Tampere University

Lauri Nuutinen

DESIGN AND IMPLEMENTATION OF A PICK AND PLACE SOLUTION USING A COLLABORATIVE ROBOT

Bachelor's thesis
Faculty of engineering and natural sciences
Examiner: Luis Gonzalez Moctezuma
January 2025

ABSTRACT

Lauri Nuutinen: Design and implementation of a pick and place solution using a collaborative robot
Bachelor of Science Thesis, 24 pages
Tampere University
Bachelor of Science and Engineering
January 2025

Robots as a supplement to the workforce are becoming increasingly popular to increase efficiency and reduce strain on human workers. The utilization of robotics in any task requires planning and programming. With the increasing usage of robotics also comes an increase in the number of people programming robots. While conventional programming is making its way to elementary schools, robot programming is not as widely present in society.

The goal of this thesis is to act as an introductory look at the process of producing a robot solution, by designing and implementing one. This thesis aims to provide insight to the thought process by methodically going over all the decisions, from the structure of the program to the shape of the carried object while providing alternative options and justifying the made choices.

The thesis is split into three parts. The first part explores current utilization and classifications of robotics through a literature review providing a brief look into how different types of problems are approached. In the second part the planning of the robot solution is presented. Requirements for the robot solution and demo setup are explained in the requirements chapter, and different possible solutions for filling those requirements are explored in the design chapter. The last third of the thesis covers the final implementation of the project, along with the explanations for the chosen design. The conclusions chapter provides a brief reflection on the lessons learned, and a few suggestions on possible future work.

Ultimately the process leads to a functional program and demo setup validating the process. The program produced in the thesis can be found in appendix A.

Keywords: robotics, cobot, robot solutions

PREFACE

The interest in robotics and programming is what originally brought me to Tampere to study automation. The bachelors' programme did not end up including much of that but starting work on this thesis and seeing the FAST-Lab for the first time resparked my dwindling motivation for studying.
Coming from zero experience in robotics and scientific writing, producing this thesis proved to be a challenge, and I would like to thank my friends and especially my fiancée for supporting me during this process. Also, a warm thank you goes to my supervisor Luis Gonzalez Moctezuma for his patience during these years and for the excellent teaching he has provided me with throughout my master's studies.

Tampere, 20.1.2025

Lauri Nuutinen

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

CNC	Computer numerical control, an automated manufacturing process that operates machinery by computer.
TCP	Tool center point, used by RAPID to define the exact difference between the last link of the robot and the contact point of the tool to make movements with the contact point instead of the last link.
FAST-Lab	Future automation systems & technologies laboratory, a laboratory at Hervanta campus.

1. INTRODUCTION

In the future the usage of robotics is estimated to rise with fourth industrial revolution (Industry 4.0). The usage of robotics to supplement the workforce should provide an increased level of efficiency and reduce the total amount of human work required for completing tasks. [1]

While using robot solutions to automate workloads greatly reduces the overall effort for completing the given tasks, it also introduces a new task of designing and implementing the solution. The goal of this thesis is to provide a glance to the current state of utilization of robotics and to act as an introduction to the process of creating a robot solution for a problem.

This thesis demonstrates the process of creating a robot solution by methodically going through the steps to solve an arbitrary problem. The task chosen to be solved by the robot solution is designing a custom pallet and moving a payload around the storage positions of the pallet.

Chapter 2 provides an overview of different types of robots and a few applications where robotics are in use today. In Chapter 3 the requirements for the task of the thesis are categorized and listed. The chapter 4 explores possible design choices to fulfil the requirements found in the previous chapter. The next chapter goes over the final implementation explaining the design choices made. Finally, in the conclusions chapter a recap of the process is given along with some ideas for future work.

2. LITERATURE REVIEW

The purpose of this chapter is to present some of the relevant topics to the thesis from the viewpoint of literature and to highlight some of the uses for robots in the current day world.

2.1 Robot utilization

The usage of robots in industry is on the rise. In 2022 the combined total of industrial robots in use reached 3,5 million robots. And the growth is estimated to continue at least for the near future. [2] Automating industry using robots is said to increase factory productivity with no significant decrease in employment numbers, but rather a shift from low skilled labour to skilled labour [2].

At first robots were used to replace humans in simple pick and place tasks to achieve more productivity and to replace humans in heavy, dangerous, and monotonous tasks even in hot or cold environments [3,4]. With technology advancing the use cases for robotics grow constantly. The following subchapters highlight a few of the current applications of robotics.

2.1.1 Palletizing and depalletizing

Loading items on to and from a pallet, known palletizing and depalletizing are essential tasks in logistics and a prime target for automation as the tasks are often straining and monotonous. With the use of industrial robotics palletizing work can be considerably sped up and the variance of human error can be negated.

Some of the largest obstacles in automating palletizing tasks are related to unstandardized properties of the goods being handled [5]. In applications where different kinds of goods with highly varied properties need to be placed on the same pallet the utilization of human workers or multiple kinds of grippers becomes inevitable.

Non-fragile goods in standardized packages can be efficiently picked in whole layers using jaw grippers. When picking fragile goods that would be damaged by clamping form-fitting fork grippers can be used instead. If goods are packaged in bag-like containers, typically a bag gripper is used. [6]

2.1.2 Assembly

The first modern industrial robots were assembly robots. In 1961 the Unimate robot started operating as a part of an assembly line at General Motors [7]. When used as a part of an assembly line robots can be highly specialized to doing just one task over and over with great accuracy and efficiency.

In industry assembly robots are often used in conjunction with human workers. The robots handle the straightforward tasks while humans can focus on the more complex operations such as checking for defects in the product and solving error situations.

2.1.3 CNC machine servicing

One interesting use case for industrial robots is using then to transfer components to a CNC machine. CNC milling machines are a large investment for many companies. Given that the CNC programs often finish at inconvenient times for human operators due to their time-consuming nature, the utilization of a robot for exchanging components between these programs can increase the active milling hours. This enables the machine to achieve greater uptime contributing to increased productivity.

2.1.4 Surgical robots

Utilizing robots for surgeries differs from many other robot use cases in that surgical robots aren't used in automating relatively simple and laborious tasks but rather to assist in highly precise operations with the human body. The regulation regarding automating medical operations on humans is understandably strict. Unlike most robots, surgical robots are not allowed to operate autonomically, a surgeon is always required to have control of the robot. The level of control required by the surgeon depends on the operations, ranging from robot assisted surgeries where the surgeon precisely controls every move the robot makes to high-level autonomous operations where a human supervisor only has a capability to stop the procedure by hitting an emergency stop button. [8]

2.2 Robot classifications

The mechanical specifications of a robot are highly dependent on the use case. Generally, the simplest robot that can complete the required tasks is the optimal robot for the solution. Using a more complex robot than required increases the cost. Many industrial problems such as palletizing and assembly can be solved using robots with less than 6 degrees of freedom to save on costs and to keep the design relatively simple [9].

Some applications inertly have more complex requirements that cannot be achieved by more conventional designs. Often logistical robots need to be able to freely move around in the prem-

ises of a logistical centre or even a city. If assembly robots are required to interface with workers, the use of co-operative robots or cobots is required to guarantee human safety [10]. And when the application is focused on servicing humans in some form instead of achieving industrial goals the used robots are classified as service robots instead of industrial robots.

2.2.1 Stationary robots vs mobile robots

Mobile robots differ from stationary robots in that they are capable of moving around autonomously using some locomotion system, such as wheels, tracks, legs etc [11]. Additionally while stationary robots can often operate essentially blindly by having predetermined locations for where work objects are supposed to be, mobile robots always must have sensors for measuring the environment [12].

Mobile robots are not limited to the ground. Perhaps the most common type of mobile robot is an unmanned aerial vehicle (UAV) or “drone”. Originally drones were developed with military action in mind, but today drones are also used in many other applications such as scientific research, product delivery, surveillance and photography [11].

In addition to the flying UAV’s there are also autonomous underwater vehicles (AUV’s). AUV’s are used for search and rescue missions, ocean exploration and monitoring. Navigating underwater in the ocean poses a challenging problem to solve. Factors such as low visibility and a dynamically changing environment with time-varying and fixed currents create a complex environment that is difficult to both navigate and model. [13]

The uses for mobile robots are endless. Ranging from store bought vacuum robots to multimillion-euro mars exploration robots. Yet from a manufacturing point of view mobile robots do not provide many benefits over stationary robots. It is usually cheaper and easier to move work objects to the robot than to move the robot to the work object. Therefore, in most industrial use cases the cheaper and simpler stationary robots are preferable.

2.2.2 Robots vs cobots

Robots can be classified to three categories based on their level of safety for human interaction. With conventional robots a safety barrier is required to keep humans separated from the robot while its operating. In some applications the existence of such barriers can be detrimental to productivity. When a worker needs to coexist in the same space as a robot the usage of a co-operative robot or collaborative robot (cobot) is essential.

What defines a robot to be a cobot instead of a co-operative robot is that while a co-operative robot is safe to touch when stationary a cobot should be safe to touch even while the cobot is moving [10]. To ensure this high level of safety, cobots are specially designed with a great focus on human safety.

Cobots are commonly built to be shaped roundly to avoid trap points where clothing or hair could get stuck and to reduce the harm caused by collisions. A cobots joint controls are handled using force-torque sensors. These sensors allow the cobot to detect contact with the environment and to stop when unexpected collisions occur. Additionally, cobots are controlled with limited velocity and force to lesser the harmful effects of collisions with humans. [10,14]

2.2.3 Industrial robots vs service robots

Service robots include all robots used for non-industrial purposes [15]. Robots used outside of industrial settings often need to interface with humans not trained for working with robots, which increases likelihood of misuse and therefore means increased safety concerns. Some typical uses for service robots are transportation, elderly care and cleaning.

Elderly care robots can be divided into two groups, assistive social robots, and rehabilitation robots. Assistive social robots support elderly people in their everyday lives by helping with eating, bathing, and getting around. Some assistive social robots are also capable of providing companionship to lonely elders and monitoring their health. Rehabilitation robots are non-communicative devices such as exoskeletons and smart wheelchairs focused on assisting the elderly with their mobility challenges. [16]

In contrast to the industrial sector, in the service industry, a robot replacing a human worker often experiences unpredictability in its schedule and periods of downtime when its services are not required.

3. REQUIREMENTS

The objective for the practical part of this thesis is to get an understanding of the process for producing a robot solution by going through the process. The task is to program a robot to perform a simple pick and place operation and to design a demo environment to demonstrate the solution.

Planning a robot solution starts from compiling the requirements. Requirements can be divided into functional and non-functional requirements. Functional requirements determine what the solution should be able to do, and non-functional requirements specify how things should be accomplished. In chapter 3.1 the functional requirements of the solution are introduced and in chapter 3.2 the non-functional requirements are specified.

3.1 Functional requirements

Firstly, to successfully complete a pick and place task the robot must be able to grip the target object securely. To eliminate the risk of the pieces falling out of the gripper during transport, the pieces should be designed to be compatible with the gripper. Additionally, the optimal angle of approach before gripping should be considered.

Secondly the robot must be able to navigate between the storage positions of the demo setup accurately enough to place the pieces correctly in their slots. The program needs to be able to find a path to every storage position of the demo setup. Also while navigating between storage positions, neither the gripper or the piece being carried should collide with the environment or the demo setup.

Next the program should be written so that it receives orders for transfer operations using column and row integer coordinates instead of real-world distances. To better mimic real-world conditions, the program should run in a loop completing pick and place orders continuously until manually stopped.

Lastly to facilitate the realities of the robot selection available the program should be made for one of the ABB’s robots at the FASTLAB, using the RAPID programming language.

3.2 Non-functional requirements

To practice principles of maintainable programming, the design of the solution should be made scalable to optimize for effortless addition of possible storage positions. To achieve this the solution should be designed to only require one taught reference point. Also, the calibration process of teaching this reference point to the program should be made as easy and exact as possible to consistently achieve a precise reference point.

Finally, in industry, robots performing pick and place tasks can often be found working in close proximity with human workers. To demonstrate the constraints of a robot working in such environments the solution should be designed to be safe for human interaction during its operations.

4. SOLUTION DESIGN

In the design of a robot solution there are endless possible approaches. In this chapter, various approaches, and options for designing the robot solution of this thesis will be explored. First section 4.1 will focus on the different options for the design and fabrication of the demo setup and what are their advantages. Section 4.2 is about the design choices of the program.

4.1 Demo setup

The demo setup should consist of a board and payload that will be transferred between assigned storage positions on the board. Producing the demo setup in the facilities of Tampere university provides multiple options of fabrication techniques. The FabLab offers two options for machine fabrication suited for making a board and a piece for the demo solution, 3D printing and CNC machining. Using 3D printed objects for the setup introduces some constraints for the designs, such as the printing areas of the available 3D printers. Unlike the 3D printers of FabLab, the CNC routers require an instructional course to use. As an advantage over the printers, the CNC routers offer ample workspace for any setup design. The university also has a woodworking workspace suitable for manually producing the setup.

The demo setup requires a board with the storage positions of the payload laid out in a constant pattern to help visualize the how accurate the pick and place operations are. The storage positions should be laid in a nontrivial but logical grid so that there is the option for the program to calculate the coordinates of each storage position. Each storage position should be identical with one another. The dimensions of a storage position should be made large enough to easily support bases payload pieces while being small enough to fit at least around ten storage positions on the board. The storage positions can be made as pits or stands on the board. Creating a board with stands is easier than a stand with pits while the pit storage positions offer some extra support to the pieces.

The payload piece transferred between the storage positions could be made to be any shape but to optimize the functionality of the solution the design should focus on easily grippable shapes. A rectangular piece design provides the most stable grip but requires

a somewhat precise angle of approach. Using a spherical payload removes the angle of approach of the gripper out of consideration but also places some extra constraints on the storage positions of the board. Using a cylindrical piece instead still has the benefit of removing the angle of approach constraint around the z-axis while being less restrictive on storage position design. Additionally using a taller piece lessens the precision required it the gripping position of the payload in the z dimension.

4.2 Robot program

When designing the program for the pick and place task specified in the requirements chapter, the necessary actions of the robot are clear. The robot needs to move to the pick object, grip it, move to the place target, and release the object. To implement these actions there are numerous design choices to be made. The details of how the movements of the robot should be implemented, what data should be stored and how, the structure of the program and how the different components of the solution should be modelled using the tools provided by RAPID.

RAPID programming language offers three different types of basic movement commands: linear movements, joint movements and circular movements. Any of the three types of movements work for a pick and place task but considering the possibility of obstacles such as other pieces on the board using a single linear or joint movement is not sufficient. A circular movement can be configured to use a circle point above the board between the pick point and the place point to avoid collisions. This approach is simple, but inefficient in the regard that the manipulator must move unnecessarily much to avoid collision with obstacles near the pick position. Using multiple linear movements allows for a more efficient path while avoiding the obstacles. Figure 1 demonstrates the inefficiency of circular movements compared to linear movements when avoiding nearby obstacles. Joint movements can achieve similar results as linear movements while being faster for the robot to execute at the expense of introducing some unpredictability to the movements.

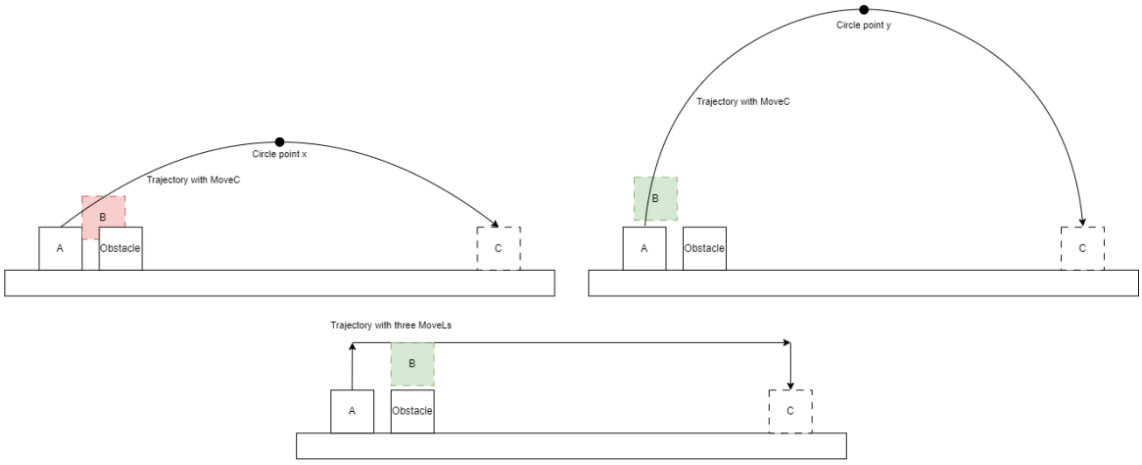

Figure 1. *Moving object A to position C causes a collision when using the circle point x. With the more distant circle point y, the collision is avoided. Using multiple linear movements also avoids the collision and the path is more efficient.*

Each move command takes a zone data and a velocity parameter. These parameters define how close to the target the tool center point (TCP) should come and at what velocity the end effector should move. To optimize the program to perform its task the zone data parameter z should be chosen according to the goal of each movement. Greater accuracy can be achieved with a smaller, more precise z parameter while the movements that do not require exact positioning can be produced using a larger z, allowing the robot to take a more efficient and rounded path. Using fine as the z parameter causes the program to finish the current movement before moving on with the execution of the program. Similarly to the zones, the velocities of the TCP should be optimized. Using slower movements helps to reduce the severity of collisions while faster movements lead to better performance of the task. For each movement the velocity parameter v should be chosen according to the goal of the movement.

To make sensible pick and place operations the program must store data about the properties of the demo setup and its state. The program must be able to create move commands to all the storage positions of the board. To achieve this, the position data of each storage slot can either be stored as a robtarget in a suitable data structure or the distances between the positions of the grid can be stored to allow the calculating the coordinates of the storage positions. Storing each storage position as a robtarget presents a more straightforward approach to making move commands but introduces a new problem in retrieving the position data corresponding with its grid coordinates. Similarly, the current position of the piece must be stored either as a robtarget or in some other way that can be translated to a robtarget when necessary. To enable the

program to run endlessly in a loop, the end position of the piece must be stored as the starting position for the next iteration of the pick and place loop.

RAPID offers the possibility to model real objects as work objects. A work object is another reference frame for the program to use. This simplifies making precise movements of the TCP along the surface of objects and is especially useful when there are multiple objects at nonparallel angles to base axes. For the demo task both the piece and the board could be modelled as work objects to simplify the movement commands to be in respect to the physical objects instead of the base frame.

When moving from the RobotStudio environment to the real world the robtargets initialized in the RAPID program need to be linked with real world positions. To achieve this the robot is manually moved to the desired position and then the position is taught to the controller. To aid this operation either the initialized robtarget should be chosen so that the exact position is easily identifiable and reachable, or some calibration object should be created to help in the process.

5. IMPLEMENTATION

In this chapter the implementation of the program and the demo setup are presented along with the decisions made during the planning process.

5.1 Choise of robot

To satisfy the requirements of using an ABB robot and being safe for human interaction, FAST-Lab offers one choice, the ABB's Dual-arm YuMi IBR 14000 pictured in figure 2. The robot is equipped with ABB Smart Grippers on both arms. Only one arm will be used for the task. This robot and the corresponding workstation introduce some new limits such as payload max weight and work envelope size, but the effect of these restrictions should be minimal.

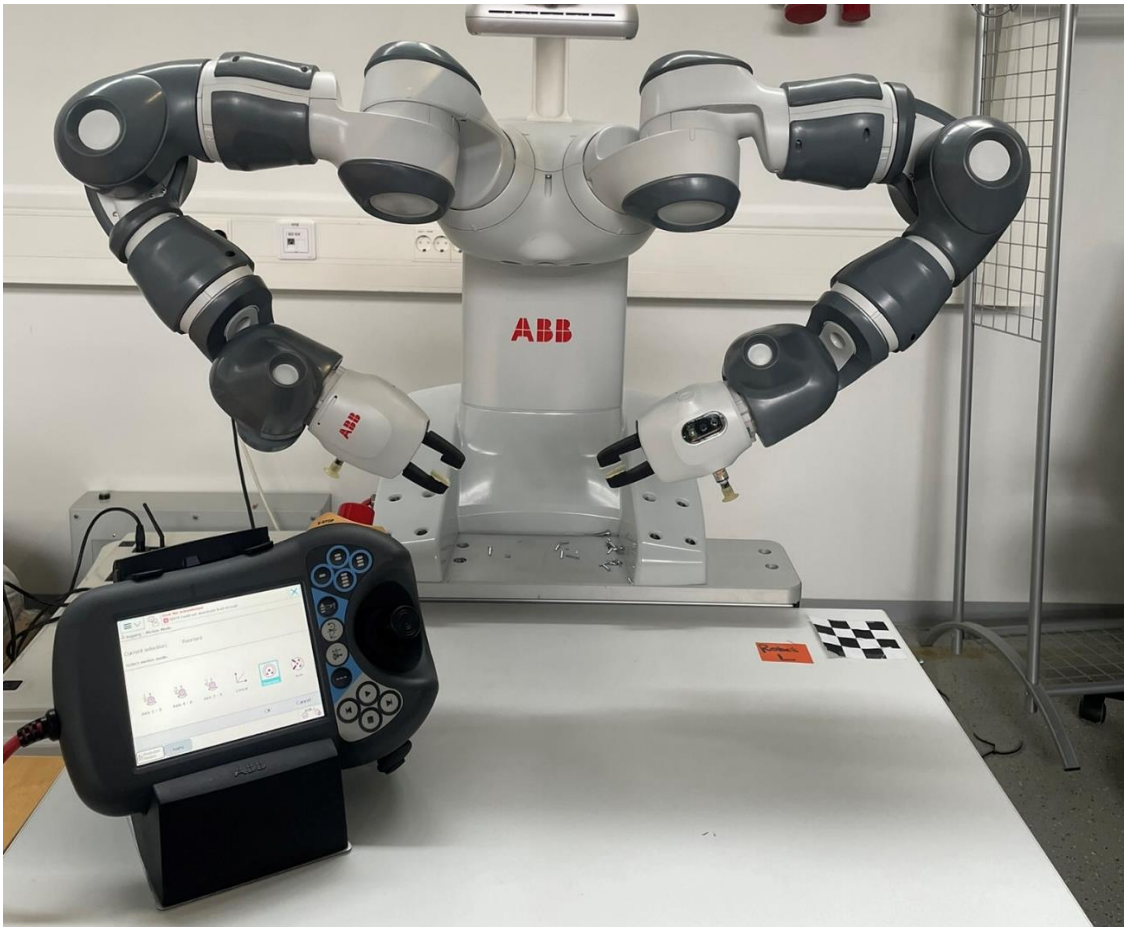

Figure 2. *The YuMi workstation at FAST-Lab.*

5.2 Demo setup implementation

Blender was used to design the models for both objects, the board and the piece. The models were fabricated using a 3D printer at the FabLab. Machine fabrication was chosen instead of manual fabrication due to the simplicity of producing the planned objects by exporting the models as .stl files. CNC machining was not chosen because of the required introductory course necessary to access the CNC machines.

The final model used for the board is presented in figure 3. The storage layout of the board was chosen to mimic another board found at FAST-Lab that fit the requirements. The design choice for a single storage position is a pit to decrease the chance of a piece falling over. The dimensions of the board are 363 mm x 200 mm x 15 mm which is larger than the maximum printing area of 250 mm x 210 mm x 210 mm available to the 3D printers of FabLab [17]. For this reason, the board had to be printed as two connectable pieces as seen in figure 4. The connection is achieved by sliding the left objects appendages into the slots of the right object form above. The appendages are made slightly smaller than the slots to allow this movement while still maintaining a relatively sturdy board design when put together.

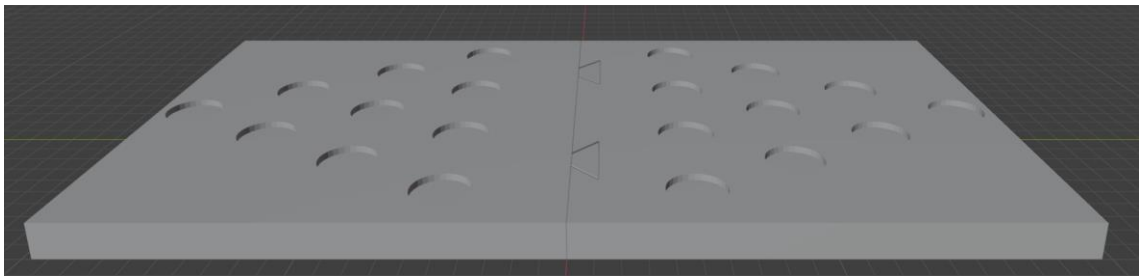

Figure 3. *Blender model of the board.*

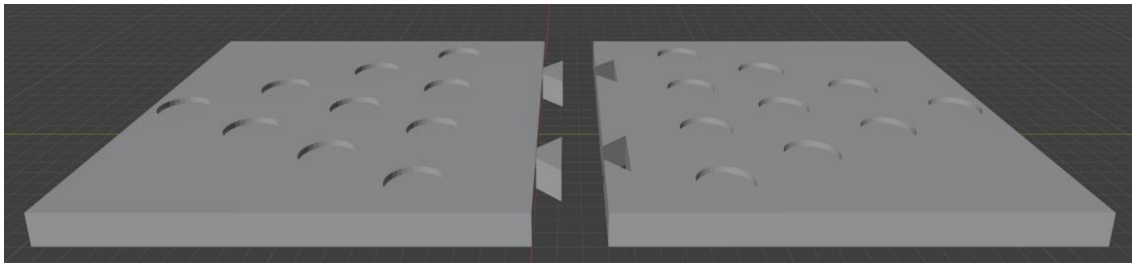

Figure 4. *Blender model of the separated board pieces.*

The board contains storage positions in eight columns and seven rows for a total of 20 storage positions. To satisfy the requirement of using numbered row and column coordinates for transfer operations each row and column are given an ordinal number.

Figure 5 presents the grid coordinate system of the board as seen from above along with two example coordinates, A and B.

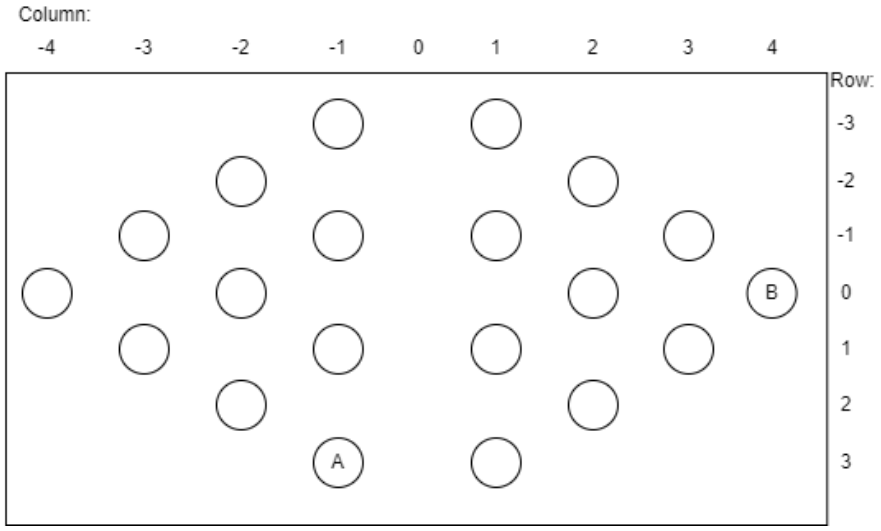

Figure 5. *Grid coordinates of the board. Example position A (3, -1) is located on row 3, column -1. Example position B (0, 4) is on row 0, column 4.*

For the piece a rectangular desing was chosen because it offers the most stable stable grip. Compared to a circular desing, this rectangular desing places more precise restraints for the approach angle of the gripper as the fingers of the gripper should be nearly parallel to the pieces sides before grasping the piece. However this was deemed a nonissue as the program can effortlessly set the precise angle of the gripper during all of the crucial operations. To match the shape of the boards storage positions, the base of the piece is also made to be circular. In most scenarios this should not affect the pick or place operations but in case of an erroneous place operation the piece should sit more easily in the storage position. Blender model of the final piece desing is presented in figure 6.

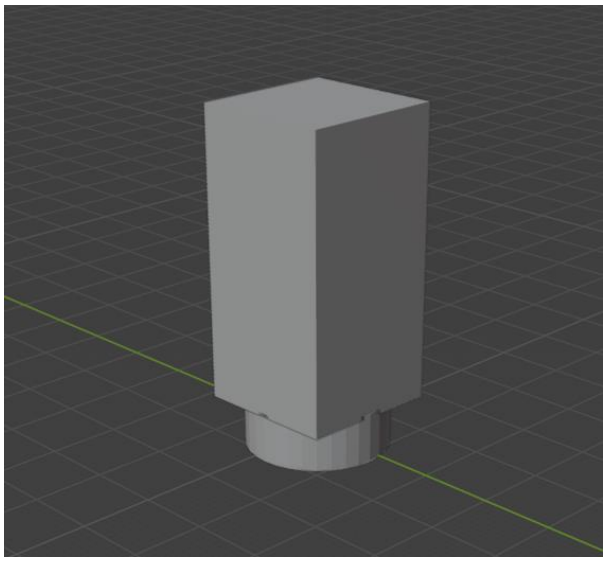

Figure 6. *Blender model of the piece.*

5.3 Program implementation

The final implementation of the RAPID program is presented in appendix a. A view of the robot and the demo setup in the RobotStudio environment is portrayed in figure 7.

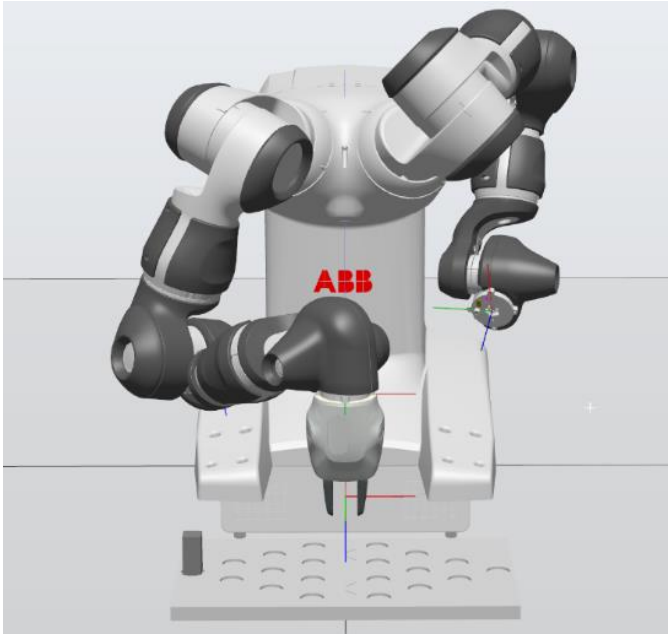

Figure 7. *YuMi and the demo setup in RobotStudio.*

At the start of an execution of the program the gripper is opened, and the robot moves to the center position. Then, to satisfy the requirement of perpetual pick and place operations the main procedure calls all operations necessary for the task from an infinite while loop. The order of operations goes as follows: at the beginning of the loop the NavigateTo procedure is called with the current row and column of the piece to calcu-

late the coordinates and to perform the movement. Next the procedure PickPiece is called. This procedure handles actions required to approach the piece and grasp it. The procedure ends at the same location where it started. After the piece is picked target grid coordinates are randomly generated by the functions GenerateRandRow and GenerateRandCol. Then NavigateTo is called for the second time with the target coordinates to move above the target position after which PlacePiece moves the gripper down and releases the piece. Finally, the new position of the piece is saved to the PieceCurrentRow and PieceCurrentCol variables and the robot moves back to the center position from where the next iteration of the loop will begin.

The final implementation of the program issues movement commands from four procedures: main, NavigateTo, PickPiece and PlacePiece. The movement commands of main are implemented as joint movements since they are executed when the robot is not holding the piece, making collisions unlikely. All movement commands outside of the main use linear movements to achieve precise and predictable motions. The velocity parameter v for each movement command is determined by to direction of the movement. For movements approaching the board the velocity v50 is used while for the movements parallel to the board and receding from the board use the parameter v200. This is done to lessen the impact force in an even of collisions. The zone parameter z is set to z10 as a baseline for most movements. In cases where the movement needs to complete before the execution of the next commands can happen the zone parameter fine is used. Procedures PickPiece and PlacePiece use this method to ensure that the movement is complete before operating the gripper. The NavigateTo procedures movement command also uses fine so that the position of the robot can be read correctly in the next procedure call.

To satisfy the requirement using a single taught reference point, only the robtarget center is initialized when the program is run. This target is then used when calculating future robtargets for the movements. In the final implementation the coordinates of the storage positions are not stored but rather calculated every time before a movement command is given. This is done to simplify the process of going from randomly generated grid coordinates to getting a robtarget. The calculations are performed by the RowToDis and ColToDis functions, utilizing constant variables ROWGAP, COLGAP, and MIDGAP. These three variables describe the dimensions of the board. To use a differently sized board with the same layout as the specific board used in this implementation, these variables can be tweaked to enable the program to run successfully. No RAPID workobjects are used to model the board or the piece. The benefit from using a workobject for the board would be minimal, as the board does not move around

during the execution of the program. And for the piece, the gripper always aims to grip the piece at the same location, meaning that no benefit would be gained from modelling it as a workobject.

5.4 Setup and calibration

To help achieve precise position of the center target an additional calibration object, presented in figure 8, was produced. This table shaped object is set on top of the board with the piece placed on top of the object, at the marked position in the middle. Then the robot can be moved into position to grip the piece, and the position is then saved as the center. The approach of using a separate calibration object instead of offsetting the center from one of the storage positions was chosen to keep the program as readable as possible. Figure 9 shows the robot at the desired position during calibration.

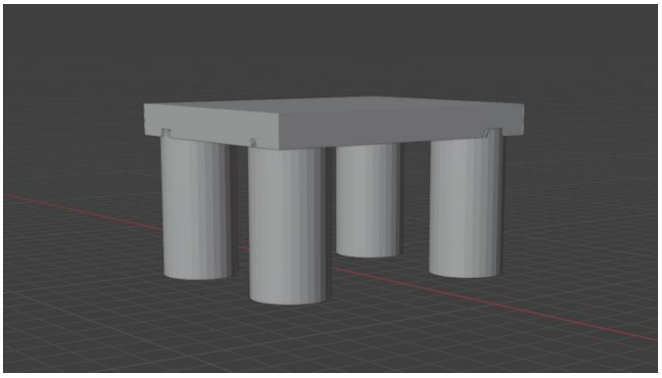

Figure 8. *Blender model of the calibration object.*

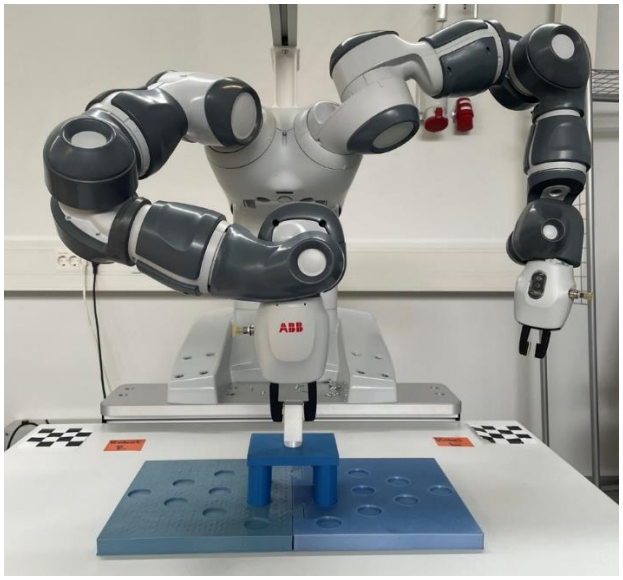

Figure 9. *YuMi during calibration of the center target at FAST-Lab.*

Before starting the program in addition to teaching the center target to the controller, the piece should be placed in the left most storage position at grid coordinates 0, -4. The storage position 0, -4 is where the pieceCurrentRow and pieceCurrentCol variables get initialized to at the beginning of the program execution. The approach of using one of the storage positions to hold the piece before the start of the program was chosen as it keeps the setup simple and the code easily understandable.

6. CONCLUSIONS

The goal of this thesis was to serve as an introductory overview of the design and implementation of robotic solution. In the literature review section, some of the current uses for robotics are highlighted, along with the differences in classifications of robots. After the literature review, the thesis explores the process of coming up with a robot solution for the given problem.

The process started with understanding the problem and gathering the requirements for the solution. Next in the design phase the objective was to explore the possible ways to design the solution to match the gathered requirements as well as possible. Finally in the implementation phase the solution was implemented on the actual robot in FAST-Lab. Overall, going through the listed steps of introducing a robot solution produced a sufficiently working solution for the given task.

With no prior experience in robot programming the process provided valuable skills in the field, such as planning the optimal route for an end effector to take and writing reusable code by calculating offsets from a reference position. Also, the experience gained in modelling objects and successfully 3D printing them has later proven a useful skill. Ultimately the most valuable lesson learned has been in the importance of managing one's own work.

6.1 Future work

Even though the solution produced for the thesis is functional and fills the requirements for the task, some modifications could make the solution more useful for actual use. Utilization of the Smart Gripper camera to calculate in which storage locations are currently occupied would go a long way to make the solution less arbitrary. Another way to modify the program could be to utilize user- or digital inputs instead of the randomly generated storage positions as targets for the piece.

APPENDIX A: RAPID PROGRAM

```
MODULE PickAndPlaceDemo
    !The reference point of the program, stationed above the center of the board.
    CONST robtarget Center:=[[399.650055016,0.079793468,99.999893924],
        [0.000000011,0.707106703,0.70710686,0.00000017],[1,-2,-2,4],
        [-108.380005898,9E+09,9E+09,9E+09,9E+09,9E+09]];
    !Constant properties of the board.
    CONST num ROWGAP:= 25;
    CONST num COLGAP:= 43.3;
    CONST num MIDGAP:= 35.1;
    !Default position of the piece.
    VAR num PieceCurrentRow := 0;
    VAR num PieceCurrentCol := -4;
    !***********************************************************
    !
    ! Module:  PickAndPlaceDemo
    !
    ! Description:
    !   Solution demonstrating a simple pick and place task with randomized target locations.
    !   Platform position incase it gets moved around: (-200, 0, 0)
    !   Piece position in leftmost slot (row = 0, col = -4): (390, -175, -35)
    !
    ! Author: Lauri Nuutinen
    !
    ! Version: 1.0
    !
    !***********************************************************


    PROC main()

        VAR num TargetRow;
        VAR num TargetCol;
        VAR num YCoord;
        VAR num XCoord;

        Reset Gripper;
        WaitTime 1;
        MoveJ Center,v200,z10,Servo\WObj:=wobj0;

        WHILE TRUE DO

            NavigateTo PieceCurrentRow, PieceCurrentCol; ! Move to pickup
            PickPiece; ! Pick

            !Generate random target location.
            TargetRow := GenerateRandRow();
            TargetCol := GenerateRandCol(TargetRow);

            NavigateTo TargetRow, TargetCol; ! Move to dropoff
            PlacePiece; ! Drop

            !Save piece storage position.
            PieceCurrentRow := TargetRow;
            PieceCurrentCol := TargetCol;

            !Return to center.
            MoveJ Center,v200,z10,Servo\WObj:=wobj0;

            !Wait for a few seconds to simulate downtime between tasks.
            WaitTime 3;

        ENDWHILE
    ENDPROC
```

```
76      FUNC num GenerateRandCol(num row)
77          !Generates random column number depending on chosen row number.
78          VAR num RandCol;
79          VAR num ColSign;
80          IF row = 0 THEN
81              !Col: -4, -2, 2, 4
82              RandCol := 2 + round((Rand()/RAND_MAX) \Dec:=0)*2;
83          ELSEIF row = 1 OR row = -1 THEN
84              !Col: -3, -1, 1, 3
85              RandCol := 1 + round((Rand()/RAND_MAX) \Dec:=0)*2;
86          ELSEIF row = 2 OR row = -2 THEN
87              !Col: -2, 2
88              RandCol := 2;
89          ELSE
90              !Col: -1, 1
91              RandCol := 1;
92          ENDIF
93
94          !Randomized sign
95          ColSign := 1 - round((Rand()/RAND_MAX) \Dec:=0)*2;
96          RandCol := ColSign*RandCol;
97
98          RETURN RandCol;
99      ENDFUNC
100
101     FUNC num RowToDis(num row)
102         !Function for calculating y coordinates based on given row.
103         RETURN row*ROWGAP;
104     ENDFUNC
105
106     FUNC num ColToDis(num col)
107         !Function for calculating x coordinates based on given column.
108         VAR num sign := 0;
109         IF col<0 THEN
110             sign := -1;
111         ELSEIF col>0 THEN
112             sign := 1;
113         ELSE
114             RETURN 0;
115         ENDIF
116         RETURN sign*MIDGAP + (col-sign)*COLGAP;
117     ENDFUNC
```

```
119     PROC PickPiece()
120
121         VAR robtarget TargetPos;
122         VAR robtarget AbovePos;
123
124         AbovePos:= CRobT(\Tool:=Servo \WObj:=wobj0);
125         TargetPos:= Offs(AbovePos,0,0,-40);
126
127         !Gripper is always open when moving to pick location.
128         Reset Gripper;
129         WaitTime 1;
130
131         MoveL TargetPos,v50,fine,Servo\WObj:=wobj0;
132
133         Set Gripper;
134         WaitTime 1;
135
136         MoveL AbovePos,v200,z10,Servo\WObj:=wobj0;
137
138     ENDPROC
139
140     PROC PlacePiece()
141
142         VAR robtarget TargetPos;
143         VAR robtarget AbovePos;
144
145         AbovePos:= CRobT(\Tool:=Servo \WObj:=wobj0);
146         TargetPos:= Offs(AbovePos,0,0,-40);
147
148         MoveL TargetPos,v50,fine,Servo\WObj:=wobj0;
149
150         Reset Gripper;
151         WaitTime 1;
152
153         MoveL AbovePos,v200,z10,Servo\WObj:=wobj0;
154
155     ENDPROC
156
157     PROC NavigateTo(VAR num row, VAR num col)
158         !Fuction for giving linear move commands to coordinates of given storage position.
159         VAR robtarget NavTarget;
160         VAR num XCoord;
161         VAR num YCoord;
162
163         XCoord:= RowToDis(row);
164         YCoord:= ColToDis(col);
165
166         NavTarget:= Offs(Center,XCoord,YCoord,0);
167         !The zone of this movement is set to fine to CRobT in pick and place procs.
168         MoveL NavTarget,v200,fine,Servo\WObj:=wobj0;
169
170     ENDPROC
```

REFERENCES

[1] Duda J, Gąsior A. Industry 4.0: A Glocal Perspective. New York: Routledge; 2021.

[2] Robotics IIF of. World Robotics Report: “All-Time High” with Half a Million Robots Installed in one Year [Internet]. IFR Int. Fed. Robot. [cited 2023 Aug 23]. Available from: https://ifr.org/ifr-press-releases/news/wr-report-all-time-high-with-half-a-million-robots-installed.

[3] Graetz G, Michaels G. Robots at Work. Rev Econ Stat. 2018;100(5):753–768.

[4] Wallén J. The History of the Industrial Robot [Internet]. Linköping University Electronic Press; 2008 [cited 2023 Aug 24]. Available from: https://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-56167.

[5] Echelmeyer W, Kirchheim A, Wellbrock E. Robotics-logistics: Challenges for automation of logistic processes. IEEE; 2008 [cited 2023 Aug 23]. p. 2099–2103. Available from: https://ieeexplore.ieee.org/document/4636510.

[6] PANASIUK J, KACZMAREK W, SIWEK M, et al. Test Bench Concept for Testing of Gripper Properties in a Robotic Palletizing Process. Probl Mechatron Armament Aviat Saf Eng. 2022;13(2):51–64.

[7] Boothroyd G. Assembly automation and product design. 2nd ed. Boca Raton, FL: Taylor & Francis; 2005.

[8] Haidegger T. Autonomy for Surgical Robots: Concepts and Paradigms. IEEE Trans Med Robot Bionics. 2019;1(2):65–76.

[9] Elfasakhany A, Yanez E, Baylon K, et al. Design and Development of a Competitive Low-Cost Robot Arm with Four Degrees of Freedom. Mod Mech Eng. 2011;01(02):47–55.

[10] El Zaatari S, Marei M, Li W, et al. Cobot programming for collaborative industrial tasks: An overview. Robot Auton Syst. 2019;116:162–180.

[11] A review of mobile robots: Concepts, methods, theoretical framework, and applications [Internet]. [cited 2023 Sep 26]. Available from: https://journals.sagepub.com/doi/epub/10.1177/1729881419839596.

[12] Jaulin L. Mobile Robotics. John Wiley & Sons; 2019.

[13] Guo Y, Liu H, Fan X, et al. Research Progress of Path Planning Methods for Autonomous Underwater Vehicle. Chen H, editor. Math Probl Eng. 2021;2021:1–25.

[14] Cohen Y, Shoval S, Faccio M, et al. Deploying cobots in collaborative systems: major considerations and productivity analysis. Int J Prod Res. 2022;60(6):1815–1831.

[15] Gonzalez-Aguirre JA, Osorio-Oliveros R, Rodríguez-Hernández KL, et al. Service Robots: Trends and Technology. Appl Sci. 2021;11(22):10702.

[16] Kachouie R, Sedighadeli S, Khosla R, et al. Socially Assistive Robots in Elderly Care: A Mixed-Method Systematic Literature Review. Int J Hum-Comput Interact. 2014;30(5):369–393.

[17] 3D -tulostimet | FabLab Tampere | Tampereen korkeakouluyhteisö [Internet]. FabLab Tamp. [cited 2024 Jan 11]. Available from: https://sites.tuni.fi/fablabtampere/laitteet/3d-tulostimet/.