UiT The Arctic University of Norway

Faculty of Engineering Science and Technology (IVT)
Department of Industrial Engineering

Digital Twin framework for Extended Reality, by integrating Nachi and SCARA Robot in XR-lab

Digital Twin in Immersive workspace

Stud. Tech. Karl Albin Vilhelmsson

Master's thesis in Industrial engineering, INE-3900, May 2022

Abstract

Digital Twin is a tool used for monitoring systems and controlling systems at a distance; the twining of systems has been around ever since NASA's Apollo mission; creating a digital twin lets people monitor the system in real-time. A digital twin also allows for modeling, configuration, testing, optimization, and research under dynamic circumstances; it also provides for hazardous free training on systems that can be hard reach, i.e., offshore. Extended reality is a gathering term for all virtual environments VR, AR, MR, and shared VR, the last one allowing a team to monitor a system together without the need for a headset displaying the information. UiT Narvik has an XR lab based on shared VR; here it is possible to create a collaborative environment around a digital twin. This project is centered around creating a digital twin in the XR lab and making a know-how framework to facilitate a base for more digital twins. The thesis has resulted in two digital twin models that can be used in the XR lab, the robots are located at UiT Narvik, and a proposed method for connecting has been made. The models are created in Unity based on the URDF for each robot; controls have been established for controlling the robots inside the XR environment at the same time as controlling the XR lab. The proposed method for communicating with the robots is to use the Open Sound Control protocol and bridge it with python to the OPC UA server connected to the robots. The know-how framework for integrating a digital twin is presented in the result part of the report.

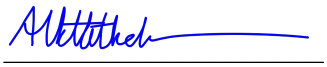
Keywords: Digital Twin, Extended Reality, XR, OSC, Robot, Extended reality, Kinematics, Unity, igloo vision

Acknowledgment

I want to thank Doctoral research fellow Halldor Arnarson for providing me with such a challenging yet exciting master's thesis. The process was frustrating and rewarding when the development of the models finally worked.

Further on, I would like to thank Prof. Wei Deng Solvang for the help in defining the thesis and for being my supervisor. I would like to thank Halldor again for being my co-supervisor and providing me with invaluable support and guidance, which would not have been completed without this thesis. Marius Wang, who knows the XR lab, has provided his expertise in how it worked, and without his knowledge, it will not have been finished in the XR-lab

I also would like to thank my fellow students for their good discussions regarding the thesis and good atmosphere. I also thank my friends for their support during my whole study, especially during my thesis work.

Sign. 

Narvik 16.05.2022

Table of Contents

List of Figures

1 Introduction

A Digital Twin (DT) is a digital representation of a physical system; this can be represented by a model or a fully connected twin in sync with the biological system. A DT can be utilized for different things throughout the life of the system it is twinning; it will range from optimization before the system is fully operatable to evaluating and verifying the system. When the system is fully operational, the DT can be used for monitoring, control, planning, and performance prediction. Further, during the entire lifetime of a system, services are needed. Here the DT can be used for predictive maintenance, fault detection and diagnosis, monitoring, and performance prediction. This makes DT an excellent tool for systems that can sometimes be hard to monitor or access. [1, 2]

Extended Reality (XR) is a gathering term for computer-generated realities; those are; Virtual Virtual Reality (VR), where the user uses a headset and is closed off from the surrounding environment, Augmented Reality (AR), where a set of glasses are used to display information and graphics for the user, and Mixed Reality (MR) where an object is projected in the "real world" letting the user asses it and "spin it around" and interact with the projected physical object. XR gathers these different terms but is not limited to them as this is an evolving field of study. The other technologies are used in various ways in the industry, such as training, making information easier to access, and planning production.[3, 4]

Joining these two topics together to create a DT in an XR environment will benefit the industry working with robots and other complex systems. Integrating a DT in the XR-lab at UiT will create a collaborative space around the twin for development, studying, and an exciting learning platform.

1.1 Background

UiT Narvik built an XR-Lab in February 2021. The room measures 10 x 7.6 meters; it has 12 projectors that project images on all walls and the floor and a 7.0 surround audio system. It brings the opportunity to create an understanding of using digital twins in an XR environment. In this project, the tasks are related to integrating existing robots (Nachi and SCARA robots) into the digital twin model in XR-lab. However, the lack of current software and hardware is the most significant challenge to be met during the project.

The result of this project will contribute to building the technical framework and know-how for integrating more physical objects and systems into the extended reality environment. The

finished project will also provide a base for the training of personnel and an engaging learning environment for the students at the department.

1.2 Objectives and scope

The objectives for this thesis are as follows relating to getting a DT into XR; the focus is on the two existing robots in the lab at UiT Narvik, the Nachi MZ07 and an older SCARA robot. Therefore, the objectives of the master thesis will be as follows:

1. Create Digital twins of Nachi and SCARA robots in XR-lab that can be controlled in the XR environment, and necessary data can be retrieved from the model to study the twin's behavior.
2. From the created digital twins, create a framework that will make it easier to import and integrate other robots into extended reality

Connected to the objectives is the scope of the work that is being done during the project, and the scope of this study is to create a framework for integrating a DT of a robot into XR; the framework will be made in Unity. In Unity, a scene for importing a robot will be created. Also, some controls for controlling the DT will be created.

To learn the required knowledge to create a framework, two of the existing robots in the UiT laboratory will be imported into Unity. Also, scripts for controls will be made. These will be integrated and tested in the XR-lab at UiT.

2 Literature review

The first part of the thesis was a literature study and time allocated to gaining knowledge for performing the project. Therefore, a short literature study on the concepts in the thesis was done, and a search for similar work. Since the time of the first part was also used for learning and gathering information about how the task could be solved, a lot of the time was spent learning Unity 3D and how it could be used in the project.

2.1 Digital Twin

Brief and straightforward, a DT is defined as a virtual model of a physical system. Different parts of the industry do not represent digital twin the same; they all agree that a DT is an exact model of the system; some differences are that the DT can range from being a simulation model used for predicting and planning to a fully connected twin and the data exchange goes both ways affecting both the virtual and real systems. [1]

A DT can then have different applications at various life cycle phases; the phases defined by [2] are the design, manufacturing, service, and retirement phases. The digital twin will play different roles; in the design phase, the DT will be used for iterative optimization, data integrity, and virtual evaluation & verification. It is mainly used in the manufacturing phase for; real-time monitoring, production control, asset management, production planning, workpiece performance prediction, human-robot collaboration and interaction, and process evaluation & optimization. It will be used during the service phase for; predictive maintenance, fault detection & diagnosis, state monitoring, performance prediction, and virtual test. There are no applications assigned during the retirement phase, and it is a possible research area. [2]

Another way of looking at DT is the different characteristics they can have; in [1] 13 have been identified. They are physical entity/twin, Virtual Entity/Twin, Physical environment, virtual environment, state, Realisation, metrology, Twinning, twinning rate, physical-to-virtual connection/twinning, Virtual-to-Physical connection/twinning, physical process, and virtual process. These sum up what an average DT has or partly is made up of.

2.2 Extended Reality

Extended reality (XR) can be seen as a gathering term for immersive technologies, such as Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR). However, XR is not

only limited to these immersive technologies, but these are the ones developed so far. XR is a relatively new term, and therefore the technologies are in constant development. [3]

XR sums up these three as an umbrella concept, unites where they differ and supports new technology. [3]

XR technologies are used in various ways in the industry. VR is used for training personnel, especially where training in real-life situations is dangerous and can result in fatal injuries. AR is used in the daily work in some parts of the industry to display and make information more visible and closer to the operator. MR is used for planning of production and training of personnel. [3, 4]

- VR is a virtual environment often generated by a computer to make the user experience something and interact with a system.

- AR is a technique where digital information overlays reality displaying helpful information to the user.

- MR is a hybrid of the systems mentioned, overlaying a virtual object into the real world, letting the user interact with the actual object and surroundings.

2.2.1 XR-Lab

The XR-lab in UiT Narvik is categorized as a shared VR environment; it is a collaborative workspace that can house groups of people sharing a virtual environment. The shared VR space is made up of a 10x7 meter big room with projections on all walls and the floor; this immersive workspace is a great working environment for collaboration and studying the behavior of a more extensive system. To analyze the behavior of a real-time system, a connection first has to be made to the system.

Figure 1 XR-lab at UiT[5]

2.3 Digital Twin in Extended Reality

In a literature review about mixed reality and digital twin, it is found that the topics regarding this subject are focused mainly on the autonomous driving vehicle, BIM, and the construction industry. [6]

A study presents a framework for creating extended reality system development in manufacturing; a five-step process is explained based on case studies. The proposed framework does not include digital twin but includes manufacturing systems in extended reality. [7]

Another study is looking at DT based on BIM used for maintenance in XR. This study looks at how XR technologies can help O&M in buildings and make the work task more accessible and more precise documentation with the help of a DT. [8]

A study presents a generic framework for BIM-DT using reality capture to extended reality. The framework will act as an automated construction progress monitoring system called DRX, built upon various XR technology, BIM, and DT. [9]

2.4 Software

2.4.1 Unity

Unity game engine is a software that was first and still is used for game development; it was founded in Copenhagen in 2004 as a software platform for video games. Through the years, Unity has developed into much more than a game engine; today, Unity is a 3D building platform for both industries and games. Vice president Sylvio Drouin says that Unity wants to be the 3D operating system of the world. Almost half of all games in the world are built with Unity. Around 60% of all augmented and virtual reality experiences for industries like movies, architecture, and automotive are made with Unity making it one of the biggest 3D making platforms in the world. [10]

When creating something in Unity, this is done by adding GameObjects to the 3D world; the GameObjects can be empty, a cube, a cylinder, or a robot link. The GameObjects get different components that determine the object's properties, such as Articulation Body, Joint, material, or color. In this project, the GameObjects rely heavily on the Articulation Body. Objects can also be assigned scripts that control behavior in gameplay, such as movement, changing color, and displaying things when activated. GameObjects also have relations binding them to each other in a parent-child hierarchy. How this is structured can be seen in Figure 2

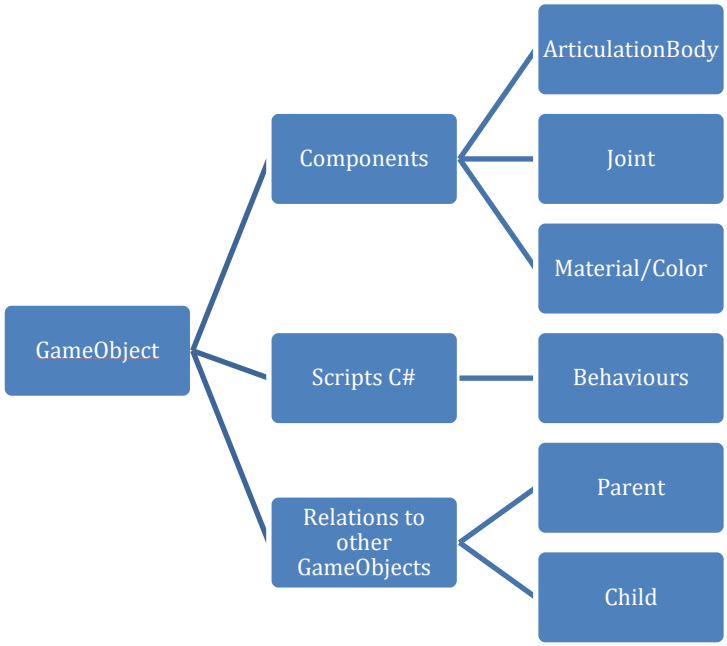


Figure 2 Unity buildup of 3D objects

Unity Technologies is developing a package for the 3D environment called URDF-importer; this package will allow us to import a robot based on a URDF (Universal Robot Description File). A drawback of this package is that it is still under development. Unity is working on multiple things to make it work better; the package is received through GitHub and not the Unity asset store as it is not fully developed. [11]

The toolkit used for projecting the scene into the XR-lab is delivered by igloo vision and is not public.

2.4.1.1 C#

Unity is based on the programming language called C# (C Sharp), which is popular in game development, mobile applications, and more. C# is developed by Microsoft and is running on the .NET framework. [12, 13]

2.4.2 Similar Software

2.4.2.1 Visual Components

Visual Components is a software developed for simulating factories and building models and twins. It has an extensive library of leading manufacturing equipment; it is fully integrable with python and predefined connections. This is great as a manufacturing simulation tool, but it is limited to the camera packages delivered by Visual Components. [14]

2.4.2.2 Game4Automation

Game4Automation is a German editor extension for Unity; the extension packages include various tools to create models of factories and digital twins in Unity. It is a framework specially developed for Unity that may be used for future use; it has a price tag of 699 €, which made it too expensive to use in this project. It provides many features researched in this project and could be a great option to try for a more stable version in the XR-lab. [15]

2.5 Robots

The DT focus will be on Industrial Robots (IRB). To create a framework, it was decided to use two of the existing IRBs in the lab at UiT Narvik. These are a Nachi MZ07 and an older SCARA robot. The IRBs will be presented with characteristics, drawings, and the information needed to build the twins in this section.

2.5.1 Nachi MZ07

Nachi-Fujikoshi is a Japanese tech company founded in 1928; they started as a domestic company to manufacture cutting and machine tools. [16] They then developed their first high-performance IRB in the 1960s. They deliver high-quality robots for various palletizing, packing, loading/unloading machines, tools, etc. They are used in automotive, electronics manufacturing, and steel construction industries. [17]

The Nachi MZ07 is a small to medium-sized IRB consisting of six rotational joints, given it six Degrees Of Freedom (DOF) and a total reach of 723 mm, and it can lift up to 7 kg. Nachi classifies it as the fastest IRB in its class and specifies repeatability of ±0.02 mm. The MZ07 is possible to mount at different locations such as the floor, wall, and inverted; as for now, it is mounted on a trolley in the lab as part of a Reconfigurable Manufacturing System (RMS). [18] The robot can be seen in Figure 3.

Figure 3 Picture of Nachi MZ07 [18]

2.5.2 SCARA

A SCARA robot is characterized by three rotational joints and a prismatic joint; it works all over the x-y plane. The prismatic joint moves up and down in the z-direction with additional rotation around the z-axis. It is a versatile robot for moving things in a plane, but it can also be

used for packaging. [19] Figure 4 shows the articulation and the frames of a general SCARA Robot

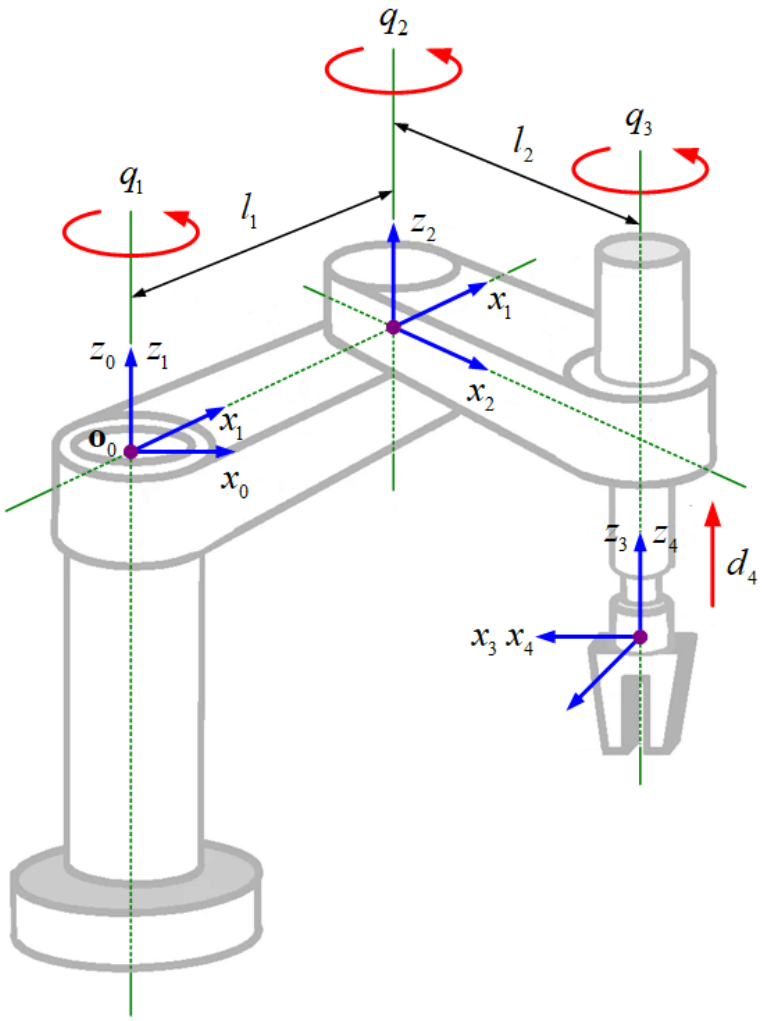


Figure 4 Articulation of SCARA robot [20]

2.6 Open Sound Control

Open Sound Control (OSC) is a communication protocol first used in real-time between music players and hardware, such as synthesizers and computers. It was introduced in 1997, replacing the old MIDI communication. Now OSC can be used for various data exchanges, as it is fast, reliable, and an entirely open communication protocol. OSC uses a pattern-matching language, making it versatile when sending many different data types over the same connection for controls. [21]

OSC relies on a connection defined by an IP address. On this IP address, a transmitter and receiver port have to be defined for each object to be able to exchange data. When the ports are defined, all messages need an address starting with a slash typical message can look like

"/Program/Part/Component "0.0" float" for sending a float value that is to be changed; it is possible to send a variety of different messages through this messaging protocol. [22] Figure 5 shows how the data exchange with ports and IP addresses works with OSC.

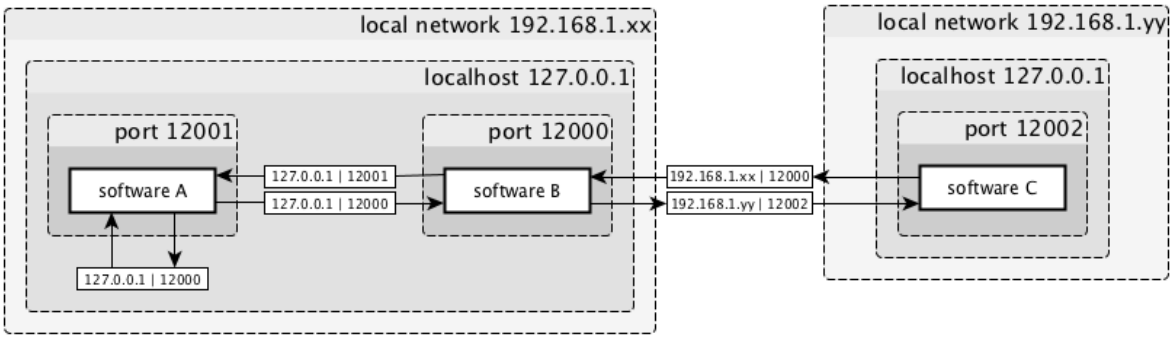


Figure 5 How the OSC transmits messages [23]

3 Problem formulation and Scope

Digital twin as a concept has been around since 2010, when NASA introduced it to improve the physical model simulation. Many studies have been performed to help enhance the idea of DT simulation, such as connectivity and two-way communication. The topic has then evolved to VR, and through different software, better simulation and more integration. The key to a good DT is that it is as exact to the physical system as possible; the simulation should behave as the physical system should have done. However, with the rising technologies of XR, a DT can be used and collaborated by a larger group or team in the virtual space and be big enough to enable good teamwork. The problem with hardware-software incompatibility comes in, and it will be time-consuming and costly to integrate a DT model in XR.

3.1 Reduced Scope

During the execution of this project, some problems have come up. Therefore, it is proposed that the scope be reduced to meet the project's time constraints. The time allocated for this project is limited to 810 hours. With the problems of importing the robot models into Unity, it has been decided to focus on the main objective of the project of getting a working model of the two robots into the XR-lab.

Therefore, the scope is reduced to that this report will be the framework for integrating DT in XR environments. The framework will be know-how on how to incorporate a DT in the XR-lab at UiT Narvik for further development, and the focus after getting the models in the XR-lab will be to connect them to the physical system.

Pick and place activity will not be performed as this is not necessary for making the articulated robot work in XR.

4 Digital Twin model

This section will present the process of creating the digital twin model in Unity; it will also show the struggles of making it work.

4.1 Import URDF of the robot

4.1.1 Use URDF-importer

To import the robots into Unity, the URDF-importer developed by Unity will be used; this package comes with scripts for robot control and components for each part. To start, the asset first needs to be imported into the Unity project; here, the git URL for the URDF-importer was used; when using the git URL, Unity will download the package, and after it can be imported into the Unity project, the package manager can be seen in Figure 6. The git URL was received from the Github page for the URDF-importer. [11]

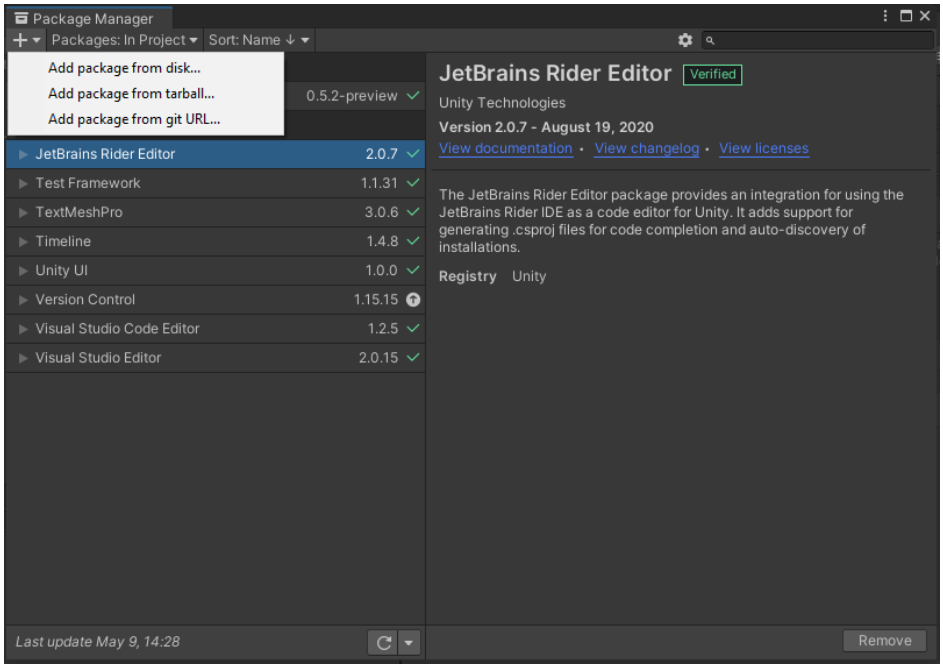


Figure 6 Unity package manager

After the URDF-importer package has been added to the project, the file hierarchy must be edited so that the importer can read the URDF-files (with meshes and visuals) that the URDF exporter in Solidworks puts out that are not correct for the importer to read. A comparison of how the file hierarchy can be seen in Figure 7; here, it can be seen that for the URDF-importer to be able to read the files, the URDF file needs to be in the root, and the meshes folder needs

to be in a folder named the same as the URDF file. What was done here was that the mesh folder was copied into a folder with the same name as the URDF file in the URDF folder from Solidworks, and the rest of the structure was kept as it was when outputted from Solidworks.

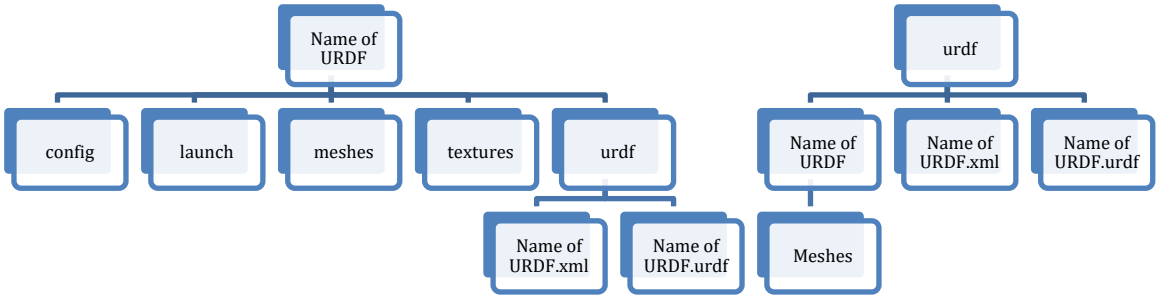


Figure 7 Left-File hierarchy from Solidworks, Right-File hierarchy needed to use URDF-importer

The final step for importing the robots to Unity was to utilize the URDF-importer by right-clicking the URDF file in the hierarchy; this can be seen in Figure 8, where we click on "Import Robot from Selected URDF file." After that, a new window will pop up with some options for importing the model.

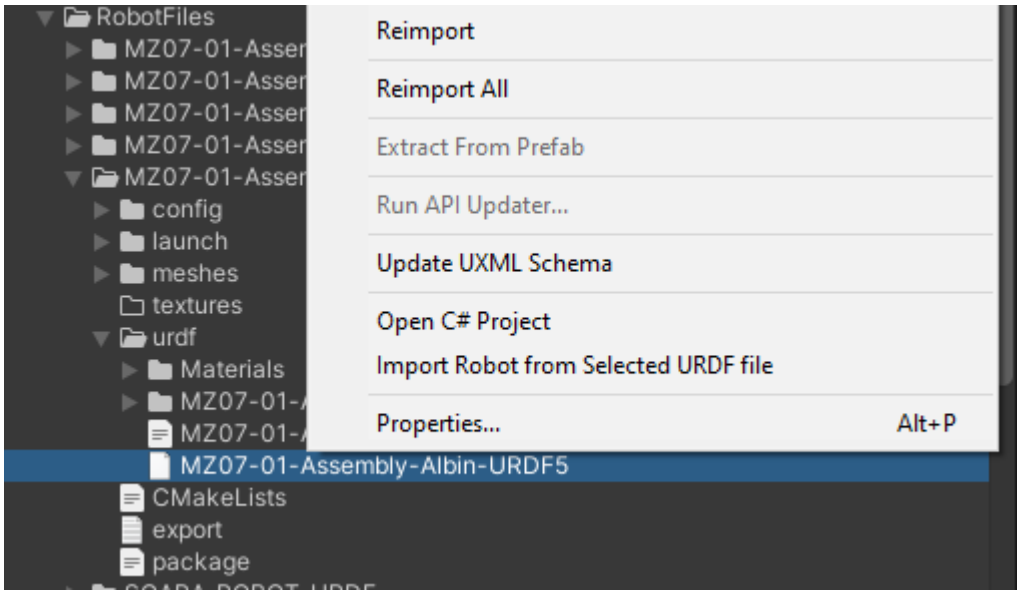


Figure 8 Picture showing menu of right-click on URDF file

These options are shown in Figure 9; here is shown that the options available are to change which axis the model should be aligned after; the URDF-importer will then convert the right-hand coordinate system of the original files to match Unity's left-hand coordinate system. The next option is the mesh decomposer; here, the options are VHACD (Volumetric Hierarchical

Approximate Convex Decomposition). This algorithm is integrated into the importer that creates convex hulls that make a more accurate mesh following the object's shape. This is a problem with the other option, the Unity mesh with a limit of 255 triangles, making the articulation body's behavior erratic and not precise for use. [24]

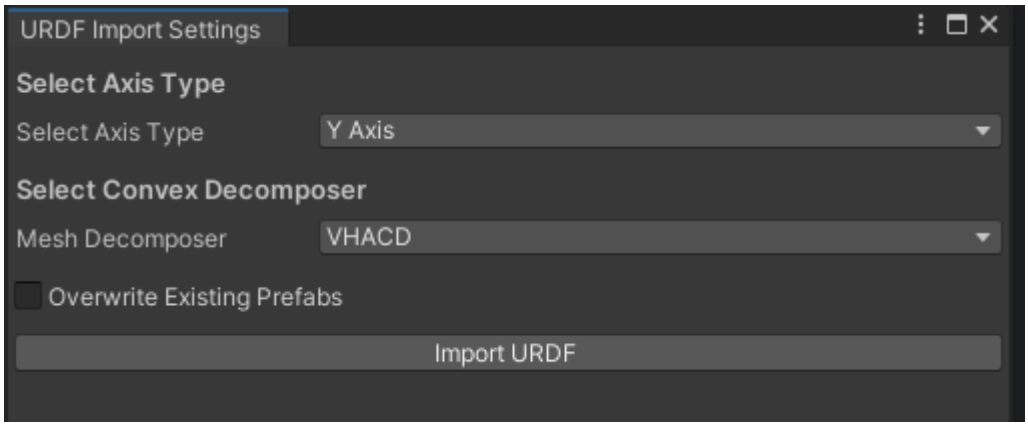


Figure 9 Settings for the URDF importer

The applied settings in this project are to use the Y-axis, and for mesh decomposer, the VHACD has been used; after these settings have been chosen, it is clicked “Import URDF,” and the model should appear in Unity, the problems accounted with this will be explained in the next section.

4.1.2 Import of the first model in Unity 3D

The first files used to import the URDF models to Unity were received from Doctoral research fellow Halldor Arnarson, and they were, as described in the pre-study, assumed to be correct. The URDF files were copied into the assets folder in Unity, and the methods in the last section were followed except for not knowing how the file hierarchy was supposed to be. It took a couple of tries before getting the correct file hierarchy. The error message indicated not finding the mesh files, so it could not create the visual and collision meshes in Unity. When getting the file structure right, the following error message was that the URDF importer could not read the mesh files to create a mesh in Unity. This error message was very cryptical. To understand what was wrong, the importer needs to read an STL (Standard Triangle Language) file, which the mesh files were. A lot of searching was done to find the cause of the problem with the STL files, but there were no good answers when searching online. Therefore, a trial and fail method was tested by saving new STL files from Solidworks and altering the save settings for the file format. Figure 10 shows the settings window from Solidworks; what worked out was setting the output to Binary and replacing the mesh files in the original folder.

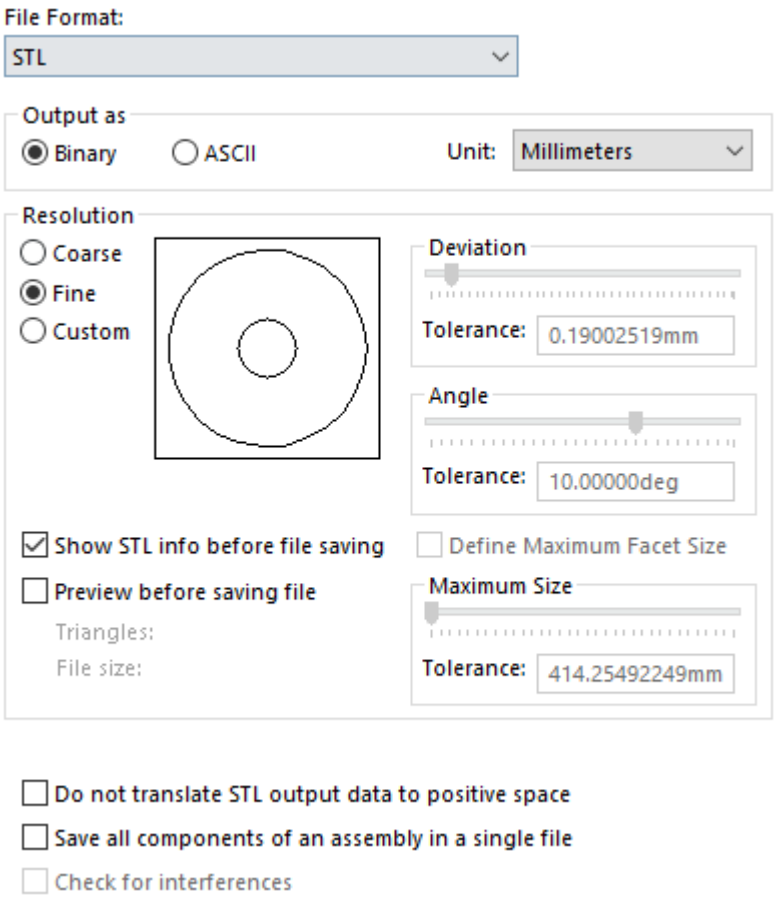
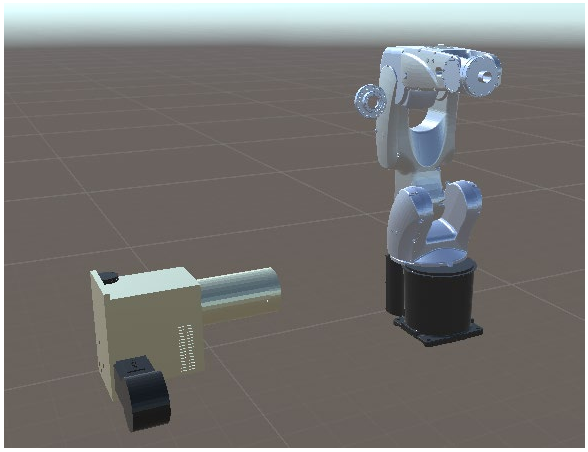


Figure 10 settings STL file

When the mesh files were replaced in the folder for that the importer could read, the importer was able to import the robots to Unity. However, all the links for both robots were not where they should be, shown in Figure 11, the SCARA links were positioned in a pile, and the Nachi links were of position. This was solved by moving the links in Unity to place them correctly; the links are organized in a parent-child hierarchy, so the movements of parts have to be started at the root of the robot. Figure 12 shows when the parts have been forced into a position relative to the last joint and structure of the real robot. As described in section 4.1.1, the URDF importer comes with a script for controlling the joint position of the imported URDF robot. In the video, it can be seen that this is not the case, as can be seen in https://youtu.be/wcy6UjKvSBQ; some of the links float away when the game is started, and the rest are unable to control as they slowly drift away from position and are not moving as they should, both for the prismatic and rotational joints, some of them are not moving at all either. The kinematics and the problem will be presented in the next section.



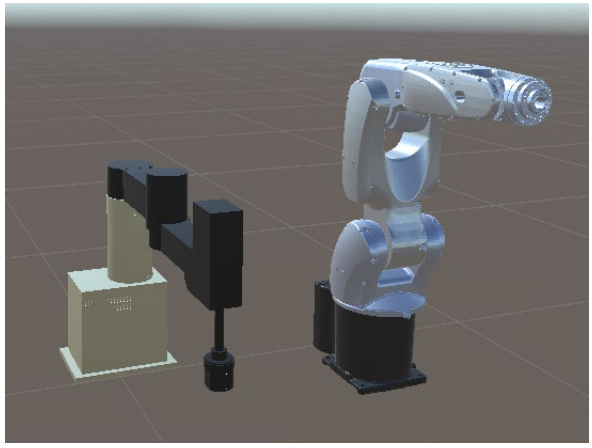


Figure 11 First import of robots

Figure 12 First Import moved into position

4.1.3 Kinematics of the first model

An essential part of a robot is kinematics; this defines how the robot moves and the properties of each joint. Therefore, this will also be a necessary part of the DT as this is to be an exact digital replica, if not the most crucial part. This section will describe how the kinematics of the first model was defined.

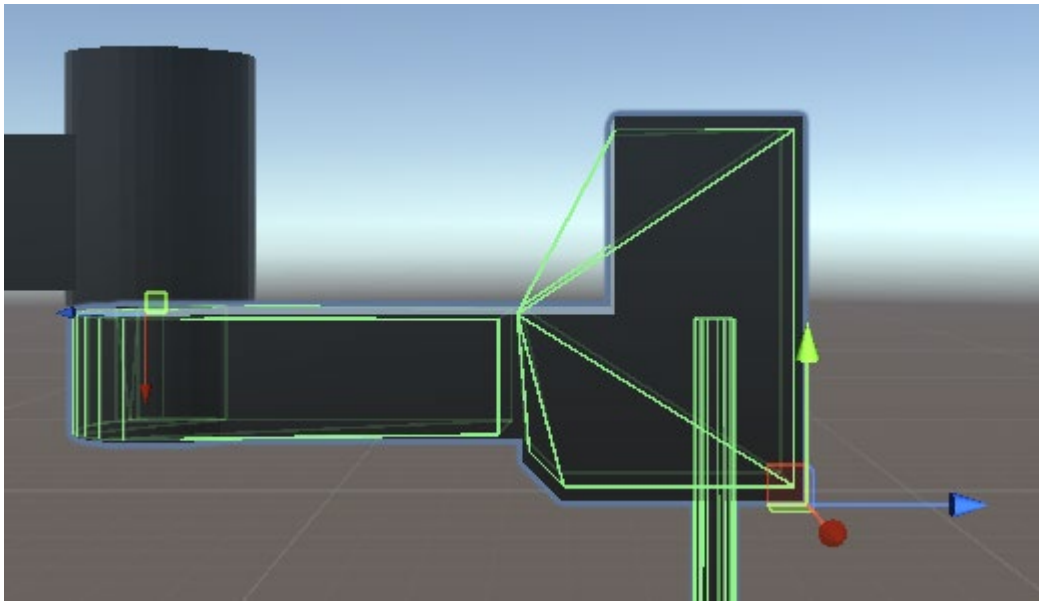


Figure 13 Second link SCARA robot mesh, coordinate system for placement, and anchor

Figure 13 shows the second link of the SCARA robot; it is possible to see the collision mesh for the link and the visual representation of the part. Two coordinate systems in the figure can also be seen; the big one is for the position, and the small one is for the anchor of the joint. The anchor of the joint needs to be positioned where the joint movement around the axis takes place;

it also has to be turned so that the GameObject component can translate the rotation around the axis with the controller.

Figure 14 Articulation body component

Figure 14 shows the GameObject component attached to the links and handles the physical properties and kinematics; here, it is possible to define the properties for the joint, limitation, damping, both angular and linear, and joint friction, and we will get information about the common. For the revolute joint, the rotation will be driven by what here is called X drive, which indicates that the anchor needs to be placed so that the process will come around the x-axis. For the prismatic joint of the SCARA robot, it is possible to choose which axis the joint will travel along, but the standard setting is to translate it along the x-axis. Therefore, all the anchors were aligned to the contact point to the parent link starting from the root base link and working up; all frames were rotated to make the rotation around the x-axis. This correction should make the robot move according to inputs with the keyboard. The video shows how it takes a long time for the robot to respond, and then it slowly turns away even with multiple inputs on the keyboard. The robot is not controlled in any way; this behavior was in different ways throughout trying in the project. Sometimes, it reached its destination as had been given through input. Then, it started resonating in larger and larger movements around that point with no explainable reason. This reaction was the same for both robots, even if the SCARA is shown in the video https://youtu.be/JBYxxb38KRg. It was confirmed through visual confirmation that the robots were moving correctly. It was just not controllable.

4.2 New digital model of Nachi

A problem that started was that the visual presentation of the Nachi MZ07 in Unity would not last when the program was closed and opened again; this can be seen in Figure 15 on the left, the collision meshes are shown to be still intact, and on the right, it can be seen that without the

meshes the robot is barely visible. Only a couple of panels are visible; here, it was tried to find if something was wrong with the files or Unity, but nothing worked. The same problems came up again after every try. At the end of March, at a status meeting with my co-supervisor Halldor, we decided that to solve this, new CAD models of the robot would be created as there had been problems with the files before.

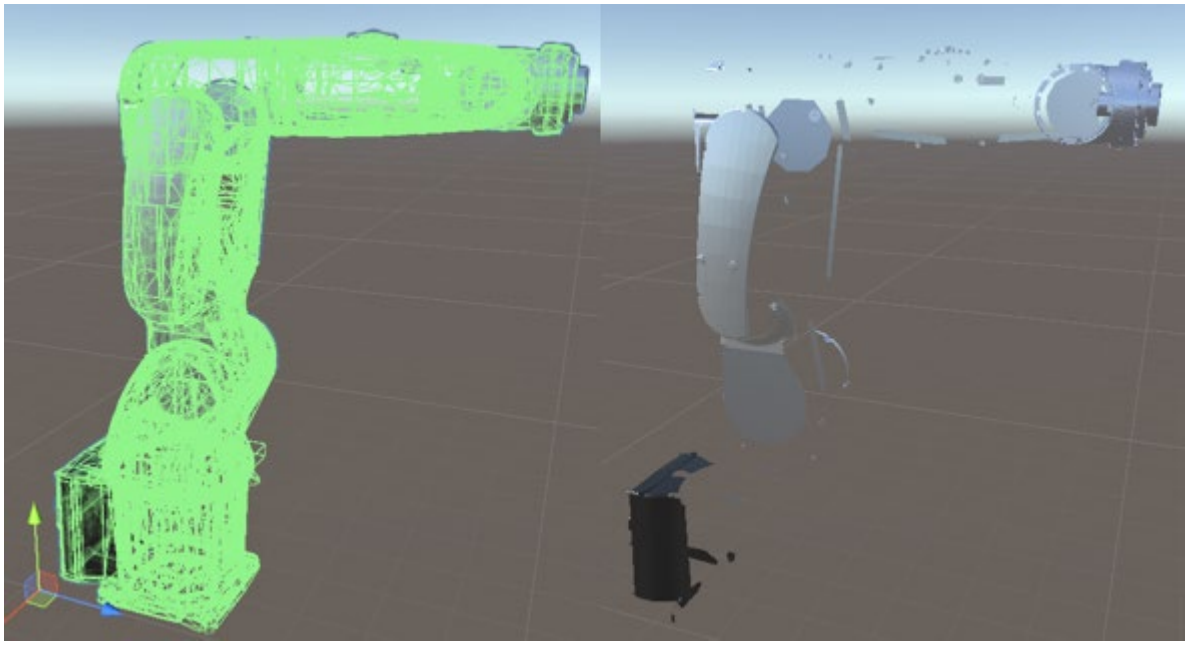
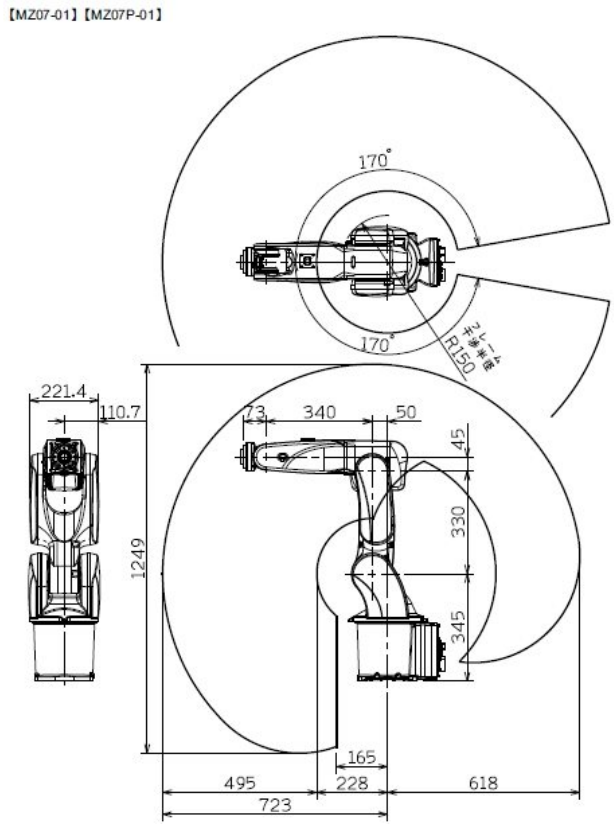


Figure 15 Problems with visuals for Nachi

The new CAD model was made in Solidworks following Nachi's specifications regarding the measurement of the robot and where the axis is placed in relation to each other and the base frame. [18] Not all measurements are provided, but the importance when creating the new model was that the axis is in the correct place and roughly looks the same; therefore, the new models were simplified to avoid complex meshes so that it will be easier to operate in the digital world. Figure 16 shows the dimensions of the MZ07, where the axis is placed about the centerline that goes through the base frame. The figure also shows the working envelope of the MZ07; this can be used to validate the model.

The drawing of the model started from the base link, and all parts were defined with an axis for the rotational movement and coordinate systems, as can be seen in Figure 17 showing link one the axis here goes through the y-axis. Adding the axes and coordinate systems was unnecessary, but it helped make the assembly of the parts easier.



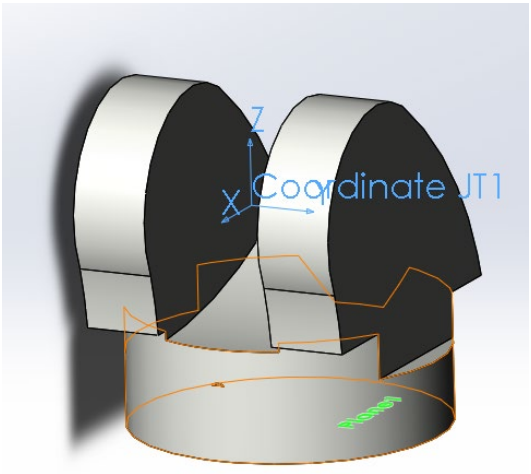


Figure 16 Left Nachi MZ07 Dimensions and Working envelope [18]

Figure 17 Right Link 1 Nachi MZ07

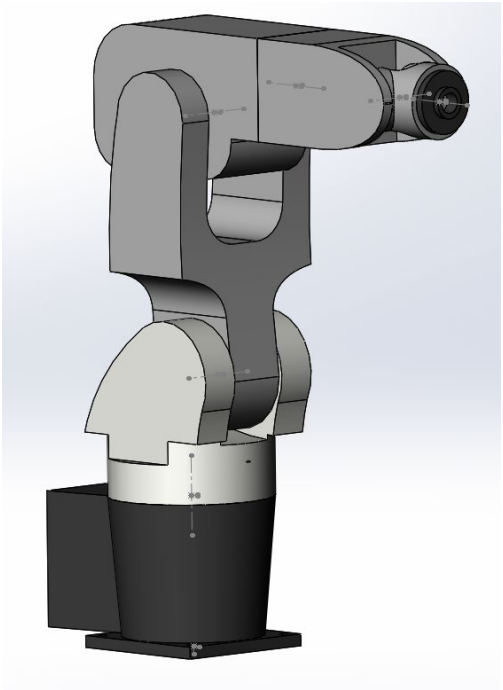


Figure 18 Complete assembly of new CAD model

Figure 18 shows the completed assembly of the new CAD model of the Nachi MZ07; as mentioned earlier, the model is made to represent the links and articulation of the robot and not visually represent it in the best way. The figure shows dotted lines and points connecting the links; this represents the URDF model created in Solidworks with the extension tool called "Solidworks to URDF exporter" [25]. The exporter can add joint limits, mass, and other relevant data to the links; it also generates meshes for the links and creates the folder system presented in section 4.1.1 that must be restructured to import the model into Unity 3D.

4.3 Second import of model in Unity 3D

The files were added to the Assets folder after the Nachi MZ07 was redrawn to fix Unity's visual problems. The file hierarchy was updated to match the URDF importer, as explained in section 4.1.1. When this was done, the robot was imported to the 3D environment. It was noted early that something was different with the model from the first one, not only the visual representation. As shown in Figure 19, a red ring appeared displaying the limits of each joint when it was chosen; another difference was that there was no need to restructure the links or the anchors when importing the model. At this point, the conclusion was that there was something wrong with the first used URDF files.

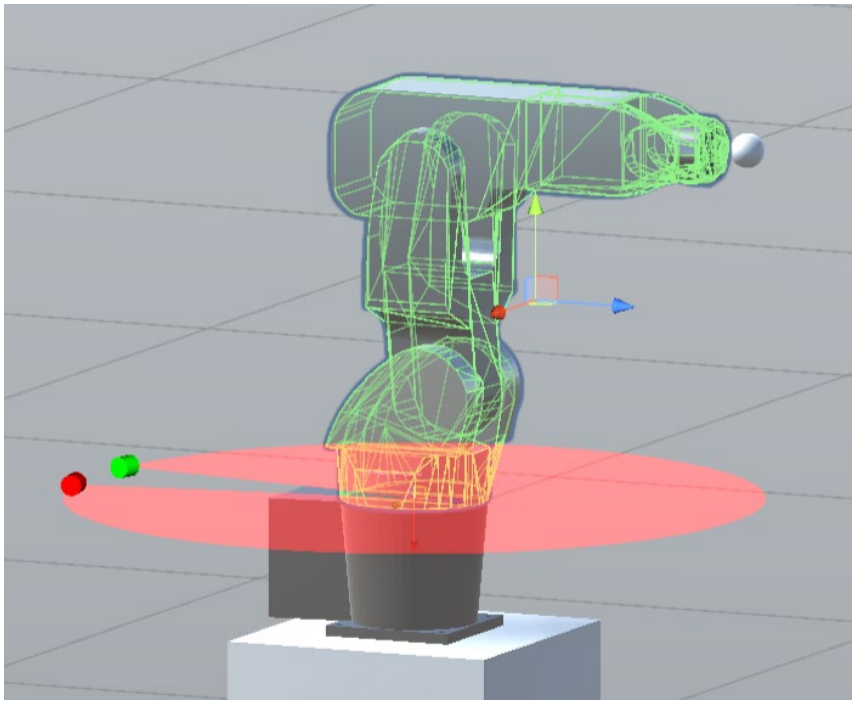


Figure 19 New model of Nachi MZ07 in Unity showing the meshes

As the new model of the Nachi resulted in a conclusion that the first URDF files were not as they should be, it was tried to save the SCARA robot. Here, it was tried to use the same Solidworks assembly as the first files to save the new URDF file for the robot.

The model imported on the new URDF files can be seen in Figure 20; here, we see the same characteristics as for the Nachi robot when it was imported with new files. The controller for the robot arms was also working with the new files. This is shown in https://youtu.be/2lf4IZjMCzI, where it can be seen that the robot moves to the desired location at a steady speed without delay and holds its position. Which link to move is chosen with the right and left arrow, and movement of the link is done by arrow up or down on the keyboard.

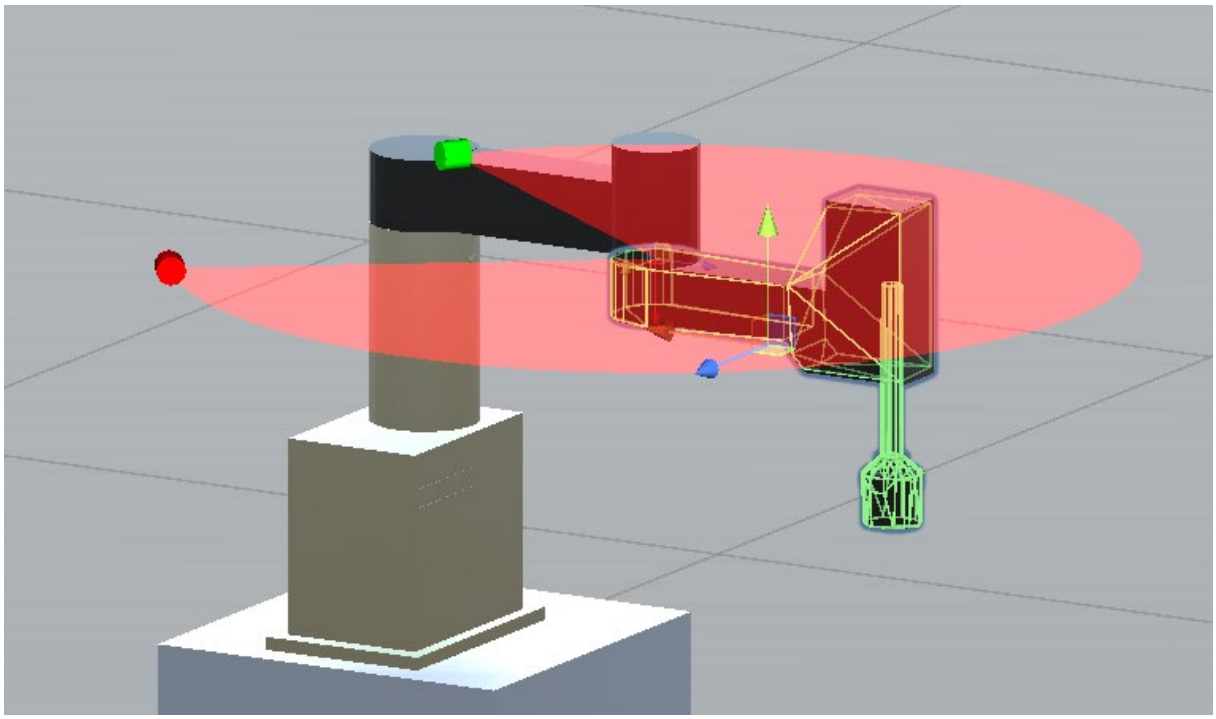


Figure 20 SCARA robot imported with new URDF file

However, the last two models still need to solve more minor problems. For the Nachi, the last link does not work here. The issue is a bit unclear. At first, it was believed to be caused by the meshes colliding with each other, as can be seen in Figure 21, the meshes did interfere with each other, and the proposed solution was to remove the center cylinder of link5 get rid of the problem. The problem continued after the tried solution; it only occurs when the simulation is started with the articulation body component activated for link 6. However, if the simulation is started with the component disabled and enabled after it is started, the robot does not crash. What happens when the robot is crashing is that some component is not withholding the laws of the physics engine in Unity resulting in the physics engine crashing. With that, the robot disappears, and over 1000 error messages in the console are coming up with the message "Invalid AABB aabb" which practically means that the physics engine has crashed. Often there is no clear solution to the problem. [26] Another error message coming up is that the part is too far away from the root, which a solution is impossible to find, but it is most likely an error in the URDF importer as this is still in a preview stage and not fully developed.

The problem with the SCARA is that movement of the prismatic joint is lagging, but this is most likely due to some interference that needs to be adjusted.

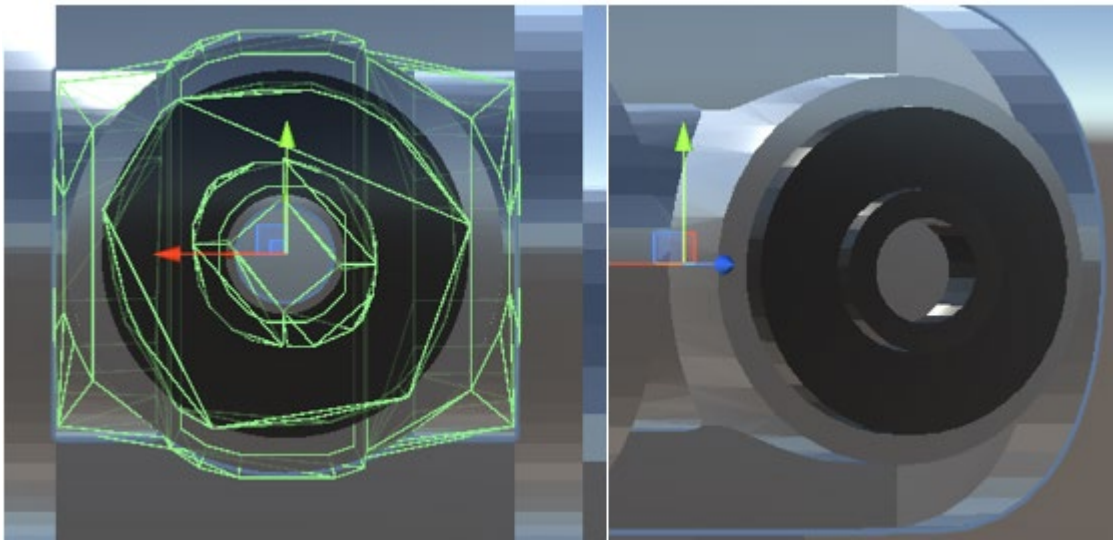


Figure 21 Left-Meshes of Link 5 and 6 interfering with each other, Right-Tried solution of removing a center cylinder of Link 5 to remove the mesh

4.4 End Effector movement

To control the robot more efficiently, it was proposed in the pre-study to add a controller to move the end effector in cartesian space. To do this, it is suggested to add a script to the root of the robot that tracks an object and matches the end effector to this object.

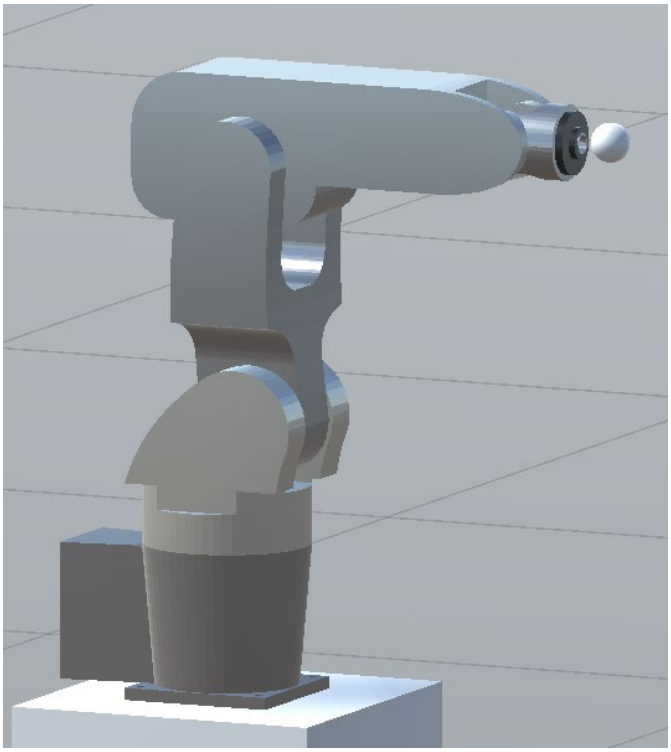


Figure 22 Nachi with a sphere in front of the end effector

In Figure 22, the Nachi robot can be seen with a sphere in front of the end effector; this sphere can easily be programmed to move with input controls. However, the plan of making the robot end-effector follow this object has not worked out during the project due to time constraints. The proposed method uses the gradient descent method to match the sphere's location in the virtual space. [27] It has been done before and should be a suitable method for creating a well-articulated robot in the virtual space.

5 Digital Twin in XR-lab

This section will describe what was needed to take the DT into the XR environment in the XR lab. The XR lab projects the environment in 360°. Therefore the first step to getting the DT into the XR environment is to add the necessary toolkit called “Igloo Unity toolkit,” which provides a 360° camera for taking up the whole environment and projecting it into the room. First, the prefab called igloo manager is added to the Unity scene; this prefab lets us integrate the location into the XR environment. The manager is seen in Figure 23. This allows us to adjust settings and add other cameras, information if the camera should follow an object, etc.

Figure 23 Igloo manager

Unity 3D has a predefined input system made for games, which is based upon an axis programmed to keys on the keyboard; for the up and down arrows and W and S, the vertical axis is mapped, and for left and right hands and A and D keys the horizontal axis is mapped. The input method for moving the player (camera) in the igloo toolkit is the vertical and horizontal axis, which crashes with the robot controls, which uses the vertical axis and left and right arrows to control the joint movements. The result here was that when moving the robot, the camera was moving as well, which can be seen in https://youtu.be/v2vargkBX9c.

The solution for this was to add a new axis to the input system called Up/Down; this axis will utilize the keys Q and E for the movement of the joint, and the buttons R and F will be used for changing the joint to move. The input keys can be seen in Figure 24, where the blue arrows indicate the camera's movement in the XR-lab, and the red arrows indicate changing of joint and plus and minus movement of the joint. This can be seen in https://youtube.com/shorts/BM8oTFSWUn0?feature=share and https://youtube.com/shorts/ajlMvXluzRA?feature=share.

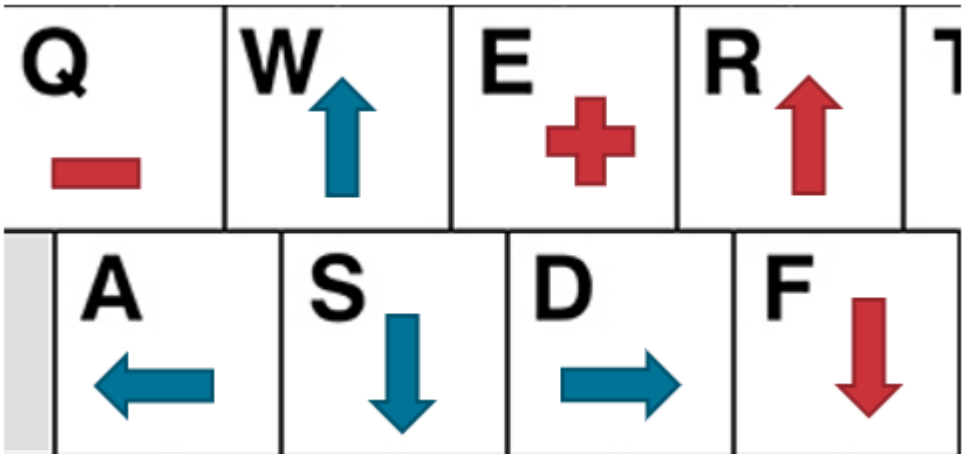


Figure 24 Input commands XR-lab

5.1 Connecting Digital twin

A big part of a DT is somehow connected to the physical system creating a bridge of data exchange, letting us monitor the system and even control from the digital version. The first planned method was to utilize the OPC UA connection as the robots are already connected to a server that utilizes this communication protocol. To use this in Unity, it is needed to add an asset to make this possible the plan here was to use the asset called GraphQL for UPC UA [28], which makes it possible to connect to the server. However, the first time it was tried to make a build with this extension, it failed multiple times; the XR-lab must run the program as a standalone build and not from the Unity editor; therefore, this method was discarded and changed over to the Open Sound Control (OSC) as the XR-lab already utilize this communication method.

5.1.1 Open Sound Control

OSC is most used for communication between music instruments such as synthesizers and computers but can also send other types of data between computers on the same network. The XR lab software and the toolkit include a library called extOSC, created by Vladimir Sigalkin

[29]. It is also a free asset for Unity to create simple communication with OSC. In Figure 25, a suggested bridge between the XR-lab and the physical system is proposed; here, it is proposed that the Unity build of the robot sends the extracted values for the joints position to a Python program that then sends the values to the nodes in the OPC UA server that after that sends the change in part to the physical system. This should be possible to reverse so that the robot can be monitored in the XR-lab when it performs tasks.

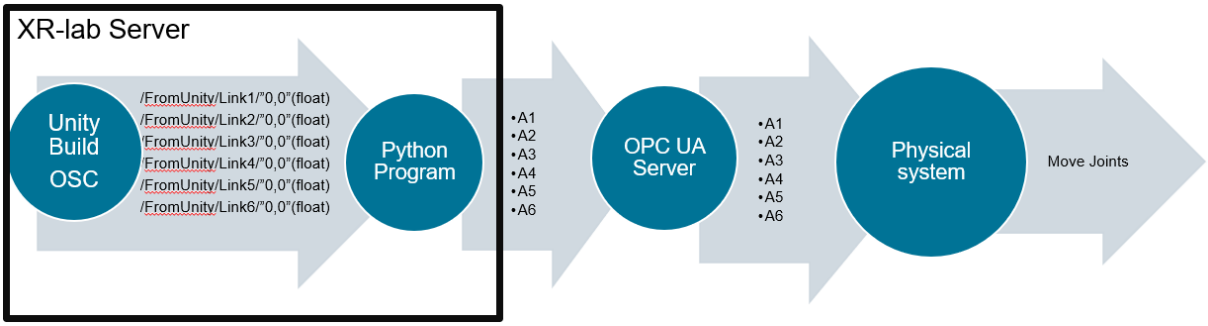


Figure 25 Structure of communication between XR-lab and physical system

As of now, python does not have a library to receive messages from OSC; it only can send messages. Therefore, it can be possible to receive data from the physical system and monitor it in the XR-lab.

To be able to send messages from Unity, it is needed to add a game object that handles the communication out from the program and a component to each link. The OSC Manager included in the exotic asset consist of a transmitter part and a receiver part, where an IP address on the same network is chosen and used for both. A port for each needs to be defined so that the program listening to that port can get the messages sent and that the program in the XR-lab listens to another port.

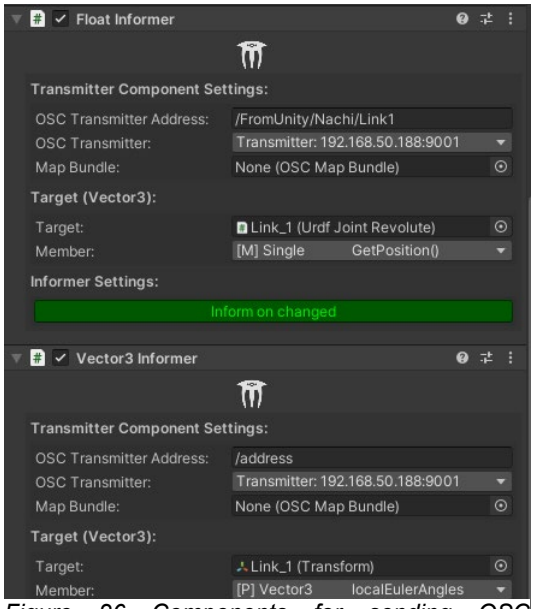


Figure 26 Components for sending OSC messages

Figure 26 shows the two components that can be used for sending position messages for each link, the one that gives the best result is the Float informer connected to the Script and gets the position in radians. The Vector3 informer sends out a vector with three values of the joint's local Euler angels that need to be converted in the python script. The Float informer is the one that gives the correct

value, but it gets disabled every time the simulation is started, and therefore, it cannot be used if not a method for turning it on during “gameplay” in the build. Therefore, Vector3 has to be utilized for now. For each transmitter component, there is an equal reiver component. The IP address is 192.168.50.188, and the transmitter port is 9001.

As can be seen in Figure 27, the message sent out has an identification address written like “/FromUnity/Nachi/Link2” that has been used to identify link two of the Nachi robot and that it comes from unity, for communication with the other way it will be possible to write ToUnity instead to identify the message.

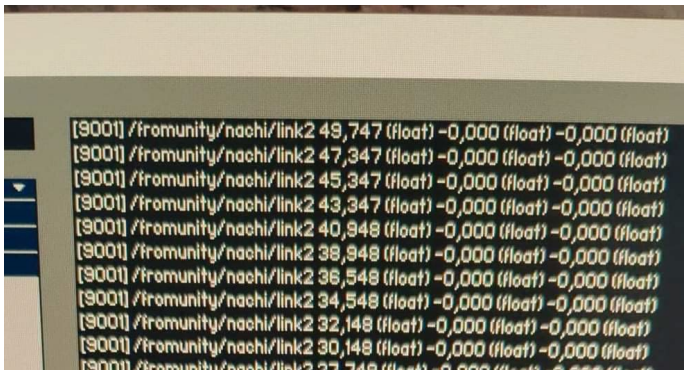


Figure 27 Picture of transmitted values from Unity

6 Results

The main objective of this project was to create digital twins of Nachi and SCARA robots in the XR-lab that could be controlled in the XR environment and that necessary data could be retrieved from the model to study the behavior of the twin. This has been done by integrating digital twins of both robots in Unity using the URDF importer. Two working models have been created for the XR-lab.

Easy inputs can control the models created with the URDF importer for the XR lab on the keyboard; the XR lab is controllable through the same keyboard without altering anything when the program is running.

Some issues led to a delay in creating the models in Unity 3D; therefore, the project's scope was reduced, and the reduced scope was fulfilled by bringing the models into the XR environment.

It is possible to retrieve data from the model through OSC that can be used to control the physical systems from the XR environment in the future. When this is utilized for communicating, it should be possible to have a two-way digital twin in the XR-lab; now, it will only be possible to make a one-way digital twin monitoring the physical system from within the XR-lab python does not support receiving messages from OSC.

The second objective of this thesis was that the create digital twins create a framework that will make it easier to import and integrate other robots into extended reality. This report acts as know-how to import other robots into the XR-lab, as it describes a method that can be used for creating other DTs of robots in the XR-lab. The step for creating other DTs should be as follows:

1. **Creating a Digital Twin model**
- Import URDF importer to Unity 3D
- Check URDF files; if uncertain, create new files
- Structure file hierarchy to match the one that is readable by the URDF importer
- Import URDF
- Create and build a scene around robots in Unity
- Adjust controls and validate that model is behaving like a physical system

2. **Adapt model for XR-lab**
- Integrate the igloo Unity toolkit
- Change axis and keycodes for controlling the robot to avoid interference with XR-lab controls

- Connect model with OSC and bridge with python to OPC UA server
- Test and validate the model with the physical system

The proposed framework can be utilized for every robot based on the URDF format for describing the robots when importing.

7 Discussion

This section of the thesis will discuss the results and methods used in this thesis. It will look at importing the model into Unity, integrating the model in XR, and connecting the model.

7.1 Digital Twin model

The process of creating the DT model in Unity was far from straightforward. It had a lot of problems delaying the progress of the project. The main issue with getting the model to work in Unity was unknown until the end of March; according to the plan, the DT model was supposed to be finished at the start of March.

The process of getting the files to work in the first place may have ruined them in the way of moving around the meshes and trying to get the correct file structure; another possible explanation is that when the meshes were swapped out to new ones saved from Solidworks one by one and not saved as a whole package may have been an imperfect solution to the problem of first getting them into Unity. And since the files for both robots were behaving the same way, there were no reference points for what was wrong and right in this case, for to understand that something is wrong, it needs to be compared to something right. Therefore, it was a bit lucky that the visual representation of the Nachi MZ07 was not correct, leading to redrawing of the robot in Solidworks, which resulted in new files that were different from the original ones. It is only because it was detected that the files were not working within the 3D environment of Unity.

For the first model, one of the main problems that were tried to be solved was the issue of joint control not working; the joints would start moving but would never stop. They would reach the desired theoretical location seen in the inspector window in Unity; the desired velocity was set to zero, but instead of stopping, the link started resonating around the point with bigger and bigger movements. This was unknown, but when importing new files, the problem was gone, and the joints were articulating their movements and stopping at the desired location.

An actual robot controller is a complex device with a lot of programming behind it, making the joints stop in time, so they don’t crash into each other; for the model, this is not the case as the game engine relies heavily on the meshes to know when to stop the model this does not work perfectly as to when a joint is run into another instead of stopping the joint is goes into the other parts of the robot making it shiver a bit until the joint is driven out from the position where it is crashing. The articulated body moves into the position it had before crashing with itself. A

solution for this can be to limit the joint's movement, but then the robot will not be able to reach the whole work envelope that it is defined for.

End effector movement without articulating the whole chain of the robot with a controller script is possible through the proposed method, it will take some time to develop, but it is believed that it could have been done in this project if it had not been for the problems importing the robot into Unity in the first place.

7.2 Digital twin in XR

After the robot model was done, it was adapted for the XR-lab; this was relatively quick, with the toolkit being intuitive and easy to handle. The camera package makes it easy to import the model to the XR-lab. There were a few problems with this. Importing DT models into the XR-lab can be quickly done; the problem is not getting the model into the lab. It had the robot controller not interfering with the camera controller in the lab.

To solve the problem of the camera controller interfering with the robot controller, the key codes and axis system were manipulated in Unity so that everything was possible to control at the same time; this solution worked well for controlling the robot in the environment and will be the proposed control method if keyboard and mouse are to be used. It is possible to use VR controllers, PlayStation controllers, or similar to control the robot's behavior by developing a script.

The XR-lab requires the program to be run as a stand-alone program, not letting us easily alter the GameObject components, like the articulation body component of the last link of the Nachi that has to be enabled after the program is started. Here a possible solution can be to write a script that turns this component on after the program has been created so that the physics engine does not crash. Further, this also makes it more complicated to connect the model as some of the components are easier to get up and running if they can be altered during the simulation

7.3 Connecting the model

The proposed method for connecting the DT model to the physical system is to use the OSC protocol; this was because of troubles with building the program with the asset for the OPC UA connection. The OSC was also chosen because the XR-lab supports this communication protocol and utilizes it for controlling input, projectors, and other equipment inside the lab.

OSC as a communication protocol is not directly usable with robot programming. Therefore a bridge between this and the OPC UA server was proposed. The bridge will be built with python, but python does not have a library for receiving messages from OSC. It only can send. This makes the proposed communication method impossible to utilize now. It is suggested that pursuing another communication protocol would be better for further development as no one knows when a library for receiving OSC messages in python will be available.

8 Conclusion

The main objective of this thesis was to create a working model of two of the existing robots in the lab, Nachi MZ07 and an older SCARA robot in the newly implemented XR-lab at UiT Narvik. Both robots have been created in Unity with the necessary assets and toolkits to make them work in the XR-lab, and controls have been adjusted to match XR-lab and robot input methods simultaneously.

The changed keycodes make joint control possible, and the robot's articulation works well. An end effector control that is possible is proposed to make the control easier for the robots.

A possible way to connect the twins was made, and information about the joints' location is likely to get out from the XR environment. However, the lack of a library in python was making the bridge proposed impossible to make at this point.

8.1 Further work

As mentioned in section 4.4, the end effector movement is not fully developed. Getting the end effector movement to work would be beneficial as this will speed up controlling the robot and collaborating around it. With the proposed method, it can be possible to add VR controllers to the sphere making it possible to drag the end effector into the lab and watch it happen on the physical robot in real-time.

Finding a way to fully connect the models in the XR-lab to the physical systems. This project suggests using the OSC protocol. Still, as this is not fully integrated with Python, it will be beneficial to look at other methods for connecting the models to the physical systems. Continue study on the OPC UA will be a possible solution as this is a proven and steady bridge to the robots in the lab. It will be possible to thoroughly test it in the editor before integrating it into the XR-lab.

The robot models need to be further validated and verified to the physical robots; here, the primary focus should be on verifying the work envelope for both models.

Works cited

[1] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the Digital Twin: A systematic literature review," *CIRP Journal of Manufacturing Science and Technology,* vol. 29, pp. 36-52, 2020/05/01/ 2020, doi: https://doi.org/10.1016/j.cirpj.2020.02.002.

[2] M. Liu, S. Fang, H. Dong, and C. Xu, "Review of digital twin about concepts, technologies, and industrial applications," *Journal of Manufacturing Systems,* vol. 58, pp. 346-361, 2021/01/01/ 2021, doi: https://doi.org/10.1016/j.jmsy.2020.06.017.

[3] F. Palmas and G. Klinker, "Defining Extended Reality Training: A Long-Term Definition for All Industries," in *2020 IEEE 20th International Conference on Advanced Learning Technologies (ICALT)*, 6-9 July 2020 2020, pp. 322-324, doi: 10.1109/ICALT49669.2020.00103.

[4] S. H.-W. Chuah, "Why and who will adopt extended reality technology? Literature review, synthesis, and future research agenda," *Literature Review, Synthesis, and Future Research Agenda (December 13, 2018),* 2018.

[5] Igloo vision. "Igloo The University of Tromso: Immersive workspace." https://www.igloovision.com/case-studies/university-of-tromso (accessed 09.01.2022.

[6] S. M. E. Sepasgozar, M. Ghobadi, S. Shirowzhan, D. J. Edwards, and E. Delzendeh, "Metrics development and modelling the mixed reality and digital twin adoption in the context of Industry 4.0," (in English), *ENGINEERING CONSTRUCTION AND ARCHITECTURAL MANAGEMENT,* vol. 28, no. 5, pp. 1355-1376, JUN 10 2021, doi: 10.1108/ECAM-10-2020-0880.

[7] L. Gong, A. Fast-Berglund, and B. Johansson, "A Framework for Extended Reality System Development in Manufacturing," (in English), *IEEE Access,* Article vol. 9, pp. 24796-24813, 2021, doi: 10.1109/access.2021.3056752.

[8] C. Coupry, S. Noblecourt, P. Richard, D. Baudry, and D. Bigaud, "BIM-Based Digital Twin and XR Devices to Improve Maintenance Procedures in Smart Buildings: A Literature Review," *Applied Sciences,* vol. 11, no. 15, 2021, doi: 10.3390/app11156810.

[9] S. Alizadehsalehi and I. Yitmen, "Digital twin-based progress monitoring management model through reality capture to extended reality technologies (DRX)," (in English), *SMART AND SUSTAINABLE BUILT ENVIRONMENT*, doi: 10.1108/SASBE-01-2021-0016.

[10] E. Peckham, "How Unity built the world's most popular game engine ".

[11] Unity Technologies. "URDF-importer." https://github.com/Unity-Technologies/URDF-Importer (accessed 09.05.2022.

[12] codecademy. "Learn C#." https://www.codecademy.com/learn/learn-c-sharp (accessed 10.05.2022.

[13] w3schools. "C# Tutorial." https://www.w3schools.com/cs/index.php (accessed 10.05.2022.

[14] Visual Componenets. https://www.visualcomponents.com/ (accessed 09.01.2022.

[15] Game4Automation. https://game4automation.com/en/ (accessed 13.05.2022.

[16] NACHI-FUJIKOSHI CORP. "Corporate Info, History and Milestones." https://mdanderson.libanswers.com/faq/26219 (accessed 07.05.2022.

[17] Nachi Europe GmbH. "Company Profile." https://www.nachi.de/nachi-europe/company-profile.html (accessed 07.05.2022.

[18] NACHI robotic systems inc. "MZ07." https://www.nachirobotics.com/product/mz07/ (accessed 07.05.2022.

[19] DIY robotics. "Everything you need to know about SCARA robots." https://diy-robotics.com/article/scara-robots/ (accessed 13.05.2022.

[20] Researchgate. "SCARA robot manipulator." https://www.researchgate.net/figure/SCARA-robot-manipulator_fig1_341191036 (accessed 13.05.2022.
[21] OpenSoundControl.org. "What is OSC?" https://ccrma.stanford.edu/groups/osc/index.html (accessed 14.05.2022.
[22] Ventuz. "Open Sound Control (OSC)." https://www.ventuz.com/support/help/latest/OpenSoundControl.html (accessed 14.05.2022.
[23] C. unit. "Processign practice #01 | OSC." http://cu.t-ads.org/processing-practice-01-osc/ (accessed 14.05.2022.
[24] Unity Technologies. "URDF Tutorials Appendix." https://github.com/Unity-Technologies/Unity-Robotics-Hub/blob/main/tutorials/urdf_importer/urdf_appendix.md#file-hierarchy (accessed 11.05.2022.
[25] S. Brawner. "Solidworks to URDF Exporter." http://wiki.ros.org/sw_urdf_exporter (accessed 10.05.2022.
[26] Unity Forum. "The dreaded invalid AABB errors." https://forum.unity.com/threads/the-dreaded-invalid-aabb-errors.463144/ (accessed 11.05.2022.
[27] A. Zucconi, "An Introduction to Gradient Descent," vol. 2022, ed, 2017.
[28] Rocworks. "GraphQL for OPCUA." https://assetstore.unity.com/packages/tools/network/graphql-for-opcua-199115 (accessed 13.05.2022.
[29] V. Sigalkin. "extOSC - Open Sound Control." https://assetstore.unity.com/packages/tools/input-management/extosc-open-sound-control-72005#description (accessed 13.05.2022.