MUNICH SCHOOL OF ROBOTICS AND MACHINE INTELLIGENCE

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Electrical Engineering and Information Technology

Automated preparation of culture media & other process chemicals through human-robot collaboration

Yevhen Prots

MUNICH SCHOOL OF ROBOTICS AND MACHINE INTELLIGENCE

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Electrical Engineering and Information Technology

Automated preparation of culture media & other process chemicals through human-robot collaboration

Author:	Yevhen Prots
Supervisor:	M.Sc. Dennis Knobbe
Advisor:	Prof. Dr.-Ing. Sami Haddadin
Submission Date:	July 9, 2020

I confirm that this bachelor’s thesis is my own work and I have documented all sources and material used.

München, July 9, 2020 Yevhen Prots

Abstract

Automation of laboratory processes in chemistry, bio, pharmaceutical, food technology, and medicine is already a reality. But many solutions that are on the market today are either too highly-priced and/or only especially and inflexibly developed and optimized only for certain processes in the laboratory. During this thesis the task of laboratory automation for the dynamic workflow in everyday laboratory work was addressed as a tool for all and the provision of an option for human-robot cooperation for example for result validation. In the course of the thesis the capability of automated production of culture media & other process chemicals by human-robot collaboration was created.

For these purposes, a light-weight robotic arm, Panda by Franka Emika, was used. To allow the robot to perform all of the required tasks, multiple toolsets for flexible end-effector utilization were created in SolidWorks and then printed out on a 3D printer out of thermoplastic. In order to analyze all of the necessary input data produced by the robot camera, several Python scripts were written. Those scripts use the OpenCV library to conduct the required operations on the input pictures to extract the necessary information, which is adopted in the entire process.

The graphical user interface (GUI) was created to provide users with a simple input option to customize the process for their needs and to protocol all of the steps throughout the process. The created GUI provides an option not only to fulfill those needs but it creates a platform for further integration of more advanced concepts as well. Those concepts were addressed and discussed too and can grant a chance to base a future development of the system with a goal of further increase of autonomy grade of the whole system or simple general system improvement as such.

Contents

Abstract III

1 Introduction 1
- 1.1 Technological state . . . 1
- 1.2 State-of-the-art . . . 2
- 1.3 Objectives and structure . . . 3

2 Theroretical background 5
- 2.1 Robotics theory . . . 5
- 2.2 Buffer solutions . . . 7

3 Materials & Methods 9
- 3.1 Equipment . . . 9
- 3.2 Buffer & Protocol . . . 11
- 3.3 Robot . . . 11
  - 3.3.1 Description . . . 11
  - 3.3.2 Manipulator prerequisites . . . 12
  - 3.3.3 Communication with the robot . . . 14

4 Development of the end-effector tools 19
- 4.1 Small manipulations toolset . . . 19
- 4.2 Bottle manipulations toolset . . . 21
- 4.3 Core inserting & extraction toolset . . . 24

5 Development of the Process Software 29
- 5.1 Control software . . . 29
  - 5.1.1 Process algorithm . . . 29
  - 5.1.2 Desk solution . . . 31
  - 5.1.3 Precision improvement . . . 34
- 5.2 Camera input software . . . 35
  - 5.2.1 General approach . . . 35
  - 5.2.2 Core position detection . . . 36
  - 5.2.3 Scales output recognition . . . 38
- 5.3 User Interface . . . 41
- 5.4 Software evaluation . . . 43

6 Discussion & Further Concepts 45
- 6.1 Remote connection . . . 45
- 6.2 Calibration . . . 47

6.3 Machine learning 49

7 Conclusion **50**

8 References **51**

1 Introduction

Since about 1875 there have been reports of automated devices for scientific research. These first devices were mostly built by scientists themselves in order to solve problems in the laboratory.[12] Automation steadily spread in laboratories through the 20th century, but then a revolution took place in the early 1980s - the first fully automated laboratory was opened by Dr.Masahide Sasaki.[6][2] In 1993, Dr. Rod Markin at the University of Nebraska Medical Center created one of the world's first clinical automated laboratory management systems.[9] Despite the success of Dr. Sasaki laboratory and others of the kind, the multi-million dollar cost of such laboratories has prevented adoption by smaller groups.[5] Everything is even more difficult because devices made by different manufacturers often cannot communicate with each other. However, recent advances based on the use of scripting languages like Autovit have made possible the integration of equipment from different manufacturers.[3] Using this approach, many low-cost electronic devices, including open-source devices,[14] are becoming more and more compatible with common laboratory instruments.

1.1 Technological state

Pharmaceutical companies employ robots to move biological or chemical samples around to synthesize novel chemical entities or to test the pharmaceutical value of existing chemical matter.[11][17] The usage of the robotic assistants in laboratory workflow proves itself extremely useful when it comes to handling dangerous mediums which might put human health or even life at risk, like in the case of performing acid digestion chemical analysis. Laboratory processes are suited for robotic automation as the processes are composed of repetitive movements (e.g. pick/place, liquid & solid additions, heating/cooling, mixing, shaking, testing).

A large obstacle to the implementation of automation in laboratories has been its high cost. Many laboratory instruments are very expensive. This is justifiable in many cases, as such equipment can perform very specific tasks employing cutting-edge technology, therefore limiting its versatility. However, there are devices employed in the laboratory that are not highly technological but still are very expensive. This is the case of many automated devices, which perform tasks that could easily be done by simple and low-cost devices like simple robotic arms. It can be anticipated that more laboratories will take advantage of this new reality as low-cost automation is very attractive for laboratories.

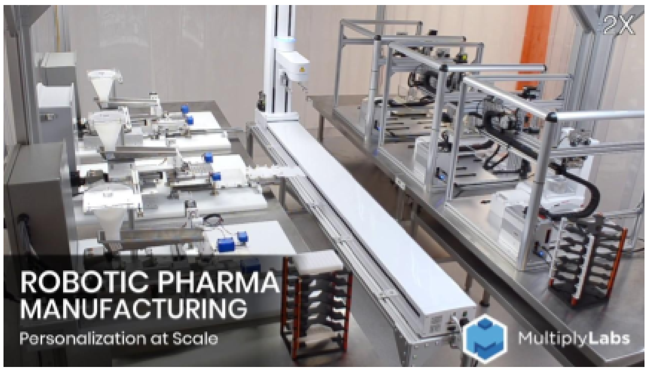


Figure 1.1: Robotic Factory

1.2 State-of-the-art

One of the examples of advanced automated solutions for laboratory usage is the Robotic Factory created by Multiply Labs shown in Figure 1.1. This complete robotic system was designed to prepare personalized medicine tailored to the patient's specific needs. The whole structure utilizes a robotic arm placed on the sliding rail, for the movement around other devices, to grab the corresponding tray and bring it to the filling station. Filling stations on their own are separate robots that fill the capsules with the needed drug in the exact prescribed dosage. After filling - the arm moves the tray into a weighing station, where the dosage contains of each capsule are verified. This process is then repeated for each new drug present in the patient's prescription and in the end, the fully custom pill is ready and personalized for the patient. Although this system is completely autonomous, it is highly specified only for this certain purpose and therefore, not versatile. The option for human-robot interaction is not utilized by this system as well.

Mahoro lab robot, Figure 1.2, is an example of a rather advanced and thus highly-priced system too. The robot employs two manipulators, with 7 joints each, to perform numerous laboratory tasks like samples taking, operating those samples, etc. The goal of the project was to develop a system, which will operate common laboratory equipment in a human-like manner, therefore

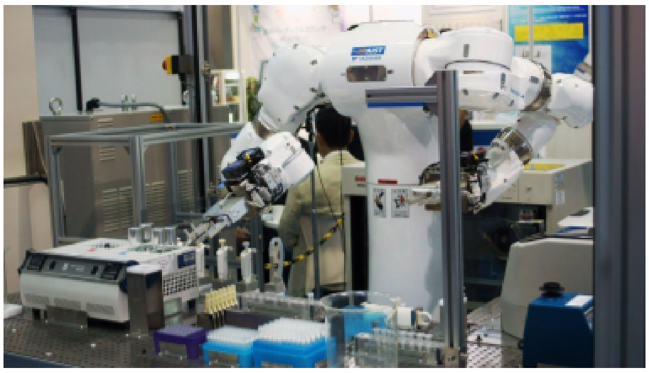


Figure 1.2: Mahoro Robot

eliminating the need for building complicated specified solutions to perform a single task, thus making the system highly versatile. The created system can perform given tasks faster and more precisely than a human being, however, leaving an option for human-robot interaction aside.

On the side of the lower-cost options, Figure 1.3 shows a developed solution at Nanyang Technological University. The solution is based upon the Denso VS-060 robotic arm and is operated in a structured environment. The system utilizes the custom-designed syringe pump and other printed 3D parts to perform handling of liquids, stirring of chemicals and a couple of other procedures, which keeps the overall cost on the relatively lower side and the versatility of the whole system high. Such a system gives an example, how cheap options can be used to automate laboratory processes. However, the system is bound to the pre-programmed locations of the devices and a human-robot interaction option is absent as well.

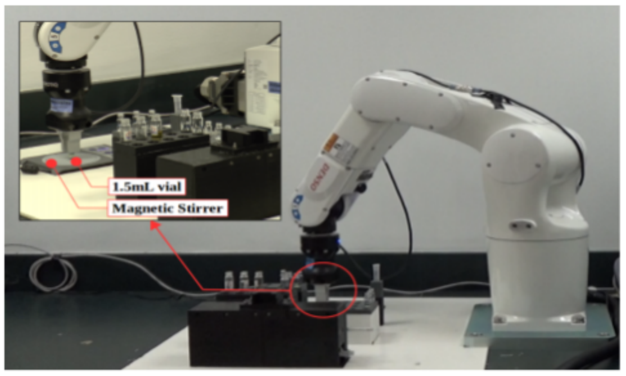


Figure 1.3: NTU Robot

1.3 Objectives and structure

The motivation for this thesis comes from the steady growth in the development of the low-cost devices and automated solutions that provide more and more compatibility with common laboratory instruments and processes as well as provide growth of degree of autonomy in those processes.

The goal of this bachelor thesis is to develop a human-centered automated solution using a collaborative robotic arm to perform a common and routine laboratory process to produce buffer solutions or other processes that would involve the same or similar operations. The algorithm of preparation of such solutions can be seen in Figure 1.4 and it consists of two steps - weighing and stirring & heating. As part of good laboratory practice, which is discussed in Chapter 3.2, each step of the process should be recorded in a report.

The main task of the weighing process is to ensure that the right amount of buffer or other reagents will be added into the solution. This process can be approached in several ways, and the option chosen in this thesis is presented and discussed later in chapter 5. The stirring & heating process, on the other hand, is quite straightforward. It consists of a series of consequent simple steps shown in Figure 1.5. In the beginning, the bottle should be placed onto the operating

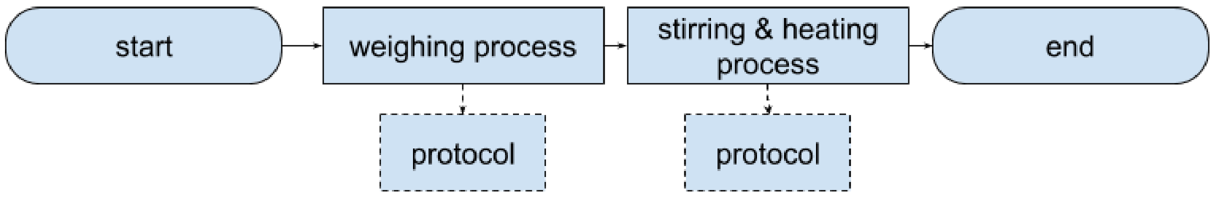


Figure 1.4: Solutions preparation

plate, after that the stir bar should be inserted into the bottle to perform the stirring. After the stirring and heating are both finished, the stir bar should be removed from the bottle. Finally, the bottle should be removed from the plate and closed with the bottle's lid.

The thesis begins with an abstract about the whole work done and then is divided into 7 chapters. The first one is the introduction to the thesis, which is followed by two chapters on the prerequisites used. In chapter 2, the theoretical basics about robotics and buffer solutions are explained to the reader. Then, in chapter 3, the materials and methods used during the course of the work will be shown and explained. The next chapter describes the development and designing of the toolsets for the robot's end-effector and evaluates them against the list of requirements. Chapter 5 explains how the process was solved on the software side. It provides the reader with the algorithms used during the process, shows the approach to the programming of the movements, explains the way the different output data were gathered and analyzed. Finally, the whole software part is evaluated for performance. The penultimate chapter of this thesis discusses the problems identified during software evaluation and presents possible solutions to these problems and some additional options for possible improvements. The thesis ends with a conclusion. In this part, the work is critically examined again and the outlook on the whole development process is given.

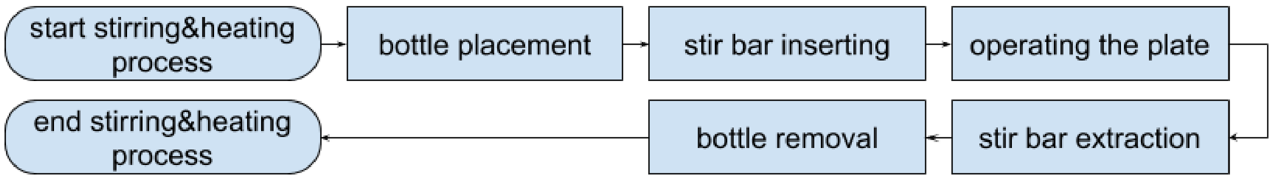


Figure 1.5: Stirring & heating process

2 Theroretical background

This background information includes the basics the robot itself functions upon and some simple nuances about buffer solutions.

2.1 Robotics theory

Pose and Orientation

A rigid body is completely described in space by its position and orientation - pose, with respect to a reference frame. The position of a point O on the rigid body with respect to the coordinate frame O–xyz is expressed by the relation:

$$\underline{o} = o_x \underline{x} + o_y \underline{y} + o_z \underline{z} \tag{2.1}$$

where o_x, o_y, o_z denote the components of the vector $\underline{o} \in R^3$

[15]

In order to describe the rigid body orientation, it is convenient to consider an orthonormal frame attached to the body and express its unit vectors with respect to the reference frame.[15] In order to compute the orientation of a body a rotation matrix can be used in combination with, for example, Euler angles for a minimal representation.

The changing of position is called translation, of orientation - rotation. Translation and rotation are represented together with a coordinate transformation. Therefore, a transformation of a point $\underline{p_b} \mapsto \underline{p_a}$ from reference frame b to the reference frame a can be defined as:

$$\tilde{\underline{p_a}} = \begin{bmatrix} \underline{p_a} \\ 1 \end{bmatrix} = T_b^a(R_b^a, \underline{r}) \cdot \tilde{\underline{p_b}} = \begin{bmatrix} R_b^a & \underline{r} \\ \underline{0^T} & 1 \end{bmatrix} \cdot \begin{bmatrix} \underline{p_b} \\ 1 \end{bmatrix}$$

(2.2)

[8] With rotational matrix $R_b^a \in R^{3\times3}$; translation vector $\underline{r}$; and T_b^a - homogeneous transformation matrix which can be generally defined as following:

$$T_b^a = \begin{bmatrix} R_b^a & \underline{r} \\ \underline{f^T} & w \end{bmatrix} \tag{2.3}$$

[reference] with $\underline{f^T}$ - perspective transformation, and w - scaling.[8]

Kinematics

The aim of direct kinematics is to compute the pose of the end-effector as a function of the joint variables - q.[15] It is described through:

$$T_E^0(\underline{q}) = \begin{bmatrix} R_E^0(\underline{q}) & \underline{r_E^0}(\underline{q}) \\ \underline{0}^T & 1 \end{bmatrix} = \begin{bmatrix} R(\underline{q}) & \underline{r}(\underline{q}) \\ \underline{0}^T & 1 \end{bmatrix} \tag{2.4}$$

[8] where $\underline{q}$ - generalized joint angle vector.

In connection with the direct kinematics, one also speaks of a manipulator-specific non-linear mapping of the joint or configuration space into the (Cartesian) world or, more clearly, into the work space - the region described by the origin of the end-effector frame.[8][15] Following, a direct kinematics equation can be written in a form:

$$\underline{w} = \underline{f}(\underline{q}); \quad \underline{w} = \begin{bmatrix} \underline{r} \\ \underline{\Omega} \end{bmatrix} \in R^W, \quad \underline{q} = \begin{bmatrix} q_1 \\ q_2 \\ \dots \\ q_N \end{bmatrix} \in R^N \tag{2.5}$$

where $\underline{w}$ - vector of world coordinates of the end-effector frame T_E^0; $\underline{r}$ - displacement vector, $\underline{\Omega}$ - orientation vector[[8]

The inverse kinematics problem consists of the determination of the joint variables corresponding to a given end-effector position and orientation. The solution to this problem is of fundamental importance in order to transform the motion specifications, assigned to the end-effector in the operational space, into the corresponding joint space motions that allow execution of the desired motion.[15] The inverse model can generally be described as the inverse of the direct model as:

$$\underline{q} = \underline{f^{-1}}(\underline{w}) = \underline{g}(\underline{w}); \quad \underline{w} \in R^W, \ \underline{q} \in R^N \tag{2.6}$$

where $\underline{g}$ is the inverse function of $\underline{f}$ [8] Out of (2.5) after switching we get:

$$\underline{f}(\underline{q}) - \underline{w} = \underline{0} \tag{2.7}$$

This nonlinear system of equations can be solved iteratively using the **Newton-Raphson** method:

$$\underline{q^{(v+1)}} := \underline{q^{(v)}} - J^{-1}(\underline{q^{(v)}})[\underline{f}(\underline{q^{(v)}}) - \underline{w}], \tag{2.8}$$

where v= 0,1,... the iteration index and $J = [\partial f_i / \partial q_j] \in R^{N \times N}$ the Jacobin matrix. For more details please refer to[8]

Differential Kinematics

The goal of the differential kinematics is to find the relationship between the joint velocities and the end-effector linear and angular velocities[15] The differential direct kinematics:

$$d\underline{w} = J(\underline{q})d\underline{q} \tag{2.9}$$

[8] For regular $J(\underline{q})$ the differential inverse kinematics can be defined as:

$$d\underline{q} = J^{-1}(\underline{q})d\underline{w} \tag{2.10}$$

[8]

Kinematic Singularities

The Jacobian J is, in general, a function of the configuration q; those configurations at which J is rank-deficient are termed kinematic singularities. To find the singularities of a manipulator is of great interest for the following reasons:[15]

1. Singularities represent configurations at which mobility of the structure is reduced, i.e., it is not possible to impose an arbitrary motion to the end-effector.
2. When the structure is at a singularity, infinite solutions to the inverse kinematics problem may exist.
3. In the neighbourhood of a singularity, small velocities in the operational space may cause large velocities in the joint space.

An important conclusion from these findings is that with regard to corresponding operations, the taking of singularities of manipulator should be avoided when planning movement or force operations.[8]

A kinematic singularity at q^* is mathematically described as:[8]

$$|J(\underline{q}^*)|_{\underline{q}\to\underline{q}^*} \longrightarrow 0, \qquad |J^{-1}(\underline{q}^*)|_{\underline{q}\to\underline{q}^*} \longrightarrow \infty.(2.11)$$

2.2 Buffer solutions

Definition

Almost every biological process is pH-dependent. A small change in pH[4] produces a large change in the rate of the process. For example cells and organisms maintain a specific and constant cytosolic pH, usually near pH 7, keeping biomolecules in their optimal ionic state. Constancy of pH is achieved primarily by biological buffers: mixtures of weak acids and their conjugate bases[4].

Buffers are aqueous systems that tend to resist changes in pH when small amounts of acid (H^+) or base (OH^-) are added. A buffer system consists of a weak acid (the proton donor) and its conjugate base (the proton acceptor). The pH is maintained due to an equilibrium established between weak acids and bases. The pH of the buffer system does change slightly when a small amount of H^+ or OH^- is added, but this change is very small compared with the pH change that would result if the same amount of H^+ or OH^- were added to pure water or to a solution of the salt of a strong acid and strong base[4].

Henderson-Hasselbach

The Henderson-Hasselbach equation can be used to calculate the ratio of acid and salt(conjugate base) that is needed to produce a buffer solution of desired pH. This equation is simply a useful way of restating the expression for the ionization constant of an acid[4]. For the ionization of a weak acid HA, the Henderson-Hasselbalch equation can be derived as follows:

$$K_a = \frac{[H^+][A^-]}{[HA]} \tag{2.12}$$

where K_a - acid dissociation constant; [H$^+$], [A$^-$], [HA] - concentrations. Equation[enter mark here] can be written as follows:

$$-\log[H^+] = -\log K_a - \log \frac{[HA]}{[A^-]} \tag{2.13}$$

Which leads to Henderson-Hasselbalch equation:

$$pH = pK_a + \log \frac{[A^-]}{[HA]} \tag{2.14}$$

[4]

Buffering capacity

Buffering capacity refers to the amount of acid or base you can add to the system without greatly affecting the pH. Buffering capacity(β) is defined as:

$$\beta = \frac{\Delta B}{\Delta pH} \tag{2.15}$$

where B = the mole equivalent of strong base or strong acid added and ΔpH is the resulting change in pH. Buffering capacity will be the greatest at the pK_a of the buffer.[4]

Choosing a buffer

There are no well defined criteria for choosing a buffer except that the pK_a should be close to the desired pH range ($\pm$1 pH unit) since the buffering capacity is greatest at the pK_a. Buffers should not participate in reactions that are being analyzed or otherwise affect the outcome of the experiment. For example, the buffer should not bind to other components in the mixture.

3 Materials & Methods

3.1 Equipment

Weighing

For the weighing process, in order to measure the correct amount of buffer Entris Analytical Balance scales by Sartorius shown on Figure 3.1(A) were used. The scales are equipped with internal and external calibration options, therefore they can be placed anywhere and can be adjusted directly on the workspace. The scales provide high precision weighing with up to 0.1mg readability and can weigh up to 220 grams. The weighing plate(88mm diameter) is surrounded by a chamber to limit the influence of the external factors on the weighing process. This chamber has the following dimensions - 230mm height and 173mm width. The output is performed through a high-contrast display on the front side of the scales. The scales have several built-in functions, out of which, Tar function is of primary attention. This function allows to calibrate the scales and pre-save the value of different containers in order to perform the weighing without taking the weight of those containers into account - the scales will automatically adjust the measuring point to zero when using a pre-saved container.

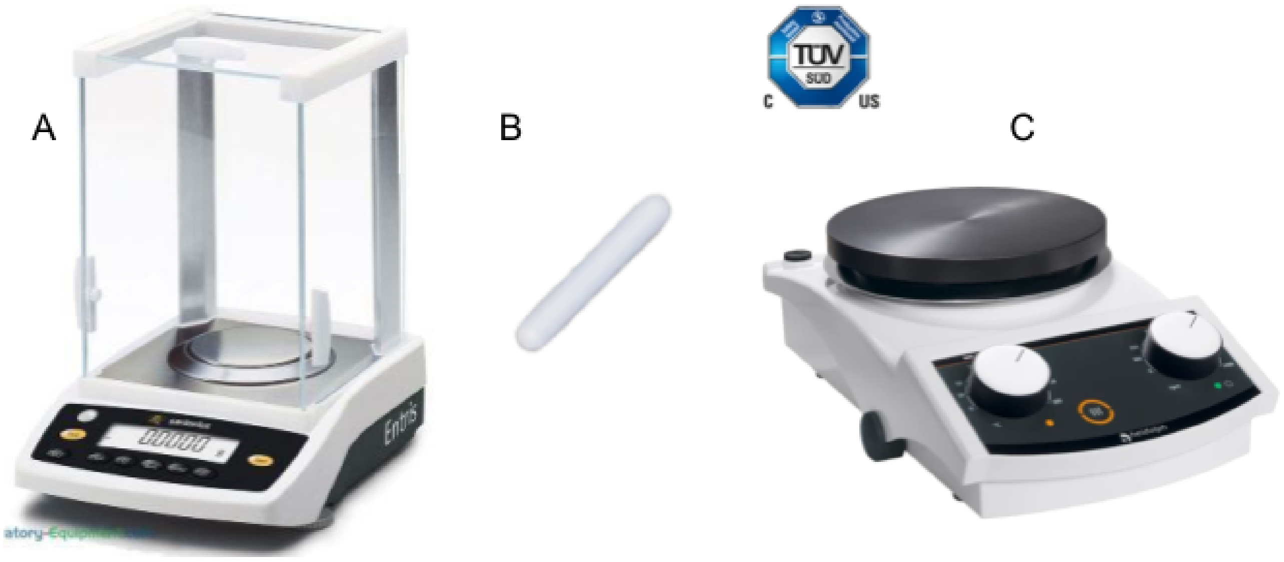
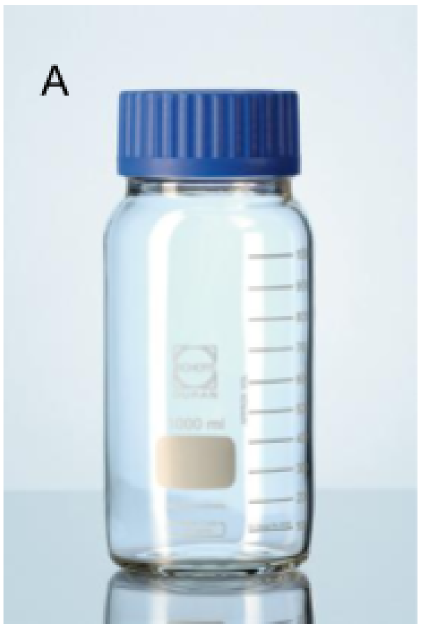


Figure 3.1: Equipment

Stirring & heating

The task of the magnetic stirrer is to heat the solution to the set temperature and to mix it with desired rotations per minute by adjusting with the corresponding dial. The plate used during the process is the Hei-Standard Magnetic Stirrer by Heidolph and it is demonstrated in Figure 3.2(B). This device allows to heat the plate up to 300 degrees Celsius and provides a rotation speed range up to 1400 rpm. The heating plate itself has a diameter of 145mm is made of silumin and covered with ceramic coating in order to provide chemical and scratch resistance. A built-in safety circuit hotplate will turn the device off when the temperature is 25 degrees Celsius over hotplate temperature to prevent any damage. The stirring is made by employing a rotating magnetic field to cause the stir bar immersed in a solution to spin very quickly, thus - stirring it. Such a stirring bar is shown in Figure 3.2(C), and during the course of the thesis such bars will be addressed as magnetic core, stirring core or in a similar fashion. There are two sizes of such cores available for the stirring process - 25mm and 50mm long. Both have similar diameter. The smaller bar is used for the solutions up to 500ml, the bigger one for 2000-5000ml. As a reservoir for a prepared solution a wide mouth laboratory bottle by GLS 80 Protect by Duran is used Figure 3.2(A). The bottle is scratch-, leak and shatter-resistance, has a capacity of 1000ml, height of 218mm and diameter of 101mm. This bottle was designed to operate under the temperature up to 140 degrees Celsius, however can withstand up to 500 degrees for short-time. The tweezers shown in Figure 3.2(B) are used to place the magnetic core into the bottle and remove it out of the bottle once the stirring and heating are finished. Tweezers itself are made of metal which does not interfere with the magnetic field on the stirrer. As for dimensions, they are 300mm long, which is enough to reach the core out of the bottle's bottom.



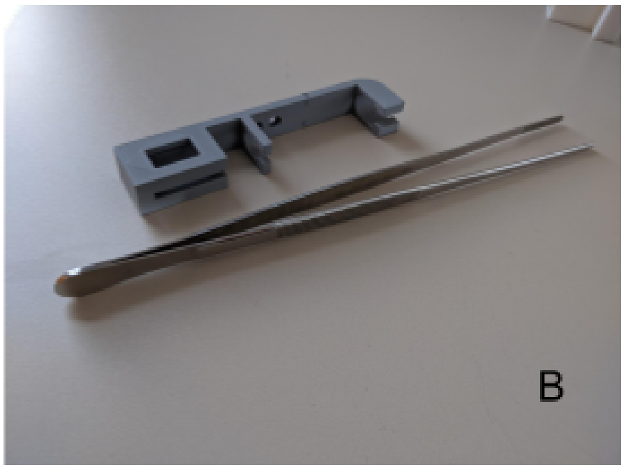


Figure 3.2: Equipment

3.2 Buffer & Protocol

Buffer

As a material for buffer solution preparation, the PBS tablets pH 7.4(for 500 ml) by PanReac AppliChem ITW Reagents were proposed.

The phosphate-buffered saline or PBS is a water-based salt solution containing disodium hydrogen phosphate(Na_2HPO_4), sodium chloride(NaCl), and, in some formulations, potassium chloride(KCl) and potassium dihydrogen phosphate(KH_2PO_4). PBS has many uses because it is isotonic and non-toxic to most cells. PBS has a pK_a between 6 and 8[4], highly soluble in aqueous solutions, and chemically stable.

The proposed PBS tablets are used for 500ml solutions and would provide a pH of 7.4 ± 0.05 to this volume of solution. The tablets consist of 2.7mM of KCl, 140mM of NaCl, and 10mM of Phosphate.

Protocol

A protocol is a pre-defined procedural method in the design and implementation of an experiment. As a part of the experimental process, protocols are created to standardize a laboratory method to ensure successful replication of results by others.[1] An important advantage of protocols is that they ease the assessment of the results through peer review. In addition to detailed procedures, equipment, and instruments, protocols will also contain study objectives, reasoning for experimental design, reasoning for chosen sample sizes, safety precautions, and how results were calculated and reported, including statistical analysis and any rules for pre-defining and documenting excluded data to avoid bias. A protocol may refer to the procedural methods of health organizations, commercial laboratories, manufacturing plants, etc. to ensure their activities are consistent to a specific standard, encouraging safe use and accurate results. Pre-defined protocols are an essential component of Good Laboratory Practice(GLP) regulations.

Protocols written for use by a specific laboratory may incorporate or reference standard operating procedures(SOP) - a set of step-by-step instructions compiled by an organization to help workers carry out complex routine operations, governing general practices required by the laboratory. As for our case, protocoling will include the description of the processes and the parameters those processes were executed with, the status of the process - whether an end result was successfully reached and several standard safety instructions. All of this will provide consistency in research and will ease the overviewing and gathering of the results.

3.3 Robot

3.3.1 Description

The platform which was used to perform the process is the robot called Panda by Franka Emika GmbH. Figure 3.3 shows the robot itself. The robot has 7 joints and the same number of degrees of freedom which provides high versatility of the whole system as a result. Figure

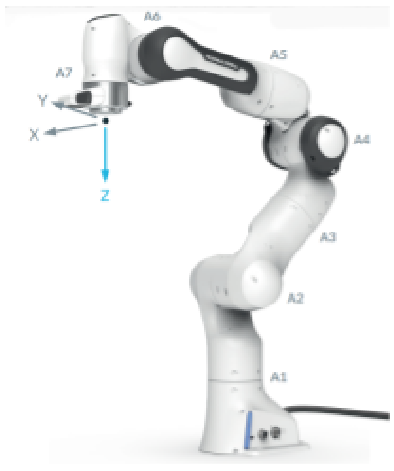


Figure 3.3: Panda Robot

3.4 [16] shows the reachable space for the end-effector flange in mm(side-view on the left, top-view on the right), which might provide some idea of robot dimensions themselves. The robot arm can handle a payload up to 3kg, which is more than enough for the course of the whole experimental process. There are torque sensors built in all of the 7 joints which are used for collision detection and can be triggered by exceeding the expected value and will abort the process and stop the robot arm safely by locking the safety bolts of all 7 axes. The arm is connected via a connection cable to the Control, which is connected to the power supply, providing necessary power input for the whole system. The external activation device, called User-Stop, is connected at the base of the arm and serves as a fail-switch - when pressed during the execution of a process it will abort it and stop the robot safely.

The hand of the robot, marked in Figure 3.3(A7), has two parallel fingertips, which can be used to mount and operate custom-built parts by adjusting the width of the fingertips. The maximum continuous force of the hand is 70N. There are two buttons on the hand of the robot, which when both pressed simultaneously will enable guiding mode of the arm, meaning the whole arm can be easily moved into the necessary position while being in this mode. Right on top of the arm there is a control panel, which provides an ability to operate the robot directly from the arm. There are status lights on both sides of the base which take on the corresponding color, similar to a traffic light. These status lights will only flash during bootup, during other processes the lights will glow continuously in the corresponding color of the status Panda is in. The detailed information about those status indicators can be found in the user manual.[16]

3.3.2 Manipulator prerequisites

The main requirement for all designed parts is to give the entire system and, subsequently, the entire process the highest possible degree of autonomy. In order to achieve this, the robot manipulator is equipped with the part called tool holder, at both fingers of the hand on the end-effector(Figure 3.5(A)) that allows the usage of self-developed parts. This tool is built of plastic, has a quadratic-shape slot of dimensions 5x5mm, going through the whole body of the tool. On the one side of the slot of this tool holder has a small, trapezoid shaped part that is

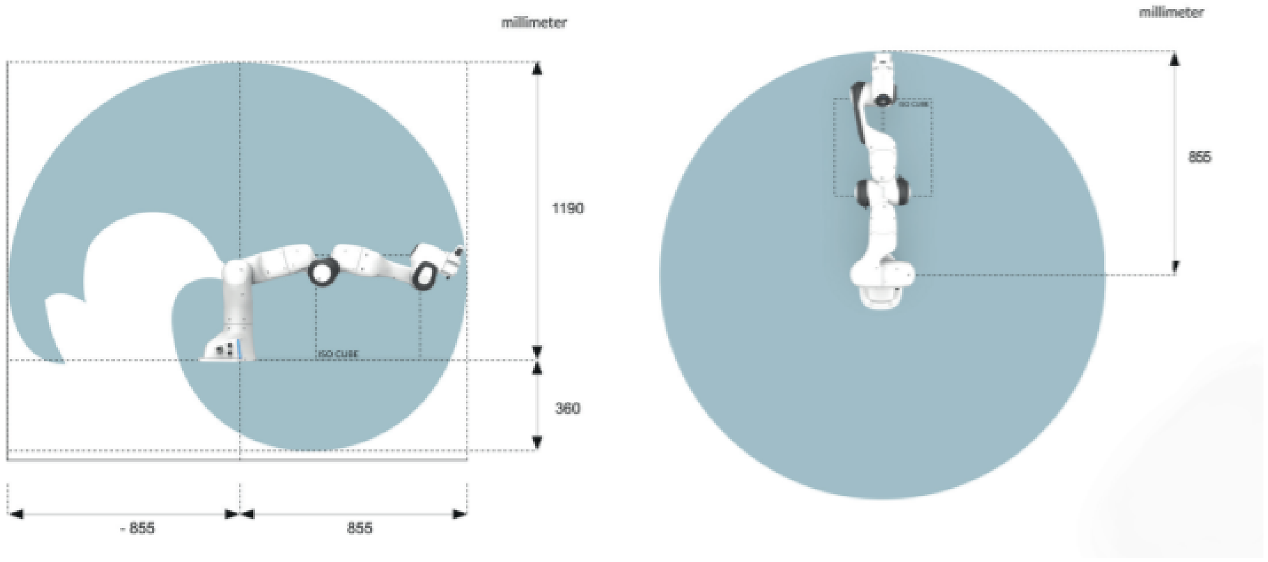


Figure 3.4: Reachable space

being held and pushed down by a small metal spring. This slot within the tool was designed to hold the adapter (Figure 3.6) in it. The hook of this adapter has a quadratic shape of 4,7x4,7mm dimensions that fits this slot of the tool holder perfectly. The hook goes through the slot and is locked in it by the trapezoid part on the spring that slides into the small cut onto the hook's body. Figure 3.6(B) demonstrates such a tool holder with the inserted adapter with a designed tool on it.

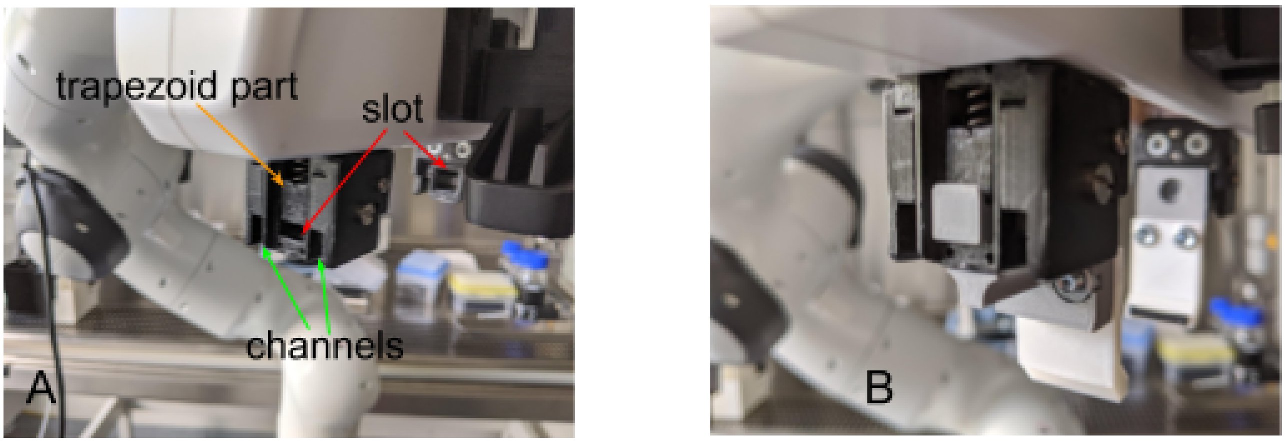


Figure 3.5: Tool holder

On both sides of the slot the tool has two tiny channels 5mm wide. Inside each of those two channels there is a small button installed. Applying the pressure on these buttons the spring goes up, releasing the hook of the adapter out of the slot. This functionality was used later in all of the designed parts simply because it makes it possible to create a so-called changing-station. Such changing-stations, when properly designed, can be used to change different end-effector

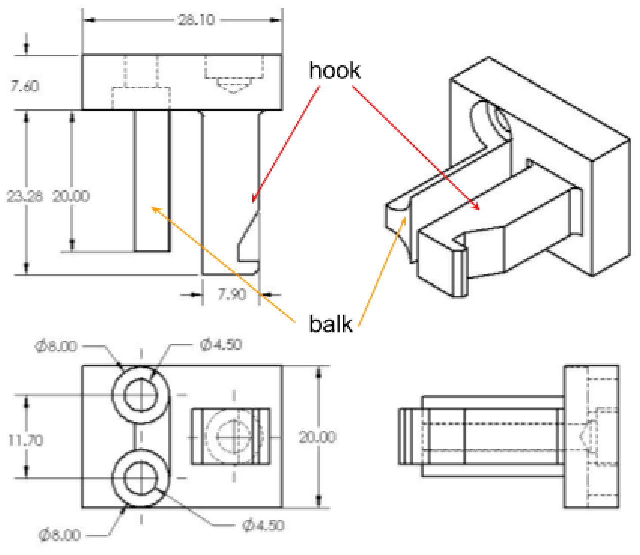


Figure 3.6: Adapter

tools without human interaction - which corresponds with the main goal of the thesis - creating an autonomous system for laboratory usage.

The adapter is made out of thermoplastic and has the dimensions shown on the Figure 3.6 in mm. Right under the previously described hook, the adapter has a slightly shorter balk that goes parallel to the hook. The main purpose of this balk - to provide additional support for the adapter and to overtake part of the pressure applied onto the hook, when being used. This purpose is achieved through contact with the bottom of the tool holder.

Right under the balk, the adapter has two circular deepenings of diameter 4,5mm which were designed to hold two 3mm nuts in it. In the middle of each of those deepenings there is a hole ofdimension that goes through the whole body of the adapter. Those holes were specifically designed to fit the 3mm bots. This makes it possible to connect and fix almost anything to the body of the adapter. Some other important features of the adapter is its production simplicity and standardisation. This adapter can be simply printed on any 3D printer anywhere in a short(depends on your printer) period of time. As for standardisation - the usage of such adapters provides the ability to develop tools for a platform that can be changed quickly in case of need - simply unscrew the tools you place on it and change to the other. And, another far more important ability - it provides an additional “fail switch”, meaning that it improves the chances not to get a far more sophisticated part broken in case something malfunctions wrong during the experiment.

3.3.3 Communication with the robot

As the Figure 3.7 demonstrates, the robot is connected via a connection cable to the Control, which is connected to the PC itself via an Ethernet cable, which allows access to Panda via the FCI programming interface. In our case, this PC is a NUC mini PC by Intel which serves the

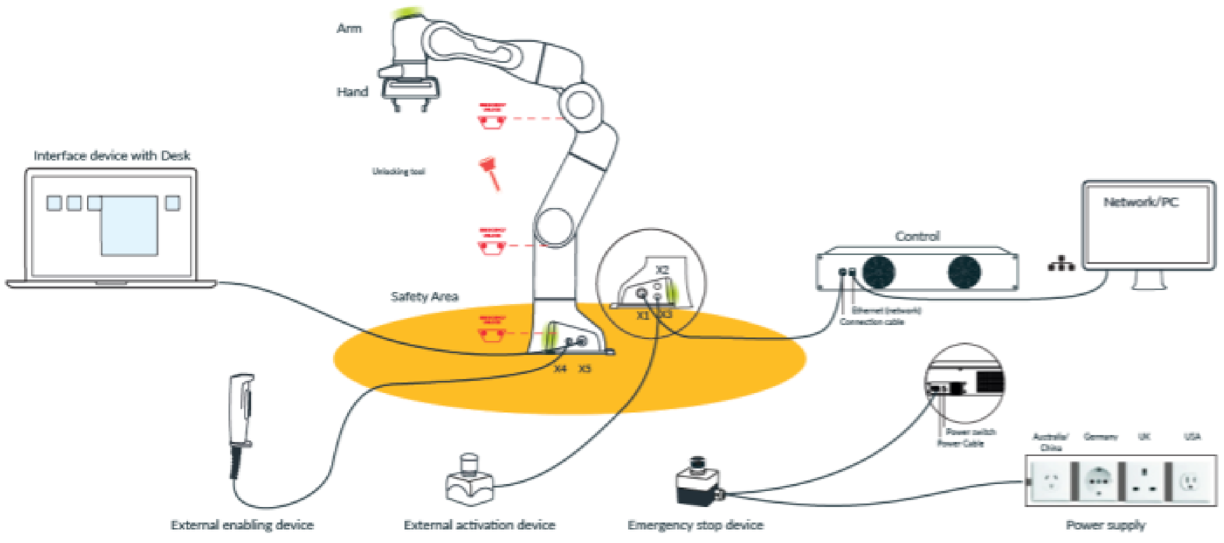


Figure 3.7: Communication with the robot

role of a control server by establishing connection to the robot. Therefore, in order to establish the connection with the robot and operate it, one should directly connect their own PC with the NUC via the local W-LAN network. The NUC serves the role of a connection hub for the camera mounted on top of the robot as well.

Desk

Desk is a web application developed by Franka Emika which allows users to program the robot to perform desired processes. As Desk is a web application therefore no pre-installing to the user's PC is required, only a modern web browser is needed. In order to establish the connection, one has to connect to the NUC, described previously, via W-LAN and enter the robot's URL into the browser. Desk allows you to create Tasks. Tasks are program sequences and consist of a chronological sequence of Apps. Apps are the building blocks of a Task and describe the basic capabilities of the robot. The Apps within a Task need to be parameterized, meaning that parameters such as poses, orientations, speeds etc. need to be set.[16]

Desk interface is displayed in Figure 3.8. The timeline(1) the area in which you can line up your apps via drag-and-drop in order to program your task. In the task area(2), the programmed tasks are stored. The app area(3) includes all of the installed apps available for programming via dragging-and-dropping them onto the timeline. In the sidebar(4) the information is shown about which guiding mode Panda is in. Thereunder are situated important notices on Panda's status, such as, for example, whether the external activation switch is on or off, or whether there is an error.

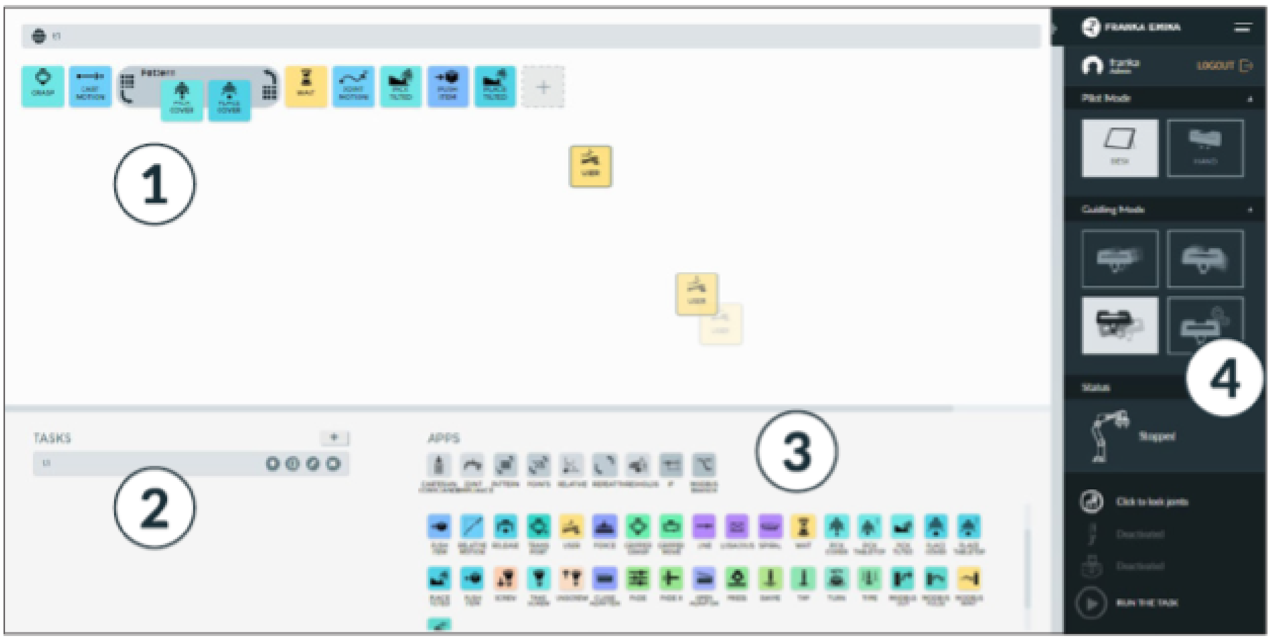


Figure 3.8: Desk interface

Mios

Mios is software built around a core module that handles all low-level control and functionalities. The structure of mios is shown in figure 1.[10] By using libfranka, Panda's open-source C++ interface, the user can send real-time control values, that can be set and changed on even the smallest scale imaginable - up to setting the exact values of the joint or cartesian coordinates. The interface provides an ability to control values with different interfaces like gravity & friction compensated joint level torque commands, joint position or velocity, cartesian pose or velocity commands. At the same time the measurements of joint data(position, velocity), various collision and contact information, and other measurements can be accessed. The user can get access to the robot model library which provides forward kinematics of all robot joints, the jacobian matrix of all robot joint, dynamics data, etc.[7]

In order to use mios, it should be accessed on the FCI-PC(Franka Control Interface - PC) which is located on the server connected directly with the robot's Control. In our case, as described previously, it is our NUC controller. The main prerequisite however for the usage of the mios is that user PC has to be operated on Linux OS with a real time kernel. Firstly, the connection with the server must be established via local W-LAN. Then the connection with the Control is done via Linux terminal, and then, the mios can be launched and used for programming.

Mios uses a knowledge base that is based on a mongodb database, which stores data using json format. To connect to this database via a GUI a robo3t is used.

The mios database contains following collections:[reference mios manual]

- environment - to store objects and locations
- reference_frames - contains reference frames

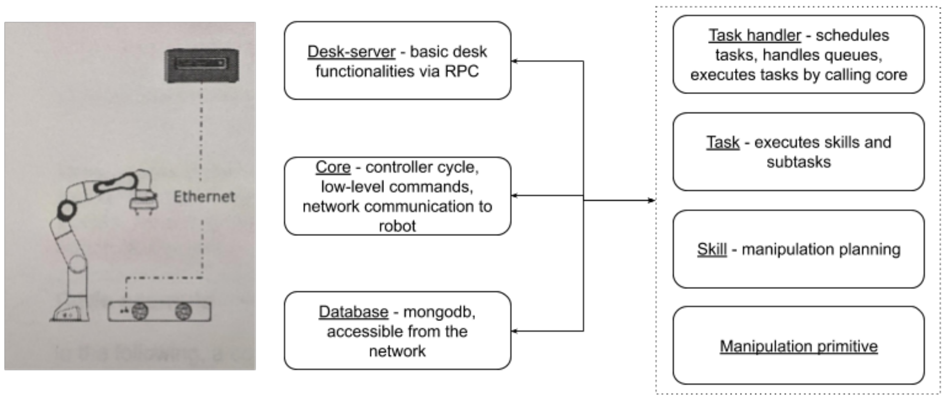

Ethernet
Desk-server - basic desk functionalities via RPC
Core - controller cycle, low-level commands, network communication to robot
Database - mongodb, accessible from the network
Task handler - schedules tasks, handles queues, executes tasks by calling core
Task - executes skills and subtasks
Skill - manipulation planning
Manipulation primitive

Figure 3.9: Mios structure

- parameters - all persistent parameters are saved here
- skills - the collection of all skill descriptions
- tasks - contains all task descriptions

Remote Procedure Call (RPC) interface allows using the methods with desired parameters saved in the database in order to execute custom processes. The methods are pre-saved, pre-programmed and described sets of tasks. A task contains a set of skills and subtasks, inserted into the mongodb database into the tasks collection, that is executed in a specific order and according to specified conditions. Tasks may be queued and terminated externally at any time.[reference mios manual] A skill is an abstract representation of a manipulation policy that is defined according to the purposes of the process. Generally, a skill consists of a set of manipulation primitives connected in a graph structure. The transitions are triggered by events. The skill description contains the basic blocks of a skill, which has to be inserted into the mongodb database into the skill collection.

The algorithm of how to create and use your own robot process in mios is shown in Figure 3.10. In order to create a task or a skill the user has to create a task/skill description in json format and save it within robo3t interface. After it's done, a task/skill can be created via calling a create_task()/create_skill() function out of meta_coding.py script. Running this function creates a simple C++ file with a corresponding header file. Note however, the script only creates the basic functions for the new task and integrates it into mios. The rest of the programming has to be done in the next managing step in a simple trial-and-error fashion. The programmed task can be tested by running the rpc_call function out of rpc_client.py skript.

One of the biggest advantages of mios is that it is open-source and is coded in C++. This allows integrating the tasks, programmed within mios, into other custom scripts, like for example, user interface, recognition and calibration methods, and many other even more sophisticated concepts, which provides a high level of versatility during the development and execution process.

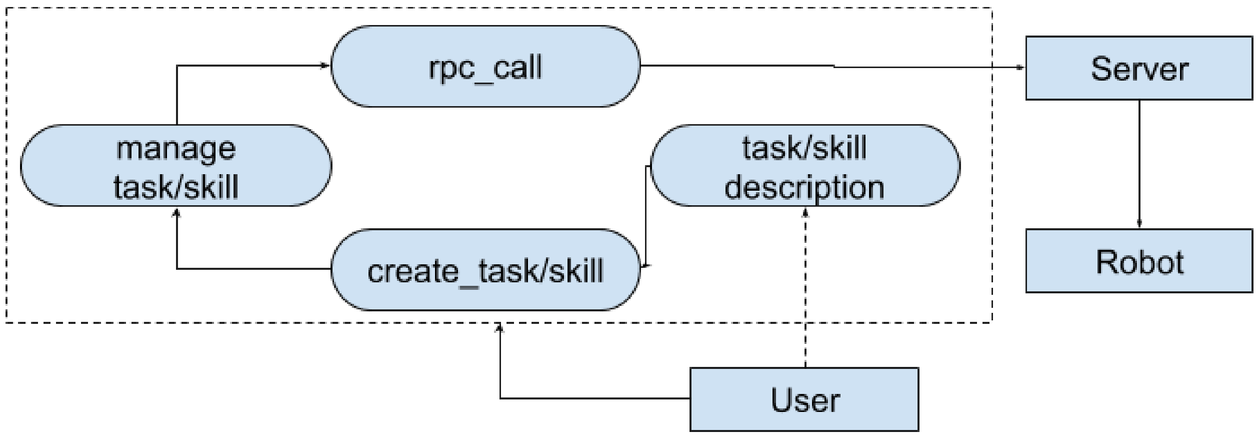


Figure 3.10: Program creation algorithm

Camera

The camera used in the process is RealSense by Intel. It is mounted right on top of the robot hand and is directed in the same direction as the end-effector. The camera is connected with the previously described NUC via USB cable, which serves the role of a server. A connection to the camera may be established through the W-LAN or the direct cable connection. The process of connecting to the camera via W-LAN is the following(Figure 3.11). Firstly, the user has to connect to the server through the local W-LAN network. Then, the video-streaming should be started on the server by setting the width, height and framerate of the streaming video in the terminal. There are multiple available picture quality settings - up to full hd resolution. The IP address of the receiving device should be entered on the server via terminal as well. After it's done, the streaming will start and the user can connect to it through the terminal on the receiving device. This simple terminal connection allows us to take a picture with the camera or to analyze the live stream video at any time desired. This grants the ability to use the camera as an input option.

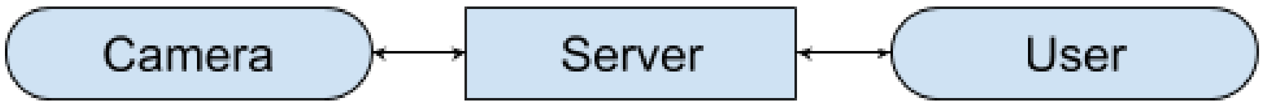


Figure 3.11: Camera connection

4 Development of the end-effector tools

This chapter provides information about the custom-designed CAD parts in Solidworks, that later were printed on the 3D printer. The general information about the toolsets themselves is given and reasoning how those were developed is provided in this chapter.

4.1 Small manipulations toolset

Small manipulations are those that will occur either with smaller objects, that are lighter and as a result, are easier to handle(smaller containers), or the manipulations that are directed towards operating the laboratory devices: turning the devices on or off, setting the right values on the control panels, opening or closing some of the elements. In our case those devices are the stirring/heating plate and the laboratory scales, used in weighing and stirring heating processes, which were described at the very beginning of the thesis in chapter 1.3.

Taking those purposes into account, the list of requirements for such a part can be stated as follows:

- system integration: should operate on the base of the tool holder(chapter 3.3.2) with the usage of the c

  orresponding adapter
- form: small and light
- high precision of the system: should provide firm contact with the operational surface
- easy-to-handle: should be easily replaced in case of construction failure or other malfunction

And the requirements for the changing-station can be defined like this:

- system integration: should fit the tool holder(chapter 3.3.2) and the developed part properly
- form: smaller - better, stiff enough not to break
- reproduction of the process: should be able to provide the same quality of the results over and over again
- easy-to-handle: must be able to be placed in any possible orientation - both horizontal and vertical

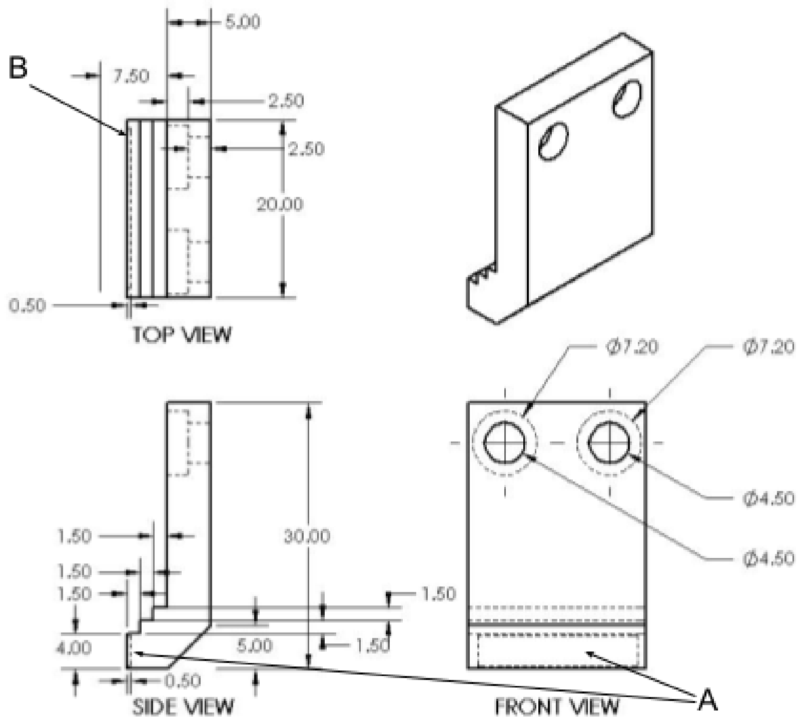


Figure 4.1: Small manipulations toolset

Model development and construction

The most important task of this toolset, as pretty much of the whole thesis, is to create the most autonomous system possible. Therefore, the usage of the adapter(chapter 3.3.2) in the combination with its end-effector slot was immediately considered the best possible option for such a purpose. As for the tool itself, a simple L-shape extension was considered the potential concept solution due to its simplicity. For the precision as well as for mobility purposes it was decided to keep the size of the tool relatively small. You can see the exact dimensions in mm in Figure 4.1. On the front side there are two 2.5mm deepenings with a diameter of 7.2mm and in the middle of them - a hole that goes through the entire body of the part with a diameter of 4.5mm. The purpose of those elements - to fit two 3mm Allen bolts, that will be screwed into the respective nuts placed in the adapter. As a result - this made our L-shape easily replaceable by the need. To provide firm contact with the operational surfaces a thin layer(approximately 1mm) of rubber was glued to the flat end of the L-shape tool(point A).

As for the changing station(Figure 4.2), first and foremost it had to provide the ability to insert and take the part off fully automatically. To do so, the station has two rails(walls) 2.5mm thick placed on the 17mm distance from one another(point A). These dimensions allow applying pressure onto the buttons within the channels of the tool holder, therefore lifting the trapezoid part on the spring, releasing the space within the slot to insert or to remove the piece placed on the adapter. The overall dimensions of these rails make the whole inserting and removing process easier and provide good enough stiffness to the walls, making the usage of some additional support between those walls unnecessary.

The container(point B) at the end fits the L-shape holding it in place properly. To provide the additional stiffness point to the L-shape tool while in the container, the small corner piece was placed on the sidewall of the container(point C). This corner helps to avoid any back-to-front wobbles, while the L-shape is being inserted into the tool holder on the robot manipulator.

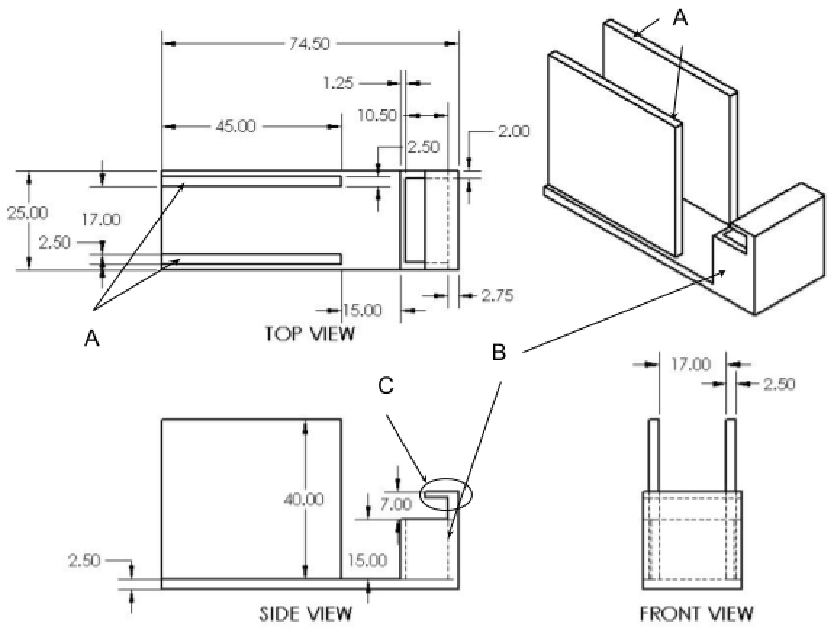


Figure 4.2: Changing station

The overall dimensions of the changing-station were designed with the thought to give the ability to place and use it in any possible orientation. Vertical and any side positioning will work out just as fine as simple horizontal placement. However, an upside-down option won't work I'm afraid.

The development process was not flawless of course. In the first prototype the rubber piece at the end of the L-shape tool rubbed against the walls of the changing station and, as a result, came off pretty easily. To solve this, a small deepening of 0.5mm was placed(Figure 4.1, point B) to glue the rubber piece deeper within and to prevent any frictions with the outer walls of the holding container. Finding the right dimensions for the container that will fit the L-shape tool properly took several tries as well. The final result fulfilled all the necessary requirements for it.

4.2 Bottle manipulations toolset

Under bottle manipulations, the kind of manipulations is meant that are executed on our main reservoir described in chapter 3.1. Placing onto and removing the bottle from the heating/stirring plate, closing and opening the bottle are those kinds of manipulations performed as was stated in chapter 1.3.

Considering this kind of manipulations, the list of requirements for this toolset consists of:

- system integration: should operate on the base of the tool holder(chapter 3.3.2), preferably with the usage of the corresponding adapter
- high precision of the system: should provide the firmest contact with the surface of the tank cap
- form: big enough to perform the task using the tool holder, but preferable to keep it as small as possible
- robust system performance: should open/close the bottle without using additional parts due to the potential high temperature of the solution inside
- reproduction of the process: repeatability of the output results and their quality
- easy-to-handle: easily replaceable

Changing station for such toolset has to fulfill the following requirements:

- system integration: should fit the tool holder and the developed part properly without damaging the part
- form: smaller - better
- reproduction of the process: should be able to provide the same quality of the results over and over again
- easy-to-handle: must be able to be placed in any possible orientation - both horizontal and vertical

Model development and construction

In order to make this part of the experiment autonomous the usage of anything else but the adapter(chapter 3.3.2) in combination with the tool holder was discarded immediately, due to the simple fact that the usage of something else might damage the autonomy of the whole process. As a result building this toolset, fulfilling this autonomy criteria, immediately creates several problems:

1. The distance between the tool holders on the robot manipulator is only slightly larger than the width of the bottle cap. This creates a challenge to fit the set of tools for bottle transportation in between.
2. The weight of the bottle creates too much pressure onto the standard adapter and as a result it breaks easily, meaning it cannot guarantee the needed consistency in process reproduction and even more, it creates a potential risk of damaging the container during such malfunction of the adapter.

To solve the first problem the tool had to be designed to step out in the opposite direction of the gap between the tool holders and minimize the limitation of the gap within as much as possible. However, such an approach creates a lever onto the adapter, which results in an even bigger force being applied to it, making the second problem even a bigger deal. Therefore, to create a suitable solution, the stock adapter had to be redesigned as well.

Regarding the tool itself, it is made out of thermoplastic, has a circle shape with dimensions only slightly exceeding the cap dimensions, that was going to clasp a cap of the bottle, was

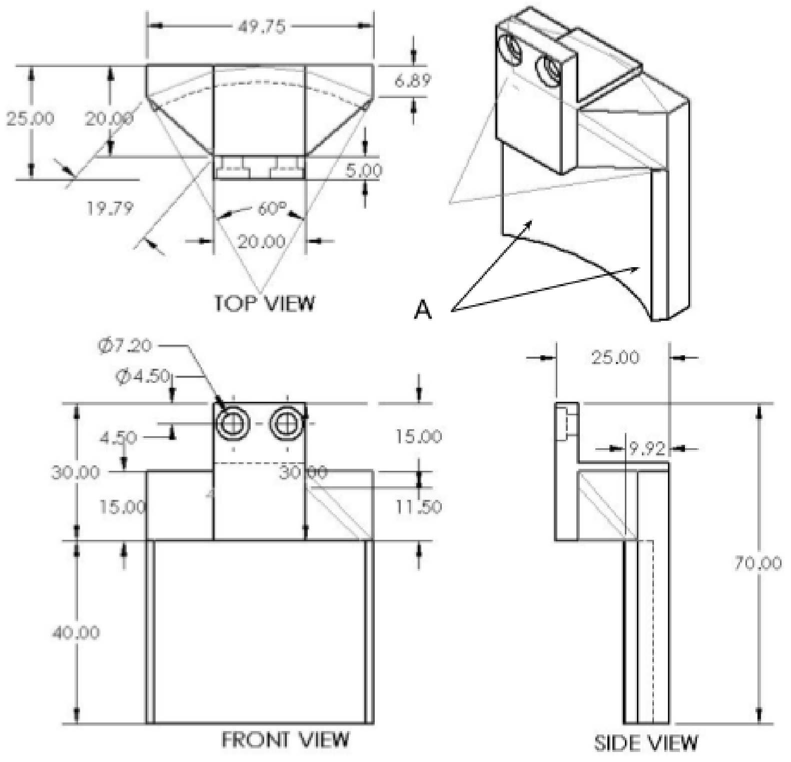


Figure 4.3: Bottle manipulations toolset

considered a first option. However, it created a problem during grasping - it tilted, therefore losing the useful grasping surface and sliding of the cap occasionally as a result. To solve this, the circle-shape was replaced by a sector-shaped form factor, which meant having smaller dimensions as well. The tool covers the 60-degree sector, which is big enough to create enough grasping force onto the bottle to provide the firm grip. You can see the exact dimensions of this tool in mm in Figure 4.3. On the top side there are two 2.5mm deepenings with a diameter of 7.2mm and in the middle of them - a hole that goes through the entire top of the part with a diameter of 4.5mm. The purpose of those elements - to fit two 3mm Allen bolts, that will be screwed into the respective nuts placed in the adapter, making this part easily replaceable by the need. The inner part of this tool(point A) has a shape of the arc and is 40mm height, which repeats the corresponding shape and height of the tank cap. In order to provide a firm grip with the cap surface, a thick, approximately 3.5-4.5 mm layer of rubber was glued to the whole inner surface(along with point A) of this tool.

As for the changing station(Figure 4.4), it had to provide the ability to insert and take the part fully automatically. Just like the changing station used for small manipulations toolset, the station has two rails(walls) 2.5mm thick placed on the 17mm distance from one another(point A), that allows by applying pressure onto the buttons within the channels of the tool holder release the adapter out of the tool holder. Unlike the changing station used previously, caused by the dimensions of the tool these rails had to be made much bigger - 82 mm, which meant the dramatic decrease in their stiffness. To solve this, two horizontal plates were placed in between those walls(point B).

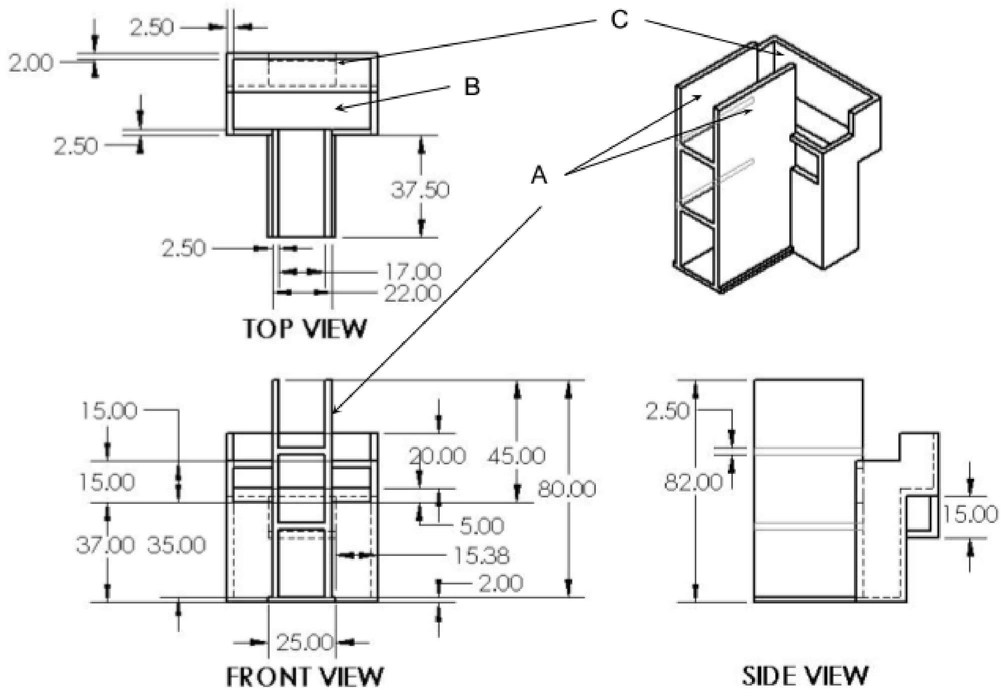


Figure 4.4: Changing station

The container(point B) at the end fits the part holding it in place properly. To provide the additional stiffness point to our part while in the container, the small wall was placed on the back of the container(point C). This corner helps to avoid any back-to-front wobbles, while the part is being inserted into or removed from the tool holder on the robot manipulator. As with the changing station for small manipulations, the changing station for tank manipulations toolset was designed to allow the usage in any possible orientation.

As mentioned before, the stock form factor of the adapter had a major flaw in this process and had to be redesigned to withstand the pressure applied to it. However, to avoid the reduction of the autonomy of the whole process caused by this matter, the simple approach to upgrade the adapter(Figure 4.5) for our purposes, rather than completely developing an alternative one, was taken. As a result, a balk under the hook of the original adapter was made much thicker, which provided contact with the surface of the tool used for tank transportation(Figure 4.3). This allowed to redirect the pressure and made the whole adapter stiffer as well, resulting in solving the stated problem.

4.3 Core inserting & extraction toolset

The primary purpose for the core inserting and extraction toolset is to operate one of the two magnetic elements that are used by the stirring plate, as stated and described in chapter 3.1.

Considering the prerequisites of this part of the experiment - the dimensions of the bottle, the sizes of both magnetic cores and the technical characteristics of the plate itself, the list of requirements for a toolset like this can be stated as follows:

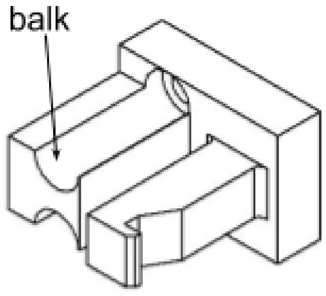


Figure 4.5: Custom adapter

- system integration: should operate on the base of the tool holder(chapter 3.3.2) with the usage of the corresponding adapter
- robust system behavior: should be able to perform core extraction out of high-temperature solutions - up to 300 degrees Celsius(maximum temperature reached by the heating plate)
- form factor: as light and as compact as possible, able to fit through the tank opening
- high precision of the system: should provide a firm grip with both sizes of the magnetic cores to safely extract it out of the solution
- easy-to-handle: should be easily replaced in case of construction failure or technical malfunction during the process

As for the requirements for the corresponding changing-station, they can be stated as:

- system integration: should fit the tool holder, the developed part and all the components placed within properly
- from: should not extend the dimensions of the parts a lot to keep as much workspace free as possible
- reproduction of the process: should be able to provide the same quality of the results over and over again
- easy-to-handle: must be able to be placed in any possible orientation - both horizontal and vertical

Model development and construction

The biggest challenge developing this whole toolset was to make it capable of withstanding high temperatures, as all of the tools were printed out on the 3D printer out of thermoplastic and high temperatures might simply melt the plastic. This meant that the direct grasping of the core out of the tank with a thermoplastic toolset is impossible. To deal with this, the option to use the stock tweezers, described in chapter 3.1, was chosen. However, directly mounting the tweezers to the end-effector via splitting it into parts and simply screwing it to the robot's body would jeopardize the autonomy completely due to the need for human interaction to mount or

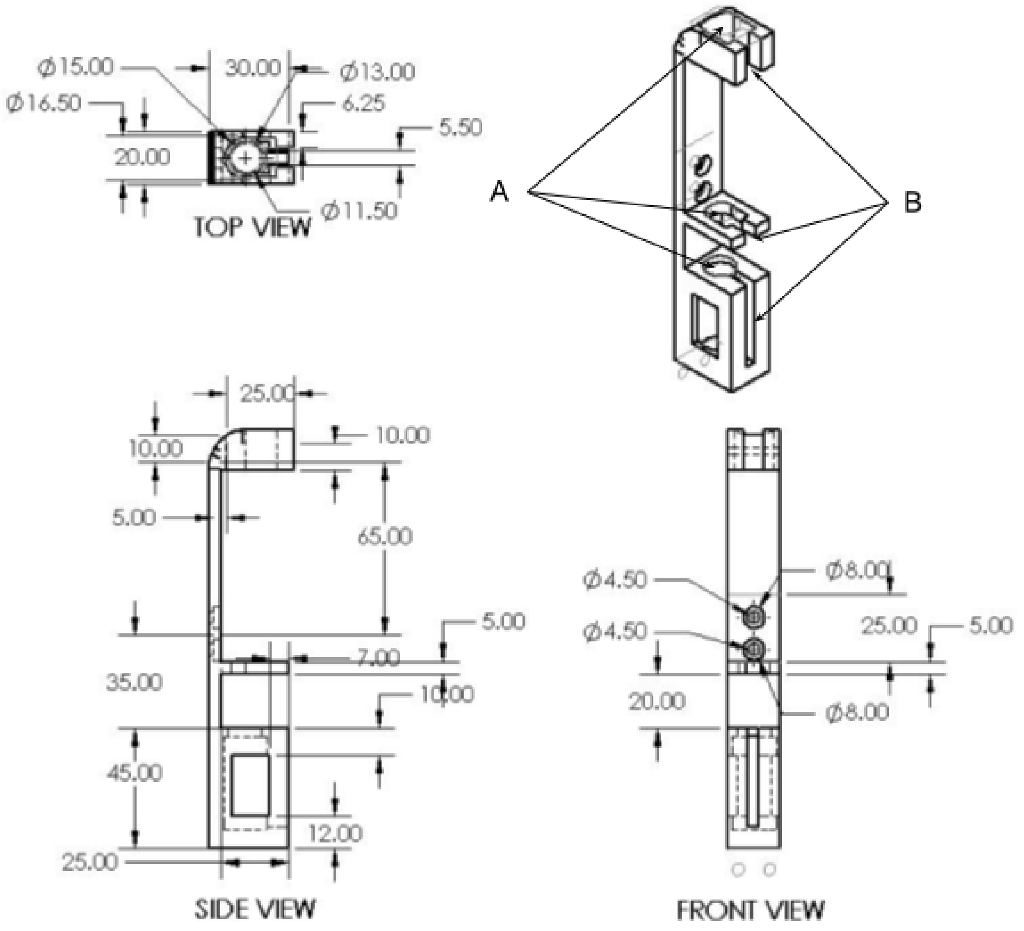


Figure 4.6: Tweezers holder

remove those tweezers. All of this drove to a conclusion - to develop a specific tool set, that will allow the robot to use those stock tweezers for the whole process. Analyzing the build characteristics and dimensions of the tweezers the decision was made to place them on the one side of the end-effector and use some tool on the other side, that will make it possible to apply pressure on the tweezers via changing the width between robot calipers, hence opening or closing the tweezers making it possible to use them in a human-like way.

After trying numerous prototypes, which failed to deliver the necessary results due to various reasons, the following tweezers holder was developed and built fully out of thermoplastic(Figure 4.6), on the same figure the exact dimensions of this part are shown in mm. The form factor was caused by the nature of the tweezers - various widths in different parts.

The main idea is the following - the tweezers are being inserted into their holder, while being closed and squeezed together, through the openings(Points A or points B alternatively). Those openings have various widths, the one on the very top is the widest and one on the bottom is the smallest, which corresponds with the form factor of the tweezers themselves. After being inserted through those openings, the tweezers are rotated at 90 degrees and then opened. When being opened tweezers are being held in place by applying the pressure on the walls of the same openings(points A) and the matter of the width differences, which prevents the tweezers from

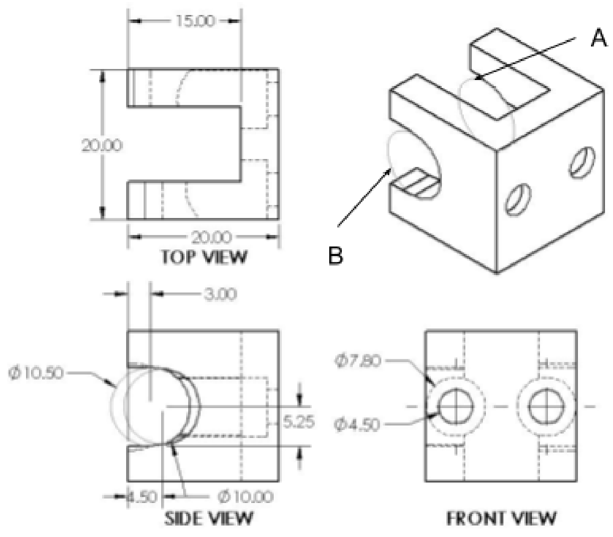


Figure 4.7: Tweezers manipulator

sliding out. On the front side of the holder there are two 2.5mm deepenings with a diameter of 7.2mm and in the middle of them - a hole that goes through the entire top of the part with a diameter of 4.5mm. The purpose of those elements is the same as in previous toolsets - to fit two 3 mm Allen bolts to bind everything together with the stock adapter(chapter 3.3.2).

After dealing with the challenge of holding the tweezers in place, the other side of the robot's gripper had to be equipped with the right tool for applying the pressure on the tweezers. After trying several prototypes out, the compact form of this part was chosen with the following dimensions in mm(Figure 4.7). The compact form factor of this part helps to prevent the flaws of using thermoplastic as a built material, namely - its high flexibility under pressure. This counterpressure problem arises if the part is too big and when it's being used to close the tweezers, the tweezers simply buckle big parts with the counterforce and simply stay open. As the tweezers holder, the second gripper part has two circular openings with different dimensions. Opening A is bigger than opening B, which allows us to use the tweezers built to our goals. As the parts before, this one has two 2.5mm deepenings with a diameter of 7.2mm and in the middle of them - a hole that goes through the entire top of the part with a diameter of 4.5mm on the front side to fit 3mm Allen bolts.

The changing stations(Figure 4.8 & 4.9) for this toolset were designed with the same thoughts as the other similar changing-stations described before. Both stations have two rails(walls) 2.5mm thick placed on the 17mm distance from one another for taking the part out or sliding it into the tool holder, granting the option of performing tools changing automatically. The walls of the changing stations were made 20mm high(points A), which allows us to place in every possible orientation - horizontal and vertical.

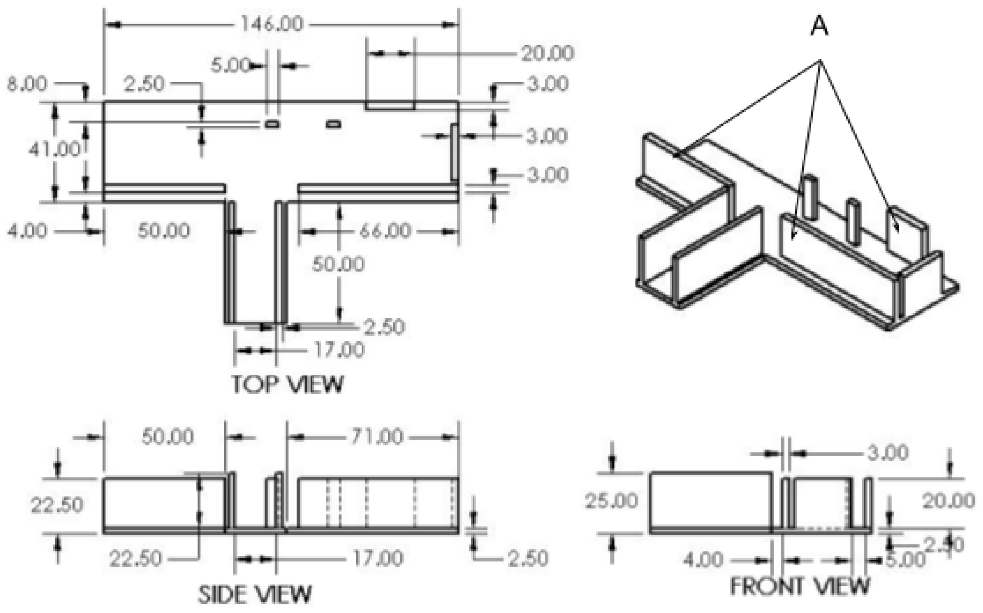


Figure 4.8: Tweezers holder changing station

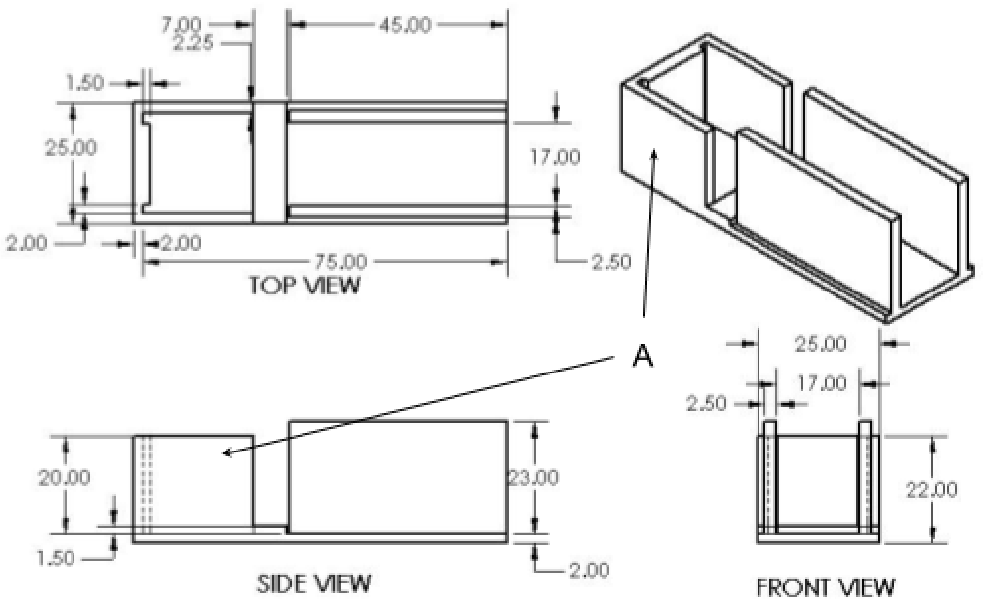


Figure 4.9: Tweezers manipulator changing station

5 Development of the Process Software

The pursued goals of the process software are straightforward - to develop a software solution that will complete all of the objectives stated in chapter 1.3, making the whole process as autonomous as possible, thereby as safe as possible for the human being, meanwhile protocolling every step of the process execution for better labor practice and granting the human-robot cooperation, when necessary, for best possible end-results.

5.1 Control software

This part of the developed software was directed towards programming every needed mechanical movement of the robot for the process completion.

5.1.1 Process algorithm

The process algorithm describes the thoughts and the approach taken to create a software solution to the entire process. As the corresponding figures shown in chapter 1.3, namely those to the buffer solution preparation and stirring & heating, and as the problem was stated, our entire process consists of two main steps - weighing and stirring and heating. While the stirring and heating process is straightforward on its own, the weighing process can be approached in multiple ways. Figure 5.1 shows the algorithm of the approach taken.

Before the entire process can begin, the user should enter the desired values for the amount of buffer reagent, that will be placed into the prepared solution, via a user interface(chapter 5.3). After this is done, the main process starts. The first step is for the user to place the buffer reagent onto the scales, meanwhile, the robot moves into the preprogrammed location to take the picture of the scales' output display. After the picture is taken, we move to the next step - analysis & interaction. In this step, the robot would analyze the picture taken previously with the help of the recognition program described later on in chapter 5.2.3. In the next step, the

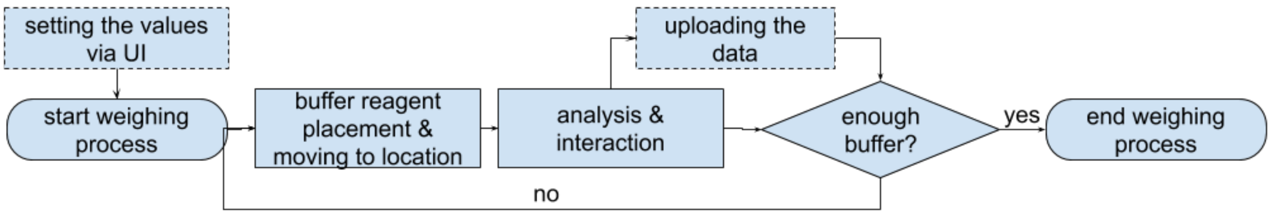


Figure 5.1: Process algorithm

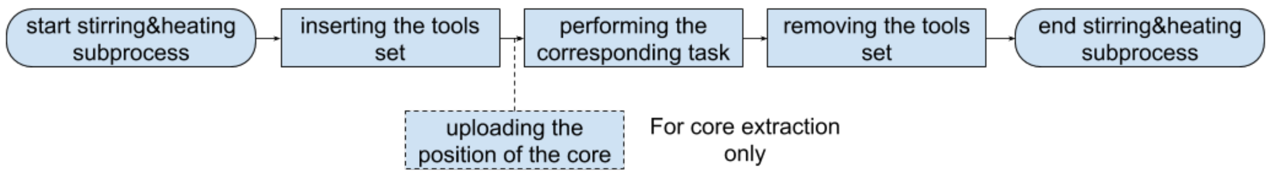

start stirring&heating subprocess
inserting the tools set
performing the corresponding task
removing the tools set
end stirring&heating subprocess
uploading the position of the core
For core extraction only

Figure 5.2: Stirring/heating sub-processes

output of the program will be uploaded to the robot and the robot awaits the user interaction to confirm or deny the correctness of the buffer reagent amount placed onto the scale compared with the initially entered value through the user interface.

Finally, the user has to toggle the robot's manipulator slightly in the downward direction to see the confirmation through the triple nodding in the up-to-down manner, which finalizes the process. If the robot nodes in the left-to-right fashion - this means that the amount of buffer reagent placed on the scales differs from the input data, thereby the entire process will be repeated until the correct amount of the buffer reagent is placed. Each of the steps will be protocoled as well for the later analysis. In the stirring and heating process, the main task is to reproduce all the actions via robot entirely. The approach taken here does not differ much from the one shown in chapter 1.3. There is only one difference - before the whole process starts the user inputs the necessary values via User Interface.

When the process is launched the series of the sub-processes are being executed one after another. These sub-processes are bottle placement/removal, core inserting/extraction and operating the stirring/heating plate. Figure 5.2 shows the logic behind almost every sub-process. The only sub-process that uses a completely different approach is str&heat_plate, which will be described later on. The core extraction is a slight exception as well, however, the only difference between the rest is shown with the dotted rectangle.

Each of the four sub-processes begins with the insertion of the right toolsets designed for this specific sub-process and described in 4.2. Straight after, the main task of each is performed. Those tasks are:

- to place the bottle onto the heating plate - for tank placement
- to insert the mixing magnetic core into the bottle - for core inserting
- to remove and close the bottle with the cap - for tank removal and
- to remove the magnetic core out of the bottle - for core extraction

However, before the task of core extraction can be performed, the information about core orientation, found with the core position detection program, described later on in chapter 5.2.2, is uploaded to reach the correct position.

The algorithm for the remaining sub-process - str&heat_plate is displayed on Figure 5.3.

Before this sub-process can be launched, the values for setting the right temperature, rpm and timer are given via the User Interface(chapter 5.3) and uploaded from there into the sub-process. Inserting and removing the toolsets are performed in the same manner as for the other sub-processes. Setting user values step is to set the user input data on the corresponding dials

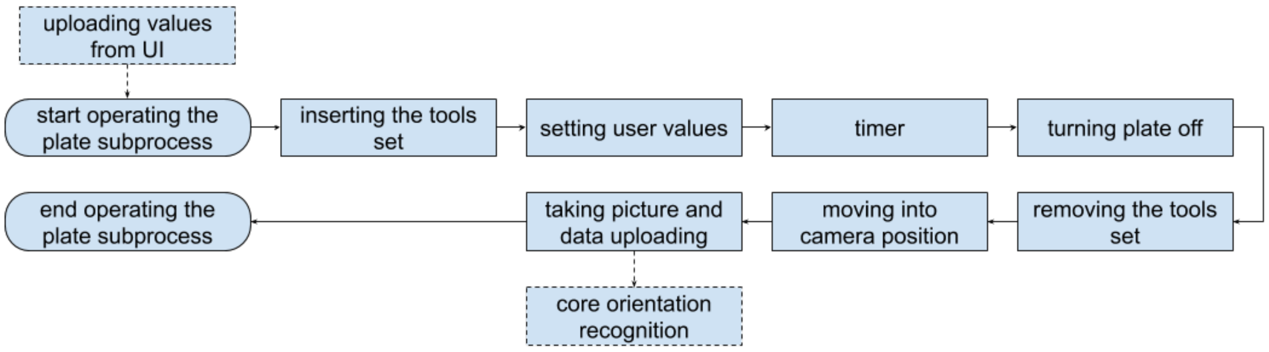

uploading values from UI
start operating the plate subprocess
inserting the tools set
setting user values
timer
turning plate off
removing the tools set
moving into camera position
taking picture and data uploading
end operating the plate subprocess
core orientation recognition

Figure 5.3: Plate sub-process

of the heating plate. During the timer step, the robot will simply wait the given amount of time in the pre-programmed location. This timer corresponds with the desired time the stirring and heating to be performed. After the timer runs out, the robot turns the plate off by setting the temperature and rpm dial back to zero.

When the toolset for this sub-process was removed, the robot moves the camera mounted on the end-effector to the set location to take the picture of the mixing core in the upcoming step. When the picture is taken it is uploaded to the core position detection program(chapter 5.2.2) for analysis. The program will analyze the data and upload the necessary response to the core extraction sub-process.

5.1.2 Desk solution

In order to create a working model, the whole process was programmed step-by-step based on the algorithms from the previous chapter within the Desk environment described in chapter 3. In Figure 5.4 the highlighted apps are the ones used during the entire programming process here.

Joint Motion app allows to control the exact motion of the robot's joints from one point to another due to the usage of the joint space coordinates within this app. Caused by limitations of the workspace and the fact that the whole workspace is filled with the devices this app was chosen to program all of the motions of the robot to reassure the avoidance of any collisions with the other objects in the workspace. Users can set the speed and acceleration in % for this motion.

During the programming the following groups of speed and acceleration were created:

- 25% speed and 25% acceleration for precise operations
- 50% speed and 35% acceleration for movement in close proximity to the objects on the workspace
- 75% speed and 50% acceleration for general movements in between

Precise operations include inserting/removing the toolsets, core inserting/removal, and operating the stirring the plate. Such operations require high accuracy to guarantee the correctness

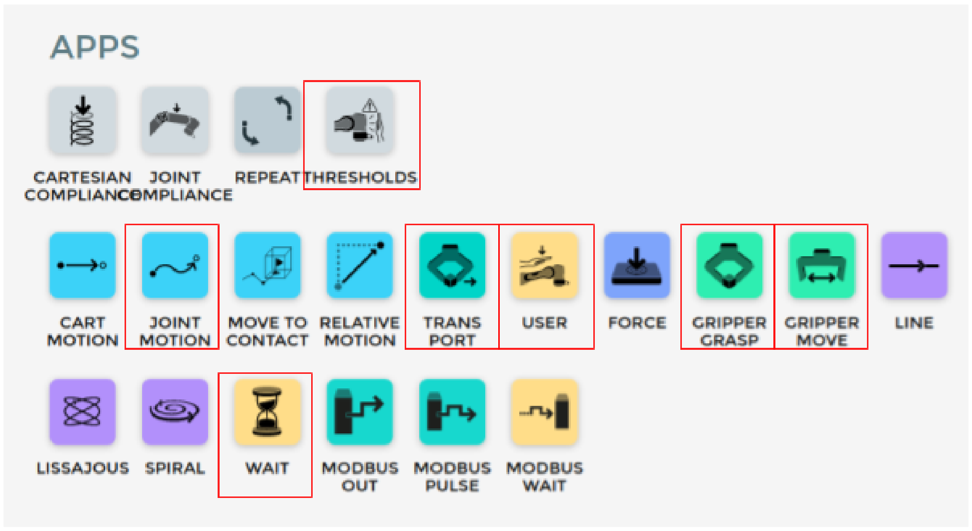


Figure 5.4: Desk applications

of the execution and to avoid any parts breakdown in case of collision. Movement in the close proximity to the object is self-explanatory, occurs for example when moving from one changing station described in chapter 4.2 to another, therefore in case of foul movement the choice of those speed and acceleration values is going to minimize the damage to the surroundings General movements are those that are performed relatively remote from the objects, where the chance of collision with the objects is minimal on its own. Joint motion is performed from one saved point to another, thereby the whole process was programmed by the teachsave method - meaning the robot was moved to the desired location, the start and endpoints were saved and then to check the correctness of the operation the program was launched. Through the complex combinations of such points, the sequences were created to automatize the inserting/removal of the toolsets sub-processes completely, with the usage of thresholds(up next), as well as to rotate the dials on the plate, during the core inserting/removing subprocesses and to reach the necessary locations to place the camera for the scales display and core position recognition.

Setting **Thresholds** allows us to reduce the robot's perceptivity to the outer distortions to collisions by setting the threshold values for forces or torques. This app proved itself useful when there was the need for a robot to contact a certain object, like when the toolset for the subprocess is being inserted into or removed from the robot's manipulator. Setting the thresholds for these sub-processes was vital, due to the nature of how the changing station operates(chapter 4.2). In the course of the program the following values were set to the app:

- 35 Newtons for every x,y,z coordinate for the tool insertion or removal
- 50 Newtons for every x,y,z coordinate when the tank was being removed or put back in its holder on the table(Figure 5.5)

Such values were chosen through a simple trial error approach.

Gripper **Grasp** and **Transport** apps were used for transportation of the used objects - the bottle, its cap and magnetic stirring core. Before the grasping of the objects, the end-effector with the right toolsets(chapter 4.2) has to be calibrated. After it's done the grasping is performed by setting the width of the gripper to grasp, the speed the gripper moves to grasp an object, and setting the grasping force.

The necessary width parameters were found through tryouts directly on the objects. After such parameters, needed to provide a firm grip, were found, they were saved into the app. The speed of the gripper movement was set to 80 mm/s to guarantee that the necessary width would be achieved without being violated by the restoring force. The grasping force was defined by the use cases, 65 Newtons to grasp the bottle, caused by its big overall mass, and 35 Newtons to grasp the mixing core, which was enough to overtake the restoring force of the tweezers used for this purpose and to provide firm grip of the core itself.

Transport app is very similar to the Joint motion in its nature, the same parameters can be set with only one additional case - the mass of the transported object. Therefore, in order to provide a smooth movement and to reduce the risk of the object to slide off the speed and acceleration were set to 25%.

After placing the transported object to the destination point the Gripper Move app was used to release the object. The movement speed of the gripper during the usage of this app was set to 50 mm/s to avoid unnecessary rocking of the transported object. This same app was used to grasp the dials of the heating plate, it provides much more gentle grasp, so the choice to use this option instead of the grasp was obvious to operate the device. Gripper movement speed during was set to 35 mm/s to guarantee the most gentle approach as possible.

The last two apps used during the course of programming were **Wait** and **User**. Wait task was used to demonstrate the timer sub-process and is performed by the same principle as a common timer. User app was used to show the moments when a human has to interact with the robot in the weighing process. The app is set by choosing the joint that is to be toggled by the user and the value for the force needed to be applied onto this joint in order to care for the rest of the process.

Although the desk solution provides an example of a fully automated process solution, it is still rather a proof-of-work of all of the components then a perfect solution. Each of the movements is performed in a pre-saved manner and unfortunately is irresponsible to any of the custom-built software options. Therefore, no customization of the process is possible in an on-the-go manner - when the user simply sets the desired values for the process and the process is performed with them.

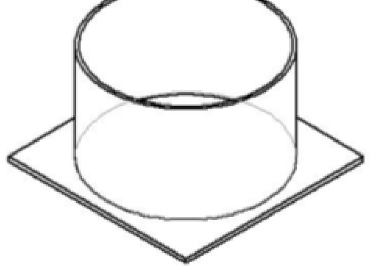


Figure 5.5: Bottle holder

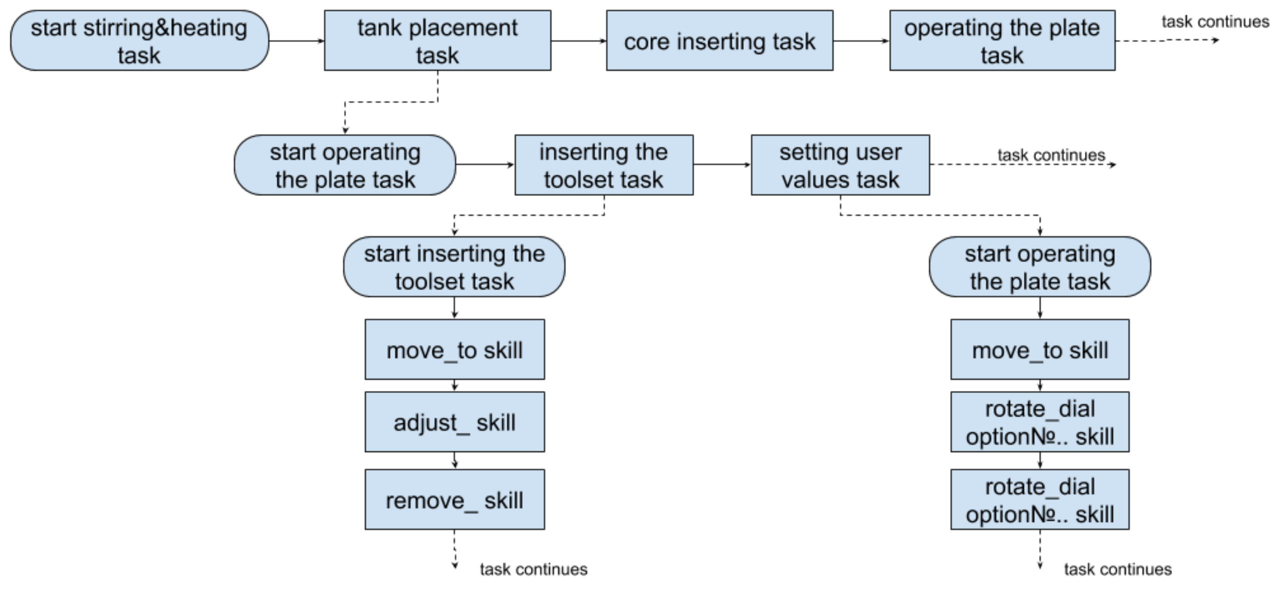

start stirring&heating task
tank placement task
core inserting task
operating the plate task
task continues
start operating the plate task
inserting the toolset task
setting user values task
task continues
start inserting the toolset task
move_to skill
adjust_ skill
remove_ skill
task continues
start operating the plate task
move_to skill
rotate_dial option№.. skill
rotate_dial option№.. skill
task continues

Figure 5.6: Stirring&heating process - mios

5.1.3 Precision improvement

Taking limitations within the Desk environment into consideration, like inaccessibility to the custom-built software, inability for the user to control the exact geometrical parameters of the robot(like joint angles or cartesian coordinates) or impossibility to choose one of pre-saved locations or positions directly during the process execution creates a necessity to use mios software described in chapter 3 to program the process and thereby improve the general autonomy. However, mios and Desk differ from one another, the algorithm for mios process implementation remains unchanged from the one described 5.1.1 and will be used to program the whole process in mios as well. The only difference is that every sub-process and each step will be replaced with the tasks or skills with the same names. Those tasks will have the same or similar purposes as the corresponding sub-processes and algorithm steps in the Desk variant.

As a first step in the development, all of the used objects have to be saved into the database by teach_object() function. This function allows saving the pose both in cartesian and joint space for the specified object. Those objects can be every possible equipment part, like dials of the plate(chapter 3.1), locations of the robot arm for taking input pictures, positions of the changing-stations with the toolsets and so on. This will allow us to program the movement of the robot into precise locations. Other functions available within mios library, like grasp_object, move_gripper and others can be utilized to replace the corresponding apps used in the Desk programming option described previously.

Figure 5.6 shows the algorithm of the concept of how to approach the creation of a solution within mios to the stirring&heating process. The process will be performed as a sequence of custom-developed tasks. Each of those tasks corresponds with every sub-process programmed within Desk in their final goal.

As it was described in chapter 3, some sub-tasks for specific operations will consist of skills, which will be used to control the simple movements of the robot and combined together into the task, which will describe more a fulfilled step of the process. For example, inserting the toolset task directly corresponds with a sub-step of the process in which a toolset for specific tasks will be mounted on the robot's hand. Such a task consists only of the manipulator movements from one point to another and therefore can be described by a sequence of skills like it is shown in figure 1. The move_to skill will describe the movement of the robot arm to the changing station, previously saved by teach_object(). After that, the manipulator will adjust its position to this changing station and will insert the needed toolset as a part of adjust_ skill. The correct position for tool insertion is pre-saved by teach_object() as well. Remove_ skill will perform all the needed manipulations to remove the part from the station. Such manipulations are pre-saved in the database in the same way too. These methods can be utilized by similar tasks of the whole process as well.

When it comes to setting of the input values, chosen in the user interface, like in case of operating the plate task, there are several options to deliver correct results. First one is simply pre-save numerous options into the database via teach_object, and when the input values are set, choose the corresponding position out of the database. In case of operating the plate task, those positions will be different pre-saved locations of the dials for different temperature or rpm values. The second option will require direct manipulations of coordinates, which then will be used within skills to move the manipulator correctly. Again, like in the case of rotate_dial skills this would grant higher precision in choosing the right values, despite the higher complication in the realisation.

Yet another interesting option in mios is the ability to create virtual walls to exclude the region from entering by the robot completely, or to arrange robots setting in a preferred way, while being within the region limited with those walls. This method can be adopted to create safety zones around specified objects to assure that nothing will be damaged by an accidental collision with the robot, thus improving the total safety.

Concluding, the mios provides an option to influence the robots coordinates, thus making it possible to perform each task with the highest precision possible. The ability to choose a value out of the database in on-the-go fashion provides the necessary options to combine the input values, out every step of the process, with the corresponding robot's manipulations, therefore improving the total autonomy of the whole system.

5.2 Camera input software

5.2.1 General approach

Just as all the other actions performed during this work, further steps to analyze the output of the scales were directed towards creating an autonomous solution, that will require minimal additional equipment upon the existing system. As a result, an option to connect the scales with an external controller was withdrawn and as well as due to the simple reason that such a controller would be directed towards a single purpose only, therefore limiting the autonomy of the whole system in the future. All this resulted in a simple choice of the solution - to use the camera mounted on the robot(chapter 3) as a method for collecting, analyzing and transferring the output data shown on the scales display. This approach showed itself useful when it came to detecting the magnetic core position after performing the stirring process(chapter 5.1.1) too. After running this stirring process several times in a row it was clear that this core will stop in an unpredictable way, creating a problem. If the extraction of the core was pre-programmed with its horizontal position, for example, the majority of different tilted orientations within a big variety of the angles, with respect to the initial horizontal positioning, could be handled purely in a mechanical way. Meaning that when the tweezers were squeezed together to extract the core, these angles will be corrected and handled by the applied force through the tweezers. The only problem in the case of horizontal pose extraction would be if the core stops in a vertical orientation, or similar orientation close to vertical. In such cases, the tweezers would simply not be able to reach the core in order to apply force to correct the orientation. If there is no guarantee that the core can be extracted, the process cannot be described as autonomous, due to the necessity of human interaction in such cases. Therefore, in order to solve this problem the camera may provide the way to gather the necessary input data to make autonomous decisions about this magnetic core position at the end of the experiment, and as a result, fit the tweezers in the right way to extract the core.

Due to the capability of the robot to repeat its movement in the same way over and over again and because the robot can stop in the same location to use the camera, the simple approach of taking the picture in the specific locations, to analyze those pictures later, was taken. Considering this important robot's ability, other ways of using the camera as data input were considered as overkill for such specified purposes. Therefore, through multiple testing and tryouts the necessary locations for camera positioning, for taking pictures that would be good enough to be analyzed, were found and stored to create the desired repeatability of the whole process.

5.2.2 Core position detection

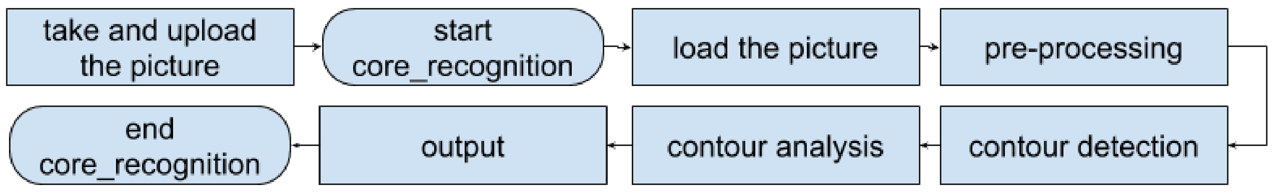


Figure 5.7: Core detection algorithm

The task of this program is to analyze the picture of the mixing core within the bottle. The picture is taken on the mounted camera in the before pre-programmed location. As stated above, this will exclude any possible fail-cases during the core extraction process, thereby improving the autonomy of the whole process dramatically. In order to make this program cross-platform, it was developed in Python with the usage of OpenCV - an open-source library aimed at real-time computer vision. Making this whole system cross-platform allows us to integrate it into almost any possible use-case scenario.

In Figure 5.7 the flow chart of the whole program is shown.

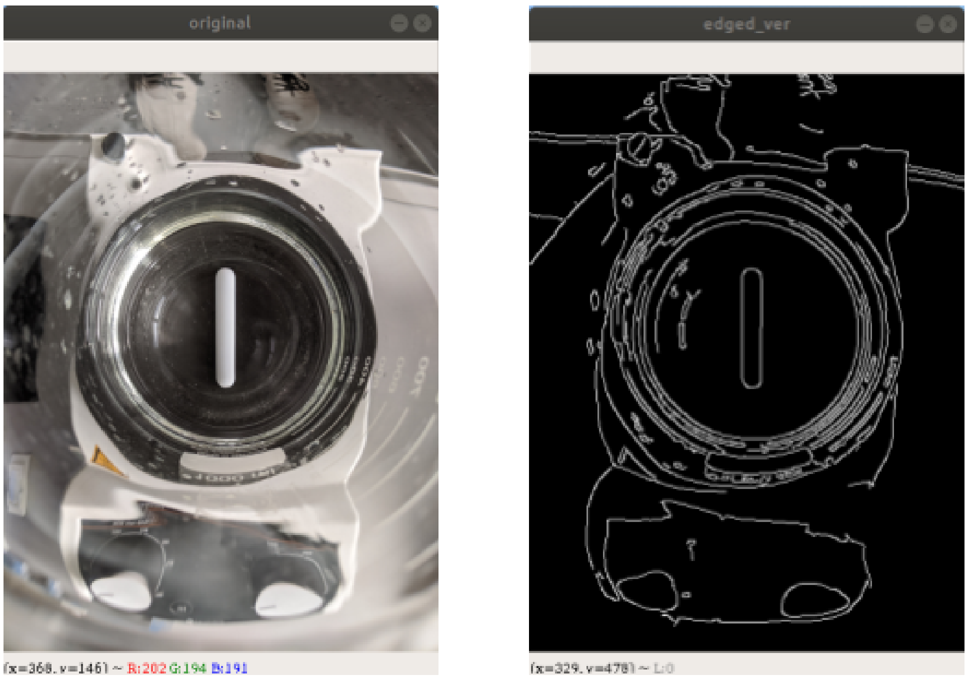
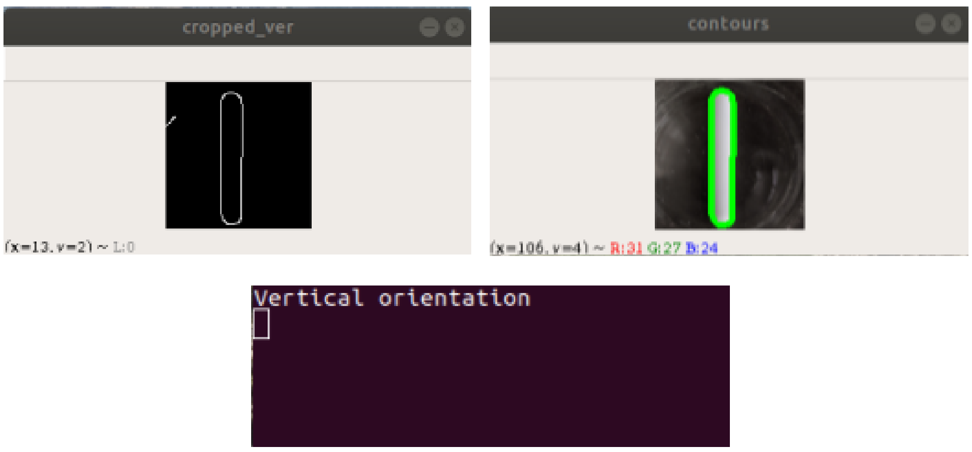


Figure 5.8: Image processing

The prerequisite step is to upload the picture made straight after the stirring and heating process(Figure 5.8(left)) described in chapter 5.1.1 at robot camera out of pre-programmed position. After the picture is taken and uploaded, the main core position detection program can be started.

The first step here will be to simply load the picture into the program. After it's done, the next pre-processing step begins. In this step, the original picture is resized to reduce the computational power needed for upcoming operations. After being resized, the picture is grayscaled to use the next methods on it, namely Gaussian blur and Canny edge detection[reference to the openCV documentation]. You may see results after using blur and edge detection in the Figure 5.8(right). The usage of those two methods makes it possible to use upcoming necessary detection methods. Analyzing the technical side of the stirring process a simple conclusion was drawn - in all circumstances, the mixing cores always stay in the middle of the plate. This allows cropping the previously preprocessed picture to the exact region that fits the core, considering the size of the core to decide the dimensions for the cropping step. The cropping improves the ability for finding the right contours later on as well as it reduces the necessary computational power even further. The cropping completes the whole pre-processing step. The final result after pre-processing is shown in the Figure 5.9(left). The next step is contour detection. In this step, the stock contour detection method out of the OpenCV library is used with a chain approximation method[13] set to none to get as many points to analyze as possible. After using the approximation method all the points are saved to be analyzed later. This concludes the contour detection step.

The contour analysis part is the most important one in the whole program. First of all, we sort the contour points out of the previous step in the descending order. The biggest object in our cropped image would be the mixing core, therefore its contour will be stored as the



```
Vertical orientation
```

Figure 5.9: Image processing

first value and hereby the core itself is found in the picture(Figure 5.9(right)). To analyze the orientation of the core we iterate through the x and y values of the corresponding contour, finding the biggest and smallest values of both x and y. After that, we compute the width and height of the core by subtracting the min value of the corresponding max value. After that, the conclusion is drawn about the orientation and printed into the terminal(Figure 5.9(bottom)).

This finalizes the whole core position detection. The output data can instead of being printed out be directly transferred later on into the software developed within mios as described in 5.1.3 and with help of this software the tweezers will be rotated in the necessary orientation to extract the core.

5.2.3 Scales output recognition

This program was created to analyze the picture of the scales screen output data, taken on the mounted camera from the pre-programmed location as a part of the weighing process(chapter 5.1.1) to analyze and double-check the correctness of the amount of buffer to be added to the solution. In order to make this program cross-platform, as well as the previous core position detection program, it was developed in Python with the usage of OpenCV.

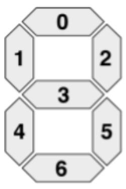


Figure 5.10: Seven-segment display

The display of the scales(Figure 5.12(top)) prints out the data utilizing several seven-segment displays, an example of such a display is shown in Figure 5.10, so the goal is to recognize each one of those numbers and transfer the result for the subsequent use. In Figure 5.11 the flow

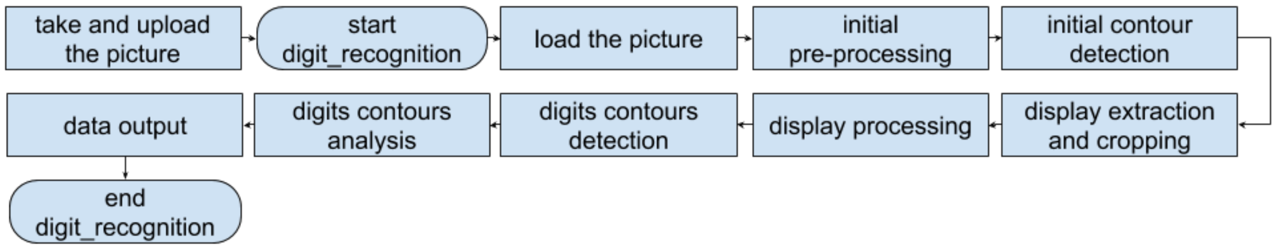


Figure 5.11: Output recognition algorithm

chart of the whole program is to be seen.

The prerequisite step is to upload the picture made on the mounted camera straight after placing the buffer reagent into the scales(Figure 5.12(top)), as a part of the weighing process described in chapter 5.1.1. After the picture is taken and uploaded the scales output recognition program can be started. The first three steps here will be very similar to the previously described core position detection program. Namely, loading the picture into the program and performing the pre-processing. In this step, the original picture is resized as in core detection. Next, the picture is grayscaled, the Gaussian blur is applied onto the grayscaled variant to reduce high-frequency noise and the edge map is computed via Canny edge detector. You may see results after using blur and edge detection in the Figure 5.12(bottom). The usage of those two methods in the picture makes it possible to use upcoming necessary detection methods.

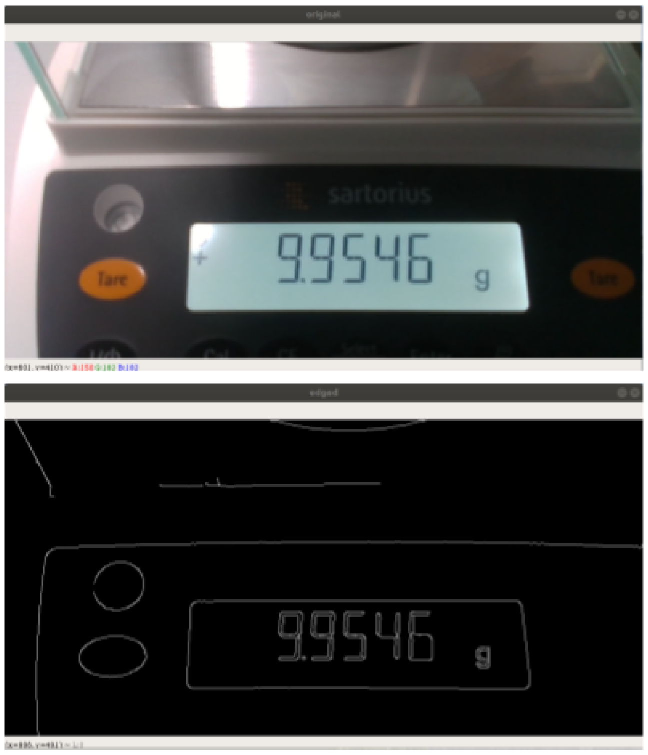


Figure 5.12: Input & pre-processing

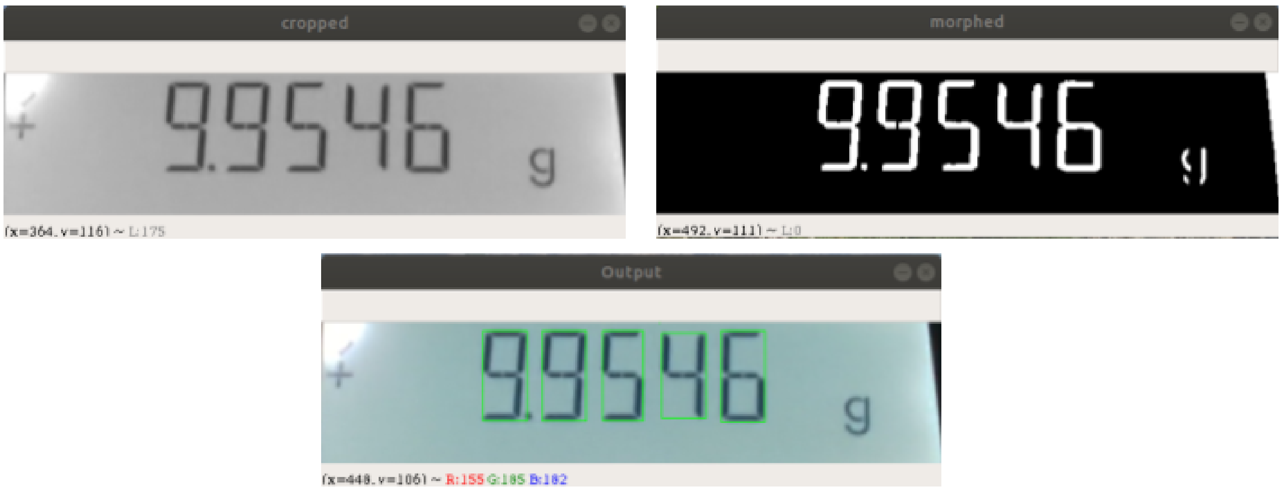


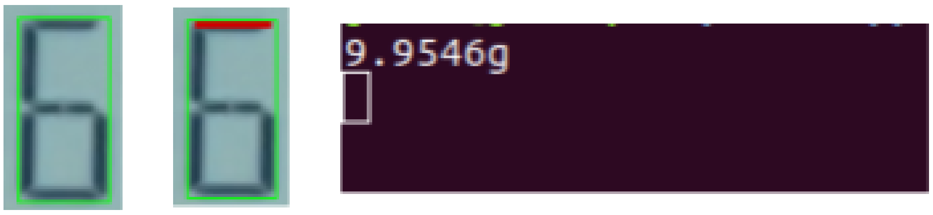
Figure 5.13: Processing

Moving on to the next step - initial contour detection is performed. This step is to find the regions of the output display on the scales. For that, we need to extract the contours of the regions in the edge map. For the next step, display extraction and cropping, the contours should be sorted by their area in the descending order and iterated over each one of them individually. In each iteration it is checked whether our contour has four vertices, if it does - it is assumed that the scales display has been found. This assumption is reasonable since the largest rectangular region in our input is the display itself. After finding those four vertices the display can be extracted via cropping. Those four vertices will be exactly the parameters needed for cropping only the necessary region with the display out. This concludes this step of the program, the result may be seen in the Figure 5.13(left).

Now moving on to the display processing step, the picture has to be prepared for the extraction of the digits. For that purpose, the image has to be thresholded to reveal the dark regions(digits) on the light background(background of the display). After trying out several threshold methods the THRESH_BINARY_INV[reference opencv doc] proved to be the best suitable option - the output quality was the best. To make the thresholded image cleaner the series of the morphological operations is applied, finalizing this step of the program. The results are shown in the Figure 5.13(right).

After having this clear picture we can begin with the detection of digits by iterating over contours once again as it was done in the display extraction. This time only the main goal is to find the actual digits. For each digit contour, the bounding box is created around to provide the validation methods that everything is done in the right way. In the case of digits, the potential candidate is found through its dimensions - appropriate width and height of the element. After finding all of the digit contours they are being sorted from left-to-right based on their x,y coordinates. The results of the digits contours detection can be seen in the Figure 5.13(bottom).

Next comes the digit contours analysis in order to recognize the number. For this purpose, the program iterates over each of the digit contours(Figure 5.14(left)). Given the digit contours, the seven segments of the digit display have to be localized and extracted. The approximate dimensions of each segment are computed based on the digit contour dimensions. The new list of x-y coordinates is defined that corresponds to the seven segments shown in the Figure 5.10. The Figure 5.14(middle) shows an example of the red line being drawn over the current segment being investigated. In order to analyze the status of each of the seven segments with each of the digits the new array is initialized. The program loops over the coordinates of each segment and analyzes the ratio of non-zero pixels in each of the seven segments, thereby getting the data about which of the segments on or off. After completing the iterations over all of the segments the array is analyzed and passed for the output data to be generated. In the end, all the data is being printed into the terminal(Figure 5.14(right)) and would be transferred to the corresponding mios segment described in chapter 5.1.3.



```
9.9546g
```

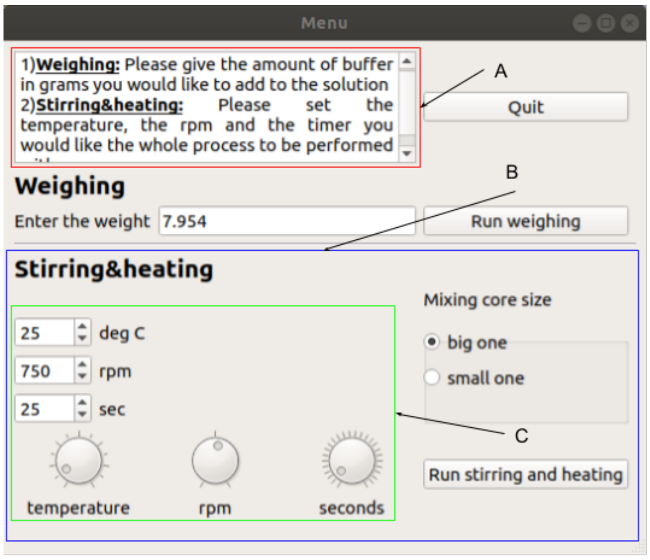
Figure 5.14: Digit & output

5.3 User Interface

The main task of the new user interface is to provide the ability to the user to control, overview, protocol and set the desired settings for the processes. This improves the autonomy of the whole process, making it more flexible and capable of performing custom operations. In order to make it possible to run on different operating systems the program was developed in Python with the usage of Qt Framework by Qt Group Plc. Being cross-platform means that this user interface can be easily integrated later with all the programs that would be developed in mios as described in chapter 5.1.3.

In Figure 5.15 the user interface with some given input values for the processes is shown. This menu consists of two parts. The first one on the top has small instructions explaining what all of the input data mean and control. Right below the instruction(Field A) is the part responsible for operating the weighing process as described in chapter 5.1.1. There is an input field, which allows the user to enter the values, that would be checked by the robot in the next step after pressing the “Run weighing” button. After pressing this button, a Weighing window appears(Figure 5.16) with several control buttons, small instruction and couple output fields. In the meantime, the robot moves into a pre-set position close to the scales display and takes a picture of it.

After the picture was taken it would be analyzed with the scales output recognition program described in 5.2.3 and the detected value output as well as status report will be shown in the fields A. In the case, as shown in this Figure 5.16, status reports us that we have to add some more buffers to the scale. After doing so, the user has the ability to run the process again by



1)**Weighing:** Please give the amount of buffer in grams you would like to add to the solution
2)**Stirring&heating:** Please set the temperature, the rpm and the timer you would like the whole process to be performed

Weighing

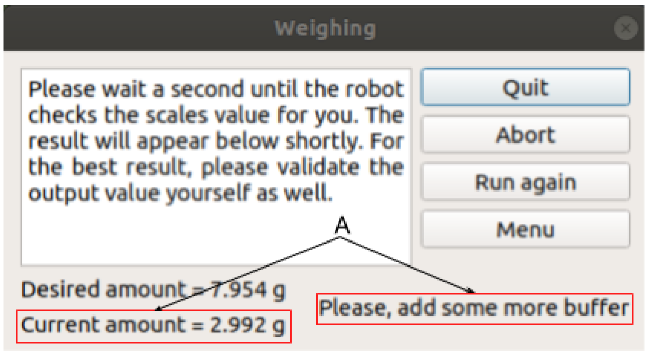
Enter the weight 7.954

Stirring&heating

25 deg C
750 rpm
25 sec

Figure 5.15: Main menu

pressing the corresponding button or by toggling the robot, as described in chapter 5.1.2. All of these steps are directed towards creating a sort of a fail switch, where the user will check the correctness of the robot output values and vice versa, the robot will check the steps performed by the user, therefore reducing the chance of potential mistake due to the human factor.



Please wait a second until the robot checks the scales value for you. The result will appear below shortly. For the best result, please validate the output value yourself as well.

Desired amount = 7.954 g
Current amount = 2.992 g
Please, add some more buffer

Figure 5.16: Weighing

After running the Abort or Menu buttons the user will be informed about the necessary actions they need to perform next via a pop-up message similar to one in Figure 5.17. Pressing the Quit button closes the program and aborts all the processes immediately.

The second part of the menu(Figure 5.15 - field B) is responsible for the input of the custom data for the stirring & heating process as described in chapter 5.1.1. The spinboxes are synchronized with the dials below(Figure 5.15 - fields C) and are used to set the corresponding values - temperature, rpm and time - defining the duration the heating and stirring would be performed for. It is possible to leave some of the values set to zero, which would allow only to stir or heat the solution for the given amount of time, which provides the user with an extended degree of freedom during the experiment and therefore improving the overall autonomy of the whole

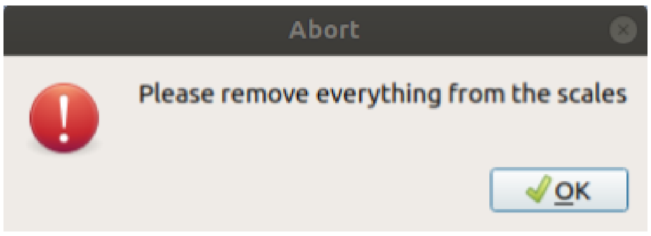


Figure 5.17: Pop-up

process. The radio buttons in the field D will allow to choose the size of the magnetic core, the stirring will be made with. After pressing the "Run stirring and heating" the chosen values for the temperature, rpm and timer would be transferred to the corresponding parts of the mios software(chapter 5.1.3), which will execute the right robot settings to set the right values on the stirringheating plate. During this process, the display window will pop up(Figure 5.18).

This window gives the user the general information about the currently running part of the process(field A) as well as it provides the state of the whole process via status bar(field B). The user always can abort the whole process by pressing the corresponding button. After pressing it, the pop-up window, similar to the ones in the weighing process, will appear, informing about the further steps.

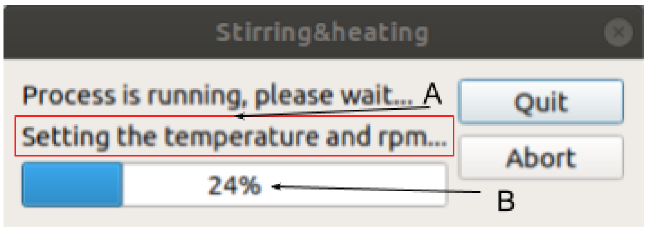

Stirring&heating
Process is running, please wait...
A
Setting the temperature and rpm...
Quit
Abort
24%
B

Figure 5.18: Stirring&heating

5.4 Software evaluation

The developed software allows to automate the whole described process completely(chapter 1.3), meanwhile providing an option for human-robot cooperation. Such human-robot cooperations reduce the chance of performing a certain step in a wrong way, allowing the user to check the data analyzed by the robot, therefore decreasing the chance of technical error, and vice-versa - for the robot to check the human operator performance, hence bringing the negative human-factor influence to the minimum. The developed user interface provides a not only simple control option for the whole process but a basic protocol tool as well. By protocoling, each of the steps of the process the experiment output data can be analyzed later with higher precision and easily be used to reproduce the necessary result.

Despite a working model for the whole process being created, it, unfortunately, has some flaws. The complete autonomy comes with a price. The execution speed of the process of the created model cannot compete with the time needed for a professional human operator to perform the same manipulations. It is caused by the need of the robot to constantly change the tools to perform each step of the process. However, the pure existence of the system allows parallel tasks execution - meaning a human operator may be concerned with other matters meanwhile the robot prepares the needed chemical solution for further usage.

Another important issue is the fact that the system is limited by the pre-programmed positions and locations. Both Desk or mios options suffer from this issue and cannot provide a solution on their own. This matter seriously narrows the use-cases under certain conditions defined by the locations of the devices and equipment used.

Last but not least is the chosen option of using the camera as an input option. Although the developed solution works and can analyze the necessary values and create the corresponding output its performance might be influenced through outer factors. Such factors are lighting conditions and the quality of the input picture. Lighting conditions may create undesired reflections from the scales output display during weighing or from the water surface during core position analysis. Reflections may damage the input data and limit the performance of both recognition processes as a result.

6 Discussion & Further Concepts

Taking the issues described in chapter 5.4 several options addressed towards resolving those problems are provided and discussed with the course of this chapter.

6.1 Remote connection

As was previously stated in chapter 5.4, the currently programmed process execution cannot compete with the human alternative in terms of speed, but it can provide interesting options for parallel task solving. The speed problem might require designing more advanced technical toolsets options, however more advanced designs might jeopardize the process versatility. Therefore, other than trying to optimize the technical part, the enhancing of the abilities for parallel cooperation might be tackled instead. A simple option occurs on the horizon immediately - create a tool that will allow a remote connection to the robot, therefore enabling and controlling the process from any possible location at any given time.

The simple user interface, described in chapter 5.3, requires a connection to the local network to be used and the development of a separate application leads to the platform dependency, and making it work on different operating systems will require a lot of time. Other than that, creating a steady and secure connection with a receiver is a complicated process. An alternative option will be to adopt Telegram messenger by Telegram Messenger Inc for our purposes. Telegram is a cloud-based instant messaging and voice over IP service which has a big number of built-in functionalities. Telegram is open-source and is available on the most popular operating systems, both for PCs and mobile devices. This immediately eliminates the necessity to create multiple OS-specific interfaces on our own. The service also provides APIs (Application Programming Interface) to independent developers and allows third-party developers to create bots.

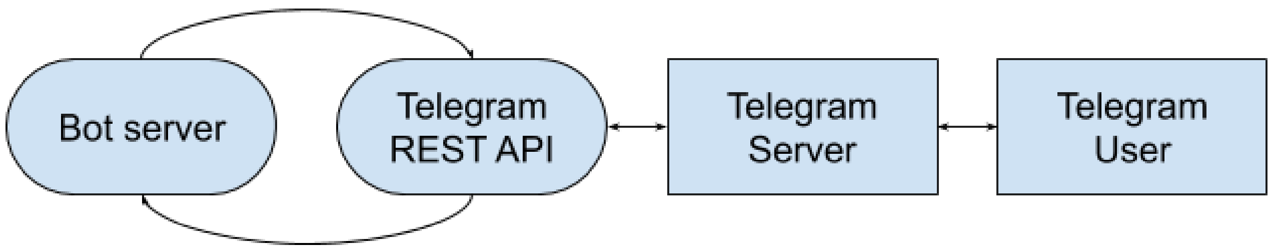


Figure 6.1: Remote connection algorithm

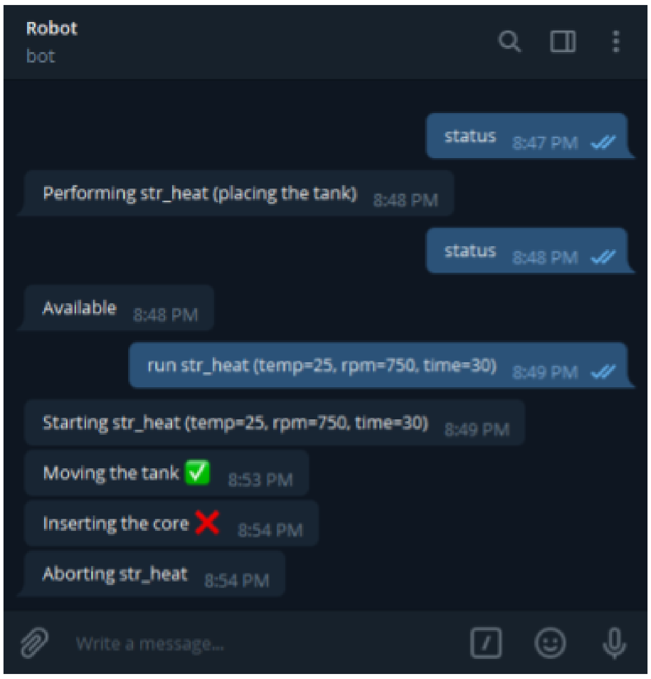
Bots are Telegram accounts operated by programs. The bots' functionalities are huge, from simple messages responding up to online payment acceptance. But what is far more important for our purposes, bots can be integrated into other custom programs, therefore can be adopted into our current workflow.

Default messages and media use client-server encryption during transit, therefore Telegram adoption into the process will not require consideration of the security matter as this side of the problem is already tackled by the Telegram developer in the application itself.

Figure 6.1 shows how such a Telegram option will operate. First and foremost the bot server must be set. In our case, it is NUC, which is used to operate the robot and will do this part just fine. The server will hold the Python written script, which will control the logic of our bot. There are libraries designed for bot-programming specifically, one of the options is python-telegram-bot library.

This Python script will provide connectivity with our previously used User Interface, therefore allowing to set the necessary for the processes values and run them directly through Telegram. Some additional functions might be used within this Python script to provide the user with some additional information about the robot status, like whether it's currently running a process and so on. Through the internet, the bot server is connected to Telegram REST API, a representational state transfer, a software architectural style that defines a set of constraints to be used for creating Web services. Telegram's REST API is essentially a part of the Telegram server used for data exchange between a custom script, used to power a Telegram bot, Telegram server, and triggering actions within the Telegram server, e.g. sending a message. Telegram REST API's requests and responses succeed via HTTPS protocol, making them both secure and easy to execute in almost any programming language.

An example of such workflow created in Telegram can be seen in Figure 6.2 below.



Robot
bot

status 8:47 PM

Performing str_heat (placing the tank) 8:48 PM

status 8:48 PM

Available 8:48 PM

run str_heat (temp=25, rpm=750, time=30) 8:49 PM

Starting str_heat (temp=25, rpm=750, time=30) 8:49 PM

Moving the tank ✅ 8:53 PM

Inserting the core ❌ 8:54 PM

Aborting str_heat 8:54 PM

Figure 6.2: Telegram workflow

Such a program can provide status information, start the processes with chosen input values, can update information about process execution, and many more. The functionality is limited only by your imagination. Concluding, such a remote connection through Telegram will not only provide a simple remote control solution, to begin the process with desired parameters but might provide an option to automatically create protocols as well. An additional feature is an allowance to update and transfer information about processes to multiple users simultaneously, which can be utilized to simplify the process of the results verification by the supervisor.

6.2 Calibration

One of the major issues described in chapter 5.4 is being limited in the execution process by the pre-programmed position constraints. This means, if something in the workplace has or will be changed due to some necessity, everything would have to be reprogrammed to function in those new changed conditions. A similar problem arises, on the smaller scale however, if some of the devices or changing stations were slightly moved by an accidental collision with a human operator or caused by a technical malfunctioning in the robot. In such a situation only some small part would have to be reviewed again to make it functional. Both cases do not limit the autonomy of the process but make the whole system less versatile.

To deal with those issues an adoption of so-called ArUco markers is proposed. An ArUco marker is a synthetic square marker composed of a wide black border and an inner binary matrix that determines its identifier (id). Figure 6.3 shows several examples of such markers. The black border facilitates its fast detection in the image and the binary codification allows its identification and the application of error detection and correction techniques. The marker size determines the size of the internal matrix. For instance, a marker size of 4x4 is composed of 16 bits.[13]

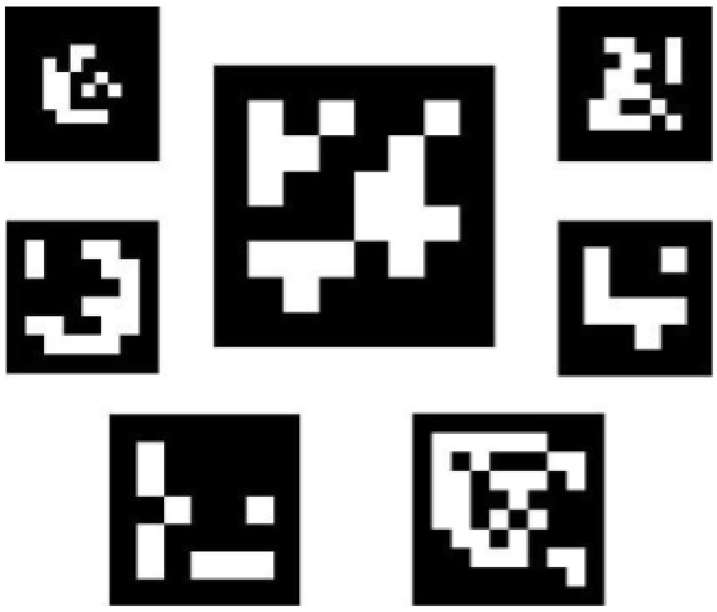


Figure 6.3: ArUco markers

Before the detection, markers need to be printed and placed in the environment. Such marker images can be generated using the drawMarker() function out of the OpenCV library. Markers can be easily rotated in space, however, the detection process needs to be able to determine its original rotation by unequivocally identifying each marker corner.

Given an image of ArUco marker, the detection process has to return a list of detected markers. Each detected marker includes the position of its four corners in the image and the id of the marker[13]. The detection process consists of two steps:

1. Detection of marker candidates - the image is analyzed in order to find square shapes that are candidates to be markers
2. After the candidates have been detected they will be analyzed whether they are actually markers or not through analyzing their inner codification

After detecting actual markers they can be used to obtain the camera pose from them. Exactly these camera attributes can be adopted further by the robot. To perform camera pose estimation the calibration parameters are needed. These parameters are the camera matrix and distortion coefficients which can be obtained by running a calibrateCamera() function provided by the OpenCV library. In the end, we will get a camera matrix with the focal distances and the camera center coordinates, and the distortion values. The estimated pose can be estimated for each marker individually. The camera pose with respect to a marker is going to be a 3D transformation from the marker coordinate system to the camera coordinate system, which is specified by rotation and translation vectors.[reference opencv doc] Figure 6.4 displays such ArUco markers with their coordinate systems.

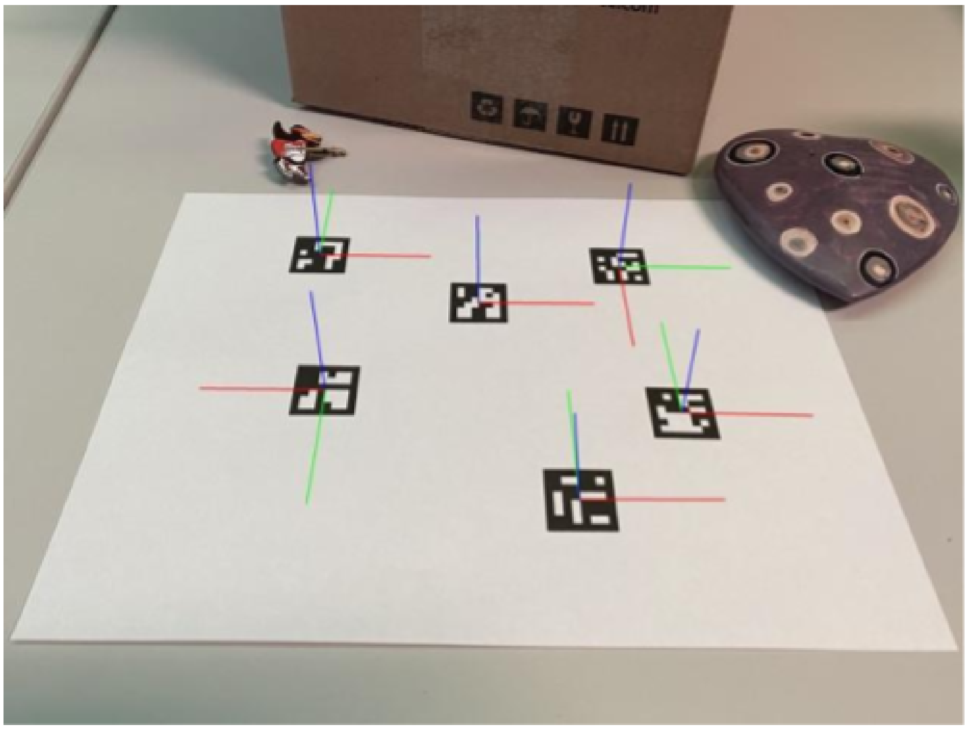


Figure 6.4: M
arkers with coordinate systems

6.3 Machine learning

Even though the software for camera input values is fully functional, as it was later discovered it has several issues described in chapter 5.4. Those issues are caused by the input quality of the picture. Not only the resolution of the camera and the technical side of the matter might cause a problem. Far more important, the surrounding experiment factors, like lighting, might have a serious negative impact on the recognition process and can deteriorate the output as well. Machine learning might provide a solution to the quality problem.

The nature of machine learning as a study is the creation of algorithms that improve automatically through experience. There are numerous examples of use cases for such algorithms, form common adoption in different applications to create a more personalized content up to complicated solutions, like facial recognition techniques or others. Thus, the algorithms can be developed to improve the quality of the output data of the recognitional techniques, described in chapter 5. Machine learning algorithms can potentially be adopted to analyze damaged data, in our case by light reflections for example, and to predict and provide a correct output.

Much more advanced techniques can theoretically be adopted to eliminate the problem of fixed locations, described in 5.4 as well. Those techniques might be used to analyze and recognize the devices used in the experiment, get their locations, and help with human-robot interaction too. However, such technologies can be extremely difficult in realization and will require much more time then the course of a single thesis.

7 Conclusion

The main task of this bachelor thesis was to develop a human-centered automated solution using a collaborative robotic arm to perform a common and routine laboratory process to produce buffer solutions or other culture media that would involve the same or similar procedures in order to be prepared. With this solution a weighing, stirring and heating processes for buffer or other media preparation were automatized. During the weighing process, human-robot interactions were introduced to improve the quality of the result and to provide bidirectional control of the input data.

In order to automate the process with Panda, a light-weight robot manipulator by Franka Emika, the required tools for each step in this process were developed and printed on a 3D printer. For these purposes, the whole process, the dimensions of the equipment, and other factors were carefully analyzed and the necessary tools were created after several tries. The next step was to analyze the output value of the scales for the weighing process and to recognize the stir bar position for its correct extraction. Both output analysis and core recognition were completed by utilizing the camera mounted on the robot's manipulator. The necessary pictures were taken out of the required positions and were processed with the developed script in Python to generate the required results - the scales output data and the orientation of the core within the bottle to use this data for further automation of the whole process.

Caused by time limitations, all of the movements of the robot were programmed within the Desk interface, however, a GUI for the further mios integration was developed. This user interface allows the user to choose the desired values for the process, therefore improving the autonomy of the whole system. Later on, the created user interface can be adopted to protocol each and every step performed to collect required data for the experiment reproduction or output analysis. The concept to base the mios solution was developed as well, which can be utilized to improve the whole process.

For further improvement, several other different concepts were offered. These concepts adopt methods for potential parallel tasks execution through the remote connection to the developed robotic system, which will be not only an option for process flexibility but will give an additional protocolling and controlling solution as well. Another concept proposed was the future usage of ArUco markers to organize the whole workspace in a more flexible manner. Using such markers can reduce the robot's dependency grade on the fixed positions of the equipment, thus, utilizing the mounted camera, the process precision can be improved and a higher level of versatility on a greater scale can be achieved.

8 References

[1] 2015. “Ethics and Experiments: Problems and Solutions for Social Scientists and Policy Professionals”. In:

[2] James Boyd. “Robotic Laboratory Automation”. In: *Science.* 2002, 517—518.

[3] Matheus C. Carvalho. “Integration of Analytical Instruments with Computer Scripting”. In: *Journal of Laboratory Automation.* 2013, 328—333.

[4] David L. Nelson; Michael M. Cox. “Sixth edition”. In: 2013, 47—75.

[5] Robin A Felder. “Modular workcells: modern methods for laboratory automation”. In: *Clinica Chimica Acta.* 1998, pp. 257–267.

[6] Robin A Felder. “The Clinical Chemist: Masahide Sasaki, MD, PhD”. In: *Clinical Chemistry.* 1998, 791—792.

[7] “Franka Control Interface”. In: *https://frankaemika.github.io/docs/overview.html.*

[8] Univ.-Prof. Dr.-Ing. Dr.-Ing. E.h. G.Schmidt. “Einführung in die Roboterregelung”. In: *Skriptum.* 2004, 21—65.

[9] “LIM Source, a laboratory information management systems resource”. In: 2009.

[10] “MIOS Manual”. In:

[11] W. Jeffrey Mortimer James A.; Hurst. “Laboratory robotics: a guide to planning, programming, and applications”. In:

[12] Kevin Olsen. “The First 110 Years of Laboratory Automation Technologies, Applications, and the Creative Scientist”. In: *Journal of Laboratory Automation.* 2012, pp. 469–480.

[13] “OpenCV Documentation”. In: *https://docs.opencv.org/master/index.html.*

[14] Joshua M. Pearce. “Chapter 1 – Introduction to Open-Source Hardware for Science”. In: 2014, 1—11.

[15] Sciavicco L. Villani L. Oriolo G. Siciliano B. “Robotics - Modelling, Planning and Control”. In: 2009, 39—107.

[16] “User Handbook – Panda by Franka Emika GmbH”. In: 2018.

[17] M. A.; Zuk W. M. Ward K. B.; Perozzo. “Automatic preparation of protein crystals using laboratory robotics and automated visual inspection”. In: *Journal of Crystal Growth.* 1988, 325—339.