Implementation of Assembly Operations with YuMi Robot

Vladimir Kuliaev

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 27.5.2019

Supervisor

Prof. Valeriy Vyatkin

Advisor

Udayanto Dwi Atmojo, PhD

Aalto University
School of Electrical
Engineering

Aalto University
School of Electrical Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Vladimir Kuliaev	
Title Implementation of Assembly Operations with YuMi Robot	
Degree programme Automation and Electrical Engineering	
Major Control, Robotics and Autonomous Systems	**Code of major** ELEC3025
Supervisor Prof. Valeriy Vyatkin	
Advisor Udayanto Dwi Atmojo, PhD	
Date 27.5.2019	**Number of pages** 42+63 **Language** English

Abstract

Nowadays, the trend in manufacturing industries is to make the manufacturing processes sustainable in the countries with higher labour cost. Factory of the Future research project of Aalto University aims to develop a futuristic platform for research and demonstration of net-centric production manufacturing model. It is hoped, that the platform can contribute in transforming Finland's manufacturing into an agile, high tech network of providers of various manufacturing services of production activities. This thesis investigates, how successfully ABB IRB14000 YuMi robot can fit in this net-centric production concept. Two different approaches have been studied: Integrated Vision approach and From Digital Twins to Product Assembly Approach. The results of both studies show that to be able to fit the Factory of the Future concept, the second approach can be used with minor modifications. Two case studies demonstrate the agility of the method. Also, possible direction of improvement has been proposed. The first study results came to be very useful for possible higher flexibility of product-centric manufacturing.

Keywords Robotics, Integrated Vision, Digital Twin, Product-Centric Manufacturing, Industrial Robot

Preface

The research work presented in this thesis work has been done in a form of contribution to the Factory of the Future research project of Department of Electrical Engineering and Automation at Aalto University, Finland between May, 2018 and May, 2019.

I would like to express my deepest gratitude to Prof. Valeriy Vyatkin for entrusting me with this work and giving his valuable support and guidance. It was an honor for me to work with him. I would also like to thank my advisor Udayanto Dwi Atmojo for his continuous support, guidance and help throughout the work. At the same time, I would like to thank Dr. Seppo Sierla and Professor of Practice Jan Blech for their support throughout their support during the second part of the thesis and giving a wonderful working and coordinating experience that lead to the acceptance of the research in a form of a scientific paper for the 17th International Conference on Industrial Informatics in July, 2019 in Helsinki, Finland.

Finally, I would like to express my deepest gratitude and love to my family for their endless support, trust and encouragement throughout my life.

Espoo, 27.05.2019

Kuliaev Vladimir

Contents

Symbols and abbreviations

Abbreviations

3D	Three Dimensional
CPS	Cyber-Physical System
CPPS	Cyber-Physical Production System
QR code	Quick Response code
LAN	Local-Area Network
IP	Internet Protocol
TCP	Transmission Control Protocol
TCP	Tool Center Point
RGB	Red Green Blue
PC	Personal Computer
OEM	Original Equipment Manufacturer
CAD	Computer Aided Design
AM	Additive Manufacturing
OPC	Open Platform Communications
OPC UA	Unified Architecture
ASP	Assembly Sequence Planning
APP	Assembly Path Planning
ID	Identification number
BOM	Bill of Materials

1 Introduction

1.1 Background

The goal of the research described in this thesis, is to figure out how successful the assembly process, using YuMi robot might be from the factory of the future concept perspective.

Factory of the future is a research project going on at the Aalto University. As a part of Aalto Industrial Internet Campus, the Aalto Factory of the Future platform will be combined with several production-related labs to form a prototype of collaborative network. State of the art technologies will be used, from robotics, mobile machines, artificial intelligence, distributed computing, 3D printing, smart materials and wireless communication to realize the features of flexibility and adaptability, self-healing, autonomous operation, collaborative operations of product design and production facilities, and better resource utilization. The goal of the project is to create an environment, where the final project is assembled without any physical person taking part in the process. The product description is coming in a digital form and then needs to be assembled. Right now, the Facory of the Future consists of the storage island with conveyor belts that can lead the needed components to different locations, autonomous vehicle, that can deliver these components to some other islands, the collaborative robot, that can be installed on top of the moving platform and can perform different kinds of assemblies in different locations.

Previously, the trend in manufacturing industries was to outsource manufacturing process to some countries with low labour cost.[1] Nowadays, the trend of Industry 4.0 is targeting the goal of making manufacturing process sustainable in the countries with higher labour cost. Publications on CPS (Cyber- Physical Systems), CPPS (Cyber-Physical Production Systems), Smart Manufacturing and Industrial Internet, for example [2, 3, 4] among many others, demonstrate a trend within potential participants to offer their own specialised services for final product manufacturing. One of these offers is presented as a concept of using YuMi robot for assembly processes.

The goal of the following study is to investigate, how the IRB14000 YuMi Collaborative Robot can fit in the factory of the future environment. Two ways towards getting the final product will be described. Two different assembly methods will be examined with their benefits and disadvantages are examined.

1.2 Thesis Structure

Following is the breakdown of the structure of the thesis work:

- **Chapter 2**: In chapter 2, the first study is described. The use of Integrated Vision together with the YuMi robot is considered. One case study is presented. Results of the described use-case are given.

- **Chapter 3**: In this chapter, the product-centric approach is considered. The planning and modelling framework is introduced together with the robot control logic. The case studies are presented with the results. Possible way of combining both of the studies is proposed.

- **Chapter 4**: In the final chapter, the concluding remarks about the whole thesis work are added.

2 Integrated Vision Approach

2.1 Background

As a representative example of customized product assemble, Aalto Factory of the Future uses structures composed of Lego^{TM} blocks. There have been attempts to assemble Lego pieces using YuMi robot already. However, these attempts have used a pretty simple concept behind. The original location of Lego pieces was predefined. It was a tower that had a certain amount of bricks assembled together. The robot's job was to pick the bricks one by one from that tower and to assemble the row of the blocks in front of the base. That project mostly demonstrated only the ability of YuMi to disassemble Lego structures and to attach blocks in some new location.
There has been a study that considered applying integrated vision capabilities of the ABB robotics. The use case is to solve the five-piece Tangam puzzle. The YuMi robot was able to solve the task either alone or in collaboration with a human through messages on FlexPendant.[5] Puzzle pieces are spread randomly by hand of a human in the area reachable by YuMi. To identify and locate the puzzle pieces, the vision module, based on a Cognex In-Sight Ethernet smart camera was used. The vision job was based on the "PatMax Pattern" pattern recognition algorithm. As a result, the robot was able to solve the puzzle without human intervention.[5]
These two already completed projects contribute the following results:

1. First of all, it is known that high-precision robot is suitable for Lego pieces assembly.

2. The second study demonstrates, how the integrated vision algorithms can help with identifying the required workpieces.

The next section describes the concept of applying methods, described above, to get the assembly operation done.

2.2 Concept

The following assumptions are taken:

- The final product is known. It is required to assemble a tower, which consists of three layers.

- Each layer has a predefined colour. The bottom layer has to be yellow, the middle one has to be blue and the top layer has to be green.

- The concept is that the robot has to identify the location of Lego blocks on the board. After that the robot has to pick the blocks of appropriate colours and then to assemble the tower.

The 4x2 blocks of 5 different colours are spread on the board. The location and orientation of the blocks are random. Location of the blocks is identified with the top

COGNEX camera that attached to the robot. After that, the identified workpiece is picked up with one robot hand, passed to the second hand for colour identification. Unfortunately, the top camera is black and white so is unable to detect colours. Identifying colours in grey scale is not reliable. In this case, each of the Lego blocks has QR codes tag on the sides. The QR code only contains the word related to the colour of the workpiece (“red”, “blue”, “green”, “white”, “yellow”). The top camera does not also have auto focus, so the QR code reading is not possible with the camera in case it is installed for location identification. In this case QR reading job is made by using the camera on one of the hands of the robot.

1. Calibration. In the following scenario, this involves the calibration of the YuMI’s gripping fingers’ arm, which is necessary for the robot software control logic. This calibration is done to obtain the absolute position of the Yumi’s gripper when it is opening and closing.

2. Preparation for Item location identification. The arms of the robot are moved on the sides. This is done to ensure that the hands are not in range of the camera, no external shadows appear, and the future workpiece search will be as precise as possible.

3. Location identification. The top camera is taking the picture of the Lego board with blocks on it. After that the location is identified and the coordinated are stored in the format of predefined work object, that was created in relation to the top camera.

4. Preparation for identification of the workpiece. After the location is known, the workpiece needs to be identified. The right hand moves towards the workpiece and picks it up. Then the workpiece is moved to the predefined point somewhere in front of the robot base (in the middle). There the workpiece is passed to the left hand.

5. Colour identification. After the left hand received the workpiece, the right hand moves to a such position, where the camera on the gripper can capture the QR code on the side of the workpiece. At the same time, the left hand needs to rotate the workpiece as well. Then, the camera takes picture of the workpiece and reads the QR on it. After that, the workpiece is passed back to the right hand, which is in the position to place it to the target place.

6. Sorting. After the colour of the workpiece is known, the program can follow on of the two possible scenarios. If the colour of the block corresponds to the needed for current layer of the product, then the assembly process can be started. If the colour is of some other type, the workpiece is placed to the place, where it will be stored, until this colour will be needed for further assembly process. Steps 2-6 will be repeated until the required workpiece is found or as long as there are Lego blocks on the Lego board.

7. Assembly. When the robot knows that the needed workpiece in its hand, the Lego block is placed to the target position. As the workpiece is passed between

two hands during the previously mentioned steps, the accuracy of the gripping is not good enough for straight pushing it in. That is why aligning process is required.

8. Aligning. The aligning process is performed in such a way that the robot arm gently pushes the workpiece from all sides one by one to ensure the perfect aligning of the workpiece before pushing it in.

9. Pushing. When it is known for sure that the workpiece is aligned, it can be pushed directly downwards.

2.3 Set-up Description

To perform the assembly described in the previous part, the following components were used:

- Lego plate. As a workspace for the robot the classic 38x38 cm baseplate was used. This plate is required to ensure that all of the Lego blocks will be always kept in the correct places. This ensures precision during the assembly process, storing and any kind of manipulations within the robot job.

- IRB14000 YuMi Robot. For implementing the described above assembly process YuMi Robot was used. This robot is equipped with two lightweight and padded arms. The arms are controlled independently and are able to handle a payload up to 500 g per arm.[6] Each arm is equipped with two-finger smart grippers. Their configurations are described further.

- Lego blocks. Classic Lego blocks of 4x2 and 2x2 sizes were used. For the testing purposes only 5 of the blocks were required. All of the bricks are of different colours: green, white, yellow, blue and red.

- Camera. As in the study, mentioned in the paper [5], the vision module for this assembly implementation is also based on a Cognex In-Sight Ethernet smart camera (ism1402). The camera has 5 Megapixel resolution. The camera has manual focus and diaphragm and has to be calibrated to a certain position. In the current case, it was calibrated for the distance from the mounting point to the Lego baseplate. This distance is approximately equal to the size of an ISO A4 paper. This camera is monochrome. As a benefit, ism1402 is one of the highest speed performance model. Compared to the previous generation model, the performance rating has increased by the factor of 2.5. The greatest benefit of the chosen camera is the compatibility with the selected robot. Integrated vision module of ABB RobotStudio provides a plenty variety of possible programs that can be loaded into camera and they are called jobs. Each vision job consists of location and inspection tools. "Location tools provide position data of objects, like for instance, blobs, edges or patterns while inspection tools examine the located objects (measure distances, diameters or create geometric references, etc.)."[5] The results of jobs are output and further used in robot control logic without any conversions or complicated decoding.

- Smart grippers. The smart grippers of the “Servo + Vision + Vacuum” configuration have been used in the study. Vacuum option has not been used for the study due to the future idea that the robot will be mounted to the moving platform. Fingers required some minor modification. Rubber pieces were attached to each finger to increase friction between them and Lego bricks.
- Fixture mechanism for camera. Fixture Mechanism had to be designed from scratch to fulfil the needs of the current study.

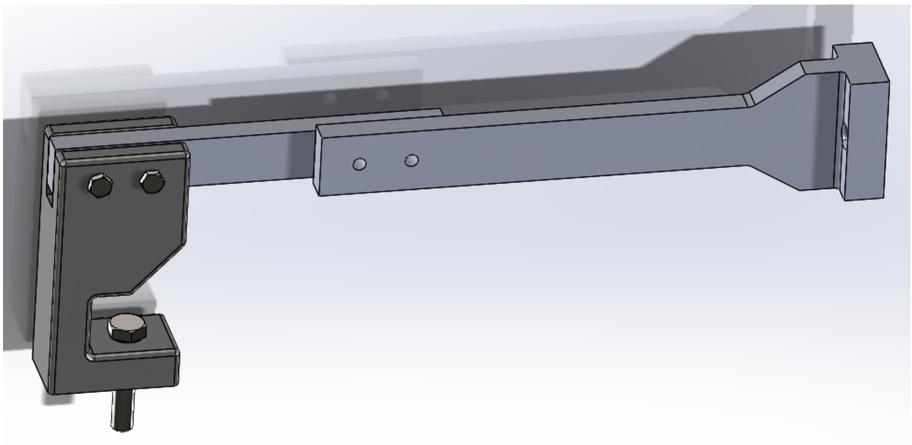


Figure 2.1: Camera fixture CAD model

The mechanism consists of the following components:

1. The base. It was decided to place the camera on top of the robot. Originally, IRM14000 has a service thread on top for lifting hook. This thread was used as a way to attach the base of the fixture. Such decision provided the possibility to have the camera always at the same spot even in case the robot is moved from the original place.
2. Extension arm. This component is required to extend the arm of the fixture. The length of this arm can be varied. In the current study the length of 10 cm was used.
3. Arm to camera fixture. This fixture is similar to the previously mentioned component. The difference is that this component has a fixture to the base or extension arm on the one side and the fixture to the camera on another one.

Figure 2.2 shows how the fixture looks like with the camera attached on top of YuMi robot. All of the components have been 3D printed out of ABS plastic and all of the CAD drawings can be found in the appendices.

2.4 Camera Set Up

To be able to set up the camera, separate fixture mechanism(described before) was designed. The mechansism consists of 3 components(figure, add a bit later/screenshot

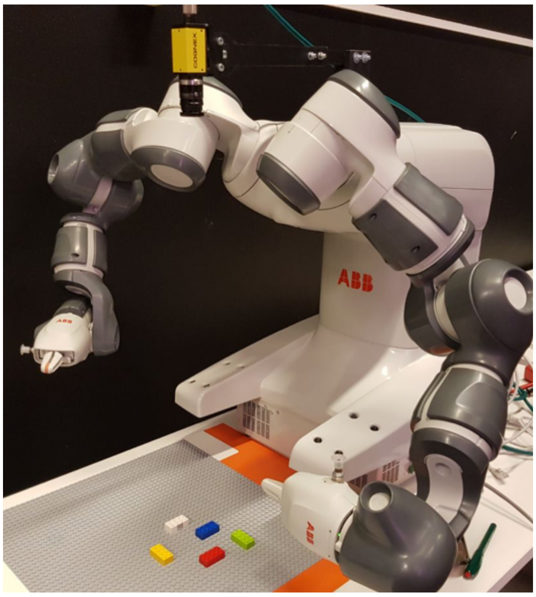


Figure 2.2: Camera fixture on top of IRB14000

needed) that need to be connected together. The ognex camera needs to be attached to the last plate. Cable connection needs to be done so that the 8 pin cable is connected to the camera on one side and to OUT port of the power injector. Then, the Ethernet cable needs to be connected to IN port of the power injector on one end and to the IRB14000 LAN 2 port on another end. Only this connection guarantees stable connection and control of the camera jobs. In case of multiple external cameras usage, the switch has to be connected to LAN 2 port. The manual provided by the robot distributor is unclear about the connection set up. It can be noticed at the figure 2.3 [6] below that there are 3 LAN ports available. However, LAN 1 port is not visible outside IRB14000. LAN 3 port is better to be isolated from private network in some cases where Ethernet/IP or PROFINET buses are used. That is why it is strongly recommended to connect the camera to LAN 2 port.

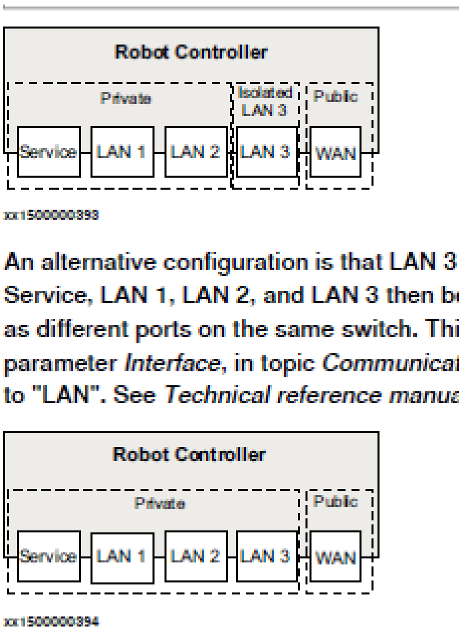
To be able to detect location of Lego blocks, the following preparations need to be done: Camera set-up. As mentioned earlier, the provided camera is monochrome with manual settings. In this case the robot operator needs to adjust the camera to the Lego baseplate.

Firstly, it is important that all of the Lego studs are clearly seen on the image. It is possible to make some test pictures via RobotStudio.

Secondly, the diaphragm value has to be chosen such that Lego bricks can be clearly differentiated from the baseplate. In greyscale some colours, such as green under certain light might look pretty similar to plain grey colour.

Thirdly, the camera to baseplate calibration is required. This process has to be done manually by using RobotStudio software.

The image consists of pixels. It is required to get the result in mm. So, for this purpose camera calibration needs to be done. The Calibrate function is available in RobotStudio. The whole process consists of two steps. First of all, it is required to



Robot Controller
Private
Isolated LAN 3
Public
Service
LAN 1
LAN 2
LAN 3
WAN
xx1500000393

An alternative configuration is that LAN 3
Service, LAN 1, LAN 2, and LAN 3 then b
as different ports on the same switch. Thi
parameter *Interface*, in topic *Communicat*
to "LAN". See *Technical reference manua*

Robot Controller
Private
Public
Service
LAN 1
LAN 2
LAN 3
WAN
xx1500000394

Figure 2.3: LAN ports configuration

convert the image pixels to mm. After that, is required to relate camera coordinated to a robot frame (Work object).[7]
The table 2.1 below describes the camera calibration process.[7]

Table 2.1: Camera calibration

	Action
1	Make sure that the camera is in **Program Mode**.
2	Click **Calibrate** in the ribbon.
3	In the **Context** window, change the **Calibration Type** to **Grid**.
4	From the **Grid Type** drop-down menu select one of the checkerboard calibration plates with fiducial, reference point.
5	If necessary, adjust the spacing, units, lens model, and number of poses settings. Use mm as the unit. The lens model depends on from where the most distortion is expected. Either because the camera is viewing from an angle (projection), or that the lens itself is distorting the image (radial). Number of poses allows to use more than one image of the calibration plate to calibrate the camera in case the plate does not cover the full field of view.
6	Click **Print Grid** to print the calibration plate. The printed image must have a high contrast and the paper must not be reflective (high gloss). Verify with a ruler that the squares are proportional.
7	Place the calibration plate on a fixed position in the center of the camera image, at the same height as the objects that the camera shall identify. The calibration paper must be completely flat, adequately illuminated, and free from gloss and shadows. Rotate the calibration plate so that the X and Y arrows corresponds to the desired direction of the camera work object.
8	In the **Context** window, click **Next**. The calibration is now being calculated by the camera, and the number of found feature points are displayed.
9	Click **Next**.
10	Click **Calibrate** to apply the calibration.
11	Click **Finish** to complete the calibration.
12	Do not change the position of the calibration plate until the work object has been defined.

2.5 Camera to Robot Calibration

The camera is calibrated to the robot by defining a work object with the same origin of coordinates as the calibration plate.
The table 2.2 below describes the camera to robot calibration process.[7]

Table 2.2: Camera to robot calibration

	Action
1	Create a pointing tool and define the tool TCP using an accurate method.
2	Create one work object for each camera.
3	Activate the pointing tool and define the user frame of the camera work object along the corresponding x- and y-axes of the calibration plate. Leave the object frame empty.
4	Test that the calibration is correct by jogging the robot in the work object.
5	The calibration plate can now be removed.

After the calibration is done, and new work object has been created, it is required to teach the robot, how to grab the brick. It is not enough for YuMi robot to get XYZ coordinated of the target object for picking it up. As the robot has 7 joints, it is required to determine at least some average configuration of the joints operating within the coordinate values that are not predeclared in the code.
Gripping a part is often not the same as moving the TCP to the target reported by the camera. Often this position must first be offset and rotated by some value to accommodate a good grip.
The easiest way to do this is by jogging the robot to the specified position and then modify the position, usually referred as ModPos.
The table 2.3 below describes the process, how to teach the robot how to grip a workpiece.[7]

Table 2.3: Gripper teaching process

	Action
1	Run the program till the point where the hand stops above the workpiece and stop the execution. At this point the object frame of mywobj has been modified and the correct tool, mytool, is activated.
2	Jog the robot to a good gripping position.
3	Mark the position myrobtarget and tap **ModPos**.
4	Run the program from the top and make sure that the part is gripped according to the taught position.
5	Move the part and run the program from the top again.

2.6 Hands Synchronization

Practically, IRB14000 is a model that contains 2 separate one-arm robots connected to one base. To synchronize hands between each other, flags have to be put. While the program is under execution, flags can guarantee that one of the hands will not arrive to certain position until the second one is ready for it to happen. Synchronization variable "syncident" is in charge of this waiting process [8]. During the program 10 of them have been used.

1. To increase a chance of successful part location detection there has to be nothing in range of the top camera when the picture of the baseplate with bricks on it is taken. The first syncident sync1 is ensuring that both hands are outside the range.

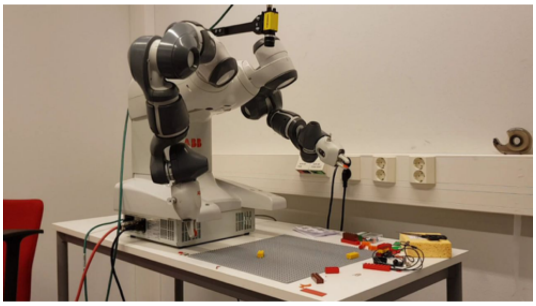


Figure 2.4: Syncident 1 caption

2. Second flag sync2 guarantees that the left arm is in suitable position for picking up the already detected block from the right hand. Also, the gripper opens and gets ready for picking the workpiece.

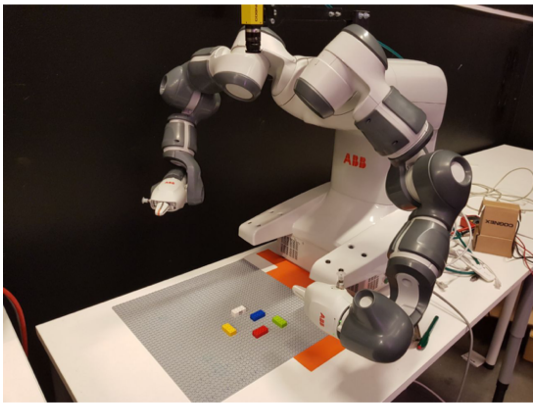


Figure 2.5: Syncident 2 caption

3. Third flag ensures that the left arm moves towards the workpiece that is still held by the right arm.

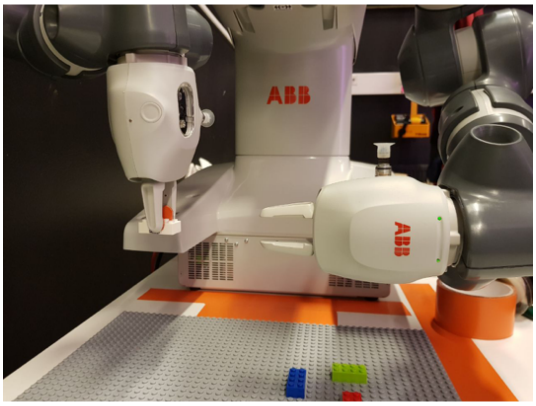


Figure 2.6: Syncident 3 caption

4. The fourth syncident guarantees that the left arm holds the workpiece while the right one is already not holding it.

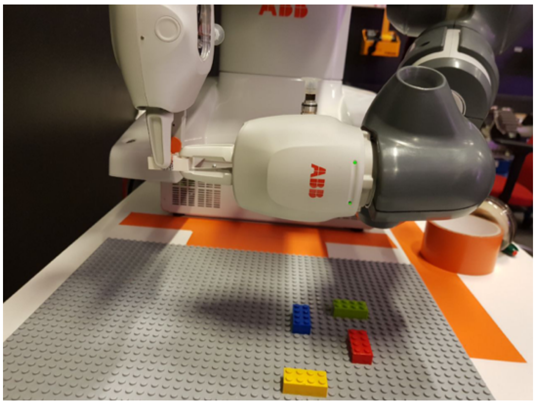


Figure 2.7: Syncident 4 caption

5. The fifth synchronization point takes place when the right hand moves away letting the left one to rotate the gripper joint for QR code reading.
6. Sync6 ensures that both of the hands are in suitable positions for QR reading.
7. Sync7 returns both of the arms back to positions where the already identified workpiece can be passed back to the arm, that is responsible for the assembly process (Right arm).
8. Sync8 is ensures the correct passing of the workpiece. One hand opens the gripper fingers and the second hand closes its fingers.
9. Sync10 gives the command to the left hand to move aside to avoid collisions with the right arm, that will start the assembly process.

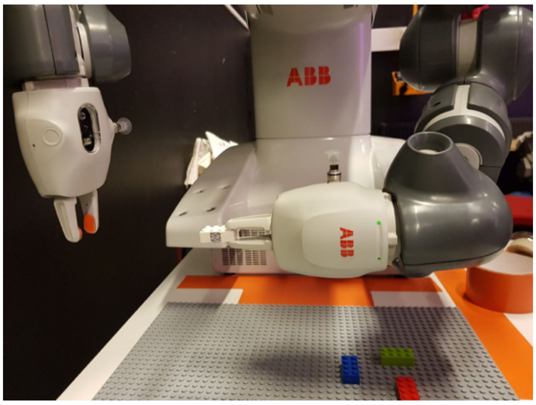


Figure 2.8: Syncident 5 caption

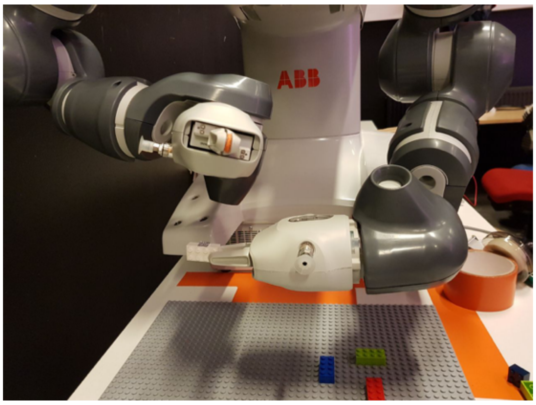


Figure 2.9: Syncident 6 caption

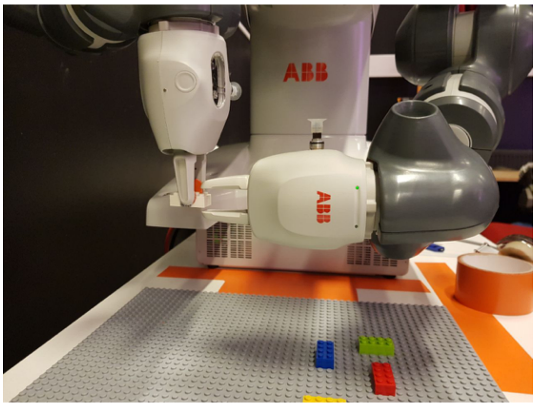


Figure 2.10: Syncident 8 caption

10. Sync9 provides that information that the left arm has reached the destination point and will no longer participate in the assembly process till the new workpiece has to be identified.

2.7 Integrated Vision Jobs

2.7.1 Location Detection

First job that has been used is “lct.job”, responsible for the workpiece location identification. "PatMax Pattern" pattern algorithm was chosen to be used for workpiece search. Typical result of the search can be seen in the figure 2.11.

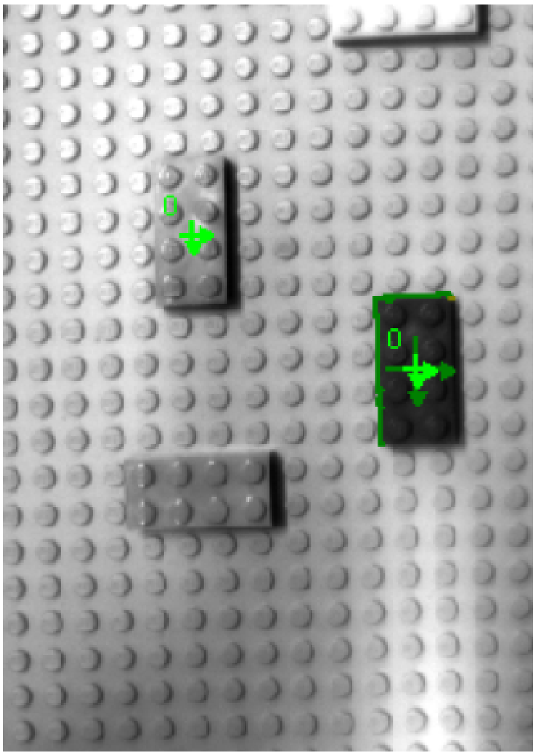
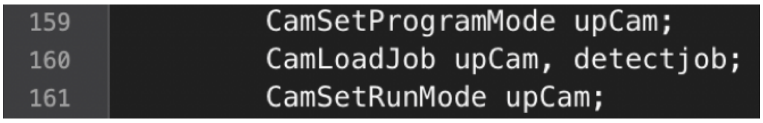


Figure 2.11: PatMax result example

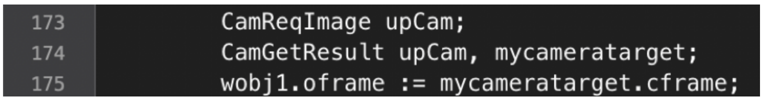
After the workpiece is found, the location coordinates are stored in mycameratarget variable. It is important to first switch the camera into program mode, then load the job to the camera and then put it into the run mode.



```
        CamSetProgramMode upCam;
        CamLoadJob upCam, detectjob;
        CamSetRunMode upCam;
```

Figure 2.12: Up Camera Preparation

Only after that, the camera is ready for taking the picture and running the search job.
As it was described before, separate workobject has been created the environment and coordinates, calculated by the camera.



```
        CamReqImage upCam;
        CamGetResult upCam, mycameratarget;
        wobj1.oframe := mycameratarget.cframe;
```

Figure 2.13: Up Camera Result capturing

2.7.2 QR Code Reading

2-D verification algorithm has been used for the QR code reading. It has been figured out that the size of the QR code has to be at least 7x7 mm for successful reading. The Smart Gripper camera was used for 2-D verification job. The typical result is presented on the figure 2.14 below.

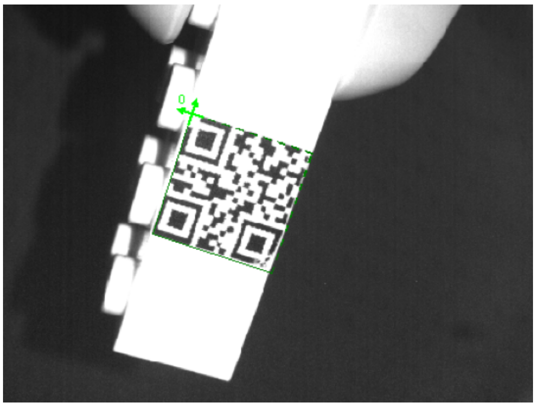


Figure 2.14: QR Reading Result Example

The results are stored in the variable cameratarget1. This variable is able to keep multiple Boolean values. Val1-Val5 have been used for 5 colours. Val1 corresponds to white, val2-yellow, val3-red, val4-blue, and val5-green. Depending on this value, according to the control logic the robot either starts the assembly or puts the workpiece aside for storing and looks for the next workpiece.

2.8 Results

The results of the study are considered as controversial. On one hand, the designed control logic works, and the robot is able to assemble the desired product. On the other hand, the possibility of failure is pretty high. Ten test runs of the same program have been made and only during 3 of them, the robot was able to finish the assembly. The item location process is very sensitive to the external lighting. If some shadow appears, the contrast is changing, and the camera is not able to detect workpieces on the Lego baseplate. Probably, the use of RGB camera can solve the issue.
From the Factory of the Future perspective, this concept has one serious constraint. Camera calibration and Camera to Robot calibration is not possible to be realized in without manual contribution of the robot operator. This goes against with the concept of the Factory of the Future and makes it impossible to have this robot running the program being installed on the moving platform.

3 From Digital Twins to Product Assembly

Second study describes the possibility of using the robot as a slave, receiving commands from Digital Twin on a remote PC.
It presents a possibility of a new development that enables the integrated approach for product-centric manufacturing. It proposes an overall solution that will allow product-centric manufacturing in the factory floor and will cover more global lifecycle from digital product description to physical assembly.
This study describes the implemented connection to a physical assembly of ABB IRB 1400 YuMi collaborative robot. The results on using the framework to assemble Lego-brick models in a product centric way are also presented.

3.1 Background

The fourth industrial revolution, Industry 4.0, is expected to bring about the dissolution of the well-established automation architecture [9, 10, 11, 12]. This architecture is based on the emerging Reference Architecture Model Industry 4.0 (RAMI 4.0). It promises real time availability of data and information lifecycle phases and organizational boundaries [13]. Nowadays, sophisticated technology in RAMI 4.0 is the OPC UA [14, 15]. In the context of RAMI 4.0, the focus so far has been on enabling technology, but the real revolution is going to occur when the technology is successfully implemented for servitization of industry with advanced agile manufacturing paradigms executed by networked enterprises [11]. The need for agile manufacturing arises not only from rapidly changing market demand but also from novel product design approaches exploring larger parts of the design space [16] and the need to perform concurrent product design and assembly planning [17].
One example of an agile manufacturing paradigm is the product-centric manufacturing. The idea behind this type of manufacturing is that the digital equivalent of a product will request manufacturing services [18]. The core benefit of product-centric manufacturing is that the manufacturing facility does not require any offline assumptions concerning the types of products that will be manufactured, the way of manufacturing each of the products, and the order of executions. This all means, that later in the future there will be more agility in a manufacturing process, better capability to react on some customized products.
The recent work has extended the product-centric manufacturing paradigm to the factory floor domain [19]. It demonstrates the concept of self-made virtualized 3D environment. Later, this work has been extended to utilize OPC UA to communicate the digital description of the product from the designer to possible manufacturers [20]. This provided the possibility of having designer and the manufacturer in different organizations. Till now, the concept in [20] has been implemented only in virtual world. It was decided to continue this approach and to extend the lifecycle from virtual assembly to some physical one. The study below presents a development that enables an integrated approach for product-centric manufacturing.

3.2 Concept

Product-centric manufacturing concept has three stages:

1. Firstly, the original equipment manufacturer (OEM) designer sends a digital product description to one or more manufacturers, who will automatically and promptly perform a virtual assembly in a virtualized 3D production cell without any need for manual and physical engineering work. This part has been already done in some previous work[19]. The work has already been done on the basis lego-bricks using collision detection. Also, there has been a study in the same source for generic CAD part designs without collision detection[19] but this example will not be considered in the current study. The concept implies that based on the results of the virtual assembly, the designer can decide whether the design is practical and economical from the assembly perspective and if the manufacturing site has suitable capabilities for the manufacturing this design. The feedback from the virtual assembly can be used for modification of the original design and to decrease the amount of potential manufacturers.

2. During the second stage the assembly is piloted with physical product parts and production equipment (“Pilot assembly”). As an example, this assembly can involve Additive Manufacturing (AM). This approach is can lead to reduction of delays involved at this stage [21] especially due to overcoming the delays from the supply chains [22]. The point about the current method is that it may not be yet be a cost-effective technology for mass production [23, 24].

3. The third stage (“Ramp up”) involves ramping up the production with the chosen manufacturer, after the design has been adjusted based on the results of the first two stages. During this stage, the parts can be produced with AM, a hybrid scenario which combines AM and conventional supply chain management [25], or even completely without AM.

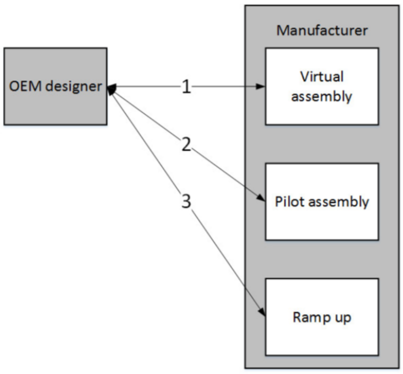


Figure 3.1: Stages of product-centric manufacturing concept

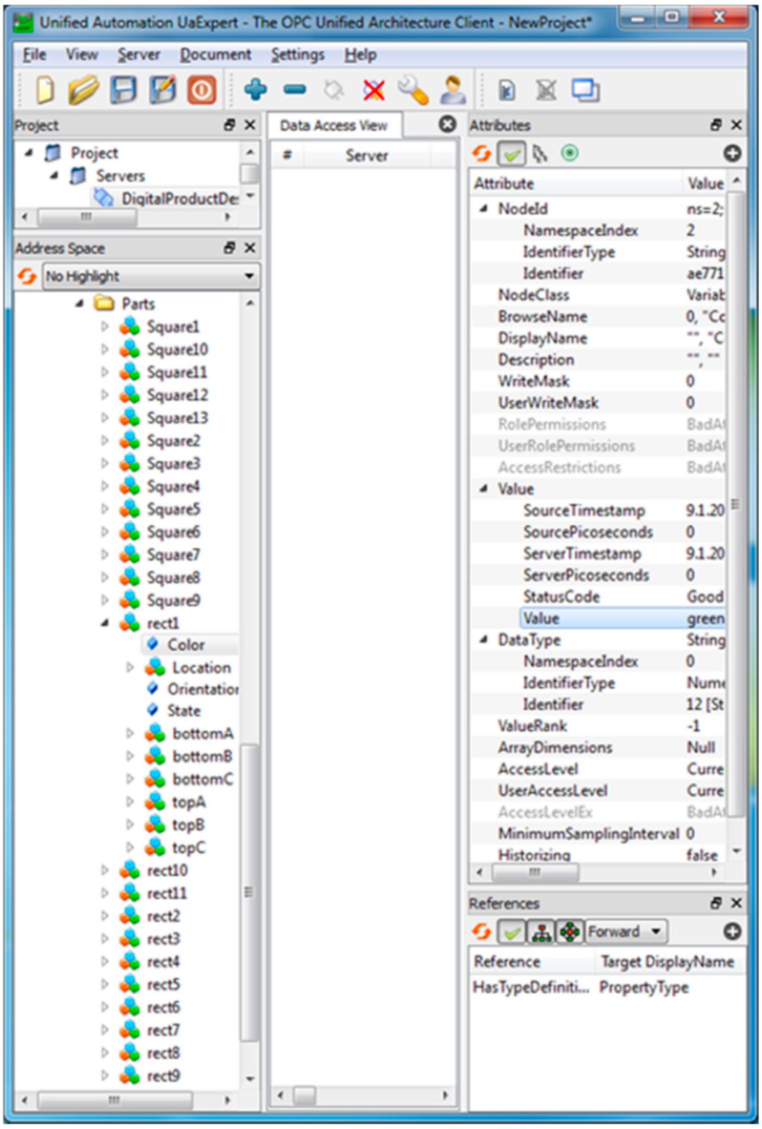
To conclude, the proposed concept may result in changes of the manufacturing sector. It opens the possibilities for new and innovative designers with small budgets to participate in the competition without investing in dedicated production facilities. The first stage has been already implemented in the some previous research [19]. The core challenge was to try to realize the second and the third stage by using the physical robot (IRB 14000 YuMi) with versatile capabilities to perform physical assembly.

3.3 Assembly Planning and Modelling Framework

Originally, the assembly planning and modelling framework was proposed in [19]. In the current chapter, the whole process concept will be briefly described. For the case of the current study the existing framework has been simplified.

As it has been mentioned in the previous paragraph, the automatic assembly of products is based on their digital descriptions. In [20], the products were built out of workpieces in forms of Lego blocks of 2 configurations: 2x2 and 2x4. OPC UA address spaces have are originally used for the product descriptions. The simple description of a typical "Lego tower" in form of OPC UA server files is shown on Figure 3.2. Both types of Lego blocks are presented as OPC UA types with corresponding connection points. All of the Lego bricks are located under a Parts folder. The screenshot from UaExpert demonstrates the example of one design with several 2x2 and 2x4 blocks. In the Address space pane, the "Colour" attribute of the 2x2 block named "rect1" is selected. Also, in the Attributes pane, it is seen that the value of this attribute is green. This means, that according to the digital description, the colour of the selected block is green.

The methodology presented in [19] is able to automatically construct a digital twin of the final product in an assembled state, using the 3D properties of the square and rectangle Lego bricks together with the corresponding connections between them. The whole process happens based on the product description. The digital twin is augmented with a product-centric control capability, including ASP and APP. The methodology in [19] performed the product-centric control in the 3D virtual environment with a cartesian robot. It is important to mention that in this environment, interconnected nodes are manipulated, and the limitations, constraints and challenges of the physical world are ignored. During the current study, the product-centric control is adapted to the physical world and the ABB YuMi robot. The figure 3.3 presents the product-centric control for the stages 2 qnd 3 of the figure 3.1. The given algorithm is executed cyclically. Each execution will exercise only a part of the code according to the guard conditions in the opt, break and alt fragments. The function nextUnassembledPart() towards the bottom of the sequence is responsible for ASP. planAssemblyPath() is responsible for APP, returning a Trajectory object consisting of waypoints and rotations of the part to be assembled. The variety of Boolean variables are in charge of controlling the translational movement to the next waypoint or rotation. After the translation or rotation is completed, the sequence proceeds to the nextPoint() method of the Trajectory object. In case this returns null, the sequence jumps to the place() operation and proceed to the next call of ASP. During the current study, the sequence is performed on the PC machine that hosts a digital twin and communicates with a physical assembly station (ABB



Project
Servers
DigitalProductDe

# | Server

Parts
Square1
Square10
Square11
Square12
Square13
Square2
Square3
Square4
Square5
Square6
Square7
Square8
Square9
rect1
Color
Location
Orientation
State
bottomA
bottomB
bottomC
topA
topB
topC
rect10
rect11
rect2
rect3
rect4
rect5
rect6
rect7
rect8
rect9

Attribute	Value
NodeId	ns=2;
NamespaceIndex	2
IdentifierType	String
Identifier	ae771
NodeClass	Variab
BrowseName	0, "Cc
DisplayName	"", "C
Description	"", ""
WriteMask	0
UserWriteMask	0
RolePermissions	BadA
UserRolePermissions	BadA
AccessRestrictions	BadA
Value	
SourceTimestamp	9.1.20
SourcePicoseconds	0
ServerTimestamp	9.1.20
ServerPicoseconds	0
StatusCode	Good
Value	green
DataType	String
NamespaceIndex	0
IdentifierType	Nume
Identifier	12 [St
ValueRank	-1
ArrayDimensions	Null
AccessLevel	Curre
UserAccessLevel	Curre
AccessLevelEx	BadA
MinimumSamplingInterval	0
Historizing	false

Reference	Target DisplayName
HasTypeDefiniti...	PropertyType

Figure 3.2: Product description of a "Lego tower" as OPC UA information model

YuMi robot). The communication is set up through socket in order to realize the product-centric manufacturing in the physical world.

3.4 From Virtual to Physical: From Digital Twin to Assembly

In the previous section, it has been described that the digital twin is in charge of controlling the assembly of its physical counterpart. In the current setup, the twin is running on a PC and communicates with the physical manipulator on the factory floor. As in the first study, the physical manipulator's role is performed by ABB IRB14000 Dual Arm Precision “YuMi” robot. Based on the digital product description, the digital twin generates a particular assembly sequence and planning

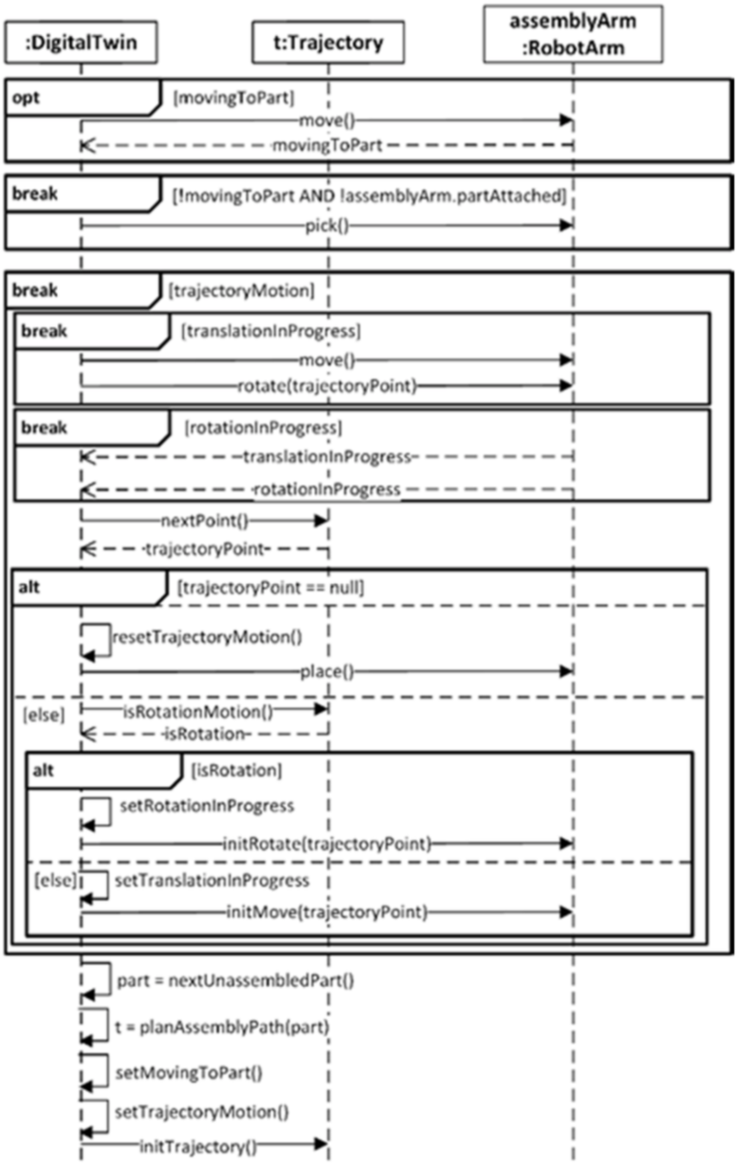


Figure 3.3: Product-centric Assembly Requests

routine which is later transferred to the YuMi robot in a specific form of messages. The YuMi robot is operated by a RAPID language-based software control logic. This logic allows the robot to receive the routine sent by external computer with a digital twin running there, decode, read and transform into physical execution and manipulation to assemble the physical counterpart of the digital twin.

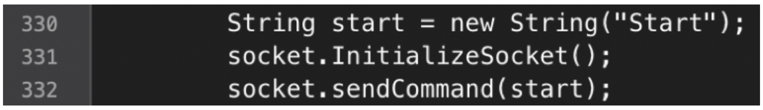
3.5 Planning and Modelling from the Framework Side

The framework described in the previous section is implemented in Java and is running on a PC. Based on the given digital product description, the framework cyclically generates a list of several information regarding each component, that together forms a final product. The required data has been taken from the available info:

- Name of the workpiece with its identification number (ID)
- Three axis target coordinates of where the workpiece has to be placed
- Information regarding the need of rotating the workpiece before placing it to the target location

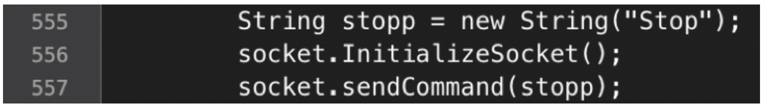
This data has to be transferred to the physical assembly station (YuMi robot). It was mentioned earlier, that the existing framework has been simplified and modified to make the setup work. The following changes have been made to the existing framework:

- First of all, the communication interface has been introduced. As it has been mentioned earlier, the generated data about every workpiece has to be somehow transferred from the PC to the physical workstation controller on the factory floor. That's why the new communication interface has been introduced. This communication channel has been realized in a form of TCP/IP communication socket. It exists both in Java (planning and modelling framework) and in RAPID programming environment (ABB YuMi robot).
- Messaging protocol between the framework and physical station controllers. The messaging protocol includes a text command to start the assembly process, cyclic stream of information regarding every Lego component and a message to finish the assembly process and return to the original position of the robot.



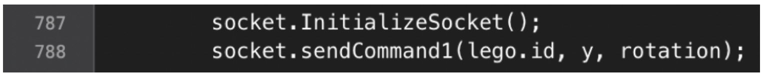
```
String start = new String("Start");
socket.InitializeSocket();
socket.sendCommand(start);
```

Figure 3.4: Command to start in DigitalTwin



```
String stopp = new String("Stop");
socket.InitializeSocket();
socket.sendCommand(stopp);
```

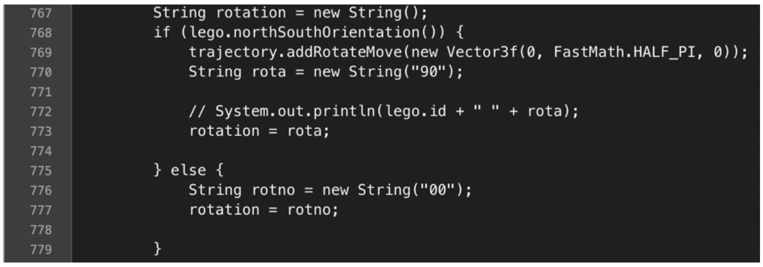
Figure 3.5: Command to stop in DigitalTwin



```
socket.InitializeSocket();
socket.sendCommand1(lego.id, y, rotation);
```

Figure 3.6: Command to send a string of relevant information in DigitalTwin

- Actuator coordination functionality in the DigitalTwin class has been modified to make the message more compact.



```
        String rotation = new String();
        if (lego.northSouthOrientation()) {
            trajectory.addRotateMove(new Vector3f(0, FastMath.HALF_PI, 0));
            String rota = new String("90");

            // System.out.println(lego.id + " " + rota);
            rotation = rota;

        } else {
            String rotno = new String("00");
            rotation = rotno;

        }
```

Figure 3.7: Rotation information in DigitalTwin

- All the OPC UA functionality has been disabled to simplify the case studies. Later on, it can be returned but some further modifications will be required. During the current study, the digital product description is generated by the client in form of the .txt file.

- Collision detection check has been also disabled. This has been done due to the fact that at the current point of the study collision in the virtual world has nothing the same with possible collisions during physical assembly. Besides that, the existing framework is created so, that during collision checking process, virtual assembly is still performed. This means, that the physical station will try to repeat even the process, when the final product will be failed to get assembled.

3.6 YuMi Robot Controller Side

The robot-controller software is implemented using ABB's RAPID programming language. It is important to point out that during the realization of this project only one hand of the robot has been used. All of the presented solutions correspond only to the right arm of Yumi. In particular, the following functionalities have been realized:

1. Calibration. During the current study, Smart Gripper for the YuMi robot has been used. Only the gripping fingers are required. Vacuum suction cups are not used and can be not attached. Before the assembly starts, calibration of these fingers is required. As the fingers have extra rubber material on the sides to increase friction while gripping, the calibration is done to obtain the absolute position of the fingers by opening the fingers and closing the fully with fixating both of the positions.

2. Initialization. This block of commands is responsible for setting up the communication channel between the Digital Twin and the YuMi controller. The network communication socket is opened and waiting for the message containing the command to start the assembly process. After the message is received, the robot control logic activates the right arm and moves it to the "safe" position. "Safe" position has been intentionally introduced by the code developer. This is such a position, that ensures safe movements towards the workpiece storage and

towards final destinations of workpieces. The code is designed so, that every time the arm needs to move between the storage and the target locations it is passes through the “safe” position. This position is required to avoid possible collisions of the arm with the table, on which the assembly is performed. Also, it is important to mention that in-built collision avoidance functionality is disabled during this study.

3. The following block describes the control logic of the assembly process. The predefined settings are made in a way that the final project can consist out of the 10 2x4 bricks and one 2x2 block. All of the Lego pieces are available at their specific locations before the assembly starts. The main program is presented in a form of a recurring loop, which consists of the following:

   - Moving the joints of YuMi’s arms to the “safe” position if they are not in this position yet.
   - Communication to transmit and receive message. This is the communication that is performed via socket. Unfortunately, RAPID programming language supports less data formats compared to Java. However, it is enough that both of the language support strings. Bute format was also an option, but the process of decoding the information is much more complicated compared to strings. String format was chosen since it can be used to transmit information of different “nature” from pure numbers to words.
   - Breaking down the message into the parts that make sense for the RAPID code. This includes type of the Lego brick, it’s ID number, the rotation requirement, and the three axis coordinate values of the target location of the workpiece at the final assembly location.
   - Actuation of the robot to move towards the appropriate workpiece, actuate gripping fingers to grab the Lego block with a certain holding force to be able to tear it off the Lego plate, move the arm with the selected workpiece towards the target coordinates passing through the “safe” point to avoid collision with the environment, and then rotate the workpiece 90 degrees in case the rotation is required.
   - Pushing the workpiece on the assembly position. In the current study, pushing is required as the assembly consists only out of Lego components. The design of Lego blocks has been made in a way that the secure connection of the bricks can be achieved only by pushing. As the YuMi robot requires gentle manipulations, pushing process is achieved by performing the pushing action twice in the middle of the rectangular block, then twice to the left of the center point and finally twice to the right of the center point. In case of square workpiece only double pushing in the center is enough. This pushing cycle ensures proper attachment of Lego pieces.
   - After the pushing cycle, robot arm (physical actuator) is returned to the “safe” position.

4. Finishing block. If the just pushed Lego block was the last one from the assembly, the robot receives "Stop" command. The control logic always checks in the beginning of the cycle if the received message contains "Stop" word in the beginning. According to the code description, the arm of the robot has to return the "home" position. Having both arms in this position, the robot is ready for transportation to some other workplace if needed.

Above described algorithm is shown in the Figure 3.8 below.

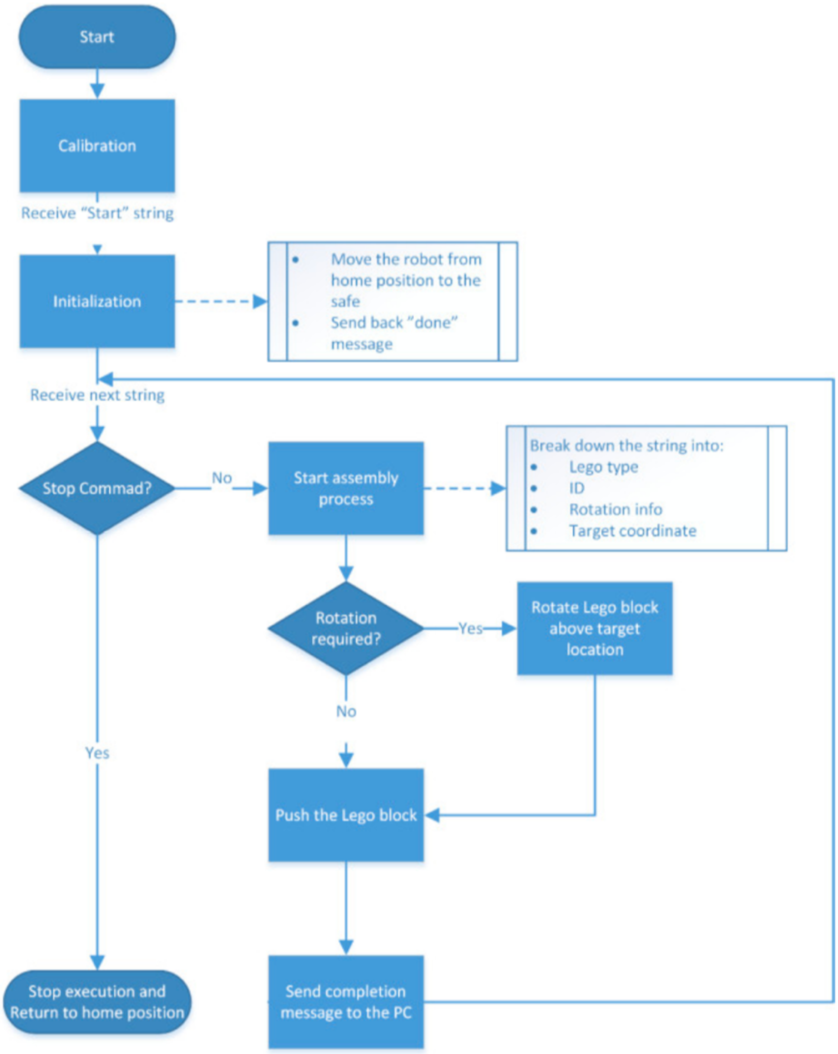


Figure 3.8: QR Reading Result Example

3.7 Case Studies

To demonstrate and evaluate flexibility and versatility of the code, two different case studies of assembly planning and execution have been considered. As OPC UA part has been disabled in the Digital Twin framework, digital product description was declared in a text format. All the relevant information is presented in form of two tables. The first table shows the typical information that can be found in a Bill of Materials (BOM) together with the angle of orientation information in the final assembly. The second table represents connections between Lego bricks. Rectangular workpieces have three connection possibilities. Every connection point is located in the center of the 4x2 studs of the Lego and each has interleaving 2x1 studs with the adjacent connection point. The three connection points on the top surface are named as topA, topB, topC. At the same time, bottom connection points are called bottomA, bottomB and bottomC correspondingly. Both of the tables help to figure out target coordinates on the workpieces in the final assembly. This forms the basis for ASP and APP of the modelling and planning framework. For the purpose of description, the case studies will be later on referred as “Square Tower” and “Pyramid Tower”.

3.7.1 Case Study 1 "Square Tower"

The Square Tower consists of 10 rectangular Lego bricks. The bricks have to be assembled in such way, that the tower has five layers. Each layer has 2 bricks assembled next to each other parallel. The orientation of each floor differs 90 degrees compared to the previous layer. Tables 3.1 and 3.2 demonstrate the description of case study 1.

Table 3.1: Square Tower Part List

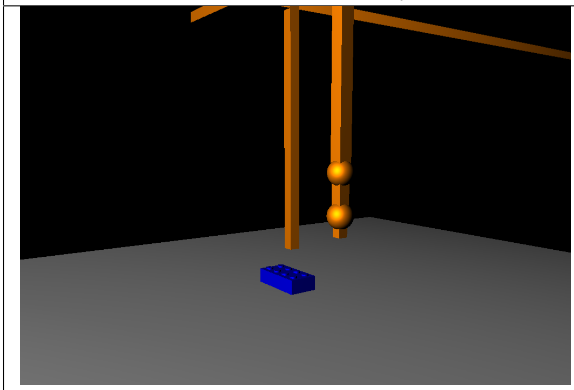
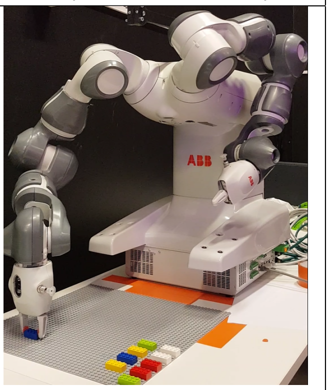
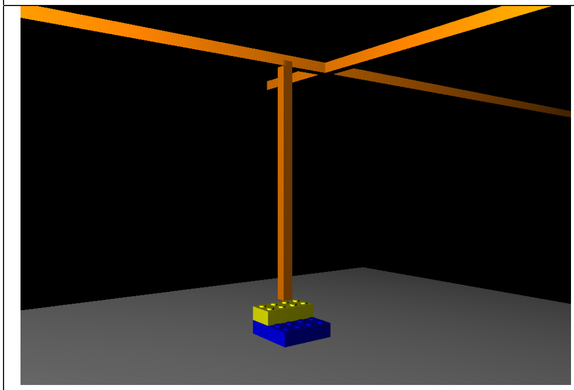
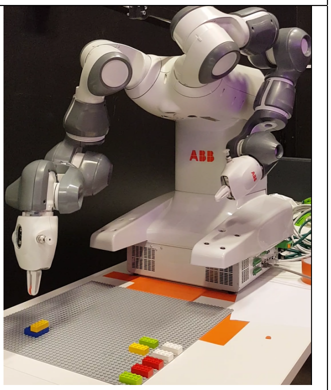
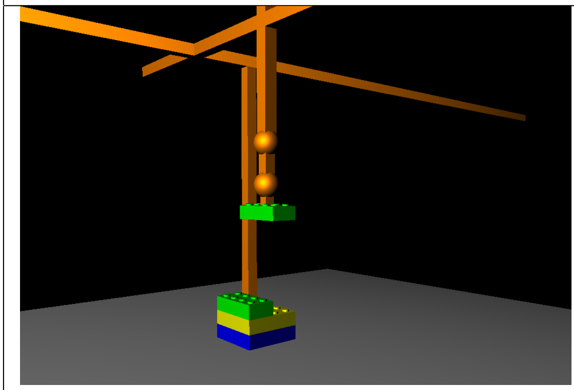
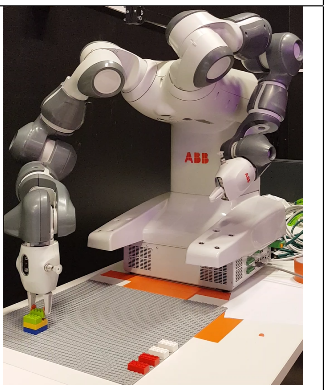
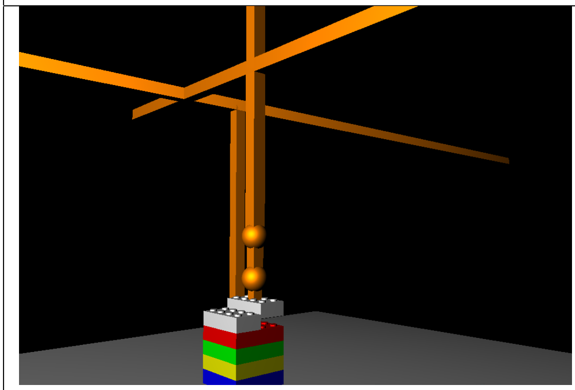
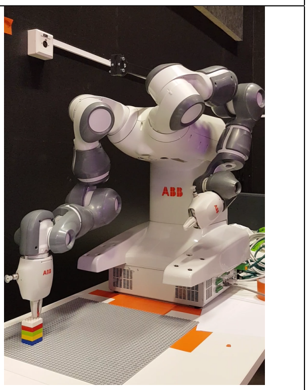
Type	**Part ID**	**Colour**	**Orientation**
RectangleLego	Rectan1	Blue	90
RectangleLego	Rectan2	Yellow	0
RectangleLego	Rectan3	Blue	90
RectangleLego	Rectan4	Yellow	0
RectangleLego	Rectan5	Green	90
RectangleLego	Rectan6	Green	90
RectangleLego	Rectan7	Red	0
RectangleLego	Rectan8	Red	0
RectangleLego	Rectan9	White	90
RectangleLego	Rectan10	White	90

As mentioned earlier, planning and modelling framework cyclically generates information, that is sent to YuMi robot for physical assembly. The following table demonstrates how the assembly process looks like in the virtual world and in physical environment. All of the snapshots have been made simultaneously during the assembly procedure.

Table 3.2: Square Tower Connection List

Connection endpoint 1		Connection endpoint 2	
Part ID	**Connection point**	**Part ID**	**Connection point**
Rectan2	bottomA	Rectan1	topA
Rectan2	bottomC	Rectan3	topA
Rectan1	topC	Rectan4	bottomA
Rectan5	bottomA	Rectan2	topA
Rectan6	bottomA	Rectan2	topC
Rectan7	bottomA	Rectan5	topA
Rectan8	bottomA	Rectan5	topC
Rectan9	bottomA	Rectan7	topA
Rectan10	bottomA	Rectan7	topC

Table 3.3: Square Tower Connection List

Virtual Assembly	Physical assembly
	
	
	
	

3.7.2 Case Study 2 "Pyramid Tower"

The second case study looks pretty similar to the first one. The first difference is that now the square Lego block has been added on the top. The Pyramid Tower has also 5 layers. Two bottom layers consist of three rectangular bricks, two of each are oriented parallel and the third one is rotated 90 degrees. The third layer consists of two 4x2 blocks, the fourth-1. The top layer is presented as a square piece. The tower has a pointed tip. The tower has only one line of symmetry, while the previous case study has two. The product descriptions of the Pyramid tower are presented in Table 3.4 and Table 3.5.

Table 3.4: Pyramid Tower Part List

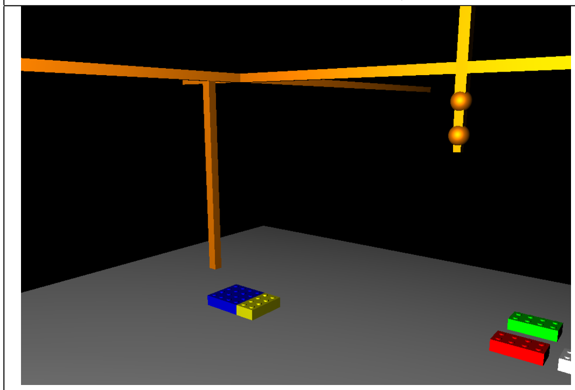
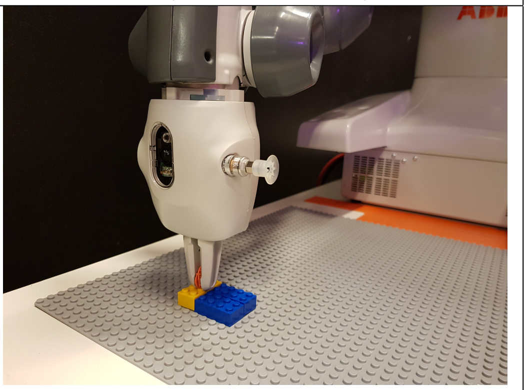
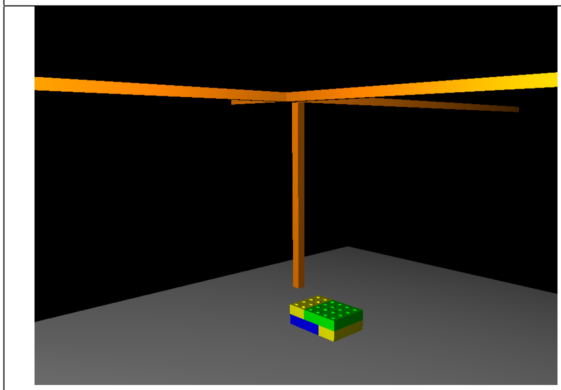
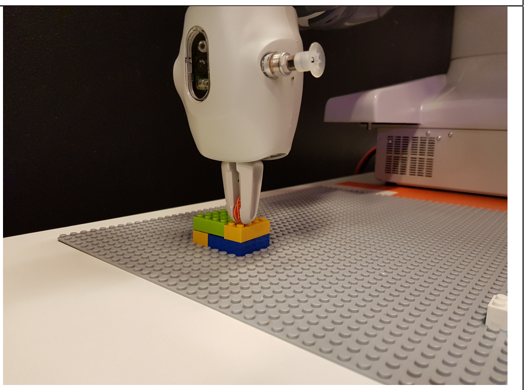
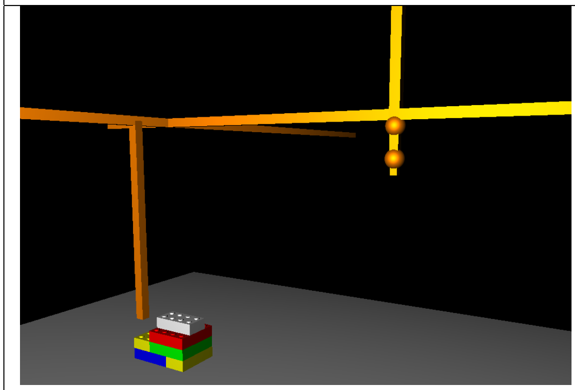
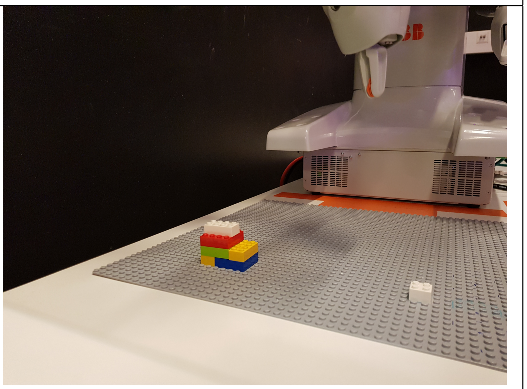
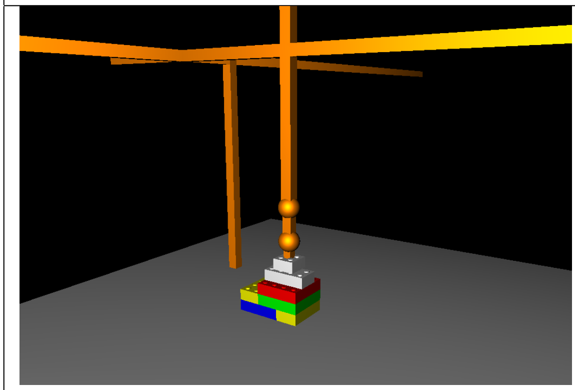
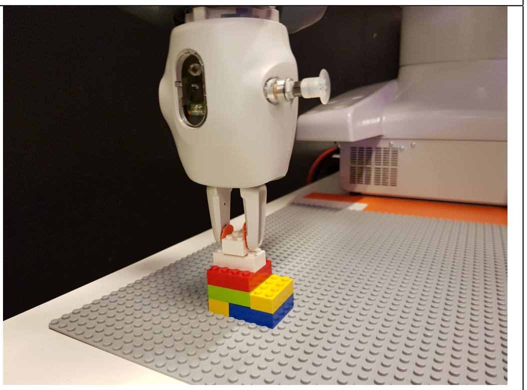
Type	**Part ID**	**Colour**	**Orientation**
RectangleLego	Rectan1	Blue	90
RectangleLego	Rectan2	Blue	90
RectangleLego	Rectan3	Green	90
RectangleLego	Rectan4	Yellow	0
RectangleLego	Rectan5	Green	90
RectangleLego	Rectan6	Yellow	180
RectangleLego	Rectan7	Red	0
RectangleLego	Rectan8	Red	0
RectangleLego	Rectan9	White	90
SquareLego	Square1	White	-

Table 3.5: Pyramid Tower Connection List

Connection endpoint 1		Connection endpoint 2	
Part ID	**Connection point**	**Part ID**	**Connection point**
Rectan2	bottomA	Rectan1	topC
Rectan2	topC	Rectan3	bottom
Rectan1	topC	Rectan4	bottomA
Rectan5	topC	Rectan4	bottomA
Rectan6	bottomA	Rectan5	topA
Rectan7	bottomA	Rectan2	topA
Rectan8	bottomA	Rectan2	topC
Rectan9	bottomA	Rectan7	topB
Square1	bottom	Rectan9	topB

As mentioned earlier, planning and modelling framework cyclically generates information, that is sent to YuMi robot for physical assembly. The following table demonstrates how the assembly process looks like in the virtual world and in physical environment. All of the snapshots have been made simultaneously during the assembly procedure.

Table 3.6: Square Tower Connection List

Virtual Assembly	Physical assembly
	
	
	
	

3.8 Results

Both of the case studies clearly demonstrate that the created code works well. The robot was able to assemble designed product assemblies that differ from each other. The key point that allows to consider achieved results as successful is that the source code of the robot has not been modified before changing the final product description. However, the drawbacks faced during the tests still seem to be quite similar to ones, faced during the first study. As both of studies are considered to be used in a Factory of the Future concept, the problem of manual defining of the workobject keeps to remain critical. It has been noticed during the test runs of the robot, that if the robot has been moved even one millimetre to any of the sides, the possibility of successful assembly process is leading to zero. From one hand, the goal of the study has been reached. From another hand, keeping in mind the whole concept of having the robot on top of the moving platform leads to the conclusion that some serious interventions to the source code of the IRB14000 is required. One of the possibilities might be the research on the way of declaring the workobject based on the image received from the external camera on top of the robot. At the moment of the study, this kind of functionality is not available from the robot producer. As a result,

3.9 Possible Enhancements

Taking into account two studies that have been done throughout current thesis, some possible future enhancements are proposed:

1. First of all, it definitely makes sense to combine methods investigated in both studies. For example, the robot instead of using predefined workpieces searches for the one that fits the description of the one that the Digital Twin requests for the assembly and then places it to the target location. This would make the assembly process even more flexible than it is in the second study. To make the search faster, it is strongly recommended to substitute the QR reading thread. The optimal way is to start using the RGB camera, to differentiate colours at the very beginning of the searching algorithm.

2. Secondly, to make the mobility of the robot possible, it is required to make the process of camera to robot calibration and camera calibration independent from the robot operator. If it is possible to make it happen in a form of code, the robot can be mounted to the moving platform and it will be able to assemble different designs at different locations.

3. Move to wireless communication between the Digital Twin and the robot framework. During the studies described in this thesis, all of communication between PC and the robot controller have been made using the wired connection. Wireless communication will increase the flexibility and the area of usage as well.

4. Bringing back the OPC UA protocol. While considering current study as a part of Factory of the Future concept, the robot has to be able to communicate with

other stations involved in the assembly process. Applying OPC UA standard would better fit the nowadays trends in automation.

4 Conclusions

In this thesis work, two different approaches towards final product assembly have been considered. The first study was focused on the way how to figure out which workpiece to choose for the assembly process. The core idea behind was that the final product always remains the same, while the range of Lego bricks suitable for the assembly is bigger than the actual number of pieces, the final product consists of. The designed control logic allowed the robot to identify randomly distributed Lego blocks, pick them one by one, identify the colour and then either to proceed to assembly, or to put the workpiece aside and keep in the memory where the already identified Lego block is stored and of which colour it is.

The second study was considering the opposite proposal. The final assembly description was presented in a way of digital description, while the list of available items for assembly always keeps being the same at predefined locations. This required to have some separately running program on external computer that configures the required assembly path, that is later sent to the YuMi robot. The versatile code has been designed, which brings flexibility to the whole system.

The results of the studies have been evaluated from the side of having the desired assembly done and from the point of having the code running in the robot on top of the moving platform. The objectives of the work are achieved with fair success, however there were some limitations and restrictions faced during the case studies and test runs of the codes. The core issue that both of the approaches require some calibrating process that cannot be done via coding at a time of the research. This leads to the limitation of robot usage together with Lego baseplate. At the same time, throughout the study, future direction of possible studies has been proposed.

References

[1] G. Zülch, H. I. Koruca, and M. Börkircher, "Simulation-supported change process for product customization–a case study in a garment company," *Computers in Industry*, vol. 62, no. 6, pp. 568–577, 2011.

[2] E. Hofmann and M. Rüsch, "Industry 4.0 and the current status as well as future prospects on logistics," *Computers in Industry*, vol. 89, pp. 23–34, 2017.

[3] A. Giret, E. Garcia, and V. Botti, "An engineering framework for service-oriented intelligent manufacturing systems," *Computers in Industry*, vol. 81, pp. 116–127, 2016.

[4] A. Fayoumi, "Ecosystem-inspired enterprise modelling framework for collaborative and networked manufacturing systems," *Computers in Industry*, vol. 80, pp. 54–68, 2016.

[5] D. Kirschner, R. Velik, S. Yahyanejad, M. Brandstötter, and M. Hofbaur, "Yumi, come and play with me! a collaborative robot for piecing together a tangram puzzle," in *International Conference on Interactive Collaborative Robotics*, pp. 243–251, Springer, 2016.

[6] A. Robotics, "Product manual – irb 14000-0.5/0.5," *Västerås, Sweden*, 2015.

[7] A. Robotics, "Operating manual robotstudio," *Västerås, Sweden*, 2007.

[8] A. Robotics, "Technical reference manual rapid overview," *ABB AB Robotics Products SE-721 68 Västerås Sweden.*

[9] V. Krueger, A. Chazoule, M. Crosby, A. Lasnier, M. R. Pedersen, F. Rovida, L. Nalpantidis, R. Petrick, C. Toscano, and G. Veiga, "A vertical and cyber–physical integration of cognitive robots in manufacturing," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1114–1127, 2016.

[10] R. Harrison, D. Vera, and B. Ahmad, "Engineering methods and tools for cyber–physical automation systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 973–985, 2016.

[11] H. Kagermann, R. Anderl, J. Gausemeier, G. Schuh, and W. Wahlster, *Industrie 4.0 in a Global Context: strategies for cooperating with international partners.* Herbert Utz Verlag, 2016.

[12] A. J. Isaksson, I. Harjunkoski, and G. Sand, "The impact of digitalization on the future of control and operations," *Computers & Chemical Engineering*, vol. 114, pp. 122–129, 2018.

[13] F. Zezulka, P. Marcon, I. Vesely, and O. Sajdl, "Industry 4.0–an introduction in the phenomenon," *IFAC-PapersOnLine*, vol. 49, no. 25, pp. 8–12, 2016.

[14] V. Jirkovskỳ, M. Obitko, P. Kadera, and V. Mařík, “Toward plug&play cyber-physical system components,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, pp. 2803–2811, 2018.

[15] S. Grüner, J. Pfrommer, and F. Palm, “Restful industrial communication with opc ua,” *IEEE Transactions on Industrial Informatics*, vol. 12, no. 5, pp. 1832–1841, 2016.

[16] D. Bacciotti, Y. Borgianni, and F. Rotini, “An original design approach for stimulating the ideation of new product features,” *Computers in industry*, vol. 75, pp. 80–100, 2016.

[17] E. Gruhier, F. Demoly, and S. Gomes, “A spatiotemporal information management framework for product design and assembly process planning reconciliation,” *Computers in Industry*, vol. 90, pp. 17–41, 2017.

[18] J. Lyly-Yrjänäinen, J. Holmström, M. I. Johansson, and P. Suomala, “Effects of combining product-centric control and direct digital manufacturing: The case of preparing customized hose assembly kits,” *Computers in Industry*, vol. 82, pp. 82–94, 2016.

[19] S. Sierla, V. Kyrki, P. Aarnio, and V. Vyatkin, “Automatic assembly planning based on digital product descriptions,” *Computers in Industry*, vol. 97, pp. 34–46, 2018.

[20] M. Yli-Ojanperä, S. Sierla, N. Papakonstantinou, and V. Vyatkin, “Adapting an agile manufacturing concept to the reference architecture model industry 4.0: A survey and case study,” *Journal of Industrial Information Integration*, 2018.

[21] M. Attaran, “The rise of 3-d printing: The advantages of additive manufacturing over traditional manufacturing,” *Business Horizons*, vol. 60, no. 5, pp. 677–688, 2017.

[22] S. H. Khajavi, J. Partanen, and J. Holmström, “Additive manufacturing in the spare parts supply chain,” *Computers in industry*, vol. 65, no. 1, pp. 50–63, 2014.

[23] Q. Li, I. Kucukkoc, and D. Z. Zhang, “Production planning in additive manufacturing and 3d printing,” *Computers & Operations Research*, vol. 83, pp. 157–172, 2017.

[24] R. F. Hartl and P. M. Kort, “Possible market entry of a firm with an additive manufacturing technology,” *International Journal of Production Economics*, vol. 194, pp. 190–199, 2017.

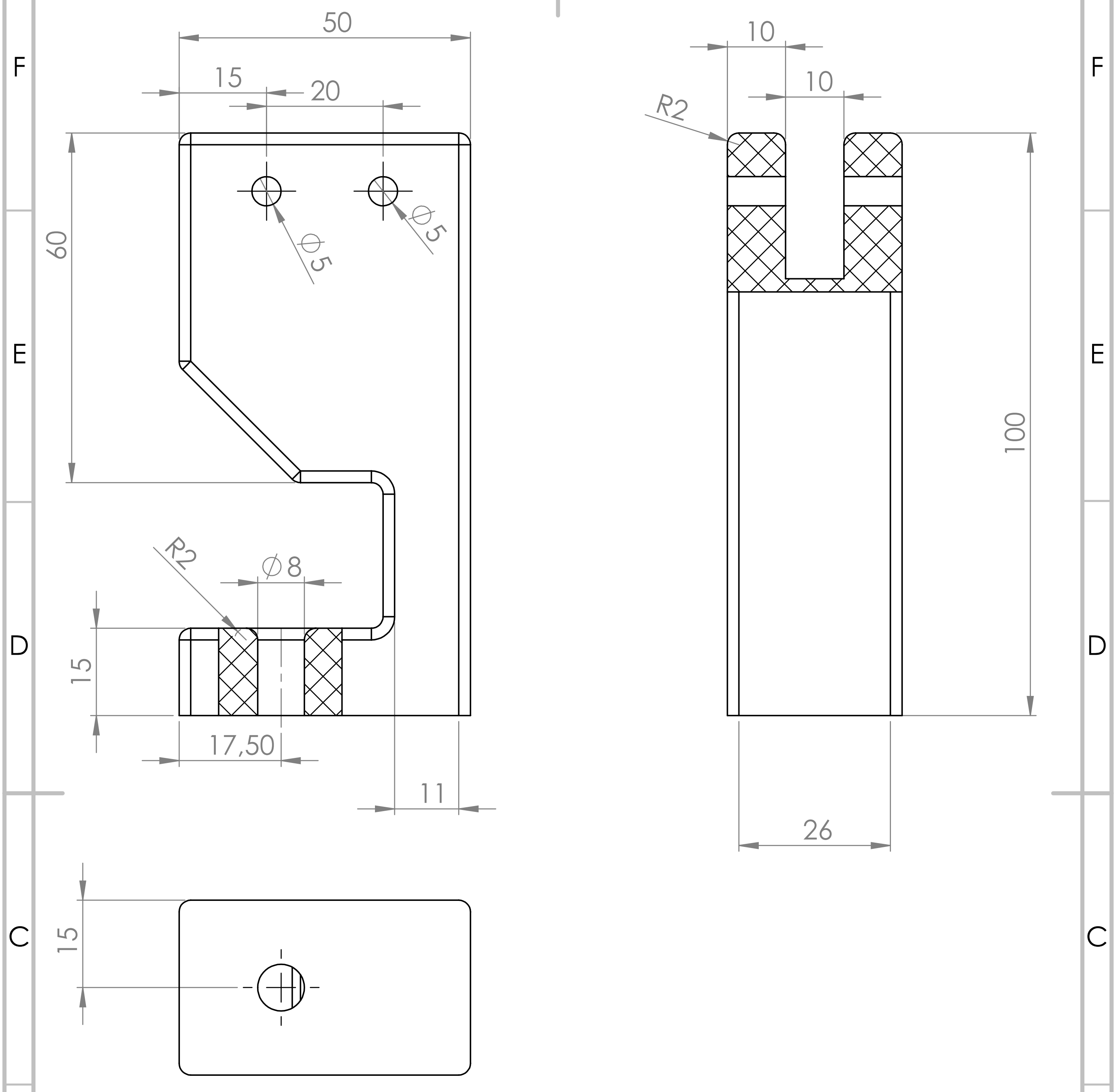
[25] S. H. Khajavi, J. Partanen, J. Holmström, and J. Tuomi, “Risk reduction in new product launch: A hybrid approach combining direct digital and tool-based manufacturing,” *Computers in Industry*, vol. 74, pp. 29–42, 2015.

[26] V. Kuliaev, "Yumilego2019." https://github.com/vladimirkuliaev/YuMiLego2019, 2019.

A Appendix

A.1 CAD Drawings

This Appendix is intended to document the CAD drawings of the Fixture mechanism described in 2.3. The models have been created in SolidWorks software. All the files and drawing in .SLDPRT and .SLDDRW formates can be found in Git repository.[26]



UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:

FINISH:

DEBURR AND BREAK SHARP EDGES

DO NOT SCALE DRAWING

REVISION

	NAME	SIGNATURE	DATE
DRAWN	Vladimir Kuliaev		
CHK'D			
APPV'D			
MFG			
Q.A			

TITLE:

MATERIAL:

DWG NO.

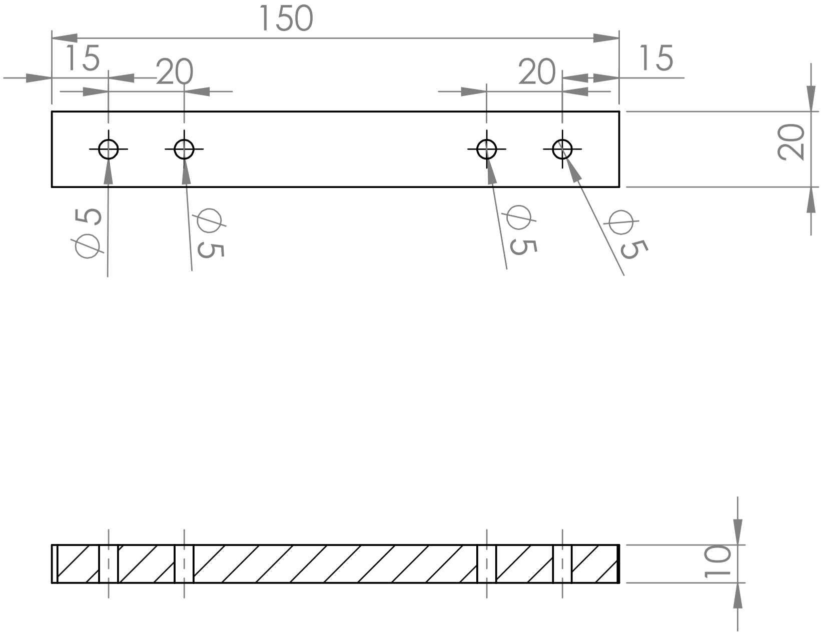
part1

A4

WEIGHT:

SCALE:1:1

SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:

FINISH:

DEBURR AND
BREAK SHARP
EDGES

DO NOT SCALE DRAWING

REVISION

	NAME	SIGNATURE	DATE
DRAWN	Vladimir Kuliaev		
CHK'D			
APPV'D			
MFG			
Q.A			

TITLE:

MATERIAL:

DWG NO.

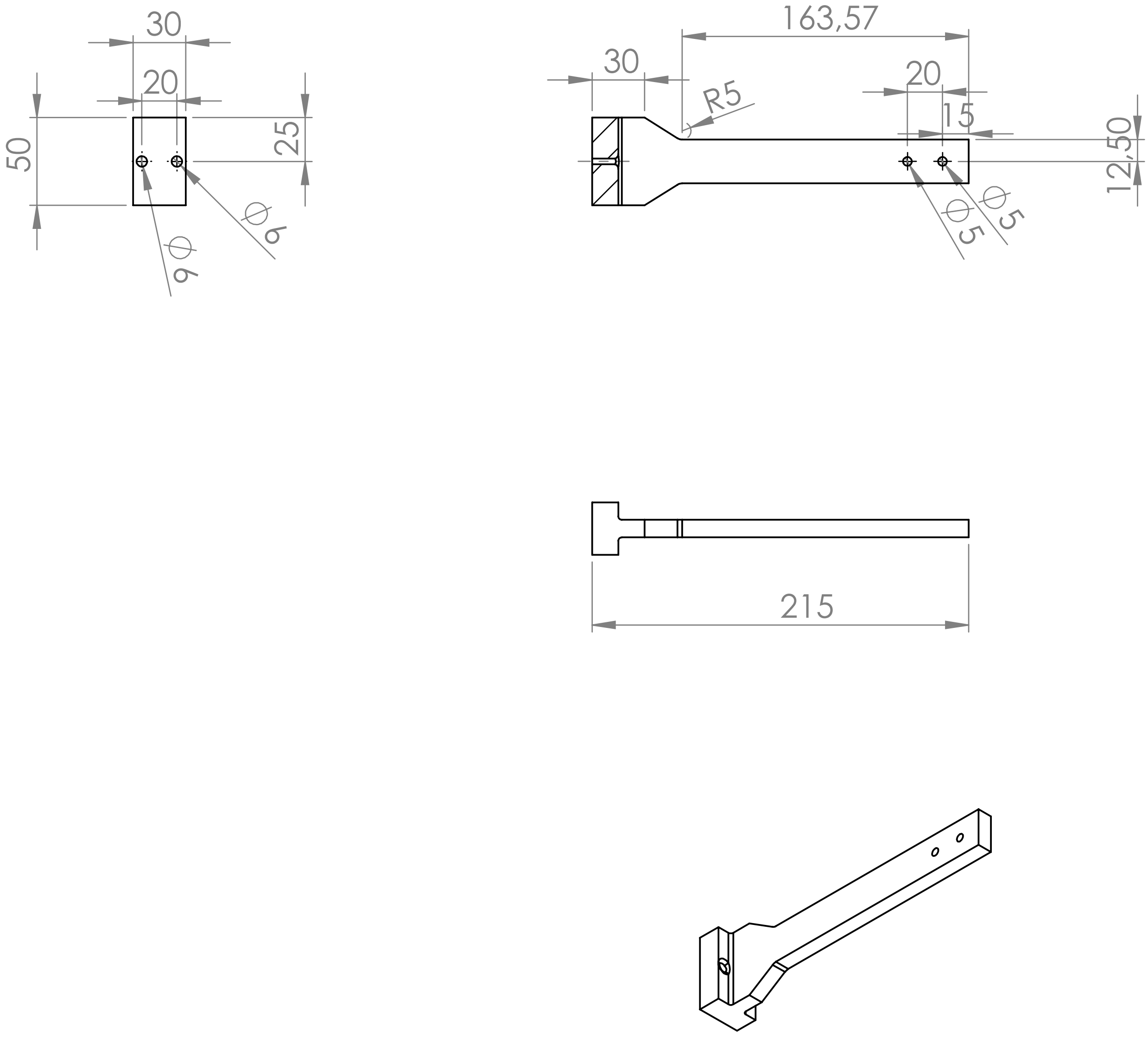
part2

A4

WEIGHT:

SCALE:1:2

SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:

FINISH:

DEBURR AND
BREAK SHARP
EDGES

DO NOT SCALE DRAWING

REVISION

	NAME	SIGNATURE	DATE
DRAWN	Vladimir Kuliaev		
CHK'D			
APPV'D			
MFG			
Q.A			

TITLE:

MATERIAL:

DWG NO.

Part3

A4

WEIGHT:

SCALE:1:4

SHEET 1 OF 1

A.2 Programs for the left and for the right hands

This Appendix is intended to document the programs for both of the YuMi arms, described in 2. The programs have been created in ABB RobotStudio software. All the files in .MOD formate can be found in Git repository.[26]

```
MODULE MainModule
        CONST robtarget safePoint:=[[194.64,287.75,197.42],
[0.0735244,0.843246,-0.113842,0.520163],[0,1,-1,4],
[102.588,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget safePoint10:=[[266.62,149.09,62.15],
[0.495985,0.508823,-0.500331,0.49474],[-1,1,-1,4],
[107.346,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget readyToOpen:=[[266.62,149.08,62.15],
[0.495987,0.508822,-0.50033,0.494739],[-1,1,-1,4],
[107.346,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget readyToOpen10:=[[254.87,69.02,59.07],
[0.496019,0.508794,-0.500334,0.494732],[-1,1,-1,4],
[108.541,9E+09,9E+09,9E+09,9E+09,9E+09]];

    VAR syncident sync1;
        VAR syncident sync2;
    VAR syncident sync3;
    VAR syncident sync4;
    VAR syncident sync5;
    VAR syncident sync6;
    VAR syncident sync7;
    VAR syncident sync8;
    VAR syncident sync9;
    VAR syncident sync10;
    VAR bool ready :=FALSE;

    PERS tasks tasklist{2}:=[["T_ROB_L"], ["T_ROB_R"]];
        CONST jointtarget
photoPos:=[[-83.0377,-142.241,15.6685,81.9131,72.8104,-152.786],
[70.0473,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget readyToOpen20:=[[254.87,244.32,59.07],
[0.496027,0.508793,-0.500331,0.494728],[-1,1,-1,4],
[108.541,9E+09,9E+09,9E+09,9E+09,9E+09]];
        PROC main()
        preparation;
        Stop;
        Hand_SetHoldForce(12);
        WHILE NOT ready
        DO
            check_operation;

        ENDWHILE






        ENDPROC
    PROC preparation()
        g_JogOut;
                g_SetForce 5;
                g_SetMaxSpd 3;
                g_JogIn;
```

```
                 g_Calibrate;
                 g_GripIn;
    ENDPROC

    PROC check_operation()
        MoveL safePoint, v1000, fine, tool0;

                WaitSyncTask\InPos, sync1, tasklist;

                 MoveL readyToOpen, v1000, fine, tool0;

                 g_MoveTo 10;

                WaitSyncTask\InPos, sync2, tasklist;

                 MoveL readyToOpen10, v30, fine, tool0;
                WaitSyncTask\InPos, sync3, tasklist;
                 g_GripIn;

                WaitSyncTask\InPos, sync4, tasklist;

                WaitSyncTask\InPos, sync5, tasklist;
        MoveAbsJ photoPos\NoEOffs, v50, fine, tool0;

                WaitSyncTask\InPos, sync6, tasklist;
                MoveJ readyToOpen10, v50, fine, tool0;
        WaitSyncTask\InPos, sync7, tasklist;
       WaitSyncTask\InPos, sync8, tasklist;
       WaitSyncTask\InPos, sync10, tasklist;
       g_JogOut;
       MoveL readyToOpen20, v50, fine, tool0;
       WaitSyncTask\InPos, sync9, tasklist;




    ENDPROC
ENDMODULE
```

```
 MODULE MainModule
        PERS wobjdata wobj1:=[FALSE,TRUE,"",
[[186.619,14.041,43.2433],
[0.741199,0.00178913,-0.00792014,-0.671236]],[[0.796976,55.6561,0],
[1,0,0,0.000286155]]];
    CONST string detectjob := "lct.job";
    CONST string readREDjob := "QR_Red.job";
    VAR cameratarget mycameratarget;

        CONST robtarget aboveWP:=[[10.04,5.14,80.52],
[0.00194954,0.743988,0.668187,0.00183217],[1,2,-1,4],
[157.199,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget startPoint:=[[266.38,-323.45,123.19],
[0.264136,-0.898026,0.3062,0.173273],[1,3,-2,4],
[-158.689,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget myrobtarget10:=[[200.32,218.76,74.49],
[0.00104262,-0.924304,-0.381193,0.0187858],[1,2,1,4],
[151.124,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget safeUp:=[[451.88,-197.47,103.88],
[0.00870065,0.69364,-0.720166,-0.0122373],[1,3,0,4],
[178.372,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget startPoint10:=[[253.71,-80.30,187.72],
[0.0085022,-0.0615952,-0.99792,-0.0170172],[1,2,1,4],
[-141.069,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget passPoint:=[[253.71,-80.30,187.72],
[0.00850392,-0.061597,-0.99792,-0.0170158],[1,2,1,4],
[-141.069,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget myrobtarget:=[[9.28,4.07,-0.32],
[0.00195056,0.743987,0.668188,0.00183229],[1,2,-1,4],
[157.198,9E+09,9E+09,9E+09,9E+09,9E+09]];


    VAR syncident sync1;
        VAR syncident sync2;
    VAR syncident sync3;
    VAR syncident sync4;
    VAR syncident sync5;
    VAR syncident sync6;
    VAR syncident sync7;
    VAR syncident sync8;
    VAR syncident sync9;
    VAR syncident sync10;

    PERS tasks tasklist{2}:=[["T_ROB_L"], ["T_ROB_R"]];
        CONST jointtarget
photoPos:=[[50.7374,-108.633,53.0334,208.473,-1.17635,-97.3027],
[-86.458,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST jointtarget
photoPos10:=[[80.6834,-101.674,35.5473,256.881,-57.7592,-178.45],
[-113.714,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget passPoint10:=[[253.71,-209.99,187.72],
[0.00851123,-0.0616072,-0.997919,-0.0170138],[1,2,0,4],
[-141.069,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
    VAR bool yellow :=FALSE;
    VAR bool white :=FALSE;
    VAR bool red :=FALSE;
    VAR bool blue :=FALSE;
    VAR bool green :=FALSE;
    VAR bool blue_in_place :=FALSE;
    VAR bool green_in_place :=FALSE;
    VAR bool yellow_in_place :=FALSE;
    VAR bool white_in_place :=FALSE;
    VAR bool red_in_place :=FALSE;

    VAR bool ready :=FALSE;

        VAR cameratarget cameratarget1;
        CONST robtarget aboveWP10:=[[262.50,42.02,19.97],
[0.00702238,0.738477,-0.674185,-0.00874356],[1,3,0,4],
[153.482,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP20:=[[458.31,24.25,71.33],
[0.00629773,0.68095,-0.732244,-0.00930247],[1,3,0,4],
[153.482,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP30:=[[454.62,24.41,50.10],
[0.00629827,0.680952,-0.732242,-0.00930283],[1,3,0,4],
[153.481,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP40:=[[454.62,24.41,65.21],
[0.0062994,0.68095,-0.732244,-0.00930237],[1,3,0,4],
[153.481,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP50:=[[454.62,50.65,62.50],
[0.00629821,0.68095,-0.732244,-0.00929812],[2,3,0,4],
[153.48,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP60:=[[454.62,50.63,48.68],
[0.00631059,0.680948,-0.732245,-0.00929163],[2,3,0,4],
[153.479,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP70:=[[454.62,43.34,48.68],
[0.00631171,0.680947,-0.732246,-0.00929236],[2,3,0,4],
[153.479,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP80:=[[454.62,55.53,48.68],
[0.00631106,0.680947,-0.732247,-0.00929061],[2,3,0,4],
[153.479,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP90:=[[496.50,55.53,75.78],
[0.00630719,0.680947,-0.732247,-0.00928447],[2,3,0,4],
[153.479,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP100:=[[496.50,24.19,75.78],
[0.00631004,0.680945,-0.732249,-0.0092846],[2,3,0,4],
[153.478,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP110:=[[496.52,24.13,75.78],
[0.00214193,-0.999409,0.0324964,0.0110175],[2,3,-1,4],
[153.478,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP120:=[[496.52,24.14,50.04],
[0.00214212,-0.999409,0.0324936,0.011017],[2,3,-1,4],
[153.477,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP130:=[[481.02,24.14,50.04],
[0.00214223,-0.999409,0.0324931,0.0110165],[2,3,-1,4],
[153.477,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP140:=[[490.42,24.13,50.04],
[0.00213632,-0.999409,0.032496,0.0110142],[2,3,-1,4],
[153.477,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP150:=[[490.42,24.13,73.57],
[0.00213405,-0.999409,0.0324978,0.0110113],[2,3,-1,4],
[153.477,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP160:=[[456.36,20.73,54.91],
[0.0180065,-0.999647,-0.0137112,0.0138836],[1,3,-1,4],
[153.975,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP170:=[[420.01,24.11,73.56],
[0.0021187,-0.999409,0.0325046,0.0110179],[1,3,-1,4],
[153.477,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP180:=[[420.01,24.09,50.17],
[0.0021036,-0.999408,0.0325085,0.0110247],[1,3,-1,4],
[153.477,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP190:=[[430.11,24.09,50.17],
[0.00210359,-0.999408,0.0325103,0.011026],[1,3,-1,4],
[153.477,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP200:=[[424.14,24.09,50.17],
[0.00210401,-0.999408,0.0325103,0.011026],[1,3,-1,4],
[153.477,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP210:=[[424.14,24.09,80.42],
[0.00210221,-0.999408,0.0325109,0.0110248],[1,3,-1,4],
[153.477,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP220:=[[453.59,24.09,80.42],
[0.00210025,-0.999408,0.0325118,0.0110221],[1,3,-1,4],
[153.477,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP230:=[[456.36,20.71,90.08],
[0.0179907,-0.999648,-0.0137035,0.013888],[1,3,-1,4],
[153.975,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP240:=[[456.36,20.71,90.08],
[0.0179898,-0.999648,-0.0137036,0.0138892],[1,3,-1,4],
[153.975,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor2:=[[456.17,24.23,71.32],
[0.00631087,0.680946,-0.732248,-0.00929786],[1,3,0,4],
[153.482,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor12:=[[456.17,24.23,60.42],
[0.00631093,0.680945,-0.732248,-0.00929792],[1,3,0,4],
[153.482,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor22:=[[456.17,44.19,71.31],
[0.00632586,0.680942,-0.732251,-0.0092921],[1,3,0,4],
[153.482,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor32:=[[456.17,45.34,56.30],
[0.00632468,0.680943,-0.73225,-0.00929128],[2,3,0,4],
[153.481,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor42:=[[456.17,43.71,56.30],
[0.00632329,0.680942,-0.732251,-0.00929227],[2,3,0,4],
[153.48,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor52:=[[456.20,53.78,56.30],
[0.00632235,0.680943,-0.73225,-0.00929103],[2,3,0,4],
[153.48,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor62:=[[456.20,53.78,83.10],
[0.00632432,0.680942,-0.732251,-0.00928933],[1,3,0,4],
[153.48,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor72:=[[492.81,22.40,83.10],
[0.00632226,0.680942,-0.732251,-0.00928759],[1,3,0,4],
[153.479,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor82:=[[492.87,22.42,83.10],
[0.0110481,-0.0434475,-0.998993,-0.00202396],[1,3,1,4],
[153.479,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor92:=[[492.87,22.42,59.95],
[0.0110464,-0.0434492,-0.998993,-0.00202316],[2,3,1,4],
[153.478,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor102:=[[480.78,22.42,59.95],
[0.0110476,-0.0434492,-0.998993,-0.00202138],[2,3,1,4],
[153.477,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor112:=[[498.49,22.41,75.55],
[0.011036,-0.0434519,-0.998993,-0.00201052],[2,3,1,4],
[153.477,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor122:=[[419.30,22.41,75.55],
[0.0110417,-0.043456,-0.998992,-0.0020105],[1,3,1,4],
[153.477,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor132:=[[429.61,26.06,56.49],
[0.0110399,-0.0434531,-0.998992,-0.00201012],[1,3,1,4],
[153.477,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor142:=[[409.35,26.06,71.28],
[0.0110395,-0.043454,-0.998992,-0.00200675],[1,3,1,4],
[153.477,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor152:=[[452.35,-2.01,71.28],
[0.0110359,-0.0434544,-0.998992,-0.00200754],[1,3,1,4],
[153.476,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor162:=[[452.35,-2.01,55.74],
[0.0110368,-0.0434533,-0.998993,-0.00200545],[1,3,1,4],
[153.476,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor172:=[[452.35,9.70,55.74],
[0.0110353,-0.0434537,-0.998993,-0.0020077],[1,3,1,4],
[153.476,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor182:=[[452.35,0.17,67.06],
[0.0110352,-0.0434537,-0.998993,-0.00200676],[1,3,1,4],
[153.476,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor192:=[[452.35,25.23,74.56],
[0.011032,-0.0434539,-0.998993,-0.00200736],[1,3,1,4],
[153.475,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor202:=[[452.92,28.61,63.68],
[0.00126995,0.00955331,-0.999856,0.0139835],[1,3,1,4],
[156.397,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor212:=[[452.94,28.28,73.06],
[0.00133556,0.00933585,-0.999849,0.0146266],[1,3,1,4],
[156.363,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor222:=[[446.97,27.99,95.74],
[0.00862998,-0.0208298,-0.999697,0.00986564],[1,3,1,4],
[153.836,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor232:=[[447.40,28.40,66.51],
[0.00827314,-0.0211588,-0.999703,0.00882712],[1,3,1,4],
[153.95,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor242:=[[446.97,27.99,95.91],
[0.00862664,-0.0208302,-0.999697,0.00986575],[1,3,1,4],
[153.836,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor252:=[[453.43,19.41,126.85],
[0.0174637,-0.0228552,-0.998993,0.034441],[1,3,1,4],
[153.77,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget floor262:=[[453.43,-212.51,126.85],
[0.0174588,-0.0228532,-0.998993,0.034443],[1,3,1,4],
[153.77,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget safeUp10:=[[319.34,-73.36,189.82],
[0.10164,0.738029,-0.66437,0.0599575],[1,2,0,4],
[-164.153,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP250:=[[447.96,24.92,77.48],
[0.010928,-0.643277,0.765552,-0.00234169],[1,2,0,4],
[-175.395,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP260:=[[448.32,25.54,69.39],
[0.0107708,-0.687358,0.726232,-0.00298403],[1,2,0,4],
[-175.394,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP270:=[[448.32,25.54,83.88],
[0.0107711,-0.687357,0.726234,-0.00298546],[1,2,0,4],
[-175.394,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP280:=[[448.32,47.56,83.88],
[0.0107714,-0.687357,0.726233,-0.00298611],[1,2,0,4],
[-175.394,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP290:=[[448.32,47.55,68.25],
[0.0107717,-0.687357,0.726233,-0.00298557],[1,2,0,4],
[-175.394,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP300:=[[448.32,42.89,68.25],
[0.0107722,-0.687359,0.726232,-0.00298747],[1,2,0,4],
[-175.393,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP310:=[[492.02,22.95,86.75],
[0.00971065,0.0328106,0.999399,0.00555773],[1,3,1,4],
[-175.393,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP320:=[[492.02,22.95,70.26],
[0.00971239,0.0328108,0.999399,0.00555609],[1,3,1,4],
[-175.393,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP330:=[[475.76,22.95,70.26],
[0.00971375,0.0328105,0.999399,0.00555645],[1,3,1,4],
[-175.393,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP340:=[[417.06,22.95,85.58],
[0.00971085,0.0328128,0.999399,0.00555814],[1,2,1,4],
[-175.392,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP350:=[[417.06,22.95,66.65],
[0.00971141,0.032813,0.999399,0.00555769],[1,2,1,4],
[-175.392,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP360:=[[424.29,22.95,66.65],
[0.00971093,0.0328128,0.999399,0.00555672],[1,2,1,4],
[-175.391,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP370:=[[448.99,0.32,88.69],
[0.00971669,0.0328153,0.999399,0.00555338],[1,2,1,4],
[-175.389,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP380:=[[448.99,0.31,67.46],
[0.00971881,0.0328164,0.999399,0.00555259],[1,2,1,4],
[-175.388,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP390:=[[438.24,9.58,67.46],
[0.00971912,0.0328163,0.999399,0.0055538],[1,2,1,4],
[-175.387,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP400:=[[452.89,23.13,90.84],
[0.00972105,0.0328154,0.999399,0.00555112],[1,2,1,4],
[-175.386,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP410:=[[450.69,20.29,81.52],
[0.00829293,0.0398968,0.99897,0.0199494],[1,3,1,4],
[-178.145,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP420:=[[450.69,20.29,72.25],
[0.00829189,0.0398974,0.99897,0.0199499],[1,3,1,4],
[-178.145,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP430:=[[453.32,50.61,86.74],
[0.00969977,0.0328153,0.999399,0.00554626],[1,2,1,4],
[-175.393,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST jointtarget
photoPos20:=[[53.6922,-67.624,49.9305,274.915,-40.6568,-146.833],
[-99.2732,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget passPoint20:=[[253.71,-195.94,187.72],
[0.00851045,-0.0616068,-0.997919,-0.0170145],[1,2,1,4],
[-141.069,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP440:=[[455.30,0.44,80.41],
[0.00208294,-0.999408,0.0325202,0.0110275],[1,3,-1,4],
[153.476,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP450:=[[455.30,0.44,51.52],
[0.00208284,-0.999408,0.0325167,0.0110258],[1,3,-1,4],
[153.476,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget aboveWP460:=[[455.30,10.51,51.52],
[0.00208341,-0.999408,0.0325183,0.0110275],[1,3,-1,4],
[153.476,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget passPoint30:=[[412.01,-175.97,158.16],
[0.00851856,-0.0616192,-0.997918,-0.0170141],[1,2,0,4],
[-141.067,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget passPoint40:=[[412.01,-175.97,55.73],
[0.0085216,-0.0616192,-0.997918,-0.0170091],[1,2,0,4],
[-141.065,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget above_blue_piece:=[[412.00,-175.98,97.27],
[0.0085328,-0.061629,-0.997918,-0.0170061],[1,2,0,4],
[-141.063,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget at_blue_piece:=[[412.01,-176.06,45.54],
[0.00834812,-0.0616105,-0.99793,-0.0164478],[1,2,0,4],
[-141.064,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget above_green_piece:=[[414.11,-106.77,97.67],
[0.00853287,-0.061626,-0.997918,-0.0170084],[1,2,1,4],
[-141.061,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget at_green_piece:=[[414.10,-106.77,48.53],
[0.00853392,-0.0616244,-0.997918,-0.0170047],[1,2,0,4],
[-141.06,9E+09,9E+09,9E+09,9E+09,9E+09]];

VAR bool first_done :=FALSE;
VAR bool second_done :=FALSE;
VAR bool third_done :=FALSE;
CONST robtarget passPoint50:=[[460.72,-106.35,69.84],
[0.0085238,-0.0616418,-0.997917,-0.0169866],[1,2,0,4],
[-141.061,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget passPoint60:=[[460.72,-106.35,45.95],
[0.0085232,-0.0616436,-0.997917,-0.0169852],[1,2,0,4],
[-141.06,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
CONST robtarget passPoint70:=[[461.75,-175.97,103.63],
[0.0085321,-0.0616276,-0.997918,-0.0170051],[1,2,0,4],
[-141.063,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget passPoint80:=[[461.75,-175.98,47.12],
[0.00853214,-0.0616288,-0.997918,-0.017002],[1,2,0,4],
[-141.062,9E+09,9E+09,9E+09,9E+09,9E+09]];
        PROC main()
       preparation;
       Stop;
       Hand_SetHoldForce(12);
       build_first_floor;


       build_second_floor;

       build_third_floor;
       Stop;


        ENDPROC

    PROC preparation()
        g_SetForce 5;
                 g_SetMaxSpd 3;
                 g_JogOut;
                 g_JogIn;
                 g_Calibrate;
                 g_JogIn;
                 !MoveJ startPoint, v1000, z50, tool0\WObj:=wobj0;
            CamSetProgramMode upCam;
        CamLoadJob upCam, detectjob;
        CamSetRunMode upCam;
        CamSetProgramMode rightHand;
        CamLoadJob rightHand, readREDjob;
        CamSetRunMode rightHand;
    ENDPROC

    PROC build_first_floor()



        MoveJ safeUp, v4000, z50, tool0\WObj:=wobj0;

        CamReqImage upCam;
        CamGetResult upCam, mycameratarget;
        wobj1.oframe := mycameratarget.cframe;
        MoveJ safeUp, v4000, fine, tool0\WObj:=wobj0;
        MoveJ safeUp10, v4000, fine, tool0\WObj:=wobj0;
        MoveJ aboveWP, v2500, fine, tool0\WObj:=wobj1;


        g_GripOut;

        MoveL myrobtarget, v1000, fine, tool0 \WObj:=wobj1;
```

```
g_GripIn;


MoveJ aboveWP, v1000, fine, tool0\WObj:=wobj1;

        WaitSyncTask\InPos, sync1, tasklist;

MoveJ passPoint, v1000, fine, tool0;

        WaitSyncTask\InPos, sync2, tasklist;

        WaitSyncTask\InPos, sync3, tasklist;

        WaitSyncTask\InPos, sync4, tasklist;
g_MoveTo 10;
MoveL passPoint10, v1000, z50, tool0;

        WaitSyncTask\InPos, sync5, tasklist;
g_GripIn;
MoveAbsJ photoPos\NoEOffs, v1000, z50, tool0;
MoveAbsJ photoPos10\NoEOffs, v1000, z50, tool0;
        WaitSyncTask\InPos, sync6, tasklist;
CamReqImage rightHand;

CamGetResult rightHand, cameratarget1;


IF cameratarget1.val2 = 1
THEN
 yellow := NOT yellow;
    startbuilding_first;
ELSEIF
    cameratarget1.val1 = 1
THEN
    white := NOT white;
    put_to_white_pos;
ELSEIF
    cameratarget1.val3 = 1
THEN
    red := NOT red;
    put_to_red_pos;
ELSEIF
    cameratarget1.val4 = 1
THEN
    blue := NOT blue;
    put_to_blue_pos;
ELSEIF
    cameratarget1.val5 = 1
THEN
    green := NOT green;
    put_to_green_pos;
ENDIF
```

```
ENDPROC

PROC build_second_floor()

    IF blue_in_place  THEN
        MoveJ above_blue_piece, v1000, z50, tool0;
        g_MoveTo 10;
        MoveL at_blue_piece, v100, fine, tool0;
        g_GripIn;
        MoveL above_blue_piece, v1000, z50, tool0;
        startbuilding_second_straight;


    ELSE

       MoveJ safeUp, v1000, z50, tool0\WObj:=wobj0;

CamReqImage upCam;
CamGetResult upCam, mycameratarget;
wobj1.oframe := mycameratarget.cframe;
MoveJ safeUp, v1000, z50, tool0\WObj:=wobj0;
MoveL aboveWP, v500, z100, tool0\WObj:=wobj1;






g_MoveTo 15;

MoveL myrobtarget, v1000, fine, tool0 \WObj:=wobj1;
g_GripIn;


MoveJ aboveWP, v1000, fine, tool0\WObj:=wobj1;

 WaitSyncTask\InPos, sync1, tasklist;

MoveJ passPoint, v1000, z50, tool0;

        WaitSyncTask\InPos, sync2, tasklist;

        WaitSyncTask\InPos, sync3, tasklist;

        WaitSyncTask\InPos, sync4, tasklist;
g_MoveTo 10;
MoveL passPoint10, v1000, z50, tool0;

        WaitSyncTask\InPos, sync5, tasklist;
g_GripIn;
MoveAbsJ photoPos\NoEOffs, v1000, z50, tool0;
MoveAbsJ photoPos10\NoEOffs, v1000, z50, tool0;
```

```
            WaitSyncTask\InPos, sync6, tasklist;
    CamReqImage rightHand;

    CamGetResult rightHand, cameratarget1;


    IF
        cameratarget1.val1 = 1
    THEN
        white := NOT white;
        put_to_white_pos;
    ELSEIF
        cameratarget1.val3 = 1
    THEN
        red := NOT red;
        put_to_red_pos;
    ELSEIF
        cameratarget1.val4 = 1
    THEN
        blue := NOT blue;
        startbuilding_second;
    ELSEIF
        cameratarget1.val5 = 1
    THEN
        green := NOT green;
        put_to_green_pos;
    ENDIF




        ENDIF





    ENDPROC

    PROC build_third_floor()

 IF green_in_place  THEN

            MoveJ above_green_piece, v1000, z50, tool0;
            g_MoveTo 10;
            MoveL at_green_piece, v100, fine, tool0;
            g_GripIn;
            MoveL above_green_piece, v1000, z50, tool0;
            startbuilding_third_straight;


        ELSE

     MoveJ safeUp, v1000, z50, tool0\WObj:=wobj0;
```

```
CamReqImage upCam;
CamGetResult upCam, mycameratarget;
wobj1.oframe := mycameratarget.cframe;
MoveJ safeUp, v1000, z50, tool0\WObj:=wobj0;
MoveL aboveWP, v500, z100, tool0\WObj:=wobj1;







g_MoveTo 15;

MoveL myrobtarget, v1000, fine, tool0 \WObj:=wobj1;
g_GripIn;


MoveJ aboveWP, v1000, fine, tool0\WObj:=wobj1;

WaitSyncTask\InPos, sync1, tasklist;

MoveJ passPoint, v1000, z50, tool0;

        WaitSyncTask\InPos, sync2, tasklist;

        WaitSyncTask\InPos, sync3, tasklist;

        WaitSyncTask\InPos, sync4, tasklist;
g_MoveTo 10;
MoveL passPoint10, v1000, z50, tool0;

        WaitSyncTask\InPos, sync5, tasklist;
g_GripIn;
MoveAbsJ photoPos\NoEOffs, v1000, z50, tool0;
MoveAbsJ photoPos10\NoEOffs, v1000, z50, tool0;
        WaitSyncTask\InPos, sync6, tasklist;
CamReqImage rightHand;

CamGetResult rightHand, cameratarget1;


IF
    cameratarget1.val1 = 1
THEN
    white := NOT white;
    put_to_white_pos;
ELSEIF
    cameratarget1.val3 = 1
THEN
    red := NOT red;
    put_to_red_pos;

ELSEIF
    cameratarget1.val5 = 1
```

```
   THEN
       green := NOT green;
       startbuilding_third;
   ENDIF



ENDIF

   ENDPROC
PROC startbuilding_first()
   MoveAbsJ photoPos\NoEOffs, v1000, z50, tool0;
   MoveAbsJ photoPos20\NoEOffs, v1000, z50, tool0;
   MoveJ passPoint10, v1000, z50, tool0;
   g_MoveTo 10;

   WaitSyncTask\InPos, sync7, tasklist;
   MoveL passPoint, v1000, z50, tool0;
   WaitSyncTask\InPos, sync8, tasklist;
   g_GripIn;
   WaitSyncTask\InPos, sync10, tasklist;
   WaitSyncTask\InPos, sync9, tasklist;
   MoveL passPoint20, v1000, z50, tool0;



   MoveL aboveWP20, v2500, fine, tool0;

   MoveL aboveWP30, v2500, fine, tool0;
   g_MoveTo 12;
   MoveL aboveWP40, v1000, fine, tool0;
   g_GripIn;
   MoveL aboveWP50, v100, fine, tool0;
   MoveL aboveWP60, v100, fine, tool0;
   MoveL aboveWP70, v100, fine, tool0;
   MoveL aboveWP80, v100, fine, tool0;
   MoveL aboveWP90, v100, fine, tool0;
   MoveL aboveWP100, v100, fine, tool0;
   MoveJ aboveWP110, v100, fine, tool0;
   MoveL aboveWP120, v100, fine, tool0;
   MoveL aboveWP130, v100, fine, tool0;
   MoveL aboveWP140, v100, fine, tool0;
   MoveL aboveWP150, v100, fine, tool0;
   MoveL aboveWP170, v100, fine, tool0;
```

```
        MoveL aboveWP180, v100, fine, tool0;
        MoveL aboveWP190, v100, fine, tool0;
        MoveL aboveWP200, v100, fine, tool0;
        MoveL aboveWP210, v100, fine, tool0;
        MoveL aboveWP220, v100, fine, tool0;
        MoveL aboveWP440, v100, fine, tool0;
        MoveL aboveWP450, v100, fine, tool0;
        MoveL aboveWP460, v100, fine, tool0;
        MoveL aboveWP450, v100, fine, tool0;
        MoveL aboveWP440, v100, fine, tool0;
        MoveL aboveWP220, v100, fine, tool0;
        g_MoveTo 4.1;

        MoveL aboveWP160, v10, fine, tool0;
        MoveL aboveWP230, v10, fine, tool0;
        g_GripIn;
        MoveL startPoint, v3000, fine, tool0;
        first_done := NOT first_done;
        build_second_floor;
    ENDPROC
    PROC startbuilding_second()
        MoveAbsJ photoPos\NoEOffs, v1000, z50, tool0;
        MoveAbsJ photoPos20\NoEOffs, v1000, z50, tool0;
        MoveJ passPoint10, v1000, z50, tool0;
        g_MoveTo 10;

        WaitSyncTask\InPos, sync7, tasklist;
        MoveL passPoint, v1000, z50, tool0;
        WaitSyncTask\InPos, sync8, tasklist;
        g_GripIn;
        WaitSyncTask\InPos, sync10, tasklist;
        WaitSyncTask\InPos, sync9, tasklist;
        MoveL passPoint20, v1000, z50, tool0;
        MoveL floor2, v2500, fine, tool0;
        MoveL floor12, v2500, fine, tool0;
        g_MoveTo 12;
        MoveL floor2, v1000, fine, tool0;
        g_GripIn;
        MoveL floor22, v100, fine, tool0;
        MoveL floor32, v200, fine, tool0;
        MoveL floor42, v100, fine, tool0;
        MoveL floor52, v100, fine, tool0;
        MoveL floor62, v100, fine, tool0;
        MoveL floor72, v100, fine, tool0;
        MoveJ floor82, v100, fine, tool0;
        MoveL floor92, v100, fine, tool0;
        MoveL floor102, v100, fine, tool0;
        MoveL floor112, v100, fine, tool0;
        MoveL floor122, v100, fine, tool0;
        MoveL floor132, v100, fine, tool0;
        MoveL floor142, v30, fine, tool0;
        MoveL floor152, v30, fine, tool0;
        MoveL floor162, v30, fine, tool0;
        MoveL floor172, v30, fine, tool0;
        MoveL floor182, v30, fine, tool0;
        MoveL floor192, v30, fine, tool0;
        g_MoveTo 4.2;
        MoveL floor212, v20, fine, tool0;
        MoveL floor202, v5, fine, tool0;
        MoveL floor212, v5, fine, tool0;
        MoveL floor222, v20, fine, tool0;






        MoveL startPoint, v3000, fine, tool0;
        second_done := NOT second_done;
        build_third_floor;
    ENDPROC
    PROC startbuilding_second_straight()

        MoveL floor2, v2500, fine, tool0;
        MoveL floor12, v2500, fine, tool0;
        g_MoveTo 12;

     MoveL floor2, v1000, fine, tool0;
        g_GripIn;
        MoveL floor22, v100, fine, tool0;
        MoveL floor32, v200, fine, tool0;
        MoveL floor42, v100, fine, tool0;
        MoveL floor52, v100, fine, tool0;
        MoveL floor62, v100, fine, tool0;
        MoveL floor72, v100, fine, tool0;
        MoveJ floor82, v100, fine, tool0;
        MoveL floor92, v100, fine, tool0;
        MoveL floor102, v100, fine, tool0;
        MoveL floor112, v100, fine, tool0;
        MoveL floor122, v100, fine, tool0;
        MoveL floor132, v100, fine, tool0;
        MoveL floor142, v30, fine, tool0;
        MoveL floor152, v30, fine, tool0;
        MoveL floor162, v30, fine, tool0;
        MoveL floor172, v30, fine, tool0;
        MoveL floor182, v30, fine, tool0;
        MoveL floor192, v30, fine, tool0;
        g_MoveTo 4.2;
        MoveL floor212, v20, fine, tool0;
        MoveL floor202, v5, fine, tool0;
        MoveL floor212, v5, fine, tool0;
        MoveL floor222, v20, fine, tool0;
        MoveL startPoint, v3000, fine, tool0;
        second_done := NOT second_done;
        build_third_floor;

    ENDPROC
    PROC startbuilding_third()

         MoveAbsJ photoPos\NoEOffs, v1000, z50, tool0;
        MoveAbsJ photoPos20\NoEOffs, v1000, z50, tool0;
        MoveJ passPoint10, v1000, z50, tool0;
        g_MoveTo 10;

        WaitSyncTask\InPos, sync7, tasklist;
        MoveL passPoint, v1000, z50, tool0;
        WaitSyncTask\InPos, sync8, tasklist;
        g_GripIn;
        WaitSyncTask\InPos, sync10, tasklist;
        WaitSyncTask\InPos, sync9, tasklist;
        MoveL passPoint20, v1000, z50, tool0;

        MoveL aboveWP250, v2500, fine, tool0\WObj:=wobj0;
        MoveL aboveWP260, v2500, fine, tool0;
        g_MoveTo 15;
        MoveL aboveWP270, v2500, fine, tool0;
        g_GripIn;
        MoveL aboveWP280, v2500, fine, tool0;
        MoveL aboveWP290, v50, fine, tool0;
        MoveL aboveWP300, v50, fine, tool0;
        MoveL aboveWP290, v50, fine, tool0;
        MoveL aboveWP430, v50, fine, tool0;
        MoveL aboveWP310, v50, fine, tool0;
        MoveL aboveWP320, v50, fine, tool0;
        MoveL aboveWP330, v50, fine, tool0;
        MoveL aboveWP320, v50, fine, tool0;
        MoveL aboveWP310, v50, fine, tool0;
        MoveL aboveWP340, v50, fine, tool0;
        MoveL aboveWP350, v50, fine, tool0;
        MoveL aboveWP360, v50, fine, tool0;
        MoveL aboveWP350, v50, fine, tool0;
        MoveL aboveWP340, v50, fine, tool0;
        MoveL aboveWP370, v50, fine, tool0;
        MoveL aboveWP380, v50, fine, tool0;
        MoveL aboveWP390, v50, fine, tool0;
        MoveL aboveWP380, v50, fine, tool0;
        MoveL aboveWP370, v50, fine, tool0;
        MoveL aboveWP400, v50, fine, tool0;
        g_MoveTo 5.1;
        MoveL aboveWP420, v20, fine, tool0;
        MoveL aboveWP410, v50, fine, tool0;
        g_GripIn;

        Stop;

    ENDPROC
```

```
PROC startbuilding_third_straight()
    MoveL aboveWP250, v2500, fine, tool0\WObj:=wobj0;
    MoveL aboveWP260, v2500, fine, tool0;
    g_MoveTo 15;
    MoveL aboveWP270, v2500, fine, tool0;
    g_GripIn;
    MoveL aboveWP280, v2500, fine, tool0;
    MoveL aboveWP290, v50, fine, tool0;
    MoveL aboveWP300, v50, fine, tool0;
    MoveL aboveWP290, v50, fine, tool0;
    MoveL aboveWP430, v50, fine, tool0;
    MoveL aboveWP310, v50, fine, tool0;
    MoveL aboveWP320, v50, fine, tool0;
    MoveL aboveWP330, v50, fine, tool0;
    MoveL aboveWP320, v50, fine, tool0;
    MoveL aboveWP310, v50, fine, tool0;
    MoveL aboveWP340, v50, fine, tool0;
    MoveL aboveWP350, v50, fine, tool0;
    MoveL aboveWP360, v50, fine, tool0;
    MoveL aboveWP350, v50, fine, tool0;
    MoveL aboveWP340, v50, fine, tool0;
    MoveL aboveWP370, v50, fine, tool0;
    MoveL aboveWP380, v50, fine, tool0;
    MoveL aboveWP390, v50, fine, tool0;
    MoveL aboveWP380, v50, fine, tool0;
    MoveL aboveWP370, v50, fine, tool0;
    MoveL aboveWP400, v50, fine, tool0;
    g_MoveTo 5.1;
    MoveL aboveWP420, v20, fine, tool0;
    MoveL aboveWP410, v50, fine, tool0;
    g_GripIn;

    Stop;
ENDPROC
PROC put_to_white_pos()
    MoveAbsJ photoPos\NoEOffs, v1000, z50, tool0;
    MoveAbsJ photoPos20\NoEOffs, v1000, z50, tool0;
    MoveJ passPoint10, v1000, z50, tool0;
    g_MoveTo 10;

    WaitSyncTask\InPos, sync7, tasklist;
    MoveL passPoint, v1000, z50, tool0;
    WaitSyncTask\InPos, sync8, tasklist;
    g_GripIn;
    WaitSyncTask\InPos, sync10, tasklist;
    WaitSyncTask\InPos, sync9, tasklist;
    MoveL passPoint20, v1000, z50, tool0;
    MoveL passPoint30, v1000, z50, tool0;
    MoveL passPoint50, v1000, z50, tool0;
    MoveL passPoint60, v1000, fine, tool0;
        g_MoveTo 10;
        MoveL passPoint50, v1000, z50, tool0;
    white_in_place :=not white_in_place;
     IF first_done THEN
```

```
        build_second_floor;
    ELSEIF not first_done THEN
    build_first_floor;
    ELSEIF first_done and second_done THEN
        build_third_floor;
     ENDIF
ENDPROC
    PROC put_to_red_pos()
    MoveAbsJ photoPos\NoEOffs, v1000, z50, tool0;
    MoveAbsJ photoPos20\NoEOffs, v1000, z50, tool0;
    MoveJ passPoint10, v1000, z50, tool0;
    g_MoveTo 10;

    WaitSyncTask\InPos, sync7, tasklist;
    MoveL passPoint, v1000, z50, tool0;
    WaitSyncTask\InPos, sync8, tasklist;
    g_GripIn;
    WaitSyncTask\InPos, sync10, tasklist;
    WaitSyncTask\InPos, sync9, tasklist;
    MoveL passPoint20, v1000, z50, tool0;
    MoveL passPoint30, v1000, z50, tool0;
    MoveL passPoint70, v1000, z50, tool0;
    MoveL passPoint80, v1000, fine, tool0;
        g_MoveTo 10;
        MoveL passPoint70, v1000, z50, tool0;
    red_in_place :=not red_in_place;
     IF first_done THEN
        build_second_floor;
    ELSEIF not first_done THEN
    build_first_floor;
    ELSEIF first_done and second_done THEN
        build_third_floor;
     ENDIF
ENDPROC
PROC put_to_blue_pos()

    MoveAbsJ photoPos\NoEOffs, v1000, z50, tool0;
    MoveAbsJ photoPos20\NoEOffs, v1000, z50, tool0;
    MoveJ passPoint10, v1000, z50, tool0;
    g_MoveTo 10;

    WaitSyncTask\InPos, sync7, tasklist;
    MoveL passPoint, v1000, z50, tool0;
    WaitSyncTask\InPos, sync8, tasklist;
    g_GripIn;
    WaitSyncTask\InPos, sync10, tasklist;
    WaitSyncTask\InPos, sync9, tasklist;
    MoveL passPoint20, v1000, z50, tool0;
    MoveL passPoint30, v1000, z50, tool0;
    MoveL passPoint40, v1000, z50, tool0;
    MoveL at_blue_piece, v100, fine, tool0;
        g_MoveTo 10;
    MoveL above_blue_piece, v1000, z50, tool0;
    blue_in_place :=not blue_in_place;
```

```
        build_first_floor;

    ENDPROC
    PROC put_to_green_pos()
        MoveAbsJ photoPos\NoEOffs, v1000, z50, tool0;
        MoveAbsJ photoPos20\NoEOffs, v1000, z50, tool0;
        MoveJ passPoint10, v1000, z50, tool0;
        g_MoveTo 10;

        WaitSyncTask\InPos, sync7, tasklist;
        MoveL passPoint, v1000, z50, tool0;
        WaitSyncTask\InPos, sync8, tasklist;
        g_GripIn;
        WaitSyncTask\InPos, sync10, tasklist;
        WaitSyncTask\InPos, sync9, tasklist;
        MoveL passPoint20, v1000, z50, tool0;
        MoveL passPoint30, v1000, z50, tool0;
        MoveL above_green_piece, v1000, z50, tool0;
        MoveL at_green_piece, v100, fine, tool0;
            g_MoveTo 10;
            MoveL above_green_piece, v1000, z50, tool0;
        green_in_place :=not green_in_place;
        IF first_done THEN
            build_second_floor;
        ELSEIF not first_done THEN
        build_first_floor;

        ENDIF

    ENDPROC


ENDMODULE
```

B Appendix

B.1 Java codes

This Appendix is intended to document Java programs, described in 3.5. The programs have been created in Eclipse software. All the files in .java formate can be found in Git repository.[26]

```
package mygame;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.PrintStream;
import java.net.Socket;
import java.net.UnknownHostException;

import com.jme3.math.Vector3f;

public class socket {

    private static Socket socket;

    public static void InitializeSocket() {

        ObjectInputStream inStream = null;
        ObjectOutputStream outStream = null;

        try {
            socket = new Socket("192.168.125.1", 4000);
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    public static Socket getSocket() {
        return socket;
    }

    public static void sendCommand(String s) throws
UnknownHostException, IOException {

        ObjectInputStream inStream = null;
        ObjectOutputStream outStream = null;

        PrintStream p = new PrintStream(socket.getOutputStream());
        p.println(s);

        InputStream input = socket.getInputStream();
        BufferedReader reader = new BufferedReader(new
InputStreamReader(input));
        String line = reader.readLine();
        System.out.println(line);

        if (line.equals("received ")) {
            System.out.println("continue");
        } else {
            System.exit(0);
            ;
        }

    }

    public static void sendCommand1(String s, Vector3f vec, String
rt) throws UnknownHostException, IOException {

        ObjectInputStream inStream = null;
        ObjectOutputStream outStream = null;

        PrintStream p = new PrintStream(socket.getOutputStream());
        p.println(s + ";" + rt + ";" + vec.toString());
        // System.out.println(p);

        InputStream input = socket.getInputStream();
        BufferedReader reader = new BufferedReader(new
InputStreamReader(input));
        String line = reader.readLine();
        // System.out.println(line);

        if (line.equals("received ")) {
            System.out.println("done");
        } else {
            System.exit(0);
            ;
        }

    }

}
```

```
package mygame;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Set;

import com.jme3.asset.AssetManager;
import com.jme3.math.FastMath;
import com.jme3.math.Vector3f;
import com.jme3.scene.Node;
import com.prosysopc.ua.ServiceException;
import com.prosysopc.ua.client.AddressSpaceException;
import com.prosysopc.ua.nodes.UaNode;
import com.prosysopc.ua.nodes.UaReference;

import aalto.types.DigitalProductDescription.CadPartType;
import aalto.types.DigitalProductDescription.CoordinateType;
import aalto.types.DigitalProductDescription.DigitalPartType;
import aalto.types.DigitalProductDescription.RectangleLegoType;
import aalto.types.DigitalProductDescription.SquareLegoType;
import opcua.Client;

/**
 *
 * @author ssierla
 */
public class DigitalTwin {
    private int legoNum = 1; // for debugging

    public Node node = new Node();
    private HashMap<String, DigitalPart> map = new HashMap<String,
DigitalPart>();
    private ArrayList connections = new ArrayList();
    private int connectionsSize = 0;
    int count = 0;
    private RobotArm assemblyArm;
    private AssemblyStation assemblyStation;
    private float assemblySurfaceHeight;
    private MobileRobot mobileRobot;

    ArrayList<DigitalPart> legos = new ArrayList(100);
    ArrayList<DigitalPart> unassembledLegos = new ArrayList(100);
    private DigitalPart lego;
    private DigitalPart twinLego;
    private Trajectory trajectory;
    float maxHeight = 4;
    int legosSize;
    float rot;
    float rotPrev = 0;
    int rotCounter = 0;
    boolean trajectoryMotion = false;
    boolean bottomReady = false;
    boolean exploringTrajectory; // with digital twin
    boolean rotationMotion; // as opposed to straight line motion

    public static boolean connectingUpward = false;
    boolean gotoLego = false;
    boolean assemblyInProgress = false;
    Vector3f trajectoryPoint;
    // int approachAttempt = 0;
    private boolean collisionDetection = false;
    private boolean virtualAssemblyComplete = false;
    private boolean physicalAssembly = false;
    private boolean waitingForLego = false;
    private boolean initialRotation = false;
    private Cell cell;
    private float biggestX = Main.dim * 10; // dimension of the
DigitalPart - default is for legos
    private float biggestZ = Main.dim * 10;

    public void initPhysicalAssembly() {
        System.exit(0);
        Main.helloText.setText("Physical assembly");
        for (int i = 0; i < legosSize; i++) {
            legos.get(i).assembled = false;
            legos.get(i).state = LegoState.ToBeFetched;
            node.detachChild(legos.get(i).node);
            if (legos.get(i).northSouthOrientation()) {
                legos.get(i).node.rotate(0, -1 * FastMath.HALF_PI,
0);
            }
            legos.get(i).translate(legos.get(i).startLocation);
        }
        physicalAssembly = true;
        bottomReady = false;
        // Main.assemblyArm.rotateBack();
        assemblyArm.initRotate(this);
        connectingUpward = false;

        // RobotArm.step = 0.02f;
        // RobotArm.maxRotationStep = 300;
        cell.attachNodeToRoot(true);
        cell.getMobileRobot().start = true;
    }

    public void receivePhysicalLego(String id) {
        // System.out.println("Received " + id);
        // RobotArm.step = 0.1f;
        for (int i = 0; i < legosSize; i++) {
            if (legos.get(i).id.equals(id)) {
                legos.get(i).state = LegoState.AtAssembly;
                node.attachChild(legos.get(i).node);
                legos.get(i).node.rotate(-1 *
legos.get(i).orientation.getX(), -1 *
legos.get(i).orientation.getY(),
                        -1 * legos.get(i).orientation.getZ());
                //
legos.get(i).translate(legos.get(i).startLocation);
            }
            /*
             * if(legos.get(i).northSouthOrientation())
{ legos.get(i).node.rotate(0,
             * -1*FastMath.HALF_PI, 0); }
             */

        }
    }

    private boolean connectUpward(DigitalPart lego) {
        if (!lego.isLego()) {
            return false;
        }
        for (int i = 0; i < connectionsSize; i++) {
            LegoConnection connection = (LegoConnection)
connections.get(i);

            // if"lego" is to be connected to the bottom of an
assembled lego return true
            if (lego.equals(connection.lego2) &&
connection.lego1.assembled) {
                String cp1 = connection.connectionPoint1;
                if ((cp1.equals("bottom") || cp1.equals("bottomA")
|| cp1.equals("bottomB") || cp1.equals("bottomC"))) {
                    return true;
                }
            }

            if (lego.equals(connection.lego1) &&
connection.lego2.assembled) {
                String cp2 = connection.connectionPoint2;
                if ((cp2.equals("bottom") || cp2.equals("bottomA")
|| cp2.equals("bottomB") || cp2.equals("bottomC"))) {
                    return true;
                }
            }
        }
        return false;
    }

    public DigitalTwin(List<DigitalPartType> parts, Set<UaReference>
conns, AssetManager assetManager, Cell c,
            Client uaClient) throws ServiceException,
AddressSpaceException {
        cell = c;
        mobileRobot = c.getMobileRobot();
        assemblyStation = cell.requestAssemblyStation();
        assemblyArm = assemblyStation.assemblyArm;
        assemblySurfaceHeight = assemblyStation.getSurfaceHeight();

        /*
         * Iterating through all the parts and creating a
corresponding 3D-object.
         */
        boolean firstPart = true;
        DigitalPart p = null;
        for (DigitalPartType part : parts) {
            if (uaClient.compareType(part,
aalto.types.DigitalProductDescription.Ids.SquareLegoType)) {
                SquareLegoType square = (SquareLegoType) part;
                String name = square.getDisplayName().getText();
                String color = square.getColor();
                p = new Square(name, color, assetManager);
            } else if (uaClient.compareType(part,
aalto.types.DigitalProductDescription.Ids.RectangleLegoType)) {
                RectangleLegoType rect = (RectangleLegoType) part;
                String name = rect.getDisplayName().getText();
                String color = rect.getColor();
                String orient = rect.getOrientation();
                p = new Rectangle(name, color, orient,
assetManager);
            } else {
                CadPartType cad = (CadPartType) part;
                String name = cad.getDisplayName().getText();
                String color = cad.getColor();
                String orient = cad.getOrientation();
                String typeName =
cad.getTypeDefinition().getDisplayName().getText();
                String path = cad.getPathToModel();
                p = new CADpart(typeName, name, color,
assetManager, orient, path);
                if ((2 * p.getXextent()) > biggestX) {
                    biggestX = p.getXextent() * 2 + Main.dim;
                }
                if ((p.getZextent() * 2) > biggestZ) {
                    biggestZ = p.getZextent() * 2 + Main.dim;
                }
            }
            p.twin = this;
            map.put(p.id, p);
            if (firstPart) {
                firstPart = false;
                p.translateTarget(new Vector3f(-7, 0, -10));
                p.assembled = true;
                p.assemblyLocation = p.targetLocation;
            }
            node.attachChild(p.node);
            legos.add(p);
        }

        /*
```

```
         * This is here only for CAD-parts to work properly. It
gets all cadTypes from
         * the address space and calls add3dparam once for each
connectionPoint of each
         * type.
         *
         * This is run every time the constructor is called and it
adds all cadTypes
         * that are in the address space even if the digitalTwin
has no cad parts.
         */
        for (UaNode type : uaClient.getCadTypes()) {
            String typename = type.getDisplayName().getText();
            for (CoordinateType connectionPoint :
uaClient.getConnectionPoints(type)) {
                String name =
connectionPoint.getDisplayName().getText();
                String coordinates = connectionPoint.getX() + "," +
connectionPoint.getY() + ","
                        + connectionPoint.getZ();
                CADpart.add3dParam(typename + name, coordinates);
            }
        }

        /*
         * Iterating through all OPC UA references between
connectionPoint and creating
         * corresponding Connection-object.
         */
        for (UaReference conn : conns) {
            UaNode connectionPoint1 = conn.getSourceNode();
            UaNode connectionPoint2 = conn.getTargetNode();
            UaNode part1 = uaClient.getPart(connectionPoint1);
            UaNode part2 = uaClient.getPart(connectionPoint2);
            LegoConnection connection = new LegoConnection();
            connection.lego1 =
map.get(part1.getDisplayName().getText());
            connection.lego2 =
map.get(part2.getDisplayName().getText());
            connection.connectionPoint1 =
connectionPoint1.getDisplayName().getText();
            connection.connectionPoint2 =
connectionPoint2.getDisplayName().getText();
            connections.add(connection);
            connectionsSize++;
        }

        assemblyStation.setSlotSpacing(biggestX, biggestZ);

        Iterator it = map.entrySet().iterator();
        DigitalPart lego;
        int index = 0;
        while (it.hasNext()) {
            HashMap.Entry pair = (HashMap.Entry) it.next();
            lego = (DigitalPart) pair.getValue();
            // lego.startLocation = new Vector3f(6.0f - 12.0f *
(index%2),
            // assemblySurfaceHeight + Main.dim, -4.0f - 2.0f *
(index/2));
            lego.startLocation =
assemblyStation.slotPosition(index, lego.getYextent());
            lego.translate(lego.startLocation);
            // it.remove(); // avoids a
ConcurrentModificationException
            index++;
        }

        // Now go through the lego connections and translate each
lego into the correct
        // place
        // We iterate until we find an unconnected connection
involving one lego that
        // was already assembled
        // With this tactic, we have to iterate over the
connections several times until
        // everything is connected
        boolean allConnected = false;
        while (!allConnected) {
            allConnected = true;
            Iterator<LegoConnection> itr = connections.iterator();
            while (itr.hasNext()) {
                Vector3f connectionPoint;

                LegoConnection connection = itr.next();
                if (connection.connected) {
                    continue;
                }
                allConnected = false;
                if (connection.lego1.assembled && !
connection.lego2.assembled) {
                    connectionPoint =
connection.lego1.getConnectionPoint(connection.connectionPoint1);
                    //
System.out.println(connectionPoint.toString());
                    // Now translate lego2 to the connectionPoint
of lego1

connection.lego2.translateTarget(connectionPoint);
                    //
System.out.println(connection.lego2.targetLocation.toString());
                    // Now translate lego2 so that its appropriate
connectionPoint is being
                    // connected

connection.lego2.translateConnectionPoint(connection.connectionPoint
2);
                    //
System.out.println(connection.lego2.targetLocation.toString());
```

```
                    connection.lego2.assembled = true;
                    connection.lego2.assemblyLocation =
connection.lego2.targetLocation;
                    connection.connected = true;
                }

                if (!connection.lego1.assembled &&
connection.lego2.assembled) {
                    connectionPoint =
connection.lego2.getConnectionPoint(connection.connectionPoint2);
                    // Now translate lego1 to the connectionPoint
of lego2

connection.lego1.translateTarget(connectionPoint);
                    // Now translate lego1 so that its appropriate
connectionPoint is being
                    // connected

connection.lego1.translateConnectionPoint(connection.connectionPoint
1);
                    connection.lego1.assembled = true;
                    connection.lego1.assemblyLocation =
connection.lego1.targetLocation;
                    connection.connected = true;
                }
            }
        }
        legosSize = legos.size();
        lego = legos.get(0);
        lego.moving = true;

        float bottom = 1000;
        for (int i = 0; i < legosSize; i++) {
            legos.get(i).assembled = false;
            if (legos.get(i).assemblyLocation.getY() < bottom) {
                bottom = legos.get(i).assemblyLocation.getY();
            }
        }
        for (int i = 0; i < legosSize; i++) {
            if (legos.get(i).assemblyLocation.getY() < bottom +
0.001f) {
                legos.get(i).bottomLayer = true;
                mobileRobot.addToQueue(legos.get(i));
            }
        }
        float distanceToFloor = bottom - assemblySurfaceHeight -
lego.getYextent(); // 3*Main.dim;
        for (int i = 0; i < legosSize; i++) {

legos.get(i).assemblyLocation.setY(legos.get(i).assemblyLocation.get
Y() - distanceToFloor);
        }
    }
    public DigitalTwin(String fileName, AssetManager assetManager,
Cell c) {
        cell = c;
        mobileRobot = c.getMobileRobot();
        assemblyStation = cell.requestAssemblyStation();
        assemblyArm = assemblyStation.assemblyArm;
        assemblySurfaceHeight = assemblyStation.getSurfaceHeight();

        try {
            BufferedReader reader = new BufferedReader(new
FileReader(fileName));
            int lineCounter = 0;
            String line;

            boolean firstPart = true;
//uncomment
            String start = new String("Start");
            // socket.InitializeSocket();
            // socket.sendCommand(start);

            while ((line = reader.readLine()) != null) {
                String words[] = line.split(" ");

                if (words[0].equals("//")) {
                    continue;
                }

                if (words[0].equals("create")) {
                    if (words[1].equals("square") ||
words[1].equals("SquareLego")) {
                        Square s = new Square(words[2], words[3],
assetManager);
                        s.twin = this;
                        map.put(s.id, s);
                        if (firstPart) {
                            firstPart = false;
                            s.translateTarget(new Vector3f(-7, 0,
-10));
                            s.assembled = true;
                            s.assemblyLocation = s.targetLocation;
                        }
                        node.attachChild(s.node);
                        legos.add(s);
                    } else if (words[1].equals("rectangle") ||
words[1].equals("RectangleLego")) {
                        Rectangle r = new Rectangle(words[2],
words[3], words[4], assetManager);
                        r.twin = this;
                        map.put(r.id, r);
                        if (firstPart) {
                            firstPart = false;
                            r.translateTarget(new Vector3f(-7, 0,
-10));
                            r.assembled = true;
```

```
                             r.assemblyLocation = r.targetLocation;
                         }
                         node.attachChild(r.node);
                         legos.add(r);
                     } else { // we assume it is a CADpart
                         CADpart p = new CADpart(words[1],
words[2], words[3], assetManager, words[4]);
                         p.twin = this;
                         map.put(p.id, p);
                         if (firstPart) {
                             firstPart = false;
                             p.translateTarget(new Vector3f(-7, 0,
-10));

                             p.assembled = true;
                             p.assemblyLocation = p.targetLocation;
                         }
                         node.attachChild(p.node);
                         legos.add(p);
                         if ((2 * p.getXextent()) > biggestX) {
                             biggestX = p.getXextent() * 2 +
Main.dim;
                         }
                         if ((p.getZextent() * 2) > biggestZ) {
                             biggestZ = p.getZextent() * 2 +
Main.dim;
                         }
                     }

                     /*
                      * if (words[1].equals("faceplateback")||
words[1].equals("FaceplateBack")) {
                      * FaceplateBack f = new
FaceplateBack(words[2], assetManager); f.twin = this;
                      * map.put(f.id,f); if (firstPart) { firstPart
= false; f.translateTarget(new
                      * Vector3f(-7,0,-10)); f.assembled = true;
f.assemblyLocation =
                      * f.targetLocation; }
node.attachChild(f.node); legos.add(f); } if
                      * (words[1].equals("boltangular")||
words[1].equals("BoltAngular")) {
                      * BoltAngular b = new BoltAngular(words[2],
assetManager); b.twin = this;
                      * map.put(b.id,b); if (firstPart) { firstPart
= false; b.translateTarget(new
                      * Vector3f(-7,0,-10)); b.assembled = true;
b.assemblyLocation =
                      * b.targetLocation; }
node.attachChild(b.node); legos.add(b); } if
                      * (words[1].equals("bolt")||
words[1].equals("Bolt")) { Bolt b = new
                      * Bolt(words[2], assetManager); b.twin = this;
map.put(b.id,b); if (firstPart)
                      * { firstPart = false; b.translateTarget(new
Vector3f(-7,0,-10)); b.assembled =
                     * true; b.assemblyLocation =
b.targetLocation; } node.attachChild(b.node);
                     * legos.add(b); } if
(words[1].equals("shaft")|| words[1].equals("Shaft")) {
                     * Shaft s = new Shaft(words[2], assetManager);
s.twin = this; map.put(s.id,s);
                     * if (firstPart) { firstPart = false;
s.translateTarget(new
                     * Vector3f(-7,0,-10)); s.assembled = true;
s.assemblyLocation =
                     * s.targetLocation; }
node.attachChild(s.node); legos.add(s); } if
                     * (words[1].equals("pendulum")||
words[1].equals("Pendulum")) { Pendulum p =
                     * new Pendulum(words[2], assetManager,
words[3], words[4]); p.twin = this;
                     * map.put(p.id,p); if (firstPart) { firstPart
= false; p.translateTarget(new
                     * Vector3f(-7,0,-10)); p.assembled = true;
p.assemblyLocation =
                     * p.targetLocation; }
node.attachChild(p.node); legos.add(p); }
                     */
                 }

                 if (words[0].equals("param")) {
                     CADpart.add3dParam(words[1] + words[2],
words[3]);
                 }

                 if (words[0].equals("connect")) {
                     LegoConnection connection = new
LegoConnection();
                     connection.lego1 = map.get(words[1]);
                     connection.lego2 = map.get(words[3]);
                     connection.connectionPoint1 = words[2];
                     connection.connectionPoint2 = words[4];
                     connections.add(connection);
                     connectionsSize++;
                     /*
                      * String cp1 = connection.connectionPoint1;
if((cp1.equals("bottom") ||
                      * cp1.equals("bottomA") ||
cp1.equals("bottomB") || cp1.equals("bottomC"))) {
                      * connection.lego1.connectUpward = false; }
                      *
                      * String cp2 = connection.connectionPoint2;
if((cp2.equals("bottom") ||
                      * cp2.equals("bottomA") ||
cp2.equals("bottomB") || cp2.equals("bottomC"))) {
                      * connection.lego2.connectUpward = false; }
                      */
                 }
```

```
            }

        } catch (IOException e) {
            System.out.println("IOexception");
        }

        assemblyStation.setSlotSpacing(biggestX, biggestZ);

        Iterator it = map.entrySet().iterator();
        DigitalPart lego;
        int index = 0;
        while (it.hasNext()) {
            HashMap.Entry pair = (HashMap.Entry) it.next();
            lego = (DigitalPart) pair.getValue();
            // lego.startLocation = new Vector3f(6.0f - 12.0f *
(index%2),
            // assemblySurfaceHeight + Main.dim, -4.0f - 2.0f *
(index/2));
            lego.startLocation =
assemblyStation.slotPosition(index, lego.getYextent());
            lego.translate(lego.startLocation);
            // it.remove(); // avoids a
ConcurrentModificationException
            index++;
        }

        // Now go through the lego connections and translate each
lego into the correct
        // place
        // We iterate until we find an unconnected connection
involving one lego that
        // was already assembled
        // With this tactic, we have to iterate over the
connections several times until
        // everything is connected
        boolean allConnected = false;
        while (!allConnected) {
            allConnected = true;
            Iterator<LegoConnection> itr = connections.iterator();
            while (itr.hasNext()) {
                Vector3f connectionPoint;

                LegoConnection connection = itr.next();
                if (connection.connected) {

                    continue;
                }
                allConnected = false;
                if (connection.lego1.assembled && !
connection.lego2.assembled) {
                    connectionPoint =
connection.lego1.getConnectionPoint(connection.connectionPoint1);
                    //
System.out.println(connectionPoint.toString());
                     // Now translate lego2 to the connectionPoint
of lego1

connection.lego2.translateTarget(connectionPoint);
                     //
System.out.println(connection.lego2.targetLocation.toString());
                     // Now translate lego2 so that its appropriate
connectionPoint is being
                     // connected

connection.lego2.translateConnectionPoint(connection.connectionPoint
2);
                     //
System.out.println(connection.lego2.targetLocation.toString());
                     connection.lego2.assembled = true;
                     connection.lego2.assemblyLocation =
connection.lego2.targetLocation;
                     connection.connected = true;
                 }

                 if (!connection.lego1.assembled &&
connection.lego2.assembled) {
                     connectionPoint =
connection.lego2.getConnectionPoint(connection.connectionPoint2);
                     // Now translate lego1 to the connectionPoint
of lego2

connection.lego1.translateTarget(connectionPoint);
                     // Now translate lego1 so that its appropriate
connectionPoint is being
                     // connected

connection.lego1.translateConnectionPoint(connection.connectionPoint
1);
                     connection.lego1.assembled = true;
                     connection.lego1.assemblyLocation =
connection.lego1.targetLocation;
                     connection.connected = true;
                 }
             }
         }
         legosSize = legos.size();
         lego = legos.get(0);
         lego.moving = true;

         float bottom = 1000;
         for (int i = 0; i < legosSize; i++) {
             legos.get(i).assembled = false;
             if (legos.get(i).assemblyLocation.getY() < bottom) {
                 bottom = legos.get(i).assemblyLocation.getY();
             }
         }
         for (int i = 0; i < legosSize; i++) {
             if (legos.get(i).assemblyLocation.getY() < bottom +
 0.001f) {
                 legos.get(i).bottomLayer = true;
                 mobileRobot.addToQueue(legos.get(i));
             }
         }
         float distanceToFloor = bottom - assemblySurfaceHeight -
 lego.getYextent(); // 3*Main.dim;
         for (int i = 0; i < legosSize; i++) {

 legos.get(i).assemblyLocation.setY(legos.get(i).assemblyLocation.get
 Y() - distanceToFloor);
         }
     }

     // goes through all elements in list "legos" and puts
 unassembled legos into the
     // list "unassembledLegos"
     // if they have a connection to an assembled lego
     // then it goes through the list again and returns the one with
 the lowest "y"
     // assembly location
     public DigitalPart nextUnassembledLego() throws
 UnknownHostException, IOException {
         unassembledLegos.clear();
         for (int i = 0; i < legosSize; i++) {
             if (legos.get(i).assembled == false) {
                 for (int j = 0; j < connections.size(); j++) {
                     LegoConnection conn = (LegoConnection)
 connections.get(j);
                     if (conn.lego1 == legos.get(i)) {
                          if (conn.lego2.assembled) {
                              unassembledLegos.add(legos.get(i));
                          }
                     }
                     if (conn.lego2 == legos.get(i)) {
                          if (conn.lego1.assembled) {
                              unassembledLegos.add(legos.get(i));
                          }
                     }
                 }
             }
         }
         if (unassembledLegos.isEmpty()) {
 //uncomment
             String stopp = new String("Stop");
             // socket.InitializeSocket();
             // socket.sendCommand(stopp);

             System.out.println("all connected");
             return null;

         }
         DigitalPart retVal = unassembledLegos.get(0);
         int index = 0;
         for (int k = 1; k < unassembledLegos.size(); k++) {
             if (unassembledLegos.get(k).assemblyLocation.getY() <
 retVal.assemblyLocation.getY()) {
                 retVal = unassembledLegos.get(k);
                 index = k;
             }
         }
         if (!physicalAssembly) {
             mobileRobot.addToQueue(retVal);
         }
         return retVal;
     }

     public void immediateAssembly() {
         for (int i = 0; i < legosSize; i++) {
             legos.get(i).translate(legos.get(i).targetLocation);
             if (legos.get(i).northSouthOrientation()) {
                 legos.get(i).node.rotate(0, FastMath.HALF_PI, 0);
             }
         }
     }

     public void collisionDetected() {
         collisionDetection = false;
         assemblyArm.rotateBack();
         assemblyArm.initRotate(this);
         rotationMotion = false; // in case a rotation was in
 progress, we clear this boolean
         createUnderneathApproachTrajectory();
     }

     public void createUnderneathApproachTrajectory() {
         trajectory = new Trajectory();
         Vector3f v = lego.startLocation;
         v = v.add(new Vector3f(0, 2.0f * Main.dim, 0));

         assemblyArm.initMove(v, this);
         // Main.markerGeom.setLocalTranslation(v);
         count++;
         /*
          * if (count > 4) { RobotArm.step = 0.01f; //
 RobotArm.maxRotationStep = 500; }
          */

         gotoLego = true;
         Vector3f v0 = new Vector3f(lego.startLocation);
         v0.setY(maxHeight);
         trajectory.addPoint(v0);
         Vector3f v1 = new Vector3f(lego.assemblyLocation);
         v1.setY(maxHeight);
         if (!physicalAssembly) {
             lego.approachAttempt += 1;
         }
         if (lego.approachAttempt == 1) { // towards negative X axis
             trajectory.addPoint(v1.add(new Vector3f(20f * Main.dim, 
 0, 0)));
             if (lego.northSouthOrientation()) {
                 trajectory.addRotateMove(new Vector3f(0, 
 FastMath.HALF_PI, 0));
             }
             Vector3f v3 = new 
 Vector3f(lego.assemblyLocation.add(new Vector3f(30f * Main.dim, 2f * 
 Main.dim, 0)));
             trajectory.addPoint(v3);
             if (lego.northSouthOrientation()) {
                 trajectory.addRotateMove(new 
 Vector3f(FastMath.HALF_PI, 0, 0));
             } else {
                 trajectory.addRotateMove(new Vector3f(0, 0, -
 FastMath.HALF_PI));
             }
             trajectory.addPoint(
                     new Vector3f(lego.assemblyLocation.add(new 
 Vector3f(20f * Main.dim, -4.5f * Main.dim, 0))));
             trajectory.addPoint(new 
 Vector3f(lego.assemblyLocation.add(new Vector3f(0, -4.5f * Main.dim, 
 0))));
             trajectory.addPoint(new 
 Vector3f(lego.assemblyLocation.add(new Vector3f(0, 0f * Main.dim, 
 0)))); // used to

                                                // be 2f

                                                // instead

                                                // of 0f

                                                // for y

                                                // coordinate
             lego.node.rotate(FastMath.PI, 0, 0);
         } else if (lego.approachAttempt == 2) { // toward positive 
 X axis
             trajectory.addPoint(v1.add(new Vector3f(-27f * 
 Main.dim, 0, 0)));
             if (lego.northSouthOrientation()) {
                 trajectory.addRotateMove(new Vector3f(0, 
 FastMath.HALF_PI, 0));
             }
             Vector3f v3 = new 
 Vector3f(lego.assemblyLocation.add(new Vector3f(-27f * Main.dim, 
 0.5f * Main.dim, 0)));
             trajectory.addPoint(v3);
             if (lego.northSouthOrientation()) {
                 trajectory.addRotateMove(new Vector3f(-
 FastMath.HALF_PI, 0, 0));
```

```
            } else {
                trajectory.addRotateMove(new Vector3f(0, 0,
FastMath.HALF_PI));
            }
            trajectory.addPoint(
                    new Vector3f(lego.assemblyLocation.add(new
Vector3f(-20f * Main.dim, -4.5f * Main.dim, 0))));
            trajectory.addPoint(new
Vector3f(lego.assemblyLocation.add(new Vector3f(0, -4.5f * Main.dim,
0))));
            trajectory.addPoint(new
Vector3f(lego.assemblyLocation.add(new Vector3f(0, 0f * Main.dim,
0))));
            if (physicalAssembly) {
                lego.node.rotate(FastMath.PI, 0, 0);
            }
        } else if (lego.approachAttempt == 3) { // toward positive
Z axis
            trajectory.addPoint(v1.add(new Vector3f(0, 0, -15f *
Main.dim)));
            if (lego.northSouthOrientation()) {
                trajectory.addRotateMove(new Vector3f(0,
FastMath.HALF_PI, 0));
            }
            Vector3f v3 = new
Vector3f(lego.assemblyLocation.add(new Vector3f(0, 2f * Main.dim,
-15f * Main.dim)));
            trajectory.addPoint(v3);
            if (lego.northSouthOrientation()) {
                trajectory.addRotateMove(new Vector3f(0, 0, -
FastMath.HALF_PI));
            } else {
                trajectory.addRotateMove(new Vector3f(-
FastMath.HALF_PI, 0, 0));
            }
            trajectory.addPoint(
                    new Vector3f(lego.assemblyLocation.add(new
Vector3f(0, -4.5f * Main.dim, -5f * Main.dim))));
            trajectory.addPoint(new
Vector3f(lego.assemblyLocation.add(new Vector3f(0, -4.5f * Main.dim,
0))));
            trajectory.addPoint(new
Vector3f(lego.assemblyLocation.add(new Vector3f(0, 0f * Main.dim,
0))));
            if (physicalAssembly) {
                lego.node.rotate(FastMath.PI, 0, 0);
            }
        } else if (lego.approachAttempt == 4) { // toward negative
Z axis
            trajectory.addPoint(v1.add(new Vector3f(0, 0, 15f *
Main.dim)));
            if (lego.northSouthOrientation()) {
                trajectory.addRotateMove(new Vector3f(0,
FastMath.HALF_PI, 0));
```

```
            }
            Vector3f v3 = new
Vector3f(lego.assemblyLocation.add(new Vector3f(0, 2f * Main.dim,
15f * Main.dim)));
            trajectory.addPoint(v3);
            if (lego.northSouthOrientation()) {
                trajectory.addRotateMove(new Vector3f(0, 0,
FastMath.HALF_PI));
            } else {
                trajectory.addRotateMove(new
Vector3f(FastMath.HALF_PI, 0, 0));
            }
            trajectory.addPoint(
                    new Vector3f(lego.assemblyLocation.add(new
Vector3f(0, -4.5f * Main.dim, 5f * Main.dim))));
            trajectory.addPoint(new
Vector3f(lego.assemblyLocation.add(new Vector3f(0, -4.5f * Main.dim,
0))));
            trajectory.addPoint(new
Vector3f(lego.assemblyLocation.add(new Vector3f(0, 0f * Main.dim,
0))));
            if (physicalAssembly) {
                lego.node.rotate(FastMath.PI, 0, 0);
            }
        } else {
            Main.helloText.setText("Unable to assemble this product
due to collisions");
            Main.freeze = true;
        }
        trajectory.initTrajectory();
        trajectoryMotion = true;
    }

    /*
     * public boolean virtualAssemblyComplete() { return
virtualAssemblyComplete; }
     */
    public void execute(float tpf) throws IOException, IOException {
        /*
         * if (connectingUpward) { Main.helloText.setText("Up"); }
else {
         * Main.helloText.setText("Down"); }
         */
        // physical assembly

        // GREEN CODE STARTS HERE
        if (waitingForLego) {

            if (lego.state == LegoState.AtAssembly) {

                waitingForLego = false;

            } else {
                return;

            }

        }
        // GREEN CODE ENDS HERE

        if (!physicalAssembly && virtualAssemblyComplete) {
            // Main.freeze = true;
            initPhysicalAssembly();
            RobotArm.step *= 2;
            // RobotArm.maxRotationStep = 1000;
            return;
        }

        if (gotoLego) { // moving to lego start position
            if (initialRotation) { // rotate before going to lego
                if (assemblyArm.rotate(trajectoryPoint, false)) {
                    initialRotation = false;
                    return;
                } else {
                    return;
                }
            }

            // BLUE CODE STARTS HERE
            gotoLego = assemblyArm.move();
            // BLUE CODE ENDS HERE

            /*
             * if(!gotoLego && (lego.orientation.getZ() > 0.1f))
{ Main.freeze = true;
             * return; }
             */

            if (collisionDetection) {
                assemblyArm.checkCollision();
            }

            if (!gotoLego && !assemblyArm.legoAttached()) {
                /*
                 * if(legoNum == 2) { lego.connectArm(assemblyArm);
                 * lego.node.setLocalTranslation(0, -3.0f *
Main.dim - lego.getYextent(), 0);
                 * Main.freeze = true; return; } else {
                 */

                String rotation = new String();
                if (lego.northSouthOrientation()) {
                    trajectory.addRotateMove(new Vector3f(0,
FastMath.HALF_PI, 0));
                    String rota = new String("90");

                    // System.out.println(lego.id + " " + rota);
                    rotation = rota;

                } else {
                    String rotno = new String("00");
                    rotation = rotno;

                }

                Vector3f y =
lego.targetLocation.subtract(legos.get(0).targetLocation);

                System.out.println(lego.id + " " + y);

//uncomment

                // socket.InitializeSocket();
                // socket.sendCommand1(lego.id, y, rotation);

                lego.connectArm(assemblyArm);

                /*
                 * if(lego.orientation != null)
{ if(lego.orientation.getZ() > 0.1f) {
                 * System.out.println(lego.getXextent());
System.out.println(lego.getYextent());
                 * System.out.println(lego.getZextent());
                 * lego.node.setLocalRotation(Quaternion.ZERO);
                 *
lego.node.setLocalTranslation(-1f*lego.getXextent(), -3*Main.dim,
                 * 1f*lego.getYextent()); } } else {
                 */
                lego.node.setLocalTranslation(0, -3.0f * Main.dim -
lego.getYextent(), 0);
                // }
                /*
                 * if(lego.orientation.getZ() > 0.1f) { Main.freeze
= true; return; }
                 */
                // lego.node.setLocalTranslation(0, -4.0f *
Main.dim, 0);

            }
            return; // we do not go on to do any of the other
motions that apply to the case that we
                    // have gripped a lego
        }

        if (trajectoryMotion) {
            if (assemblyInProgress) {
                assemblyInProgress = assemblyArm.move();
                if (collisionDetection && connectingUpward &&
lego.isLego()) {
                    assemblyArm.checkCollision();
                }
```

```
                 return;
             }

             if (rotationMotion) {
                 boolean bothJoints = true;
                 if (lego.orientation != null) {
                     if (lego.orientation.getZ() > 0.1f) {
                         bothJoints = false;
                     }
                 }
                 boolean ready = assemblyArm.rotate(trajectoryPoint,
bothJoints);
                 if (collisionDetection && lego.isLego()) {
                     assemblyArm.checkCollision();
                 }
                 if (ready) {
                     rotationMotion = false;
                 } else {
                     return;
                 }

             }
             trajectoryPoint = trajectory.nextPoint();
             if (!virtualAssemblyComplete) {
                 collisionDetection = true; // it will be set to
true when the arm is at lego start position with lego on
                                              // board
             }
             if (trajectoryPoint == null) { // we are done with this
lego
                 /*
                  * if(lego.id.equalsIgnoreCase("rect11"))
{ Main.freeze = true; return; }
                  */
                 legoNum++;
                 // approachAttempt = 1; // reset for next lego
                 lego.location = lego.node.getWorldTranslation();
                 trajectoryMotion = false;
                 lego.assembled = true;

                 lego.disconnectArm();
                 node.attachChild(lego.node);
                 if (connectingUpward) {
                     assemblyArm.rotateBack();
                     connectingUpward = false;
                     lego.node.rotate(FastMath.PI, 0, 0);
                     // assemblyArm.step = 0.05f;
                 } else {
                     assemblyArm.initRotate(this); // initRotate
resets everything else apart from the y rotation
                     assemblyArm.rotateBack();
                 }

             } else {
                 if (trajectory.rotationMotion()) {
                     assemblyArm.initRotate(this);
                     rotationMotion = true;
                     return;
                 }
                 assemblyArm.initMove(trajectoryPoint, this);

                 if (lego.id.equalsIgnoreCase("rect2")) {
                     //
Main.helloText.setText(lego.assemblyLocation.toString() + " : " +
                     // lego.targetLocation.toString());
                 }

                 assemblyInProgress = true;
                 return;
             }
         }

         // if we get down here it means that the previous lego was
assembled and now we
         // need to find the next lego
         collisionDetection = false;
         if (bottomReady) {
             // RED CODE STARTS HERE
             lego = nextUnassembledLego();
             // RED CODE ENDS HERE
             if (lego == null) {
                 if (physicalAssembly) {
                     Main.freeze = true;
                     cell.getMobileRobot().start = false;
                     return;
                 }
                 virtualAssemblyComplete = true;
                 return;
             }
             if (physicalAssembly && (lego.state !=
LegoState.AtAssembly)) {
                 // System.out.println("WAITING for: " + lego.id);
                 waitingForLego = true;
             }

             if (connectUpward(lego)) {
                 connectingUpward = true;
                 createUnderneathApproachTrajectory();
                 // lego.node.rotate(FastMath.PI, 0, 0);
                 return;
             }

             trajectory = new Trajectory();
             if (lego.rotationAxis.equalsIgnoreCase("z")) {
                 trajectoryPoint = new Vector3f(FastMath.HALF_PI, 0,
0);
                 initialRotation = true;
             }
             lego.moving = false;
             if (lego.rotationAxis.equalsIgnoreCase("z")) {
                 // assemblyArm.step = 0.05f;
                 // RobotArm.maxRotationStep = 500;
                 Vector3f v = lego.startLocation.add(new Vector3f(0,
0, lego.getZextent()));
                 assemblyArm.initMove(v, this);
                 assemblyArm.connectionPoint = "z";

             } else {

assemblyArm.initMove(lego.startLocation.add(lego.getGripperOffset())
, this);
                 assemblyArm.connectionPoint = null;
             }

             gotoLego = true;
             Vector3f v0 = new Vector3f(lego.startLocation);
             v0.setY(maxHeight);
             trajectory.addPoint(v0);
             /*
              * Vector3f v1 = new Vector3f(lego.startLocation);
v1.setY(maxHeight);
              * trajectory.addPoint(v1);
              */

             Vector3f v2 = new Vector3f(lego.assemblyLocation);
             v2.setY(maxHeight);
             trajectory.addPoint(v2);

             /*
              * if(lego.northSouthOrientation())
{ trajectory.addRotateMove(new
              * Vector3f(0,FastMath.HALF_PI,0)); String rota = new
String("90");
              * //System.out.println(rota); socket sock = new
socket();
              * sock.sendCommand(lego.id,v2,rota);
              *
              *
              * } else { String rotno = new String("00");
System.out.println(lego.id + " " +
              * rotno); socket sock = new socket();
sock.sendCommand(lego.id,v2, rotno);
              *
              * }
              */
             if (!lego.isLego() && (!lego.orientationZero)) {
                 trajectory.addRotateMove(lego.orientation);
             }

trajectory.addPoint(lego.assemblyLocation.add(lego.getGripperOffset(
)));
```

```
            trajectory.initTrajectory();
            trajectoryMotion = true;
            return;
        }

        // if we get down here it means that the bottom layer is
not yet assembled
        boolean unassembledBottomLegoExists = false; // ready =
true;
        // look for an unassembled bottom layer lego
        for (int i = 0; i < legosSize; i++) {
            if (legos.get(i).bottomLayer && !
legos.get(i).assembled) {
                unassembledBottomLegoExists = true;

                trajectory = new Trajectory();
                lego = legos.get(i);
                lego.moving = false;

assemblyArm.initMove(lego.startLocation.add(lego.getGripperOffset())
, this);

Main.markerGeom.setLocalTranslation(lego.startLocation.add(lego.getG
ripperOffset()));
                gotoLego = true;
                Vector3f v1 = new Vector3f(lego.startLocation);
                v1.setY(maxHeight);
                trajectory.addPoint(v1);
                Vector3f v2 = new Vector3f(lego.assemblyLocation);
                v2.setY(maxHeight);
                trajectory.addPoint(v2);
                // trajectory.addPoint(new
Vector3f(lego.assemblyLocation.add(new Vector3f(0,
                // 0.5f*Main.dim, 0))));

trajectory.addPoint(lego.assemblyLocation.add(lego.getGripperOffset(
)));
                trajectory.initTrajectory();
                trajectoryMotion = true;
                if (physicalAssembly && (lego.state !=
LegoState.AtAssembly)) {
                    // System.out.println("WAITING for: " +
lego.id);
                    waitingForLego = true;
                }
                return;
            }
        }
        if (unassembledBottomLegoExists == false) {
            bottomReady = true;

        }

    }
```

```
}
```

B.2 RAPID codes

This Appendix is intended to document RAPID program, described in 3.6. The program has been created in ABB RobotStudio software. All the files in .MOD formate can be found in Git repository.[26]

```
MODULE MainModule
    VAR socketdev serverSocket;
    VAR socketdev clientSocket;

    VAR string data;

    VAR string start;
    VAR string stopcheck;
    VAR bool okX;
    VAR bool okY;
    VAR bool okZ;
    VAR bool okid;
    VAR num foundx;
    VAR num foundxend;
    VAR num xlength;
    VAR num foundy;
    VAR num foundyend;
    VAR num ylength;
    VAR num foundz;
    VAR num foundzend;
    VAR num zlength;
    VAR num findid;
    VAR num findrot;
    VAR num findsp;
    VAR num idlength;
    VAR num rotlength;
    VAR num tryX;
    VAR num tryY;
    VAR num tryZ;
    VAR num wpID;

    VAR string rotinfo;
    VAR string partx;
    VAR string party;
    VAR string partz;

    VAR string nameWP;
    VAR string rotation;
    VAR string partID;

    VAR bool dummy:=FALSE;

    VAR bool rotreq:=FALSE;

    PERS wobjdata wobj1:=[FALSE,TRUE,"",[[402.123,-85.4556,45.7888],
[0.714337,0.00384811,-0.00185838,-0.699789]],[[0,0,0],[1,0,0,0]]];

    CONST robtarget aboveSquare:=[[392.82,84.68,85.85],
[0.008429,0.720698,0.693198,0.000710105],[1,-2,2,4],
[-157.745,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget atSquare:=[[392.82,84.68,46.20],
[0.00843173,0.720699,0.693197,0.000710762],[1,-2,2,4],
[-157.745,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget aboveRect3:=[[390.39,123.44,88.20],
[0.0129949,0.702215,0.711759,0.0111123],[1,-2,2,4],
[-157.585,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget atRect3:=[[390.39,123.44,47.84],
[0.0129977,0.702214,0.71176,0.0111121],[1,-2,2,4],
[-157.585,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget atRect8:=[[391.14,172.92,45.46],
[0.0128495,0.708706,0.70537,0.00485425],[2,-2,2,4],
[-158.649,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget aboveRect8:=[[391.14,172.92,84.90],
[0.0128481,0.708706,0.70537,0.00485415],[1,-2,2,4],
[-158.649,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget atRect7:=[[425.86,172.54,47.86],
[0.0128518,0.708709,0.705368,0.00482815],[2,-2,2,4],
[-158.652,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget aboveRect7:=[[423.27,172.56,88.78],
[0.0128498,0.708707,0.705369,0.00485353],[2,-2,2,4],
[-158.652,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget atRect6:=[[452.98,172.55,48.75],
[0.0128577,0.708712,0.705364,0.00483975],[2,-2,2,4],
[-158.657,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget aboveRect6:=[[455.40,172.56,90.45],
[0.0128522,0.70871,0.705367,0.0048541],[2,-2,2,4],
[-158.656,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget atRect9:=[[359.26,172.53,45.95],
[0.012845,0.708706,0.70537,0.00481702],[1,-2,2,4],
[-158.656,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget aboveRect9:=[[359.75,172.57,93.39],
[0.0128469,0.708708,0.705369,0.00485374],[1,-2,2,4],
[-158.656,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget atRect10:=[[325.50,172.57,45.66],
[0.0128436,0.708708,0.705368,0.00484549],[1,-2,2,4],
[-158.655,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget aboveRect10:=[[327.54,172.57,91.06],
[0.0128477,0.708708,0.705369,0.00485525],[1,-2,2,4],
[-158.655,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget atRect5:=[[327.49,124.36,46.05],
[0.0128455,0.708708,0.705368,0.00485574],[1,-2,2,4],
[-158.65,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget aboveRect5:=[[327.49,124.36,89.91],
[0.0128451,0.708708,0.705369,0.00485555],[1,-2,2,4],
[-158.65,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget atRect4:=[[358.18,124.35,46.47],
[0.012843,0.708709,0.705367,0.00484029],[1,-2,2,4],
[-158.647,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget aboveRect4:=[[359.41,124.36,87.84],
[0.0128455,0.708709,0.705367,0.00485432],[1,-2,2,4],
[-158.647,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget atRect2:=[[423.36,124.36,47.83],
[0.0128491,0.708709,0.705367,0.00485238],[1,-2,2,4],
[-158.648,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget aboveRect2:=[[423.36,124.36,95.78],
[0.0128489,0.708712,0.705364,0.00485348],[1,-2,2,4],
[-158.647,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget atRect1:=[[455.30,124.36,48.38],
[0.0128577,0.708716,0.70536,0.00485569],[2,-2,2,4],
[-158.647,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget aboveRect1:=[[455.29,124.37,107.15],
[0.0128557,0.708719,0.705357,0.00485413],[1,-2,2,4],
[-158.647,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget safepos:=[[-85.23,-112.11,194.62],
[0.0451027,-0.414455,-0.908361,0.0327424],[1,-2,3,4],
[-176.223,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST jointtarget home:=[[0,-130,30,0,40,0],
[-135,9E+09,9E+09,9E+09,9E+09,9E+09]];

    VAR robtarget trytarget;
    VAR robtarget trytarget2;
    VAR robtarget pushtarget;
    VAR robtarget pushtarget2;
    VAR robtarget abovetargetrot;

    VAR orient rotresta:=[0.00860358,-0.694976,0.718946,0.00714151];
    VAR confdata rotrestb:=[1,-2,1,4];
    VAR extjoint rotrestc:=[-146.79,9E+09,9E+09,9E+09,9E+09,9E+09];

    PROC main()





        Hand_SetHoldForce(12);





        Calibration;
        Initialization;





        WHILE dummy DO

            SocketCreate serverSocket;
            SocketBind serverSocket,"192.168.125.1",4000;
            SocketListen serverSocket;
            SocketAccept serverSocket,clientSocket,\Time:=WAIT_MAX;
            SocketReceive clientSocket\Str:=data;

            stopcheck:=StrPart(data,1,4);
            IF stopcheck="Stop" THEN
                dummy:=FALSE;
                MoveAbsJ home\NoEOffs,v7000,fine,tool0\WObj:=wobj0;
                SendConfirmation;
            ELSE
```

```
!Read rotation info

!Get the ID number

findid:=StrFind(data,1,STR_DIGIT);
findsp:=StrFind(data,1,";");
idlength:=findsp-findid;
partID:=StrPart(data,findid,idlength);
okid:=StrToVal(partID,wpID);


findrot:=StrFind(data,findsp+1,";");
rotlength:=findrot-(findsp+1);
rotation:=StrPart(data,findsp+1,rotlength);
IF rotation="90" THEN
    rotreq:=TRUE;
ELSEIF rotation="00" THEN
    rotreq:=FALSE;
ENDIF




!Get the ID number

findid:=StrFind(data,1,STR_DIGIT);
findsp:=StrFind(data,1,";");
idlength:=findsp-findid;
partID:=StrPart(data,findid,idlength);
okid:=StrToVal(partID,wpID);
nameWP:=StrPart(data,1,findid-1);
IF nameWP="Square" THEN
    MoveJ aboveSquare,vmax,fine,tool0;
    g_MoveTo 19.7;
    MoveL atSquare,vmax,fine,tool0;
    g_JogIn;
    MoveL aboveSquare,vmax,fine,tool0;
ELSEIF nameWP="Rectan" THEN
    IF wpID=1 THEN
        !go to square1
        MoveJ aboveRect1,vmax,fine,tool0;
        g_MoveTo 19.7;
        MoveL atRect1,vmax,fine,tool0;
        g_GripIn;
        MoveJ aboveRect1,vmax,fine,tool0;
    ELSEIF wpID=2 THEN
        !go to square2
        MoveJ aboveRect2,vmax,fine,tool0;
        g_MoveTo 19.7;
        MoveL atRect2,vmax,fine,tool0;
        g_GripIn;
        MoveJ aboveRect2,vmax,fine,tool0;
    ELSEIF wpID=3 THEN
```

```
        MoveJ aboveRect3,v1000,fine,tool0;
        g_MoveTo 19.7;
        MoveL atRect3,v100,fine,tool0;
        g_GripIn;
        MoveJ aboveRect3,v1000,fine,tool0;
    ELSEIF wpID=4 THEN
        !go to square4
        MoveJ aboveRect4,vmax,fine,tool0;
        g_MoveTo 19.7;
        MoveL atRect4,vmax,fine,tool0;
        g_GripIn;
        MoveJ aboveRect4,vmax,fine,tool0;
    ELSEIF wpID=5 THEN
        !go to square5
        MoveJ aboveRect5,vmax,fine,tool0;
        g_MoveTo 19.7;
        MoveL atRect5,vmax,fine,tool0;
        g_GripIn;
        MoveJ aboveRect5,vmax,fine,tool0;
    ELSEIF wpID=6 THEN
        !go to square6
        MoveJ aboveRect6,vmax,fine,tool0;
        g_MoveTo 19.7;
        MoveL atRect6,vmax,fine,tool0;
        g_GripIn;
        MoveJ aboveRect6,vmax,fine,tool0;
    ELSEIF wpID=7 THEN
        !go to square7
        MoveJ aboveRect7,vmax,fine,tool0;
        g_MoveTo 19.7;
        MoveL atRect7,vmax,fine,tool0;
        g_GripIn;
        MoveJ aboveRect7,vmax,fine,tool0;
    ELSEIF wpID=8 THEN
        !go to square8
        MoveJ aboveRect8,vmax,fine,tool0;
        g_MoveTo 19.7;
        MoveL atRect8,vmax,fine,tool0;
        g_GripIn;
        MoveJ aboveRect8,vmax,fine,tool0;
    ELSEIF wpID=9 THEN
        !go to square9
        MoveJ aboveRect9,vmax,fine,tool0;
        g_MoveTo 19.7;
        MoveL atRect9,vmax,fine,tool0;
        g_GripIn;
        MoveJ aboveRect9,vmax,fine,tool0;
    ELSEIF wpID=10 THEN
        !go to square10
        MoveJ aboveRect10,vmax,fine,tool0;
        g_MoveTo 19.7;
        MoveL atRect10,vmax,fine,tool0;
        g_GripIn;
        MoveJ aboveRect10,vmax,fine,tool0;
```

```
                    ENDIF


                ENDIF



                foundx:=StrFind(data,1,"(");
                foundxend:=StrFind(data,1,",");
                xlength:=foundxend-(foundx+1);
                partx:=StrPart(data,foundx+1,xlength);
                okX:=StrToVal(partx,tryX);
                foundz:=StrFind(data,foundxend+1,",");

                zlength:=foundz-(foundxend+1);
                partz:=StrPart(data,foundxend+1,zlength);
                okZ:=StrToVal(partz,tryZ);
                foundy:=StrFind(data,foundz+1,")");
                ylength:=foundy-(foundz+1);
                party:=StrPart(data,foundz+1,ylength);
                okY:=StrToVal(party,tryY);

                trytarget:=[[tryX*24,tryY*24,tryZ*24+30],
[0.00702104,-0.00906265,-0.999928,0.00352561],[1,-2,2,4],
[-145.573,9E+09,9E+09,9E+09,9E+09,9E+09]];




                MoveJ safepos,vmax,fine,tool0\WObj:=wobj1;

                MoveJ trytarget,vmax,fine,tool0,\Wobj:=wobj1;

                IF rotreq=TRUE THEN
                    !apply rotation of the last joint

abovetargetrot:=[[tryX*24,tryY*24,tryZ*24+30],rotresta,rotrestb,rotr
estc];
                    MoveJ
abovetargetrot,vmax,fine,tool0\WObj:=wobj1;


trytarget2:=[[tryX*24-1.6,tryY*24,tryZ*24+2.5],rotresta,rotrestb,rot
restc];
                    MoveL trytarget2,v100,fine,tool0\WObj:=wobj1;

                    g_GripOut;

                    MoveL
abovetargetrot,vmax,fine,tool0\WObj:=wobj1;


                    FOR i FROM 1 TO 2 DO
```

```
pushtarget:=[[tryX*24,tryY*24+5.44,tryZ*24+20],
[0.00707566,0.000428977,-0.999969,0.00344461],[1,-2,1,4],
[-105.113,9E+09,9E+09,9E+09,9E+09,9E+09]];

                        MoveL
pushtarget,v100,fine,tool0,\Wobj:=wobj1;

                        g_MoveTo 1;

pushtarget2:=[[tryX*24,tryY*24+5.44,tryZ*24+7.64],
[0.00707566,0.000428977,-0.999969,0.00344461],[1,-2,1,4],
[-105.113,9E+09,9E+09,9E+09,9E+09,9E+09]];
                        MoveL
pushtarget2,v10,fine,tool0,\Wobj:=wobj1;
                        MoveL
pushtarget,v100,fine,tool0,\Wobj:=wobj1;

pushtarget2:=[[tryX*24,tryY*24+5.44-10,tryZ*24+7.64],
[0.00707566,0.000428977,-0.999969,0.00344461],[1,-2,1,4],
[-105.113,9E+09,9E+09,9E+09,9E+09,9E+09]];
                        MoveL
pushtarget2,v10,fine,tool0,\Wobj:=wobj1;
                        MoveL
pushtarget,v100,fine,tool0,\Wobj:=wobj1;

pushtarget2:=[[tryX*24,tryY*24+5.44+10,tryZ*24+7.64],
[0.00707566,0.000428977,-0.999969,0.00344461],[1,-2,1,4],
[-105.113,9E+09,9E+09,9E+09,9E+09,9E+09]];
                        MoveL
pushtarget2,v10,fine,tool0,\Wobj:=wobj1;
                        MoveL
pushtarget,v100,fine,tool0,\Wobj:=wobj1;
                    ENDFOR
                    MoveJ safepos,vmax,fine,tool0\WObj:=wobj1;
                ELSE
                    !just continue

                    trytarget2:=[[tryX*24+1,tryY*24+2,tryZ*24+1.5],
[0.00702104,-0.00906265,-0.999928,0.00352561],[1,-2,2,4],
[-145.573,9E+09,9E+09,9E+09,9E+09,9E+09]];
                    MoveL trytarget2,v100,fine,tool0\WObj:=wobj1;


                    g_MoveTo 19.7;

                    MoveL trytarget,vmax,fine,tool0,\Wobj:=wobj1;
                    FOR i FROM 1 TO 2 DO
                        pushtarget:=[[tryX*24,tryY*24,tryZ*24+20],
[0.00209069,-0.727174,-0.686409,0.00744442],[1,-2,2,4],
[-98.2663,9E+09,9E+09,9E+09,9E+09,9E+09]];

                        MoveL
pushtarget,v100,fine,tool0,\Wobj:=wobj1;
```

```
                        g_MoveTo 1;

pushtarget2:=[[tryX*24,tryY*24+1,tryZ*24+7.64],
[0.00209069,-0.727174,-0.686409,0.00744442],[1,-2,2,4],
[-98.2663,9E+09,9E+09,9E+09,9E+09,9E+09]];
                        MoveL
pushtarget2,v10,fine,tool0,\Wobj:=wobj1;
                        MoveL
pushtarget,v100,fine,tool0,\Wobj:=wobj1;

pushtarget2:=[[tryX*24+7,tryY*24+1,tryZ*24+7.64],
[0.00209069,-0.727174,-0.686409,0.00744442],[1,-2,2,4],
[-98.2663,9E+09,9E+09,9E+09,9E+09,9E+09]];
                        MoveL
pushtarget2,v10,fine,tool0,\Wobj:=wobj1;
                        MoveL
pushtarget,v100,fine,tool0,\Wobj:=wobj1;

pushtarget2:=[[tryX*24-7,tryY*24+1,tryZ*24+7.64],
[0.00209069,-0.727174,-0.686409,0.00744442],[1,-2,2,4],
[-98.2663,9E+09,9E+09,9E+09,9E+09,9E+09]];
                        MoveL
pushtarget2,v10,fine,tool0,\Wobj:=wobj1;
                        MoveL
pushtarget,v100,fine,tool0,\Wobj:=wobj1;


                    ENDFOR
                    MoveJ safepos,vmax,fine,tool0\WObj:=wobj1;
                ENDIF

                SendConfirmation;


            ENDIF
        ENDWHILE






    ERROR
        IF ERRNO=ERR_SOCK_TIMEOUT THEN
            RETRY;

        ELSEIF ERRNO=ERR_SOCK_CLOSED THEN
            SocketClose clientSocket;
            SocketClose serverSocket;
            SocketCreate serverSocket;
            SocketBind serverSocket,"192.168.125.1",4000;
            SocketListen serverSocket;

            SocketAccept serverSocket,clientSocket,\Time:=WAIT_MAX;
            RETRY;





        ENDIF
    ENDPROC


    PROC Calibration()
        g_SetForce 5;
        g_SetMaxSpd 10;
        g_JogOut;
        g_JogIn;
        g_Calibrate;
        g_JogIn;
    ENDPROC

    PROC Initialization()
        SocketCreate serverSocket;
        SocketBind serverSocket,"192.168.125.1",4000;
        SocketListen serverSocket;
        SocketAccept serverSocket,clientSocket,\Time:=WAIT_MAX;
        SocketReceive clientSocket\Str:=data;
        start:=StrPart(data,1,5);
        IF start="Start" THEN

            dummy:=TRUE;
            MoveJ safepos,vmax,fine,tool0\WObj:=wobj1;
        ELSE
            dummy:=FALSE;
        ENDIF

        SendConfirmation;
    ENDPROC

    PROC SendConfirmation()

        SocketSend clientSocket\Str:="received ";
        SocketClose clientSocket;
        SocketClose serverSocket;
    ENDPROC



ENDMODULE
```

…