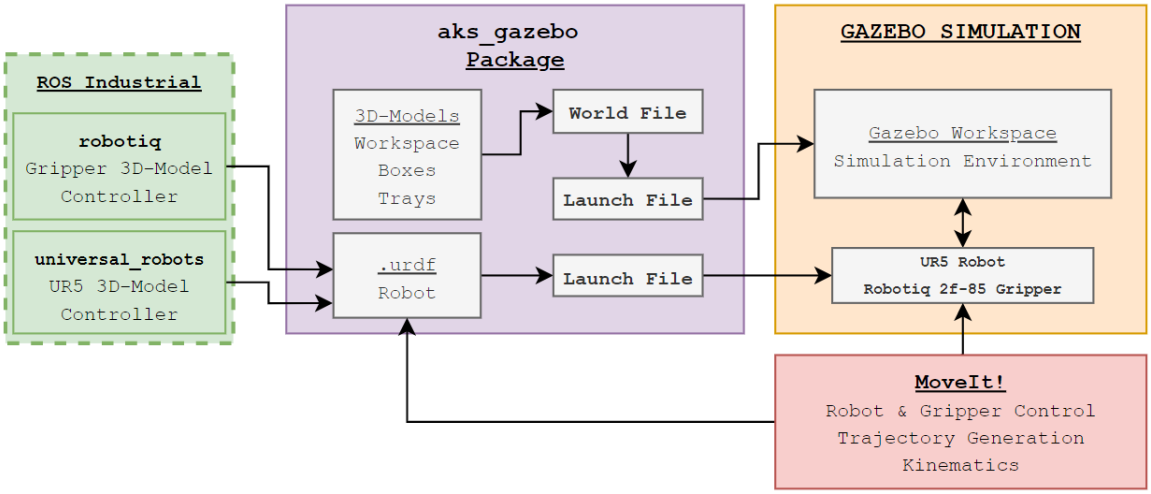


Figure 34. Software Architecture (III)

4.2.4 Improve grasping efficiency

Gazebo is a powerful simulation tool which simulates different Robotic applications precisely. Nonetheless, it requires a great amount of parameters and variables to be defined in order to obtain an acceptable and realistic simulation, and this can often result in unsuccessful operations. This was the case of the pick and place operations in the Automated Kitting Station, since the robot was not able to pick the boxes properly.

Thus, and after some research on how this issue could be fixed, the *Gazebo grasp fix plugin* [22] developed by *Jennifer Buehler* was found and implemented.

4.2.4.1 The Gazebo grasp fix plugin

Especially in older Gazebo versions, the robot may display strange behaviour when grasping an object: It may wobble, explode, or fly off into space. This is because the physics engine is not optimized for grasping yet. The plugin provided in the package ***gazebo_grasp_plugin*** helps to overcome such issues.

Therefore, the ***gazebo-pkgs*** package was cloned, and the *Gazebo plugin* in charge of controlling the grasping task was added to the ***.urdf*** file created in *Step 4.2.2.1*. Briefly explained, the plugin fixes the box which is grasped to the robot hand to avoid problems with physics engines and to help the object staying in the gripper without slipping out.

4.2.4.2 Software Architecture

Figure 35 illustrates how the Software Architecture looks like after implementing this 4th step:

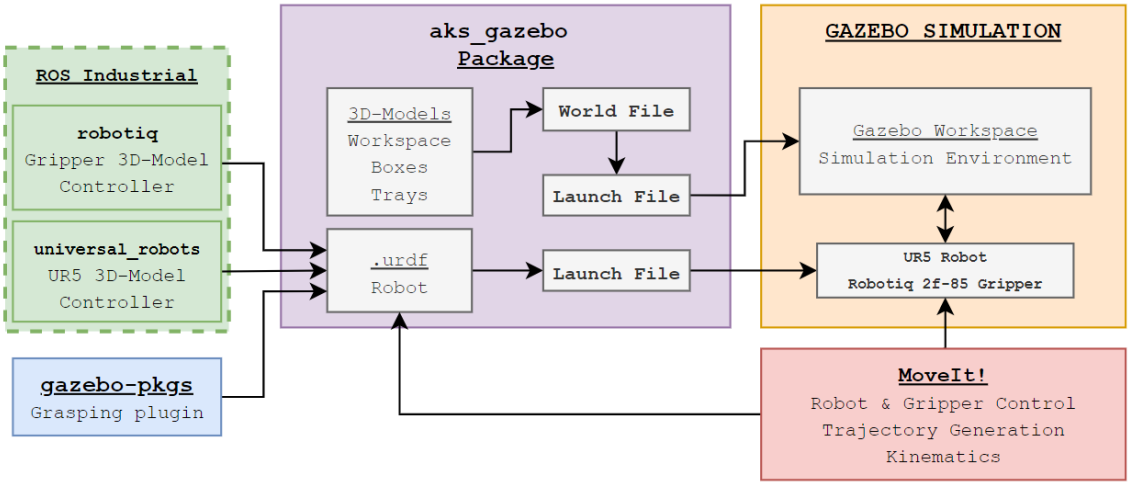


Figure 35. Software Architecture (IV)

4.2.5 Operations. Programming Code

The Gazebo Simulation Environment is configured and ready to undertake the Simulation of the Automated Kitting after *Step 4*. The whole operation is executed and controlled using *Python Files*, which are run by the ***rosrun*** command. The Robot and Gripper motion controllers are executed on the one hand, and all operational steps are executed one by one on the other hand. The main functions of these files will be introduced in this section, and the code development will be explained in *Section 4.3*.

4.2.5.1 Robot and Gripper Control

The performance of the Robot and the Gripper is controlled by the ***aks_robot.py*** and ***aks_gripper.py*** python scripts. These files are executed just after the workspace is ready and they immediately enter an infinite loop, where they are continuously asking for commands and instructions:

- The Robot is continuously asking for waypoints and trajectories, which are subsequently converted to *Joint Values*.
- The Gripper is continuously asking for the *Gripper Open* or *Gripper Close* actions.

Both *Python Scripts* are subscribed to nodes that are published from the files in *Section 4.2.5.2* (operations), and are directly connected to the controllers managed by *MoveIt!* and simulated in Gazebo.

4.2.5.2 Simulation of the Operations

The different processes described in the *Concept of Operations* are simulated by executing the following *Python Scripts*:

- ***step1.py*** asks the user to manually generate the Material List. It asks for the exact amount of spares of each type of object, and creates a .txt file with that information.
- ***step2_rack.py*** and ***step2_tray.py*** scripts verify the availability of both the materials and the racks, and allocate the slots in the MDU and AT for the ML generated before. The availability is checked by consulting the ***MDU.txt*** and ***AT.txt*** files, which contain all the data regarding the situation of both the MDU (***Table 5***) and the Assortment Trays. Two .txt files are created to store the data of the allocated slots.
- ***step3.py*** publishes the data (allocated slots) to the node which the ***aks_robot.py*** script is subscribed, so the robot can receive the commands and start the pick and place (Automated Kitting) task.
- Once the Kitting task is finished, ***step4.py*** automatically updates the data in the ***MDU.txt*** and ***AT.txt*** files, according to the ML (Kit) generated.
- An ***extra.py*** script has been implemented as well, which includes all the extra operations that are not involved in the Automated Kitting but are part of the system:
    - ~ Check the ***MDU.txt*** file.
    - ~ Check the ***AT.txt*** file.
    - ~ Add a box (return) to the MDU.
    - ~ Add a specific amount of spares to a box.
    - ~ Add an AT (return).

Due to the implementation and coordination of these *Python* Scripts, the whole operation of the AKS is simulated manually using the *Terminal*.

4.2.5.3 Software Architecture

Figure 36 illustrates how the Software Architecture looks like after implementing this 5th step. Thus, this is how the system looks like after having implemented the whole Simulation Environment:

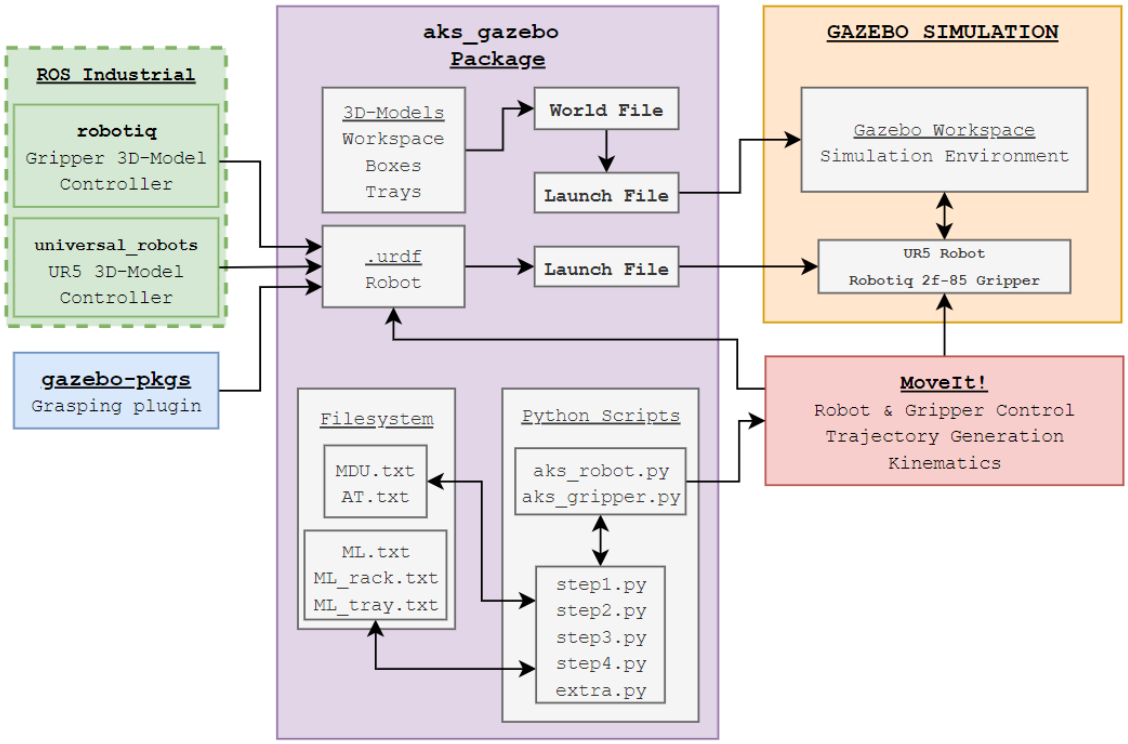


Figure 36. Software Architecture (V)

4.3 Code Development

The design and development process of the programming code used to run the simulation of the AKS will be explained in this section. The code has been developed using ***rospy*** (*Python* code applied to ROS), and following the description provided in *Section 4.2.5*, it can be divided in 2 main parts:

- ***aks_robot.py*** and ***aks_gripper.py***, which are in charge of controlling the simulation of the Robot and Gripper.
- ***step1.py, step2_rack.py, step2_tray.py, step3.py, step4.py*** and ***extra.py,*** which are in charge of running the simulation of the AKS operations.

Different flowcharts will be provided in order to enhance the explanations about how each specific python script works.

4.3.1 Python Code. Robot and Gripper

Figure 37 represents how both ***aks_robot.py*** and ***aks_gripper.py*** files manage to control the operation of the Robot and Gripper during simulation:

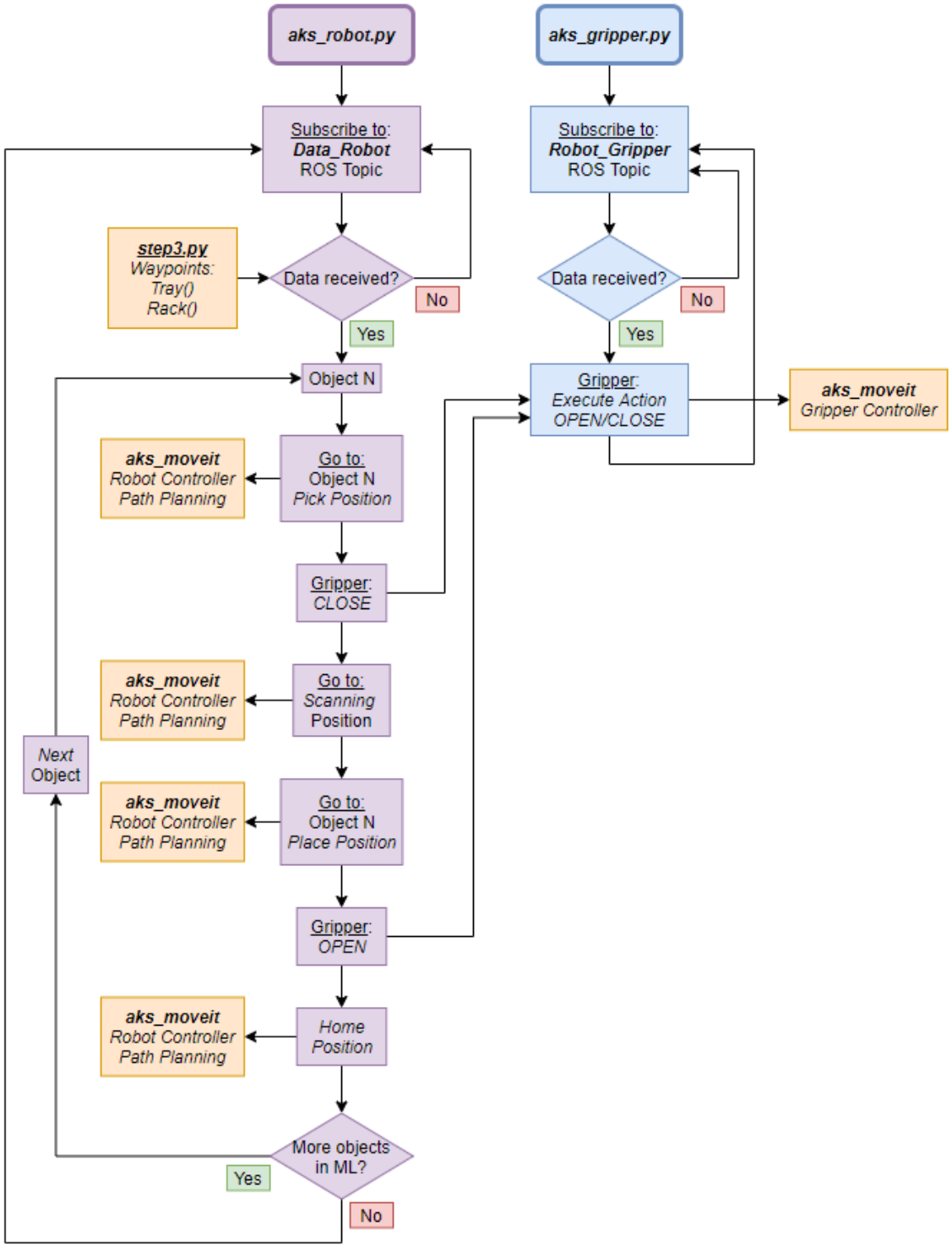


Figure 37. Python Code. Robot and Gripper

As soon as the scripts are executed, both devices embark on an infinite loop, which is based on subscribing to a *ROS Topic* that will be sending the required data to operate.

- In the case of the *Robot*, immediately after receiving the data of the MDU and AT slots, it starts the Kitting Process, which consists in converting the box and tray positions intro robot poses, and sending them to *MoveIt!*.
- The *Gripper* will be always asking for an *action*, which will be sent directly from the *aks_robot.py* script when the action is required.

Figure 38 is an example of how a *robot pose* command is sent to *MoveIt!* from the ***aks_robot.py*** script:

```
# 2. Box3.2 Pick Position:
print "2. STEP: MOVING ROBOT TO B3.2_PICK POSITION"
joint_goal = move_group.get_current_joint_values()
joint_goal[0] = CONV * -146
joint_goal[1] = CONV * -73
joint_goal[2] = CONV * 72
joint_goal[3] = CONV * 1
joint_goal[4] = CONV * 124
joint_goal[5] = CONV * 0

move_group.go(joint_goal, wait=True)
rospy.sleep(1)
```

Figure 38. Sending a Robot Pose to Simulation

Figure 39 shows how the actions are sent to the gripper in order to *open/close* it in simulation:

```
# GRIPPER OPEN/CLOSE ACTION:
if (gripper_data == "open"):
        print "OPENING GRIPPER"
        gripper_goal = move_gripper.get_current_joint_values()
        gripper_goal[2] = 0.00
        move_gripper.go(gripper_goal, wait=True)
        gripper_previous = "open"
elif (gripper_data == "close"):
        print "CLOSING GRIPPER"
        gripper_goal = move_gripper.get_current_joint_values()
        gripper_goal[2] = 0.39
        move_gripper.go(gripper_goal, wait=True)
        gripper_previous = "close"
```

Figure 39. Sending Gripper OPEN/CLOSE actions in Simulation

4.3.2 Python Code. Operations

This section covers how the logistics and data management behind the operation of the Automated Kitting Station have been implemented into the Simulation. Since no APP Software is used in this case, all the operation steps have been simulated manually using the *Terminal*, and are based on 2 main principles:

- The data will be created, managed and shared using ***.txt*** files. Thus, all the information regarding the MDU and AT and all the inputs regarding new ML requests will be written to and read from text files.
- The data stored in .txt files will be converted into ***Python Dictionaries*** for posterior manipulation inside the *Python Scripts*. This feature is useful to compare, assign, add, remove, write and read variables in Python.

Flowcharts in "***Appendix B. Code Development. AKS Operations***" demonstrate how the programming code for the Simulation of the AKS Operation has been developed using ROS and Python.

4.4 Results and Validation

The Simulation results will be provided in this section. The simulation of 2 kitting requests and all the extra operations have been done in order to validate the performance of the Automated Kitting Station. All the steps involving the logistics, data management and the actual Kitting Simulation will be evidenced by providing screenshots of both the *Terminal* and *Gazebo Simulation*. The same ML will be requested twice, and both will be called *BOM001* and *BOM002*, which contain:

- 15 M3 Bolts (**Black** boxes).
- 20 M4 Nuts (**Orange** boxes).
- 20 M4 Washers (**Pink** boxes).

4.4.1 Step 1

The first step of the operation consists in processing the ML request. Thus, the system asks for the amount of spares requested in the ML and creates the *ML.txt* file with the data, as shown in ***Figure 40***.

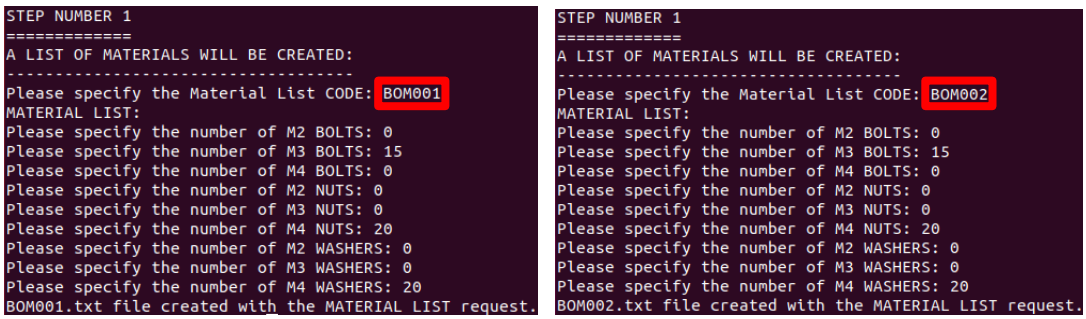


Figure 40. Simulation results. Step 1

4.4.2 Step 2

The system validates the availability of the trays and the materials in the MDU, and creates new *ML_rack.txt* and *ML_tray.txt* files with the allocated slots. The system will always allocate the first box/tray available, after having compared that the amount of spares/slots are bigger than the values requested in the ML.

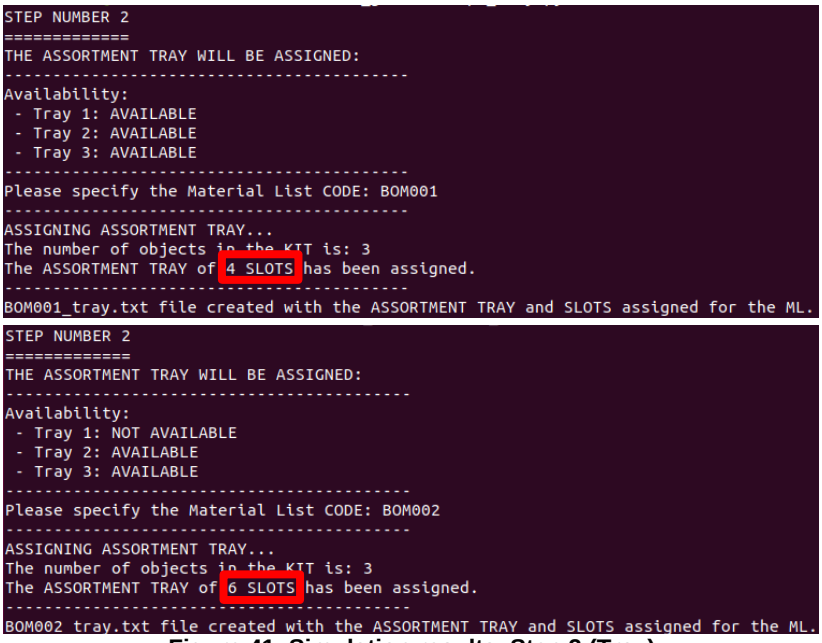


Figure 41. Simulation results. Step 2 (Tray)

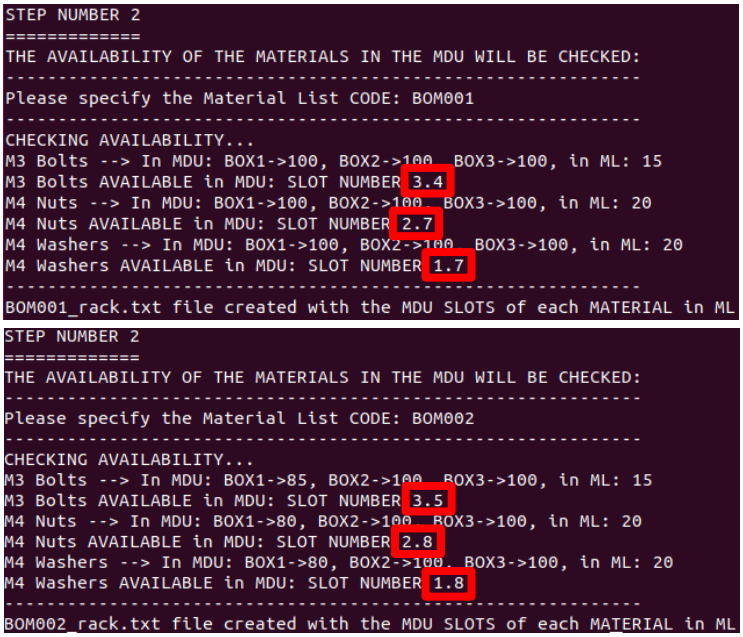


Figure 42. Simulation results. Step 2 (Rack)

It can be observed in both ***Figure 41*** and ***Figure 42*** that, even though the same ML request has been made, the system has allocated different MDU and AT slots

for *BOM002*. This is due to the fact that the system has not been notified about the return of the boxes and tray used for the *BOM001*, and those are not available when the kitting request of *BOM002* is made. Thanks to this implementation, the system is capable of processing kitting requests with the same type of boxes and/or different number of spares of the same type, and this is achieved thanks to the system's capacity of monitoring the material flow in real time.

4.4.3 Step 3

Step 3 consists in sending the data to the robot and undertaking the kitting process. Firstly, the data is gathered and sent to the *ROS Topic* the Robot is subscribed to (***Figure 43***):

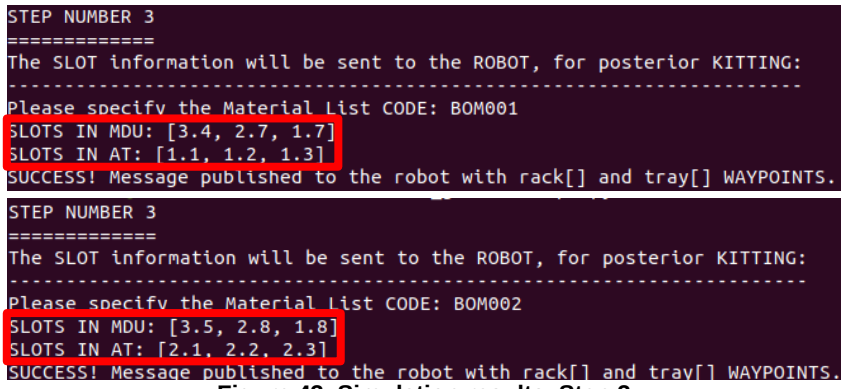


Figure 43. Simulation results. Step 3

As soon as the data is received by the Robot, the Automated Kitting process starts and the boxes are correctly placed in their corresponding slots in the Assortment Trays (***Figure 44***):

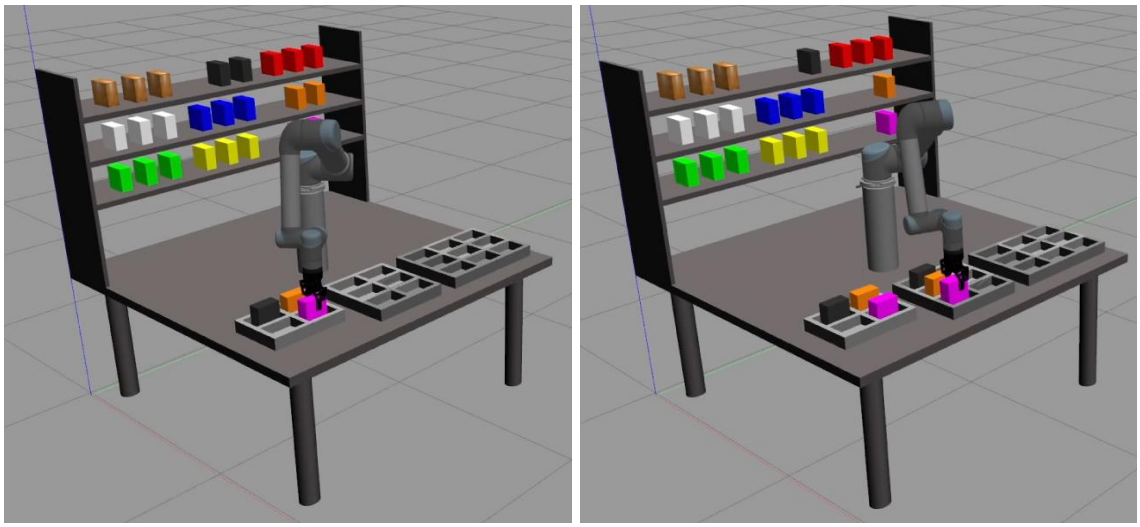


Figure 44. Simulation results. Automated Kitting complete!

All the screenshots about the performance of the Robotic Arm (Pick and Place tasks) in the Automated Kitting Process simulation can be seen in "***Appendix C. AKS Simulation"***.

4.4.4 Step 4

This last step simply consists in manually updating the system once the Automated Kitting Process has finished, as shown in ***Figure 45***:

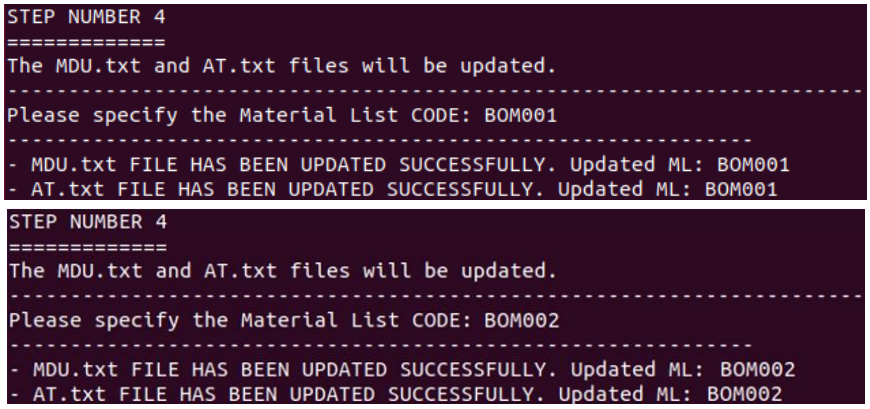


Figure 45. Simulation results. Step 4

4.4.5 Extra Operations

The extra operations are covered in the ***extra.py*** python script and contain the following procedures (***Figure 46***), which are essential for the AKS operation:

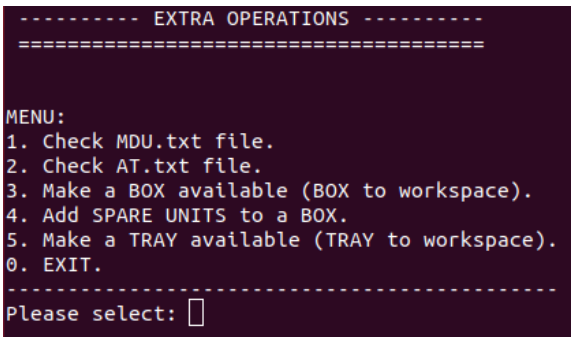


Figure 46. Simulation results. Extra Operations

Make a Box available (Box to workspace)

Each time a box is put into a kit, the MDU.txt file is updated in *Step 4* and that box is no longer available for future kitting operations (until it is returned to the MDU). In the real process, the box would go to the Assembly Line and return to the MDU from the Labelling Station. This process can manually be considered in

Gazebo by manually translating the box back to the MDU, and by manually updating the MDU.txt file, as shown in ***Figure 47***.

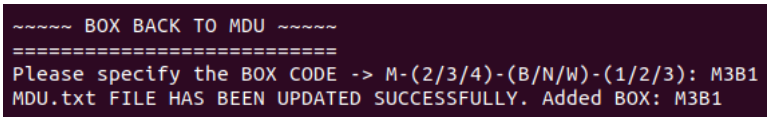


Figure 47. Simulation results. Box back to MDU

Add spare units to a Box

The option of adding spare units to a box has also been added to the process, which allows the user to simply manually add (or remove) spare units from a specific box (***Figure 48***). This operation helps to have the full control of the material flow inside the process.

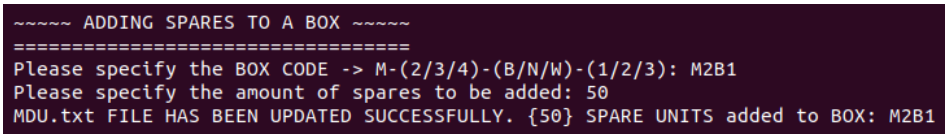


Figure 48. Simulation results. Add spare units to a Box

Make a Tray available (Tray to workspace)

The same principle as the *Box return process* is applied to the Assortment Trays in this case (***Figure 49***). The Tray would be available again in the AKS once the AGV would have transported it back from the Assembly Line.

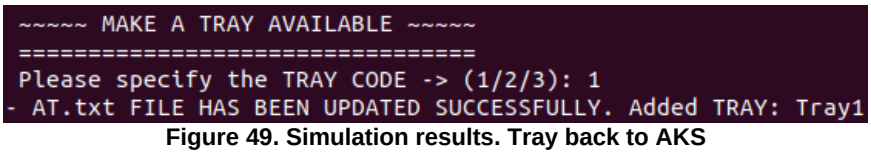


Figure 49. Simulation results. Tray back to AKS

AT.txt file

The AT.txt file can be monitored in real time during the process. ***Figure 50*** illustrates how the availability of the AT-s changes after the 1st (top-right) and 2nd (bottom-left) kitting processes, and after applying the operation shown in ***Figure 49*** (bottom-right).

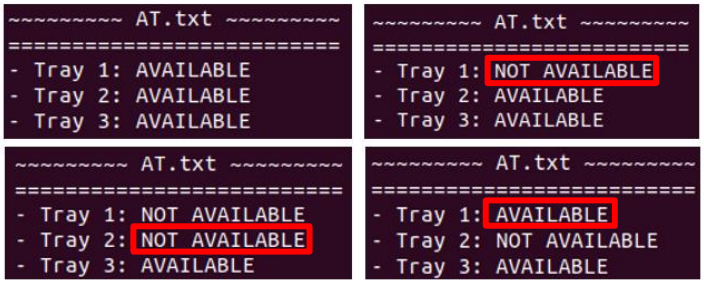


Figure 50. Simulation results. AT.txt

MDU.txt file

The situation of all boxes in the MDU can be monitored in real-time thanks to the *Option 1* in the Extra Operations file. ***Figure 51*** demonstrates how the MDU.txt file is updated after both kitting operations (top-middle/right) and after both extra operations (bottom):

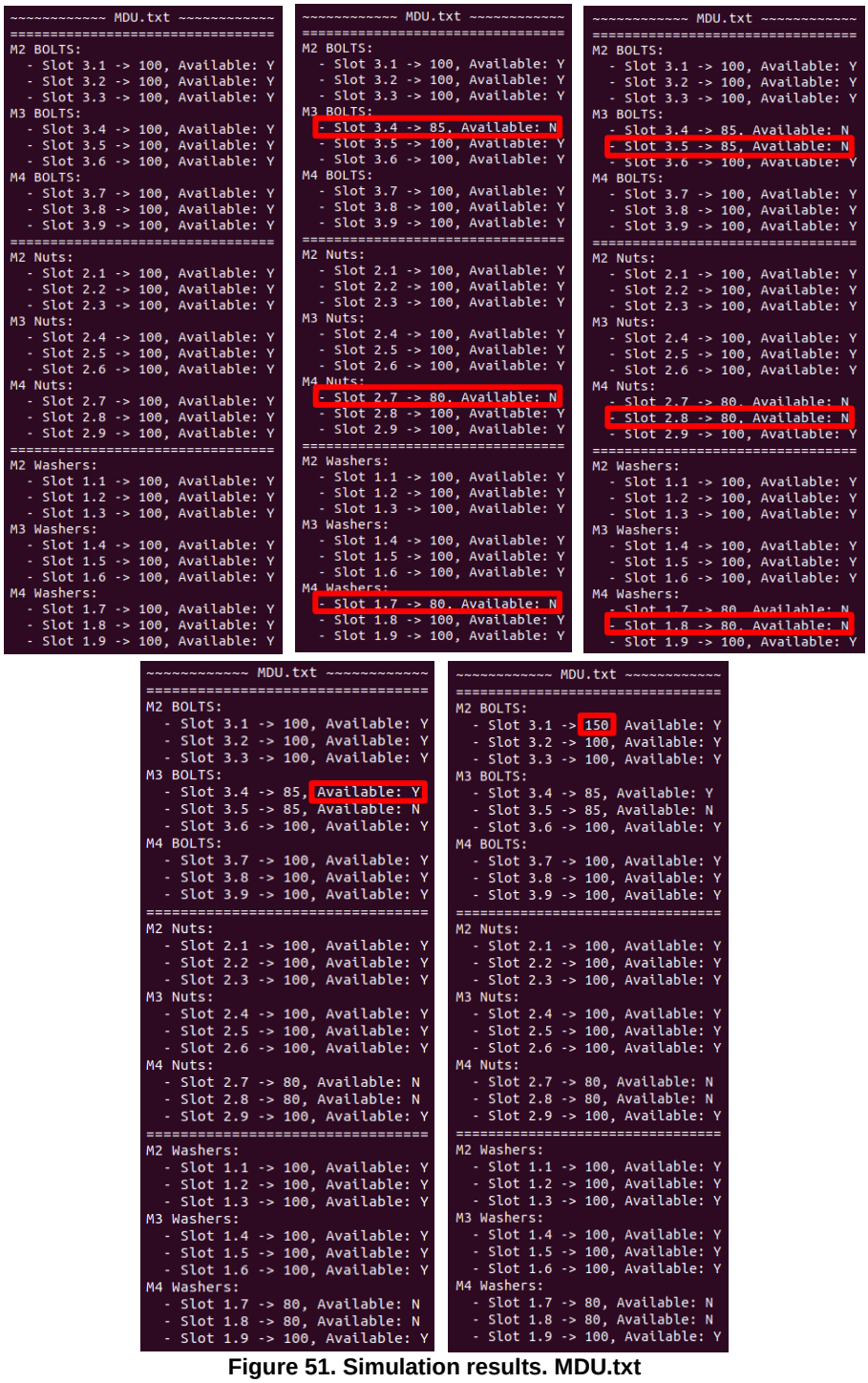


Figure 51. Simulation results. MDU.txt

5 CONCLUSIONS

The aim of the project is to design and develop an Automated Robotic Kitting System demonstrator and its operation, and to validate the system, its capabilities, operation, application and potential implementations using a ROS/Gazebo-based simulation environment.

The results demonstrate that the main objective of designing an Automated Kitting Station and its operation has been met, since the system is able to adequately process a kitting request and prepare the kits automatically. ***Figure 44*** and ***Appendix C. AKS Simulation*** show evidence about the capability of the Robot Manipulator to pick and place the boxes and prepare the kits as soon as a ML request is done, and *Section 4.4* demonstrates how all the process must be simulated in a ROS+Gazebo environment in order to operate the AKS.

The Automated Kitting Station owns innovative and outstanding features, which make it a good solution for future applications:

- It demonstrates an effective method to monitor the material flow and make real-time changes to the data throughout the process.
- The AKS can be implemented within an Automated Production Process, and operate in collaboration with other modules such as the Labelling Station and the AGVs. The different modules can operate autonomously and in parallel, which is a key feature to continuously be operating, avoid bottle necks and save time and money. The continuous material flow has been enhanced by implementing 3 boxes for each type of material and 3 different assortment trays. What is more, the system can perfectly be enlarged by adding more boxes, trays or even adding another AKS to the process, which would guarantee the continuous material and work flow even more.
- The simulated AKS design and operation can be used as a starting point for a future implementation of an Automated Kitting process in a real production environment. The system is flexible in terms of future work and implementations, and this will be evidenced in *Section 6 (Future Work)*. Automated processes require having to consider a great amount of

factors, variables and calculations, and the AKS designed is open to new changes, transformations or implementations.

Apart from the conclusions mentioned above, some other thoughts have emerged throughout the development of the Individual Research Project:

- Kitting is an essential operation inside every single mass production process, and it involves having to invest some time and effort which is not value-adding, but that can be solved with Automated Kitting.
- Manufacturing systems and factories fully depend on well organised and settled logistics and management networks, and Robotics & Automation are key technologies which are essential for those processes to operate effectively. If future engineers manage to implement new working solutions such as the Automated Kitting Station, the future of the industry will directly depend on Automation and Robotics.
- ROS is a powerful and useful platform to simulate and test new Robotic applications and designs. Nonetheless, robot programming and trajectory planning can often be complex and exhausting, since it lacks a high-level and intuitive interface to undertake this operation. Developing a custom software/interface for this task could be an interesting potential IRP for a future Robotics MSc student.

6 FUTURE WORK

The results and analysis provided demonstrate that the operation and logistics behind an Automated Kitting Station have been successfully implemented and validated in this project. Nonetheless, some improvements and implementations must be applied to the AKS before actually considering it as a concept that could be implemented within a real process:

a) The *Bin-Picking* based Automated Kitting could be implemented. An automated kitting process where the Robot, probably equipped with a double-gripper, would firstly place the box on top of the table, and pick and place the exact amount of required spares directly into the kit. The “Box return to MDU” process would be avoided with this new implementation. Thanks to this implementation, the assumption of completely monitoring the material flow would no longer be needed.
b) The operation of the scanner could be included in the simulation.
c) A vision system could be implemented, which would calculate and estimate the pose of the boxes in the MDU and the slots of the Assortment Trays which are on top of the AGV-s. Thanks to this implementation, the assumption of the pre-established box poses and slot positions would no longer be needed.
d) A custom external software could be designed and directly connected to ROS, in order to perform the data exchange and ML request operations autonomously. This implementation would replace the neediness of having to input the data and commands directly from the *Terminal*.
e) The rest of the interactions between the modules inside the APP should be considered and included in the Concept of Operations. Those processes are:
    - Interaction between the Kitting Station and AGV (I) module:
        ~ The process of transporting the kit from the AKS to the Assembly Line.
        ~ The process of transporting an empty Assortment Tray from the Assembly Line back to the AKS.

- Interaction between the Labelling Station and AGV (II) module:
  - ~ The process of transporting a box from the Assembly Line to the Labelling Station.

f) The whole Automated Production Process could be implemented in Gazebo, in order to verify and validate its operation. This would include the Labelling Stations, the AGV modules and all the implementations mentioned above.

g) After successfully validating the whole APP in *e)*, the concept of operations and logistics behind the AKS could be used and implemented within a real production process.

REFERENCES

[1] Weatherwax, J., Unknown. *BlueCart - Kitting Meaning: What is Kitting?.* [Online]
Available at: https://www.bluecart.com/blog/kitting-meaning
[Accessed June 2021].

[2] Marketing, C.-B., Apr 3, 2017. *WESCO - How to implement a Kitting Program.* [Online]
Available at: https://blog.wesco.com/how-to-implement-a-kitting-program
[Accessed June 2021].

[3] Karl, Oct 20, 2020. *MRPeasy - Kitting. A comprehensive guide for manufacturers and distributors.* [Online]
Available at: https://manufacturing-software-blog.mrpeasy.com/kitting/
[Accessed June 2021].

[4] Logistics, K., Apr 12, 2017. *KL - Understanding Kitting Services.* [Online]
Available at: https://www.kanbanlogistics.com/understanding-kitting-services/
[Accessed June 2021].

[5] Adams C. & Berlin C., 2017. Manual Materials Handling. In: *Production Ergonomics: Designing Work Systems to Support Optimal Human Performance.* London: Ubiquity Press, pp. 175-188.

[6] Corakci, M. A., 2008. *An Evaluation of Kitting Systems in Lean Production,* Boras: University of Boras - School of Engineering.

[7] Smith, M., Jul 228, 2020. *Symbia Logistics - Why it's important to automate your warehouse kitting process today.* [Online]
Available at: https://www.symbia.com/blog/2020/7/23/why-its-important-to-automate-your-warehouse-kitting-process-today
[Accessed June 2021].

[8] GmbH, R., Oct 30, 2020. *Robominds GmbH - Making Robots Smart.* [Online]
Available at: https://www.robominds.de/en/
[Accessed June 2021].

[9] Pavlichenko, D., Martín García, G., Koo, S. & Behnke, S., 2018. *KittingBot: A Mobile Manipulation Robot for Collaborative Kitting in Automotive Logistics.* Bonn, Germany, 15th International Conference on Intelligent Autonomous Systems (IAS-15).

[10] Domae, Y., Noda, A., Nagatani, T. & Wan, W., Aug 31, 2020. *Robotic General Parts Feeder: Bin-Picking, Regrasping and Kitting.* Paris, France, International Conference on Robotics and Automation (ICRA).

[11] Tarr, C., Unknown. *Kardex Remstar - Automating your warehouse Kitting process.* [Online]
Available at: https://us.blog.kardex-remstar.com/automating-kitting-process
[Accessed June 2021].

[12] Hanafy, M. & Ingemarsson, S., 2018. *Bin Picking and Placing of complex geometric objects using convolutional neural networks,* Gothenburg, Sweden: Chalmers University of Technology.

[13] Mahler, J. & Goldberg, K., n.d. *Berkeley Automation - DexNet (Dexterity Network).* [Online]
Available at: https://berkeleyautomation.github.io/dex-net /
[Accessed June 2021].

[14] Pickit3D, Apr 29, 2020. *Pickit - Random Bin Picking.* [Online]
Available at: https://www.pickit3d.com/en/3d-vision-applications#3-random-bin-picking
[Accessed June 2021].

[15] Ademovic, A., 2016. Toptal Developers. [Online]
Available at: https://www.toptal.com/robotics/introduction-to-robot-operating-system
[Accessed July 2021].

[16] Dattalo, A., Aug 08, 2018. ROS Wiki. [Online]
Available at: http://wiki.ros.org/ROS/Introduction
[Accessed July 2021].

[17] Gazebo Simulator., n.d. What is Gazebo?. [Online]
Available at: http://gazebosim.org/tutorials?cat=guided_b&tut=guided_b1
[Accessed July 2021]

[18] Wiki, R., Mar 25, 2020. *Ubuntu install of ROS Melodic.* [Online]
Available at: http://wiki.ros.org/melodic/Installation/Ubuntu
[Accessed June 2021]

[19] ROS-Industrial, 2019. *ROS - Universal Robots Package.* [Online]
Available at: https://github.com/ros-industrial/universal_robot
[Accessed July 2021]

[20] ROS-Industrial, 2018. *ROS - Robotiq Package.* [Online]
Available at: https://github.com/ros-industrial/robotiq
[Accessed July 2021]

[21] MoveIt!, n.d. *MoveIt - ROS Melodic.* [Online]
Available at:
http://docs.ros.org/en/melodic/api/moveit_tutorials/html/index.html
[Accessed July 2021]

[22] Buehler, J., Jun 28, 2021. *The Gazebo grasp fix plugin.* [Online]
Available at: https://github.com/jenniferBuehler/gazebo-pkgs/wiki/The-Gazebo-grasp-fix-plugin
[Accessed July 2021]

APPENDICES

Appendix A – Instructions to run the AKS Simulation

Appendix B – Code Development. Automated Kitting Station. Operations

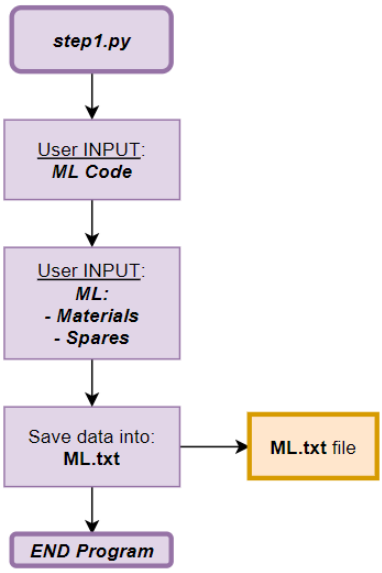
Appendix C – Automated Kitting Station. Simulation

Appendix A. Instructions to run the AKS Simulation

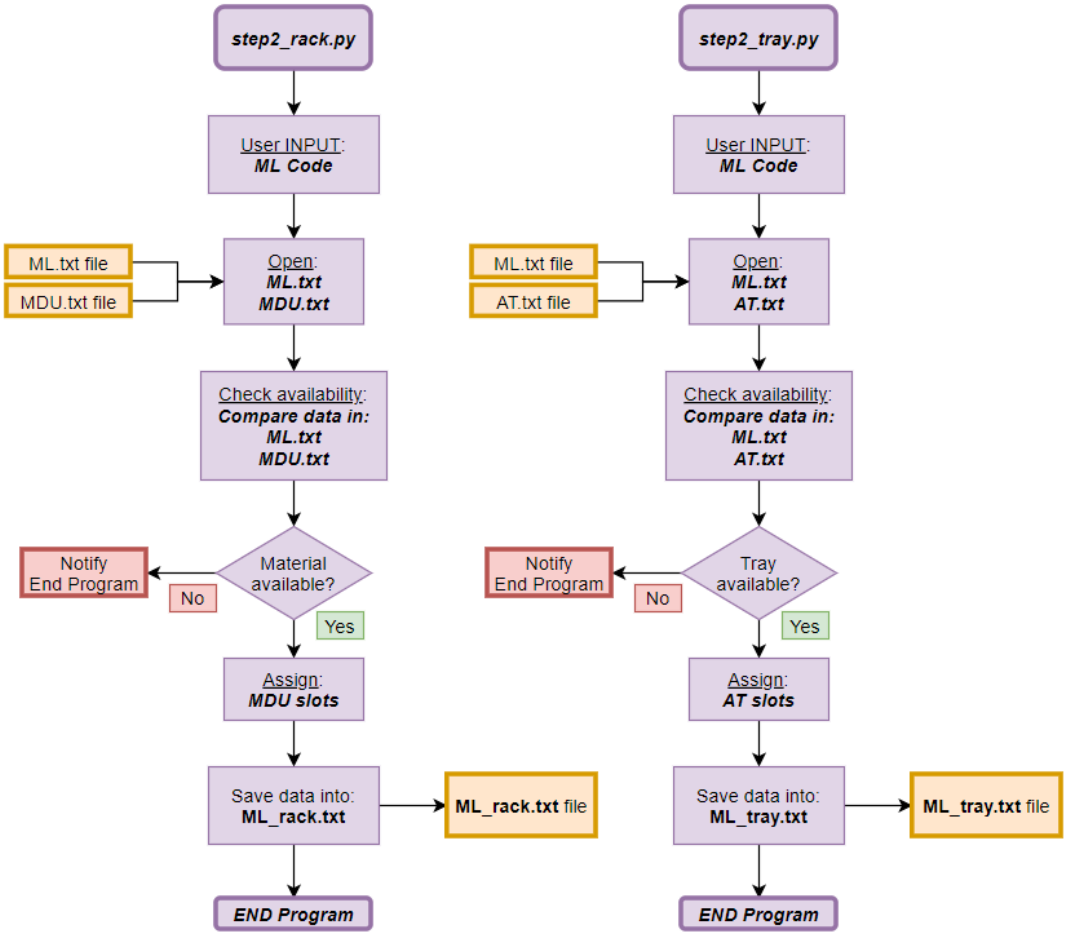
The following process must be followed using the *Ubuntu Terminal* in order to run the AKS Simulation. Each step should be independently executed in a new terminal tab/window:

1. Run the ROS Master.
   Command: ***roscore***
2. Launch the Gazebo world file, which will open the Gazebo simulator and the simulation workspace.
   Command: ***roslaunch aks_gazebo aks_workspace.launch***
3. Spawn the Robot to the simulation workspace.
   Command: ***roslaunch aks_gazebo spawn_robot.launch***
4. Launch the MoveIt! package, which will launch the controllers and control the performance of the Robot.
   Command: ***roslaunch aks_moveit aks_moveit.launch***
5. Run the Python Script in charge of controlling the Robot.
   Command: ***rosrun aks_gazebo aks_robot.py***
6. Run the Python Script in charge of controlling the Gripper.
   Command: ***rosrun aks_gazebo aks_gripper.py***
7. Run the stepX.py scripts, by following the simulation procedure provided in *Section 4.4*.
   Command: ***rosrun aks_gazebo stepX.py***
8. Run the extra.py script, which will allow the user to monitor the material flow and execute the extra operations.
   Command: ***rosrun aks_gazebo extra.py***

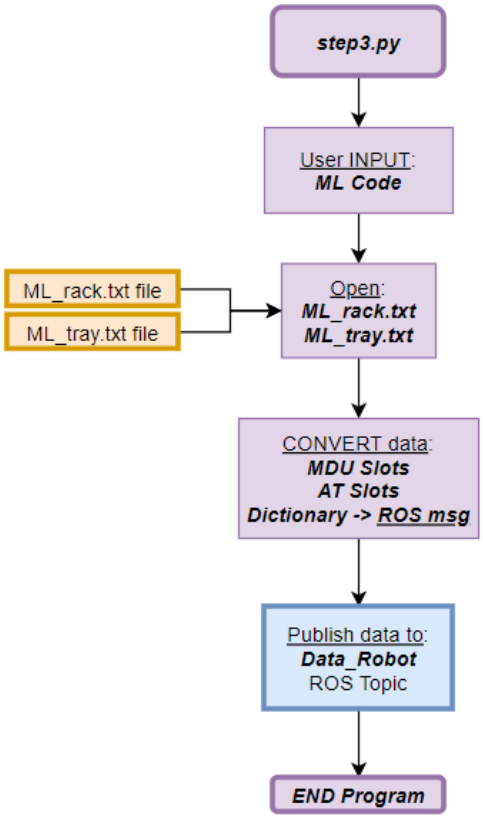
Appendix B. Code Development. AKS Operations



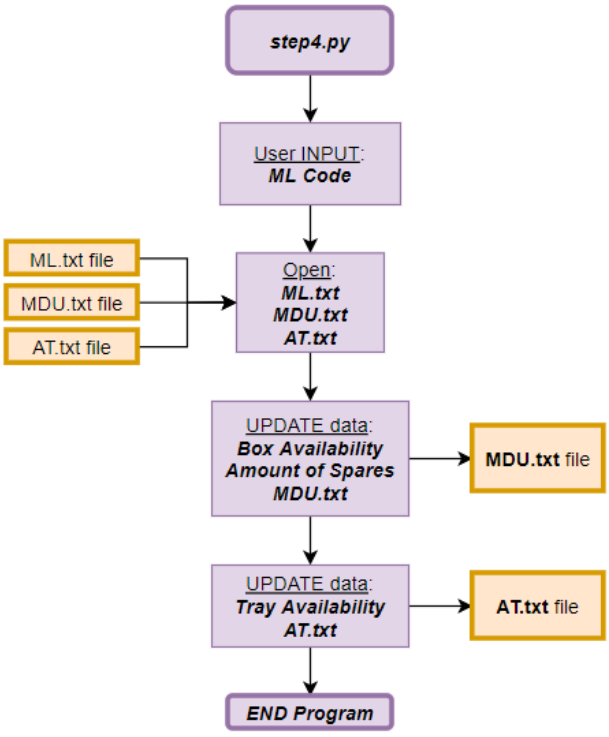
Figure_Apx 1. Code Development. step1.py



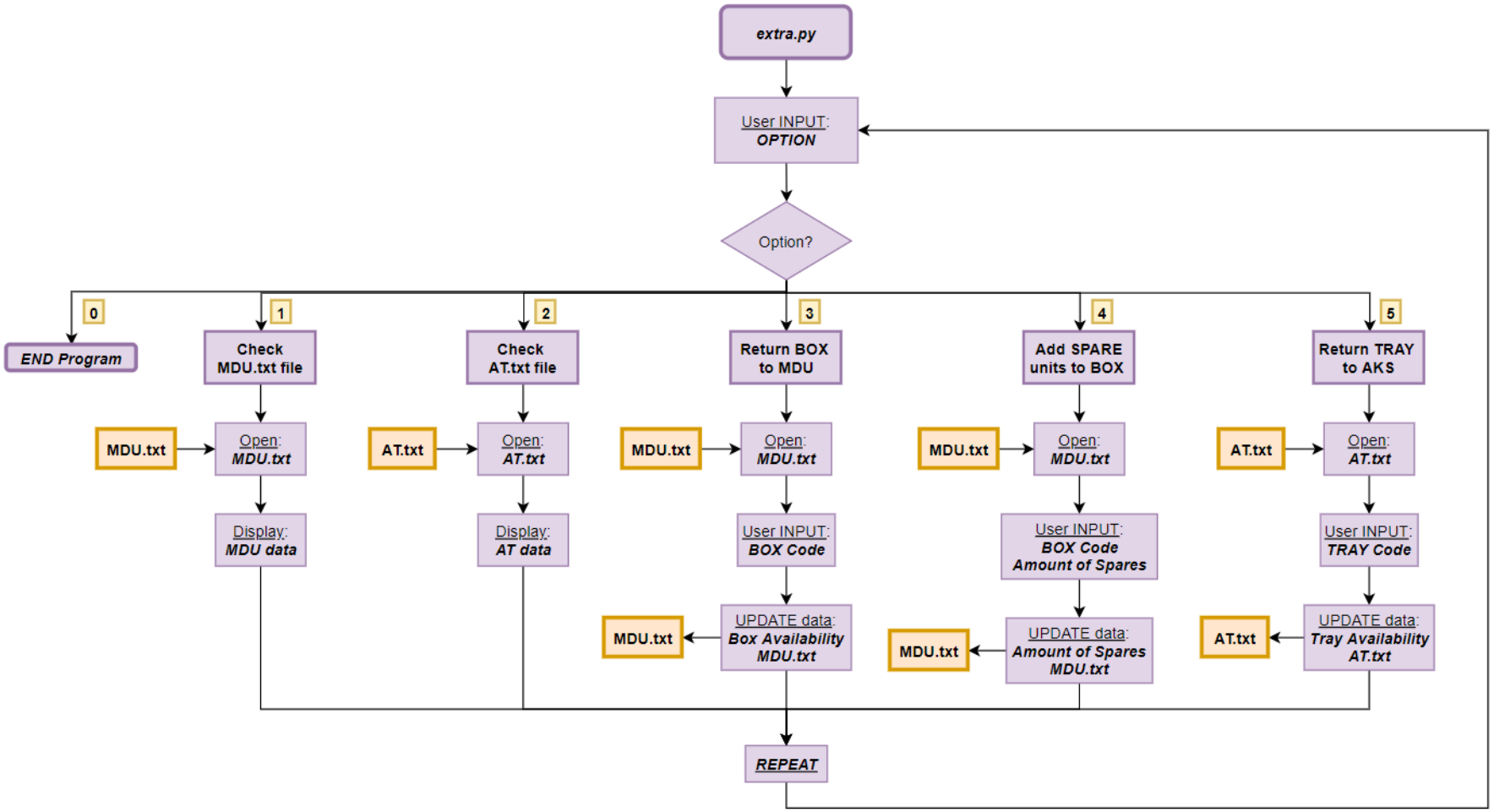
Figure_Apx 2. Code Development. step2.py



Figure_Apx 3. Code Development. step3.py

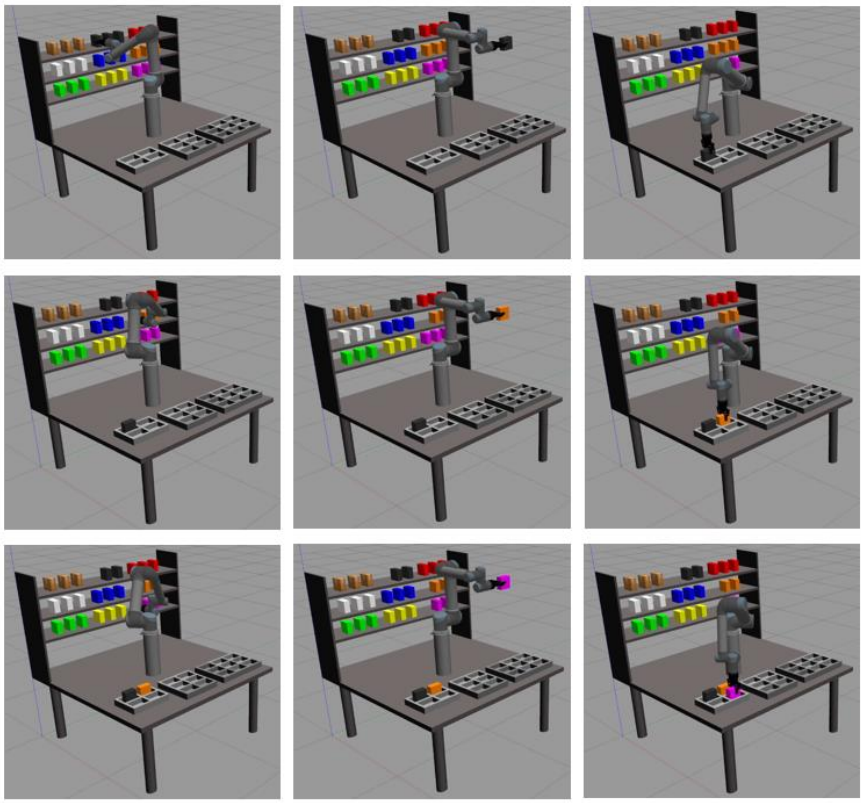


Figure_Apx 4. Code Development. step4.py

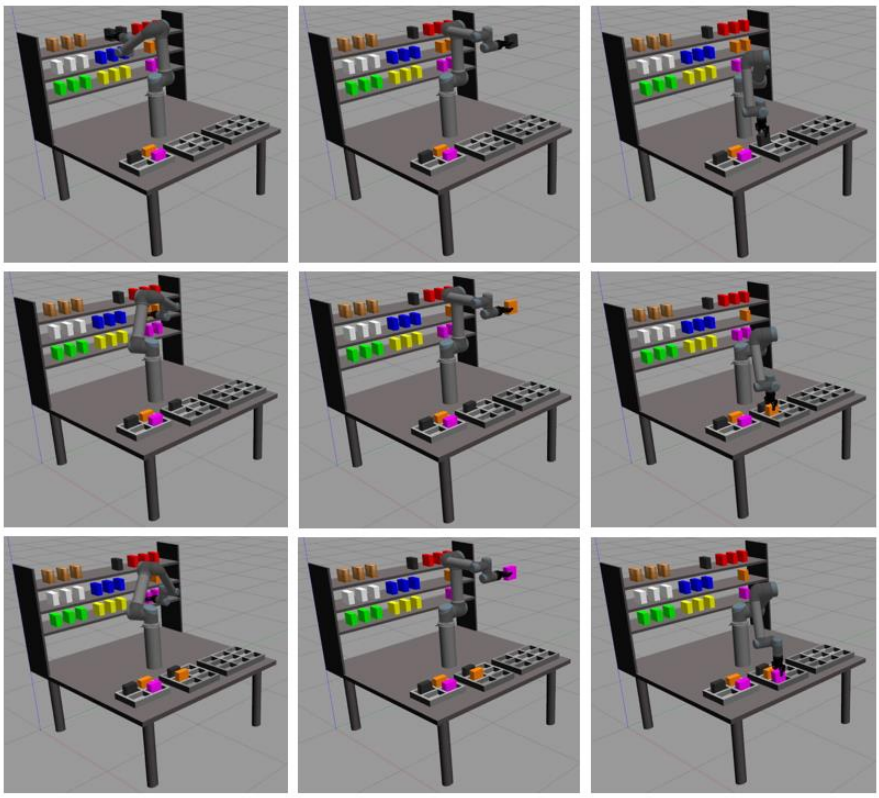


Figure_Apx 5. Code Development. extra.py

Appendix C. AKS Simulation



Figure_Apx 6. Simulation of the BOM001 Kitting Request



Figure_Apx 7. Simulation of the BOM002 Kitting Request