



TECHNICAL UNIVERSITY OF MUNICH

DEPARTMENT OF INFORMATICS

Project Report

Building a Modular Robot - User Interface

Leonardo Maximilian Freiherr von Lerchenfeld and
Emanuel Buchholz



TECHNICAL UNIVERSITY OF MUNICH

DEPARTMENT OF INFORMATICS

Project Report

Building a Modular Robot - User Interface

Bau eines modularen Roboters - Benutzeroberfläche

Authors:	Leonardo Maximilian Freiherr von Lerchenfeld and Emanuel Buchholz
Supervisor:	Prof. Dr.-Ing. Matthias Althoff
Advisor:	M.Sc. Stefan Liu
Submission Date:	23.08.2018

We confirm that this project report is our own work and we have documented all sources and material used.

Wir versichern, dass wir diese Praktikumsarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet haben.

Munich, 23.08.2018
and Emanuel Buchholz

Leonardo Maximilian Freiherr von Lerchenfeld

Acknowledgments

We want to thank our supervisor Matthias, advisor Stefan and our teammates Hanna, Florian, Daniel, and Alex for making this project a truly remarkable experience.

Abstract

Modular robots bring a high flexibility to the design of assembly lines, but are still complicated to configure. This report shows how someone without any expertise in robotics can set up the modular robot we are building.

Setting up a modular robot is split in four major steps: defining the task, managing the modules, assembling the robot, and calibrating it. To facilitate this process, our graphical user-interface (GUI) is connected to V-Rep and Simulink. Furthermore, the GUI incorporates an algorithm that calculates the required module configuration for a given task. To define the task, the user can load a CAD model of his or her work cell into V-Rep and define the desired positions in the model by moving the end effector. After the task definition, the user will be presented with suggested module configurations which have been calculated by the algorithm. To facilitate the assembly, we have created an assembly instruction. Finally, the user can calibrate the robot by controlling the robot in the real world.

This clear and intuitive GUI provides the user with many functionalities, while keeping the workflow straightforward, such that a beginner can configure the modular robot without any expert knowledge in robotics.

Contents

Acknowledgments	iii
Abstract	iv
1. Introduction	1
2. Procedural Method	4
3. Requirements	7
4. Graphical User Interface	10
4.1. Main	13
4.2. Module Management	15
4.2.1. Own Modules	15
4.2.2. Catalog	17
4.3. Positions	18
4.4. Configuration	21
4.5. Calibration	24
4.6. Teaching	26
5. Evaluation	29
6. Conclusion	33
7. Division of Workload	34
A. V-Rep CAD Software	35
B. Requirement List	38
Glossary	46
Acronyms	47
List of Figures	48

Contents

List of Tables	49
Bibliography	50

1. Introduction

Fixed-morphology robots have been an essential part for the construction of assembly lines in big companies, but they have not yet been used in small companies with a more constrained budget and the need for variable application possibilities. These small companies are the target of variable morphology robots or simply modular robots. The shape of a modular robot can be changed by the user through disassembling it and rearranging the assembly. This provides the following advantages: first, they are easy to repair because modules such as links and joints can be easily replaced if broken. Second, if modules are produced in large quantities, each module becomes more economical, and hence the entire robot as well. Third, one can always create another robot out of different modules for a different task. The modules can be specialized for the execution of varying tasks. For this reason, modular robots enable the creation of highly flexible assembly lines.

Nevertheless, modular robots have not yet been very successful since creating a new robot by varying its module composition does not only have advantages, but also a major disadvantage. A new robot composition requires a new controller and programming a robot is expensive. For the modular robot described in this report, Giusti and Althoff solved this by storing the kinematic and dynamic information in each module and by detecting the current robot configuration [1]. Thus, the centralized controller synthesizes model-based control laws on-the-fly.

In the project *Building a Modular Robot*, the hardware team designed joint [2] and link modules [3] that make up the modular robot. Nevertheless, how can we know how to assemble a robot that solves our tasks without expert knowledge? In order to solve this problem, the algorithm team has developed an algorithm that returns task specific solutions for possible robot configurations [4]. The algorithm is based on [5], [6], and [7]. A possible robot configuration is shown in Figure 1.1.

The tools are now available, but still not usable without expert knowledge of the platform. For this reason, a graphical user interface (GUI) has been implemented to intuitively guide a beginner through the design process. This includes managing the modules, specifying a task, and calibrating the robot. Moreover, the solutions provided by the algorithm are presented to the user in an clear way, also including assembly instructions. The implementation and workflow of this GUI is the focus of this report.

The main goal is to make the workflow of designing the modular robot as under-

standable and convenient for the user as possible:

1. The user can load a CAD model of the work cell and within this model, he¹ can define the positions that the end effector of the robot must reach. Furthermore, the CAD work cell also serves as an obstacle model for the motion planning of the robot.
2. The algorithm calculates the modules required for the task and returns several solutions for a robot composition.
3. Finally, the user can assemble the robot and directly control it with the GUI to calibrate the positions in the real world.

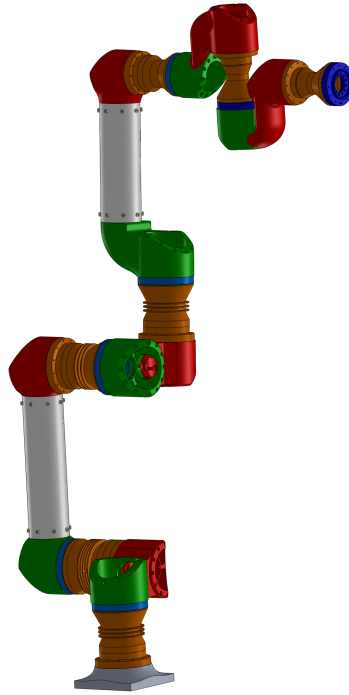


Figure 1.1.: CAD model of a possible robot configuration

The rest of the report is structured as follows. In Chapter 2, we describe the procedural method that we used to ensure that the user-interface is structured, clear, and intuitive. This process includes the requirements definition as described in Chapter 3.

¹In this report, we will refer to the user as *he* by convention in order to not constantly repeat *he or she*.

A user instruction of the GUI, as well as a scenario example, are given in Chapter 4. The evaluation of our GUI follows in Chapter 5. In Chapter 6, we draw conclusions about our results. Finally, in Chapter 7 we indicate how the work was divided within the user-interface team.

In the first appendix V-Rep, we explain how the connection of the GUI to V-Rep works, which allows the user to define the required positions within the CAD model. The V-Rep section is particularly relevant if changes in the GUI are needed. Additionally, the complete requirement list is provided in the second appendix.

2. Procedural Method

Our aim is to design and implement a GUI that is clear, structured, and intuitive. To achieve this, we have followed the recommendations of *DIN EN ISO 9241-210: Ergonomics of human-system interaction – Human-centered design for interactive systems*¹ [8]. Figure 2.1 shows the interdependence of human-centered design activities.

In this chapter, we analyze the characteristics of the robot users, where the robot will be used, and how to ensure a great user experience.

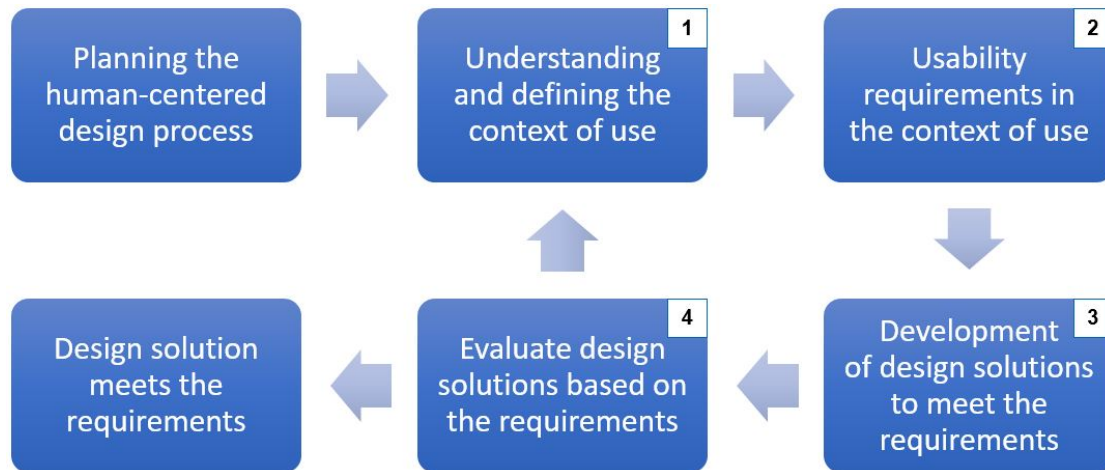


Figure 2.1.: Human-centered design process [8]

Understanding and defining the context of use

As a first step, we determined who will be interested in our modular robot in terms of companies and especially who will use our modular robot in terms of employees. As mentioned in Chapter 1, small companies have a high interest in modular robots. However, they have no experts for programming those robots. Therefore, we assume

¹Note that we had only the German version of the norm and hence the translated terms may not directly correspond to the original English version of the norm.

that our user has at least a high school diploma or a completed vocational training, but not necessarily an university degree in robotics and programming. Our user is characterized with an age ranging from 18 to 70 years. The user's aim is to configure the robot for a particular task, like a pick-and-place task, that might be integrated into an assembly line. Moreover, the production hall is characterized by day light, but also artificial light, especially in the winter season. The temperature inside a production hall is usually between 20 and 30 degrees Celsius. To understand the context of use, we have prepared a short survey in order to improve the user-interface. The survey questions are listed below:

- What is an acceptable cost range for modular robots?
- Which functions should the modular robot have?
- How fast must the robot be? What should the weight of the workpieces be? How big must the action radius of the robot be?
- Who will assemble the robot and how much time should it take?
- Who will reconfigure the robot?
- How is the robot environment regarding available space, lighting and thermic conditions?

For a commercialization of our modular robot, we strongly recommend to conduct this survey at several small companies in order to better understand the context of use.

Usability requirements in the context of use

One of the GUI's main goals is to provide a good user experience due to a good usability of the system. This is achieved by creating an intuitive workflow which helps the user to learn the usage of the system on the go, increasing therefore the accessibility of the system and productivity of the user. To interact with the GUI, the user currently needs a Windows machine and MATLAB (2018a). He can interact with the GUI by using a keyboard and a mouse. The output shown on the screen is self-explanatory and guides the user through the configuration process. The requirements regarding safety and design are explained in more detail in Chapter 3 and in the appendix.

Development of design solutions to meet the requirements

This activity has been by far the most iterative part. Before we started programming, we drew figures such as Figure 2.2 to visualize our ideas. After the other team members

evaluated our ideas, we rapidly started designing the graphical interface with MATLAB Appdesigner, first without implementing the functions within the GUI. In further iterations, we then started implementing the GUI functions.

For example, this process of design iterations can be shown with the module management tab. The current Module Management (as in Figure 4.3) allows the insertion of the inertial tensor matrix. Additionally, we provided a button showing the legend of kinematic parameters. Moreover, for motion planning, the algorithm needs a STL-file of the module. Last but not least, the modular robot team has found that it was necessary for modules to define the input- and output connector size. All these functionalities were implemented iteratively after the feedback of our team members. Visually, the iterative process can be observed when comparing Figure 2.2 with Figure 4.3.

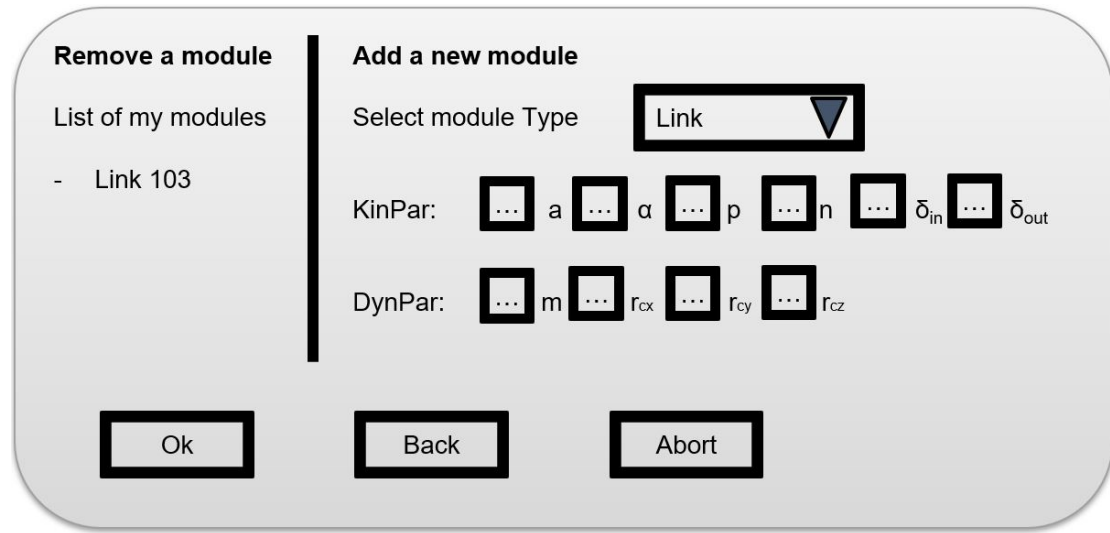


Figure 2.2.: First concept for module management

Evaluate design solutions based on the requirements

In the weekly meetings with the entire team, we continuously revised our design for mainly two reasons: first, to include more functionalities and second, to enhance the user experience regarding clarity, intuitiveness, and structure. Afterwards, the cyclic process was repeated by updating the requirements and developing new design solutions.

As previously mentioned, the next chapter presents the GUI requirements and in Chapter 5 follows the evaluation, which shows that we have fulfilled most of them.

3. Requirements

This chapter outlines the usability requirements for the configuration of the modular robot and how the user is led through the configuration process. To illustrate this process, we have created a use-case diagram, as shown in Figure 3.1. In this diagram, we have three actors: the user, the robot, and the ModRob Company that sells the robot modules.

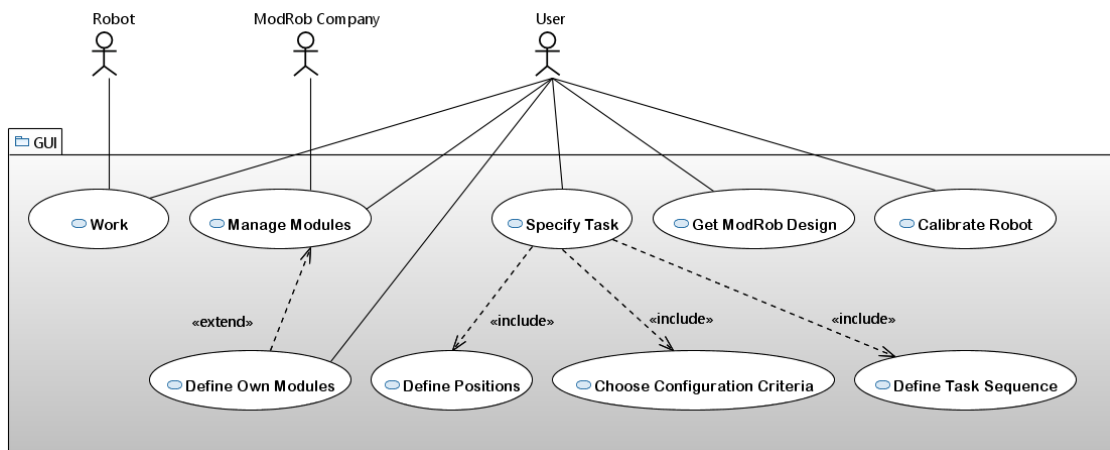


Figure 3.1.: Use case diagram for the GUI

The user desires a robot that does a specific task for him. In order to achieve this, the user can obtain modules from the ModRob company or define his own modules. Then, the user must specify the task, which can be further decomposed into defining the positions, choosing configuration criteria, and defining the task sequence. After the task is defined, he will obtain a suggestion for a robot composition in order to assemble the robot. When the robot is assembled, the user must calibrate the robot positions in the real world. Finally, the robot is ready to work.

The use case diagram describes the functionalities that the robot must have, but not how to design this process intuitively. For this reason, we have defined a requirement list with the recommendations from the norms DIN 9241 [8], VDI 3694 [9], DIN1332 [10],

DIN 15066 [11], and DIN 10218 [12]. The requirement list has been iteratively updated as described in the procedural method in Chapter 2. It also defines requirements for the robot hardware and control to ensure a great user experience. The entire requirement list is attached at the end of the report, where requirements that are not fulfilled or that are the responsibility of another team are marked in red. In the following, we show the structure of our requirement list, where each section is provided with an example:

1. Safety

- If singularities can lead to dangerous movements, the user must be warned before reaching a singular configuration.

2. System Requirements

- MATLAB R2018a

3. Aims of the Interface

- These principles are followed for the design of interactive systems:
 - suitability of the task
 - ability to describe itself
 - evidence of user-expectations
 - suitability for learning
 - controllability
 - tolerance for errors

4. Interaction with the Interface

- While calculating a robot configuration, the user can see the progress of the calculation.

5. Adjustment of the Robot

- The robot should calculate its trajectory by itself.

6. Operation of the Robot

- The robot should have the following interaction functions:
 - Select modules
 - Add new modules
 - Add new position
 - Define configuration criteria and calculate a suitable configuration

- Calibrate
- Edit sequence of motions
- Turn on / shut down
- Work

In the next chapter, we present the GUI and in Chapter 5 we evaluate the requirements.

4. Graphical User Interface

In this chapter, we explain how to use the GUI. We will lead you through the entire GUI and show you every single function. Buttons are underlined, e.g. Apply, and letters in brackets, e.g. (a), refer to the figures of the corresponding tab. At the end of each section, we will give you an example as a scenario. The scenario is setting up the modular robot for a pick-and-place task using a self-defined end effector.

Setting up the Graphical User Interface

In order to use the GUI, you need to obtain MATLAB 2018a. Make sure you install MATLAB including Simulink Real-Time and Robotics System Toolbox. Since it only works on Windows, make sure to install MATLAB on a Windows machine, preferably Windows 10.

Now that MATLAB is set up, you need to get the *modular-robot-toolbox* repository. After you are added to the repository, you can clone it to your hard drive with `git clone 'urlToRepository'` and update it with `git pull -all`. You will find the URL in the web interface of GitLab. Then you need to install the V-Rep software. The GUI was tested with the V-Rep version V-REP PLAYER V3.5.0 rev4. You can get it either by downloading it from their website or by navigating to the folder `*/HumanInterface/UI` and installing it from there.

Now everything is set up and you can start the GUI by opening MATLAB, navigating to the UI folder and then typing `GUI` into the MATLAB console.

Overview

In general, to make the robot work, you have to select the modules that you have, define your task, and finally, calibrate the start and end positions of the task in the real world. As mentioned in Chapter 3, defining the task can be further decomposed into defining the positions that the end effector should reach, choosing your configuration criteria, (for example, maximum payload) and defining the order of positions and tasks the robots should accomplish.

The general design of the GUI is shown in Figure 4.1. The GUI consists of six major parts: lamps, state, tabs, description, help button, and the main window. In the following, these six parts are described.

1. The **lamps** on the left show you which steps you already have done (green) and which steps you still have to do (red). A lamp turns from red to green after pressing Apply in the corresponding tab if all necessary parameters are defined.
2. The **state** on the bottom tells you what the next step is and which tab you have to click. Moreover, the state provides feedback according to your input.
3. The **tabs** on the top represent the steps, which are ordered from left to right.
4. Each tab provides a short **description** of what you can do in each tab.
5. Press **Help** to open the the report. In the state on the bottom, the corresponding page to the tab is mentioned.
6. The **main window** of each tab includes colored buttons and blue numbers, which mark the necessary steps. Red buttons usually make the robot stop or reset a tab, blue buttons are necessary to complete a step, gray buttons provide additional information or properties to set, and only the Work button in the Main tab is green.

In the following sections, only the main window will be shown for clarity.

Hint

If you do not know what to do next, look at the lamps on the left and on the state at the bottom. They will show you which steps are finished and which step follows next. In each tab, follow the blue numbers. If you are not sure anymore, read the short description below the tab bar or press Help to show the detailed description in this report.

4. Graphical User Interface

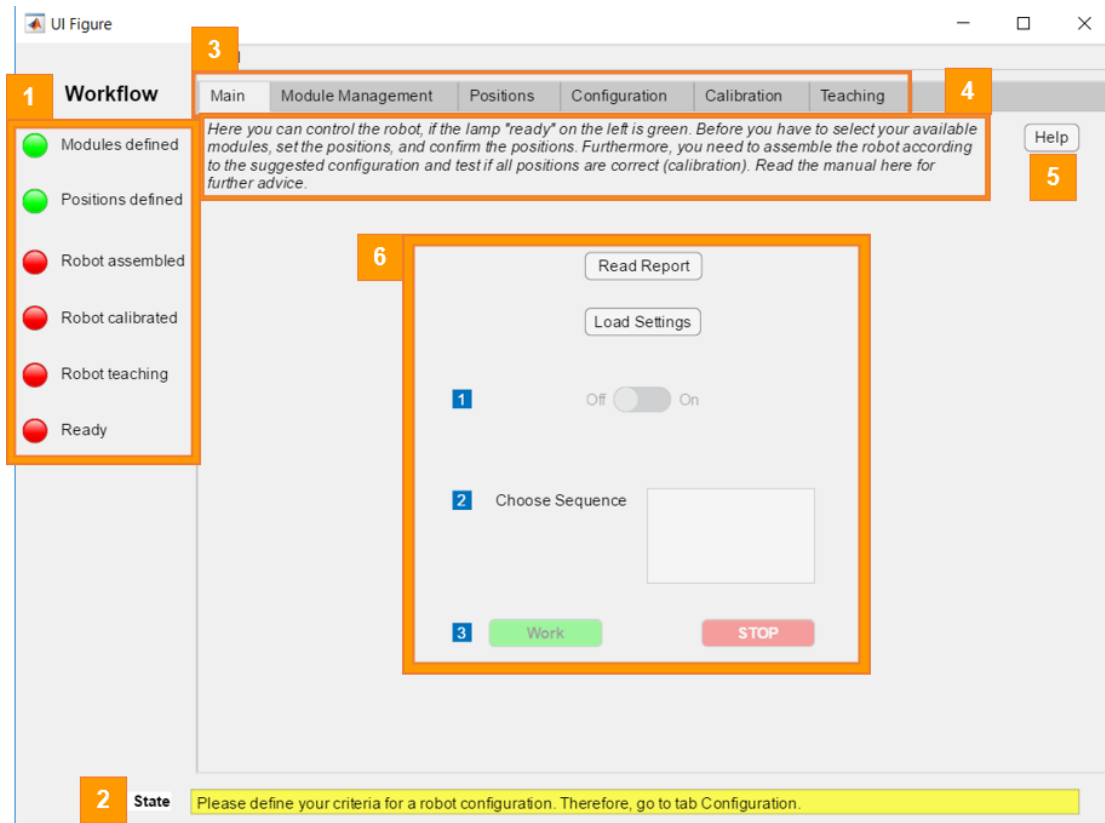


Figure 4.1.: General design of the GUI

Scenario Example

Suppose we want to outsource the tedious task of moving dishes from the kitchen table to the sink. We already possess a starter kit of the modular robot, which has the same configuration as the robot shown in Figure 1.1. Since we will be handling dishes, we need a special end effector. From this, we can derive the following task description:

- Available joints: XL, L (x3), S, XS (x2)
- Available links: L_90_compact_LL (x2), L_360_400_LL, L_90L_400_LS, L_wrist3, L_wrist2
- Own module: soft gripper - end effector (suitable for dishes)
- Positions: table and sink

- Criteria for robot:
 - Energy Consumption: irrelevant
 - Task Time: as fast as possible
 - Payload: 0.5 kg
 - Assembly: rather keep configuration
 - Monetary: very economical

All the examples in the following chapter are based upon this scenario in order to give you a better understanding of how the GUI works and how to fully specify a task for the modular robot.

4.1. Main

The Main tab is used to control a fully assembled robot and only becomes important after you have worked through all the other tabs. It is at the beginning though, because controlling a fully assembled robot (Main tab) is a more common scenario than completely reconfiguring the robot (other tabs).

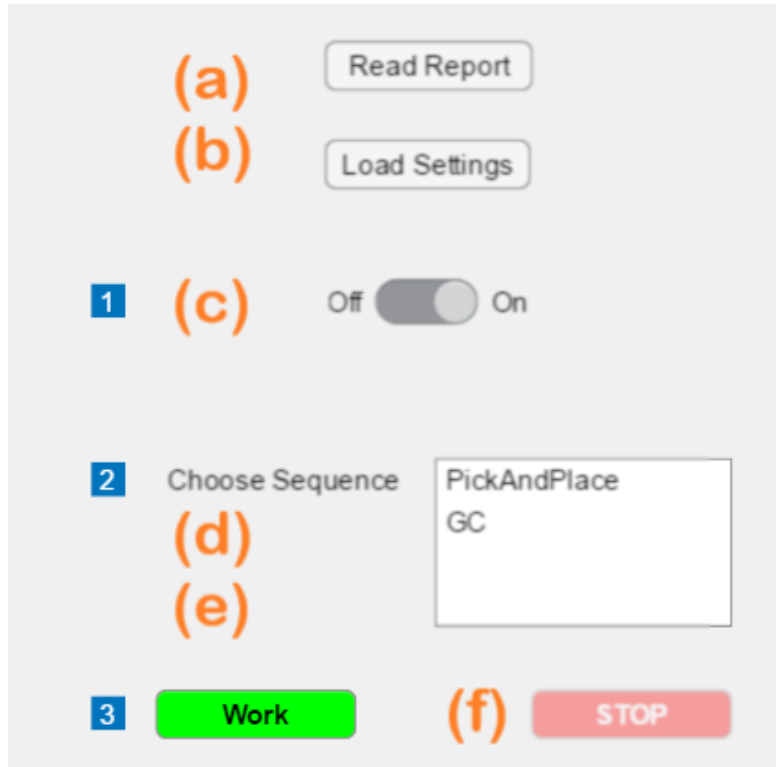


Figure 4.2.: Tab: Main

With the switch (a) you can either turn off the robot, which brings it to the homing position, or turn on the robot in order to select a defined sequence (b). After a sequence is selected, press Work (c) to start the sequence, which will be infinitely repeated until STOP (d) is pressed.

Furthermore, you can open this report with Read Report. If you want to use the Main tab after a restart, you can load the previously defined settings with Load Settings. This will load the taught sequences into the Choose Sequence list and also previously defined parameters into the other tabs, such as, Available Modules in the Module Management tab.

Scenario Example

After we defined the sequences *PickAndPlace* and *GC* in the Teaching tab, they appear in the Choose Sequence list (d). Now we turn the robot on (c), choose the *PickAndPlace* sequence and press Work. After an hour, we stop the robot (f), choose the *GC* sequence and start the robot again (e).

4.2. Module Management

In the Module Management tab, you can either select modules from the catalog or specify your own designed module. For this reason, this section is divided into two subsections: Own Modules and Catalog. If you do not want to specify your own module, you can directly go to Section 4.2.2.

4.2.1. Own Modules

In this section, you can specify your own designed module, which may be a link, a joint, an end effector, or a base. For each module, the windows look differently, but in general the workflow is always as follows:

First, you give your module an unique name (a). Second, with Explore (b) you can select the STL file of your module, which is necessary for the motion planning of the algorithm [4]. Then, you need to define the kinematic and dynamic parameters. If you do not know how the kinematic parameters of your module are defined, press show module legend (c). A PDF file will open, which contains an enlarged image of the *Kinematic Notation for Module Characterization* [1]. Furthermore, you find a description of each parameter in the *DB_Legend.xlsx*.

As mentioned, the necessary kinematic (d) and dynamic parameters (e) vary for each type of module, but in general, the workflow is the same. Nevertheless, there is one additional feature for end effectors: actions. You can either select from predefined actions or type the actions of your end effector (f) in the drop-down menu. Depending on which end effector you use for your robot, you can select the actions, which the robot should do.

After you have everything specified, press Add To Catalog (g) and in the Catalog tab, you will find your module under User-defined Modules.

The screenshot shows the 'Define Own Module' tab in the Module Management interface. The interface is divided into sections for Kinematic Parameters (KinPar) and Dynamic Parameters (DynPar).

KinPar Section:

- Name:** SoftGripper (a)
- STL:** SoftGripper.STL (b)
- Buttons:** Explore, show module legend (c)
- Link:** Joint
- Endeffector:** Base
- Parameters:**
 - δ_{out} : 3.141 rad
 - a : 0.1 m
 - α : 1.571 rad
 - p : 0 m
 - n : 0 m
 - δ_{in} : 0 rad

DynPar Section:

- Parameters:**
 - m : 0.1 kg
 - rc : 0.05 m
 - Inertial Tensor I:**
 - Top row: 0.3, 0, 0
 - Middle row: 0, 0.3, 0
 - Bottom row: 0, 0, 0.3
 - Units: in kg*m^2, m
- Actions:** select or type Actions: Pick, Place, No Action (f)
- Buttons:** Add To Catalog (g)

Figure 4.3.: Tab: Module Management - Define Own Module

Scenario Example

We have designed a soft gripper that is very suitable for the pick-and-place task, hence, we want to integrate it into the modular robot. First, we type an appropriate name. Second, we select the STL file of our gripper. Third, we declare the parameters of our module. The kinematic parameters are the extended DH-parameters as defined in [1]. The dynamic parameters include the mass, the inertial tensor, and the center of gravity. The gripper has the actions *Pick* and *Place*.

4.2.2. Catalog

After you have defined all of your designed modules, here you can select which and how many modules you have.

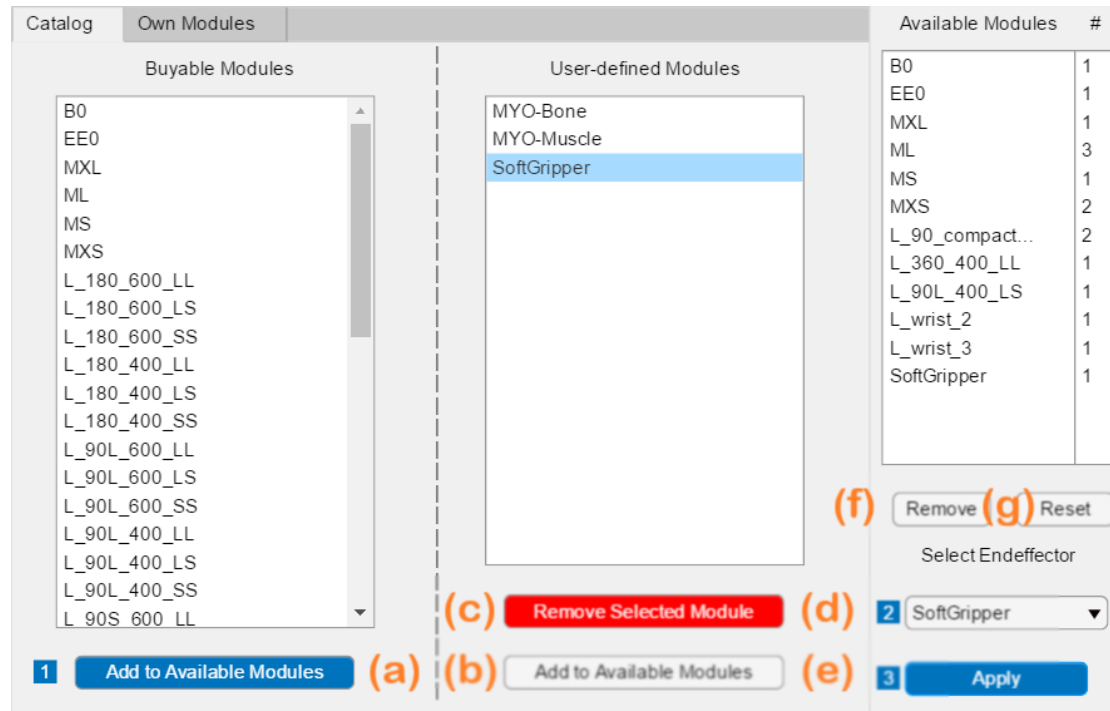


Figure 4.4.: Tab: Module Management - Module Catalog

First, select each module that you have from the standard catalog (which is defined by the hardware- team [3]) and press Add to Available Modules (a). Then, the module appears on the right side. If you want to raise the quantity, press the button (a) again. Select the modules and how many you already have.

Second, select each module that you have defined yourself and press Add to Available Modules (b). Again, press the button (b) until the number on the right of the name of your module corresponds to the number of modules you have. You can also remove User-defined Modules if you do not possess them anymore, you do not want to use them, or if you have accidentally added one. To do this, press Remove Selected Module (c).

Third, choose the end effector (d) that you want to use for your task and finally, press Apply. If you have selected more Available Modules than you actually have, you

can either Remove (f) one or Reset (g) the Available Modules list to the last time you pressed Apply (e).

Scenario Example

In Figure 4.4, we have selected the modules from the standard catalog, which were contained in the modular robot starter kit and we have defined our SoftGripper end effector. Furthermore, we have a link MYO-Bone and a joint MYO-Muscle in our lab. Unfortunately, the MYO modules are not compatible with the other modules. (In [3], you can read the required interface conditions for User-defined Modules.) Therefore, we have not added them to Available Modules. We have added the SoftGripper and we have also selected it as the end effector. We press Apply and the module management is finished.

4.3. Positions

After the Available Modules are successfully defined in Section 4.2, you need to define the start, end, and base positions of the robot. For future versions of the GUI, the functionality of defining additional positions between start and goal positions will be made available. This is already implemented (c), but currently deactivated, since the algorithm only works with these three positions [4]. You have two methods of specifying the positions; either you can use the Manual tab and insert the Cartesian coordinates and Euler angles of the end effector manually or you can choose the V-REP tab and use the 3D environment of V-Rep to specify the coordinates.

Both tabs ask you to provide an *.stl file containing the obstacles (a) of the robot's work cell, but this step is not required in order to fully define the positions. Since the Manual tab just requires the straightforward entry of the coordinates, it is not further covered here.

To enter coordinates with the V-Rep software, you need to start V-Rep by pressing the Start Vrep button (b). After the software is started, the work cell and the robot's base are loaded. The base's position and orientation can then be manipulated with the spinner buttons (b) by either clicking on the up or down arrows on the right or entering a number manually. After the base is at the desired position, it is defined by pressing confirm (e). Then, the currently highlighted Position Type (d) is removed from the list and the defined position is added to the Confirmed Position list with its name and the numerical values of the position for you to review (f).

Now the next position is highlighted in the Position Type list (d), the end effector is added to the 3D environment of V-Rep and the next position can be defined. This is repeated until all the positions in the Position Type list (d) are removed and thus

defined. This workflow can be problematic if you need to redefine an already defined position, but it also provides a user-friendly work flow by restraining the options that need to be controlled by the user. In the rare case that you need to redefine a position, you can do this by pressing Reset Position Type (h). This will activate the Position Type list and reset it. Now you need to manually choose the position you would like to add, move the end effector to this position and confirm the position. After all the positions have been defined, you need to press Apply (g) to finish the position definition process. The confirmed positions will be saved and can be recalled in a future session.

The screenshot shows the 'Manual' tab in the V-REP interface. At the top, there's a 'Choose Obstacles' field with 'Kitchen_WorkCell.stl' and an 'Explore' button (b). Below this is a 'Start Vrep' button (c). The main section is divided into 'Cartesian Coordinates' and 'Euler Angles'. Under Cartesian Coordinates, X is 30 mm, Y is -840 mm, and Z is 620 mm. Under Euler Angles, Roll is 270 grad, Yaw is 0 grad, and Pitch is 270 grad. To the right of the Euler Angles is a 'confirm' button (d). Below this is a 'Name' field (a), a 'Position Type' dropdown menu with 'End' selected (e), and 'Add' and 'Remove' buttons. To the right is a 'Confirmed Positions' list showing 'Base: Euler(grad):[0 0 0] Cart.(mm)[420 -1200 60]' and 'Start: Euler(grad):[-90 0 -120] Cart.(mm)[600 -580 500]', with a 'Remove' button (f). At the bottom, there's a 'Reset Position Type' button (g) and an 'Apply' button (h). A progress indicator shows steps 1, 2, and 3.

Figure 4.5.: Tab: Positions

Scenario Example

We have the choice to either provide the position coordinates manually with the Manual tab or we can use the V-Rep tab. Since we are not absolutely sure about the coordinates of the Start and End positions, we choose the V-Rep tab. First, we provide a work cell as an obstacle model by pressing Explore (a). Then, we start V-Rep and wait until the

4. Graphical User Interface

V-Rep window opens. After the V-Rep window is open and the work cell of our kitchen is loaded, we start to move the yellow robot base to the position we desire and confirm it. We repeat this until all positions are defined. Then, the V-Rep window looks like Figure 4.6. Now that all positions are defined, we just need to press Apply and we are finished with the position definition.

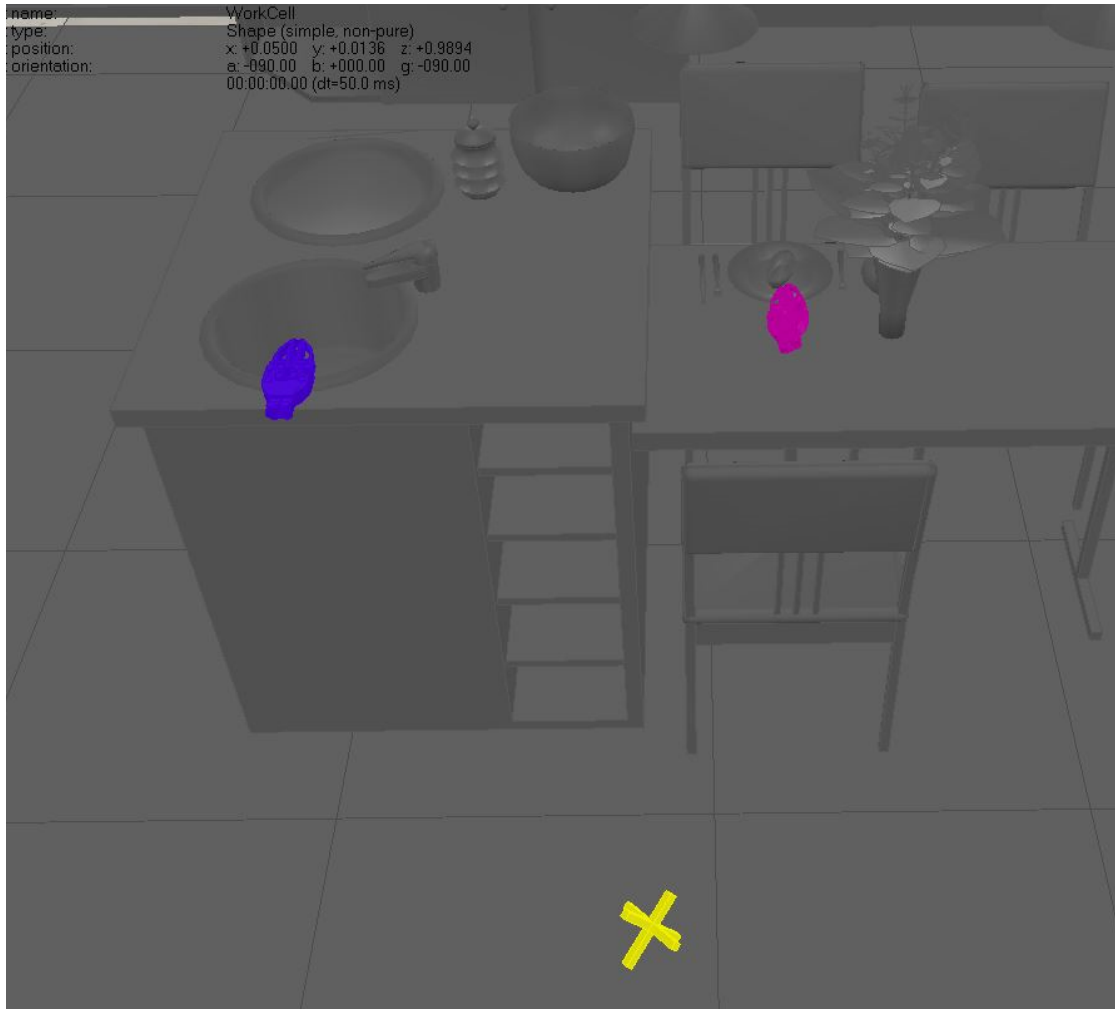


Figure 4.6.: Defining the positions within the CAD work cell in V-Rep

4.4. Configuration

In this tab, you can choose your configuration criteria and you will get several design suggestions for your modular robot. Beforehand, the available modules have been selected and the positions in task space have been defined.

The screenshot displays the 'Configuration' tab of a modular robot design interface. On the left, there are several input fields and buttons. At the top, 'Energy Consumption' is set to 'medium', 'Task Time' to 'fast', 'Assembly' to '1', and 'Monetary' to 'Buy 0'. Below these are buttons for 'Calculate Configuration' (labeled (b)), 'Stop Calculating' (labeled (c)), and 'Finished Assembly' (labeled (f)). A 'Choose Configuration' list (labeled (d)) shows 'Option_2' selected. The main area on the right is titled 'Robot Composition' and shows a diagram of the robot's structure. The diagram consists of a sequence of modules connected by dashed lines, representing joints and links. The modules are: Base (B0), Joint 1 (ML), Link 1 (L_90_compact_I), Joint 2 (MXL), Joint 4 (ML), Link 3 (L_90_compact_I), Joint 3 (ML), Link 2 (L_360_400_LL), Link 4 (L_90L_400_LS), Joint 5 (MS), Link 5 (L_wrist_3), Joint 6 (MXS), Joint 8 (NotDefined), Link 7 (NotDefined), Joint 7 (MXS), Link 6 (L_wrist_2), Link 8 (NotDefined), Joint 9 (NotDefined), Link 9 (NotDefined), and End Effector (SoftGripper). The diagram is labeled with (e) and (f) at the bottom left.

Figure 4.7.: Tab: Configuration

First, you have to select your configuration criteria (a). Table 4.1 explains you how the different criteria lead to different results. Second, press Calculate Configuration (b) in order to start the algorithm. You can stop the calculation at any point with the Stop Calculating button. A design suggestion will be shown in (d) any time, meaning that solutions appear even though the algorithm is still calculating. If you click on an option in the Choose Configuration list (d), the robot composition is shown on the right and the fulfilled criteria on the left.

Energy Consumption and Task Time	Generally, the algorithm always tries to optimize energy consumption and task time. However, if you have any preferences, you can choose them with the slider. Moving the slider to the left results in solutions with low energy consumption, whereas moving it to the right leads to solutions with fast task time.
Payload	Here you can specify the maximum payload of your workpieces. Every suggested design configuration is able to transport the maximum payload that you have defined. However, note that the project <i>Building a Modular Robot</i> is generally designed for a maximum payload of 5 kg [3].
Assembly	<p>In assembly, you can influence the proposed design suggestions significantly. If you choose <i>keep current configuration</i>, the algorithm will determine if the robot can reach the defined positions and the declared maximum payload with the current configuration. Thus, as a result, you will either get one solution or none.</p> <p>On the other hand, if you choose <i>consider all configurations</i>, then your current configuration will be completely ignored. Furthermore, you can choose <i>rather keep configuration, few changes possible</i>, and <i>many changes possible</i>. These options will not influence the results considerably but are used for the internal evaluation of the different solutions from the algorithm.</p>
Monetary	<p>Monetary criteria will influence the proposed design suggestions significantly as well. If you choose <i>no additional modules</i>, then the algorithm will only consider the modules that you actually have. This implies two benefits for the user: firstly, the calculation of the algorithm is faster, and secondly, the user does not need to buy other modules. Naturally, it might also result in no solution.</p> <p>On the other hand, if you select <i>irrelevant</i>, the algorithm will consider the entire catalog. This is especially relevant, if you still do not have a module and want to define your first task or if you want to find the optimal solution in terms of energy consumption, payload, and task time. Unfortunately, then the calculation might be very long. Moreover, you can choose <i>very economical</i>, <i>economical</i>, and <i>generous</i>. These options will not influence the results considerably but are only used for the internal evaluation of the different solutions from the algorithm.</p>

Table 4.1.: The influence of the different optimization criteria on the robot composition

Now you have to assemble your robot. Press Assembly Instruction to see how to connect a joint to a link. Then, a PDF opens, which looks like Figure 4.8. When you are finished with the assembly, press Finished Assembly and the robot will check, if you have assembled it correctly. The feedback is given in the state, which is at the very bottom of the GUI window.

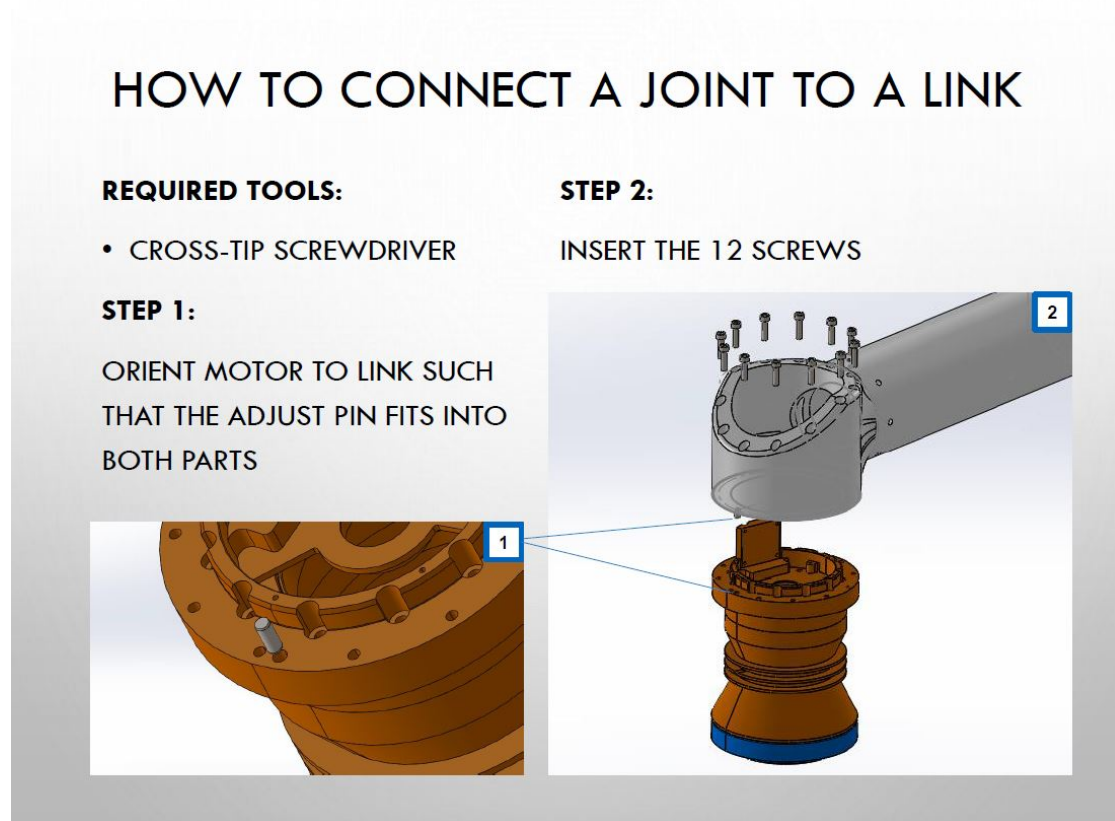


Figure 4.8.: Assembly Instruction: How to Connect a Joint to a Link

Scenario Example

In our task, we do not care about the energy consumption, but we prefer the fastest possible solutions. Our workpieces (the dishes) have a maximum weight of 0.5 kg. Moreover, we do not want to significantly change the current configuration and we are not willing to buy many additional modules. As a result (see Figure 4.7), we only need to change the end effector and can keep the current configuration.

4.5. Calibration

Now that the robot is successfully assembled, you can calibrate the robot in the Calibration tab. This section is separated into two parts: the first part explains how the Calibration tab works with the Simulink simulation of the robot and the second part describes how it works for an assembled modular robot.

With Simulink

First, you need to load the Simulink model and start Simulink. This is done by pressing Connect (a). After the Simulink model is compiled and has started, you can start to move the virtual robot. If you want to control the position of the end effector, choose the Task Space tab and specify the end effector speed. However, if you want to manipulate single joints of the virtual robot, change to the Joint Space tab. Here, you can choose the joint and set its rotation speed. Once you are done, a pop-up window is opened by pressing Set & Control (c). This window contains green buttons that allow you to move the robot in the specified direction. As soon as the button is pressed, it will turn red, and the robot will move until you release the button again.

After the robot is at the desired position, press Disconnect (g). This will stop the Simulink simulation and give you access to the simulation data. By pressing Confirm Calibrated Position (d), the GUI reads out the position and also plot the movement of the robot. Here you need to make sure that you have the whole *modular-robot-toolbox* in your MATLAB path, which contains dependencies of the plot function. To calibrate the next position, you need to start the Simulink simulation again by pressing the Connect button and repeating the procedure above.

Using the Remove Calibrated Position (e), it is also possible to remove a calibrated position if it is not as desired. Once all positions are calibrated and appear in the Calibrated Positions list, you can confirm them by pressing Apply (f).

With the modular robot

The first step is again to press Connect, which starts Simulink Real-Time. After the connection is established, the Automatic Move button (b) is activated and when pressed, it will move the robot to the position you have specified in the Positions tab. You can stop this process anytime by pressing STOP or the emergency stopper on your desk. After the defined position is reached, you can either choose the Task Space tab to control the Cartesian position of the end effector or the Joint Space tab to control single joints. After choosing the respective speed and pressing the Set & Control button (c), a pop-up window will open. Here you can move the robot in the specified direction by

pressing a button. As soon as you release the button, the robot stops moving.

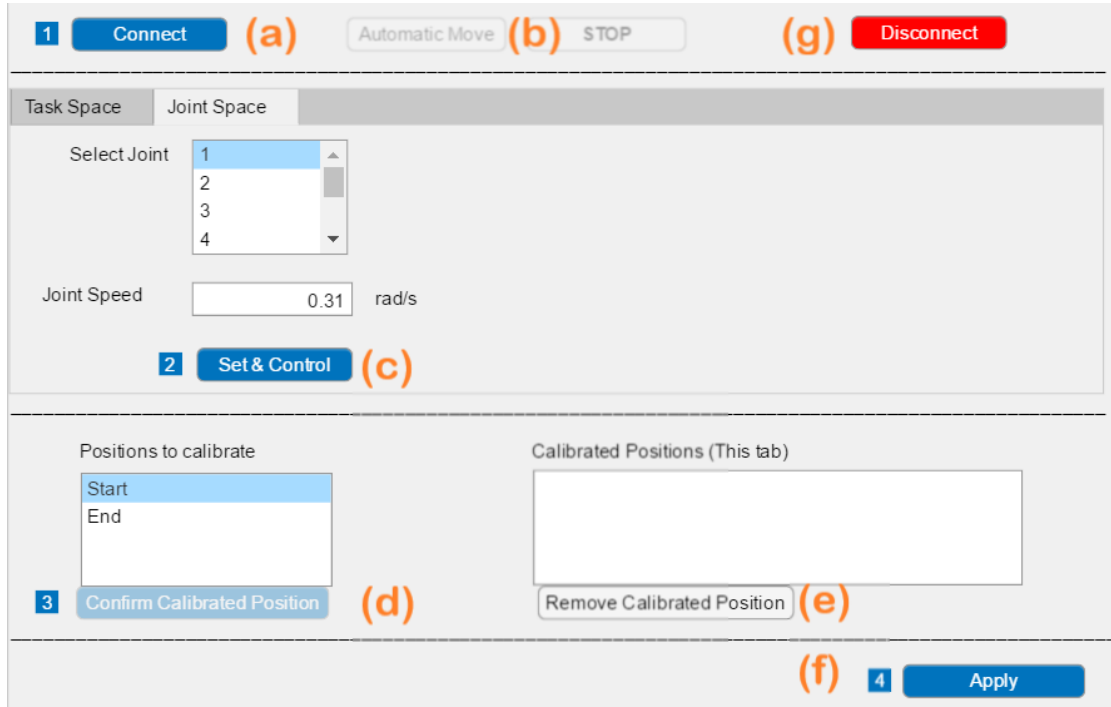


Figure 4.9.: Tab: Calibration

Scenario Example

After we have successfully assembled the robot, we need to calibrate the positions in the real world, which we have defined earlier in the Positions tab. First, we connect with the robot (a) and decide to calibrate the *Start* position. Since this is already selected (d), we press the Automatic Move button which moves the robot to the *Start* position. Once this position is reached, we realize that the robot is still slightly off the desired position. For this reason, we choose the Task Space tab, set the moving speed very low to 0.1 m/s and press Set & Control (c). With the buttons in the pop-up window, we move the end effector to the desired *Start* position and press the Confirm Calibrated Position button (d). Consequently, the *Start* position is calibrated and we repeat the process for the *End* position.

Now that all positions are calibrated, we press Apply button (f).

4.6. Teaching

In the Teaching tab, potential functionalities for the modular robot are implemented, but they do not work with its current version (August 2018).

Robot teaching defines the sequence of positions that the end effector reaches and the actions that it executes. Beforehand, the robot has been assembled and calibrated.

The screenshot shows the 'Teaching' tab of a software interface. It features several interactive elements:

- Top Section:** A row of controls including a 'Type of Movement' dropdown (PTP, LINE, CIRC), 'Available Positions' (Start, End), 'Gravity Compensation' (On/Off with a lightbulb icon), 'Available Actions' (Pick, Place), 'Wait Time' (0 seconds), 'Available Signals' (Robert), and 'Desired Value' (0).
- Buttons:** Below the top section are buttons for 'Add Position', 'Add Action', 'Add Delay', and 'Add Signal'.
- Sequence List:** A central box labeled 'Sequence' contains a list of actions: GC_Start, Act_Pick, GC_End, and Act_Place.
- Right Side:** Includes an 'Expert Mode' checkbox, 'Remove' and 'Reset' buttons, and a 'Confirm Sequence' button.
- Bottom Section:** A 'Defined Sequences' list shows 'PickAndPlace'. To its right are 'Load Sequences', 'Remove', and 'Apply' buttons.
- Other Elements:** A 'STOP' button, a 'Follow Sequence' button, a 'Velocity' input (0.25 m/s), and a 'Sequence Name' input (GC) are also present.

Figure 4.10.: Tab: Teaching

First, select the calibrated position (a) and press Add Position (b). Second, select an action (c) from the end effector and press Add Action. Additionally, you can add waiting times by specifying the time (d) and pressing Add Delay. Then, define your task sequence as desired. Furthermore, you can also activate the Gravity Compensation Mode (e), in which you can move the robot as you desire with your hands. If the robot has reached the desired position, press Add Position (b).

In **Expert Mode** (f), you can also define the type of trajectory in order to reach the desired positions (for further advice, look at Table 4.2). Moreover, you can send and receive signals (h). The uppermost name is the name of your robot (g) and below are the names of the machines and devices to which the robot is connected. If you have

your own robot selected, you can send a signal, which may be numerical or a string, to all devices. When you have another machine or device selected, you can decide with the switch (i) whether your robot should wait for a specific signal (switch In) or if you want to send a specific signal only to the selected device (switch Out).

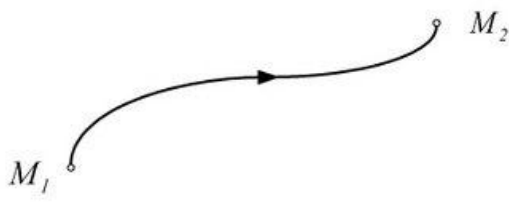
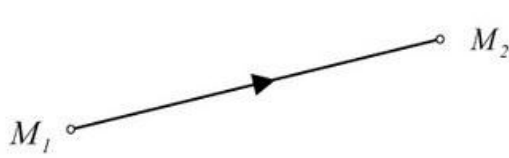
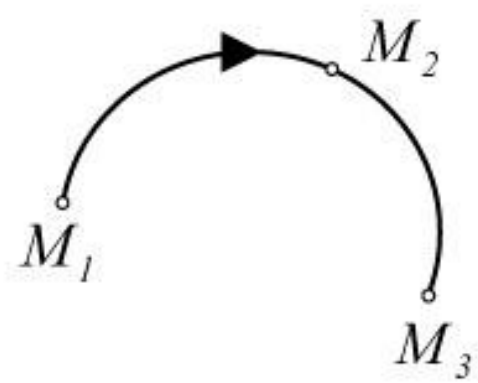
PTP	Point-to-point (PTP) movement is a movement between two given points in task space. The robot axes move synchronically to the target point (M_2), which results in a curved trajectory of the end effector.	
LINE	In a linear movement, the tip of the end effector moves linearly between two points in task space.	
CIRC	In a circular movement, the end effector tip moves along a circular path from the current point (M_1) to the target point (M_3) in task space. To define this trajectory, you need at least one mid-point (M_2). You can also add several mid-points to induce an elliptical movement.	

Table 4.2.: Description of robot trajectories [13]

You can Remove (j) single parts of your sequence (k) or Reset (l) it entirely. After you have defined it, set the maximum velocity of your end effector (m) and press Follow Sequence (n). The robot will then follow the defined sequence. Press STOP

(o) or the emergency stopper on your desk when the robot gets dangerously close to an obstacle. If the trajectory is successful, give a name to the sequence (p) and press Confirm Sequence (q).

By selecting a sequence (r), you can view its composition above (k). You may also Remove (s) a selected sequence. By pressing Apply, the Defined Sequences appear in the Main tab and are saved for future sessions. Press Load Sequences (u) to load all your sequences from the last time you pressed Apply.

Scenario Example

Our simple task is easy to teach, thus we can turn off the Expert Mode. Now we select *Start* in the Available Positions list and press Add Position. Then, we select *Place* from Available Actions (of our end effector) and press Add Action. We do the same procedure with the position *End* and the action *Place*. The sequence is then defined. As a next step, we press Follow Sequence and the robot will follow the sequence. We have our emergency stopper close to us in case the robot gets dangerously close to an obstacle. After the end effector has reached the correct positions and has executed the desired actions, we give the sequence an appropriate name and press Confirm Sequence. Finally, we have the desired sequence defined and press Apply.

5. Evaluation

The GUI must provide all functionalities for the configuration of the modular robot and must lead the user intuitively through the configuration process.

This chapter consists of three parts: the first one evaluates whether all necessary configuration functionalities are provided by the GUI. The second part evaluates whether this process is user-friendly and fulfills the requirements that we have defined at the beginning. The third part presents further improvements that are based on the results of the two previous parts.

Evaluation of the functionalities

In order to test the functionalities of the GUI, we have defined three different types of tests, which are illustrated in Figure 5.1. In the following, each test type is explained:

- **Low-Level-Tests:** we have defined a function that activates every single function of the GUI and changes every single input box in a random order. The tests are passed when there is no error message in the MATLAB console.
- **Mid-Level-Tests:** these tests check whether every use case is fulfilled. Hence, these tests are passed when all of the following can be achieved with the GUI:
 - Managing modules (plus defining own modules)
 - Defining positions
 - Choosing configuration criteria
 - Defining task sequences
 - Obtaining a robot configuration (requires the connection to the algorithm)
 - Calibrating the robot (requires the connection to the robot)
- **High-Level-Tests:** finally, we test whether the GUI works as whole, in essence, if the tabs work together. Therefore, the tests are passed if the GUI is able to define the first task from scratch as well as to define another task, when the robot is already built and ready.

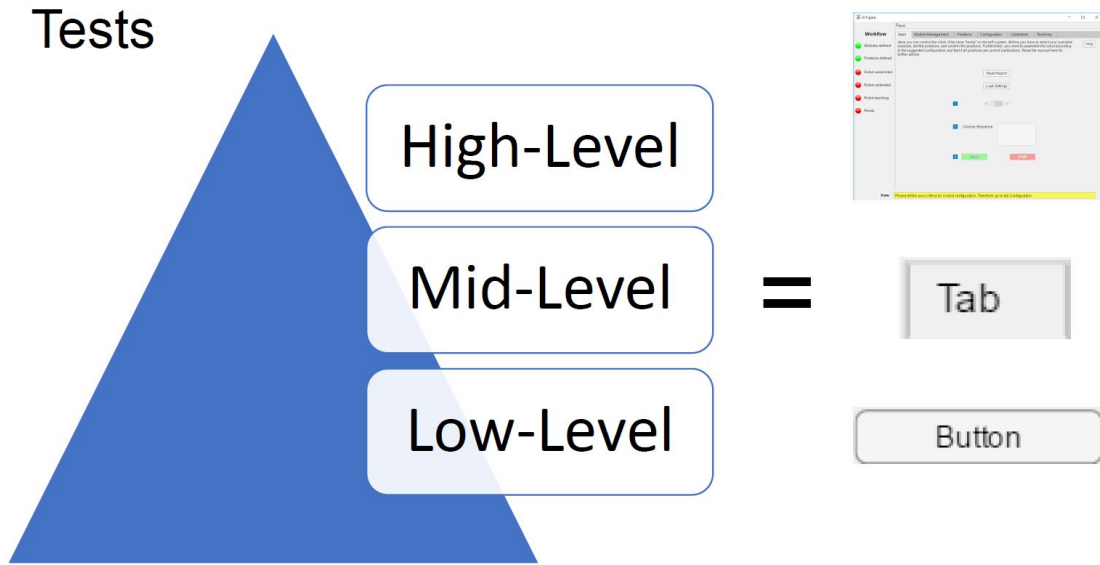


Figure 5.1.: The three different types of tests

As we have laid out the evaluation criteria, we present the results of the tests. The GUI never causes an error within MATLAB and all the potential errors that could occur are processed by providing the user feedback about the cause of the error in the state textfield. The mid-level-tests are passed for all functionalities except the calibration. However, as the modular robot is not built yet, we were not able to test if the calibration works with our GUI. For this reason, we have connected our GUI to a simulation of the modular robot, which was successful. Therefore, we consider this test as passed. Last but not least, we have passed the high-level-tests, as a result the GUI works as a whole. Consequently, our GUI provides all the required functionalities to configure a modular robot. In the following section, we evaluate the user-friendliness of the process.

Evaluation of the user-friendliness

All the requirements regarding user-friendliness have been fulfilled. The only exception that has not been fulfilled completely is requirement (req.) 2; the robot is only prevented from entering a singular configuration, but the user is not warned about it. This will be further explained below.

Besides the requirements, we have three main design goals: clarity, structure, and intuitiveness. The GUI has the structure that the configuration process in each tab always goes from left to right and from top to bottom respectively. Furthermore, the

user is guided through short descriptions at the top of each tab, the lamps on the left, the blue colored numbers, and especially state messages at the bottom. Consequently, the next step is always clear to the user and hence our GUI is considered user-friendly. Nevertheless, the user-interface can still be improved. Later in this chapter, we show how the user-interface can be further improved by changing the GUI itself and also by adjusting the hardware.

While calibrating the robot, the user can directly control it. Nevertheless, as we expect a user without expertise in robotics, he may not know what a singular configuration is, when it is reached and how dangerous it is to be in one. For this reason, we prevent the robot from moving towards singular configurations by slowing it down to a halt at the approach of a singular configuration. However, the user is not informed about why the robot does not move any further (req. 2). The reason of this limitation is that we control our robot with Simulink, which does not output any information (during a simulation).

Concerning Simulink and the general MATLAB environment, there is a further major drawback. The licenses are quite expensive for commercial users. However, we have decided to implement the GUI with MATLAB because the entire *modular robot toolbox* is implemented in MATLAB and the controller is implemented on Simulink real-time. If the GUI was programmed in another programming language, it would be possible to include more complex graphics and make the GUI more intuitive, for example, the Teaching tab (req. 34b). However, given that the aim of the project was to implement a GUI that works and given the hard time constraints, it was the right decision to implement the GUI in MATLAB.

In the weekly meetings, the other team members evaluated our GUI and we incorporated their recommendations. Moreover, the research assistant Julius Jankowski (Julius.Jankowski@DLR.de) at the *DLR Institute of Robotics and Mechatronics* has evaluated our GUI and has called it "very user-friendly". Furthermore, he was impressed by the functionalities that we have incorporated. Unfortunately, we did not have the opportunity to obtain an evaluation from employees of small companies, but only robotic experts.

Further improvements

First of all, we would like to perform experiments on the connection between the GUI and the built modular robot. Considering the hardware requirements, we demand a physical emergency stop button for the robot, as mentioned in the requirement list for the req. I. Additionally, it is desirable for the operator to be able to move the robot when it is turned off (req. II). After the robot is assembled, it must check itself whether it is assembled correctly (req. 16c). As described in Chapter 1, a modular robot has the benefit of easily exchanging modules if they break. For this reason, it is desirable for

the user to obtain a message notifying about the broken module (req. 15a). It is the task of the GUI to convey such a message to the user, but the hardware is required to detect such an error. Moreover, to efficiently integrate the robot into an assembly line, it must have a universal interface to send and receive signals (req. 27). All these requirements could not be fulfilled because there is no working version of our modular robot available yet.

Besides the hardware, we can also improve the GUI itself. As a next step, we would like to obtain an evaluation from small companies in order to have feedback about the intuitiveness of the GUI. Additionally, we would further investigate possibilities to inform the user, when the robot is approaching a singular configuration. Moreover, we would implement the GUI as a web-interface. This has the major advantage that it would run on every platform and every device with a browser, while not requiring any license. In addition, we could include more complex graphics to make the user-interface more intuitive and more clear. As an example, we would change the definition of a sequence in the Teaching tab into a drag-and-drop menu, as illustrated in Table 5.1.

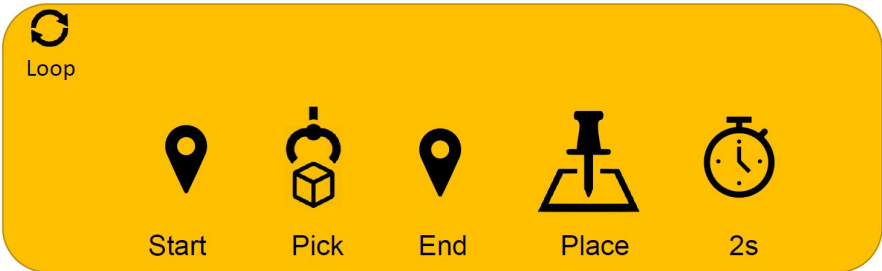
Teaching the robot	
PTP_Start Act_Pick PTP_End Act_Place Wait_sec_2	
Currently	Ideas for future versions of the GUI

Table 5.1.: Comparison of the current version of the sequence within the Teaching tab and ideas for further improvement

6. Conclusion

We present a graphical user-interface (GUI) for the process of configuring the under-development modular robot. We have incorporated all functionalities that are required to configure the robot and lead the user intuitively through this process with a clear structure, the lamps, and the state messages. Moreover, we have passed all the tests regarding the functionalities required for this process and we have fulfilled most of the requirements regarding user-friendliness. Hence, a user can configure the modular robot without any expertise in robotics.

Nevertheless, the GUI can be further improved. First, we would like to understand the user better through the survey we have prepared in Chapter 2 and we would like to have feedback from target users by allowing them to test our GUI. Second, it is desirable to connect the GUI with the built modular robot and to perform experiments on this interaction. Third, we would like to program a web-based interface in order to require less software licenses and to make the GUI more intuitive by including more complex graphics.

7. Division of Workload

In general, the user-interface team has made all decisions together, has defined the requirements together, and the team members always consulted each other. Leonardo has designed the concepts for the GUI, has implemented all tabs, except for the Positions tab and the Calibration tab, and has conducted all the tests. Emanuel has implemented the Positions tab and Calibration tab as well as the required API for V-Rep and Simulink. A detailed division of workload can be found in Table 7.1.

Task	Main Responsibility	
Requirement list	Emanuel	Leonardo
Conception of the GUI		Leonardo
Main tab		Leonardo
Module Management tab		Leonardo
Positions tab	Emanuel	
V-Rep	Emanuel	
Configuration tab		Leonardo
Connection to the algorithm		Leonardo
Assembly instruction	Emanuel	Leonardo
Calibration tab	Emanuel	
Simulink	Emanuel	
Connection to the robot (control)	Emanuel	
Teaching tab		Leonardo
Testing		Leonardo

Table 7.1.: Division of workload

A. V-Rep CAD Software

V-Rep is a 3D-CAD software for robot simulations. It has very good application programming interface (API) support for all major programming languages including MATLAB. For this reason, we used it in our Positions tab in order to define the *Start* and *End* positions intuitively. This section gives a brief overview of V-Rep and conveys some important information that will provide a better understanding of this software.

V-Rep APIs with MATLAB

In general, there are two kinds of API functions available. First, the remote API functions [14] that support MATLAB, Python etc. with reduced functionalities and second, the regular API functions [15] that support Lua and C with full functionalities.

By using MATLAB, we are considerably constrained in available functionalities. However, there is the `simxCallScriptFunction`, which allows to call a script in the V-Rep environment, which has access to the regular API functions. If you want to see an example of such a function, it can be found in the `VREP_Standard_Environment.ttt` file in the repository. These functions need to be written in Lua.

It is also helpful to have a look into the remote API constants [16], where each function takes an operation mode and will return an error code according to the success of its execution. For the implementation in the Positions tab, it is important to operate some functions in blocking mode, which means that the function will wait until it receives a feedback. In other cases, it is better to use another mode that does not wait for feedback, since this will slow down the program.

Example

To make the remote API work, you need to include the files `remoteApi.so`, `remoteApi.dll` and the MATLAB specific files `remApi.m`.

Then you can start the files in Appdesigner with the following code:

```
app.vrep=remApi('remoteApi');  
app.clientID=app.vrep.simxStart('127.0.0.1',19997,true,true,...  
5000,5);
```

This creates a vrep object containing all possible API commands and the clientID object containing address and port information of the API connection as well as timeout parameters and connection speed.

Now for inserting an object into V-Rep, the following code is used:

```
%Insert the Gripper into the VREP
res retInts retFloats retStrings ...
retBuffer =app.vrep.simxCallScriptFunction(app.clientID , ...
'remoteApiCommandServer',app.vrep.sim_scripttype_childscript , ...
'loadFile',[],[rand 0 rand], ...
[app.gripperPath 0 app.PositionListBox.Value 0] ...
,[],app.vrep.simx_opmode_blocking);
```

We access each API function in the vrep object and give it the clientID. Then, this specific function needs the object in the V-Rep environment that contains the Lua script. Afterwards, you need to specify the type of script and the name of the function you want to call loadFile. Then, you are able to transfer data to the function via the four arrays, which contain integers, floating point number, strings and a buffer in this order. For the strings, make sure to separate them in the array using a zero. Finally, you need to specify the operation mode. Here the blocking mode is chosen to ensure the completion of the function.

The above code will call the following function within V-Rep:

```
loadFile = function(inInt , inFloat , inStrings , inBuffer)

    vertices , indices , reserved , names \z
    =sim.importMesh(4 , inStrings [1] , 0 , 0.0001 , 1)
    if ( vertices ) then
        for i=1,#vertices ,1 do
            h=sim.createMeshShape(2,2 , vertices [ i ] , indices [ i ])
            sim.setShapeColor(h,"",sim.colorcomponent_ambient,\z
                { inFloat [1] , inFloat [2] , inFloat [3] })
            objHandle = sim.getObjectHandle('Shape')
            sim.setObjectName(objHandle , inStrings [2])
        end
    end
end
```

As you can see, the function takes the same arrays as input that has been specified previously. The importMesh function is a regular API function that imports an *.stl file. Then, the imported mesh data is used to create a mesh object in the V-Rep

environment in the color specified by RGB values. Also an object handle and a name are assigned, which are taken from the string array. Now the object will appear in the V-Rep environment.

Of course, this overview does not cover everything, but hopefully helps to avoid some pitfalls for future development and offers an entry point into the extensive documentation of V-Rep.

B. Requirement List

Requirements

Building a Modular Robot: User-Interface

Contents

I – Safety	39
Hardware	39
Control	39
User-Interface	39
II – System Requirements	39
III – Aims of the Interface	40
V – Interaction with the Interface	41
VI – Assembly of the Robot	41
VII – Adjustment of the Robot	42
VIII – Operation of the Robot	43
IX – Sources	44

I – Safety

Safety is by far the most important category for requirements. It goes beyond the scope of user-interface and therefore it is divided into three parts: hardware, control, and user-interface.

Hardware

- I. The robot must have an emergency stop.
- II. The robot must be movable even without operating power (turned off).
- III. Dangerous components must be protected by covers.

Control

- i. After a loss of energy, the robot should not directly move after turning it on.
- ii. The robot should move slowly ($< 2\text{m/s}$) during its adjustment to avoid damage.
- iii. The robot should have a standby position that is not dangerous to itself and others.

User-Interface

1. The robot interface should allow the control of the robot only from one terminal (and not several computers).
2. If singularities can lead to dangerous movements, the user must be warned before reaching a singular configuration.
3. The robot may not move without user permission.
4. One can set a limit to the available space for the robot.
5. The robot must support different modes (manual, automatic) and the current mode must be clearly visible.

II – System Requirements

6. MATLAB R2018a with
 - a. Robotics System Toolbox
 - b. Simulink
 - c. Simulink Real-Time
7. Windows for Simulink Realtime
8. VREP (optional)

III – Aims of the Interface

The interface should be designed such that:

9. The interface is user-friendly: The robot must be clearly and thus efficiently to apply.
10. The robot is satisfying in the application.
11. The dialog with the robot is the following:
The user notices the information via on or several output devices of the robot.
12. The planning of human-centered design contains the following:
 - a. Definitions of the methods to integrate the user action
⇒ Computer Mouse
 - b. Identification of the persons and organization that are responsible for the human-interface design
⇒ Leonardo Lerchenfeld and Emanuel Buchholz
 - c. Development of effective methods to set up messages to exchange information via human-centered actions
⇒ State within the GUI
 - d. Definition of milestones for human-centered design actions
 - i. Requirement list
 - ii. Draft of GUI
 - iii. Interfaces to hardware
 - iv. Interface to human → programming the GUI
 - v. Integration into the robot system
 - vi. Establish a manual for the easy application
 - vii. Evaluation
 - e. Definition of times to respect project plan
 - i. GUI ready until end of June 2018
 - ii. Integration into the robot system during August 2018
13. The following principles are respected for the design of interactive systems:
 - a. suitability of the task
 - b. ability to describe itself
 - c. evidence of user-expectations
 - d. suitability for learning
 - e. controllability
 - f. tolerant for errors

V – Interaction with the Interface

14. Input from user:

- a. Which kind of modules are available and the amount of them.
- b. Coordinates (the origin of the robot base, positions of the end effector and an obstacle model).
- c. Robot composition criteria.
- d. Robot calibration through control panel.
- e. Definition of the sequence of positions and actions.

15. Output to the user:

- a. **Hardware health checks results.**
- b. Suggestion for robot composition of different modules.
- c. Check if the modules are assembled correctly.
- d. Robot State.

16. Input from the hardware:

- a. **Hardware health check**
- b. **Order of the assembled modules**
- c. **Check if modules are correctly assembled**

17. Input for the algorithm

- a. What modules are available
- b. Start and end position of the robot sequence
- c. Base position of the robot
- d. Obstacle

18. Output from the algorithm:

- a. Robot composition with ordered module for assembly order of the robot.
- b. Criteria for each robot composition.

19. While calculating a robot configuration, the user can see the progress of the calculation.

VI – Assembly of the Robot

20. The program is able to work with a CAD-model that contains

- a. the robot base position,
- b. positions for the end effector, and
- c. obstacles

roughly (+/- 1 cm) marked, to recommend several combinations of modules.

21. The boundary of the working space can be defined.

22. Entry and free space must be defined:

- a. Necessary free space around obstacles
 - b. Accessibility for operator
 - c. Reasonable and clear contact between parts of the robot system and the operator
23. The modules can be assembled with standard tools → cross-tip screwdriver
24. The program must hint to the order of the modules-assembly.
25. For the assembly, an understandable instruction should be available.
26. The connection of the modules should be so clear that a wrong assembly is not possible.
27. The communication from the production plant to the robot should be possible via a universal interface to receive signals from other machines in the plant and to send signals to other machines.
28. The robot should recognize the order of modules, how they are assembled and send this information to the central control unit.

VII – Adjustment of the Robot

29. In the main menu of the terminal the robot has the following submenus:
- a. Select the available modules including the definition of own modules
 - b. Add new position (to “learn” a new position)
 - c. Define robot composition criteria
 - d. Show solutions for robot compositions
 - e. Edit sequence of motions (to edit the production steps of the robot)
 - f. Work (the robot is doing his job)
 - g. Shut down (go in save position and stop consuming energy)
30. Add new position:
- 1. The user loads the 3D model (CAD) of the work cell.
 - 2. The user loads the 3D model (CAD) of the end effector and selects its type. The type defines the actions of the tool that can be selected later in “edit sequence of motions”.
 - 3. The user sets user points in the 3D model that can be extracted by the program. The coordinates can be used for the generation of the trajectory and the selection of modules. Furthermore, the user should be able to select the modules he already has.

- 31. The user should assemble the robot with a help of the manual. The robot recognizes if it is correctly assembled.
- 32. To define the exact positions the robot can be controlled intuitively via inverse kinematics thus no control of the single joints. For this procedure the robot has the following actions:
 - a. select position
 - b. forward / backward
 - c. left / right
 - d. up / down
 - e. pitch (bidirectional)
 - f. yaw (bidirectional)
 - g. roll (bidirectional)
 - h. stop
 - i. confirm
- 33. The robot should confirm the aim positions by moving there in its energetic best configuration.
- 34. Edit sequence of motions
 - a. The user can choose the sequence of motions from signals, the learnt positions and the available actions of the tool.
 - b. The teaching of the robot is visualized by icons.**
- 35. The robot should calculate its trajectory by itself.

VIII – Operation of the Robot

- 36. Ergonomics and Human-Machine-Interface
 - a. Clear operating elements
 - b. Error through operator
 - c. Required education and abilities of the operator
 - i. read manual
 - ii. instruction through a qualified person
- 37. Boundary of application:
 - a. description of the tasks including the required education of the operator
- 38. Plant level (display, layout)
 - a. Terminals (Display-Layout)
- 39. The state of the robot can be read out all the time. The following state messages must be sent:
 - Ready for calibration
 - Doing calibration

- Ready to work
- Working
- Current assembly is not possible
- module X broken
- Please clear the working space
- Please refill motor oil
- Critical error: please contact your supplier

40. The robot should have the following interaction functions:

- Select modules
- Add new modules
- Add new position
 - confirmation of new position
- Define configuration criteria and calculate a suitable configuration
- calibrate
- edit sequence of motions
 - start position
 - intermediate position(s) (optional)
 - end position
- turn on / shut down
- work

IX – Sources

VDI/VDE 3694: Lastenheft/Pflichtenheft für den Einsatz von Automatisierungssystemen

DIN EN 1332-1: Identifikationskartensysteme – Mensch-Maschine-Schnittstelle – Teil 1:

Gestaltungsgrundsätze für die Benutzerschnittstelle

DIN ISO/TS 15066 Roboter und Robotikgeräte – kollaborierende Roboter

DIN EN ISO 10218-1 Industrieroboter - Sicherheitsanforderungen: Teil 1: Roboter

ISO 9241-210 Ergonomie der Mensch-System-Interaktion

Glossary

***.stl** File format for 3D CAD software. first.

MYO Myorobotics envisions the use of biological principles such as modularity and reconfigurability in the design and construction of robotics systems for the future innovation of the robotics industry.

V-Rep 3D CAD Software provided by Coppelia Robotics Inc.

work cell Convex Space containing at least every point a robot can reach. In this report a 3D-rendering of the robots work environment is meant.

Acronyms

API application programming interface.

GUI graphical user interface.

PTP Point-to-point.

req. requirement.

TUM Technical University of Munich.

List of Figures

1.1. CAD model of a possible robot configuration	2
2.1. Human-centered design process [8]	4
2.2. First concept for module management	6
3.1. Use case diagram for the GUI	7
4.1. General design of the GUI	12
4.2. Tab: Main	14
4.3. Tab: Module Management - Define Own Module	16
4.4. Tab: Module Management - Module Catalog	17
4.5. Tab: Positions	19
4.6. Defining the positions within the CAD work cell in V-Rep	20
4.7. Tab: Configuration	21
4.8. Assembly Instruction: How to Connect a Joint to a Link	23
4.9. Tab: Calibration	25
4.10. Tab: Teaching	26
5.1. The three different types of tests	30

List of Tables

4.1. The influence of the different optimization criteria on the robot composition	22
4.2. Description of robot trajectories [13]	27
5.1. Comparison of the current version of the sequence within the Teaching tab and ideas for further improvement	32
7.1. Division of workload	34

Bibliography

- [1] A. Giusti and M. Althoff. "Automatic Centralized Controller Design for Modular and Reconfigurable Robot Manipulators." In: *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2015.
- [2] G. D. and M. P. and. "Building a modular robot - mechanical design." In: *Master-Praktikum Building a modular robot WS 2017/18*. 2018.
- [3] H. Krasowski and F. Skiba. "Development and Design of Modular Robot Links." In: *Lab course: Building a Modular Robot - TUM Chair of Robotics, Artificial Intelligence and Real-time Systems*. 2018.
- [4] D. Perez and A. Zhu. "Building a Modular Robot: Algorithm Design." In: *Lab course: Building a Modular Robot - TUM Chair of Robotics, Artificial Intelligence and Real-time Systems*. 2018.
- [5] E. Icer, H. A. Hassan, K. El-Ayat, and M. Althoff. "Evolutionary Cost-Optimal Composition Synthesis of Modular Robots Considering a Given Task." In: *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2017.
- [6] E. Icer and M. Althoff. "Cost-Optimal Composition Synthesis for Modular Robots." In: *Proc. of the IEEE Multi-Conference on Control Applications (CCA)*. 2016, pp. 1408–1413.
- [7] E. Icer, A. Giusti, and M. Althoff. "A Task-Driven Algorithm for Configuration Synthesis of Modular Robots." In: *Proc. of the IEEE International Conference on Robotics and Automation*. 2016.
- [8] D. D. I. für Normung e. V. *DIN EN ISO 9241-210:2010 Ergonomie der Mensch-System-Interaktion – Teil 210: Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme*. Beuth Verlag GmbH. 2011.
- [9] V. D. Ingenieure. *VDI/VDE 3694: Lastenheft/Pflichtenheft für den Einsatz von Automatisierungssystemen*. Beuth Verlag GmbH. 2014.
- [10] D. D. I. für Normung e. V. *DIN EN 1332-1: Identifikationskartensysteme – Mensch-Maschine-Schnittstelle – Teil 1: Gestaltungsgrundsätze für die Benutzerschnittstelle*. Beuth Verlag GmbH. 2009.

- [11] D. D. I. für Normung e. V. *DIN ISO/TS 15066 Roboter und Robotikgeräte – kollaborierende Roboter*. Beuth Verlag GmbH. 2017.
- [12] D. D. I. für Normung e. V. *DIN EN ISO 10218-1 Industrieroboter - Sicherheitsanforderungen: Teil 1: Roboter*. Beuth Verlag GmbH. 2012.
- [13] Research and D. institute Lola Ltd. *Description of the robot movement*. 2018. URL: <https://www.lira.rs/en/robotics/robot-controllers-and-industrial-robot-language/1-irl-and-ide/description-of-the-robot-movement> (visited on 08/04/2018).
- [14] C. R. Inc. *Remote API Functions Matlab*. 2018. URL: <http://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsMatlab.htm> (visited on 08/16/2018).
- [15] C. R. Inc. *Regular API Functions*. 2018. URL: <http://www.coppeliarobotics.com/helpFiles/en/apiOverview.htm> (visited on 08/16/2018).
- [16] C. R. Inc. *Remote API Constants*. 2018. URL: <http://www.coppeliarobotics.com/helpFiles/en/remoteApiConstants.htm#functionErrorCodes> (visited on 08/16/2018).