Lab University of Applied Sciences
Technology, Lappeenranta
Mechanical Engineering and Production Technology

Dong Le

Application for the ABB IRB 14000 YuMi robot using Integrated Vision and 3D printing

Thesis 2020

Abstract

Dong Le
Application for the ABB IRB 14000 YuMi robot using Integrated Vision and 3D printing, 73 pages, 6 appendices
Lab University of Applied Sciences
Technology, Lappeenranta
Department of Mechanical Engineering and Production Technology
Thesis 2020
Instructors: Jouni Könönen, Lab University of Applied Sciences.

The purpose of the study was to design a demo application for ABB IRB 14000 YuMi robot. The thesis work was made for the Department of Robotics at Technical University of Ostrava, situated in Ostrava, Czech Republics. Computer-aided design was conducted by the use of PTC Creo Parametric, ABB rapid programming, image processing, and virtual simulation were conducted in ABB RobotStudio.

Based on the thought of how to make the best use of the Yumi robot, the truly collaborative robot of ABB, the idea of the demo application came up. Although several ideas were brought to the table, the demo application using dices was chosen, because the Yumi robot can use 2 arms synchronically instead of using 2 robots to carry out the task, furthermore, the application required an image processing task, thus the visual function in the left arm was necessary. In this demo application, the ABB IRB 14000 proves that it is outweighed other traditional industrial robots in some tasks that require robots mimic human action and vision task without an outsource camera system.

This thesis describes the whole demo application process from brainstorming ideas, background research, 3D concept design, calculation, program, virtual simulation to the finished tests on the YuMi robot. After completing the demo application, the end-user has the ability to run the application without set-up, pre-knowledge of robotics, and programming.

The result of the project can be helpful for further research of the collaborative robot, 3D printing technology, and image processing on ABB integrated camera for industrial robots.

Keywords: collabotive robot, ABB YuMi, 3D printing, ABB camera, raspid program, ABB RobotStudio

Table of contents

List of Terminology

ABB	ASEA Brown Boveri
IRB	Industrial Robots
PTC	Parametric Technology Corporation
DOF	Degrees of Freedom
ISO	International Organization of Standardization
ROI	Return on investment
Cobot	Collaborative robot
TCP	Tool center point
EoAT	end-of-arm tooling
VGR	Vision Guided Robotics

1 Introduction

Lacking human features in industries using robots is causing some arguments. Along with robots, there are many things that a human can brings into their jobs to be a plus in industries such as analysis, communication, creativity, and decision-making in unique situations. That’s why cobots appeared to combine human factors with robotics, which leads to incredible success in the industry.

Whereas a traditional industrial robot is designed to complete a specific pre-defined task quickly, accurately in a specific area without interupting, a cobot is designed to interact and collaborate with humans safely in a shared workspace.

The advent of Industry 4.0 technologies and the smart factory makes an traditional industrial robot out of cages, barriers, and become smaller, more flexible, brought to the table to work along side with humans.

1.1 Goal

The goal of the thesis is to design a demo application for the collaborative robot ABB Yumi. The application should perform successfully in the virtual simulation, as well as in the ABB Yumi robot. The demo application will pick up a holder and a cover, then shake dices, after that throw the dices into a playground, return the holder and cover to the original positions, and uses its built-in camera to recognize which number of dice on the playground/workplace, then order from high to lower, finally drops the dices back into the holder and repeat the application.

1.2 Objectives

The first objective of this thesis is to expose a principle knowledge of cobot and ABB industrial robot YuMi. The study provides the functions of ABB cobot, ABB Rapid language program, and RobotStudio software for simulation. From that, the application was programmed on RobotStudio and ran on the YuMi robot successfully.

The second purpose is to conduct research on 3D printing technology and Prusa printers, the knowledge was obtained to design mechanical gripper fingers for the

robot by using PTC Creo CAD design software for 3d modeling and the original Prusa i3 MK3 printer for 3D printing.

The third goal is to conduct some researches on Cognex cameras for the industrial robots with Insight software, and the built-in camera on the gripper of the ABB YuMi robot. Base on the learning, the camera's jobs were processed image recognition with 100% accurate during the application.

The last target is to have problem-solving skill, critical thinking ability, and able to manage time efficiently when several technical issues were faced.

1.3 Structure

The study is included in four principal sections. The first part is theoretical and presents a basic overview of industrial collaborative robots and ABB RobotStudio software. The second part is a 3D printing technology preview and 3D modeling of parts in the application. The third one is about the integrated vision of the industrial robot, and image processing in the operation. The next one is to program the application from the early stages of the design process to laboratory tests and fixing unexpected issues. Finally, the study shows the necessary types of equipment need to connect to the IRB 14000 Yumi robot to run the application.

In the end, the section "summary and discussion" provides a conclusion for the study, considers what developments can be done, and reflects the application on factories.

2 Theorical study

2.1 Collaborative robots

Collaborative robots have low initial investment, do not require advanced knowledge to operate, and this type of robot can boost efficiency, manufacturing speed, quality by working along with humans (Figure 1). One of the main purposes of cobots is to make the workplace safer, cobots are suitable for automating repetitive work such as packaging and palletizing, assembly, material processing, screw and nut driving, and more.

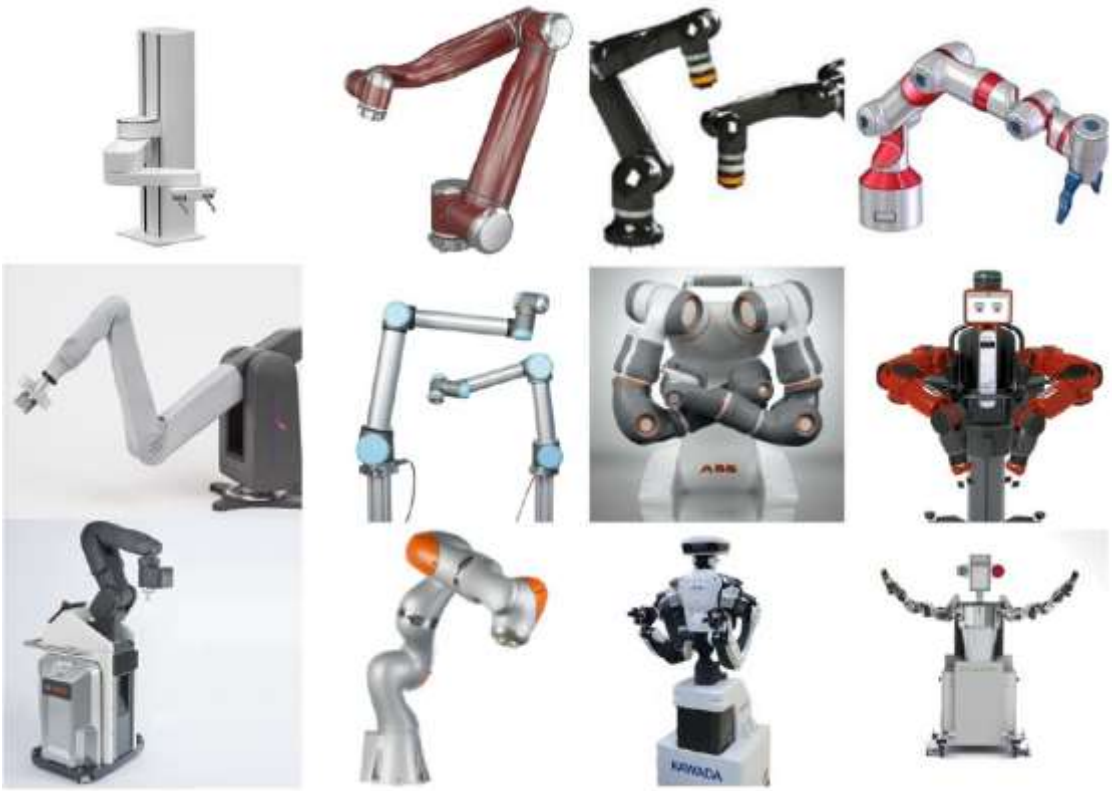


Figure 1 shows Collaborative robots **(Barrette, 2016)**

The purpose of industrial robots are designed for mass-production, extraordinarily precise, and high-speed production, so, it is typically huge and equipped with fixed tools. Although industrial robots have advantages, the speed of the robot's movement is fast, which can cause risks to human workers. To ensure that the working environment is safe, some safety measures are required such as cages, sensors, no-entry zones,....

Collaborative robots have an undoubted ability to boost a company's efficiency and productivity significantly. However, introducing collaborative robots into the manufacturing process, and making it fit into the factory is an important decision, and choosing the right cobot for the task is not simple. There are a few factors that need to be considered before choosing cobots:

- **Payload**: How many kilogram does the collaborative robot can carry during its operation. Without the weight of the end effector, the payload is calculated to get a given payload for collaborative.
- **Horizontal Reach**: How long does the collaborative robot's wrist reach. The distance is taken from the base of the robot.
- **Repeatability**: Repeatability is that the collaborative robot's end-effector reachs several positions for the same programed position, repeated several times under the same conditions.
- **Degrees of Freedom (DOF)**: "Degrees of freedom, in a mechanics context, are specific, defined modes in which a mechanical device or system can move. The number of degrees of freedom is equal to the total number of independent displacements or aspects of motion.The term is widely used to define the motion capabilities of robots." (Crowe, n.d.)

In the current market of collaborative robots, there are several providers and hundreds of available cobots for customers. However, in this report, Table 1 shows few specific famous cobots to compare such as Universal Robot, Omron, KUKA, Fanuc, Kawasaki, Kinova, and definitely ABB.

Table 1: Collaborative Robots Comparison (Crowe, n.d.)

Cobot	Payload (kg)	Horizontal Reach (mm)	Repeatability (mm)	DOF
ABB Dual-Arm Yumi	0.5	559	0.02	14
ABB Single-Arm Yumi	0.5	559	0.02	7
Universal Robots UR10e	10	1300	0.03	6

KUKA LBR iiwa 14 R820	14	820	0.1	7
Fanuc CR-35iA	35	1813	0.03	6
Omron Techman TM12	12	1300	0.1	6
Kawasaki duAro 2 Dual Arm	6	760	0.05	8
Kinova Gen2	4.4	984		7

When a company considers automating their manufacturing process, 2 types of robots which are brought to the table are traditional industrial robots and collaborative robots. To successfully make the right choice, the company needs to know the huge difference between those types, it was depicted in Table 2, each type is necessary for different situations, factories, industry..

Table 2: The difference between collaborative robots and traditional industrial robots (Cobot trends, n.d.)

Traditional robots	Cobots
Big batches, little variability Ideal for large companies that manufacture high volumes of the same products for long periods	**Low-volume, high-mix** Designed for low-volume, high-mix production, where the robot is often redeployed for new processes
Complex deployment Requires extensive programming skills and takes days or weeks to set up	**Fast and easy deployment** Easy to deploy with simple programming that inexperienced users can set up in minutes
Requires constancy	**Adapts to environment**

Programmed for unchanging environment and the same movement with minimal need to adapt	Flexible to adapt to changing environment and workpieces to be handled
Not safe without guarding Typically requires safety guarding to keep human workers out of the robot's work cell	**Collaborative and safe** After risk assessment, humans can work alongside robot in collaborative applications
Focus on the robot Repeats the same actions for years, with unchanging tool that is integrated for a specific process	**Focus on the EOAT** As robot arm becomes a commodity, focus shifts to EOAT to increase robot utilization
Big investment, longer ROI Expensive robots, system integration, and operator training requires larger upfront investment and takes longer for ROI	**Lower upfront cost, faster ROI** Competitive pricing, in-house integration, and ease-of-use minimize upfront costs and speed integration, uptime and ROI

2.2 ABB YuMi robot

ABB has come up with a dual-arm robot since 2015 to open tremendous worldwide automation potential in the industry with advancement concept. ABB IRB 14000 which has a common name YuMi is designed for tasks, which humans and robots can work side-by-side without cages, strict safety rules (Figure 2). The ABB YuMi brought to the market a new era of the industry, that concerns the most is safety. Production workers can forget about fencing and cages because the ABB YuMi will make collaboration easier and more productivity.

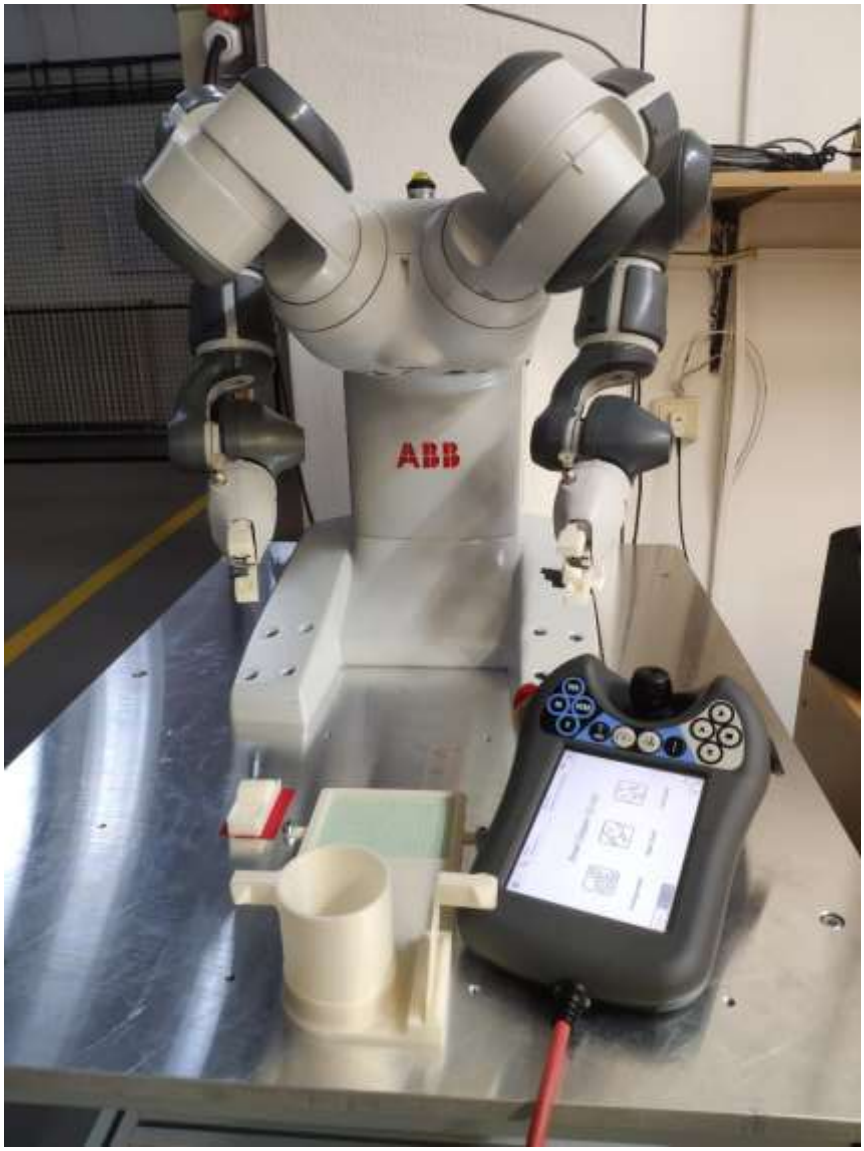


Figure 2 shows the ABB YuMi robot at Department of Robotics

The ABB YuMi is a collaborative, dual-arm robot which is equipped with two flexible hands, vision system and state-of-the-art robot control (Figure 3). The best function of the YuMi is its “inherently safe” design, which reduces the risk to an acceptable level and suitable for humans to work.

Robot version	Reach (m)	Payload (kg)	Armload
IRB 14000 - 0.5/0.5	0.559	0.5	No armloads
Number of axes	14		
Protection	Std: IP30 and Clean Room		
Mounting	Table		
Controller	Integrated		
Integrated signal and power supply	24V Ethernet or 4 Signals		
Integrated air supply	1 per Arm on tool Flange (4 Bar)		
Integrated ethernet	One 100/10 Base-TX ethernet port/per arm		

Figure 3 shows Specification of ABB Yumi **(ABB Group, n.d.)**

Benefits of Collaborative Robots and ABB YuMi robot:

- Adapt to changing markets: Thanks to the new design with inherent safety, the ABB Yumi has the ability to work alongside humans. Thus, collaboration leads to greater speed and better efficiency.
- ABB's comprehensive offering: The ABB YuMi has a low payload, it is suitable for applications such as pick&place small parts and inspection tasks. However, industrial robots with higher payloads and faster speeds can be transformed into a collaborative robot by the unique ABB SafeMove software.
- Simple start up: thanks to ABB's tools and software services, collaborative robots are easier to install than industrial robots, program and operation are suitable for every end-users.

2.3 ABB RobotStudio

RobotStudio which is an ABB's simulation and programming software, allows to program robots on a PC without interrupting the production system. To increase the profitability of the robot system, RobotStudio provides the tools to perform tasks such as designing stations, getting to use, programming, and improving without disturbing production. This provides numerous benefits including:

- Risk reduction
- Quicker start-up
- Shorter change-over
- Increased productivity (ABB Group, n.d.)

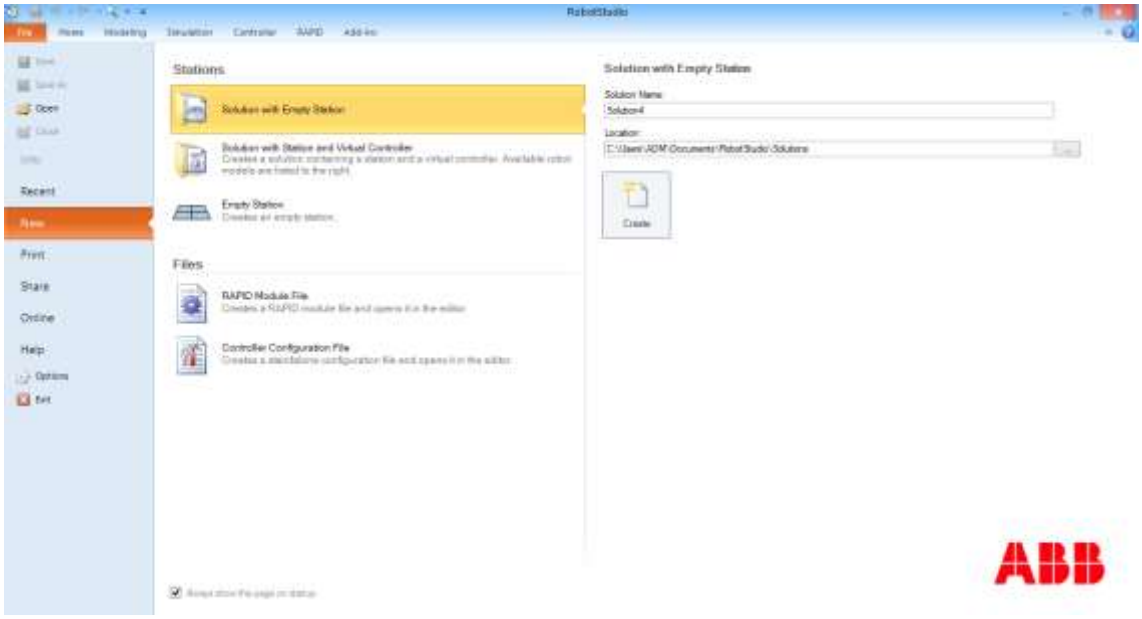


Figure 4 shows RobotStudio interface in Mircrosoft Window

RobotStudio can be runed in any platform, in this application, the RobotStudio was operated in Microsoft Window(Figure 4). This allows very realistic simulations to be performed, using real robot programs and configuration files identical to those used on the shop floor. (ABB Group, n.d.)

3 3D printing

3D printing which is a centurial invention brings benefits to many businesses. A design can be evaluated quicker than before, in some practical terms, 3D prototypes can be tested the usability, feedback is obtained easily to make changes base on the target market. Accordingly, products are brought to market more rapidly, and the company does not spend thousands of dollar on testing, analyzing a prototype which could be a failure.

In this application, 3D printed modelings play a hugely important role in testing and the application development process. Several prototypes were made to improve the process efficiently, stable.

3.1 3D printer

The 3D printer that was used in this application is the Original Prusa i3 mk3s (Figure 5). According to all3dp.com and Make: magazine, the Original Prusa i3 mk3s is the best 3D printer and the editor's choice award in the 2019 digital fabrication guide.

“The mk3s features a rebuilt extruder, numerous sensors, and various smart features. plus, a new magnetic mk52 heatbed with replaceable pei spring steel print sheet. dozens of various tweaks and upgrades improve the reliability and ease of use. Mk3s‘s functionality can be further expanded with the unique multi-material upgrade 2.0 addon for printing with up to 5 materials simultaneously.” (Prusa Research, n.d.)

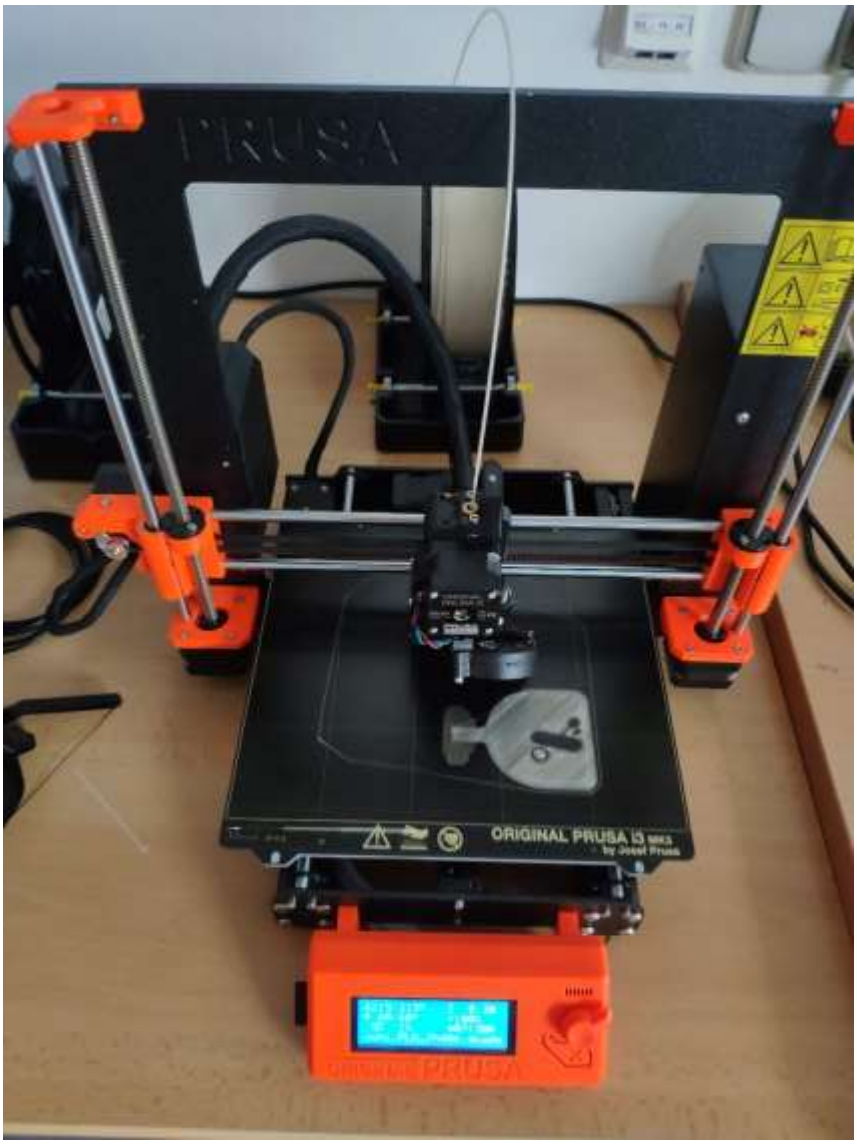


Figure 5 shows Original Prusa i3 MK3 at Department of Robotics

3.2 Gripper and gripper fingers

3.2.1 Gripper fingers

“Human-level manipulation of various objects is a difficult technical challenge, and robotics developers and vendors have responded with a range of solutions. From claw, parallel, and rotary grippers to bellows, magnetic, and vacuum grippers, robotic manipulation has evolved to meet the needs of industries including automotive and electronics manufacturing and food processing. With the rise of collaborative robot arms, or cobots, machine manipulation has diversified to handle a widening variety of objects. Industrial robots could use

parallel or rotary grippers for fast, repetitive handling of identical parts, but cobot users need flexibility and safety over throughput.” (Demaitre, 2019)

Base on the dimension of the getting-started finger (Figure 6), a new finger was uniquely designed for this application.

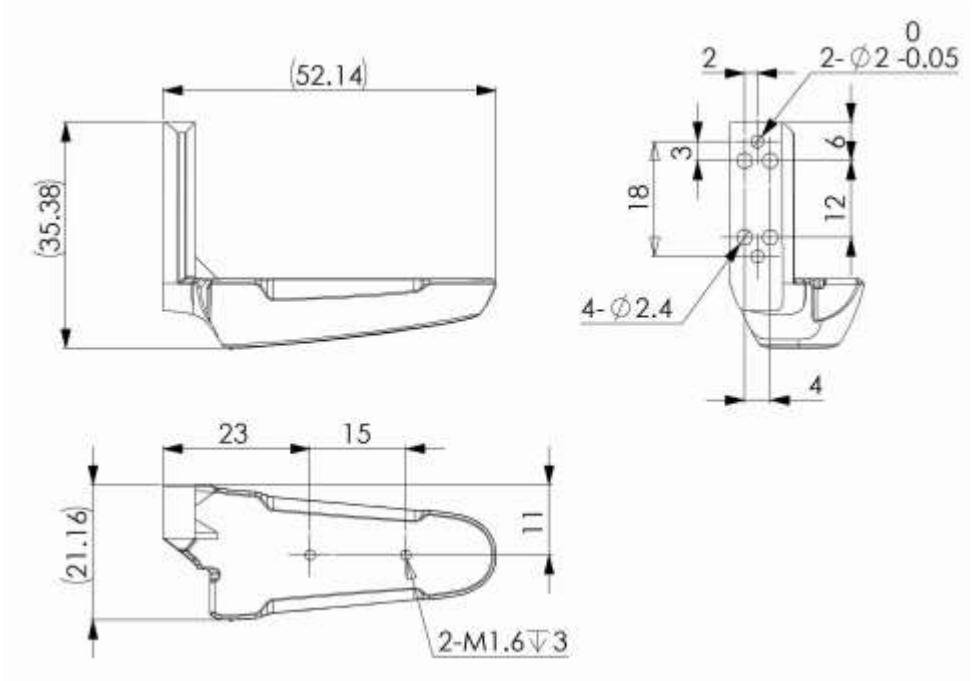


Figure 6 shows the dimension of the getting-started finger **(ABB AB, Robotics, 2018, p. 33)**

To perform a pick-place process, the grippers create the force, which have to be strong enough to compensate the gravitational force of objects in motion. Therefore, the grippers have to be mated with objects to ensure that the objects do not either fall off or get damage from the grippers. The 2-finger parallel gripper was chose because it is the most flexible design and is able to carry out a most percentage of applications (Figure 7).

”The reason why the two-jaw parallel design is the most commonly used is because it can handle so many part shapes and sizes. It can also do the same job as the other two basic gripper types, the three-jaw gripper and the two-jaw angular” (Richards, 2011)

Figure 7 shows The designed finger gripper was inserted on the right gripper

3.2.2 Gripper on ABB Yumi

The ABB YuMi is offered gripper options (Figure 8). The basic function of the options is to grasp parts using a parallel grip, there is some upgrade like suction cups, an embedded vision system. The user will decide their needs, budget to select the best option for the ABB YuMi.

Figure 8 shows ABB Grippers **(Bélanger Barrette, 2015)**

There are 5 options for a gripper to ensure that all application can be handle by ABB Yumi (Figure 9).

Combination	Weight (g) without fingers, suction cup(s), and filter(s)[i]	Weight (g) of the whole gripper	Max. load capacity (g) without fingers, suction cup(s), and filter(s)[ii]	Max. load capacity (g) of the whole gripper[ii]
Servo	215	230	285	270
Servo + Vacuum 1	225.5	248	274.5	252
Servo + Vacuum 1 + Vacuum 2	250	280	250	220
Servo + Vision	229	244	271	256
Servo + Vision + Vacuum 1	239.5	262	260.5	238

i The getting-started fingers weights 15 g, and the standard suction cups and filters weight 7.5 g per set.

ii Load capacity = 500 - Weight
Center of gravity (CoG) limitations applied. See the robot load diagram.

Figure 9 shows Weight and load capacity of 5 options gripper **(ABB AB, Robotics, 2018, p. 21)**

Figure 10 shows The right gripper

The right hand has 1 servo and 2 vacuums, the servo has speed $20m/s$, force $12N$, 2 pneumatic vacuums has gas pressure $98kPa$ (Figure 10)

Figure 11 shows The left gripper

The left hand has 1 servo, 1 vacuum, and 1 vision, the servo has speed $20m/s$, force $12N$, the pneumatic vacuum has gas pressure $98kPa$ (Figure 11)

3.3 Dice shakers and dices

The most important part in this application is the holder and the cover, without the dice shakers, the game could not start, the dices could not be shaken, and could be placed at a wrong position.

3.3.1 Dices

The dices are not manufactured, it is the only thing which was brought at a store. The dimension of the dices is 15x15x15, the weight is 100 gram (Figure 12).

Figure 12 shows Dices in the application

3.3.2 Dice shaker

The purpose of a dice holder is to shake the dices inside, so it has to be spacious enough for the dices to shake. With the support of the cover, when the dices were shaken, the cover's aim is to avoid the dices bounce off of the holder, and when the dices are thrown out into a playground, the cover will hold the dices to stop the gravitational force and slowly move away for the dices gently fall off into the playground (Figure 13).

The mating between the gripper and the holder, the cover has to be fit to avoid either collision or falling off.

Figure 13 shows Grippers holding objects

3.4 Workplace

The workplace was designed to meet the requirements that have been pointed out at brainstorming and have been appeared during the process.

The requirement at the brainstorming: the workplace is the place where the dice shaker stores at, and the spot where the dices are thrown out to do the image process. To fulfill the need, a prototype of the workplace was printed.

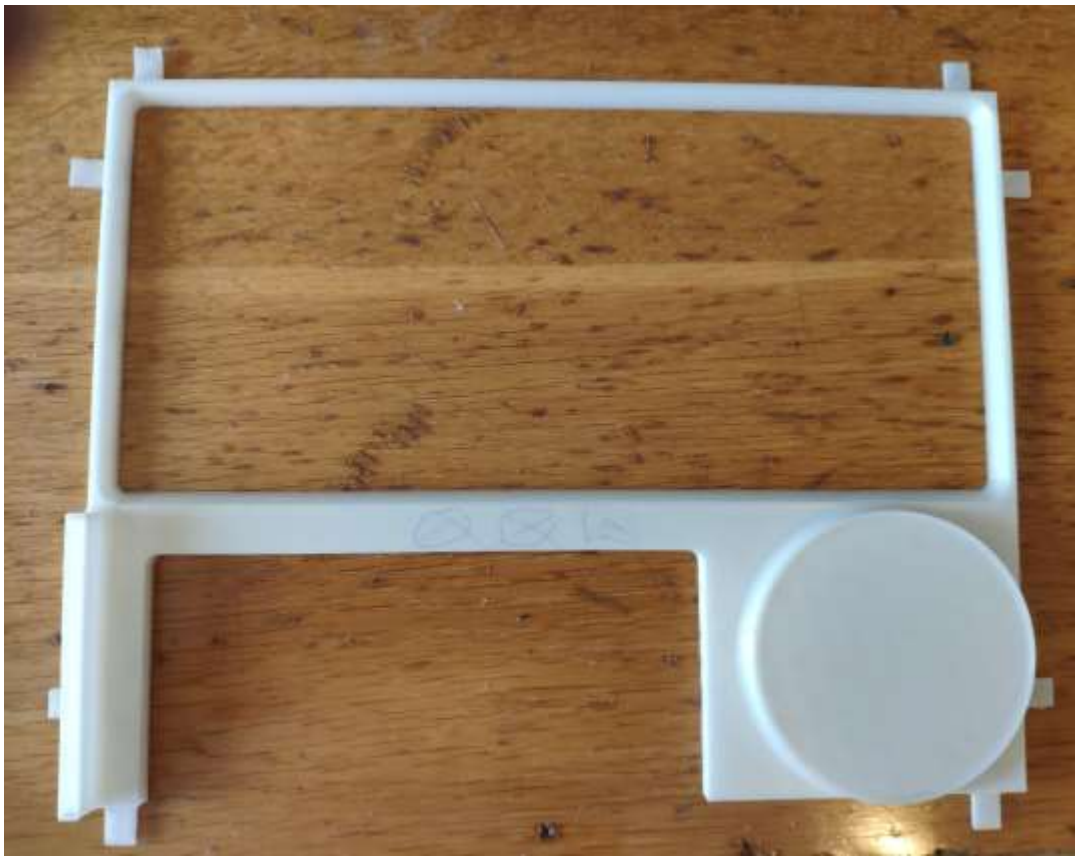


Figure 14 shows Old version prototype of the workplace

This prototype has been done a good job to store the holder and the cover, it was spacious enough for the dices to stay inside (Figure 14).

However, this prototype showed some issues during the testing. First of all, 8 square pieces around the workplace were attached black adhesive tape to keep the playground stable, but it was not the best option, not efficient, and the spot can be moved away easily by accidents or an wrong action of the robot. Secondly, the area for image processing is $200x80\ mm^2$, too spacious, it caused problems for the built-in camera on the left hand, the camera lens focuses in a large area, meanwhile, the dices are in small-scale, the result was the camera could not recognize the dices.

The solution for that issue came up by recognizing that the table, which Yumi robot is on has some screw holes M6, also there is a small object which is unremovable. Therefore, a new design was made, it not only met the requirements but also downsizing the image processing area into $80 * 80\ mm^2$ (Figure 15).

Figure 15 shows The workplace and an unremovable object

The built-in lights provides basic front lightning for standard applications. Allows user to start using the camera without a separate lightning system. (ABB, 2015)

***Figure 16 shows the built-in light of the lens* (ABB, 2015, p. 4)**

However, the built-in lights (Figure 16) caused a problem, all images that took from the camera lens turn out to be too bright and the YuMi robot was unable to detect the dices. The problem caused by the aluminum table surface reflected the light, it made the brightness of images was too high. Accordingly, a piece of green paper was inserted under the workplace, images were excellent (Figure 17).

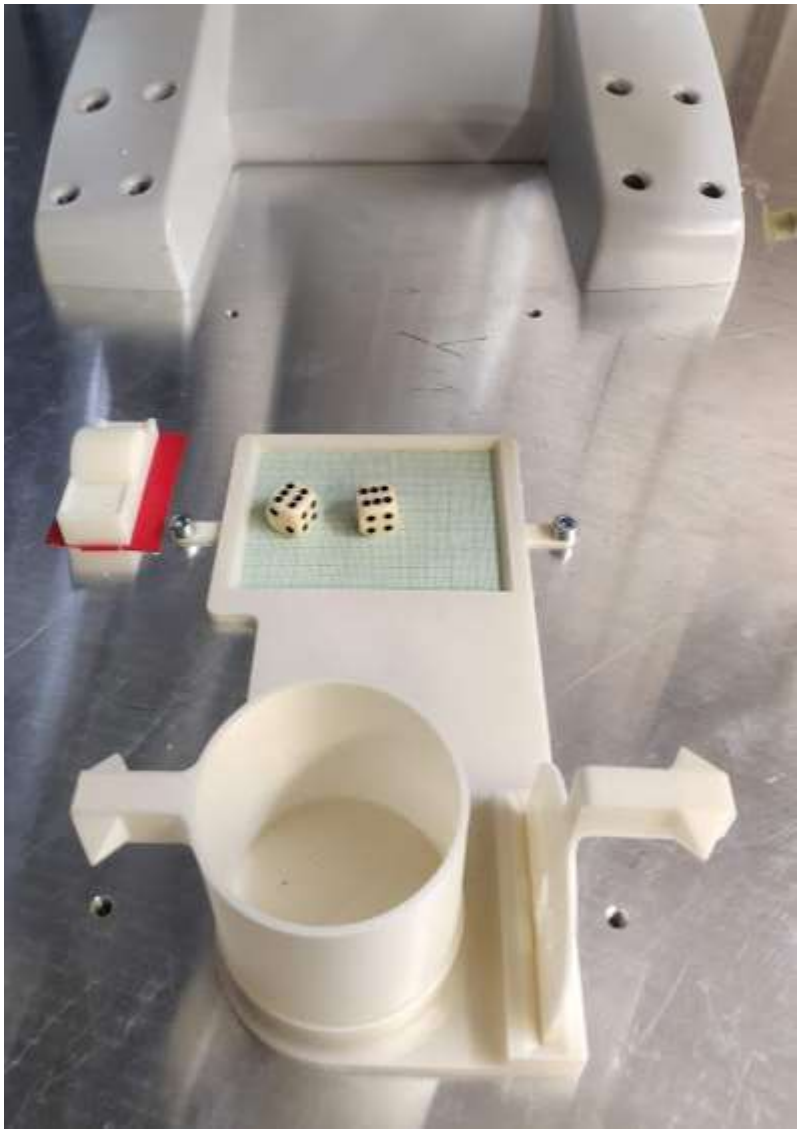


Figure 17 shows The playground includes holder, cover, dices, workplace was set

4 Cognex camera and Intergrated Vison on ABB YuMi

4.1 Overview

To make robots more flexible in unique situations, vision systems have been attached to the robots for required vision tasks such as inspection, identification. Cognex and ABB decided to make the future come closer by providing a Cognex AE3 camera into the ABB YuMi robot's vision module, which makes the excellent cobot has a powerful and reliable vision for image processing (Figure 18).

With the help of the Cognex camera, ABB's Integrated Vision system has the ability to present a strong visual system for VGR applications. The IRC5 robot controller and the RobotStudio program integrates a system that provides a fully functional complete software and hardware solution for any vision tasks. The Cognex In-Sight smart camera family brings the vision capability the power to embedded image processing and an Ethernet communication interface.

Figure 18 shows Cognex AE3 **(ABB Group, n.d.)**

***Figure 19 shows ABB YuMi camera lens* (ABB, 2015)**

Even though the camera in the ABB Yumi is an add-on camera (Figure 19), the specification shows in the Table 3 proves the Cognex AE3 is a strong, great camera.

Table 3 shows Cognex AE3 camera specification (ABB AB, Robotics, 2018)

Description	Data
Resolution	1.3 Megapixel
Lens	6.2mm f/5
Illumination	Integrated LED with programmable intensity
Software engine	Powered by Cognex In-Sight
Application programming software	ABB Integrated vision or Cognex In-Sight Explorer

4.2 Connect to camera and identify the dices

4.2.1 Connect to the camera and set-up image

To access the Integrated Vision tab, from the Controller tab right-click on the Integrated Vision (Figure 20).

Figure 20 shows Integrated Vision on the Controller tab

From the vision tab, the available cameras should appear under Vision System in the left-hand controller tree, because kamera_nova is the name of the camera lens on this ABB YuMi, right-click on Connect ”kamera_nova” (Figure 21).

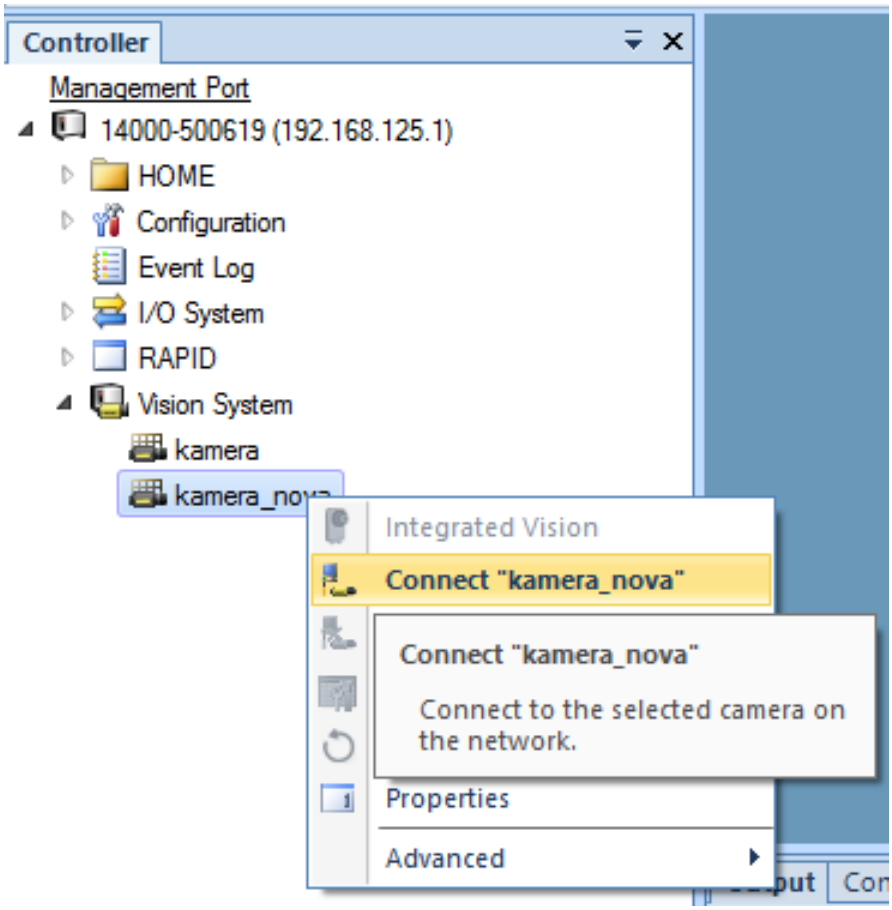


Figure 21 shows Connect "kamera_nova"

Once connected to the kamera_nova, the left arm of the YuMi had to travel to an exact position to acquire images, this position plays a hugely important role, it must stay stable and the left arm always has to be in the position precisely. Otherwise, there are 2 consequences if the left arm could not be in the right position. The first result is that the robot can not recognize the dices, it will run a loop, which never ends, the second one is that even the dices are realized, the coordinate of the dices is not precise, it causes the right arm could not pick the dices, there is a possibility that it can cause a loop.

Figure 22 shows the taskbar in the Vision tab

After an image was acquired, the image had to be adjusted to be saw and processed, select Setup Image to modify. The exposure needs to be adjusted,

mostly the default value is 8 msec, however, it made the image too bright, therefore, at 2 msec, the resolution is sharpness and avoid noise (Figure 23).

Figure 23 shows Setup image tab

When the image was good and ready to recognize the dices, however, the coordinate of the dices was in pixels, accordingly, to calibrate the images from pixels to millimeters, select Calibrate. There are several types to calibrate an image, but in this case, type ”Edge to Edge” was chosen (Figure 24).

Figure 24 shows Calibrate an acquired image

4.2.2 Identify the dices

To identify the dices, the YuMi has to recognize the sides of dice and locate the coordinate, however, some sides are quite familiar such as the side number 2-4, 3-5, 4-5, 2-3, 1-5,1-3, because they have the same diagonal lines, horizontal lines, vertical lines with 2-3 dots on a line (Figure 25).

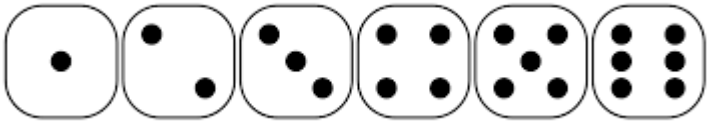


Figure 25 shows 6 sides of a dice

Each side of a dice was identified by a camera task/a job, describing exposure parameters, calibration, and what vision tools to apply (ABB AB, Robotics, 2018), for 6 sides of a dice, 6 jobs were conducted, each job has 5 steps.

First of all, in the Add Part Location Tool menu, select Edge Intersection to define the x-axis and y-axis, and the intersecting point between 2 edges is the origin of coordinates (Figure 26). Edge_1 is the x-axis, Edge_2 is the y-axis, and Intersect_1 is the origin of coordinates (Figure 27).

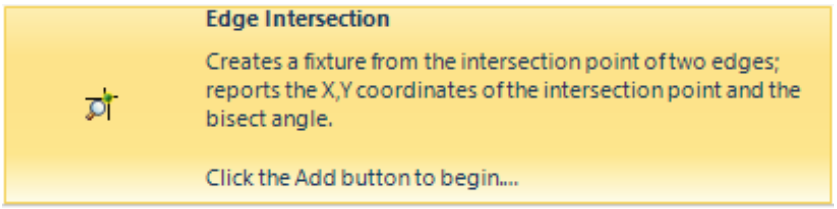


Figure 26 shows Edge Intersection tool

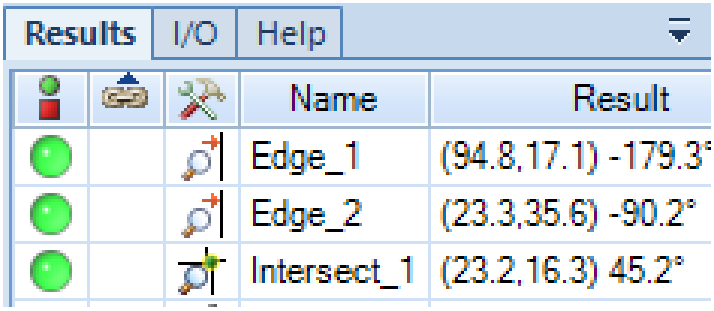


Figure 27 shows Edge tools result

Secondly, plenty of tools are available in the Add Part Location Tool menu to locate target objects, in this case, the Pattern tool was chosen (Figure 28). The result reported back part location (X, Y coordinate), orientation, and image score, the more score one image got, the precise it is.

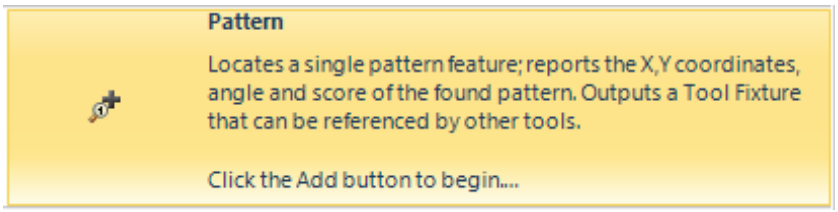


Figure 28 shows Pattern tool

In the Pattern setting, the value of Rotation Tolerance needs to change to 180 degrees, the purpose of this value to guarantee that even the side of the dice

rotates from 0° to 180°, it can be recognized. In addition, the Accuracy section changed to accurate, even though it took more time to process, it only costs less than 0.05 section (Figure 29).

Figure 29 shows Pattern setting

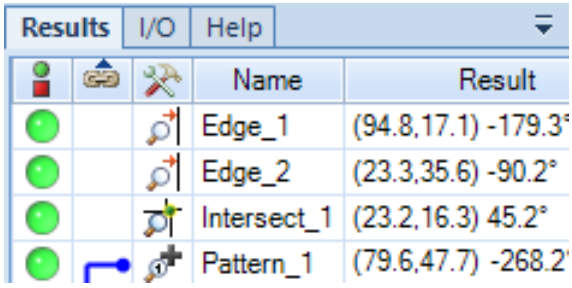


Figure 30 shows Pattern reports back the coordinate of a dice

The third step to avoid any mistake and collision causes by picking up a wrong dices, another tool in the Add Part Inspection Tool menu, which is Blobs was added (Figure 31).

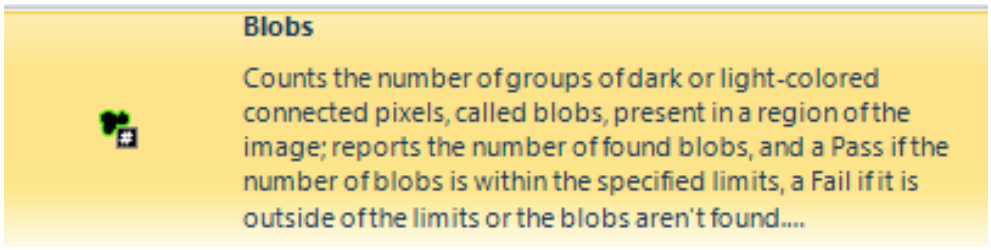


Figure 31 shows Blobs tool

The most important thing in the Blobs tool setting is the tool fixture, it needs to be stuck to Pattern_1, the reason to choose Pattern_1.fixture is that the Blobs tool will search in the Pattern_1 area (Figure 33).

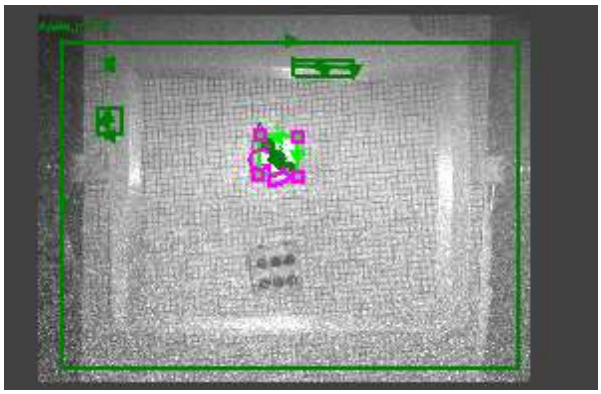


Figure 32 shows the blobs red square

Figure 33 shows Gerenal tab in Blobs tool

The color of 3 dots on a side of the 3 number dice are black, therefore, the Blobs Color changed to Black (Figure 34).

Figure 34 shows Setting tab in Blobs tool

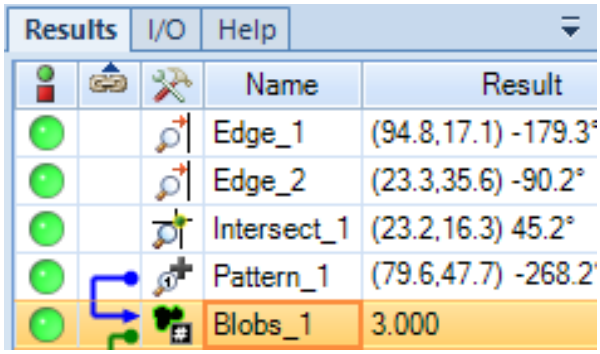


Figure 35 shows Blobs tool result

The fourth stage is to use Logic tool (Figure 36).

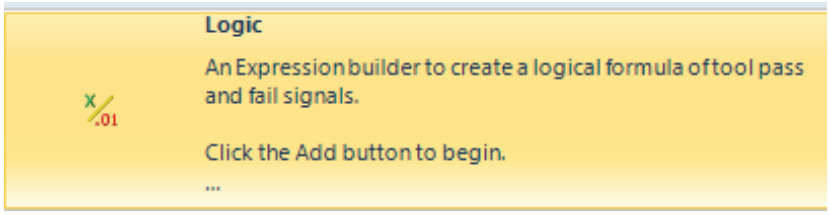


Figure 36 shows Logic tool

The result of the Logic tool express that the process recognize a dice and the number on the dice, so the expression was included Pattern_1.Pass and Blobs_1.Pass (Figure 37).

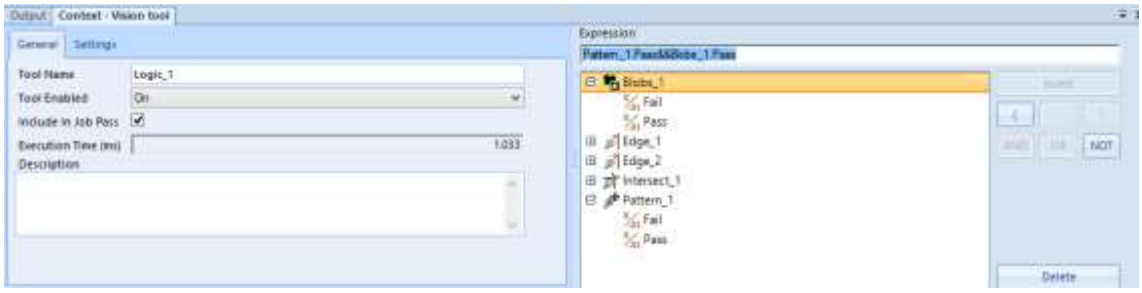


Figure 37 shows Logic tool setting

The result returned True, because the camera detected Pattern_1 and Blobs_1 (Figure 38).

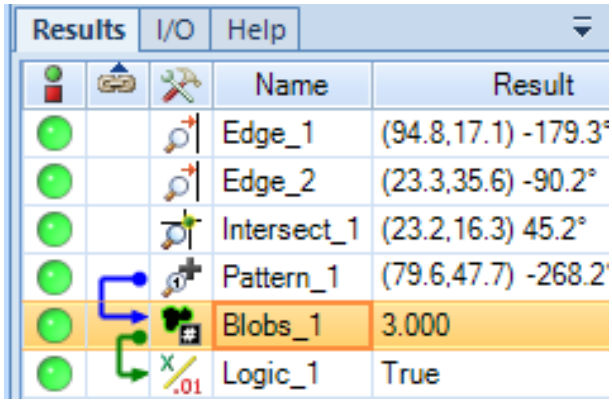


Figure 38 shows Logic tool result

The next step is to transfer the result of the image process to the rapid code, select Output to Rapid, Context-Setup Communication tab was showed up. In this case, the side number 3 was recognized, so the Part Names was changed to Part_3, the Pattern_1 group returned 2 values of x-axis and y-axis based on

the image coordinate, the Logic_1 group reported back the value of the Logic tool, the value ”True” is considered as 1, the reverse value ”False” is considered as 0 (Figure 39).

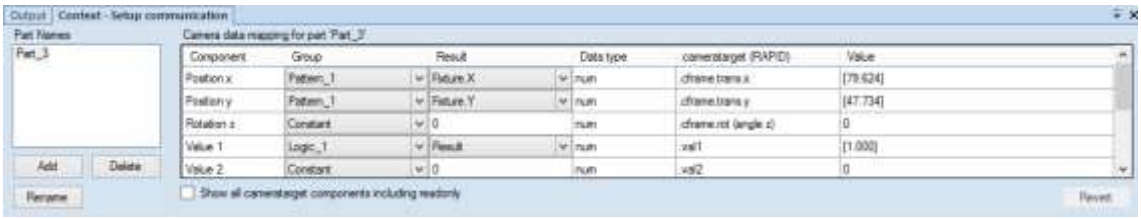


Figure 39 shows Output to Rapid result

Finally, the job needs to be saved to the Yumi, otherwise, the image is not processed in the rapid code. The job is stored in the folder kamera_nova, which is in the memory of the YuMi robot (Figure 40).

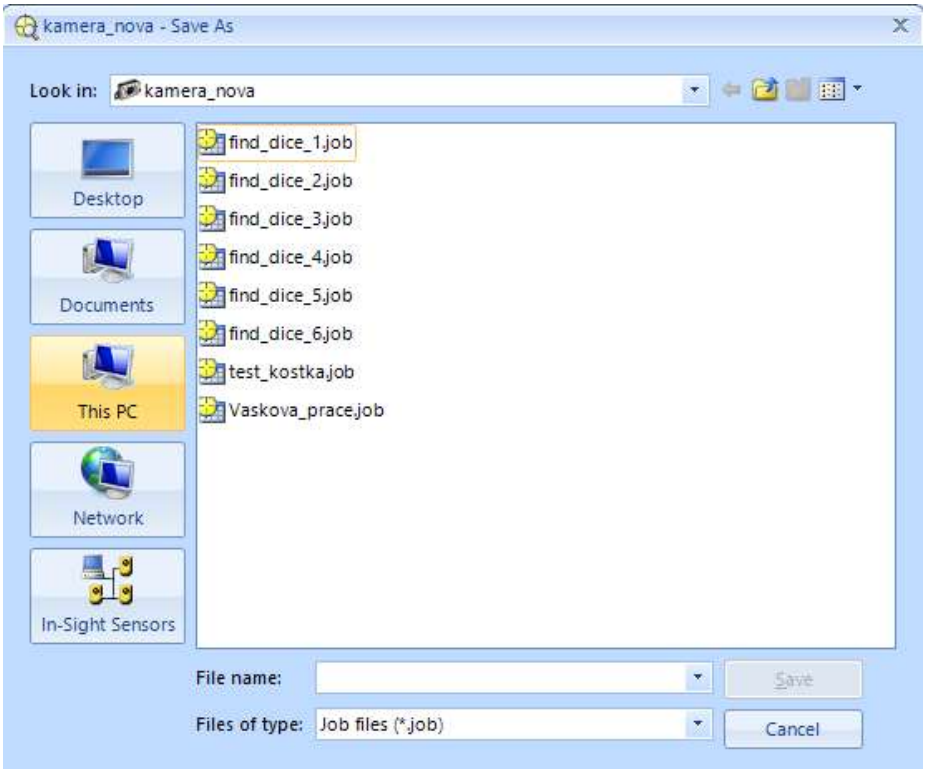


Figure 40 shows 6 jobs were saved in the kamera_nova folder

4.2.3 Calculating the dices coordinate

The coordinate which was returned back to the rapid code from a job result is not the exact coordinate to pick up.

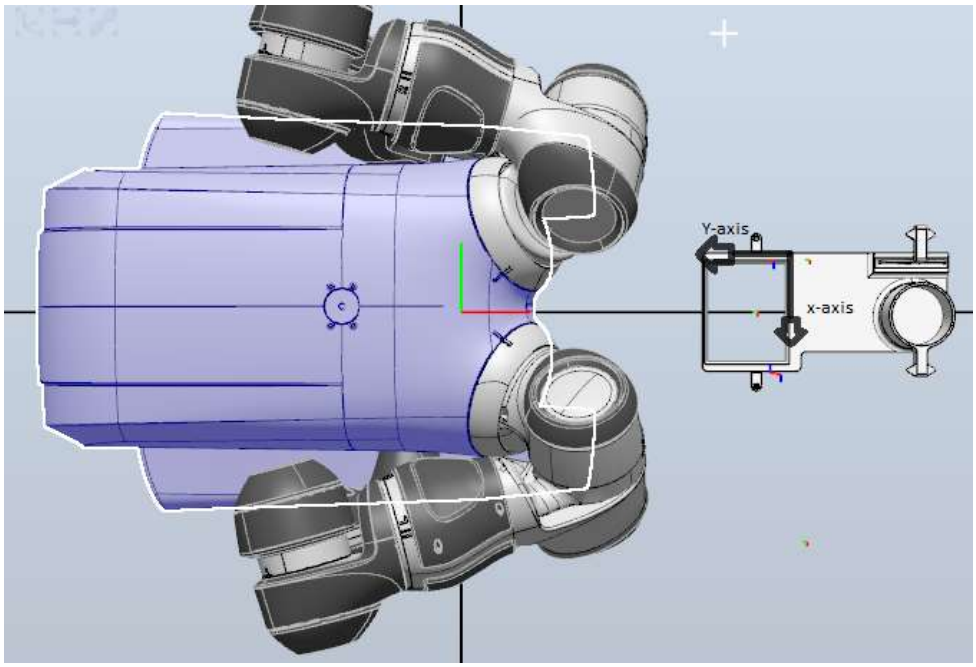


Figure 41 shows The world coordinate and the coordinate of the image

The origin world coordinate of the robot is the intersected point between 2 lines, the red line is the x-axis and the green line is the y-axis. On the other hand, the coordinate of the image was exposed to figure 41.

The coordinate of a dice, which was processed in a job needs to be calculated to transfer to the exact coordinate that the right arm can pick up. pTarget_dice_6.trans.x:=366-camera_target.cframe.trans.y;
pTarget_dice_6.trans.y:=84-camera_target.cframe.trans.x;

In these rapid code, the side number 6 was processed:

- pTarget_dice_6.trans.x is the x-axis value of the exact coordinate
- pTarget_dice_6.trans.y is the y-axis value of the exact coordinate
- camera_target.cframe.trans.y is the y-axis of the image processing coordinate
- camera_target.cframe.trans.x is the x-axis of the image processing coordinate

5 Programming on ABB RobotStudio

Based on the advantages of ABB RobotStudio: reduce the risks, improve safety, productivity-increasing, and quick error detection, the project was recommended to run, program, test on RobotStudio software.

The 3D modelings from PTC creo parametric 3d modeling software can be transferred to the modeling in RobotStudio software by changing the format of the 3d file from .prt to .sat.

The Yumi was taken in the ABB library/Collaborative Robots section (Figure 42).

Figure 42 shows ABB library menu

2 Grippers were gotten from the Import Library menu/Equipment folder/Tools section (Figure 43).

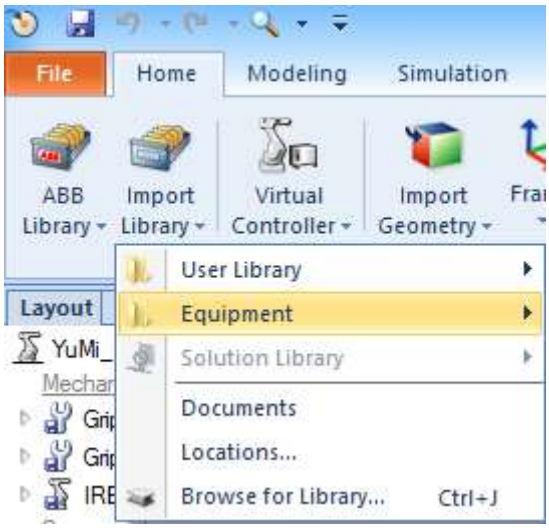


Figure 43 shows Import Library menu

The left and right gripper have different features and functions, so the ABB Smart Gripper does not the same at both arms. The feature can be chosen in a window which appears after clicking ABB Smart Gripper, the right gripper has a servo, two vacuum cups, and the left gripper has a servo, one vacuum cups, one camera (Figure 44).

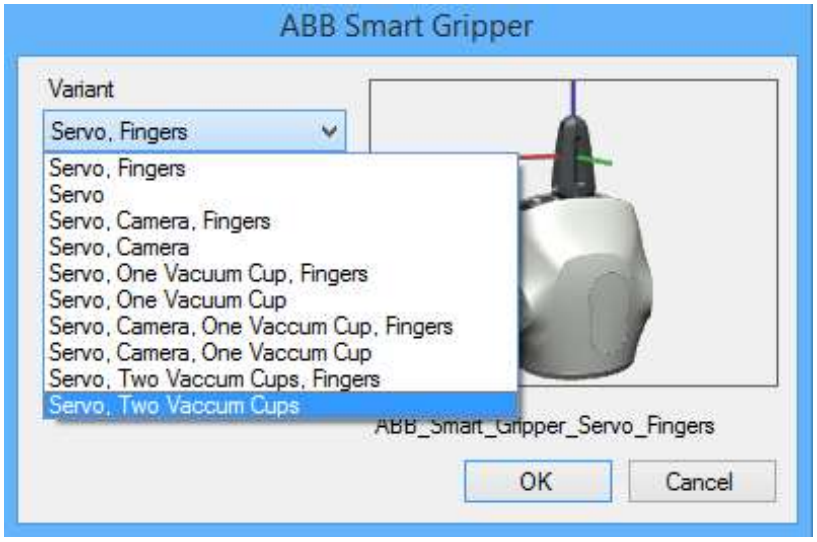


Figure 44 shows ABB Smart Gripper selection

Calculation of the probability of dice is impossible in ABB RobotStudio, this is the disadvantage of the software, it causes the simulation to stop at shaking task. The dices can be modeled by PTC creo parametric 3d modeling software, however, the dices could not randomly change. Furthermore, the virtual camera on the left arm could not process in the simulation.

Therefore, the simulation is included the IRB 14000 Yumi, 2 grippers, 1 cover, 1 holder, 1 workplace, 4 fingers for grippers, which is exposed in Figure 46.

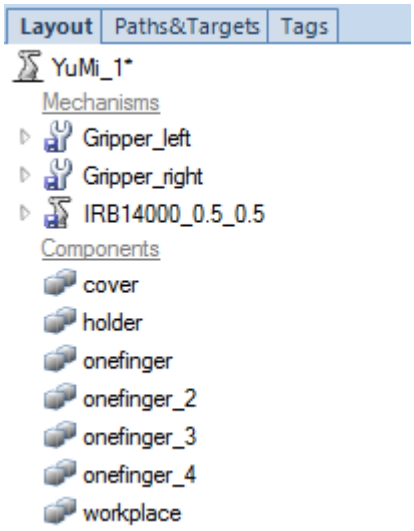


Figure 45 shows Layout of the simulation

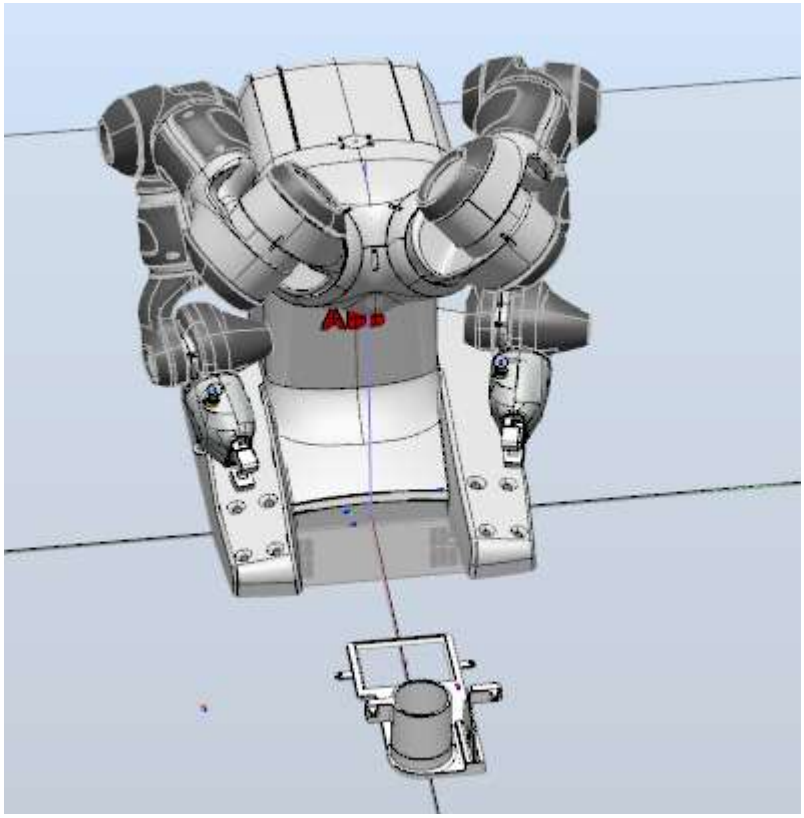


Figure 46 shows the project in RobotStudio

5.1 Multimove system

Operators require a high level of accurate synchronization so robots work as a team and complete tasks that a single robot cannot perform. Two robots or 2-hand robots could lift an object that is too heavy or complex for one, or they could work simultaneously on an object as it moves or rotates. These tasks require a very high degree of synchronization, which is now possible thanks to ABB's new IRC5 controller. To synchronize is to make sure that the RAPID program in the virtual controller corresponds to the program in the station, synchronization can be made from the station to the virtual controller and vice versa.

The Yumi integrated controller is based on the standard IRC5 controller, it makes the Yumi fulfills the shaking task at the same time and same speed.

5.2 Movement path for the left arm

The left arm of the Yumi starts from its home position to the pick-up cover spot (Figure 47).

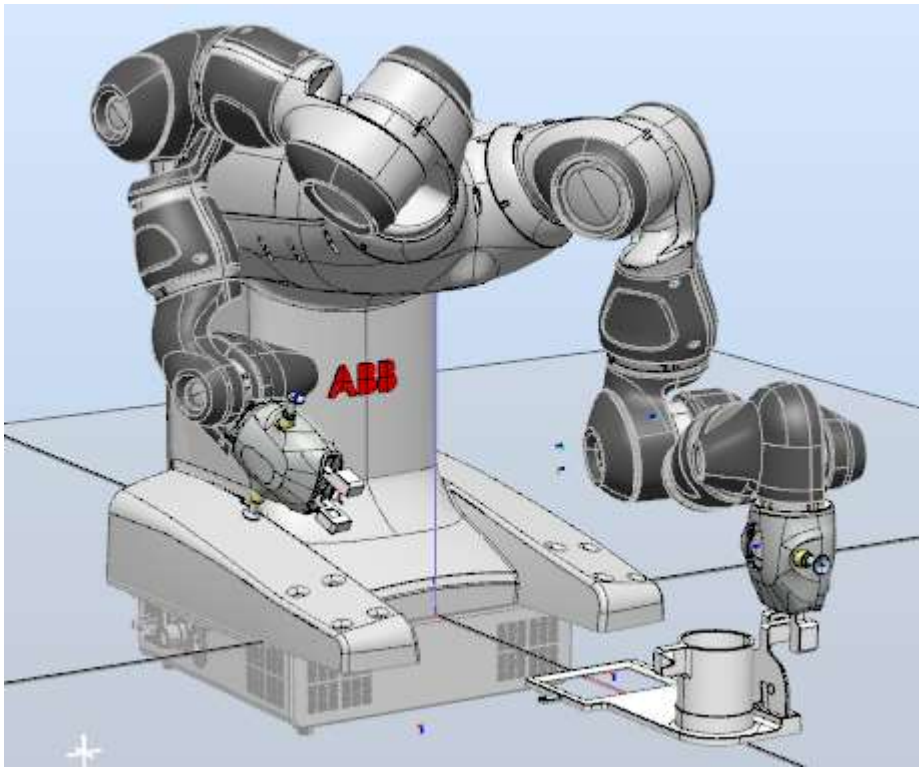


Figure 47 shows the left arm picks up the cover

After the pick-up task was done, the left arm with the cover travels to the shaking position (Figure 48), then both arms will do shake and go down to throw the dices out.

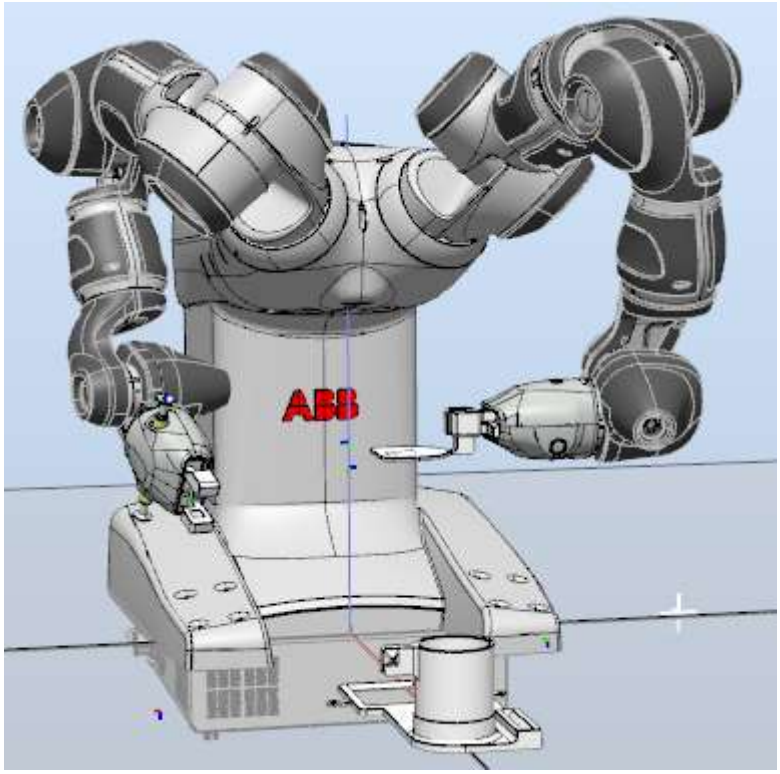


Figure 48 shows the left arm at the shaking position

Accordingly, the left arm will go back to the pick-up position to store the cover. Then, it will go to the acquired photo spot to run the image processing task (Figure 49). Finally, it goes back to the home position.

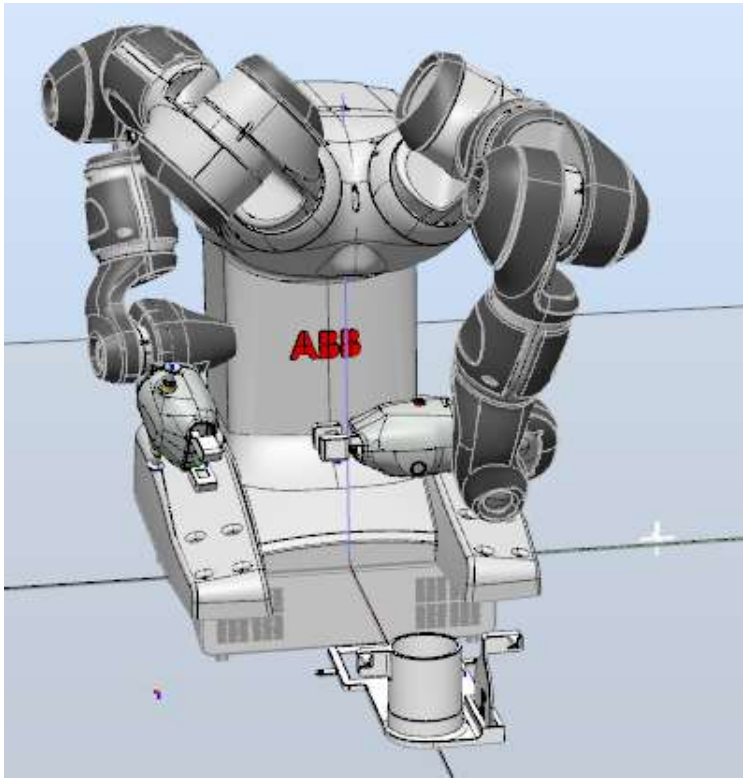


Figure 49 shows the left arm taking pictures

5.3 Movement path of the right arm

The right arm of the Yumi starts from its home position to the pick-up holder spot (Figure 50).

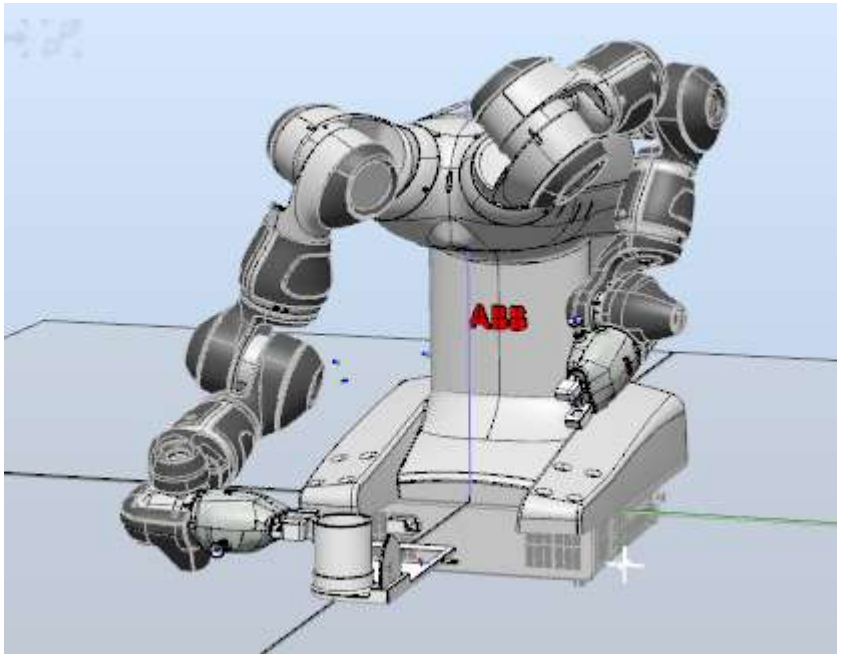


Figure 50 shows the right arm picks up the holder

After the pick-up task was done, the right arm with the holder travels to the shaking position (Figure 51), then both arms will do shake and go down to throw the dices out.

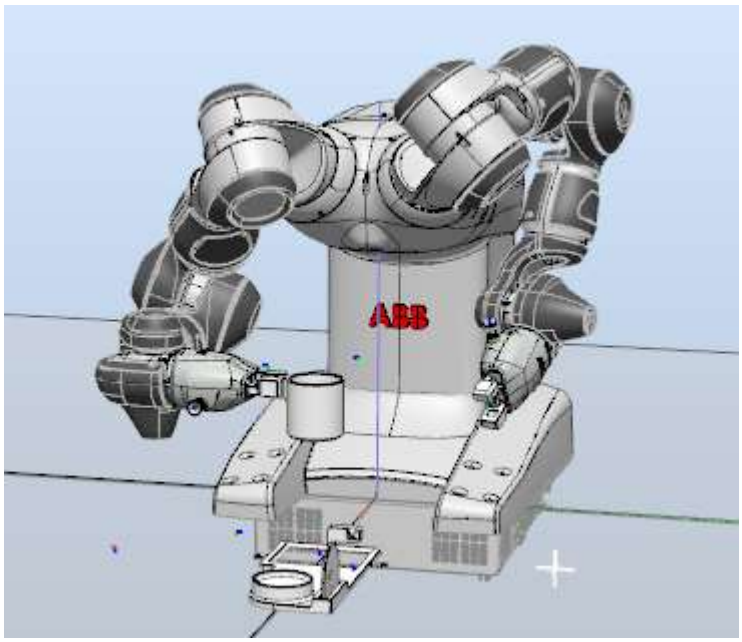


Figure 51 shows the right arm with the holder at the shaking position

Accordingly, the right arm will go back to the pick-up position to store the cover. Then, it will go back in the z-axis direction to wait for the processing task of the left arm finish to pick up the dices (Figure 52).

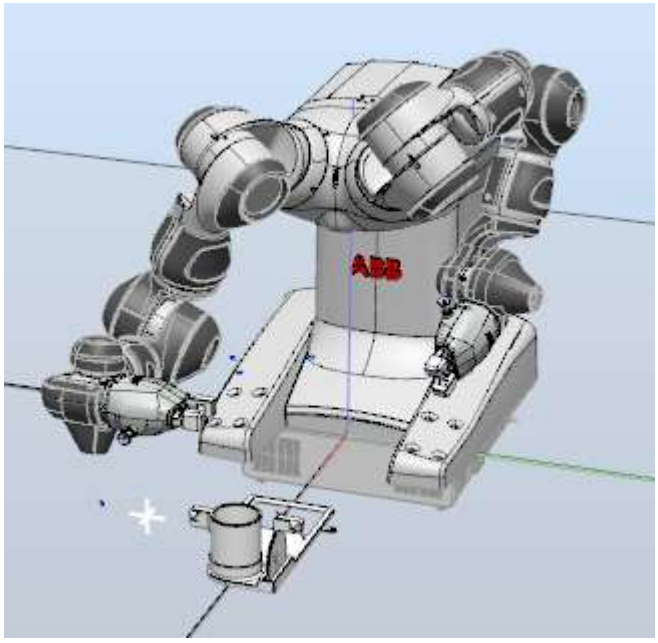


Figure 52 shows the right arm waiting for picking-up dices

5.4 Essential code lines

MoveJ is used to move the robot quickly from one point to another when that movement does not have to be in a straight line. The robot and external axes move to the destination position along a non-linear path. All axes reach the destination position at the same time. (ABB AB, Robotics, 2018)

MoveL is used to move the tool center point (TCP) linearly to a given destination. When the TCP is to remain stationary then this instruction can also be used to reorientate the tool. (ABB AB, Robotics, 2018)

WaitRob (Wait Robot) waits until the robot and external axes have reached stop point or have zero speed. (ABB AB, Robotics, 2018)

WaitTime is used to wait a given amount of time. This instruction can also be used to wait until the robot and external axes have come to a standstill. (ABB AB, Robotics, 2018)

WaitSyncTask is used to synchronize several program tasks at a special point in each program. Each program task waits until all program tasks have reach the named synchronization point. (ABB AB, Robotics, 2018)

SyncMoveOn is used to start a sequence of synchronized movements and in most cases, coordinated movements. First, all involved program tasks will wait to synchronize in a stop point and then the motion planner for the involved program tasks is set to synchronized mode. (ABB AB, Robotics, 2018)

SyncMoveOff is used to end a sequence of synchronizedmovements and, in most cases, coordinated movements. First, all involved program tasks will wait to synchronize in a stop point, and then themotion planners for the involved program tasks are set to independent mode.

The instruction ***SyncMoveOff*** and ***SyncMoveOn*** can only be used in a MultiMove system with option Coordinated Robots and only in program tasks defined as Motion Task. (ABB AB, Robotics, 2018)

Offs is used to add an offset in the object coordinate system to a robot position. (ABB AB, Robotics, 2018)

RelTool (Relative Tool) is used to add a displacement and/or a rotation, expressed in the active tool coordinate system, to a robot position. (ABB AB, Robotics, 2018)

6 Connect to the IRB 14000 ABB Yumi robot

The IRB 14000 integrated controller is based on the standard IRC5 controller, and contains all functions needed to move and control the robot. (ABB, 2020, p. 79)

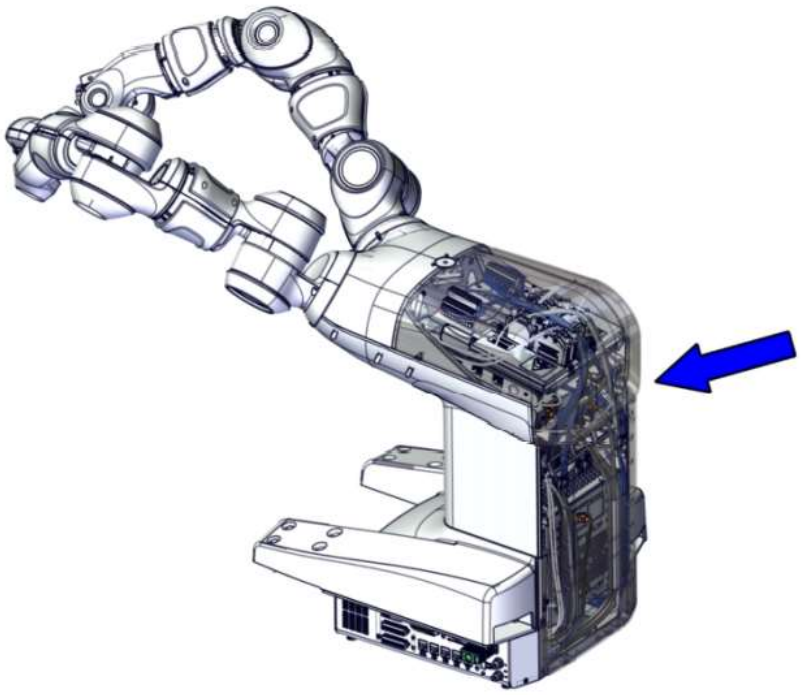


***Figure 53 shows IRB 14000 controller* (ABB, 2020, p. 79)**

The pick-up dice task requires the hydraulic vacuum on the right arm, the air supplier of the vacuum was provided by an air compressor, which is connected on the left side panel of the controller (Figure 54), and the Table 4 is the explaination for all symbols in the left side panel.

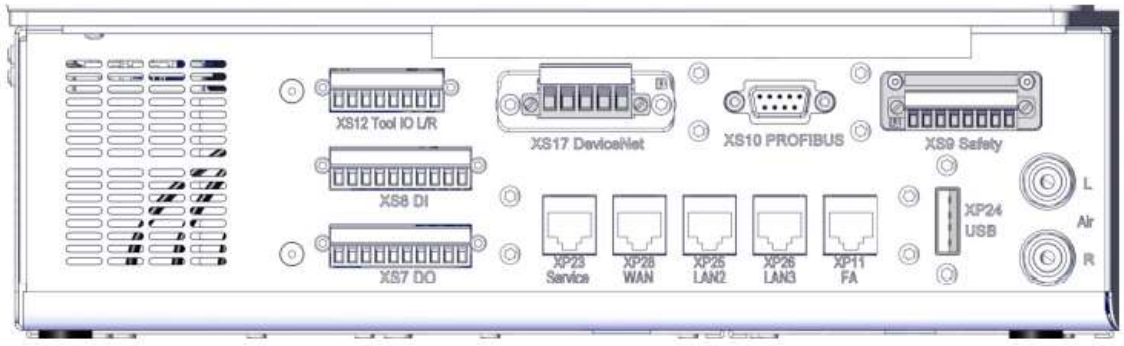


***Figure 54 shows The interface on the left side panel of the controller* (ABB, 2020, p. 80)**

Table 4: The clarification of symbol of the left panel (ABB, 2020)

XS12	Tool I/O, left and right arm 4x4 digital I/O signals to the tool flanges, to be cross connected with XS8 and/or XS9. This is alternative to Ethernet on the tool flange.
XS17	DeviceNet Master/Slave
XS10	Fieldbus adapter PROFIBUS Anybus device (fieldbus adapter option)
XS9	Safety signals
XS8	Digital inputs 8 digital input signals (approx. 5 mA) to the internal I/O board (DSQC 652) Pin number 9 (24 V = max current 3A)
XS7	Digital outputs 8 digital output signals (150 mA/channel) from the internal I/O board (DSQC 652) Pin number 9 (24 V = max current 3A)
XP23	Service
XP28	WAN (connection to factory WAN).
XP25	LAN2 (connection of Ethernet based options).
XP26	LAN3 (connection of Ethernet based options).
XP11	FA = Fieldbus adapter PROFINET or EtherNet/IP (fieldbus adapter option)
XP24	USB port to main computer
Air L	Air supply, left arm O.D. 4 mm air hose, 0.6 MPa air pressure
Air R	Air supply, right arm O.D. 4 mm air hose, 0.6 MPa air pressure

Figure 55 shows The left side panel of the robot

In the Figure 55, the white cable was connected to a computer for the image processing purpose, the blue cable was attached to an air compressor for air supplying (Figure 56).

Figure 56 shows Puma air compressor

On the other hand, the right side panel (Figure 57) is included power switch, FlexPendant socket, and the Power Input (Table 5)

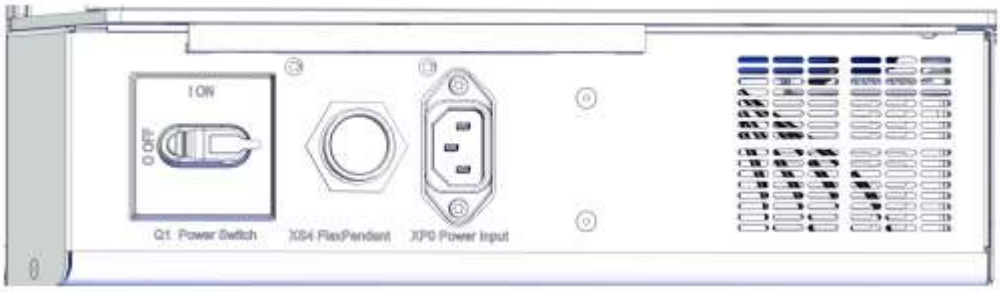


***Figure 57 shows the interface on the right side panel of the controller* (ABB, 2020, p. 81)**

Table 5: The clarification of symbol of the right panel (ABB, 2020)

Q1	Power switch
XS4	FlexPendant
XP0	Power input Main AC power connector, IEC 60320-1 C14, 100-240 VAC, 50-60 Hz

The IRB 14000 Yumi robot was connected to a computer, an ABB FlexPendant, an air compressor, and 220V outlet. (Figure 58)

Figure 58 shows the overview of the workplace

7 Conclusion and discusstion

The conducted project was finished by using the design of the CAD model that complied with the requirements, the Rapid program langue, ABB RobotStudio software, and the Cognex camera process. The project was adjusted several times to reach a better outcome and to support further extend if required.

Link for the simulation: https://youtu.be/VUHca6nQsYE

Link for the real robot: https://youtu.be/PjaHGfCNt30

During the process of the project, several problems were faced and dealt with. The original scope of the project intended to do the pick-up task by grippers and used 6 dices instead of 2 dices, but the actual situation appeared to be different. After the image processing was delivered the result to the right arm. If the dices stay near together, the right gripper has to go down and pick one dice up, it could cause a collision between the gripper finger and the other dices. To solve the issue, the rapid program needs to calculate the vector of the coordinate of the dices and the job of the gripper finger is to choose the orientation to pick up without touching the other dices, otherwise, it could change the side of the dices or collision with them. However, the duration of the project was too short to change the rapid code, and to pick up 6 dices perfectly, it needs more time to set-up and study more about the safety of the robot. This problem combined with a short period of time lead to a decision to conduct the project stop at a 2-dice application and use a vacuum cup to pick up.

The application may be used to support other or further researches in the field of Robotics. The reader can use this study as a tutorial for making an application for the IRB 14000 Yumi robot, otherwise, this project can be developed to be a complex application, for example, the Yumi is combined with other robots to be a process line with multiple tasks in a workplace.

Tables

Figures

References

ABB. (2015). ABB Robotics US Training. *Section 8: Programming YuMi*, pp. 2-38. Retrieved 8 2020

ABB. (2020, January). Operating manual RobotStudio. Retrieved August 2020

ABB. (2020, June 1). Product specification IRB 14000. pp. 0-126. Retrieved September 2020

ABB AB, Robotics. (2018). *Product manual IRB 14000 gripper.* ABB AB Robotics. Västerås: ABB Group. Retrieved 8 2020

ABB AB, Robotics. (2018, October 11). Technical reference manual. *RAPID Instructions, Functions and Data*, p. 1804. Retrieved September 2020

ABB Group. (n.d.). *Collaborative Robots*. Retrieved 8 2020, from https://new.abb.com: https://new.abb.com/products/robotics/collaborative-robots

ABB Group. (n.d.). *Detailed information for: 3HAC051676-001*. Retrieved 8 2020, from https://new.abb.com/: https://new.abb.com/products/3HAC051676-001/cognex-ae3-camera

ABB Group. (n.d.). *Technical data IRB 14000 YuMi*. Retrieved 8 2020, from https://new.abb.com: https://new.abb.com/products/robotics/industrial-robots/irb-14000-yumi/irb-14000-yumi-data

ABB Group. (n.d.). *The world's most used offline programming tool for robotics*. Retrieved 8 2020, from https://new.abb.com: https://new.abb.com/products/robotics/robotstudio

ABB Group. (n.d.). *YuMi® - IRB 14000 | Collaborative Robot*. Retrieved 8 2020, from https://new.abb.com: https://new.abb.com/products/robotics/industrial-robots/irb-14000-yumi

Barrette, M. B. (2016, January 27). *What Are the Best Collaborative Robots?* Retrieved August 2020, from https://blog.robotiq.com/: https://blog.robotiq.com/what-are-the-best-collaborative-robots

Bélanger Barrette, M. (2015, April 15). *Is ABB's YuMi the Next Generation of Collaborative Robot?* Retrieved 8 2020, from https://blog.robotiq.com: https://blog.robotiq.com/does-abb-yumi-is-the-next-generation-of-collaborative-robot

Bélanger-Barrette, M. (n.d.). *What Are the Best Collaborative Robots?* Retrieved September 2020, from http://www.robottechsupport.com: http://www.robottechsupport.com/collaborative-robots-compared-2/

Bloom, L. (2019, 1 22). *What Kind of Impact Do Robots Make on Humans?* Retrieved 8 2020, from https://bizfluent.com/: https://bizfluent.com/info-8478847-kind-do-robots-make-humans.html

Bredin, C. (2005, May 15). *ABB MultiMove functionality heralds a new era in the applications of robots*. Retrieved September 2020, from https://www.interempresas.net/: https://www.interempresas.net/Robotica/Articulos/10847-La-funcionalidad-MultiMove-de-ABB-anuncia-una-nueva-era-en-las-aplicaciones-de-robots.html

Cobot trends. (n.d.). *Cobots vs. industrial robots: what are the differences?* Retrieved September 2020, from https://www.cobottrends.com: https://www.cobottrends.com/cobots-vs-industrial-robots-what-are-differences/

Crowe, S. (n.d.). *Collaborative Robots Comparison Tool*. Retrieved September 2020, from https://www.cobottrends.com: https://www.cobottrends.com/cobot-comparison-tool/

Demaitre, E. (4. 3 2019). *6 advances in robotic grippers to watch*. Haettu 8 2020 osoitteesta https://www.therobotreport.com: https://www.therobotreport.com/robot-grippers-advance/

Essentra Components. (2020, 2 12). *Industry 4.0: Rise of the cobots?* Retrieved 8 2020, from https://www.essentracomponents.com: https://www.essentracomponents.com/en-gb/news/news-articles/industry-40-rise-of-the-cobots

Prusa Research. (n.d.). *THE ORIGINAL PRUSA I3 MK3S 3D PRINTER*. Retrieved 8 2020, from https://www.prusa3d.com: https://www.prusa3d.com/original-prusa-i3-mk3/

Richards, F. (2011, 9 1). *The difference between robotic grippers with parallel, three-finger, and angled designs*. Retrieved 8 2020, from https://www.machinedesign.com: https://www.machinedesign.com/markets/robotics/article/21833348/the-difference-between-robotic-grippers-with-parallel-threefinger-and-angled-designs

Roberson, D. (2020, 2 2). *The benefits of 3D printing by technology and industry*. Retrieved 8 2020, from https://ultimaker.com: https://ultimaker.com/learn/benefits-of-3d-printing-by-technology-and-industry

Teradyne Inc. (2020, 7). *Collaborative Robots*. Retrieved 8 2020, from https://www.teradyne.com/: https://www.teradyne.com/industrial-automation/collaborative-robots/#:~:text=Collaborative%20robots%20%28cobots%29%20are%20low-cost%2C%20easy-to-deploy%20robots%20that,processing%2C%20screw%20and%20nut%20driving%20and%20more%20

Appendix 1: Rapid program code of the left arm

```
MODULE Module1
    TASK PERS wobjdata wobj_workplace_left:=[FALSE,TRUE,"",[[493,60,0],[0,0,0,1]],[[0,0,0],[1,0,0,0]]];
    !define tooldate for the left gripper fingers
    PERS tooldata Servo_left:=[TRUE,[[0,0,110],[1,0,0,0]],[0.24,[8.2,12.5,48.1],[1,0,0,0],0.00022,0.00024,0.00009]];
    PERS tasks task_list_yumi{2}:=[["T_ROB_L"],["T_ROB_R"]];
    VAR syncident sync1a;
    VAR syncident sync2a;
    VAR syncident sync3a;
    VAR syncident sync4;
    VAR syncident sync5;
    VAR syncident sync6;
    VAR syncident sync7;
    VAR syncident sync8;
    VAR syncident sync9;
    VAR syncident last;

    !k is number of dices
    PERS num k:=0;
    !n is number of dices that are taken
    PERS num n:=0;
    !camera jobs are defined
    CONST string find_dice_1:="find_dice_1.job";
    CONST string find_dice_2:="find_dice_2.job";
    CONST string find_dice_3:="find_dice_3.job";
    CONST string find_dice_4:="find_dice_4.job";
    CONST string find_dice_5:="find_dice_5.job";
    CONST string find_dice_6:="find_dice_6.job";

    VAR cameratarget camera_target;
    !robot target for each camera jobs to save the coordinate of the dices
    PERS robtarget pTarget_dice_1;
    PERS robtarget pTarget_dice_2;
    PERS robtarget pTarget_dice_3;
    PERS robtarget pTarget_dice_4;
    PERS robtarget pTarget_dice_5;
    PERS robtarget pTarget_dice_6;
```

```
!the position to acquire photos
CONST robtarget pTake_photo:=[[288.89,-68.30,245.24],[0.0109576,0.00950534,0.698934,-0.715039],[-1,2,-2,5],[169.622,9E+09,9E+09,9E+09,9E+09,9E+09]];
!the home position
CONST robtarget calibL:=[[86.99,155.19,151.86],[0.074316,0.839707,-0.0982844,0.528875],[0,0,0,4],[102.083,9E+09,9E+09,9E+09,9E+09,9E+09]];
!the position to shake
CONST robtarget Target_10:=[[140.14,19.35,235.20],[0.70174,-0.711535,0.0314016,-0.0171335],[0,-1,1,0],[-105.624,9E+09,9E+09,9E+09,9E+09,9E+09]];
!define the position to pick up the cover
CONST robtarget pTake_cover:=[[29.91,-17.19,89.50],[0.0257692,-0.00252476,0.999456,-0.0204078],[-1,2,-2,4],[129.57,9E+09,9E+09,9E+09,9E+09,9E+09]];

PROC main_dices()
    k:=0;
    n:=0;
    !open gripper
    g_GripOut;
    !move the left arm to its home position
    MoveJ calibL,v2000,z50,Servo_left\WObj:=wobj0;

    !Wait until the arm reachs its destination
    WaitRob\InPos;
    Pickup_cover;
    Shaking_left;

    WaitSyncTask sync7,task_list_yumi;
    WHILE n<2 DO
        MoveJ pTake_photo,v3000,fine,Servo_left\WObj:=wobj0;
        Camera_Process;
        MoveJ Offs(pTake_photo,0,200,0),v1000,z100,Servo_left\WObj:=wobj0;
    ENDWHILE
    !move back to its home position to finish one process
    MoveJ calibL,v300,z50,Servo_left\WObj:=wobj0;
    WaitSyncTask last,task_list_yumi;
ENDPROC

PROC Pickup_cover()
    !path to pick up the cover
    MoveJ Offs(pTake_cover,100,0,100),v3000,z100,Servo_left\WObj:=wobj_workplace_left;
    MoveJ Offs(pTake_cover,0,0,100),v3000,z100,Servo_left\WObj:=wobj_workplace_left;
    MoveL pTake cover,v50,z0,Servo left\WObj:=wobj workplace left;
```

```
75          WaitRob\InPos;
76          !close gripper
77          g_GripIn;
78          WaitTime\InPos,0.5;
79          MoveL Offs(pTake_cover,0,0,100),v3000,z50,Servo_left\WObj:=wobj_workplace_left;
80      ENDPROC
81
82      PROC Shaking_left()
83          MoveJ Offs(Target_10,0,-100,0),v3000,z100,Servo_left\WObj:=wobj_workplace_left;
84          WaitRob\InPos;
85          !Wait for the right arm reachs the waitsynctask sync1a code line
86          WaitSyncTask sync1a,task_list_yumi;
87          WaitTime\InPos,0.5;
88          WaitSyncTask sync2a,task_list_yumi;
89
90          !Using multimove to make both arms shaking at the same speed
91          FOR i FROM 1 TO 3 DO
92              SyncMoveOn sync3a,task_list_yumi;
93              MoveL Offs(Target_10,40,0,0)\ID:=10,vmax,z200,Servo_left\WObj:=wobj_workplace_left;
94              MoveL RelTool(Offs(Target_10,40,0,0),0,0,0\Rz:=40)\ID:=20,vmax,z200,Servo_left\WObj:=wobj_workplace_left;
95              MoveL Offs(Target_10,0,-40,0)\ID:=30,vmax,z200,Servo_left\WObj:=wobj_workplace_left;
96              MoveL RelTool(Offs(Target_10,0,-40,0),0,0,0\Rz:=-40)\ID:=40,vmax,z200,Servo_left\WObj:=wobj_workplace_left;
97              MoveL Offs(Target_10,0,0,-40)\ID:=50,vmax,z200,Servo_left\WObj:=wobj_workplace_left;
98              MoveL RelTool(Offs(Target_10,0,0,-40),0,0,0\Rz:=40)\ID:=60,vmax,z200,Servo_left\WObj:=wobj_workplace_left;
99              MoveL Offs(Target_10,0,40,0)\ID:=70,vmax,z200,Servo_left\WObj:=wobj_workplace_left;
100             MoveL RelTool(Offs(Target_10,0,40,0),0,0,0\Rz:=-40)\ID:=80,vmax,z200,Servo_left\WObj:=wobj_workplace_left;
101
102             MoveL Target_10\ID:=90,vmax,z200,Servo_left\WObj:=wobj_workplace_left;
103             SyncMoveOff sync4;
104         ENDFOR
105
106         !Both arms go downward to throw the dices out
107         SyncMoveOn sync5,task_list_yumi;
108         MoveL Offs(Target_10,0,0,-105)\ID:=10,v100,z10,Servo_left\WObj:=wobj_workplace_left;
109         MoveL Offs(Target_10,50,0,-105)\ID:=20,v100,z10,Servo_left\WObj:=wobj_workplace_left;
110         MoveL RelTool(Offs(Target_10,50,0,-105),0,0,0\Rz:=90)\ID:=30,v10,z0,Servo_left\WObj:=wobj_workplace_left;
111         MoveL Offs(RelTool(Offs(Target_10,50,0,-105),0,0,0\Rz:=90),-10,0,-62)\ID:=40,v10,z10,Servo_left\WObj:=wobj_workpl
112         MoveL Offs(RelTool(Offs(Target_10,50,0,-105),0,0,0\Rz:=90),-10,-20,-62)\ID:=50,v10,z10,Servo_left\WObj:=wobj_work
113         SyncMoveOff sync6;
114         WaitTime\InPos,0.8;
115
```

```
116         MoveL RelTool(Target_10,50,50,-105\Rz:=90),v1000,z100,Servo_left\WObj:=wobj_workplace_left;
117         MoveL RelTool(Target_10,0,0,0\Rz:=90),v1000,z100,Servo_left\WObj:=wobj_workplace_left;
118         !bring the cover back to its storage
119         MoveJ Offs(pTake_cover,0,0,100),v3000,z100,Servo_left\WObj:=wobj_workplace_left;
120         MoveL Offs(pTake_cover,0,4.5,5),v100,z0,Servo_left\WObj:=wobj_workplace_left;
121         WaitRob\InPos;
122         !open gripper
123         g_GripOut;
124         WaitTime\InPos,0.5;
125         MoveL Offs(pTake_cover,0,0,100),v3000,z50,Servo_left\WObj:=wobj_workplace_left;
126     ENDPROC
127
128     PROC Camera_Process()
129         !the number of used dices is 2, so if the taken dices are 2, the camera_process is stop
130         IF n=2 THEN
131             RETURN ;
132         ENDIF
133
134         k:=k+1;
135         pTarget_dice_6.trans.x:=0;
136         pTarget_dice_5.trans.x:=0;
137         pTarget_dice_4.trans.x:=0;
138         pTarget_dice_3.trans.x:=0;
139         pTarget_dice_2.trans.x:=0;
140         pTarget_dice_1.trans.x:=0;
141         find_dice_number6;
142         find_dice_number5;
143         find_dice_number4;
144         find_dice_number3;
145         find_dice_number2;
146         find_dice_number1;
147
148
149     ENDPROC
150
151     PROC find_dice_number1()
152         IF n=2 THEN
153             RETURN ;
154         ENDIF
155         !Orders the camera to go to program mode
156         CamSetProgramMode kamera_nova;
```

```
        !Load a camera task into a camera
        CamLoadJob kamera_nova,find_dice_1;
        !Orders the camera to run mode
        CamSetRunMode kamera_nova;
        !Order the camera to acquire an image
        CamReqImage kamera_nova;
        !Gets a camera target from the collection
        CamGetResult kamera_nova,camera_target\MaxTime:=1;
        IF camera_target.val1=1 THEN
            pTarget_dice_1.trans.x:=366-camera_target.cframe.trans.y;
            pTarget_dice_1.trans.y:=84-camera_target.cframe.trans.x;
            n:=n+1;
            MoveJ Offs(pTake_photo,0,200,0),v1000,z100,Servo_left\WObj:=wobj0;
            WaitSyncTask sync8,task_list_yumi;
            WaitSyncTask sync9,task_list_yumi;
            MoveJ pTake_photo,v1000,fine,Servo_left\WObj:=wobj0;
            WaitRob\InPos;
            Camera_Process;
        ENDIF
        !if the job doesnt get any result, it goes to error
    ERROR
        IF ERRNO=ERR_CAM_MAXTIME THEN
            RETURN ;
        ENDIF
    ENDPROC
    ! the explaination is the same in the other routine

    PROC find_dice_number2()
        IF n=2 THEN
            RETURN ;
        ENDIF

        CamSetProgramMode kamera_nova;
        CamLoadJob kamera_nova,find_dice_2;
        CamSetRunMode kamera_nova;
        CamReqImage kamera_nova;
        CamGetResult kamera_nova,camera_target\MaxTime:=1;
        IF camera_target.val1=1 THEN
            pTarget_dice_2.trans.x:=366-camera_target.cframe.trans.y;
            pTarget_dice_2.trans.y:=84-camera_target.cframe.trans.x;
            n:=n+1;
```

```
198             MoveJ Offs(pTake_photo,0,200,0),v1000,z100,Servo_left\WObj:=wobj0;
199             WaitSyncTask sync8,task_list_yumi;
200             WaitSyncTask sync9,task_list_yumi;
201             MoveJ pTake_photo,v1000,fine,Servo_left\WObj:=wobj0;
202             WaitRob\InPos;
203             Camera_Process;
204         ENDIF
205     ERROR
206         IF ERRNO=ERR_CAM_MAXTIME THEN
207             RETURN ;
208         ENDIF
209     ENDPROC
210
211     PROC find_dice_number3()
212         IF n=2 THEN
213             RETURN ;
214         ENDIF
215
216         CamSetProgramMode kamera_nova;
217         CamLoadJob kamera_nova,find_dice_3;
218         CamSetRunMode kamera_nova;
219         CamReqImage kamera_nova;
220         CamGetResult kamera_nova,camera_target\MaxTime:=1;
221         IF camera_target.val1=1 THEN
222             pTarget_dice_3.trans.x:=366-camera_target.cframe.trans.y;
223             pTarget_dice_3.trans.y:=84-camera_target.cframe.trans.x;
224             n:=n+1;
225             MoveJ Offs(pTake_photo,0,200,0),v1000,z100,Servo_left\WObj:=wobj0;
226             WaitSyncTask sync8,task_list_yumi;
227             WaitSyncTask sync9,task_list_yumi;
228             MoveJ pTake_photo,v1000,fine,Servo_left\WObj:=wobj0;
229             WaitRob\InPos;
230             Camera_Process;
231         ENDIF
232     ERROR
233         IF ERRNO=ERR_CAM_MAXTIME THEN
234             RETURN ;
235         ENDIF
236     ENDPROC
237
```

```
238     PROC find_dice_number4()
239         IF n=2 THEN
240             RETURN ;
241         ENDIF
242 
243         CamSetProgramMode kamera_nova;
244         CamLoadJob kamera_nova,find_dice_4;
245         CamSetRunMode kamera_nova;
246         CamReqImage kamera_nova;
247         CamGetResult kamera_nova,camera_target\MaxTime:=1;
248         IF camera_target.val1=1 THEN
249             pTarget_dice_4.trans.x:=366-camera_target.cframe.trans.y;
250             pTarget_dice_4.trans.y:=84-camera_target.cframe.trans.x;
251             n:=n+1;
252             MoveJ Offs(pTake_photo,0,200,0),v1000,z100,Servo_left\WObj:=wobj0;
253             WaitSyncTask sync8,task_list_yumi;
254             WaitSyncTask sync9,task_list_yumi;
255             MoveJ pTake_photo,v1000,fine,Servo_left\WObj:=wobj0;
256             WaitRob\InPos;
257             Camera_Process;
258         ENDIF
259     ERROR
260         IF ERRNO=ERR_CAM_MAXTIME THEN
261             RETURN ;
262         ENDIF
263     ENDPROC
264 
265 
266     PROC find_dice_number5()
267         IF n=2 THEN
268             RETURN ;
269         ENDIF
270 
271         CamSetProgramMode kamera_nova;
272         CamLoadJob kamera_nova,find_dice_5;
273         CamSetRunMode kamera_nova;
274         CamReqImage kamera_nova;
275         CamGetResult kamera_nova,camera_target\MaxTime:=1;
276         IF camera_target.val1=1 THEN
277             pTarget_dice_5.trans.x:=366-camera_target.cframe.trans.y;
278             pTarget_dice_5.trans.y:=84-camera_target.cframe.trans.x;
```

```
279             n:=n+1;
280             MoveJ Offs(pTake_photo,0,200,0),v1000,z100,Servo_left\WObj:=wobj0;
281             WaitSyncTask sync8,task_list_yumi;
282             WaitSyncTask sync9,task_list_yumi;
283             MoveJ pTake_photo,v1000,fine,Servo_left\WObj:=wobj0;
284             WaitRob\InPos;
285             Camera_Process;
286         ENDIF
287     ERROR
288         IF ERRNO=ERR_CAM_MAXTIME THEN
289             RETURN ;
290         ENDIF
291     ENDPROC
292
293     PROC find_dice_number6()
294         IF n=2 THEN
295             RETURN ;
296         ENDIF
297         pTarget_dice_6.trans.x:=0;
298         pTarget_dice_6.trans.y:=0;
299         CamSetProgramMode kamera_nova;
300         CamLoadJob kamera_nova,find_dice_6;
301         CamSetRunMode kamera_nova;
302         CamReqImage kamera_nova;
303         CamGetResult kamera_nova,camera_target\MaxTime:=1;
304         IF camera_target.val1=1 THEN
305             pTarget_dice_6.trans.x:=366-camera_target.cframe.trans.y;
306             pTarget_dice_6.trans.y:=84-camera_target.cframe.trans.x;
307             n:=n+1;
308             MoveJ Offs(pTake_photo,0,200,0),v1000,z100,Servo_left\WObj:=wobj0;
309             WaitSyncTask sync8,task_list_yumi;
310             WaitSyncTask sync9,task_list_yumi;
311             MoveJ pTake_photo,v1000,fine,Servo_left\WObj:=wobj0;
312             WaitRob\InPos;
313             Camera_Process;
314         ENDIF
315     ERROR
316         IF ERRNO=ERR_CAM_MAXTIME THEN
317             RETURN ;
318         ENDIF
319     ENDPROC
```

Appendix 2: Rapid program code of the right arm

```
MODULE Module1
    TASK PERS wobjdata wobj_workplace_right:=[FALSE,TRUE,"",[[493,60,0],[0,0,0,1]],[[0,0,0],[1,0,0,0]]];
    !define tooldata for the right gripper finger
    PERS tooldata Servo_right:=[TRUE,[[0,0,110],[1,0,0,0]],[0.25,[7.4,12.4,44.8],[1,0,0,0],0.00025,0.00029,0.00012]];
    PERS tasks task_list_yumi{2}:=[["T_ROB_L"],["T_ROB_R"]];
    VAR syncident sync1a;
    VAR syncident sync2a;
    VAR syncident sync3a;
    VAR syncident sync4;
    VAR syncident sync5;
    VAR syncident sync6;
    VAR syncident sync7;
    VAR syncident sync8;
    VAR syncident sync9;
    VAR syncident last;

    !number of dices are used for the application
    PERS num k;
    !number of dices are taken by the vacuum cup
    PERS num n;

    !robot targets to save the coordinate of dices from camera jobs
    PERS robtarget pTarget_dice_1;
    PERS robtarget pTarget_dice_2;
    PERS robtarget pTarget_dice_3;
    PERS robtarget pTarget_dice_4;
    PERS robtarget pTarget_dice_5;
    PERS robtarget pTarget_dice_6;
    !define tooldate for the vacuum number one on the right gripper
    PERS tooldata VaccumOne:=[TRUE,[[63.5,18.5,37.5],[0.707106781,0,0.707106781,0]],[0.25,[7.4,12.4,44.8],[1,0,0,0],0.00025,0.00029,0.0001
    !the position to pick up the holder
    CONST robtarget pTake_holder:=[[31.67,126.02,75.33],[0.00839056,0.00417169,-0.713822,0.700264],[1,-1,2,4],[-127.294,9E+09,9E+09,9E+09,
    !the position to shake
    CONST robtarget Target_10:=[[143,135,195],[0.000000616,0.000000192,0.70710706,-0.707106502],[1,-1,2,4],[-101.964430178,9E+09,9E+09,9E+
    !if 2 dices have the same value, the order_dice target will change to make sure that there is no collision
    PERS robtarget pOrder_dice:=[[260,-252,18],[7.5E-08,0.707107,-0.707107,-6.4E-08],[1,1,2,4],[93.3088,9E+09,9E+09,9E+09,9E+09,9E+09]];
    !the position to pick up dices from the workplace, it will be changed in the program
    !the purpose to define this because the figuration of the gripper remain the same
    PERS robtarget Target_Takedices:=[[281.149,2.63836,17.5],[3.74E-07,0.707107,-0.707107,-1.15E-07],[2,1,2,4],[93.3088,9E+09,9E+09,9E+09,
```

```
    !home position of the right arm
    CONST robtarget calibR:=[[87.85,-162.60,154.25],[0.0587748,-0.838077,-0.1254,-0.527682],[0,0,0,4],[-101.673,9E+09,9E
    !this target will save the coordinate of the first taken dice
    !after the dices are picked up from the workplace, it needs to be back into the holder
    !that's why this p1 target needs to be defined
    PERS robtarget p1:=[[260,-232,17.5],[7.5E-08,0.707107,-0.707107,-6.4E-08],[1,1,2,4],[93.3088,9E+09,9E+09,9E+09,9E+09
    !the positon to drop the dices back to the holder
    CONST robtarget pDrop_dice:=[[461.41,-3.49,87.95],[8.01496E-05,0.707103,-0.707111,2.72539E-05],[2,1,2,4],[101.76,9E+

    PROC main_dices()
        !open the gripper
        g_GripOut;
        !move back to its home positon
        MoveJ calibR,v3000,z50,Servo_right\WObj:=wobj0;
        !wait until all movement are stop
        WaitRob\InPos;

        Pickup_holder;
        Shaking_right;
        WaitSyncTask sync7,task_list_yumi;
        !the loop to make sure the application will pick up all dices
        WHILE n<=2 DO
            WaitSyncTask sync8,task_list_yumi;
            Process_dices;
            !if the number of dices is enough, stop the loop
            IF n=2 THEN
                WaitTime\InPos,2;
                Return_dices;
                WaitSyncTask last,task_list_yumi;
                RETURN;
            ENDIF
        ENDWHILE

    ENDPROC
```

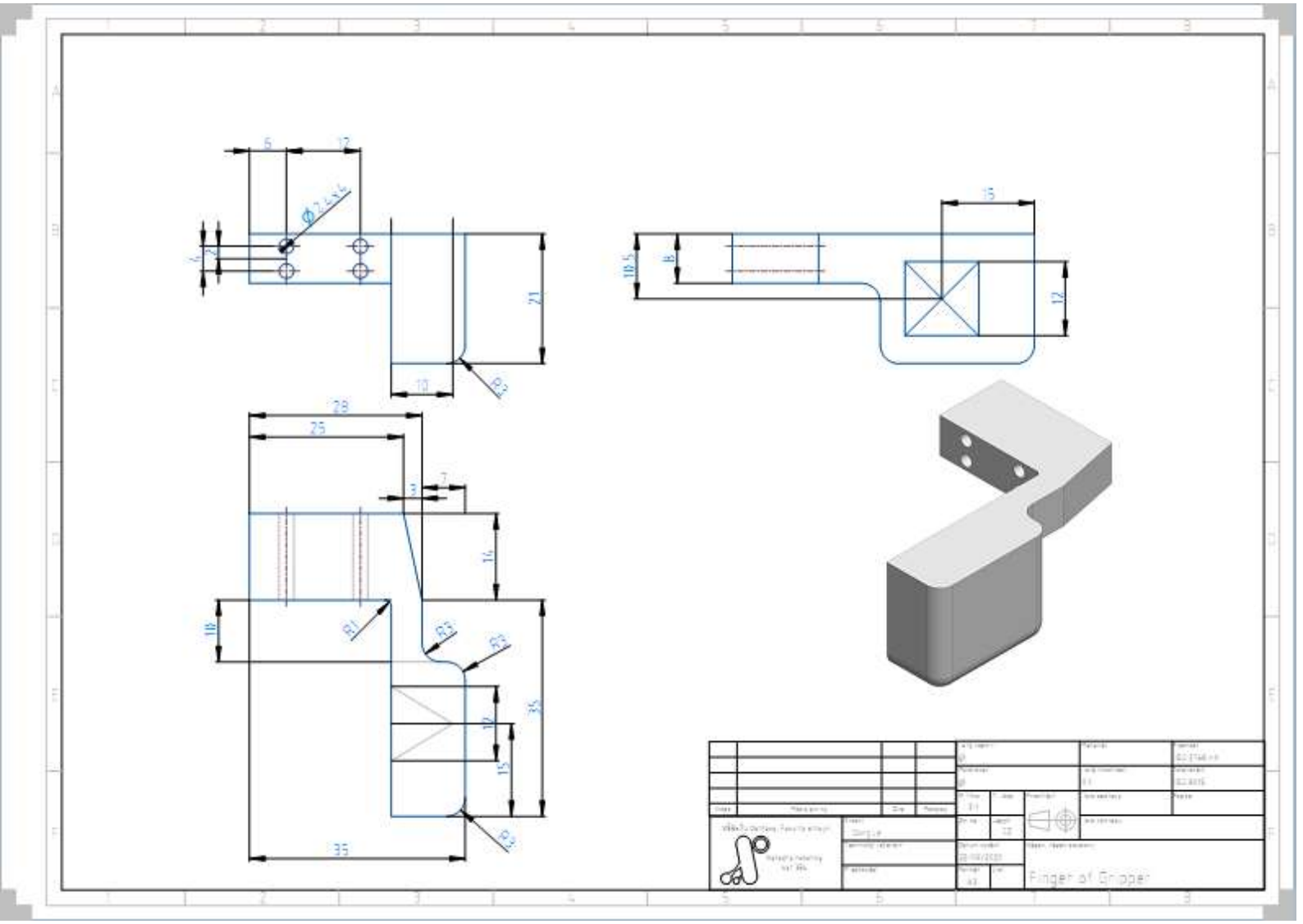
```
PROC Pickup_holder()
    !pick up the holder
    MoveJ Offs(pTake_holder,0,0,100),v3000,z100,Servo_right\WObj:=wobj_workplace_right;
    MoveL pTake_holder,v50,z0,Servo_right\WObj:=wobj_workplace_right;
    WaitTime\InPos,0.2;
    !close the gripper to hold
    g_GripIn;
    WaitTime\InPos,0.5;
    MoveL Offs(pTake_holder,0,0,100),v3000,z50,Servo_right\WObj:=wobj_workplace_right;
ENDPROC

PROC Shaking_right()
    MoveJ Offs(Target_10,0,50,0),v3000,z50,Servo_right\WObj:=wobj_workplace_right;
    WaitSyncTask sync1a,task_list_yumi;
    MoveL Target_10,v50,z0,Servo_right\WObj:=wobj_workplace_right;
    WaitTime\InPos,0.5;
    WaitSyncTask sync2a,task_list_yumi;

    !the loop to shake the dice, it can increse to shake more
    FOR i FROM 1 TO 3 DO
        SyncMoveOn sync3a,task_list_yumi;
        MoveL Offs(Target_10,40,0,0)\ID:=10,vmax,z200,Servo_right\WObj:=wobj_workplace_right;
        MoveL RelTool(Offs(Target_10,40,0,0),0,0,0\Rz:=-40)\ID:=20,vmax,z200,Servo_right\WObj:=wobj_workplace_right
        MoveL Offs(Target_10,0,-40,0)\ID:=30,vmax,z200,Servo_right\WObj:=wobj_workplace_right;
        MoveL RelTool(Offs(Target_10,0,-40,0),0,0,0\Rz:=40)\ID:=40,vmax,z200,Servo_right\WObj:=wobj_workplace_right
        MoveL Offs(Target_10,0,0,-40)\ID:=50,vmax,z200,Servo_right\WObj:=wobj_workplace_right;
        MoveL RelTool(Offs(Target_10,0,0,-40),0,0,0\Rz:=-40)\ID:=60,vmax,z200,Servo_right\WObj:=wobj_workplace_righ
        MoveL Offs(Target_10,0,40,0)\ID:=70,vmax,z200,Servo_right\WObj:=wobj_workplace_right;
        MoveL RelTool(Offs(Target_10,0,40,0),0,0,0\Rz:=40)\ID:=80,vmax,z200,Servo_right\WObj:=wobj_workplace_right;

        MoveL Target_10\ID:=90,vmax,z200,Servo_right\WObj:=wobj_workplace_right;
        SyncMoveOff sync4;
    ENDFOR

    !the multimove system to go down and throw dices at the same speed
    SyncMoveOn sync5,task_list_yumi;
    MoveL Offs(Target_10,0,0,-105)\ID:=10,v100,z10,Servo_right\WObj:=wobj_workplace_right;
    MoveL Offs(Target_10,0,0,-125)\ID:=20,v100,z10,Servo_right\WObj:=wobj_workplace_right;
```

```
        MoveL RelTool(Offs(Target_10,0,0,-125),0,0,0\Rz:=-90)\ID:=30,v10,z0,Servo_right\WObj:=wobj_workplace_right;
        MoveL RelTool(Offs(Target_10,0,0,-125),0,0,0\Rz:=-103.5)\ID:=40,v10,z10,Servo_right\WObj:=wobj_workplace_right;
        MoveL Offs(RelTool(Offs(Target_10,0,0,-125),0,0,0\Rz:=-108),0,0,-8)\ID:=50,v10,z10,Servo_right\WObj:=wobj_workpl
        SyncMoveOff sync6;
        WaitTime\InPos,2;

        !bring the holder back to its storage
        MoveL RelTool(Offs(Target_10,0,0,-125),0,0,0\Rz:=-90),v300,z100,Servo_right\WObj:=wobj_workplace_right;
        MoveL RelTool(Offs(Target_10,0,100,-125),0,0,0\Rz:=-90),v300,z100,Servo_right\WObj:=wobj_workplace_right;
        MoveJ Offs(pTake_holder,0,0,100),v300,z50,Servo_right\WObj:=wobj_workplace_right;
        MoveL Offs(pTake_holder,0,0,5),v50,z0,Servo_right\WObj:=wobj_workplace_right;
        WaitRob\InPos;
        g_GripOut;
        WaitTime\InPos,0.5;
        MoveL Offs(pTake_holder,0,0,100),v300,z50,Servo_right\WObj:=wobj_workplace_right;
    ENDPROC

    PROC Process_dices()
        !if the x-axis of the target pTarget_dice_6 is changed based on the image process from left arm
        !the pOrder_dice and the Target_Takedices are changed to engage the Pick_place_dices routine
        IF pTarget_dice_6.trans.x<>0 THEN
            pOrder_dice.trans.x:=200;
            pOrder_dice.trans.y:=-232-(k-1)*20;
            Target_Takedices.trans.x:=pTarget_dice_6.trans.x;
            Target_Takedices.trans.y:=pTarget_dice_6.trans.y;

            Pick_place_dices;
        ENDIF

        IF pTarget_dice_5.trans.x<>0 THEN
            pOrder_dice.trans.x:=220;
            pOrder_dice.trans.y:=-232-(k-1)*20;
            Target_Takedices.trans.x:=pTarget_dice_5.trans.x;
            Target_Takedices.trans.y:=pTarget_dice_5.trans.y;

            Pick_place_dices;
        ENDIF

```

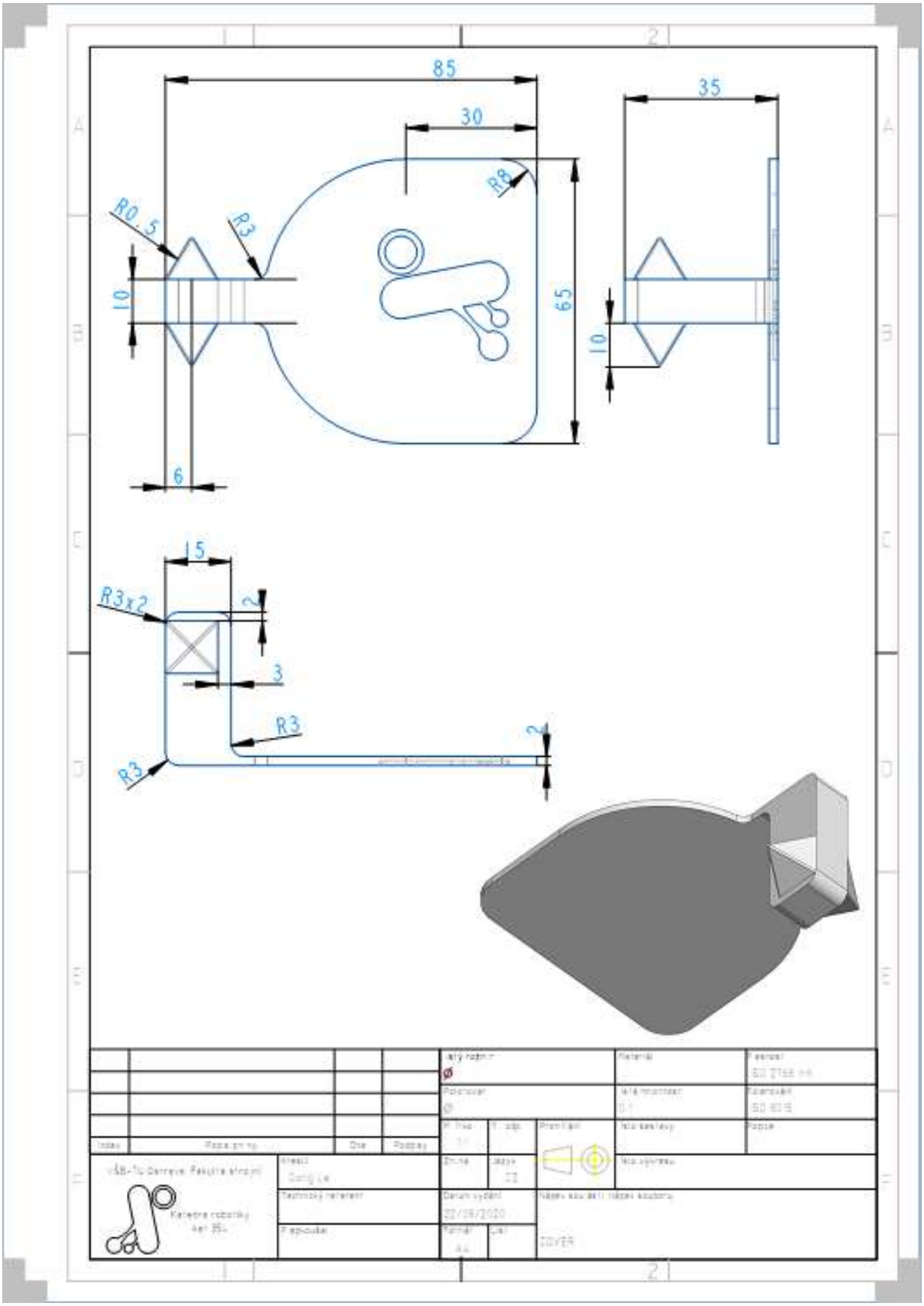
```
151         IF pTarget_dice_4.trans.x<>0 THEN
152             pOrder_dice.trans.x:=240;
153             pOrder_dice.trans.y:=-232-(k-1)*20;
154             Target_Takedices.trans.x:=pTarget_dice_4.trans.x;
155             Target_Takedices.trans.y:=pTarget_dice_4.trans.y;
156 
157             Pick_place_dices;
158         ENDIF
159 
160         IF pTarget_dice_3.trans.x<>0 THEN
161             pOrder_dice.trans.x:=260;
162             pOrder_dice.trans.y:=-232-(k-1)*20;
163             Target_Takedices.trans.x:=pTarget_dice_3.trans.x;
164             Target_Takedices.trans.y:=pTarget_dice_3.trans.y;
165             Pick_place_dices;
166         ENDIF
167 
168         IF pTarget_dice_2.trans.x<>0 THEN
169             pOrder_dice.trans.x:=280;
170             pOrder_dice.trans.y:=-232-(k-1)*20;
171             Target_Takedices.trans.x:=pTarget_dice_2.trans.x;
172             Target_Takedices.trans.y:=pTarget_dice_2.trans.y;
173 
174             Pick_place_dices;
175         ENDIF
176 
177         IF pTarget_dice_1.trans.x<>0 THEN
178             pOrder_dice.trans.x:=300;
179             pOrder_dice.trans.y:=-232-(k-1)*20;
180             Target_Takedices.trans.x:=pTarget_dice_1.trans.x;
181             Target_Takedices.trans.y:=pTarget_dice_1.trans.y;
182 
183             Pick_place_dices;
184         ENDIF
185     ENDPROC
186 
```

```
    PROC Pick_place_dices()
        !if the number of the taken dice is 1, save the coordinate to the p1 target
        IF n=1 THEN
            p1.trans.x:=pOrder_dice.trans.x;
            p1.trans.y:=pOrder_dice.trans.y;
        ENDIF
        MoveJ Offs(Target_Takedices,0,0,80),v300,z50,VaccumOne\WObj:=wobj0;
        MoveL Target_Takedices,v10,fine,VaccumOne\WObj:=wobj0;
        WaitRob\InPos;
        !vacuum is on
        g_VacuumOn1;
        WaitTime 1;

        MoveJ Offs(Target_Takedices,0,0,12),v300,z200,VaccumOne\WObj:=wobj0;
        MoveJ Offs(Target_Takedices,-30,0,20),v300,z200,VaccumOne\WObj:=wobj0;
        MoveJ Offs(pOrder_dice,0,200,80),v300,z200,VaccumOne\WObj:=wobj0;
        MoveJ Offs(pOrder_dice,0,0,80),v300,z200,VaccumOne\WObj:=wobj0;
        MoveL pOrder_dice,v10,z200,VaccumOne\WObj:=wobj0;
        WaitRob\InPos;
        !vacuum is off
        g_VacuumOff1;
        MoveL Offs(pOrder_dice,0,0,80),v10,z200,VaccumOne\WObj:=wobj0;
        WaitSyncTask sync9,task_list_yumi;
    ENDPROC

    PROC Return_dices()
        !take the first dice
        WaitTime\InPos,1;
        MoveJ Offs(pOrder_dice,0,0,50),v1000,z50,VaccumOne\WObj:=wobj0;
        MoveL pOrder_dice,v1000,z50,VaccumOne\WObj:=wobj0;
        WaitRob\InPos;
        g_VacuumOn1;
        WaitTime 1;
        MoveL Offs(pOrder_dice,0,0,50),v1000,z50,VaccumOne\WObj:=wobj0;
        MoveJ Offs(pDrop_dice,0,0,50),v1000,z50,VaccumOne\WObj:=wobj0;
        MoveL pDrop_dice,v1000,z50,VaccumOne\WObj:=wobj0;
        WaitRob\InPos;
        g_VacuumOff1;
        MoveJ Offs(pDrop_dice,0,0,50),v1000,z50,VaccumOne\WObj:=wobj0;


        !take the second dice
        MoveJ Offs(p1,0,0,50),v1000,z50,VaccumOne\WObj:=wobj0;
        MoveL p1,v1000,z50,VaccumOne\WObj:=wobj0;
        WaitRob\InPos;
        g_VacuumOn1;
        WaitTime 1;
        MoveL Offs(p1,0,0,50),v1000,z50,VaccumOne\WObj:=wobj0;
        MoveJ Offs(pDrop_dice,0,0,50),v1000,z50,VaccumOne\WObj:=wobj0;
        MoveL pDrop_dice,v1000,z50,VaccumOne\WObj:=wobj0;
        WaitRob\InPos;
        g_VacuumOff1;
        MoveJ Offs(pDrop_dice,0,0,50),v1000,z50,VaccumOne\WObj:=wobj0;
        MoveJ Offs(pDrop_dice,0,-50,50),v1000,z50,VaccumOne\WObj:=wobj0;
    ENDPROC
ENDMODULE
```

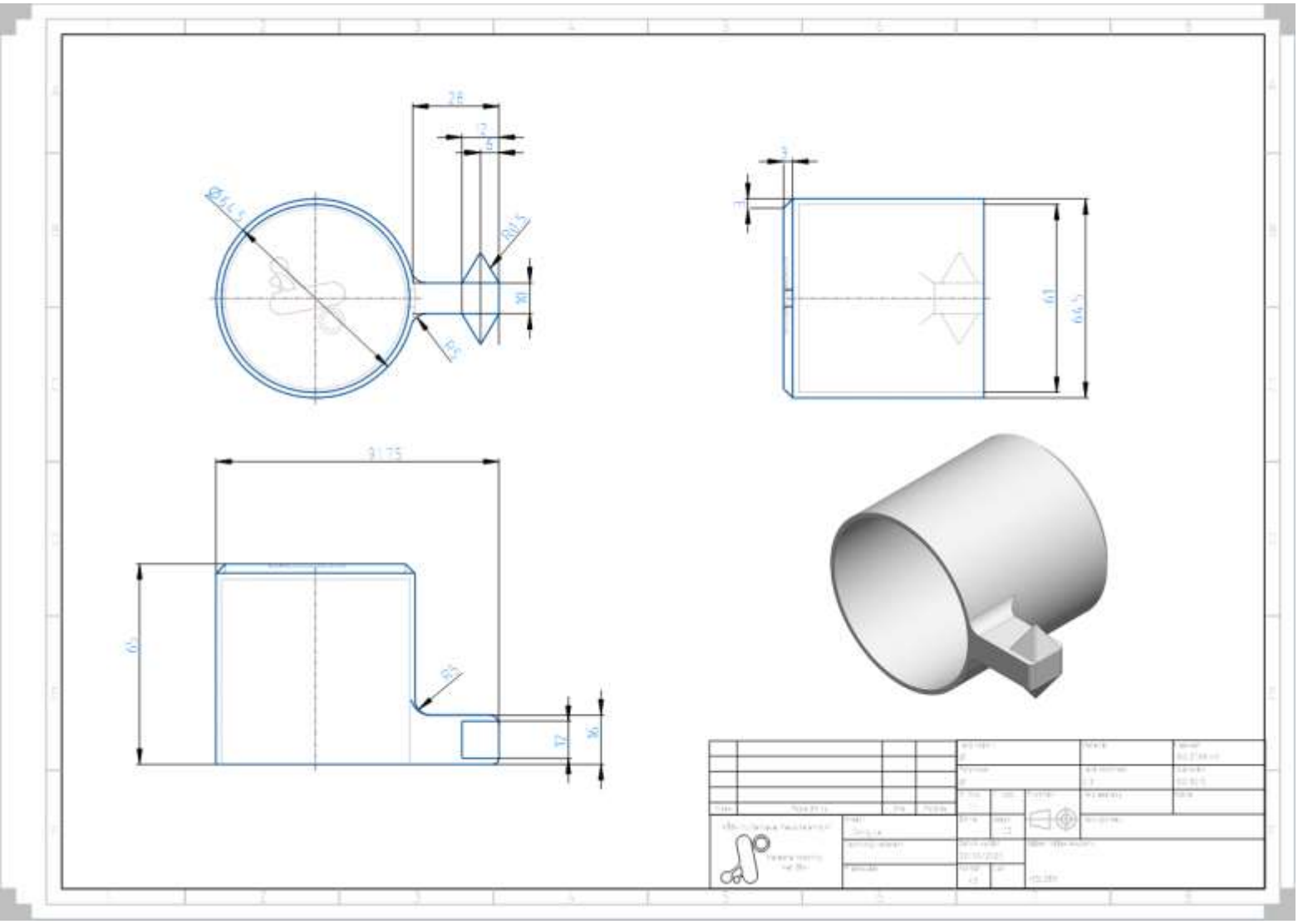
Appendix 3: Manufacturing drawing of the gripper finger



Appendix 4: Manufacturing drawing of the cover



Appendix 5: Manufacturing drawing of the holder



Appendix 6: Manufacturing drawing of the workplace