ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
FACULTY OF INFORMATICS

DIPLOMA WORK

OE-NIK	Student's name:	**Zsoldos Panna**
2023	Student's registration number:	**T009479/FI12904/N**

Óbuda University
John von Neumann Faculty of Informatics
Institute of Biomatics and Applied Artificial Intelligence

DIPLOMA THESIS
TOPIC DESCRIPTION

Student name: **Zsoldos Panna**
Registration number: T009479/FI12904/N
Neptun's code: DFX6VN

Title of diploma thesis:

Integration of mobile robots into the SiLA 2 laboratory automation framework

Internal supervisor: Dr. Galambos Péter
External supervisor Wolf Ádám

Deadline: 15th of May, 2023

Topic description:

Mobile robots are now widely used in a variety of environments. As a result, there is a growing need for standardized control environments. A specific but relatively broad application area is laboratory automation. The Standardization in Laboratory Automation (SiLA) Consortium has created a standard for such initiatives to simplify the integration of different subsystems. It also provides a convenient, easy-to-use framework for subsystem developers and other stakeholders.

The aim of the task is to develop and demonstrate the capabilities and their use based on SiLA 2. In the sample environment to be implemented, the subsystems will communicate in a standard way and will operate correctly in a dynamic environment. The device potentially involved in the implementation is the PlatypOUs mobile robot, an open-source robot developed at the Antal Bejczy Center for Intelligent Robotics (IROB) at the University of Óbuda.

The diploma thesis should cover:

- A review and summary of the literature on the subject.
- Overview of SiLA 2 operation, description of existing uses.
- The system design of the software environment.
- The design of the test environment.
- A detailed description of the implementation process.
- Testing the system for specific tasks.
- Developer and user documentation in sufficient detail.
- Evaluation of the results and opportunities for further improvement.

Ph.

ÓBUDAI EGYETEM * Neumann János Informatikai Kar. * 1. Budapest

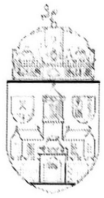
.....
Dr. György Eigner
head of institute

Term of limitation: **15th of May, 2025.**
ÓE HKR Section 54, paragraph (10)

The diploma thesis is adequate and can be submitted:

.....
external supervisor

.....
internal supervisor



ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Faculty of Informatics

STUDENT'S DECLARATION

I the undersigned student hereby declare that this diploma work is the result of my own work; I have disclosed the references and tools used in an identifiable manner. The results indicated in my diploma work completed may be used by the university and the institution announcing the task for their own purposes free of charge.

Dated: Budapest, 2023.05.12.

Zoldos Panna

student's signature

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

John von Neumann Faculty of Informatics

Log of consultations

Name of the student:	Neptun code:	Program:
Zsoldos Panna	DFX6VN	Master training
Phone number:	Address:	
+36706149487	2030 Érd Darukezelő street 58.	

Title of the thesis work:

Integration of mobile robots into the SILA 2 laboratory automation framework

Internal supervisor:	External supervisor:
Dr. Galambos Péter	Wolf Ádám

Please use BLOCK CAPITAL LETTERS to fill in this form!

Ocassion	Date	Topics discussed	Signature
1.	2022.02.23	GOALS FOR THE SEMESTER	
2.	2022.03.16	REVIEWING RESOURCES	
3.	2022.04.13	CONTENT REFINING	
4.	2022.05.04	SYSTEM DRAFT DISCUSSION	

This log should be signed on each consultation (by a supervisor), 4 times altogether.

The student fulfilled the requirements of the Thesis work subject, I allow his/her participation on the presentation/defence[1].

Budapest, 2022.05.19.

Internal supervisor

[1] Underline the appropriate one.

Log of consultations

Name of the student:	Neptun code:	Program:
Zsoldos Panna	DFX6VN	Master training
Phone number:	Address:	
+36706149487	2030 Érd Darukezelő street 58.	

Title of the thesis work:

Integration of mobile robots into the SILA 2 laboratory automation framework

Internal supervisor:	External supervisor:
Dr. Galambos Péter	Wolf Ádám

Please use BLOCK CAPITAL LETTERS to fill in this form!

Ocassion	Date	Topics discussed	Signature
1.	2022.09.26	GOALS FOR THE RESOURCES	
2.	2022.10.17	DISCUSSION OF THE IMPLEMENTATION PLAN	
3.	2022.11.07	IMPLEMENTATION PROCESS CONTROL	
4.	2022.11.21	DISCUSSION OF FURTHER TASKS	

This log should be signed on each consultation (by a supervisor), 4 times altogether.

The student fulfilled the requirements of the Thesis work subject, I allow his/her participation on the presentation/defence[2].

Budapest, 2022.12.09.

Internal supervisor

[2] Underline the appropriate one.

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Faculty of Informatics

CONSULTATION LOG

Student's name:	Neptun code:	Specialty:
Zsoldos Panna	DFX6VN	Robotics
Phone number:	Mailing address (e.g. domicile):	
+36706149487	2030 Érd, Darukezelő street 58.	

Diploma work title in Hungarian:

Mobil robotok integrációja a SILA 2 laboratórium-automatizálási keretrendszerbe

Diploma work title in English:

Integration of mobile robots into the SILA 2 laboratory automation framework

Internal supervisor:	External supervisor:
Dr. Galambos Péter	Wolf Ádám

Please write the data in printed capital letters!

Occ.	Date	Content	Signature
1.	2023.03.06.	SETTING UP GOALS FOR THE SEMESTER	
2.	2023.03.22.	FINALIZING IMPLEMENTATION	
3.	2023.04.12.	TESTING AND REVISIONING	
4.	2023.04.26.	FINISHING TOUCHES AND REVIEWING THESIS	

The Consultation log is required to be countersigned by either supervisor on a total of 4 occasions of consultation.

The student has complied with the requirements of the subject titled Thesis 4, (MSc) and is allowed to proceed for defense.

The mark recommended by the supervisor: excellent (5)

.....

internal supervisor

Budapest, 2023.05.11.

ABSZTRAKT

A legújabb fejlesztések és a kifinomult laboratóriumi berendezések iránti növekvő igény a laboratóriumi automatizálás szükségességét eredményezte. Mivel a laboratóriumi berendezések és a robotika területén működő számos vállalat különböző javaslatokat hozott létre, hiányzott egy szabványosított, nyílt forráskódú megoldás.

Ez vezetett a Standardization in Lab Automation (SiLA) konzorcium létrehozásához, ahol szabványokat és interfészeket dolgoznak ki a laboratóriumi környezetekbe történő gyors eszközintegrációhoz [1].

A SiLA 2 keretrendszer felhasználásával a mobil robotok segítségével történő, állomások közötti mikrolemez-szállítás demonstrációs megvalósítása a cél. A kiválasztott mobil robot a PAL Robotics TIAGo++ mobil manipulátora [2] [3].

A szállítás szekvenciája két állomás között történik, ahol a robot szabadon navigálhat és elvégezheti a mikrolemez felvételének és elhelyezésének feladatát, tehát a transzportálást. A magas szintű parancsok képesek a teljes szekvencia vagy annak egyes részeinek automatikus futtatására a SiLA Universal Client segítségével, amely egy webböngészőn keresztül elérhető grafikus felhasználói felület [4].

A projekt fejlesztése és tesztelése két különböző laboratóriumban valósult meg, az egyik egy multinacionális gyógyszeripari vállalat (a továbbiakban: "a Vállalat") osztrák R&D telephelyén, a másik az Óbudai Egyetem Bejczy Antal Intelligens Robotikai Központjának (IROB) Robotikai Laboratóriumában. A tesztkörnyezetek mindkét helyen kialakításra kerültek.

A munka áttekintést nyújt a háttérben álló motivációkról és a laboratóriumi automatizálás legújabb eredményeiről, a felhasznált technológiákról, mind általánosságban, mind pedig a projektben részt vevő robot, egy TIAGo++ mobil manipulátor jelenlegi megvalósítása tekintetében. A robot használatát, funkcióit és tulajdonságait jól ismertetjük. A tervezési és megvalósítási fázisok és lépések dokumentálva vannak, az eredmények és a továbbfejlesztési javaslatok pedig a projekt lehetséges folytatását mutatják.

ABSTRACT

Recent advances and the increasing need for sophisticated laboratory equipment have led to laboratory automation. As many companies working in laboratory equipment and robotics created different proposals, there was a lack of a standardized, open-source solution.

This led to the creation of the Standardization in Lab Automation (SiLA) consortium. They are developing standards and interfaces for rapid device integration in laboratory environments [1].

Using the SiLA 2 framework, a demonstration implementation of microplate transportation between stations using mobile robots was set as the goal. The mobile robot chosen is the TIAGo++ mobile manipulator from PAL Robotics [2] [3].

The transportation sequence is between two stations where the robot can freely navigate and perform the task of picking and placing the microplate, i.e. transporting it. High-level commands can call the whole sequence or parts of it to run automatically through the SiLA Universal Client, a graphical user interface accessed through a web browser [4].

The development and testing of the project have taken place at two different laboratories, one at the Austrian R&D site of a multinational pharmaceutical company (hereafter referred to as "the Company") and the other at the Robotics Laboratory of the Antal Bejczy Center for Intelligent Robotics at the University of Óbuda (IROB). The experimental laboratory environments have been set up at both locations.

The work presents an overview of the background motivations and recent advances in laboratory automation, the used technologies thoroughly in a general manner as well as specifically for the current implementation for the robot taking part in the project, a TIAGo++ mobile manipulator. The usage, functionalities, and properties of the robot are presented. The planning and implementation phases and corresponding steps are documented and the results and suggestions for further development reveal a possible continuation of the project.

CONTENTS

1 INTRODUCTION

During the 20th-century robotic laboratory automation has undergone significant advancements and created machines like automated teller machines, vending machines, and electronic switchboards. Technological devices are increasingly replacing humans because of some important advantages like accuracy, speed, convenience, and cost [5]. As for the 21st century, automation with robotics is ready to change the laboratory environment.

Unimate was the first die-casting industrial robot in 1961. This started a "technological revolution" in robotics which overlapped with the advances in computing [6].

As computer-based control methods improved, the size of robots got smaller, and as the United States space program got more developed in the 1960s and 1970s, new robotic devices arose. This got clinical laboratories more and more interested in applying them.

In the early 1980s programmable, compact, and microprocessor-controlled robotic arms were introduced in the field. A robot arm with interchangeable hands was developed by Zymark Corporation [5]. This allowed the development of such laboratory workstations that were capable of carrying out programmable and multi-step sample handling processes. Being programmable made these devices adjustable to several research and sample-handling approaches.

This new generation of robots got applied almost instantly for preparing analytical samples as well as for improving the efficiency and stability in the pharmaceutical industry.

In the 1980s, a laboratory was founded at the Japanese Kochi Medical School, initiated by Dr. Masahide Sasaki [5]. The laboratory had robots, which handled test tubes, a conveyor belt, which transported patient samples to various analytical workstations, and automated pipetting robots, which collected samples for necessary testing. At some workstations, single-arm stationary robots performed pipetting and dosing steps to carry out more complex analytical tasks. The workstations were linked together by conveyor belts and operated without human intervention.

By the early 1990s, Sasaki's vision had helped to facilitate the automation of clinical laboratories through the use of robotic implementations. In clinical laboratories, Sasaki's approach and its extensions became known as Total Laboratory Automation (TLA), while remote, personnel-free laboratories and labs using manual clinical analyzers became known as Point-of-Care (POC) automation [7].

The lack of standards for communication between robotics and laboratory devices posed an additional barrier to the acceptance and implementation of robotic automation. Without such standards, laboratories would not be able to easily integrate a wide range of

laboratory devices and automated robots from different manufacturers.

Clinical laboratories have been slow to adapt to robotic automation, as most medium-sized hospital laboratories, which are typically handling no more than 2,500 samples a day, have found it difficult to justify the purchase of multi-million dollar systems.

In contrast, pharmaceutical companies invested significant amounts in robotic automation to increase their drug discovery rates, which saw three to five times growth. As these companies earned most of their revenue from new drugs, justifying the costs was easier.

There are already some trends visible in clinical laboratory robotics [5]. Fewer standalone robot arms are used since manipulator's arms for sampling from conveyor belts are often directly integrated into clinical analyzers. Mobile robots that transport laboratory samples use more sophisticated ultrasound and infrared wave-based navigation technology than early models that simply followed painted lines on the floor. These models successfully navigate complex hospital corridors and elevators.

Besides hardware development, a lot of attention is drawn toward the design of process control software, which is capable of controlling and integrating various automation components. Such software is needed for sample transportation and storage, as well as to support automatic repeating and tracking strategies.

For an overview of the development of laboratories, see the article [5].

2 LABORATORY AUTOMATION

Automation is already quite widespread in biotechnology and pharmaceutical research, especially in high-throughput applications where a huge number of samples are processed. Automatic pipetting robots, so-called liquid handlers, are commonly used for this purpose and are considered a relatively mature technology.

Laboratory automation generally includes tools, processes, or software that minimize the need for human intervention. The spectrum of automatable tools includes liquid handlers, centrifuges and sample handlers, carousel storage systems, robots and conveyor belts, workflow control, instrument control, and data acquisition software.

In laboratories where sample processing and screening are done in large quantities, automation, and robotics are inevitable. Such screening procedures include sample preparation, chromatography, and optical reading, where small amounts of liquid are transferred to carriers and processed in multiple steps. Since the samples can have different physical states and shapes, the manipulators need to have various configurations.

Laboratory automation cannot be achieved without the appropriate software for controlling and monitoring processes and workflows. Laboratory robots are expected to collaborate with humans, and safety is a critical aspect of this.

Wolf et al. present a general review on robotics in laboratory automation [8].

2.1 Total Laboratory Automation

In the Total Laboratory Automation (TLA) model, many analyzers performing different types of tests are physically integrated as a modular system or connected to form a linear process, similar to an assembly line. The ability to integrate multiple diagnostic disciplines on a single line has been shown to be useful in a variety of factors. These include improving the efficiency, organization, standardization, quality, and safety of laboratory testing, while also providing a significant return on investment in the long term and allowing for staff retraining. The development of the TLA model also raises some potential issues alongside the many benefits, which are summarised in the [9] and [10] research papers.

Although there is no single definition, laboratory automation is generally classified according to the complexity of instrument integration, from no automation to partial laboratory automation to total laboratory automation (TLA). An example is shown in figure 2.1.

In the extended models of TLA, many analytical and post-analytical steps (e.g. sam-

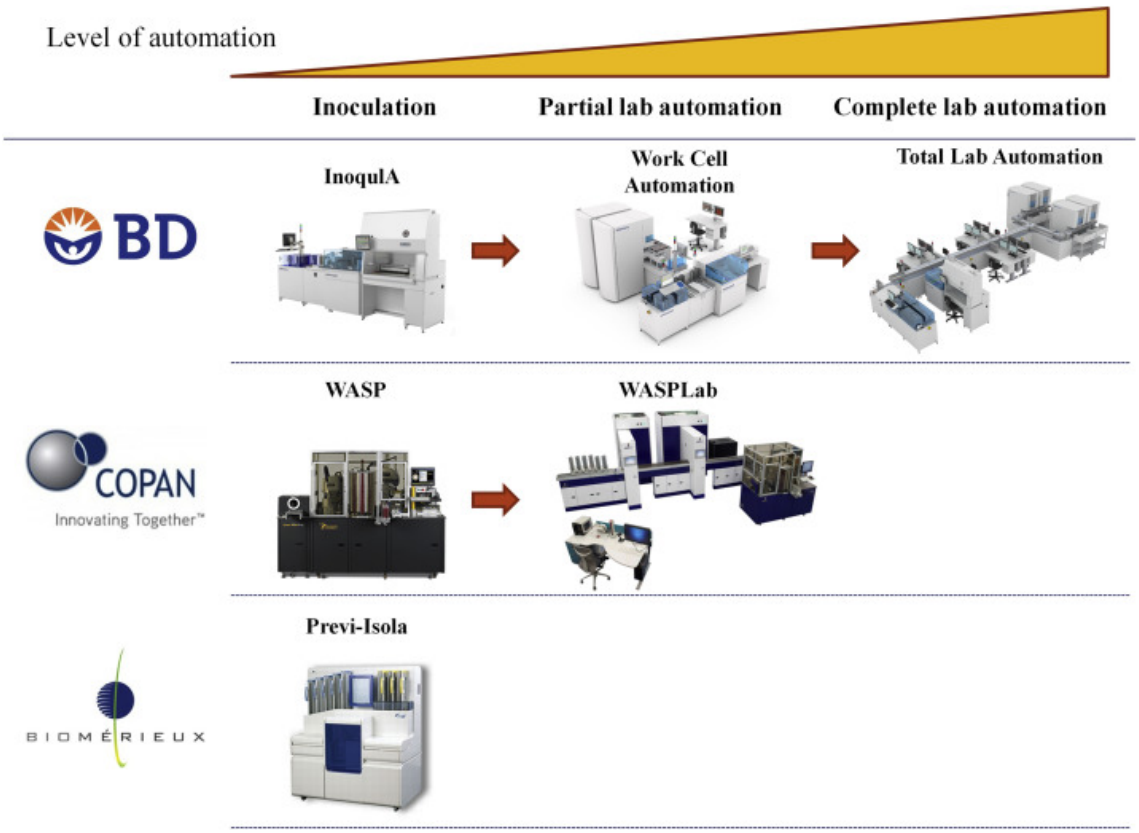


Figure 2.1: Examples for automation level classifications, [11]

ple introduction, logging, sorting, cap removal, centrifugation, separation, sub-sampling, sealing, and storage) are performed automatically on workstations physically connected to the analyzers and efficiently managed by software.

2.1.1 Advantages

Long-term lower costs

An effective TLA model can successfully reduce the cost of laboratory diagnostics, with a net benefit (i.e. return on investment) that can be effectively demonstrated in the long term.

In essence, TLA's significant economic gains from the consolidation of multiple diagnostic platforms are not only a reduction in the physical labor required to manage the high volume of testing but can also be attributed to reduced pre-analytical and post-analytical costs.

Reducing congestion

The reduction in the number of staff required to perform the same number of tests after

the introduction of TLA will lead to a reduction in laboratory congestion.

In fact, an optimized layout of integrated workstations would eliminate the need for technicians to move from one analyzer to another, minimizing the distance staff has to travel to perform tasks on multiple, different instruments.

Increasing efficiency

In addition to the cost reduction benefits that are particularly appreciated by healthcare decision-makers and administrators, TLA also offers other benefits in the laboratory environment, most of which can be attributed to the use of customizable assembly lines that can be reorganized to meet the specific needs and layouts of different laboratories.

A well-designed TLA can be variably effective in reducing turnaround time (TAT) while increasing laboratory productivity (i.e. throughput).

Some models of TLA are equipped with loading stations where blood tubes can be randomly inserted manually or, more efficiently, with physically linked pneumatic tube systems.

Improved sample handling and traceability

Information technology (IT) has contributed greatly to improving the work and organization of medical laboratories. The latest generation of laboratory equipment is also equipped with advanced software that enables better sample handling.

Decision-making mechanisms based on pre-defined criteria now allow automatic data verification, automatic re-analysis of samples with highly abnormal or suspicious results, and replacement of reflexive and add-on tests. This ultimately contributes to improved data quality and safety.

Sample traceability is continuously improved in all routine and stat testing samples, meaning the turnaround time is less than an hour to get the test results from a specimen, stored in a unique environment, enabling digital tracking of all processes in which a medical tube has been used, from delivery to the laboratory to storage after testing.

Improved standardization for authentication

Continuous monitoring of all stages of the entire testing process is a cornerstone of the quality of laboratory diagnostics, requiring exceptional analytical activities.

It is now widely acknowledged that integrating various diagnostic areas into a single workspace would reduce administrative efforts required for developing and updating Standard Operating Procedures (SOPs). When multiple analyzers are integrated into the same TLA model, several pre-analytical and post-analytical sample handling procedures can be combined [9].

The automation of operations has allowed for increased accuracy and repeatability, which can have an impact on the overall testing process. This would also bring significant benefits in terms of standardization, simplifying the certification and accreditation process.

Improve the quality of testing

Standardization and harmonization are critical aspects of laboratory diagnostics, as they can improve the quality of the testing process and reduce the risk of diagnostic errors, particularly those arising from manual tasks in the pre-analytical phase.

Better integration of test results

Significant advances in information technology allow laboratory staff to navigate and manage the flow of data from the delivery, analysis, and archiving systems. The middleware of most TLA models allows the integration of a wide range of test results generated by different analyzers.

This not only allows more comprehensive, detailed, and accurate validation criteria to be established for automation but also provides laboratory staff with a more comprehensive view of the patient's results.

Reduced biological risk for operators

Industrial automation offers a significant advantage when it comes to worker safety. By eliminating the need for operators to be physically present in the work area, automated systems effectively protect them from the potential hazards associated with performing tasks involving bio-hazardous materials.

Minimization of workers

Minimizing manual work is one of the main benefits of TLA, resulting in a net saving in the number of staff (technical or support) required to manage laboratory workflows.

2.1.2 Limitations

Short-term higher costs

The investment required to implement TLA will inevitably involve an initial increase in the cost of the system installation project and new hardware. This can be a problem in some facilities where the budget allocated by hospital management to a laboratory for a new investment does not cover the amount required.

Increased maintenance costs

The introduction of the new hardware, which essentially consists of analytical workstations, assembly lines, and sample storage units, has additional costs associated with the operation and maintenance of the system. This becomes increasingly expensive as the TLA model grows.

Constraints on space and infrastructure

Incorporating several analyzers and new hardware into a pre-existing environment can present difficulties, particularly if the building was not originally designed for that purpose. It is often simpler to construct a new space rather than renovate an old one, particularly if the building's infrastructure is outdated. In such situations, flexible TLA models may offer more benefits. As a solution, utilizing mobile manipulators is a potential answer to this challenge as they can operate in an unstructured, human-optimized environment so their use can be involved in existing configurations.

Noise, heat, and vibration are increased

The clustering of multiple analyzers in a single area results in a concentration of noise, heat, and vibration, which can lead to amplified heating and heightened exposure to acoustic or electrical noise within the workplace.

Increasing downtime risk

The more complex the system, the greater the risk that a system failure will have serious consequences. In the event of critical system failure, manual sample recovery procedures are required.

To address this issue, appropriate measures such as backup power, hardware, software, contingency procedures, or even point of care testing (POCT) analysts should be implemented as an effective alternative to minimize downtime.

Psychological dependence on automation

The automation of manual activities has significant consequences, including reduced employee accountability, quick skill decay, and ineffective manual operation when automation fails. Moreover, when automation fails, humans tend to lose confidence in their manual abilities, which can be almost irreversible.

An even greater challenge is the lack of manual skills to perform the activities that have been transferred to automation. This challenge is compounded for young or new staff who may have little experience in manual laboratory work, which can paralyze the laboratory.

Generating potential bottlenecks

Managing urgent testing effectively is a critical challenge in laboratories. As the vol-

ume of routine testing increases, the risk of bottlenecks also increases, which can reduce system productivity and increase turnaround times. One solution is to develop rules and criteria that enable urgent samples to bypass routine samples.

Therefore, the priority criteria must be properly balanced, allowing for changes to be introduced even after the system is up and running.

Disruption of staff trained in specific technologies

The specific background of laboratory professionals to perform increasingly complex tests needs to be significantly enhanced. This will also pose some risks, as the greater the amount of knowledge required, the lower the competence related to the technologies.

Increased workforce flexibility may ultimately contribute to a reduction in skills and qualifications for some tasks, particularly in laboratory services moving to a TLA model with several different diagnostic lines.

In general, automation can lead to a loss of skills and expertise in analytical procedures because it reduces the exposure of laboratory personnel to different experiences.

Risk of switching to a manufacturer-led laboratory

Constructive collaboration with manufacturers and efficient software programs are crucial for the success of highly automated clinical laboratories. To achieve effective TLA, developing strategic relationships with vendors is essential.

The advantages and limitations were based on studies [10] and [9].

2.2 The role of mobile robots

The study [12] mentions the forms of use of mobile robots in the laboratory.

Stationary robots, commonly used in laboratories, are fixed and can only operate within a limited workspace defined by their type. Mobile robots, on the other hand, can theoretically move around in a workspace that is not geometrically limited, but subject to changing environmental conditions. Unlike stationary robots, the workspace and obstacles of mobile robots are undefined, and the workspace can be large and constantly changing.

A "mobile robot" refers to a robot that is capable of moving freely in an unconfined environment. A variety of mobile robot systems are currently available on the market, including the KMR iiwa (Kuka, Augsburg, Germany), the Talos and REEM-C humanoid platforms, as well as wheeled platforms with different equipment (PAL Robotics, Barcelona, Spain) or the proAnt platform (ASTI Mobile Robotics GmbH, Berlin, Germany).

With the introduction of mobile robots, transport and movement tasks can be taken over by human workers. They move in the same space as people, so proper safety precautions are needed to prevent the machines from colliding with or holding up people.

For example, camera-based methods can be used to accurately map the mobile robot's environment. Apart from ensuring collision detection and avoidance, intelligent systems should enable the creation of alternative routes through suitable algorithms to address such situations. Moreover, direct communication between humans and robots is another aspect that can be implemented via the display or touch screen of mobile systems in the simplest case.

The use of mobile robots in laboratories is still in its infancy, but their integration could lead to the full automation of life science laboratories, as described in [12].

Recent developments include the use of flexible mobile manipulators for complex tasks, according to [13]. This involves adding one or more robotic arms to the basic mobile robot. This opens up many more possibilities for their use and provides more versatile interactions with their environment.

3 TECHNOLOGICAL BACKGROUND

3.1 SiLA

3.1.1 Basic concepts

SiLA (Standardisation in Lab Automation) is an emerging standard, which aims to define the interface between sample processing tools and the automation software system. In the article [14], the authors provide an overview of the standard specifications developed during the first two years of its life cycle.

There have been previous attempts to standardize the lab, reported in [15], [16], but none of the earlier ones have become truly widespread.

Liquid handling robots and analytical instruments are commonly used in traditional pharmaceutical research laboratories, often as separate entities. Although the integration of one or two instruments can result in improved efficiency and continuous operation, several factors hinder the cost-effective and rapid integration of these instruments into fully automated systems. High costs, long implementation times, and inflexible systems are all barriers to entry for smaller laboratories.

An automated laboratory includes instruments for sample preparation and analysis, such as incubators, pipettes, dispensers, and plate readers. These instruments are linked to a software system that streamlines the workflow from set-up, but without the use of a standard, they have a number of disadvantages. These disadvantages include:

- different interfaces for each device
- completely different command sets for devices with the same or similar functions
- significant differences in timing and structural behavior
- different status descriptions for each device
- huge differences in error messages and error handling capabilities of the devices

To address these issues, the SiLA Consortium (Standardisation in Lab Automation), a consortium of instrument suppliers, integrators, and pharmaceutical companies, was formed in 2008 to standardize laboratory automation networking protocols. Simplifying and accelerating the integration of laboratory instruments into drug discovery automation systems, enabling small-scale integration across a broader range of laboratories, facilitating the quick replacement of instruments for better performance or newer technologies, and reducing integration, support, and maintenance costs are some of the benefits of this approach.

SiLA standards are based on interface technology that enables multi-user access to de-

vices using simplified, easy-to-understand commands and common data formats. Specific features and requirements can be easily integrated within the framework of the standards, but they are not completely rigid and restrictive.

Many device manufacturers have developed their own internal standards and internal driver libraries for a wide range of devices, but none have made them available for public use. The development of a standard interface could eliminate the need for repetitive work in this area.

The official website of SiLA is [1] and its official documentation is [17].

3.1.2 SiLA 1

A universal interface definition for laboratory instrumentation enables the development of flexible and modular automation systems that are independent of vendors. This integration approach can save time, reduce costs and be more accessible to researchers and technicians.

The flexibility and modularity of the integrated laboratory automation system are also increased by using the same interface for all instruments. Figure 3.1 shows a typical setup of such an integrated laboratory automation system.

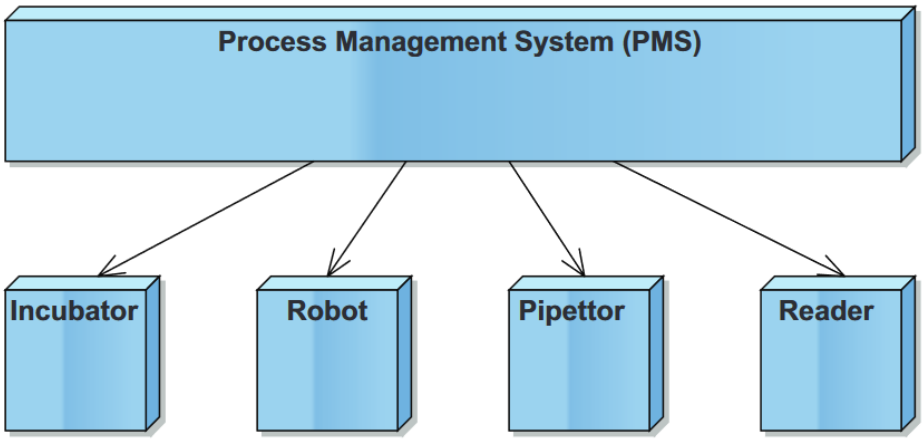


Figure 3.1: General model of an integrated automated laboratory system

The tools integrated into the system are controlled by the Process Management System (PMS, also referred to as laboratory execution system or scheduler), which is housed in a computer that is connected to the devices via a hardware interface. The hardware interface, which can be a serial port connection, USB, CAN bus, Ethernet, or WiFi, enables the PMS to communicate with the devices through commands, and for the devices to send messages back to the control software. This facilitates the seamless flow of information

between the PMS and the devices.

Instrument Control and Data Interface Specification

SiLA has developed the Device Control & Data Interface Specification as a fundamental standard that outlines how devices should be connected to the PMS, including the hardware and communication protocols required for communication.

Web services

The primary aim of the communication is for the PMS to trigger processes on the devices and fetch the device's current status. This is accomplished by executing remote procedure calls on the device and examining the resulting responses.

SiLA requires the use of Simple Object Access Protocol (SOAP) and Web Service Description Language (WSDL) for such communication, both of which are based on XML.

A full description of the command set that must be downloaded by the PMS is provided by each device. This command, including its parameters, is expressed in a SOAP message that is sent to a device. The device interprets the message and starts to work on the command in question.

Asynchronous event communication

Processing time for different commands can be completed in less than a second or up to 20 minutes, so asynchronous communication for all commands is provided by SiLA, handling this problem.

The event handler running on the PMS receives events, such as status or occurring errors.

State machine

SiLA utilizes a state machine to depict the functionality of a device. The main state machine is shown in Figure 3.2.

Network specifications

The specification defines a standard Ethernet network connection that enables devices to be connected to a PMS on a standard computer. The use of Ethernet connections eliminates the previous limitations commonly encountered with standard interface methods, such as restricted cable length or a limited number of ports, thereby providing more flexibility in operation.

Common command dictionary

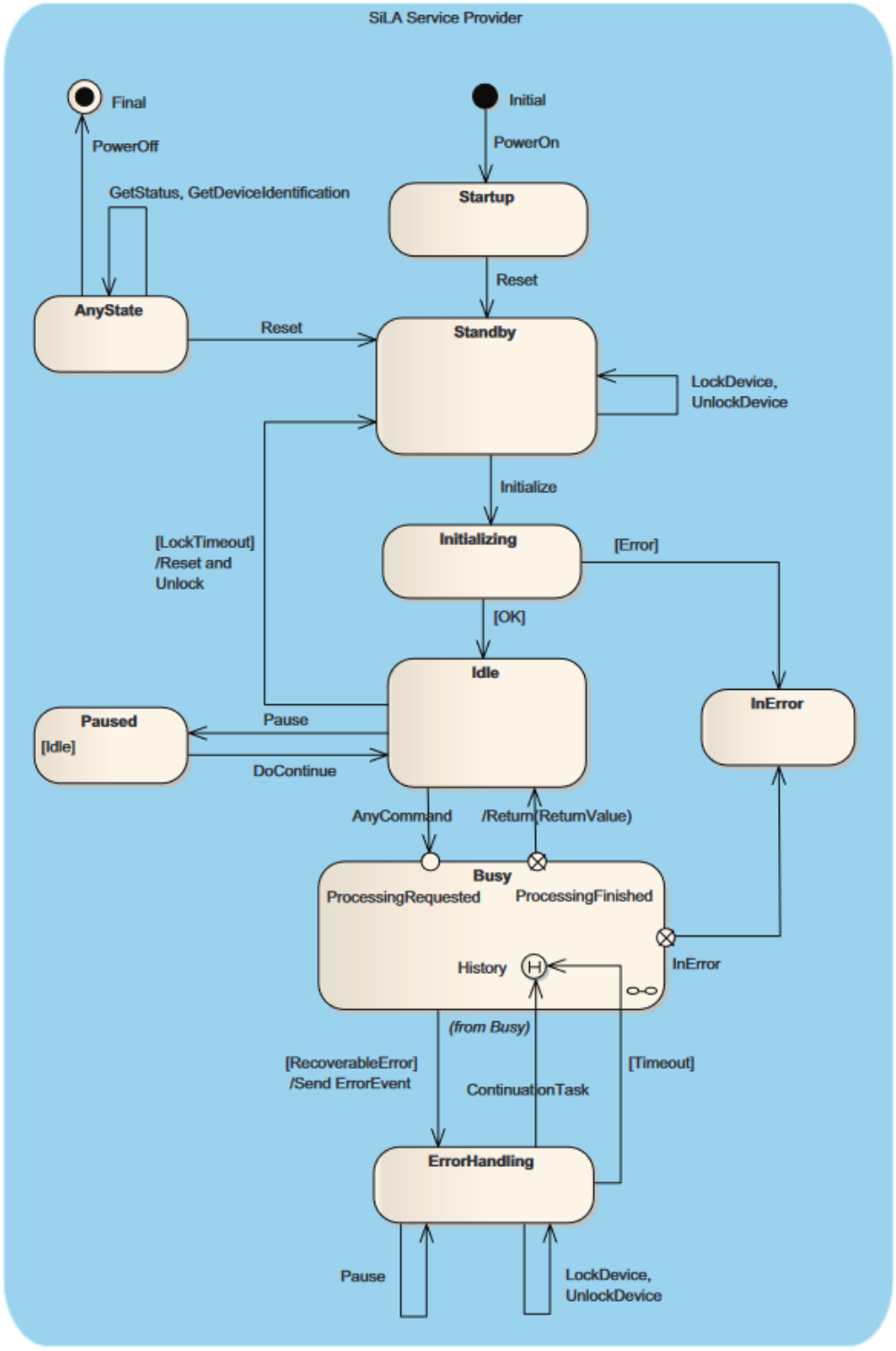


Figure 3.2: SiLA system, main state machine

The PMS controls and schedules various tasks for the device to perform, and initiates them by sending commands to the device. One of the goals of SiLA is to replace command

names by more descriptive names, and also to strive to define commands to an appropriate level of complexity and ease of use.

Therefore, devices are grouped into about 30 different classes, typical device classes being pipettes, dispensers, incubators, or robotic arms. Devices can also be classified according to different levels of complexity.

A common command set (CCS) describes a minimum set of commands for each class of devices, which must be implemented and executed for all devices in the class. There are two categories: required commands and optional commands. Required commands from the device class must be implemented by all devices, while optional commands are only implemented if the device has the described function.

New functions can be added by the programmer, where the name of the command must be descriptive of how the command works. Parameter names must be understandable, parameter types are limited to standard data types.

Data capture

An emerging issue is where to store the data. There are two basic options, the first is to store it on the device, in which case a link must be sent to the PMS, and the second is to transfer all the data to the PMS so that it can be handled separately.

Detailed documentation of the framework based on the SiLA standard can be found at [17]. Link to the official SiLA site [1].

3.1.3 SiLA 2

According to the SiLA website [18] and the official specification [17], the following new features have been introduced in SiLA 2:

- SiLA 2 follows HTTP/2 and REST-like communication via SiLA 1.x XML Soap-based technologies.
- For data transfer, it follows the format specified by gRPC
- Both wired and wireless communication
- The SiLA 1.x versions follow a service-oriented structure rather than a type-based classification of assets.
- It uses features that allow users to choose devices based on functionality, regardless of manufacturer.
- Thanks to its self-expanding nature, SiLA 2 enables true plug & play.
- SiLA 2 is not backward compatible with SiLA 1.x versions. The working group decided early in the development process not to implement backward compatibility in order not to compromise the proper design of the new standard.

– Simplifies the management of security certificates for encrypted and secure connections
– It is recommended to implement authentication and authorisation according to the specification [17] "Part B".
– Devices can be connected to the cloud.
– Provides access to laboratory instruments in isolated networks via a corporate network.

The SiLA 2 specification [17] is divided into two main parts: Core and Mapping. The concepts (Part A) are separated from the technical implementation details (Part B) for separate maintenance. The structure of the specification is shown in 3.3.

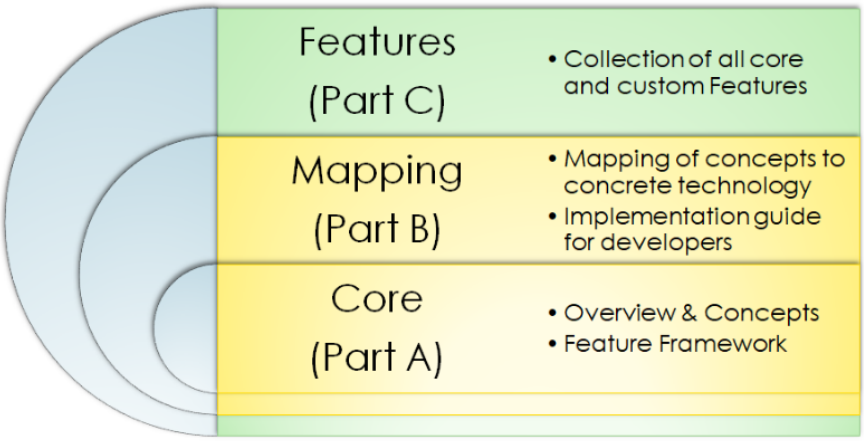


Figure 3.3: Parts of specification

Some reference implementations available throughout the community are written in Python, C++, C#, and Java for implementing parameters and properties for each SiLA feature, but SiLA itself follows a language-agnostic viewpoint.

3.1.4 SiLA Robotics Feature Unification

SiLA 2 is dynamically evolving as a standard, especially in the modular design of the framework, found in the official documentation's Part C section [17]. As a member of the SiLA 2 Robotics Working Group, it is clear that there are several interested parties included, such as robot vendor members from different companies, SiLA members, and other developers and students who are interested in the process. For robotic purposes, a proposal has been created in collaboration with the Working Group. A feature definition set is being developed by the group as an example and as a template for later use and development, discussed in regularly occurring meetings online with the team.

This includes the basic features needed for a laboratory automation setup, the most

important and most developed is *LabwareTransferController* as this is the backbone for a pick-and-place task.

For the robotic purposes a proposal has been created for preview by the SiLA leadership in a new namespace, *sila_base_robotics* forked from the official repository found at [19], *sila_base*. For every development on the SiLA features a process starts for acceptance. This is represented in figure 3.4.

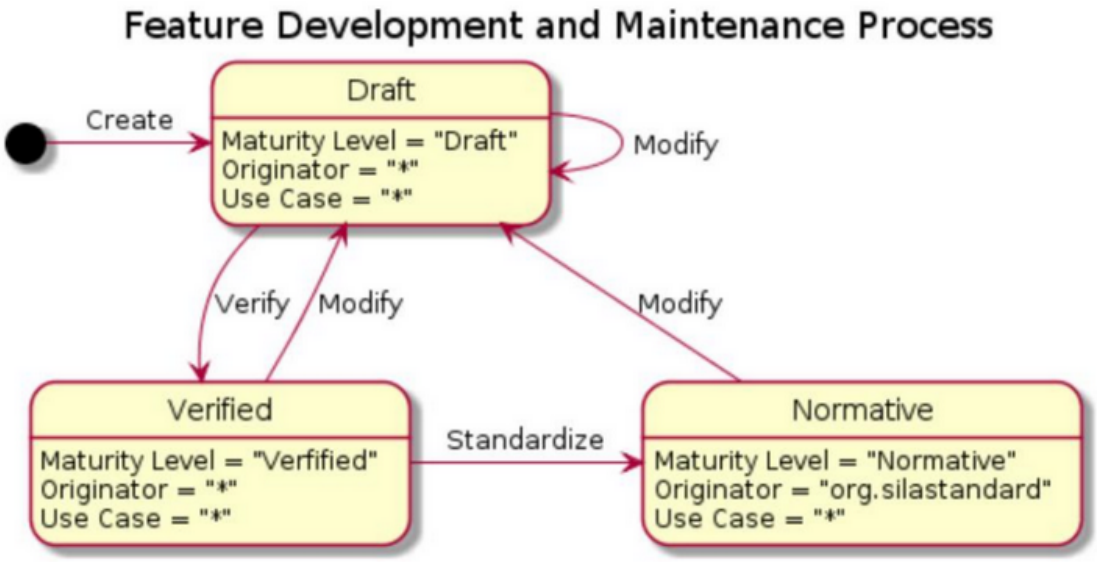


Figure 3.4: Process for accepting a developed SiLA Feature from [17]

Every Feature must have an Originator, Category, Maturity Level and Feature Version to be considered valid. After creation the Feature reaches the Draft phase, if it gets verified, meaning the Feature follows the Best Practices defined in the Documentation, it changes state to Verified, and for it to be Normative, it has to be standardized, meaning the Originator must be *org.silastandard*. These rules can be found in [17], the official documentation for SiLA. The goal of making the robotics definition is to get to the Normative phase.

Some of the design principles set for the definitions are

- to be vendor agnostic, so the usage does not depend on the manufacturer
- to not be dependent on whether the robot is stationary or mobile
- to be modular with distinct categories
- to have a high-level and a low-level implementation possibility

The current template for robotic features is the following, shown in figure 3.5. This is not yet filled out with the implemented feature attributes but only the idea for what should

be implemented in a SiLA robotics Feature Definition.

Figure 3.5: Template for Feature Definitions

From these, the most important for the pick-and-place task for the TIAGo++ robot is the *LabwareTransferContorller* and some functions from the *lowlevel* features for monitoring and providing status.

3.1.5 Existing applications

On the official website [20] you can find a collection of laboratory devices and software that support the SiLA 2 standard.

Relevant to this task is the *Robotic Lab Assistant*, manufactured and distributed by *Astech Projects Ltd.*. It is a mobile manipulator with six degrees of freedom, equipped with a MiR100 mobile robot base and a UR5 robot arm. The gripper is capable of adaptive force feedback and has an enclosed storage compartment for transporting and storing multiple objects. Its purpose is to transport samples and carry out automated workflows in the laboratory.

In addition, several stationary robotic arms, liquid handlers, and other laboratory instruments can be seen on the site.

The [21] source gives a very detailed description of how they built the system using SiLA 2. The solution presented was made flexible and scalable.

3.1.6 BioSASH Hackathon series

Hackathons are social coding events, also called codefests. These bring together interested parties on any topic with the goal to build new or improve on existing software programs. This can include generating new ideas, building prototypes, giving presentations, and connecting with sponsors on a given topic.

The BioLAGO SiLA 2 AniML Serial Hackathon (BioSASH) is a series of hackathons organized by BioLAGO and the SiLA team. They promote the spread of open standards for communication interfaces between laboratory equipment from different suppliers. These events give an opportunity for people from different fields to work together in hopes of finding new solutions to solve common challenges in process automation.

BioLAGO is "the network for the regional health industry", working in different sectors such as medical technology, diagnostics, bioinformatics, and pharmaceuticals.

The fourth installment of bioSASH at the end of September in Konstanz, Germany was held in order to "SiLA-fy" the labware transportation process for robot arms. A working implementation was delivered for the UR3 and a PreciseFlex robotic arm, performing a pick-and-place task from one robot to another through a shared station between them. With the help of this demo, it is possible to implement this feature for the TIAGo++ mobile robot too for the diploma project.

Participation in the event has led to a basic understanding of the UR3 robot arm library called *ur_rtde* which was used for this implementation in C++ together with a SiLA server

and client, connecting the two programs through the network [22].

A summary of the event can be watched at [23] in video form [24][25].

3.2 Robot Operating System (ROS)

The ROS (Robot Operating System) is a collection of software libraries that is the primary way for developing robotic applications, as stated on the official website [26]. It was introduced in [27] in 2009. This project is based on the distribution Melodic, ROS 1 since the robot in use is integrated into that version, although ROS 2 has already started to spread for newer robotic uses.

ROS has five fundamental design goals:

- *Peer-to-peer*, as the hosts are connected by a peer-to-peer topology
- *Tools-based*, as the code and packages written in ROS can be based on multiple languages, the most used are C++ and Python
- *Multi-lingual*, as instead of a monolithic development, the ROS consists of components and small tools
- *Thin*, as the purpose of ROS is to be able to re-use codes and libraries, one should aim to develop drivers and algorithms that are not dependent on ROS
- *Free and Open-Source*, as the source code of ROS is public and the additional libraries are community based[27]

The fundamental concept is a publish-subscribe-based communication between ROS Nodes via topics, services, and actions. Nodes are modular computational processes. A ROS environment can be built up from multiple nodes which communicate asynchronously through Topics in a peer-to-peer fashion. A Topic is for transporting messages, one node can publish a message to a Topic, and another can subscribe to this Topic to get the message. The synchronous model for communicating is managed by Services and Actions. These have goals, responses, feedback, and strict message types. Every Topic, Service, and Action can be published/called either from the command line, from the robot's graphical interface, or via the ROS API, from code.

If there are multiple ROS instances on different machines, they communicate with each other through a ROS Master, a dedicated device that connects to all other machines in a bi-directional manner.

There is a way to start selected nodes at the machines' startup with a file with *.launch* extension. This makes a script that starts a ROS core or connects to the Master and starts its own nodes.

In ROS there are multiple useful visualization tools for monitoring and debugging, like

the program RViz, which is used for viewing different types of data such as images, point clouds, robot poses, trajectories, and so on. These data are gathered from Topics.

Another useful tool is *rqt* which is capable of showing Topic and Node connections and monitors Topic data. There is a plugin for it called *node graph* which shows the connections in a graph manner.

There is a tool called *tf* which constructs a transformation tree between all frames in the system in an automatic and systematic approach.

The website for ROS is at source [26] and the main website for searching for tutorials and understanding the concepts is the so-called ROS Wiki at [28]. Here are detailed steps for installation and basic examples for the most important libraries. The message types required by the Topics can be found here to find out what is their definition, and what fields are required by them.

A more in-depth description of the used packages can be seen in 5.1.

4 PLANNING

4.1 System requirements

The system on a higher level has to include two workstations, a mobile robot either including a computer powerful enough to execute high-throughput tasks or an additional computer for development.

The robot has to have a system as simple as possible while completing the transportation task between the two workstations. The current implementation only has one robot but for keeping the possibility open for seamless integration of other devices, the ability to communicate between them is mandatory. An interface for maintaining and handling tasks has to be included in the system.

Considering section 3, for accomplishing the goal of the project the ROS framework and the SiLA standard are both capable solutions.

For the robot, the TIAGo++ mobile manipulator is the best available option as it can be provided by the Company hence the robot's system had to be running on ROS for leaving enough room for future developments. As ROS is wildly used and continually updated, it is up to date with currently used technologies and gives the possibility to customize any of the packages and nodes. The robot itself is most suitable for manipulation tasks considering the dual-arm setup. In addition, PAL Robotics has made extensive modifications to base ROS functionalities for a more customized foundation for their robots. They also provide great documentation and a handbook about the usage of TIAGo++. This results in the use of a Ubuntu operating system as the main computer running on the robot supplemented by a laptop for making the development process easier.

As the robot runs on ROS, it is the obvious choice for the system's base framework although it is also the most commonly used technology in the case of robots. ROS would be the primary choice in most robotic uses since it is widely known, has plenty of different packages for standard robotic applications, is open-sourced and highly maintained, and is fundamentally built for development and prototyping.

As for communicating and connecting all devices on a higher level, the SiLA communication framework is included. This allows the user of a use-case based on the concepts presented in this project to be able to easily manage the system and execute tasks and sub-tasks without problems and with appropriate monitoring possibilities.

Based on these requirements, the system architecture can be created, seen in section 4.2.

4.2 System architectures

The goal of the diploma project is to implement a SiLA 2 compliant interface for a mobile manipulator in a laboratory environment. The system's plan can be seen in figure 4.1.

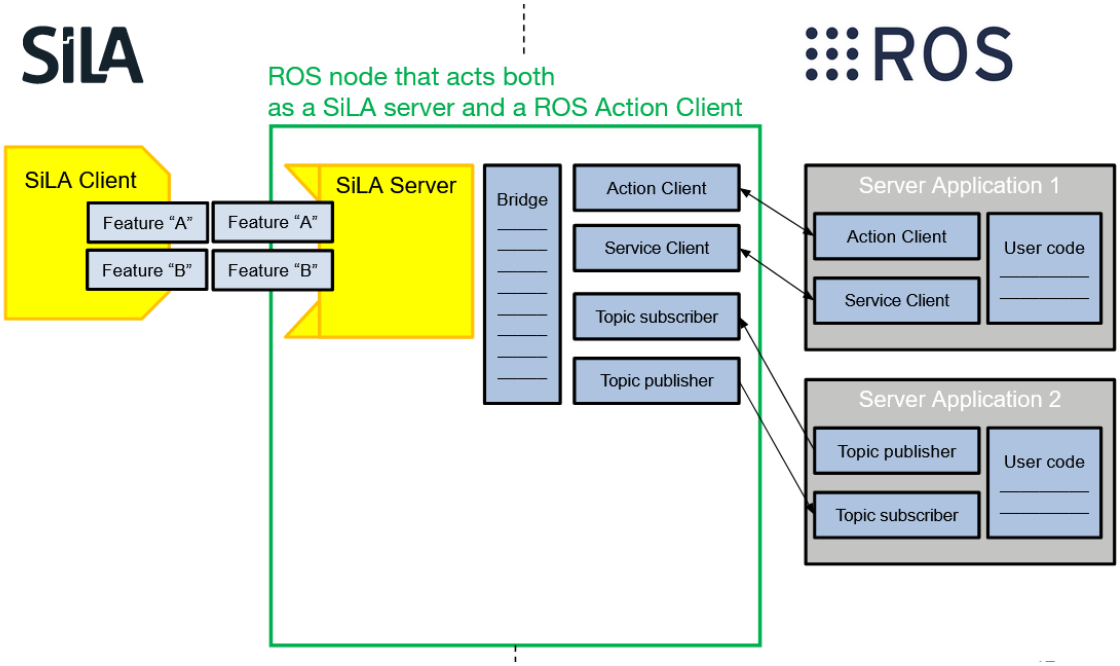


Figure 4.1: System architecture

The SiLA *Client* and *Server* will communicate with the help of *Features*. The *Bridge* represents the SiLA-ROS-bridge package, for which a reference implementation was developed by the SiLA community, it can be found at [19] as a GitLab repository. From the ROS side, the communication is carried out by *services*, *topics*, and *actions*.

The mobile manipulator was chosen to be the TIAGo++ by PAL Robotics [2], the name TIAGo stands for Take It And Go. The environment for the first phase of the project was executed at the robotics laboratory of the Company.

As for the second phase, the Robotics Laboratory at Antal Bejczy Center for Intelligent Robotics at Óbuda University (IROB), Budapest was the best option for continuing the project. The robot had to be transported to the laboratory from Vienna.

The goal was to integrate TIAGo++ in the planned system by implementing the SiLA Features needed for doing tasks and testing the setup for realistic scenarios and demonstrating if the approach is correct.

5 IMPLEMENTATION

5.1 Experimental environments

5.1.1 Pharmaceutical laboratory, Vienna

During the summer internship at the Company in Vienna, the first main part of the diploma project has been achieved. The ROS environment has been made and the robot was set up to do the tasks needed for the project. This chapter gives a thorough description of the work that was done during this phase.

The goal of the internship was making a demonstration with the TIAGo++ robot in a laboratory to be able to transport a sample carrier from one station to another, this is a so-called pick-and-place task.

The Company's robotics laboratory was the main space for the development processes. The laboratory contains some desks, around ten square meters of flat space, and some analytical devices on tables. Two desks have been assigned to be included in the project for representing two stations in a liquid handling process, so two ArUco markers have been used to mark the stations in dedicated areas.

The robot can move freely on the flat space in the room, this excludes an exception where a drain lid on the floor breaks the flat surface causing the wheels of the robot to get stuck if it navigates over it. This is why this spot has been marked as a virtual obstacle during navigation.

The layout of the room is shown in figure 5.1. The white space is where the robot can freely move, the grey areas are out of bounds. The two stations were used for accomplishing the pick-and-place task, the docking station is used for charging the robot.

It is also important that this room was used by multiple people as a workspace and also as a passage room, so the navigation had to include collision detection for safety and for achieving full autonomy and collaboration.

5.1.2 IROB

The second phase of the development process has been done at Óbuda University's IROB Laboratory during the last semester. This robotics laboratory has a bigger floor area, so the navigation tasks can be a bit more complex, since there are obstacles throughout the room, including desks, chairs, and other robots.

A small problem occurred with the navigation at first caused by the linoleum floor

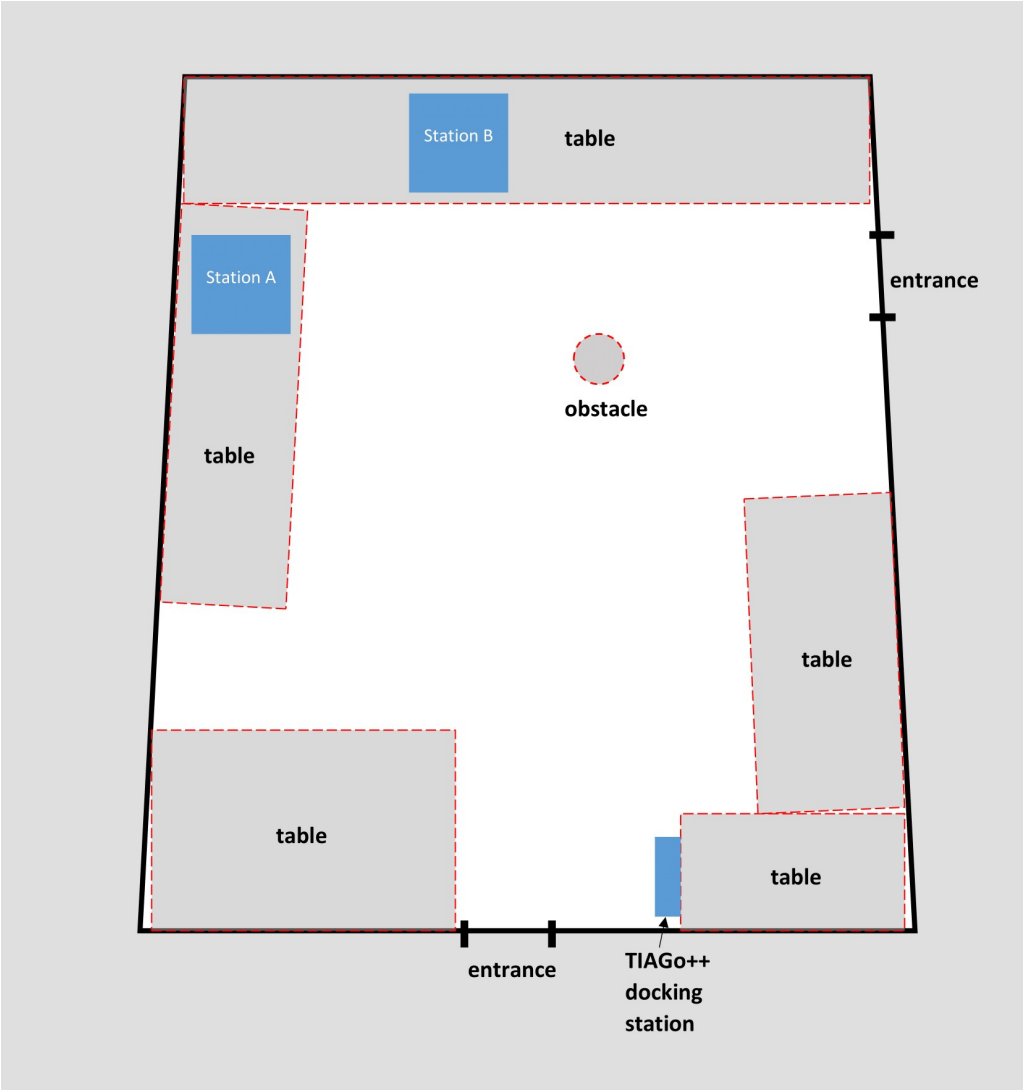


Figure 5.1: Map of the robotics laboratory

being bumpy in places. Because of this, the robot could get stuck on these as there are four wheels in the robot's base and when it gets a command for turning, it could be stuck in one place for a few moments. This causes the navigation to drift. When the robot perceives a turning motion, it also turns the mapped floor around itself. This leads to a contradiction between the real position and the virtual one. A few parameter changes in the navigation solved this, so the robot corrects its position on the map more frequently.

The map of the lab is shown in picture 5.2. The two stations have been also set up in the lab. This shows all the tables and desks the robot cannot go to in gray, the important places like the stations and the docking station, and the area where the robot can move around in white. The robot was not let out of the doorways.

As for the workspaces at the two stations, two different versions of the same idea had

Figure 5.2: Map of the robotics laboratory, IROB, created by TIAGo++ navigation

been created. Both workspaces had been fixed to the chosen desks for the automation process. Both have the exactly same space for the objects moved by the robot in position relative to the markers as well as the area reserved for them.

Station A holds a 3D-printed plate with two sockets for the object handled for the pick and place task and an engraved part for correctly positioning the ArUco marker. A picture taken of the station can be seen here 5.3. The objects fit in the sockets perfectly, with a minuscular room to move, so they cannot get stuck. This is important for the picking task carried out by the robot as well as for placing the object.

Station B uses a bit different solution. The base plate is made from fiberboard cut with

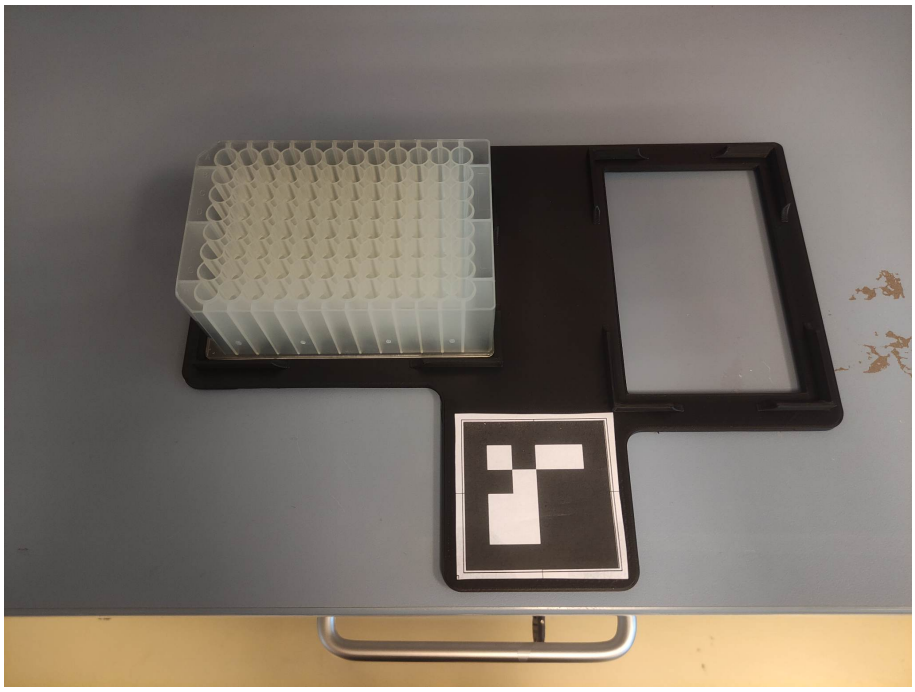


Figure 5.3: Station A workspace, 3D printed model

a laser cutter machine and the sockets are 3D printed. The positions are exactly the same as at the other station, relative to the marker. This model can be seen in figure 5.4.

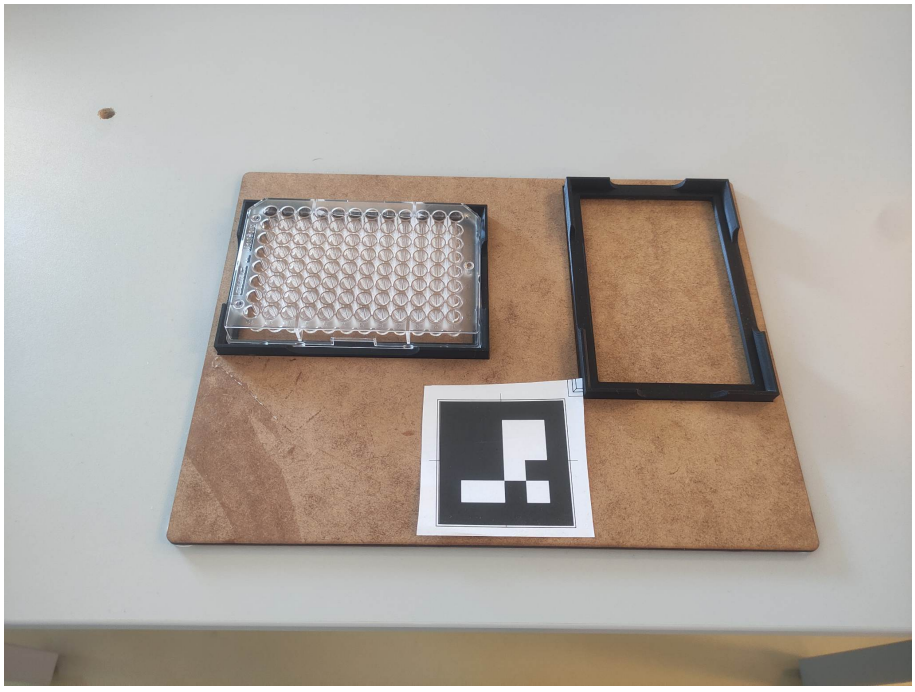


Figure 5.4: Station B workspace, laser-cut fiberboard base, 3D printed sockets

The third "station" is located on the base of the robot, a so-called onboard nest for

placing the manipulated object while navigating between stations. This secures the object from moving around in these phases. This is also a 3D printed frame that can hold the object, the placement on the base can be seen in figure 5.5.

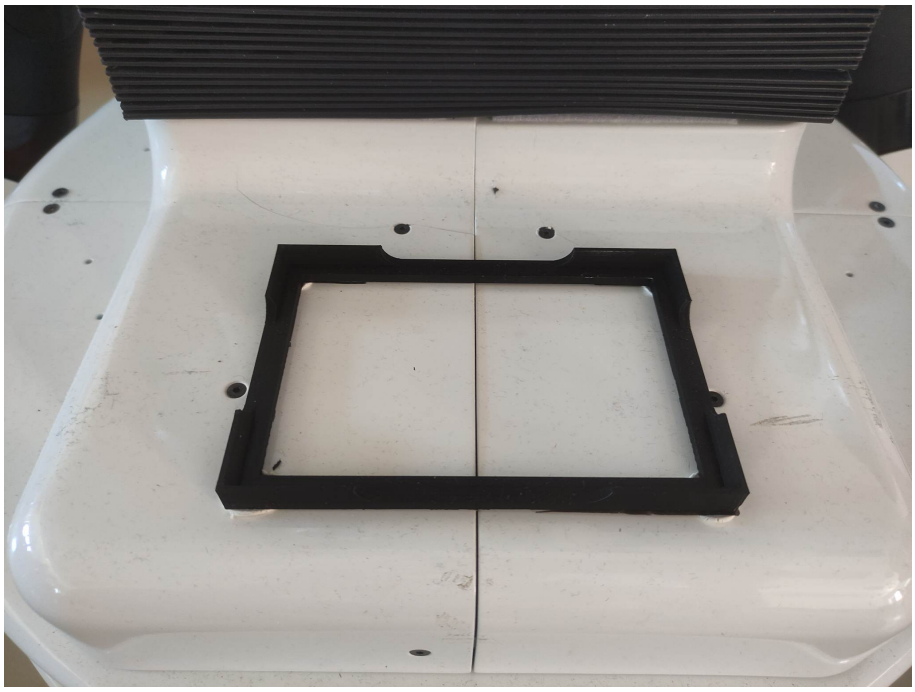


Figure 5.5: Station located on the robot base, 3D printed

5.2 TIAGo++

5.2.1 Hardware

The robot itself was developed by PAL Robotics based in Barcelona, primarily focusing on providing robotic solutions for research purposes. This Spanish company makes multiple variations of these robots with different hardware and software. There are also humanoid robots that are made by them.

Their website on TIAGo models can be browsed at [2], where the TIAGo++ configuration can be found, which is the two-armed variation.

As seen in picture 5.6, the robot consists of a mobile base with driven wheels, a laser range-finder for measuring distances on the horizontal plane for navigation and mapping, a speaker, a docking station connector for charging, and a small tray area in the front on top of the base. There are additional LED stripes for visually communicating some basic states like indicating turning motion or blinking when running low on battery.

The torso has two 7-degree-of-freedom arms attached to its sides with force sensors

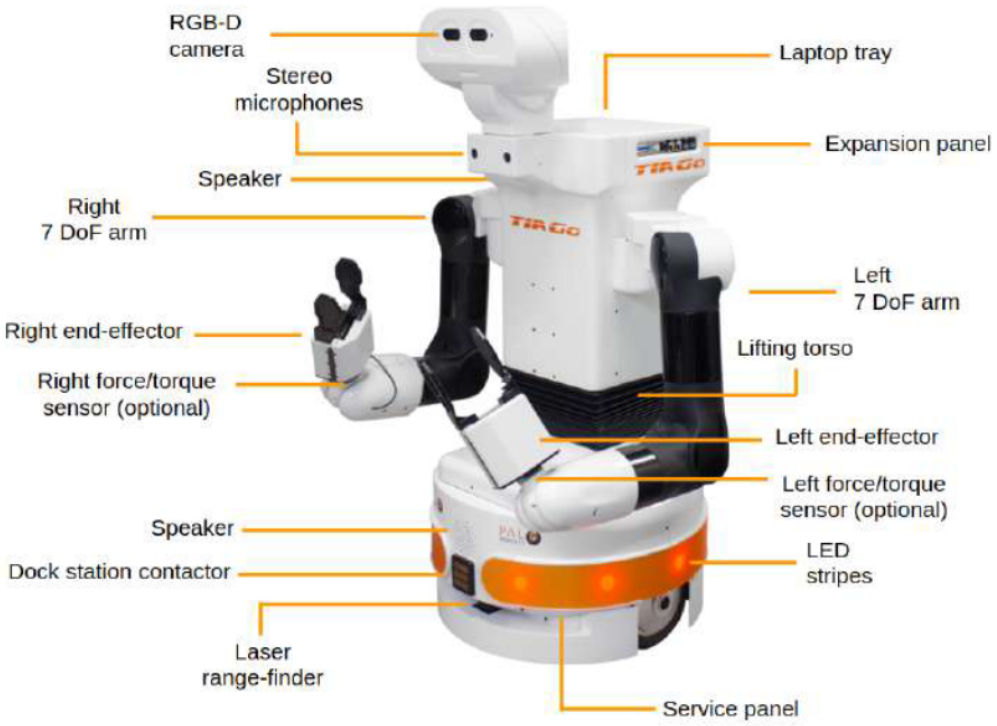


Figure 5.6: TIAGo++ mobile robot from PAL Robotics Handbook [29]

and parallel grippers as end-effectors. These also include hand-eye cameras in both end-effectors.

The torso of the robot can be lifted by an internal mechanism, so its workspace is relatively wide. The head can also be tilted and rotated so the stereo camera can move independently of the body.

There is also an expansion panel on top which works as a laptop tray for on-site programming or testing, and the "head" with a built-in RGB-D camera, where the "D" stands for depth, meaning with infrared projection it gives a depth image consisting of a point cloud. There are also stereo speakers and microphones integrated in the panel which are used to record audio in order to use speech recognition. Figure 5.6 is from the original handbook [29].

There are some more sensors not included in the image, such as

- sonars, which are capable of low to mid-range distance measurements, located on the mobile base on the back side placed in three different directions
- Inertial Measurement Unit (IMU), which is used for measuring internal forces to estimate the robot's position or movement

The end-effector is a parallel gripper with two independently moveable fingers, but

the whole end-effector or either of the fingers can be removed and changed. The default grippers were considered too narrow to be able to grasp the objects needed for the pick and place task, so a 3D printed model had to be custom made including four identical fingers which can be screwed to the end-effectors. A closeup picture is shown in figure 5.7. The fingertips are covered with silicon sheet cutouts for better gripping.

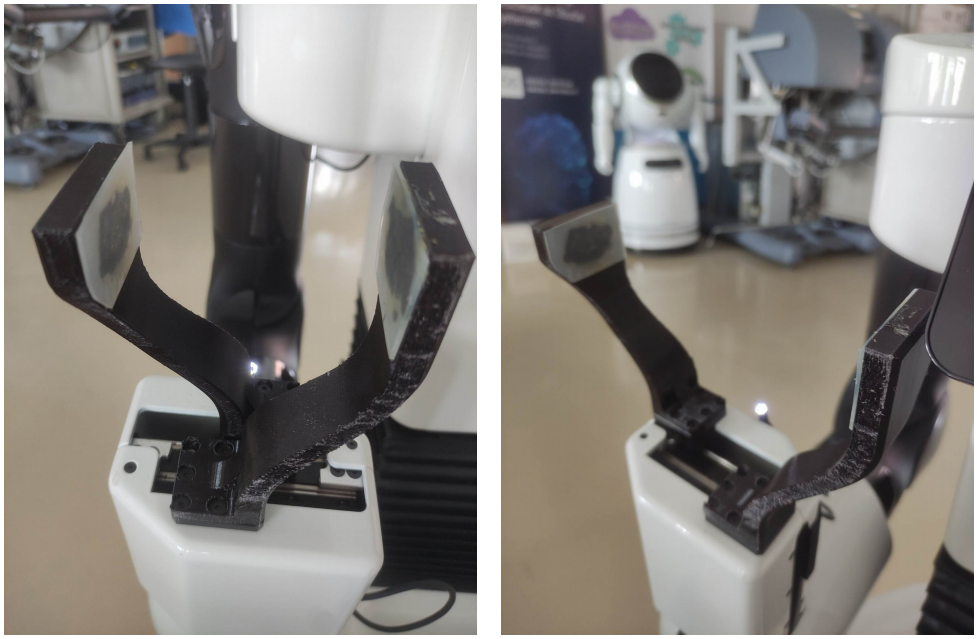


Figure 5.7: Custom 3D printed grippers

There are multiple safety mechanisms built into the TIAGo++ robot:

- checking the battery percentage
- the arms will fall down slowly to prevent damage in case of a low battery charge
- detects if a motor in the robot is too hot it can shut down itself
- built-in collision detections for the environment and also for self-collisions

5.2.2 Software environment

There are a lot of important properties also mentioned in the handbook on the topic of software environment and modules, here are some of the most important ones.

The primary way of establishing a connection with the robot's computer is using the Secure Shell (SSH) protocol, for making an encrypted secure connection between a client and a server. More about SSH on the official website at [30].

There are relatively high computational requirements for the development setup, meaning it is recommended by the manufacturer that this setup has a powerful graphics card,

at least an 8-core CPU, and support for Nvidia cards. This is because an important aspect during development is using the simulation environment, discussed later.

The robot's computer hosts a web page as a graphical interface, provided by PAL Robotics, called WebCommander, which can be reached from the robot's touchscreen which is located on the front of the robot, or by connecting to it with the correct IP address. Examples are shown in figure 5.8 in four panels, from left to right Movements, Diagnostics, Startup, and Video.

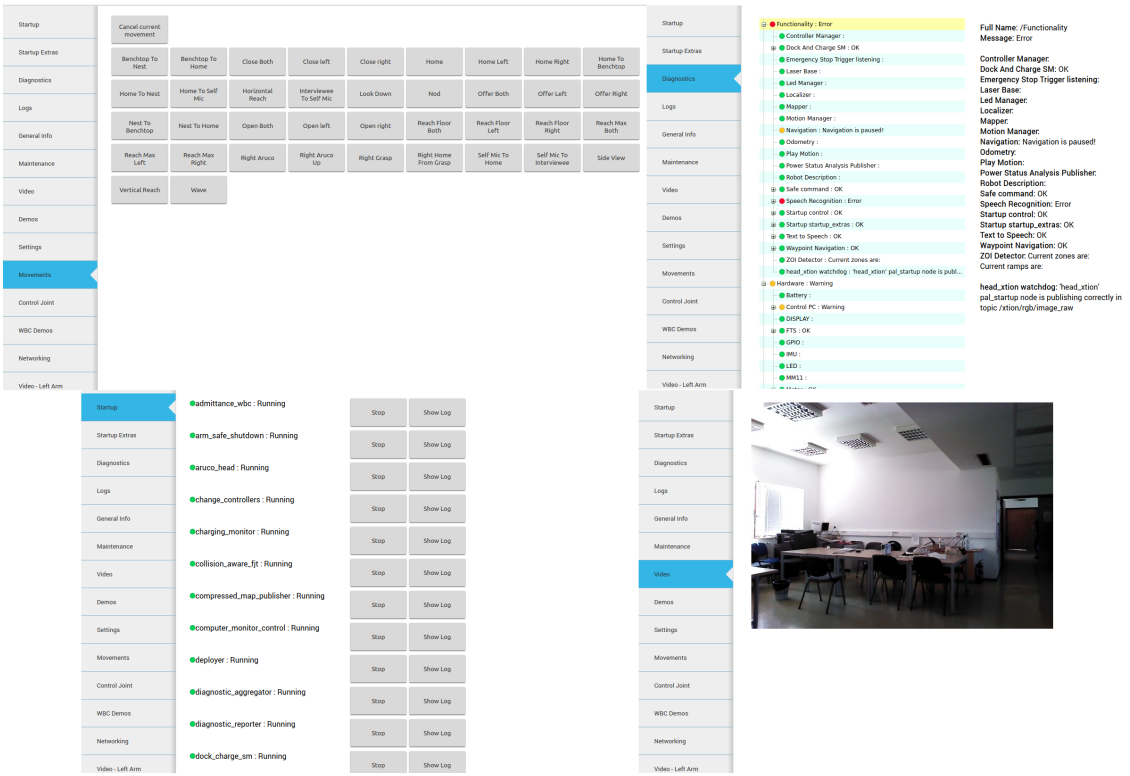


Figure 5.8: WebCommander from the TIAGo++ handbook [29]

The tab names on the sidebar correspond to the content included when clicked on. All can be configured manually as pleased in a YAML file. By default, it contains some important information such as the robot's state, running applications, installed libraries, and configuration tools. The tabs are initialized by a plugin from the robot's file system.

Some important tabs are the Diagnostics, which is most useful for debugging, and the Movements tab where pre-programmed movements can be given as commands with a click. This is where the needed movements for the pick-and-place sequence were added and used most of the time during development and testing. There is also a Control Joint tab, where the joint values can be set.

Purchasing the robot also comes with an operating system ready to be used. This was used during the internship as a virtual machine running with Hyper-V, on a provided

laptop with sufficient hardware requirements, such as Intel Core i7-10750H CPU, 32 GB RAM, Nvidia Quadro P620 GPU, and 1024 GB of SSD storage, and later on another laptop natively at IROB with an Intel Core i7-8565U, Intel UHD Graphics 620 and also 32 GB RAM. The architecture can be seen in figure 5.9.

Figure 5.9: Software architecture provided by PAL Robotics [29]

The operating system is Ubuntu LTS, at the time of development, version 18.04 was considered the newest working version of the robot. The software also includes the fitting ROS release Melodic for the Ubuntu version, since there is a strict way of matching the two. This includes all the needed ROS packages for working properly with the robot. There are two separate paths in the file system created for these files, one contains the official ROS Melodic distribution packages and the other is for the packages specified by PAL Robotics which incorporates a new distribution called Ferrum. There is also a separate path for storing newly developed packages by the customers.

The ROS Master is set on the robot's computer. This way the development laptop can connect to it easily if it is already running and has launched the needed nodes. It is important to note that the two computers have to be on the same network and the IP addresses have to be set in the computers' network settings and in ROS settings also, in the file *.bashrc*.

5.2.3 Movements and Motions

There are multiple ways for controlling the base motions of the robot. One is using a Logitech F710 wireless gamepad, which has the highest priority if used considering the possibility of an emergency. There is a ROS API that is capable of controlling the robot through a Topic `/mobile_base_controller/cmd_vel` with a message type

including linear and angular velocities, these control the two drive wheels. The 5.10 diagram shows the different nodes that send velocity messages in the direction of the base controller to the discussed topic.

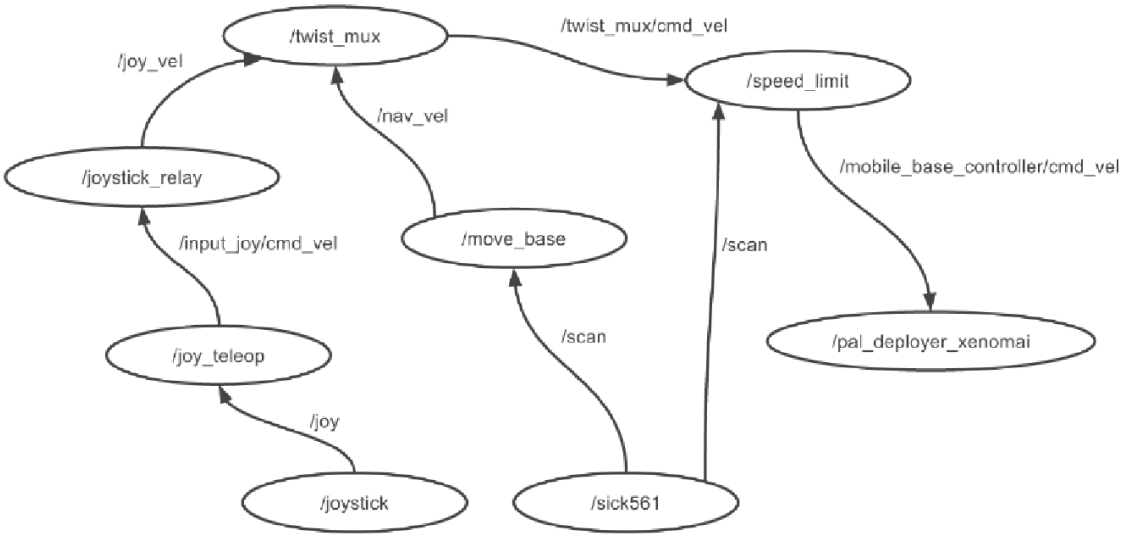


Figure 5.10: Mobile base control diagram [29]

The torso can also be moved with either the controller triggers or the ROS API through the Topic *torso_controller/command* or an Action interface *torso_controller /follow_joint _trajectory*, given with positions.

Same as the torso, the head can be moved with the controller or the ROS API and in addition with an *rqt* GUI. The latter provides a window with sliders for both of the joints in the head, one for turning sideways, the other for looking up or down, and a slider for setting the speed scaling for the movements. This method can also be used for the gripper movements.

The arm movements were the most important for the pick-and-place task. These also can be used by the methods presented before but the most use for the project was in the Action interfaces provided by the ROS API. These include */arm_left_controller /follow_joint_trajectory* and the same for the right arm. These encapsulate a *trajectory_msgs/JointTrajectory* message type which consists of a header, the joint names in a string array, and the corresponding trajectory points.

The hands and grippers have the same methods for control, the used Service for the gripper grasping motion is for the left gripper */parallel_gripper_left_controller/grasp* and for the right gripper also. This Service starts the grasping process, and when detecting an object from force feedback the parallel gripper fingers stop in place while looking out for the motor to not overheat.

A useful motions engine is used by the TIAGo++ robot, the *play_motion* ROS package. This can play back predefined motions involving the upper body. There are default motions included by installing the provided software but for the pick-and-place task, some unique ones had to be defined as well. The idea is that this package is able to store a sequence of motion primitives and save them as one motion. These include the positions and the time intervals between them. A YAML file is responsible for storing these so the user is able to either modify this or use the PlayMotion Action interface or the Action client with a GUI for easy use. These motions can also be advertised to the WebCommander as buttons so they can easily be tested and used.

For two-dimensional navigation the standard ROS *navigation* package is used, the so-called Navigation Stack. This is the most popular way of using planar navigating when working with mobile robots. Of course, the manufacturer configures the settings of this stack before selling a TIAGo++. The Navigation Stack uses the popular SLAM (Simultaneous Localization and Mapping) for creating maps from the laser's and the sonars' data. For selecting a goal on the created map there are multiple ways, including a Topic, Action interfaces, and Services.

There are some navigation goals that were manually chosen for the pick-and-place task, called Points of Interest (POI). The first is in the position just in front of the docking station, so the robot can call the Docking Service if needed. Then for the two stations, there are four points in total, an approach and a "station reached" position. These are important for the robot to approach from an accepted angle, so approaching the positions at the desks a basic forward motion was used. This means a linear velocity was set in the "forward" direction for a set time duration, making this repeatable and not changing. In figure 5.11 the orange circles can be seen as the most important locations on the map at the laboratory.

A close-up of the stations themselves is seen in figure 5.12. This basic setup consists of an ArUco Marker, frequently used for marking positions in an environment, and the placeholder object placed in a fixed location, always the same distance and orientation from the Marker. This ensures that when the robot is up close to a station and either the head stereo camera or the hand-eye camera can see the Marker then the object location is known. The fixed place for the object is called Site.

As for the setup at IROB, the map also includes the POIs for the task, seen in figure 5.13. The station closeup can be seen in figure 5.3 and 5.4.

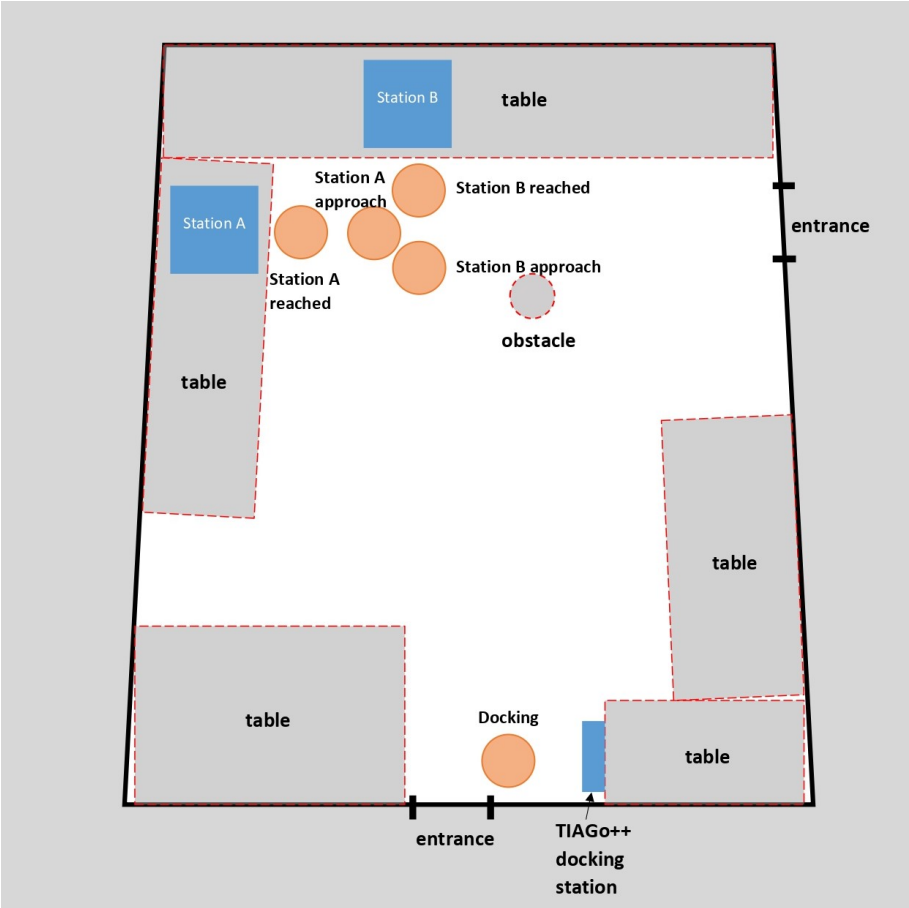


Figure 5.11: Navigation goals at the laboratory

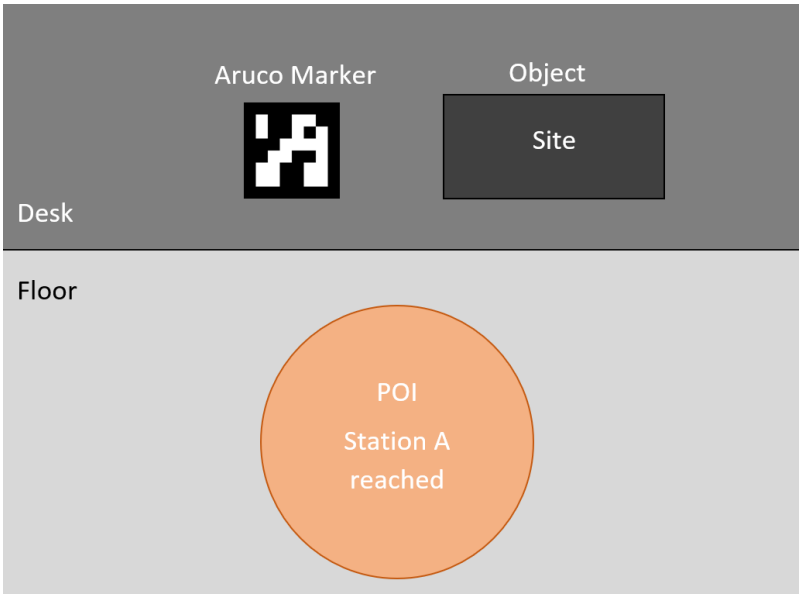


Figure 5.12: Station layout

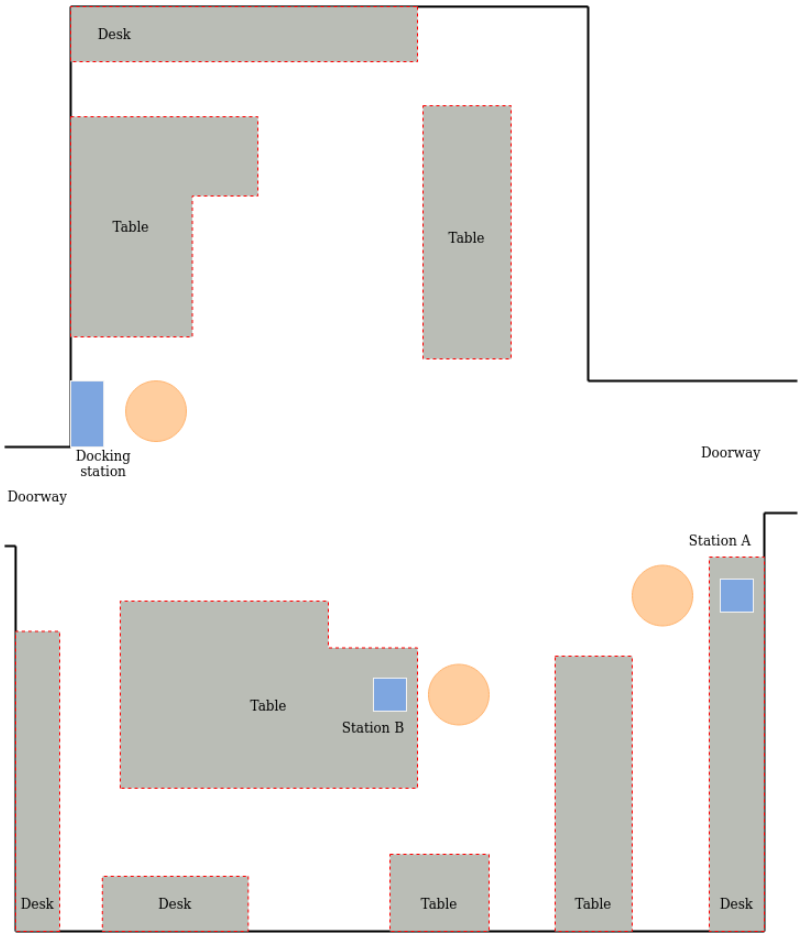


Figure 5.13: Navigation goals at IROB

5.2.4 Motion planning

For the motion panning with arm movements and torso movements, the MoveIt! ROS package and GUI were used [31]. For the pick-and-place task, multiple motions had to be pre-programmed, where the planning can be done in joint or in cartesian space. These are the movements between the following states.

- *Home*, a tucked-in position, used during navigation and at startup
- *Benchtop*, a configuration with the right arm brought up and forward, for picking tasks
- *Nest*, which is used for placing objects on the front tray

The states can be seen in figure 5.14 from two angles. 1 and 4 are Home, 2 and 5 are Benchtop, and 3 and 6 are the Nest configurations.

The connecting movements are used in the final sequence of the pick-and-place transportation which is saved to a YAML file, containing the joint positions and the timings

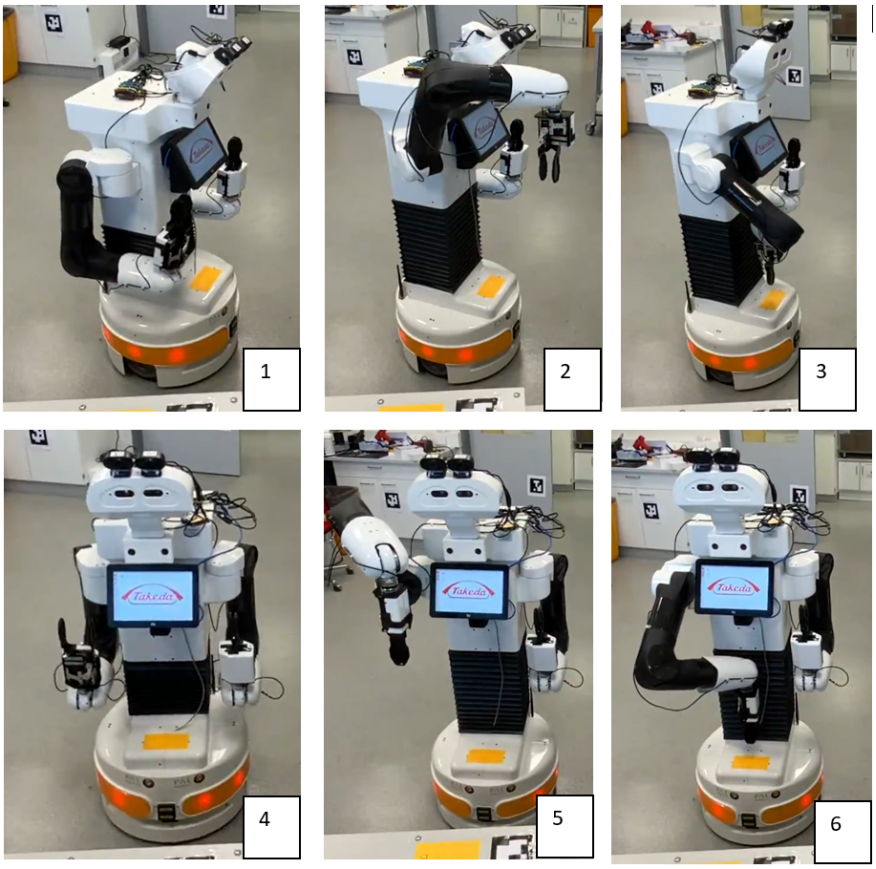


Figure 5.14: The 3 states for the motions

between each step. One movement can consist of multiple partial movements to guide the arms better.

The GUI for motion planning is a custom plugin (provided by PAL Robotics) for *RViz* and is relatively easy to use. A few things need to be specified for it to work, such as the planning group, which defines from which joint is considered the start of the kinematic chain, every joint before that does not move. There has to be a Start State and a Goal State for the MoveIt! to be able to plan the trajectory between them. The setting of the Goal State is achieved either by using the sliders for the joints individually or using the graphical indicators in RViz, the developer is able to grab the end-effector and move it around inside the workspace. If the position is valid, MoveIt! planner will try to calculate the movement.

There are additional Actions for starting or stopping the Docking and Undocking maneuvers. This works only if the robot is looking in the direction of the docking station from a 90-degree angle with +/- a few degrees.

5.2.5 Simulation

Most of the development has been done in the simulation environment Gazebo [32]. Before motions by MoveIt! were deployed to the real robot, all got tested in the simulation before trying it out on it's real counterpart. The very detailed URDF robot model was provided by PAL Robotics as part of the software package, this was used during development.

The figure in 5.15 shows the TIAGo++ in the simulation environment.

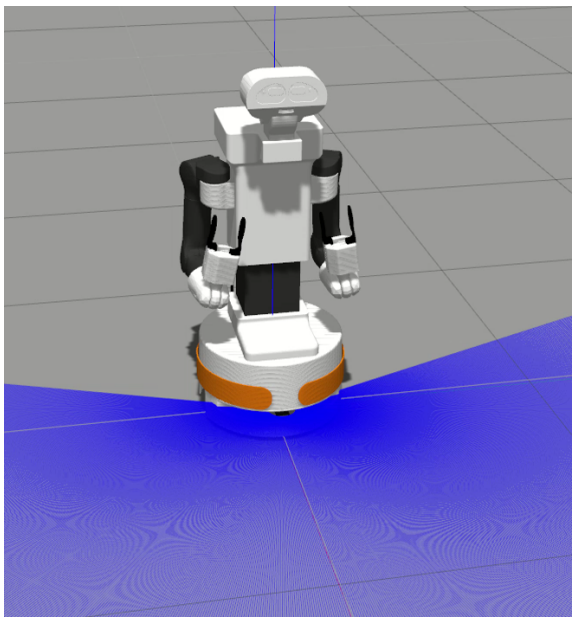


Figure 5.15: TIAGo++ in Gazebo, on the ROS virtual machine [33]

The simulation setup is important for times when the robot is not available or when work from away from the laboratory can be completed. This is called offline programming in robotics.

5.3 Transportation sequence

As the main part of the internship a pick-and-place transportation task was implemented with the TIAGo++ mobile robot, meaning an object had to be delivered from one station to another only with the robot, in a fully automated way. As a placeholder object, a small box was used for testing instead of a microplate, which is frequently used for transporting and storing liquid samples in a pharmaceutical R&D laboratory like at the Company. The point of laboratory automation, as discussed in the introduction would be mostly used in these kinds of laboratories.

Later at IROB, an actual microplate was used to be able to create an environment which is more connected to a real laboratory use.

For this transportation (as seen in figure 5.11 and 5.13) there are two stations for the transport, and the docking station for the start and finish of the demonstration. The main steps for the task are the following:

1. The robot starts in the docking station
2. Undocking Action is called
3. Go to "Station A Approach" POI
4. Carry out "Home" motion
5. Carry out "Home to Benchtop" motion
6. Moving forward to desk
7. Carry out "Look down" motion - this only moves the head slightly down, too look at the contents of the desktop
8. Move gripper over Object at Station A
9. Open right gripper with Service
10. Move arm to grabbing position
11. Call Grasping Service
12. Move arm back to "Benchtop" configuration
13. Drive backwards for the robot to have enough space for further movements
14. Carry out "Benchtop to Nest" movement
15. Open gripper, so the object is placed on the onboard nest
16. Carry out "Nest to Home" movement
17. Go to "Station B Approach" POI
18. Carry out "Home to Nest" movement
19. Grasp object
20. Carry out "Nest to Benchtop" movement
21. Move forward to desk
22. Carry out "Look down" motion
23. Move arm over Site while holding onto the object
24. Release gripper, this places the object to the Site at Station B
25. Move arm back to Benchtop position
26. Drive backwards
27. Carry out "Benchtop to Home" movement
28. Go to "Charger" POI, facing the charger
29. Call "Dock" Action

This sequence had to be implemented by following the architectural plan presented on figure 4.1, so the setup has a ROS side, a SiLA side and a common node for connecting the two sides. These are explicated in sections 5.5 and 5.6.

5.4 Steps for setting up the development environment

After unboxing the robot, the steps for setting up the environment properly has been carried out and documented thoroughly.

The first step is installing the environment on the development laptop, based on the *.iso* file provided by PAL Robotics. It is a Ubuntu 18.04 with the preinstalled ROS packages needed by the robot. Additional important tools, like Visual Studio Code [34] also need to be installed with the following extensions: Git Graph, ROS, C/C++, Python.

For the development laptop provided by the Company for the first phase of the project, the environment had to be installed via a Virtual Machine because of IT restrictions. The installation process was a bit more complicated than in the second phase at IROB because of this. As for the second phase, the whole setup was directly installed on the laptop's hard drive.

The git repository, where the codes for the project are located is called *dev_ws*, which stands for development workspace. Here, all the modified and additionally installed packages can be reached. The highlighted files and codes can be found in the supplementary material.

To be able to connect to the robot from the developer's laptop, network configuration is required. For this, fixed IP addresses had to be set for simplicity. At both of the locations, a dedicated WiFi router had been set up for communicating with robots in the laboratory, at IROB also with an internet connection. The development laptops connect to the router via Ethernet cables for faster connection, the robot can only connect through WiFi.

Some additional adjustments were needed for a full configuration. This includes the editing of a file located at */etc/hosts*. This configuration file matches the hostnames to the IP addresses they can be reached at. The next line was added to this file, as the robot's hostname is *tiago-117c*: `192.168.1.54 tiago-117c`. On the graphical network settings interface in Ubuntu, the laptop's fixed IP can be set.

After the network is correctly set up and verified by using the command `ping tiago -117c`, the package *ssh* can be used for connecting to the file system of the robot from the laptop so it can easily be modified if needed. For this, the file */.ssh/config* on the development laptop has to be edited with the lines:

```
Host tiago-117c
HostName 192.168.8.160
User pal
```

As mentioned before in section 5.2.1, new, 3D-printed grippers have been placed on

the end-effectors. Because of this, the gripper model stored as *stl* files included in the robot model's *urdf* file needs to be changed to match the new configuration. This is important for the correct placement of the TCP (tool center point) to grip objects at the correct position and for calculating movements with the arms so self-collision does not occur. For this, the virtual model of the grippers is stored as an *stl* file, this overwrites the original gripper models.

The possibility for contacting the PAL Support team is also available at any time for any institution owning any of PAL Robotics' robots if there is any problem with it, hardware-wise or software-wise. This turned out to be useful regarding some problems with the navigation and the package *play_motion* made by PAL Robotics.

5.5 ROS side

In additionally to the network settings, choosing the proper workspaces and the ROS system participants by IP addresses is crucial. This primarily can be done in a file called *.bashrc* usually found in the home directory of a system. In this, the path to the ROS system setup as *source /opt/ros/melodic/setup.bash* and the robot's setup file as *source /opt/pal/ferrum/setup.bash* have to be sourced. As for the ROS addresses, the ROS Master is set to be running on the robot's computer through WebCommander's network settings tab. To be able to reach it with the development laptop, the line *export ROS_MASTER_URI=http ://tiago-117c:11311* has to be added to the *.bashrc* file. The laptop also has to export its IP address, with *export ROS_IP=192.168.1.52* in the current setup but needs to be changed for the correct address on every new setup. These setups make it possible for a continuous connection between the two systems.

For some of the functions to work the *dev_ws* package has to be sourced also, running on the development laptop. Since the connection to the robot through *SSH* only runs in the console, graphical information cannot be passed. For example to view ROS topics regarding the navigation in *rviz*, *rviz* itself has to run on the development laptop but every ROS message can be displayed through it.

The nodes modified are in the *dev_ws* package on the development laptop. With the help of a script *deploy*, these nodes can easily be copied to the robot in a workspace called *deployed_ws*. This contains any modified package or any additional ones that differ from the robot's default setup. On startup, this workspace overwrites and supplements the default packages and nodes. This way during development the original files do not get overwritten and can be restored anytime if something is not working properly.

An important step for the navigation setup is to map the robot's environment. This is done by starting an *rviz* plugin called PAL Map Editor that provides advanced navigation

functionalities, some are:

- download and upload maps
- define Zones Of Interests
- define Points Of Interests

This plugin can be opened and displayed on the development laptop. A picture of the full map in the plugin at IROB is shown in figure 5.16, which corresponds to the planned setup in figure 5.2 entirely.

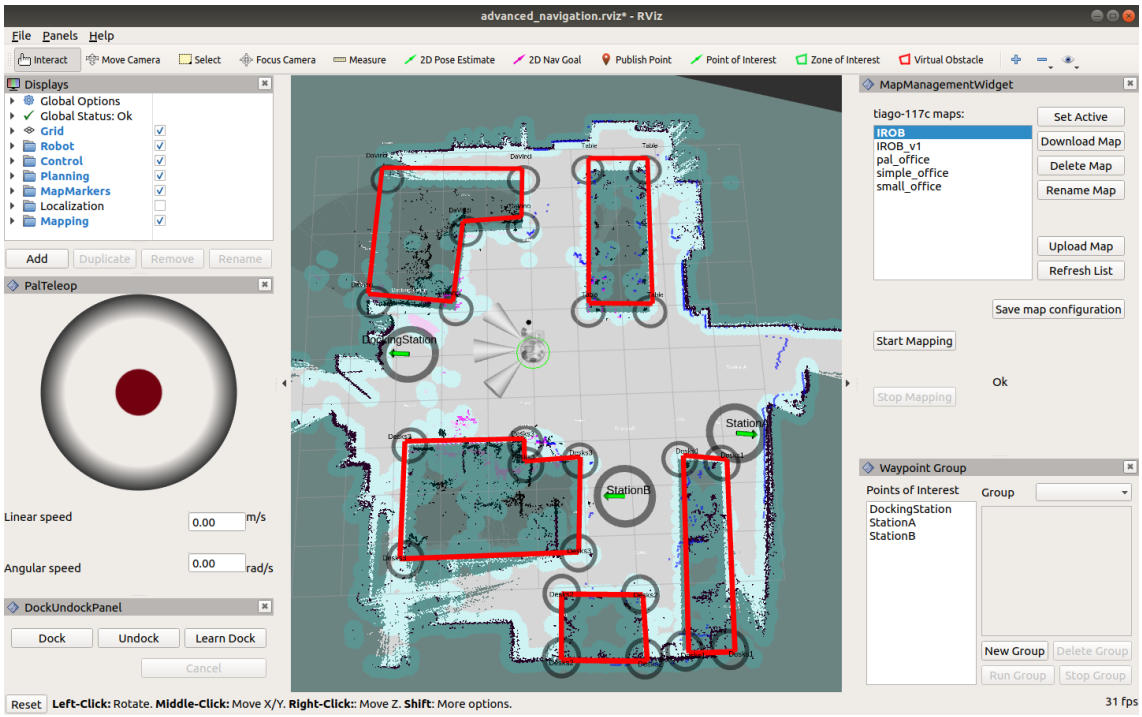


Figure 5.16: Rviz configuration for advanced navigation, legend for the map: grey-free movement, dark grey-out of bounds, areas bordered by red polygons-virtual obstacles

Here the light grey areas are the areas where the robot can move around freely without any fixed obstacles, the darker gray is out of bounds. This can happen in two different ways, one is during mapping when the sensor detects an obstacle like walls and desks, the areas behind them are inaccessible. The other way is to select the obstacles by hand, these are shown as the areas bounded by the red polygons with the tool found at the top, called *Virtual Obstacle*. The marked POIs are also shown with their chosen directions as green arrows.

The robot is shown in the middle in a thin green circle, with the model semi-transparently shown also. Data from the sensors are also shown, with blue dots around the obstacles in front of the robot are the real-time measurements of the built-in LIDAR on the robot's base, the three-cone shapes behind the robot are the sonars with a much shorter range than

the laser.

On the sides, there are panels with useful functions provided, such as having the docking/undocking services as buttons, the *MapManagementWidget* for saving and loading the maps and easy managing of POIs.

5.5.1 Transportation node

The sequence is implemented in a Python script in a custom-made ROS node *transportation*, containing all the motions and service calls. There are some wait times included for the motions to succeed. The script was created as an Action client for later action calls to be possible. One of these action calls is the action *ArucoAction*, which calls the server created in the node described in the next section 5.5.2.

5.5.2 ArUco Motions node

As for the movements using the ArUco Markers, the ROS package *tf* had to be used. This package is keeping track of multiple coordinate frames over time and builds a tree-like structure from the relationships between frames [35] [36]. This has a *lookupTransform* function which defines a transformation matrix between the two frames chosen. In this case, the transformation between the */base_footprint* and the correct ArUco Marker's frame chosen to be */aruco_frame* had to be used. As this package stores all the transformations between the robot model's frames the gripper positions can be easily found out.

With the information at hand, the transformation between the Marker and the gripper can be used as an indicator of where the desired position for the gripper. For this project, the desired position for the grasping and releasing tasks is a few centimeters above the microplate sockets, called Sites. This means a fixed translation in all three directions has to be stored relative to the marker. At this stage of the development, the values for the translation are rooted in the code.

The code for the *tiago_dual_aruco_motions* ROS node was created in C++ and developed as a ROS Action server so when the robot starts up, this node also starts running and anytime an action client calls the created *ArucoMotion* action, the code executes. The recurring use of the node is when the *transportation* node is running and calls the ArUco-based motions.

For the action to work an *.action* file has to be created with the field's goal, feedback, and result. The action got the name *ArucoMotionsAction* with fields

```
# goal
string marker_id
---
# result
bool is_ready
---
#feedback
string feedback_msg
```

These can be used to pass information between an Action client and server. With this created in the *tiago_dual_aruco_motions* node, after a rebuild, the action can be included and used.

5.6 SiLA side

The next step was to include this sequence as SiLA features. This required a SiLA Server and Client implementation, based on the GitLab reference implementation called *sila_python* [4] by the creators, as well as a connection to ROS so the packages can be reached and used.

To use this, the Python environment must be configured for example creating a virtual Python environment with *venv* package for the correct Python version. With command `python3 -m venv sila2` a virtual environment named sila2 will be created then with `source ~/sila2/bin/activate` is activated in the console the command was run in. The packages needed for the SiLA 2 to work can be installed with

```
pip install sila2
pip install sila2[codegen]
pip install sila2[cryptography]
```

By running this after *venv* activation, the package will only be installed in that virtual environment. The architectural diagram for the SiLA and ROS connection is given in figure 4.1 before.

A new Python package has been created called *sila_labware_transfare* for holding all the parts needed for SiLA to work, including the Server and the Client side.

5.6.1 SiLA Server

The SiLA Server is formed through generation with the help of the *sila2[codegen]* package. This works by getting an *.xml* file as input which contains the feature definitions

that need to be implemented. These features can range from simple, basic tasks to more higher-level and complex tasks implemented in one feature. A feature can contain other features implemented or their own commands and/or properties.

For this project, the feature definition proposed by the SiLA Robotics Working Group is used. From the *LabwareTransferController* a SiLA Server was generated. This includes all the properties and commands but only a few are expanded, which are needed for the pick and place task with a mobile robot, by default they are empty at this time of development. More on the feature definitions in section 3.1.4.

The implemented features are:

- PrepareForInput
- PrepareForOutput
- GetLabware
- PutLabware

These contain the corresponding codes carried through from the *transportation* node from section 5.5.1 which is based on section 5.3 following the planned sequence steps.

5.6.2 SiLA Client

The client is created with the SiLA Universal Client, a web interface freely available for SiLA developers. Here a secure connection to the SiLA Server can be made by giving the IP address, and the generated certificate file which must be regenerated and selected at every new connection. All the broadcasted commands and properties are shown here, ready for interaction.

An example of the Client can be seen in figure 5.17 connected to the running Server, showing Server information.

In figure 5.18 the Client's Properties tab is open. Here the properties are shown as an expandable list with items following each other, divided by different feature categories. In this instance, the Client is calling for the property *Server Description* by pressing the *READ ONCE* buttons. This reaches out to the Server, getting a string as a result.

This way all the implemented properties and commands can be called, showing simple or more complex results. For giving commands from the *Labware Transfer Manipulator Controller* command list, the corresponding ROS Action is called and the robot can start its task.

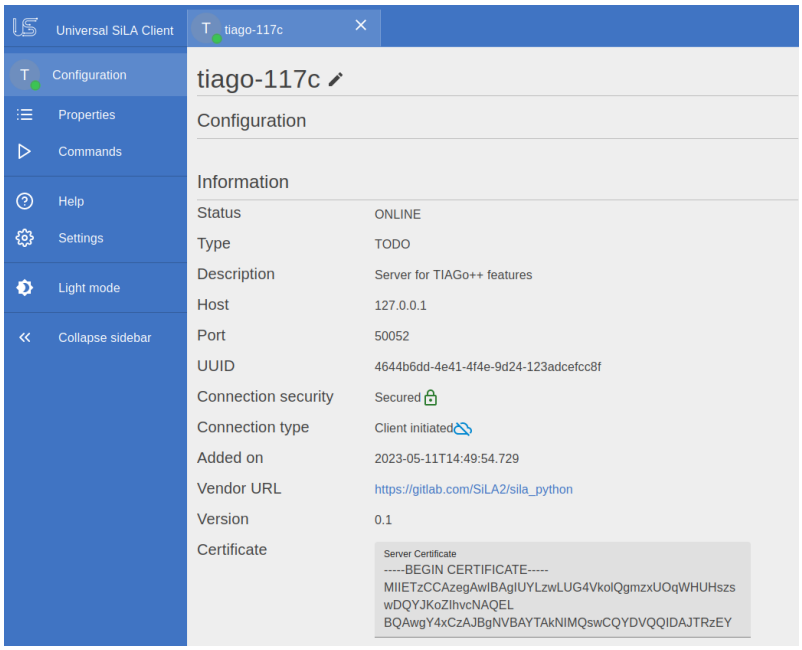


Figure 5.17: SiLA Universal Client, Server information

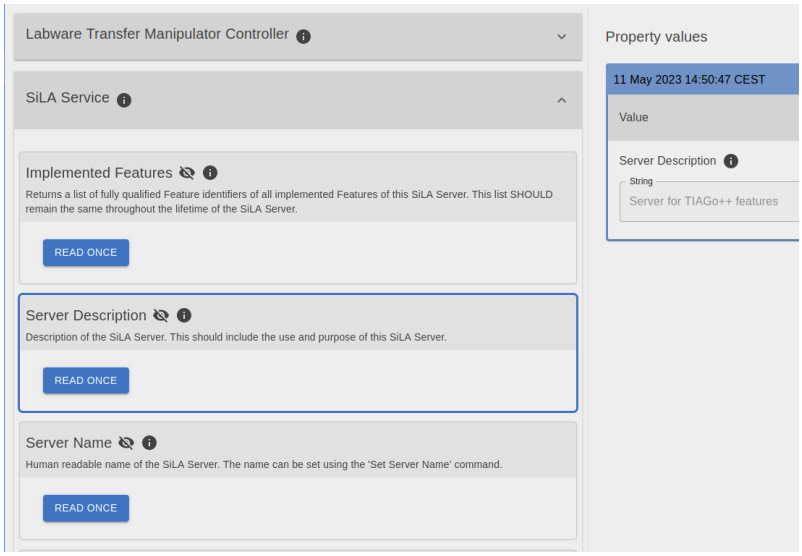


Figure 5.18: SiLA Universal Client, getting property value

5.6.3 SiLA bridge

As mentioned before and seen in image 4.1, a middle node is needed for connecting the two sides. The *tiago_dual_aruco_motions* node acts as the bridge in this project, meaning it acts both as the SiLA Server and the ROS Action client. The prototype implementation of the SiLA-ROS-bridge is on GitLab [37]. When a command or property is queried from the client, the bridge passes the call to the ROS Actions.

6 TESTING

Setting up the testing environment at both locations has been done during the initial setup and configuration phase as the stations had to be selected at an early stage for mapping purposes.

The testing phase consists of the same pick-and-place transportation based on ROS in addition to the verified SiLA Feature Definitions for the *LabwareTransfareController* feature. The testing environment is done in the lab at the selected stations and the transported object. For testing at the Company's laboratory, a placeholder box has been used instead of a microplate for safety purposes and in stages where the motions based on the ArUco placement had not been precise yet. In the IROB laboratory, the object has been replaced with microplates for a more realistic testing scenario.

An important viewpoint for testing is the accuracy of the robot while handling the microplates. This turned out to be precise and accurate enough for handling objects with the size of the microplates. As the 3D-printed plate holders are the correct size for holding the plates in place, the placement is accurate enough to be placed in the holder correctly even with very little room for error. After using the ArUco-based motions multiple times throughout development, the gripper positioning always turned out to be within the threshold for it to be able to grab the objects.

In figure 6.1 the robot's head camera image as well as the octomap built by the robot, seen as blue squares can be seen. This shows the tool-center-point position as a coordinate frame on the right and the frame of the ArUco frame in relation.

A picure of the robot executing the grasping of the microplate is shown in figure 6.2.

Further testing will follow on this topic with repeated motions with only picking and placing the microplates in a looped manner.

The SiLA features can be tested by calling commands and properties through the Universal Client. Further testing is needed as part of the upcoming development.

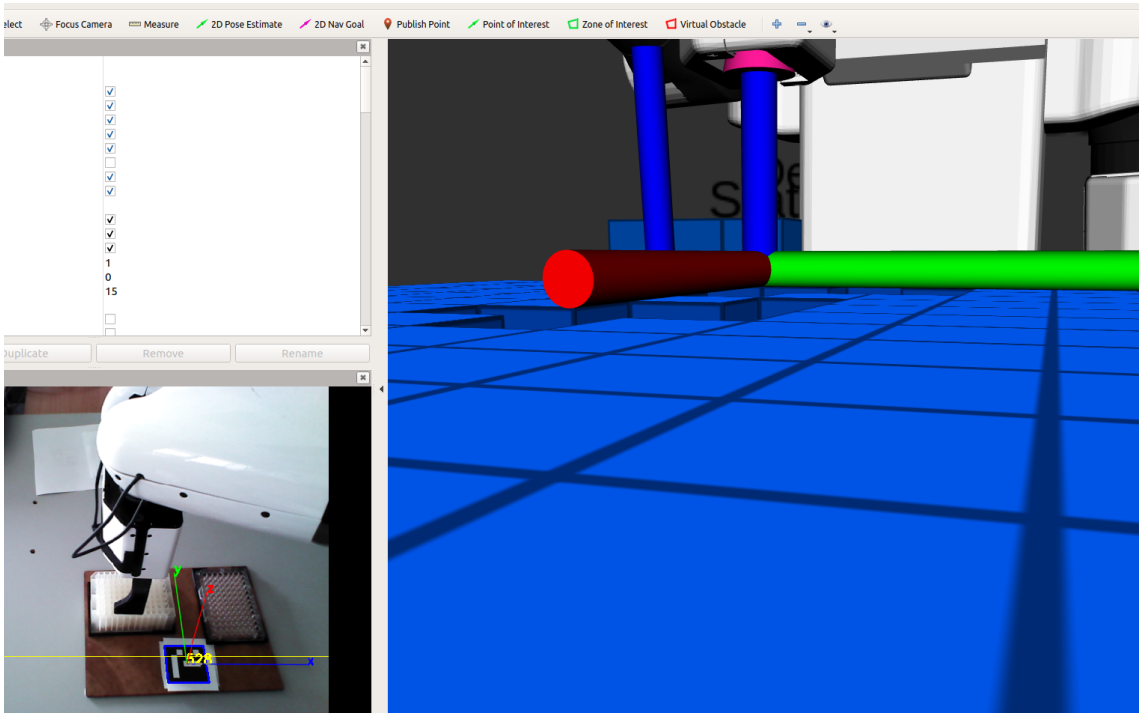


Figure 6.1: ArUco motion testing tracked in rviz

Figure 6.2: TIAGo++, grasping the microplate, Station B

7 DEVELOPER AND USER DOCUMENTATION

As the project in its current phase is a prototype, there is no strict disconnection between the developer and the user, a person acts as both parties. On this note, the documentation is mixed but gives clear instructions on how to use the hardware and software in its current state.

7.1 Kanban board

Documenting the progress of the development has been an important aspect of the development process. For this, a kanban board in the online tool Trello is used where the states of tasks and sub-tasks can be tracked and updated. This was also shared with the supervisors of the project. Multiple labels were created for different types of tasks, which are

- Literature, for reading about a topic, searching for suitable solutions
- Hardware, for working on the robot and the workstations
- Setup, for making the initial steps before the main part of the development
- LAPP, steps for implementing the LAPP framework [13]
- Simulation, for setting up and working without the robot on Gazebo
- Documentation, for making notes and recording important steps during development
- Software, for working on the implementation through the development laptop

In figure 7.1 a snippet from this board can be seen during the development phase.

The colors represent the labels mentioned in the list above, the three columns are the phase where they take place in the implementation. There are also sub-tasks as checklists inside cards.

7.2 Visual map

Another way for tracking the project steps and tasks is to make a so-called visual map or mind map, with visual indications on each task. For this, an intuitive and easy-to-use online tool, Miro was used. This is a convenient tool for outlining concepts, planning system architecture, placing components, and correlating tasks with them. The main node in the middle from which the tasks branch out is *Pick and Place* since this is the main goal. All tasks have to be finished for the main task to be able to work.

For example, the steps for the transportation sequence in the visual map can be seen on figure 7.2.

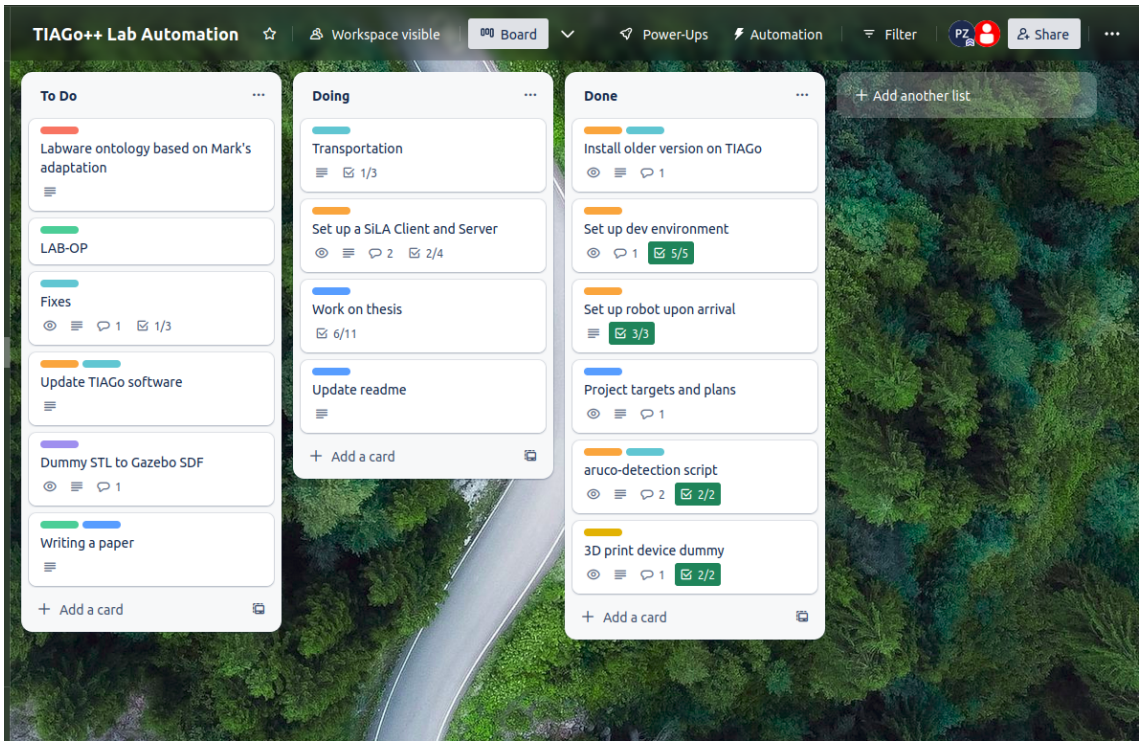


Figure 7.1: Trello kanban board

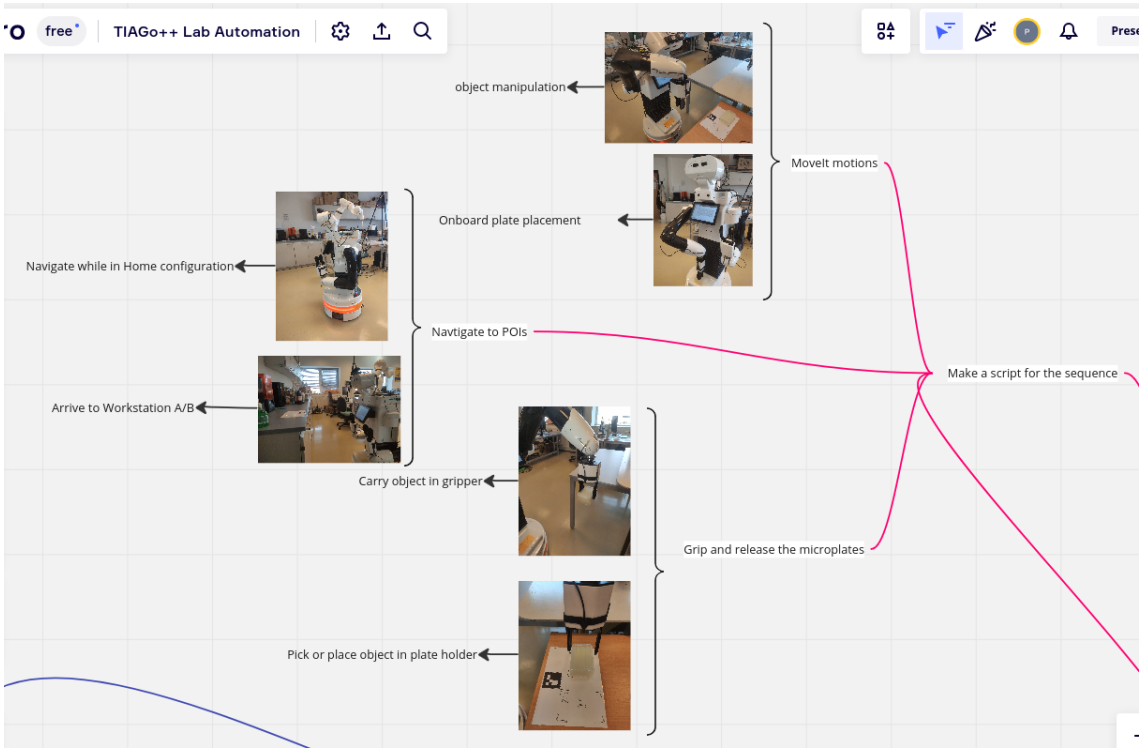


Figure 7.2: Visual map of transportation sequence

On the right side, the workspace setup reserves an area, seen in figure 7.3.

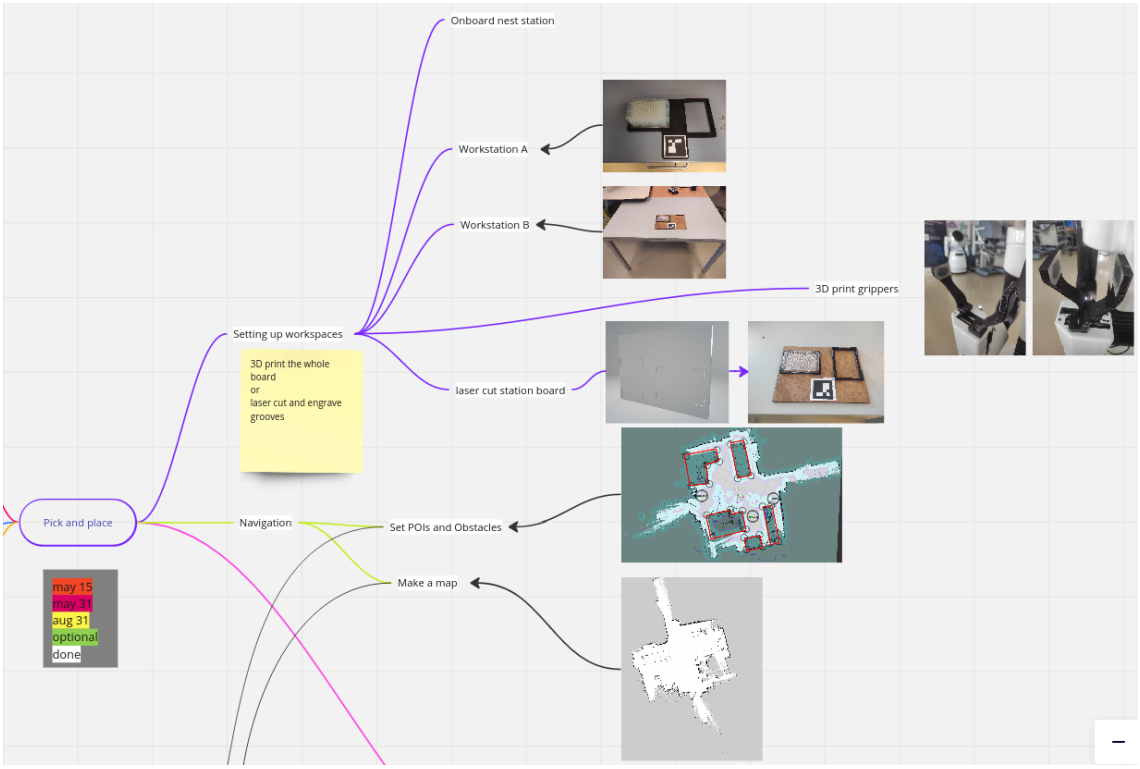


Figure 7.3: Visual map of the workspace creation

Every other branch follows the same mentality with pictures included on the corresponding topics and with each node, it depends on their child nodes.

The main nodes branching directly from the main task are

- Make a script for the sequence
- Make a SiLA client
- LAPP
- Setting up workspaces
- Navigation
- Image recognition

Also included on the board is a chart containing the correlation between the *LabwareTransferController* features and the tasks, and sub-tasks on lower levels such as ROS actions, services, and topics. This figure 7.4 is also helpful for understanding the different levels of the whole process.

On image 7.4 the black rectangles represent the structure of the *LabwareTransferController* with different levels based on a hierarchical decomposition. The levels from the highest to lowest are:

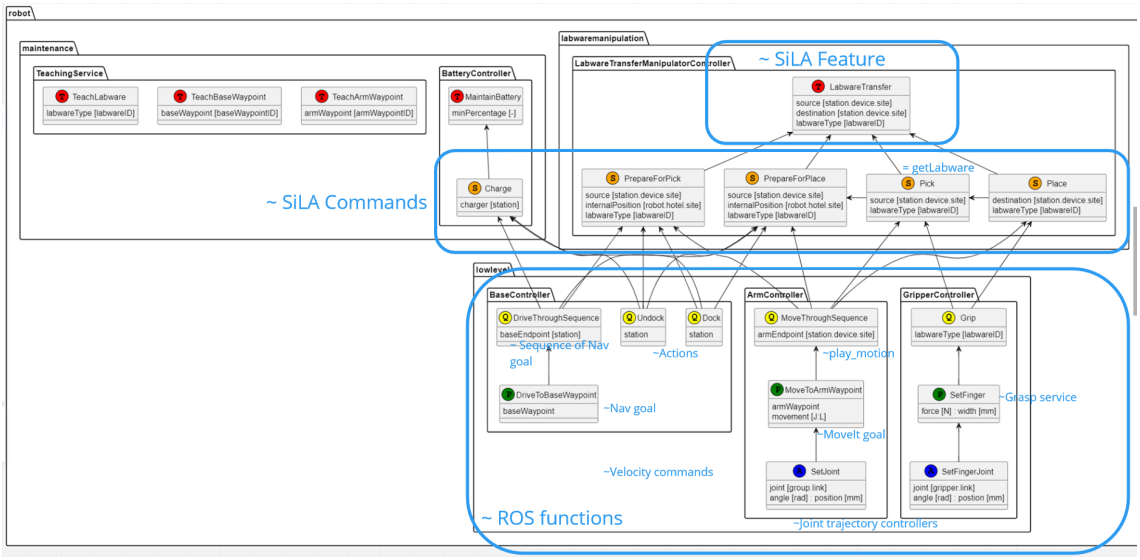


Figure 7.4: LabwareTransferController levels

- Service
- Procedure
- Task
- Subtask
- Motion sequence
- Motion primitive
- Actuator primitive

From this, Task and Subtask levels correspond to SiLA features and commands, Motion sequences, and primitives to ROS functionalities implemented as Actions, Services, or Topic subscriptions.

This hierarchical decomposition was presented at the Pharma Automation & Robotics event in London, UK [38].

7.3 Step by step documentation

The written documentation was done as a markdown document also found in the supplementary materials as *Project TIAGo.html*. Its structure and some important remarks are the following.

7.3.1 Setup guide

To start the robot, the arm positions have to be lying on the sides of the robot base. The buttons on the base are shown in figure 7.5.

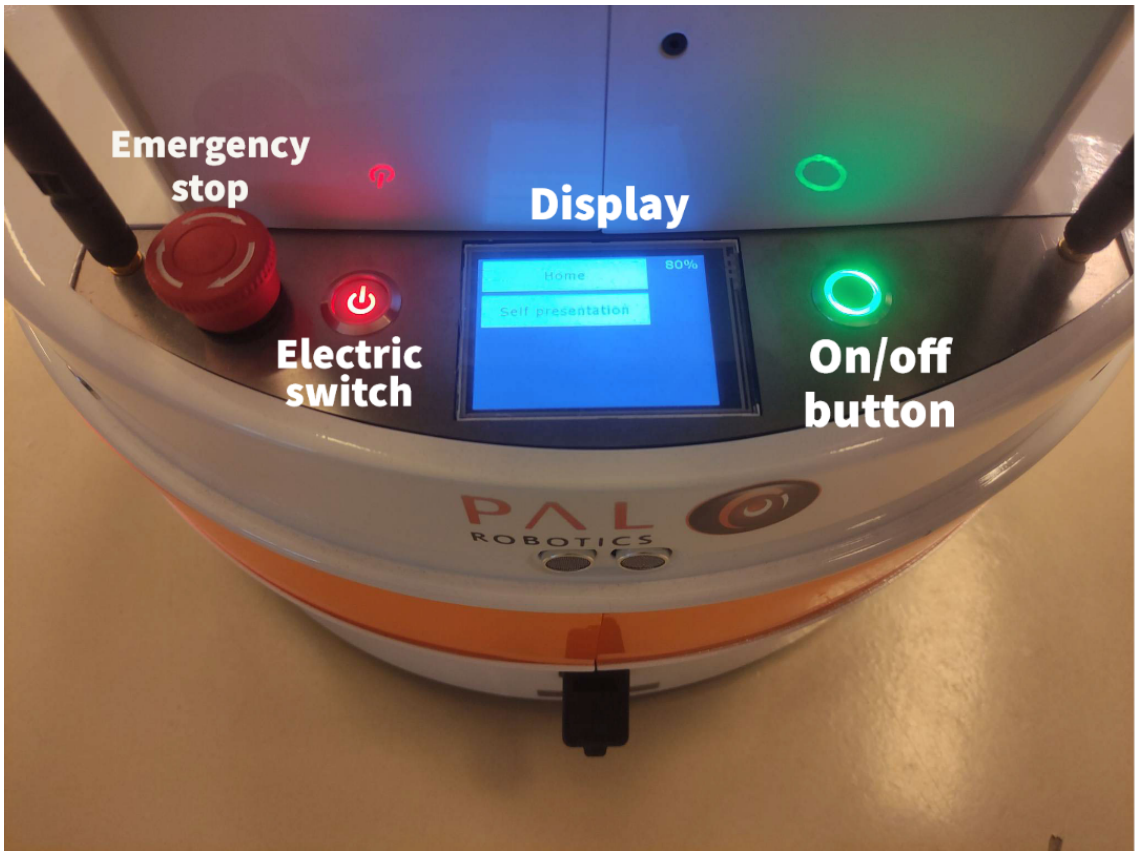


Figure 7.5: Robot mobile base buttons

From left to right, the buttons are the emergency stop button, the electric switch glowing red, and the on/off button in green. In the middle, the information display, where the robot's battery state and some arbitrary, configurable buttons can be seen on the touchscreen. For turning the robot on, the electric switch has to be pushed once then the on/off button is held for a few moments until it starts blinking. In a turned-off state, these buttons and the display do not light up.

The steps for creating a virtual machine are gathered if needed, and network configurations for each location, both natively or in a virtual machine. Both graphical configurations are in Ubuntu's Network Settings interface and in a terminal are included.

On the development laptop, in additionally to configuring the network for connecting either through WiFi or Ethernet, the file *.bashrc* has to be configured by the following lines, in case the IP address of the development laptop is 192.168.1.52 as a fixed address and the ROS version is Melodic.

```
source init_pal_env.sh
source /opt/ros/melodic/setup.bash
source /opt/pal/ferrum/setup.bash
```

```
export ROS_MASTER_URI=http://tiago-117c:11311
export ROS_IP=192.168.1.52
```

This way the robot and the laptop can communicate through ROS. This configuration was set up in the IROB laboratory.

SSH also has to be configured on the laptop for it to connect to the robot's filesystem. The *config* file at */.ssh/config* has to contain the following.

```
Host tiago-117c
HostName 192.168.8.160
User pal
```

To connect to the robot through *ssh*, the next command has to be run from a terminal.

```
ssh tiago-117c
```

The following software has to be installed manually on the development laptop.

- Visual Studio Code with extensions Git Graph, ROS, C/C++, Python
- *dev_ws* repository, where modified and additional packages are stored
- Additional ROS packages, if needed

ROS packages can be installed with the following command with the *apt-get* package.

```
sudo apt-get install ros-melodic-<name_of_package>
```

To reach the repositories correctly where the *apt-get* package searches for files, they have to be included in file */etc/apt/sources.list*.

To be able to use packages, a workspace has to be created and built with the commands.

```
mkdir -p ~/catkin_ws/src -- create a workspace
cd ~/catkin_ws/          -- go in the created directory
catkin build             -- build the workspace
```

For setting up Python correctly on the development laptop, helping websites are linked in the documentation.

When the usage of Python is needed, the next line has to be run to activate the virtual environment.

```
source ~/sila2/bin/activate
```

SiLA 2 has the Python package which has to be installed to use the SiLA implementation with additional packages. It is installed with

```
pip install sila2
pip install numpy
pip install sila2[codegen]
pip install sila2[cryptography]
```

To generate a SiLA Server from an *.xml* file containing a feature definition, use:

```
sila2-codegen new-package -n sila2_test \
    ./ExampleFeatures-v1_0.sila.xml
```

To set up the Universal SiLA Client, follow the steps on the GitLab repository[4]

To start the Server, use

```
python -m labware_transfare --ca-export-file ca.pem
```

This generates a certification file called *ca.pem* which has to be given through the Client when connecting to a Server.

7.3.2 Development

The deploy script has to be called with

```
rosrun pal_deploy deploy.py tiago-117c -p <name_of_package>
```

The packages needed to be deployed from the *dev_ws* are

- tiago_dual_bringup
- pal_gripper
- pal_gripper_description
- tiago_dual_moveit_config
- tiago_dual_aruco_motions
- transportation

These are the manually modified nodes and the two are additionally created at the bottom. These two nodes are started manually from the development computer as of now but can also be started at robot startup. The commands for starting them are the following.

```
roslaunch ~/dev_ws/src/tiago_dual_aruco_motions \
    /launch/aruco_action.launch
python ~/dev_ws/src/transportation/nodes \
    /transportationScript.py
```

With the next line, the file system can be visually mapped to the laptop's storage manually from the terminal, with *sshfs*

```
sudo sshfs -o allow_other root@tiago-117c:/ /mnt/tiago
```

Some commonly used commands during development arise, such as launching *rviz* configuration files for better visualization, launching the developed nodes, and other services from the terminal. There are listed in the documentation. The two most important are the following.

```
rosrun rviz rviz -d `rospack find tiago_dual_2dnav`/config \
    /rviz/advanced_navigation.rviz -- for navigating
rosrun rviz rviz -d /home/pal/dev_ws/src/aruco_eef/config \
    /rviz/aruco_tf.rviz -- for testing ArUco detection
```

The used ArUco marker at both stations has ID 528.

7.3.3 Other

At the top of the guide, hardware specifications can be found about the robot and the development laptops used at both project locations.

Other than this, a Troubleshooting section at the bottom can be seen where any upcoming problems, if solved, are featured here.

8 RESULTS

The initial phase of the project gave me an understanding of the history and concepts of laboratory automation, including the automation levels, most importantly, the Total Laboratory Automation with its advantages and disadvantages. It is clear that mobile robotic automation in a laboratory environment can make processes faster and safer. However, as a precondition, the robots have to be able to operate in unstructured human-centric environments.

The viewpoint of keeping the implementation as simple as possible but still representative was kept in mind during the whole project and will be an important aspect. The goal is to successfully demonstrate an automated mobile robotic application for labware transportation in a pharmaceutical laboratory environment.

Creating the workstations and managing the experimental environments in two physical locations gave me an in-depth understanding of the realistic representation of a real laboratory. Through this experience, the environment models were prepared considering the common laboratory target objects and mechanical interfaces.

Working with TIAGo++ allowed me to be involved in a complex mobile robotics system integration using the Robot Operating System for managing the use cases in terms of interfacing and flow control. Every major aspect of a common mobile robot system has been included and implemented in the project, ranging from navigation, planned motions with collision detection, writing ROS nodes for different tasks, etc. Learning how to work with a complex mobile robot running on ROS, gave me the knowledge and understanding for a more fluent launch of any upcoming project.

Throughout the learning, development, and implementation process, SiLA has proven its very important role in laboratory automation. It lets the developer implement features on wildly used laboratory devices and robots. SiLA 2 is a good approach for standardizing frameworks in these environments, so getting them more recognition for parties interested in robotics or laboratory automation is most important. Taking part in the gatherings of the SiLA Robotics Working Group and working together on the Feature Unification proposition gave an insight into the inner workings of SiLA as a group and the basic ideas behind the framework. Participation and involvement in the fourth BioSASH Hackathon also added to this knowledge. The results of the Hackathon gave a base for the SiLA implementation for the diploma work.

This reference implementation presented in the diploma work is useful for other SiLA developers and an important step in the direction of using mobile robots in the lab. Implementing TIAGo++ in a SiLA environment serves as a basis for further integration tasks, which can be useful in the field of robotic automation.

My diploma work fits into a broader endeavor, which aims to achieve plug & play integration for laboratory robots, as mentioned in section 9.1. The different milestones of this initiative can be assigned to the technology readiness level (TRL) scale, as presented in [39]. The Laboratory Automation Plug & Play (LAPP) initiative spans across the SiLA robotics working group, the Company's internal proof-of-concept project, and the present diploma work.

The concept is shown in figure 8.1. The results of the diploma work can be placed on Level 4 of the TRL table.

TRL	Definition
1	Basic principles observed
2	Technology concept formulated
3	Experimental proof of concept
4	Technology validated in lab
5	Technology validated in relevant environment (industrially relevant environment in the case of key enabling technologies)
6	Technology demonstrated in relevant environment (industrially relevant environment in the case of key enabling technologies)
7	System prototype demonstration in operational environment
8	System complete and qualified
9	Actual system proven in operational environment (competitive manufacturing in the case of key enabling technologies; or in space)

Figure 8.1: Technology Readiness Levels

A video made at the University is enclosed with the supplementary materials which shows the whole process done automatically by the TIAGo++.

9 FURTHER DEVELOPMENT

9.1 LAPP

As this project is part of a bigger project at the Company, further steps will be taken to be able to make this solution for lab automation with mobile robots widespread amongst pharmaceutical companies. This means the project will surely be continued for at least three months after the end of the semester.

An anticipated next step regarding the project is to use the Laboratory Automation Plug & Play (LAPP) concept. This enables the framework to store important information on laboratory devices (version, position, etc.) and get them from a database [13] [40]. Two visual markers (bar code, ArUco markers, etc.) are located on all laboratory devices. When a robot with a camera approaches the device and reads the code, information from the database is gathered and used for further commands or actions.

This concept would replace some hard-coded values from the current implementation such as the position needed for grasping the microplates relative to the ArUco marker.

9.2 Other robots

Another improvement would be to use another robot, especially one which is made directly for more industrial use and not only for research.

Since this project is based on a demonstration or proof-of-concept, using a robot specialized for this task in a real environment would require much development regarding safety. For real-life use, multiple exception handlers have to be implemented so the automation process can run so it is undisturbed and continuously.

9.3 Multiple robots

An approximate final goal for the project is to start the implementation with multiple lab devices for them to be able to work together. This ranges from any stationary, mobile, basic to complex solutions as long as they are capable of working together and with human workers in the lab.

For this, a solution where the integration of a second stationary robot is done seems to be a valid next step or another project.

9.4 SiLA further developments

Since this project is also part of a collaboration with the SiLA Robotics Working Group, sharing the finding and the example implementation could be a following step in the direction of making the SiLA standard more popular. A reference implementation like this one is helping the community to improve the quality and quantity of the implementation base and the examples.

This can lead to more participation in future Hackathons like the BioSASH or even more collaborations with similar projects to share the results with and make the process of standardizing laboratory automation faster.

10 ÖSSZEFOGLALÁS

A diplomamunka alaposan bemutatta a laboratóriumi automatizálás áttekintését a legfontosabb korlátokkal és előnyökkel, különösen a TLA (teljes laboratóriumi automatizálás) területén. A mobil robotok és a mobil manipulátorok bevonása egy ilyen környezetbe különböző kihívásokat vet fel az emberek és robotok közötti teljesen kollaboratív megoldás tekintetében.

A SiLA mind keretrendszer, mind szabvány, részletes koncepcionális leírása egyaránt tárgyalásra kerül. A munka során számos együttműködés történt különböző szervezetekkel, amelyeket átfogóan ismertetünk, mint például a SiLA Robotics Working Group (SiLA robotikai munkacsoport) és a negyedik bioSASH Hackathon.

A ROS működése is összefoglalásra került, hogy az olvasó megfelelő háttérismereteket kapjon ahhoz, hogy megértse, hogyan működik egy robot a keretrendszerrel.

A tervezési fázis egy egyszerű, de hatékony felállást javasolt. A megvalósításba a SiLA és a ROS együttes bevonását vezeti fel, valamint az elvárt technológiákra való tekintettel a kettő közötti kapcsolat megteremtésére. A projekt fő célja, egy folyadékhordozó mikrolemez mobil manipulátorral történő szállítása megvalósíthatóvá vált.

A megvalósítási fázis két helyszínen, egy bécsi gyógyszergyár R&D telephelyén és az Óbudai Egyetem Bejczy Antal Robottechnikai Közont Robotikai Laboratóriumában, Budapesten zajlott. A kialakított kísérleti környezetek kialakítása egy valós laboratóriumot reprezentál, ahol a mikrolemezek kezelése mindennapos feladat, amelyet az emberek kézzel végeznek.

A TIAGo++ egy kutatás-központú mobil manipulátor, amely különösen alkalmas demonstrációs célokra. A robot hardver- és szoftverkörnyezetét a projekt szempontjából döntő fontosságú, minden szempontból részletesen tárgyaljuk.

A megvalósítás első lépései a ROS-környezet helyes beállítását, valamint minden szoftvercsomag és ROS node helyes telepítését és használatát foglalja magába. A ROS-rétegen egy SiLA szerver-kliens implementáció helyezkedik el. Az USC (univerzális SiLA kliens) felhasználóbarát felületet biztosít a laboratóriumi eszközökkel való interakcióhoz, a szerveroldal pedig képes csatlakozni az előzetesen megvalósított ROS Action-ökhoz és Service-ekhez.

A projekt fontos szempontja az olyan menedzsmenteszközök használata, mint a Trello, a Miro board, a GitHub és az írásos dokumentáció. Ezek használatát és tartalmát a munka bemutatja.

11 SUMMARY

The diploma work presented an overview of laboratory automation thoroughly with the most important limitations and advantages, especially in the field of Total Laboratory Automation. The inclusion of mobile robots and mobile manipulatory in such an environment proposes different challenges regarding a fully collaborative solution between humans and robots.

An in-depth conceptual description of the SiLA framework and standard are discussed equally. Several collaborations with different organizations happened during the work and are explained on a comprehensive level, such as the work with the SiLA Robotics Working Group and the fourth bioSASH Hackathon.

The operation of ROS was also summarized to give adequate background knowledge for the reader to understand how a robot works with this framework.

The planning phase proposed a simple but effective setup for including both SiLA and ROS in the implementation and making the connection between them by paying attention to the required technologies. With this, the main goal of the project, transporting a liquid carrier microplate using a mobile manipulator was set up and ready to be implemented.

During the implementation phase, two locations, one at an R&D site of a Pharmaceutical Company in Vienna and the other at the IROB Robotics Laboratory at Óbuda University, Budapest were involved. The experimental areas were formed in an effective way, representing a real-life laboratory where handling microplates is an everyday task, done by humans manually.

TIAGo++ is a research-centered mobile manipulator especially useful for demonstrational applications. The hardware and software environment of the robot is discussed extensively in every aspect crucial for this project.

The first steps of implementation meant the correct setup of the ROS environment and installing and using every package and node correctly. On the ROS layer, a SiLA server-client implementation is present. The Universal SiLA Client gives a user-friendly interface for interacting with laboratory devices and the Server side is able to connect to the ROS Actions and Services implemented beforehand.

An important aspect of the project is the usage of project management tools like Trello, Miro board, GitHub, and written documentation. The use of these and their contents are presented in the work.

12 BIBLIOGRAPHY

[1] "SiLA rapid integration." (Aug. 22, 2017), [Online]. Available: https://sila-standard.com/ (visited on 05/18/2022).

[2] "TIAGo - the mobile manipulator robot platform for your research." (), [Online]. Available: https://pal-robotics.com/robots/tiago/ (visited on 05/13/2022).

[3] J. Pages, L. Marchionni, and F. Ferro, "TIAGo: The modular robot that adapts to different research needs,"

[4] "SiLA2 / universal_sila_client / sila_universal_client · GitLab," GitLab. (Apr. 25, 2023), [Online]. Available: https://gitlab.com/SiLA2/universal-sila-client/sila_universal_client (visited on 05/08/2023).

[5] J. Boyd, "Robotic laboratory automation," *Science*, vol. 295, no. 5554, pp. 517–518, Jan. 18, 2002, Publisher: American Association for the Advancement of Science. DOI: 10.1126/science.295.5554.517.

[6] A. Gasparetto and L. Scalera, "From the unimate to the delta robot: The early decades of industrial robotics: Proceedings of the 2018 HMM IFToMM symposium on history of machines and mechanisms," in *History of Mechanism and Machine Science*, Journal Abbreviation: History of Mechanism and Machine Science, Jan. 1, 2019, pp. 284–295, ISBN: 978-3-030-03537-2. DOI: 10.1007/978-3-030-03538-9_23.

[7] "Devices and systems for laboratory automation | wiley," Wiley.com. (), [Online]. Available: https://www.wiley.com/en-us/Devices+and+Systems+for+Laboratory+Automation-p-9783527348329 (visited on 05/13/2023).

[8] Á. Wolf and K. Széll, "A review on robotics in life science automation," Nov. 14, 2019.

[9] G. Lippi and G. D. Rin, "Advantages and limitations of total laboratory automation: A personal overview," *Clinical Chemistry and Laboratory Medicine (CCLM)*, vol. 57, no. 6, pp. 802–811, Jun. 1, 2019, Publisher: De Gruyter, ISSN: 1437-4331. DOI: 10.1515/cclm-2018-1323.

[10] I. Holland and J. A. Davies, "Automation in the life science research laboratory," *Frontiers in Bioengineering and Biotechnology*, vol. 8, 2020, ISSN: 2296-4185.

[11] A. Croxatto, G. Prod'hom, F. Faverjon, Y. Rochais, and G. Greub, "Laboratory automation in clinical bacteriology: What system to choose?" *Clinical Microbiology and Infection*, vol. 22, no. 3, pp. 217–235, Mar. 1, 2016, ISSN: 1198-743X. DOI: 10.1016/j.cmi.2015.09.030.

[12] K. Thurow, "System concepts for robots in life science applications," *Applied Sciences*, vol. 12, no. 7, p. 3257, Jan. 2022, Number: 7 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2076-3417. DOI: 10.3390/app12073257.

[13] Á. Wolf, D. Wolton, J. Trapl, *et al.*, "Towards robotic laboratory automation plug & play: The "LAPP" framework," *SLAS Technology*, vol. 27, no. 1, pp. 18–25, Feb. 1, 2022, ISSN: 2472-6303. DOI: 10.1016/j.slast.2021.11.003.

[14] H. Bär, R. Hochstrasser, and B. Papenfuß, "SiLA: Basic standards for rapid integration in laboratory automation," *Journal of Laboratory Automation*, vol. 17, no. 2, pp. 86–95, Apr. 1, 2012, Publisher: SAGE Publications Inc, ISSN: 2211-0682. DOI: 10.1177/2211068211424550.

[15] H. Liu, N. Stoll, S. Junginger, and K. Thurow, "Mobile robot for life science automation," *International Journal of Advanced Robotic Systems*, vol. 10, no. 7, p. 288, Jul. 1, 2013, Publisher: SAGE Publications, ISSN: 1729-8806. DOI: 10.5772/56670.

[16] J. Bai, L. Cao, S. Mosbach, J. Akroyd, A. A. Lapkin, and M. Kraft, "From platform to knowledge graph: Evolution of laboratory automation," *JACS Au*, vol. 2, no. 2, pp. 292–309, Feb. 28, 2022, Publisher: American Chemical Society. DOI: 10.1021/jacsau.1c00438.

[17] "SiLA documentation." (), [Online]. Available: https://sila2.gitlab.io/sila_base/ (visited on 05/18/2022).

[18] "Standards | SiLA rapid integration." (Aug. 24, 2017), [Online]. Available: https://sila-standard.com/standards/ (visited on 05/18/2022).

[19] "SiLA2 · GitLab," GitLab. (), [Online]. Available: https://gitlab.com/SiLA2 (visited on 05/19/2022).

[20] "Products | SiLA rapid integration." (), [Online]. Available: https://sila-standard.com/product-store/ (visited on 05/18/2022).

[21] M. Porr, S. Schwarz, F. Lange, *et al.*, "Bringing IoT to the lab: SiLA2 and open-source-powered gateway module for integrating legacy devices into the digital laboratory," *HardwareX*, vol. 8, e00118, Oct. 1, 2020, ISSN: 2468-0672. DOI: 10.1016/j.ohx.2020.e00118.

[22] "API reference — ur_rtde 1.5.5 documentation." (), [Online]. Available: https://sdurobotics.gitlab.io/ur_rtde/api/api.html#rtde-control-api (visited on 12/15/2022).

[23] SiLA Consortium. "bioSASH final result of hackathon: Pick & place labware transportation with benchtop and robots." (2022), [Online]. Available: https://www.youtube.com/watch?v=9_6XLgyhDl8 (visited on 12/15/2022).

[24] "bioSASH." (Jan. 16, 2023), [Online]. Available: https://www.biolago.org/en/projects/details/biosash.html (visited on 05/12/2023).

[25] "Home." (), [Online]. Available: https://www.biolago.org/en/ (visited on 12/15/2022).

[26] "ROS: Home." (), [Online]. Available: https://www.ros.org/ (visited on 12/13/2022).

[27] M. Quigley, K. Conley, B. Gerkey, *et al.* "ROS: An open-source robot operating system." (Jan. 1, 2009).

[28] "Documentation - ROS wiki." (), [Online]. Available: http://wiki.ros.org/Documentation (visited on 12/13/2022).

[29] P. Robotics, *TIAGo++ Handbook*. Barcelona, 2020.

[30] "SSH secure shell home page, maintained by SSH protocol inventor tatu ylonen. SSH clients, servers, tutorials, how-tos." (), [Online]. Available: https://www.ssh.com/academy/ssh (visited on 12/13/2022).

[31] "MoveIt motion planning framework." (), [Online]. Available: https://moveit.ros.org/ (visited on 12/13/2022).

[32] "Gazebo." (), [Online]. Available: https://gazebosim.org/home (visited on 12/13/2022).

[33] "Robots/TIAGo++/tutorials - ROS wiki." (), [Online]. Available: http://wiki.ros.org/Robots/TIAGo%5C%2B%5C%2B/Tutorials (visited on 12/13/2022).

[34] "Visual studio code - code editing. redefined." (), [Online]. Available: https://code.visualstudio.com/ (visited on 05/09/2023).

[35] "Tf - ROS wiki." (), [Online]. Available: http://wiki.ros.org/tf (visited on 05/06/2023).

[36] T. Foote, "Tf: The transform library," in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, ser. Open-Source Software workshop, Apr. 2013, pp. 1–6. DOI: 10.1109/TePRA.2013.6556373.

[37] "SiLA2 / sila_robotics / sila_ros · GitLab," GitLab. (), [Online]. Available: https://gitlab.com/SiLA2/sila_robotics/sila_ros (visited on 05/08/2023).

[38] Á. Wolf, "Reference architecture model for the integration of lab robots," paper, paper, Pharma Automation & Robotics, London, UK, Apr. 20, 2023.

[39] Á. Wolf, "Teaching-free robot integration with LAPP digital twin," presentation (preprint), presentation (preprint), Future Labs Live, Basel, Switzerland, May 30, 2023.

[40] Á. Wolf, S. Romeder-Finger, K. Széll, and P. Galambos, "Towards robotic laboratory automation plug & play: Survey and concept proposal on teaching-free robot integration with the lapp digital twin," *SLAS Technology*, vol. 28, no. 2, pp. 82–88, Apr. 1, 2023, Publisher: Elsevier, ISSN: 2472-6303, 2472-6311. DOI: 10.1016/j.slast.2023.01.003.

13 LIST OF FIGURES

14 ACRONYMS

SiLA	Standardization in Lab Automation
API	Application Programming Interface
ROS	Robot Operating System
IMU	Inertial Measurement Unit
GUI	Graphical User Interface
POI	Point Of Interest
R&D	Research and Development
USC	Universal SiLA Client
TRL	technology readiness levels
URDF	Unified Robotics Description Format
SSH	Secure Shell or Secure Socket Shell