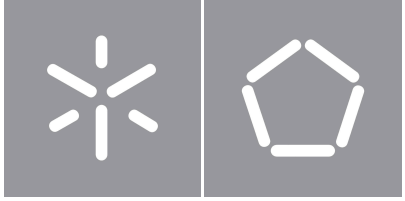
University of Minho
School of Engineering

Guilherme da Silva Amorim Martins

Development of a front-end application for a Human-Robot collaborative framework

october 2023



University of Minho
School of Engineering

Guilherme da Silva Amorim Martins

Development of a front-end application for a Human-Robot collaborative framework

Masters Dissertation
Master's in Informatics Engineering

Dissertation supervised by
Cristina Manuela Peixoto dos Santos

october 2023

Copyright and Terms of Use for Third Party Work

License granted to users of this work:

[Caso o autor pretenda usar uma das licenças Creative Commons, deve escolher e deixar apenas um dos seguintes ícones e respetivo lettering e URL, eliminando o texto em itálico que se lhe segue. Contudo, é possível optar por outro tipo de licença, devendo, nesse caso, ser incluída a informação necessária adaptando devidamente esta minuta]

CC BY

https://creativecommons.org/licenses/by/4.0/ *[Esta licença permite que outros distribuam, remixem, adaptem e criem a partir do seu trabalho, mesmo para fins comerciais, desde que lhe atribuam o devido crédito pela criação original. É a licença mais flexível de todas as licenças disponíveis. É recomendada para maximizar a disseminação e uso dos materiais licenciados.]*

Acknowledgements

Aos meus pais, quero expressar o meu mais profundo agradecimento por todo o apoio incansável que me deram ao longo de todo o meu percurso académico. Compreendo que, por vezes, tenha sido desafiador suportar as despesas financeiras associadas, mas a vossa inabalável dedicação nunca vacilou. O vosso amor e encorajamento fizeram com que esta jornada fosse possível.

Um agradecimento muito especial ao meu único irmão, que sempre foi o meu exemplo e guia, orientando-me no melhor caminho em todas as circunstâncias. Considero-te um dos meus três principais pilares, e acredita que esta conquista também é tua.

Aos meus colegas de curso, quero expressar a minha gratidão pela vossa amizade e apoio ao longo dos últimos cinco anos. Sem a vossa camaradagem, este percurso teria sido muito mais difícil. Juntos, enfrentámos desafios e celebrámos sucessos.

Aos meus amigos de infância, que sempre estiveram presentes para me ajudar a relaxar e regressar ao nosso mundo de criança, onde os problemas não existem e a inocência é gigante. O vosso apoio e amizade são inestimáveis.

Quero dedicar um agradecimento muito especial à professora Cristina e à Sara, que me orientaram e motivaram de forma excecional ao longo da realização desta dissertação. Nos momentos de desânimo, a vossa orientação foi fundamental para a conclusão deste projeto.

Por último, quero dedicar todo o meu esforço e esta conquista a um ser que, apesar de não ter compreendido o quão especial foi para mim, deixou-me uma marca muito profunda. Infelizmente, partiu nesta jornada, mas as lições que me ensinou, mesmo sem proferir uma única palavra, serão inesquecíveis. Obrigado, Cody.

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, Braga, october 2023

Guilherme da Silva Amorim Martins

Abstract

ErgoAware aims at developing technology that reduce the worker's exposure to ergonomic risk (e.g. poor postures) as there is a high incidence of Work-Related Musculoskeletal Disorders (WRMSDs), which represents an economic burden of 240 billion euros and is driven by the Industry 4.0 paradigm. Another goal of ErgoAware is also to optimize Human-robot Collaborative processes to shape the robot's assistance according to the individual physiological requirements of each user.

This dissertation main goal is the development of a desktop graphical interface application to allow visualization of the workers kinematic model, allow real-time monitoring of postural and fatigue metrics, and provide an abstraction layer for the **ROS** backend in order to allow the configuration of the integrated sensing technologies and develop control strategies. The objective will be for users to be able to manipulate ErgoAware's adjustable parameters in an intuitive manner, with a user-friendly interface for easy tool usability.

Keywords Ergonomics, Human-Robot Collaboration, User Interface, Posture control

Resumo

A ErgoAware visa desenvolver tecnologia que reduza a exposição do trabalhador ao risco ergonómico (por exemplo, posturas deficientes) já que existe uma elevada incidência de distúrbios músculo-esqueléticos relacionados com o trabalho (WRMSD) que representa um encargo económico de 240 mil milhões de euros e é impulsionada pelo paradigma da indústria 4.0. Outro objetivo da ErgoAware passa também por otimizar os processos de colaboração entre o robot humano e o robot para moldar a assistência do robot de acordo com os requisitos fisiológicos individuais de cada utilizador.

O principal objetivo desta dissertação assenta no desenvolvimento de uma interface gráfica para uma aplicação desktop para permitir a visualização do modelo cinemático dos trabalhadores, permitir um acompanhamento em tempo real da métrica postural e de fadiga e fornecer uma camada de abstracção para o backend **ROS**, a fim de permitir a configuração das tecnologias de deteção integradas e desenvolver estratégias de controlo. Esta será desenvolvida para que os utilizadores consigam manipular os parâmetros ajustáveis do ErgoAware de maneira intuitiva, com uma interface amigável para uma fácil usabilidade do desfrutador da ferramenta.

Palavras-chave Ergonomia, Colaboração Homem-Robot, Interface do utilizador, Controlo de Postura

Contents

List of Figures

List of Tables

Acronyms

API Application Programming Interface. 1, 8, 9, 14

HRC Human-Robot Collaboration. 1, 2, 3, 4, 8, 9, 15, 19, 21, 22, 23, 24, 30, 33, 41, 43, 46, 48, 58

ROS Robotic Operating System. 1, iv, v, 1, 3, 23, 24, 26, 27, 28, 29, 38, 41, 42, 43, 44, 45, 46, 58

SUS System Usability Scale. 1, vi, 10

TRL Technology Readiness Level. 1, 2

UCD User-centered Design. 1, viii, 5, 6, 41

UI User Interface. 1, 9, 30, 33, 37, 38, 41, 42, 43

UX User Experience. 1, 8

Chapter 1

Introduction

The advancement of robotics technology has led to an increased demand for Human-Robot Collaboration (**HRC**) in various industrial settings. **HRC** allows for improved productivity, efficiency, and safety, but the successful implementation of these systems requires the development of user-friendly interfaces that can be easily understood and used by non-technical personnel. This manuscript presents the main considerations and challenges of this research proposal. It intends to develop user-friendly front-end for a **HRC** framework, with an intuitive and straightfoward usage, acessible for non-technical users. The proposed application aims to take the team's framework to a higher technology readiness level(**TRL**). The research methodology includes the analysis of current state-of-the-art in **HRC** frameworks, the requirements gathering and the user-centered design. The research findings will be used to inform the design of future **HRC** systems.

1.1 Motivation and Problem Statement

Motivated by the high incidence of Work-Related Musculoskeletal Disorders (WRMSD), which represents an economic burden of 240 billion Euros [1], and driven by the Industry 4.0 paradigm [2], ErgoAware aims to develop technology that reduces worker exposure to ergonomic risk (e.g. poor postures) and optimizes **HRC** processes to tailor robot assistance according to the individual physiological requirements of each user. The team's framework, developed using **ROS**, is composed of sensory systems (Ergowear [3] (developed by the team), MVN Xsens commercial, and Trigno Avanti from Delsys) whose data are processed to monitor the worker's posture and fatigue state to enable real-time ergonomic assessment. This information is of utmost importance for the production line manager, as well as for ergonomic teams planning workspaces and task set-ups. In addition, sensory information acquired by the team's hardware is used to feed human-centered control strategies that customize the task to each worker, to increase the efficiency and safety of manufacturing processes.

The development of intuitive and user-friendly graphical interfaces for **HRC** systems is of paramount importance. These interfaces serve as the primary means of communication between the human operator and the robot, and their effectiveness plays a crucial role in determining the overall performance of the system. A well-designed interface allows for easy and efficient interaction, reducing cognitive load on the operator and minimizing the potencial for errors [4]. Furthermore, a user-friendly interface can also increase the overall acceptance of the technology and facilitate its integration into various industries. As such, the design and evaluation of these interfaces should be given due consideration throughout the development process of **HRC** products.

The primary motivation for this research is to address this problem by developing a user-friendly front-end application for a **HRC** framework. Is required to create a user-centered design approach that addresses the needs of non-technical personnel, and to demonstrate the potential of **HRC** to improve productivity, efficiency, and safety in the manufacturing industry.

1.2 Goals and Objectives

The main goal of this research is to develop a front-end application for a **HRC** framework that prioritizes a user-centered design approach, making it easy to understand and use by non-technical personnel. The objectives of this research are:

1. To conduct a comprehensive analysis of the current state-of-the-art in grahical interfaces for **HRC** frameworks and understand user-centered design methodology.

2. To gather requirements for the application from non-technical personnel, taking into consideration their needs and goals for **HRC**.

3. To design and implement a user-friendly front-end application to be integrated with a **HRC** framework.

By achieving these objectives, it is expected that this research will contribute to the advancement of the **HRC** framework's **TRL** and to the graphical user interfaces knowledge, making them more accessible and user-friendly for non-technical personnel and by providing evidence of the potential of this technology to improve industrial processes.

1.3 Thesis Outline

The dissertation report is divided into three chapters. The first chapter, Introduction, provides background information on **HRC**, presents the problem statement, the research goals and objectives, and an overview of the organization of the thesis. The second chapter, Literature Review, reviews the previous work in **HRC** frameworks and provides a critical overview of the current state-of-the-art. The third chapter, Ergoaware Framework introduces the Ergoaware Framework, emphasizing its integration with ROS, analyzing its architecture and revealing the biggest difficulties dealing with the framework and **ROS**. The development of the user-friendly frontend interface is the main topic of chapter 4, Developing the Ergoaware Frontend, which also highlights how its design has evolved and how it integrates with ROS to enable productive human-robot collaboration. The fifth chapter, Results and Discussion, was used to check that the final result of the application worked. The final chapter, Conclusion, gives a thorough summary, highlighting the value of user-centered design and the ways in which technology can improve ergonomics and industrial productivity.

Chapter 2

Literature Review

Designing interfaces and controls that facilitate effective communication and cooperation between humans and robots is a key challenge in the field of **HRC**. Researchers have developed a variety of approaches to this problem, and there is no one "best" solution that fits all scenarios. Instead, the design of interfaces and controls for **HRC** depends on the specific context in which the system will be used, as well as the abilities and limitations of both the human and robot collaborators.

European Union has a strong commitment to the development of **HRC** systems, and recognizes the importance of user-friendly interfaces in making these systems scalable and marketable, so that they can be adopted in a wide range of industries. The EU has several ongoing initiatives to support the development and integration of **HRC** systems, including the Horizon 2020 program [5], which provides funding for research and innovation in this area.

The EU is aware of the importance of user-centered design and user-friendly interfaces in making these systems more accessible and appealing to non-technical users [6]. The EU is actively promoting the development of user-friendly interfaces for **HRC** systems, and is supporting research and innovation that aims to make these systems more user-friendly and easy to use. The EU also encourages the development of training and education programs to help non-technical users understand and use these systems effectively.

The design of interfaces and controls for **HRC** should be focused on creating a system that is easy to use, flexible, and safe for the human collaborator, while also allowing for effective communication and cooperation with the robot.

2.1 Design methodologies

2.1.1 User-centered design

User-centered design (**UCD**) is a design methodology that emphasizes the centrality of the end user in the design process. The methodology is grounded in the understanding that the design of a product or system should be informed by a thorough understanding of the user's needs, goals, and workflows. The philosophy of **UCD** is to design systems that are usable, efficient and effective for the intended users.

It was originated in the field of human-computer interaction (HCI) in the early 1980s, with pioneers such as Donald Norman and Jakob Nielsen, who proposed the user-centered design approach as an alternative to the traditional technology-centered design. It is an iterative process that typically includes several phases, such as user research, requirements gathering, prototyping, and evaluation. In his book [8], Donald Norman proposed the key principles of this type of design, which are:

1. **Visibility:** the design should make it clear what actions are possible and what the current state of the system is. This can be achieved through clear labeling, highlighting important information, and providing visual cues that guide the user.

2. **Feedback:** the system should provide immediate and informative feedback to the user to indicate the results of an action. This can include visual, auditory, or haptic feedback, depending on the context.

3. **Constraints:** the design should limit the user's options in a way that guides them towards the correct action. This can be achieved through physical constraints, such as a button that can only be pressed in one direction, or logical constraints, such as a menu that only displays relevant options.

4. **Mapping:** the relationship between controls and their effects should be clear and consistent. This can be achieved through clear labeling, using conventions that are familiar to the user, and providing a visual representation of the relationship between controls and effects.

5. **Affordances:** the design should make it clear what actions are possible by providing physical cues, such as a handle that can be pulled or a button that can be pressed.

6. **Signifiers:** the design should make it clear what actions are possible by providing visual cues, such as an icon that indicates what a button does.

7. **Conceptual model:** the design should provide a clear and consistent mental model of the system to the user, which includes the system's goals, functions, and constraints. This can be achieved by providing clear and consistent language, labeling, and visual cues.

UCD is an empirically-grounded methodology that aims to create systems that are tailored to the needs of the users, by involving them in the design process and testing the system with them. This approach is based on the understanding that the usability, user experience and overall satisfaction of the end users are key factors for the success of a product or system.

Figure 1 shows the key steps to make a process that supports user-centred design and then each step of that process is explained.

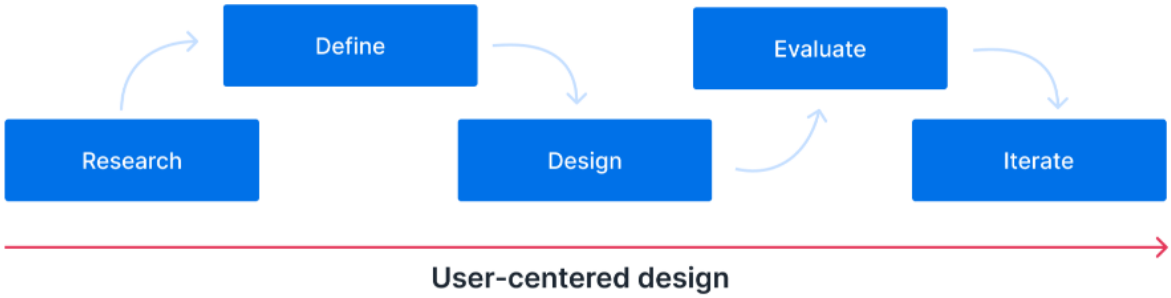


Figure 1: **UCD** [7]

Research

This step focus on developing a deep understanding of user's needs and requirements. If done correctly, the insights derived from this research will play a big role in developing a compelling user-centric product.

Clear identification and definition of user personas can be a huge help at the beginning of the design process. It gives an example of the archetypal user and allows to effectively group users with similar patterns – their behavior, backgrounds, needs, and goals. A product can target multiple personas, but it is crucial to identify the intended user demographic and determine which personas' needs and preferences should be prioritized in the design of specific flows and features within the product.

The design process must be started with the user, not the product. By focusing on the user, it is possible to understand which features to prioritize, how to market to them, and even what visual design elements to incorporate.

Research is not only a crucial stage in the **UCD** process, but it also is one of the most challenging.

Collecting the right data, keeping it organized, mapping it onto user journeys and deriving concrete actions from it isn't easy. It is important to turn user research data into meaningful action.

Define and align requirements

It is important to design a technically feasible and financially viable product. That's where setting the right goals and requirements comes in. It is necessary to define a clear scope for the designs and then ensure they align well with the user needs.

Design solutions

In this phase, it is important to be ready to start prototyping the designs, after collecting all the information that is needed.

One can start with a good wireframe and work from there. Even though it is not at the official testing stage yet, is required to validate the decisions at each step. A lot of useful data has already been collected to inform this process at this time, and regular feedback will help ensure alignment with the desired trajectory and prevent deviation from the established course..

At the end of the design, one can analyze it by asking questions like "is it accessible?" or "is it easy to understand?" to ensure that users can quickly locate the information they need and language is short and immediate comprehension.

Evaluate with feedback

This phase consists of approaching the end-user, asking him to test the prototype and getting his feedback. this can be done by asking question such as "Does this design solve the users' primary issues?" or "What can be done to improve this design?".

Design methods from the user research phase can be repeated – focus groups, usability testing, etc. – to gather further understanding on the degree of alignment with the intended outcome.

Iterate

Iteration is crucial as it allows for the refinement of the product, ultimately leading to a design that effectively addresses the needs and preferences of its intended users. The design process should begin with gathering feedback, followed by the creation of an improved solution, and continued iteration to further enhance the design.

2.1.2 User Experience (UX)

User experience (**UX**) is a multidimensional construct that encompasses the cognitive, affective, and physiological responses of an individual to the use of a product, system, or service [9]. The cognitive dimension of **UX** pertains to the ease of use, learnability, and efficiency of the product, system or service, which is commonly referred to as usability. The affective dimension of **UX** pertains to the emotional response of the individual to the product, system, or service, often referred to as desirability. The physiological dimension of **UX** pertains to the accessibility of the product, system, or service to individuals with diverse abilities, commonly referred to as accessibility.

UX design is an iterative process of understanding user's needs and goals, researching, testing, and iterating on design solutions to create products, systems or services that are easy to use, accessible and desirable. It encompasses a wide range of research methods and design techniques, such as user research, usability testing, information architecture, and interaction design. The ultimate aim of **UX** design is to enhance the overall quality of the user experience by creating products, systems or services that are intuitive, efficient, and satisfying to use [10].

2.1.3 Design of the user interface

The **HRC** frameworks are designed to allow humans and robots to work together effectively, and the user interface is an important aspect of facilitating communication and cooperation between the two.

There are several key considerations for ergonomics in the design of user interfaces for **HRC** frameworks. First of all, **Usability**, the user interface should be easy for humans to use and understand, with clear and intuitive controls and visualizations. **Customizability**,the user interface should be customizable to fit the needs and preferences of individual users, with adjustable settings and features to suit different users and tasks. And finally, **Compatibility**, the user interface should be compatible with the hardware and software components of the **HRC** system, including any relevant **API**s(Application Programming Interface) or data formats [11].

In short, the design of the user interface for a **HRC** framework is critical to the success of the system, and ergonomics plays a key role in ensuring that the interface is easy to use, comfortable and compatible with the needs of the users.

2.1.4 Frontend development frameworks

Front-end development frameworks are software libraries or platforms that provide pre-written code and tools for building the user interface (**UI**) of a web or mobile application. These frameworks are designed to make it easier and faster for developers to create interactive and visually appealing **UI**s, and they are often used in conjunction with back-end development frameworks or platforms to create full-stack web or mobile applications.

There are many different front-end development frameworks available, each with its own unique features and capabilities. Some of the most popular front-end development frameworks include React, Angular, Vue.js, Bootstrap, Foundation, among others.

When evaluating the suitability of different front-end development frameworks for building a front-end application for a **HRC** framework [12], there are several factors that one should consider:

- Ease of use: The front-end development framework should be easy for developers to learn and use, with clear and concise documentation and a intuitive and user-friendly **API**.
- Performance: The front-end development framework should be able to handle the performance requirements of the application, including fast rendering times and efficient data handling.
- Customizability: The front-end development framework should allow developers to customize the **UI** and features of the application to meet the specific needs of the **HRC** system.
- Compatibility with back-end frameworks: The front-end development framework should be compatible with the back-end frameworks or platforms that will be used to build the **HRC** system, including any relevant **API**s or data formats.
- Community support: The front-end development framework should have a strong and active community of developers and users, with a wealth of resources and support available for troubleshooting and problem-solving.

The best front-end development framework for building a front-end application for a **HRC** framework will depend on the specific requirements and goals of the project, as well as the skills and preferences of the development team.

The popular integrated development environment (IDE) Visual Studio Code (VS Code) was used to develop the front-end application for the Ergoaware framework. A stable and adaptable platform for writing, debugging, and managing the application's codebase was offered by VS Code.

2.1.5 System Usability Scale (SUS)

The System Usability Scale (**SUS**) is a widely used, validated and reliable measure of usability, which was developed by John Brooke in 1986 as part of his PhD thesis at City University London [13]. The SUS is a self-report questionnaire that consists of 10 items that assess the perceived usability of a product or system. Each item is rated on a 5-point Likert scale, where 1 is "strongly disagree" and 5 is "strongly agree".

It is designed to provide an overall score of usability that can be used to compare different designs, track changes in usability over time, or benchmark against other systems. The **SUS** has been widely used in many different domains, such as software, mobile apps, websites, and medical devices and it is been found to have good test-retest reliability and construct validity [14].

Moreover, the **SUS** psychometric properties have been investigated, showing good reliability and validity. It has been found to be a robust and reliable measure of usability. Its score is easy to interpret and it is widely used in academic and industry research due to its simplicity and easiness of use for measuring usability.

2.2 Case studies

2.2.1 Scalefit

Scalefit [15] is a company that provides a software based on Xsens MVN Awinda sensors for ergonomic risk analysis. The software has 21 load parameters and an avatar display. The company's products are designed to evaluate ergonomics with the aim to improve comfort, reduce fatigue, and promote good posture while working, with a focus on ergonomic design and adjustability, by identifying hidden health risks.

They work with several companies that want to reduce musculoskeletal disorders and physical stress (with real-time stress analysis) by using their framework to assess the ergonomic risk associated with each occupation. Several factors are evaluated, as shown in Figure 2 and 4, such as head tilt, trunk tilt, knee flexion, among others. Figure 3 illustrates the ergonomic risk associated with various tasks, such as weight training or lifting weights from the floor, to each vulnerable body part like neck, shoulders, lower back, and hands. It presents the ergonomic risk level per segment in an intuitive and user-friendly way, using colors to express itself, such as green for good posture and red for bad posture.

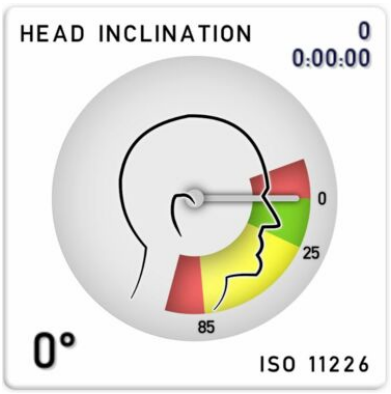


Figure 2: Scalefit evaluated head inclination

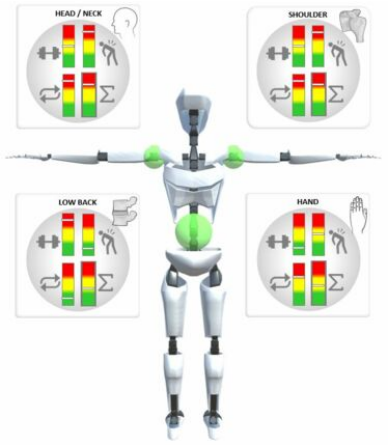


Figure 3: Scalefit evaluated body parts

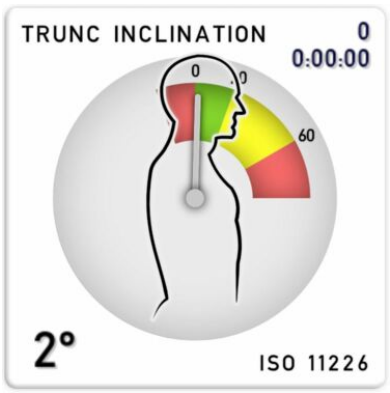


Figure 4: Scalefit evaluated trunc inclination

Figure 5 shows a new feature from *Scalefit*, "Industrial Athlete Grasp Space Visualization", that visualizes the grasp space separately for each hand and is therefore an important concept for ergonomic workplacedesign - either in the planning phase or in real time analysis at existing workplaces and it is demonstrated in the figure how the framework operates. The inclination of the body in degrees is displayed in real-time, indicating the areas that may pose a risk and the areas where excessive pressure on the lumbar disk may be exerted. This can all be analyzed later so that posture changes can be adopted and back strain greatly reduced.

Figure 5: Scalefit Graphical Interface

2.2.2 KUKA

KUKA [16] is a German manufacturer of industrial robots and solutions for factory automation. One of their products is a line of small, lightweight robots called "universal robots", which are designed to work in collaboration with humans.

The KUKA universal robots are known for their ease of use and flexibility, they have a modular design that allows them to be easily integrated into various systems and applications. These robots are also designed to be safe to work with, with built-in safety features such as force-sensing and safety-rated motion control. They can be controlled using a wide range of programming languages and interfaces, and they come with a variety of tools and accessories that can be used for different applications.

The robots have a user-friendly interface, which allows easy programming, and it can be used in a variety of industrial environments. They are often used in manufacturing, assembly, packaging, and material handling tasks, however due to their flexibility, the robots can be used for different application fields.

The KUKA application interface for the robots is typically a touch-screen based interface which allows for easy programming and control of the robot. The interface usually allows the user to program the robot using a graphical programming language, such as a flowchart, a drag-and-drop interface, or a block-based programming language. It also allows the user to control the robot's movement, adjust its speed and acceleration, and monitor its status. Additionally, it allows the user to perform tasks such as creating and editing programs, monitoring the robot's status and performance, and troubleshooting issues. In Figure 6, an illustration of KUKA's interface is presented. However, some users have reported that the interface can be somewhat difficult to navigate, particularly for those who are not familiar with industrial robots. Moreover, the interface may not be as customizable as some users would prefer, which could limit its usefulness in certain applications. Some users have also reported that the interface may not be as responsive as they would like, which could lead to delays in the execution of certain tasks. Overall, the interface of KUKA universal robots is generally considered to be effective and user-friendly, but there may be room for improvement in terms of customizability and responsiveness.

Figure 6: KUKA

2.2.3 Vivelab Ergo

Vivelab Ergo [17] is a company that designs and manufactures ergonomic products and solutions for the office and other work environments.

ViveLab Ergo software helps to create the right working conditions and processes for the people who work in occupational safety and engineering fields. It replaces paperbased methods performing ergonomic analyzes, creating an accurate measurement option that assigns a risk value to the tests.

Figure 7 shows the interface of this framework. It provides the ergonomic risk level of each segment. However, contrary to the Scalefit interface, which associates the risk level with a color code, i.e., green for low risk and red for high risk, the ViveLab Ergo provides a quantitative risk score. This score is given both per segment and as a global score.

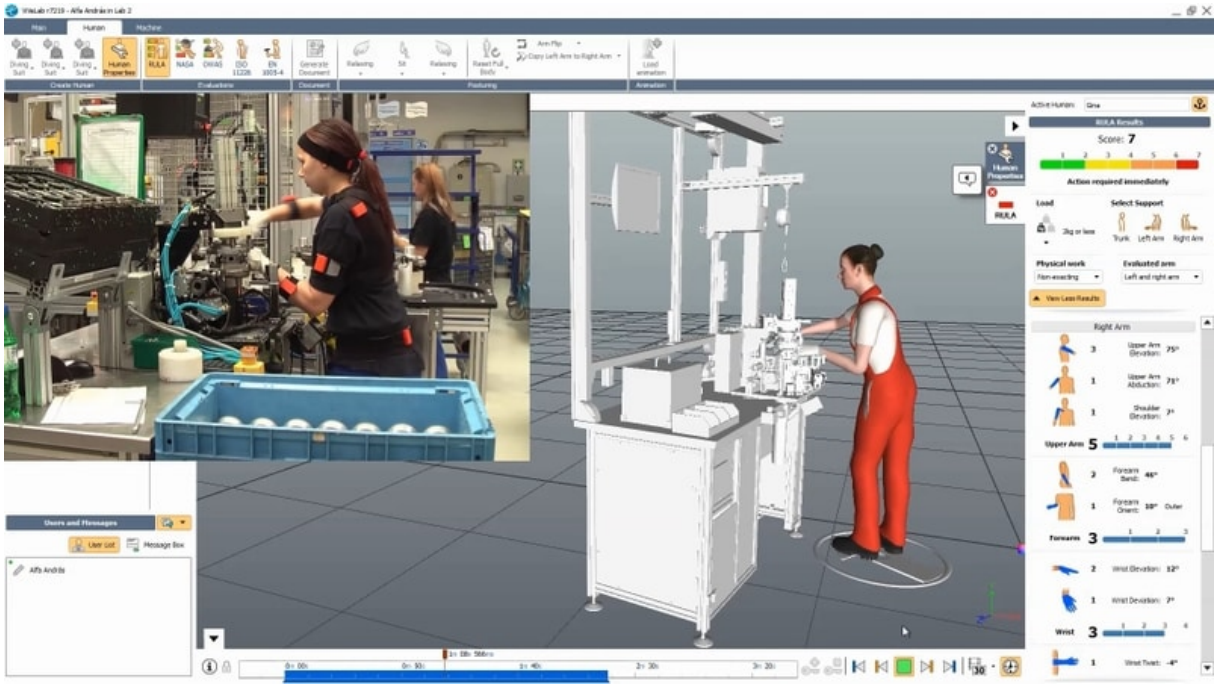


Figure 7: ViveLab Ergo

2.2.4 RoboEarth

RoboEarth is a cloud-based platform for robots, developed by the Robotics and Perception Group at the University of Zurich. It aims to provide robots with the ability to share and reuse knowledge, in order to improve their autonomy and functionality. The platform is designed to serve as a common repository for robot-related information, such as sensor data, maps, and task plans.

RoboEarth's architecture consists of a number of interconnected components, including a database, a web-based interface, and a set of **API**s. The database stores information about the robots and their environment, such as sensor data, maps, and task plans. The web-based interface provides a user-friendly way to access and manage the data stored in the database, while the **API**s allow other systems and applications to interact with the platform.

The interface provided by RoboEarth is a web-based one, it allows users to upload, download and share information, such as maps, plans and sensor data, and it also allows users to access and execute remote procedures. However, it has been criticized for not providing a simple and intuitive way to interact with the platform, and for requiring a certain level of technical expertise in order to use the platform effectively.

The RoboEarth platform offers three different types of interfaces [19] for interacting with its database. The first is a web interface that allows humans to interact with the database using HTML forms. This interface can be used to access the RoboEarth database and provide labels for the perceptions of a robot.

The second interface is based on the Representational State Transfer (REST) architecture [20]. It is a stateless system that transfers all necessary information as part of the request, without the need to store any information on the server. This allows for high scalability and fast response times due to server-side optimizations such as load balancing and failover capabilities. The REST-style interface encodes data in JavaScript Object Notation (JSON) making it suitable for interaction between robots and the database [21].

The third interface is based on the publish/subscribe message exchange pattern, which acts on changes in the database. This type of interface would allow a robot to subscribe to a notification for a specific event, avoiding unnecessary traffic caused by periodic requests for updates. This is particularly useful for a robot to be notified when a patient leaves a hospital room, for example, to carry out cleaning operations without constantly querying the database for the patient's location.

2.2.5 ABB's Collaborative Robots (cobots)

ABB's Collaborative Robots (cobots) are a class of robots designed for safe **HRC** in various industries such as assembly, picking and packaging, and laboratory work. The cobots are equipped with advanced safety features such as force sensing and speed and separation monitoring, allowing them to operate in close proximity to humans without the need for physical barriers or safety cages.

On their webpage [22] it is possible to build an application using the given parameters and simulate it. As observed in Figures 8-12, the type of selection can be made by:

- Application: Machine Tending, Screw Driving, Part Handling and Assembly
- Mounting: on the ceiling, wall or table
- Gripper: Vaccum or Servo, or for Screw Driving, Screw Driver
- Infeed: fixed position, free position with vision and tray pattern, or for Screw Driving, Screw Feeder
- Outfeed: fixed position and tray pattern

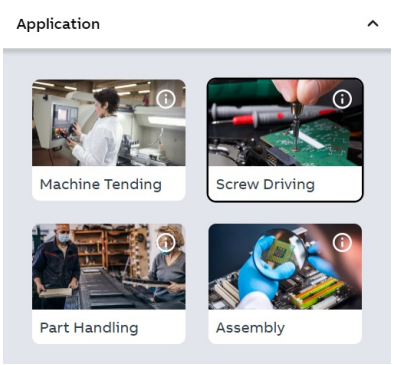


Figure 8: Application

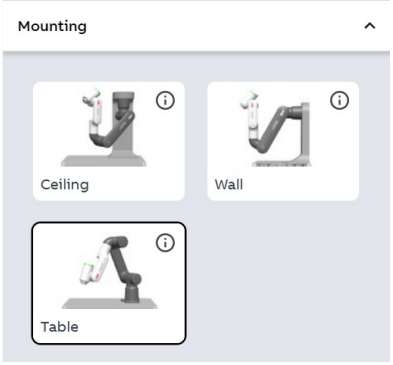


Figure 9: Mounting

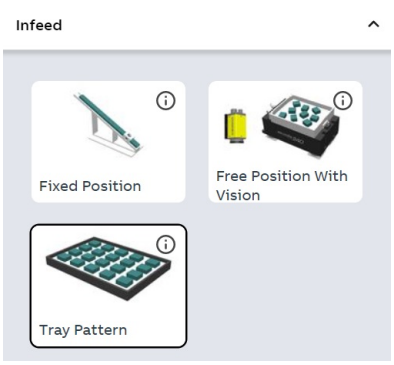


Figure 10: Infeed

Figure 11: Gripper

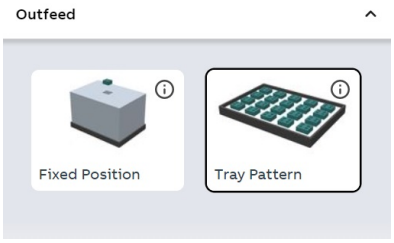


Figure 12: Outfeed

Figure 13 shows exactly how the interface works with all the features mentioned and they also have it available for iOS, as depicted in Figure 14. Figure 14 also show their bot feature, available to answer the user's questions, useful to clarify doubts about the interface in a fast and effective way. After selecting the intended options for the application, one can simulate it, as can be observed in Figure 15.

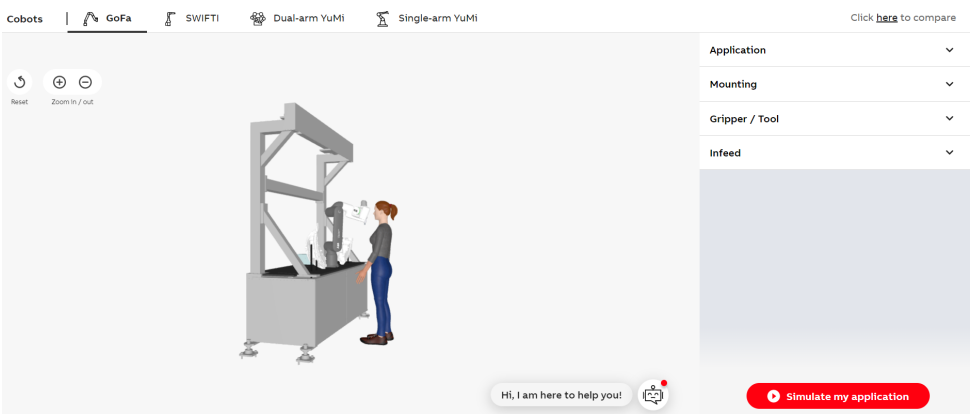


Figure 13: ABB's Collaborative Robot application builder

Figure 14: iOS App

Figure 15: Robot's simulation

In terms of the user interface, ABB's cobots are designed to be intuitive and user-friendly. The robots are programmed through a web-based interface that allows for easy drag-and-drop programming and control of the robot. The interface also features a built-in simulation environment, allowing users to test and fine-tune the robot's movements before executing them in the real world.

One of the main advantages of ABB's cobot is its ability to integrate with a variety of sensors and other devices, making it highly versatile and adaptable to different types of work environments.

However, it is important to note that despite their user-friendly interface, the implementation of cobots in a production line or facility can still be a complex task, involving multiple stakeholders and processes. It is necessary a good understanding of the task or process to be automated, the specific requirements, and the available resources. In addition, the implementation of cobots also requires a high level of technical expertise and specialized skills to program, install, and maintain the robots.

2.2.6 Rethink Robotics' Baxter and Sawyer

Rethink Robotics' Baxter and Sawyer [23] are examples of collaborative robots, or cobots, that have been developed with the goal of working alongside human operators in a variety of industrial and commercial settings. They are designed to be easy to program and use, with intuitive interfaces that allow for quick deployment and minimal training requirements.

Baxter and Sawyer are both equipped with a wide range of sensors and actuators that allow them to perceive and interact with their environment in a safe and effective manner. They also feature advanced control systems that enable them to adapt to changing conditions, adjust their behavior in response to operator inputs, and respond to unexpected events.

In terms of their interfaces, both cobots feature a touch screen interface which allows operators to control the robot's movement and behavior, as well as program and customize tasks. The interface is intuitive and simple, and it is designed to be easy to learn and use. The platform allows users to teach the robot new tasks, even without prior coding or programming experience. The train-by-demonstration feature is provided by the software, allowing for ease of interaction with the robot's arm and training of new tasks as demonstrated in Figure 16. Figure 17 illustrates the Intera Studio, a powerful interface, which can be accessed through a laptop. This interface enables users to access and adjust tasks for the robot with ease. Intera Studio allows for efficient creation, modification, and monitoring of tasks, as well as advanced controls for the robot. Figure 18 illustrates real-time data analysis of production metrics, such as cycle times and part counts, through the robot interface, this allows for early identification of production issues. The same customizable data is also available in Intera Studio, giving visibility to other team members. This feature allows for a more efficient production process by providing critical performance indicators directly on the factory floor.

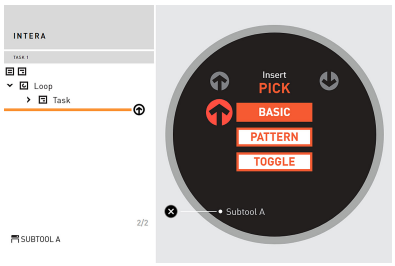


Figure 16: Task training

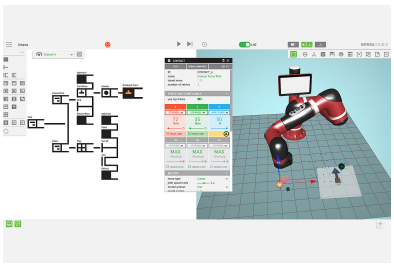


Figure 17: Intera Studio

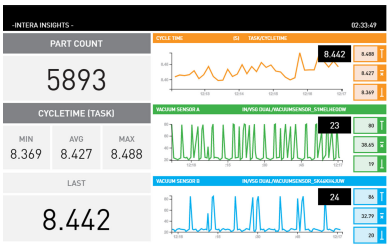


Figure 18: Intera Insights

However, it could be argued that the interface could be improved in some ways, such as the use of more advanced visualization tools to help operators better understand the robot's behavior, or the incorporation of more advanced user-centered design principles to make the interface more ergonomic and user-friendly.

2.2.7 FANUC

FANUC [24] is a company that designs and manufactures industrial robots and cobots (Collaborative Robots) for a wide range of industrial applications. Their cobots are designed to work safely and collaboratively alongside human workers, in order to increase efficiency and productivity in a wide range of

tasks.

FANUC cobots feature a range of advanced technologies that support **HRC**, such as force sensing, vision systems, and intuitive user interfaces. The interface is designed to be easy to use, like Figure 19 shows, with a simple programming interface that allows for quick and efficient setup and operation of the cobot.

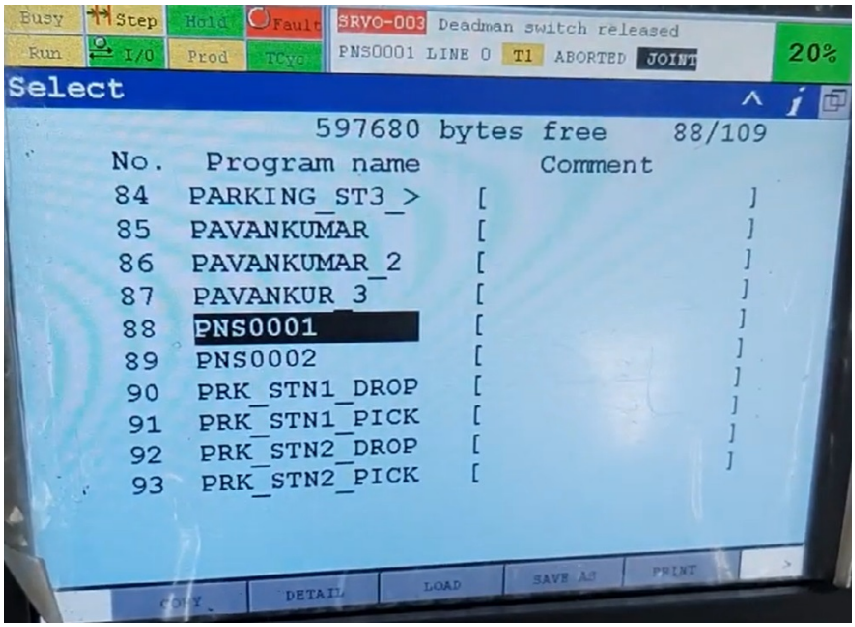


Figure 19: FANUC interface

However, one of the critical aspect that can be reconsidered is that their interface and software are not as flexible and customizable as some of the other cobot manufacturers and is too outdated, adopting an old system that may confuse younger generations. This can limit the ability of users to adapt the robot to specific tasks or workflows.

2.2.8 Yaskawa Motoman

Yaskawa Motoman [25] is a leading manufacturer of industrial robots, including collaborative robots (cobots). Their cobots are designed to work alongside human workers in a variety of industries, including manufacturing, packaging, and assembly. Their cobots feature advanced safety features, such as force-sensing and speed and separation monitoring, which allow them to operate safely in close proximity to humans. They also have user-friendly interfaces that make it easy for operators to program and operate the robots. The interface is usually a teach pendant, a handheld device that allows the operator to control the robot and program its movements. Figure 20 shows how Yaskawa's interface works, and it can be

observed that a significant number of features have been incorporated in a simple and intituive way, making it easy for operators to train the robot and perform new tasks.

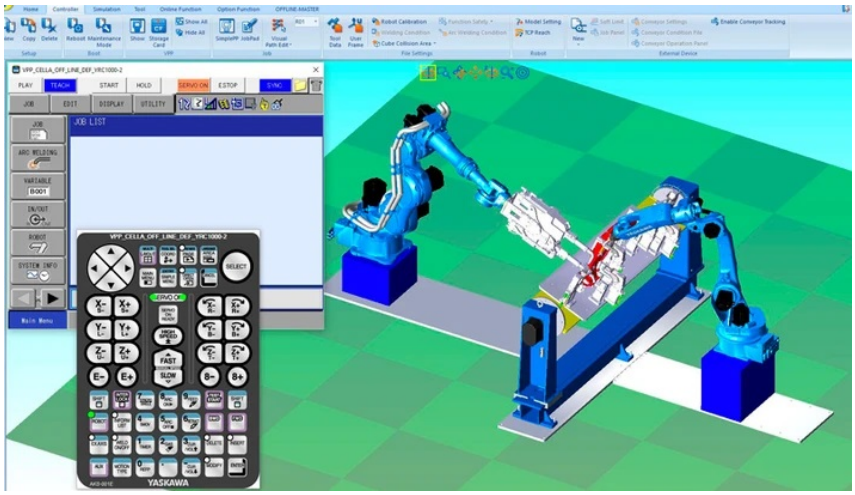


Figure 20: Yaskawa interface

However, some users may find the interface to be less flexible than other cobot options, as it may require more expertise to program and customize the robot's movements. Additionally, Yaskawa Motoman cobots are generally more expensive than other cobot options, making them less accessible to small and medium-sized businesses.

2.2.9 Kawasaki Robotics

Kawasaki Robotics [26] is a subsidiary of Kawasaki Heavy Industries, which is a leading manufacturer of industrial robots. They offer a wide range of robots for various applications such as welding, assembly, material handling, and palletizing. They have a number of collaborative robots in their portfolio that are designed to work alongside humans in a safe and efficient manner. These cobots are known for their high precision and speed, as well as their ability to be integrated into existing manufacturing systems.

In terms of their interface, Kawasaki Robotics utilizes a user-friendly, intuitive interface that allows for easy programming and operation of their robots. The interface is designed to be simple and easy to understand for users with little to no programming experience. The company also offers support and training to assist with the integration and operation of their robots. As shown in Figure 21, they have an interface with a nice design, organized and simple.

Figure 21: Kawasaki interface

Overall, Kawasaki Robotics is known for their high-quality, reliable industrial robots and their user-friendly interfaces. Their cobots are designed to increase productivity and efficiency in various industrial applications, while ensuring safety and ease of use.

2.3 Critical overview

In this literature review, several **HRC** frameworks were examined and the majority of the reviewed frameworks are designed to improve the overall performance and safety of **HRC** by allowing robots to adapt their behavior based on the actions and intentions of the human partner. Most of the reviewed frameworks aim to allow easy programming of the collaborative robots.

A common theme among these frameworks is the emphasis on adaptability and flexibility. Many of the frameworks are designed to be easily integrated into various systems and applications, and they often use graphical programming languages that make them easy to use and understand.

However, one limitation of these frameworks is that they are mostly developed and tested in laboratory settings, and there is a need for more research on the practical implementation and scalability of these frameworks.

Table 1 compares the features of the previously discussed solutions, which are important when evaluating **HRC** products. The main goal of this comparison is to identify key takeaways for developing a similar solution and to identify areas for improvement that may have been overlooked by these solutions. The table is intended to help distinguish the main differences and similarities between the solutions and

to identify possible areas for further development.

Solution \ Features	Platform(s)	Web Plataform/Remote Data Logging	Focus	Purpose	Sensors/Actuators
Scalefit [15]	Website	✓	Ergonomics	Monitoring (Human posture)	Accessories (chairs, monitor arms...)
KUKA [16]	Windows / Mobile devices (iOS and Android) [18]	✓	Industrial Automation	Monitoring (Industrial robots)	Cameras / Force-torque sensors / Grippers
Vivelab Ergo [17]	Windows	✓	Ergonomics	Monitoring (Human posture)	Accessories (chairs, monitor arms...)
RoboEarth [19]	Web browser	✓	Knowledge sharing	Monitoring (Robot autonomy)	Robot's sensors (cameras, laser rangefinders...) and robotic arms
ABB [22]	Website and Mobile devices	✓	Collaboration	Monitoring (Robot autonomy)	Cameras and force-torque sensors
Rethink Robotics [23]	Windows and Linux	✓	Collaboration	Monitoring (Industrisl robots)	Cameras / Force-torque sensors / Grippers
FANUC [24]	Tablet	✓	Automation	Monitoring (Industrial robots)	Cameras / Force-torque sensors / Grippers
Yaskawa Motoman [25]	Website	✓	Automation	Monitoring (Industrial robots)	Sensors for position, force, and tactile sensing
Kawasaki Robotics [26]	Website	✓	Industrial automation	Monitoring (Industrial robots)	Accessories (chairs, monitor arms...)

Table 1: Systems Overview

It is possible to verify that each framework has its own strengths and weaknesses some are more flexible and easy to use, others are more precise and reliable and not all frameworks have applications. However, those that do have applications do not always have a user-friendly interface. Further, intuitiviness can be improved with features such as color feedback and visual feedback of the task being performed by the user, as demonstrated by Scalefit and ViveLab Ergo. Also, it is important to minimize the learning curve of the application usage. Clean interfaces, that required minimal cognitive effort can help to accomplish this. For example, Scalefit has only the essential information being displayed. ABB's, in turn, offers a configuration interface that groups the main features in categories, allowing a easy and sequential configuration. As for interesting features to be presented to the user during the task performance, Rethink robotics presents, in real time, the task's cycle time metrics (average, maximum, and minimim time) and parts produced. Scale fit and VivoLab present the ergonomic risk of the posture being sustained by the user, at the current time. All applications allow data storage. This is also a interesting feature as this data can be processed and generate a automatic report with the task statistics. On the other hand, FANUC application presented a non-applealing graphical interface that impairs its usability and market interest.

In conclusion, the literature review has revealed that **HRC** frameworks are an active research area, but there are still gaps and limitations in the existing literature. The proposed research aims to address these gaps by developing a front-end application for a **HRC** framework that is tailored to the needs of a particular application domain, easy to use, and has been tested in a real-world environment.

Chapter 3

Ergoaware Framework

A noteworthy development in the area of Human-Robot Collaboration (**HRC**) is the Ergoaware Framework. As robotics technology becomes more and more integrated into industrial settings, solutions that prioritize worker well-being in addition to productivity and efficiency become increasingly important. The Ergoaware Framework is a system created to improve ergonomics, safety, and task efficiency inside collaborative contexts. It sits at the intersection of these goals.

The design of Ergoaware makes use of **ROS**(Robot Operating System), a well-known and reliable middleware for creating robotic systems due to its versatility. It is not limtied to a single programming language. Futhermore, by facilitating the smooth integration of different parts and modules, **ROS** makes it possible for Ergoaware to effectively receive information from a variety of sensory systems and implement advanced control strategies to allow intelligent and sensible interaction between Humans and cobots. Ergoaware is committed to a modular and extensible approach in the field of **HRC**, as evidenced by its reliance on **ROS**.

Ergoaware is primarily dependent on a group of sensory systems, which include the team's self-developed Ergowear, a upperbody smart garment for posture monitorization, and the the commercial sensory systems owned by BiRDLAB: MVN Xsens and Trigno Avanti from Delsys. Together, these devices collect information and enable ergonomic assessments in real time by monitoring posture, fatigue level, and productivity of the workers. This information can assist production line managers and ergonomic teams in making decisions regarding workstation planning and job sets.

The power of Ergoaware resides in its capacity to create human-centered control techniques by utilizing sensory data. These strategies customize work tasks for each employer, increasing production processes' efficiency and safety. By adjusting in real-time to each user's physiological requirements, the framework minimizes ergonomic risks and optimizes productivity.

3.1 ROS: The Backbone of Ergoaware

The Robot Operating System (**ROS**) provides a solid foundation upon which the Ergoaware Framework is constructed. The robotics community has come to recognize the open-source middleware framework **ROS** for its scalability, flexibility, and broad support for robotic applications. The core of Ergoaware's architecture and the basis for its effective human-robot collaboration is its integration with **ROS**.

Since Willow Garage's original development of ROS in 2007, the software has grown to become a vital resource for roboticists and researchers everywhere. It offers an extensive collection of conventions, tools, and libraries that make developing sophisticated robot systems easier. By facilitating the modularity of robotic components, **ROS** makes it possible for sensors, actuators, and high-level control systems to communicate with each other seamlessly. [27]

Furthermore, **ROS** offers the framework required to manage human-robot interactions in Ergoaware. It makes it possible to implement control strategies that smoothly adjust robot behavior to each user's unique physiological needs. Real-time data exchange is made possible by ROS's messaging system and middleware functionalities, which help Ergoaware respond to changing conditions within optimal manufacturing processes and well-informed decisions.

ROS's dedication to extensibility and community-driven development is one of its main advantages. This provides Ergoaware with access to a multitude of **ROS** packages and libraries developed by researchers and developers across the globe.

This gives **HRC** a reliable and adaptable infrastructure. It is a very good option for coordinating human-robot interactions, optimizing ergonomic assessments, and seamlessly integrating sensory data because of its modular design, communication capabilities, and extensibility. Ergoaware's deep integration with **ROS** positions it as a creative and flexible response to the demands of contemporary manufacturing.[28]

It's crucial to remember that ROS2, ROS's replacement, has also become a viable option. Although the two versions have similar basic goals—for example, offering a robust and adaptable robotics platform—they diverge in important ways. With its enhanced security features, better support for non-Linux operating systems, and improved real-time capabilities, ROS2 is an excellent option for a wider variety of applications. The decision between **ROS** and ROS2, which is still in its early stages of development, is contingent upon particular project requirements and compatibility considerations. In this instance, **ROS** was chosen.

3.2 Ergoaware architecture and backend

One important aspect of Ergoaware is its sophisticated sensory systems. The research team's proprietary Ergowear technology and commercial products like MVN Xsens are examples of these sensory systems. These sensors, positioned on the worker's body, are used to monitor and record information about the posture, motions, and physiological condition of an employee. The data streamed in real-time by these sensors is processed to ergonomic evaluations that provide information about possible dangers associated with weariness and bad posture.

Human-centered control strategies are implemented through the backend of Ergoaware. It makes use of the sense data gathered to tailor and optimize robot work according to the needs of each individual worker. These control strategies adjust robot assistance to each user's unique physiological requirements in an effort to improve manufacturing processes' efficiency and safety. This customized strategy improves user experience and productivity overall while reducing ergonomic hazards.

The architectural base of Ergoaware is scalable and modular. Fundamentally, it makes use of the capabilities of the Robot Operating System (Fig.22), which acts as the principal system for coordination and communication between the different parts of the framework. There are intentions to switch to ROS2 in the future. The team's Ergowear, Trigno Avanti, and other commercial sensory systems, as well as a variety of other sensory systems, were all integrated into the framework. Except for the EMG software interface, which is awaiting validation, all of the software interfaces in charge of online data streaming between these systems and the processing station have been developed. At the moment, the framework supports raw data from the team's Ergowear, EMG raw data from Trigno Avanti, and online processed data from MTw Awinda. Specifically, the Ergowear Angle estimation block uses algorithms to transform the data that Ergowear collects into segment orientations that are comparable to the output of MTw Awinda. These segment orientations are converted into joint angles by means of a human subject URDF model, which is produced from a .mvn file obtained from the Xsens motion capture system. The ergonomic evaluation, which happens within the Ergonomic Assessment block, depends on the joint angles.

The data obtained from Ergowear, which operates at a 100Hz sampling frequency, is downsampled to 60Hz in order to guarantee compatibility with the 60Hz Xsens Awinda Analyze. Components for risk management and estimation work at the user's pace because they initiate fresh ergonomic assessments with every delivery to the subject. Beyond cobot control and actuation, the system can also be used to control vibrotactile motors incorporated into Ergowear textile for biofeedback and to control a simulation environment via virtual reality (VR) glasses that are interfaced with the Gazebo robotics simulator.

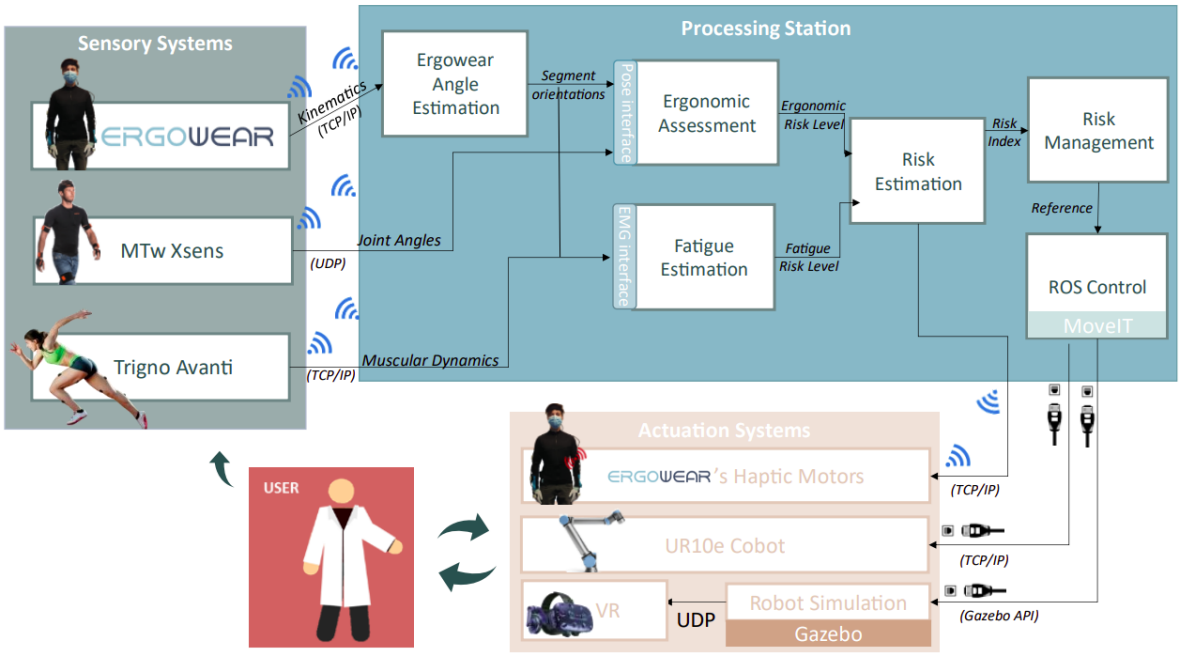


Figure 22: Diagram of the current HRC framework

The project uses Python programming for efficient integration and control, and it includes a wide variety of ROS packages that enable smooth communication across ROS topics, like ros-joint, ros-moveit, ros-controller, catkin, rviz, ros-desktop-full, rospy, simulators.

The Robot Operating System (**ROS**) uses publish-subscribe communication patterns to enable communication between various "nodes" or components, within a robotic system. The Publisher and Subscriber objects are used by the ROS nodes to facilitate this communication. Within the robotic system, a "node" in ROS refers to an autonomous program that carries out a particular function. Every node is in charge of carrying out different tasks, such as directing an actuator, processing data, or managing a sensor. A "Publisher" is a ROS object connected to a particular topic that permits data transmission from a node to that topic. A node creates a Publisher object and specifies the topic name and message type to be published when it wants to share data. After serializing the data into the message format, the Publisher adds it to a message queue reserved for that subject. The Publisher then forwards the message to the ROS Master, publishing it. A ROS object linked to a particular topic that permits a node to receive data from that topic is called a "Subscriber". A node creates a Subscriber object with the topic name and expected message type specified in order to access data on a topic. A callback function that the subscriber registers is called whenever fresh information on the subject is obtained. All registered Subscribers receive the message when it is identified by the ROS Master as a new message on the subject. The Subscriber calls its callback function to handle the incoming data when it receives a message.

Here's how the process works step by step:

1. **Node creation:** Numerous ROS nodes, each with a distinct function and purpose, are operating independently. These nodes could be dispersed throughout a network or operating on a single robot.

2. **Publisher setup:** A Publisher object is created by a node that has data to share. It gives the name of the topic and the kind of message it plans to publish. Data is serialized by the Publisher and added to the topic's message queue. This data is channeled through the topic name.

3. **Publication:** The Publisher sends the serialized message to the ROS Master. The ROS Master keeps a list of all topics and which nodes are publishing and subscribing to them. The ROS Master, acting as a centralized message broker, delivers the published message to all Subscribers registered to that topic.

4. **Subscriber setup:** When nodes wish to receive data from a specific topic, they create Subscriber objects with the topic name and the desired message type specified. A callback function that specifies what should happen when a new message on the subject arrives is also registered by each subscriber.

5. **Subscription:** All Subscribers who have registered for a topic receive new messages when they are identified by the ROS Master. As each Subscriber executes its callback function, the incoming message is processed as required.

ROS enables nodes to communicate in a modular and flexible way by using the publish-subscribe communication pattern with Publisher and Subscriber objects. Distributed control, scalability, and effective information sharing are made possible in robotic systems by this architecture.

3.2.1 Difficulties

The architecture and backend of the Ergoaware framework are fundamental components that underpin its capabilities and functionalities. However, its complexity and the use of ROS, a previously unfamiliar domain, presented several difficulties in understanding the architecture, such as:

- **Complexity of Middleware:** The middleware's inherent complexity posed a significant challenge during the learning process of the **ROS** architecture. As a middleware framework, **ROS** adds a level of abstraction that can be confusing at first. It took a thorough understanding of middleware

concepts to comprehend how nodes, topics, and messages communicate with one another within **ROS**.

- **Node Interactions:** **ROS** is largely dependent on the idea of nodes, which are autonomous computational entities that carry out particular functions. One of the biggest challenges was coordinating these nodes and understanding how they interacted with the rest of the system. Further layers of complexity were introduced by the complexities involved in launching nodes, setting up communication channels, and handling dependencies.

- **Message Exchanging:** In **ROS**, efficient message exchanging is essential for inter-node communication. Understanding message types, crafting unique messages when needed, and guaranteeing data synchronization and consistency among nodes were part of the difficulties faced.

- **Package Management:** Code and resources are organized using a package-based system by **ROS**. There was a steep learning curve involved in managing packages, dependencies, and version compatibility. It was not an easy task to make sure the right packages were installed, configured, and integrated into the framework.

- **Documentation and Resources:** Even though **ROS** has a ton of community-contributed resources and documentation, it was difficult to navigate such a large repository that **ROS** has. Finding pertinent guides, tutorials, and best practices for particular use cases frequently required careful consideration and investigation.

- **Debugging and Troubleshooting:** Because the framework is distributed, debugging ROS-based applications presented particular difficulties. Determining and fixing problems with node crashes, communication breakdowns, or message mismatches required extensive knowledge of diagnostic tools and **ROS** tools.

- **Learning Curve:** The vast ecosystem of ROS consists of a multitude of libraries, tools, and frameworks. The significant learning curve to become proficient with these elements presented a formidable obstacle. It was essential to be familiar with debugging tools like rqt_console, as well as visualization tools like RViz and command-line tools.

- **Version Compatibility:** One recurring issue was making sure that packages and **ROS** versions were compatible. Ergoaware framework modifications or updates may be required in response to incompatibility issues caused by modifications or updates in **ROS** distributions.

It took a committed and methodical approach to overcome these obstacles, combining in-depth research, practical experience, cooperation with the **ROS** community, and continual learning and adaptation. The development and comprehension of the Ergoaware framework were greatly enhanced by these experiences in navigating the intricacies of the **ROS** architecture, which also enhanced the research process and knowledge acquired.

Chapter 4

Developing the Ergoaware Frontend

In this chapter, the critical stage of developing the Ergoaware frontend—a crucial part of the Human-Robot Collaboration (**HRC**) framework—is explored. This section walks readers through the entire process of turning ideas into a concrete, intuitive user interface, covering important topics such as requirements, user interface (**UI**) design, and implementation. The ultimate goal is to clarify the steps and design considerations that result in a frontend application customized to non-technical users' unique requirements, allowing them to easily engage with the Ergoaware **HRC** framework.

4.1 Requirements for User-Friendly Interfaces

The Ergoaware frontend's design is centered on user-friendliness. Creating an interface that appeals to non-technical users requires a careful examination of their requirements and expectations. To create an Ergoaware frontend that is easy to use, a thorough process of defining, prioritizing, and identifying the necessary requirements must be undertaken.

The new user interface (**UI**) is designed to conform with the specified design requirements and is intended to be viewed on desktop screen sizes. In turn, these prerequisites will function as sub-objectives for the purposes of this thesis:

- **Parameter Configuration Page** It is necessary to create a specific page where users can input the necessary parameters to start the Ergoaware framework. The parameters are user selection, simulation setting (virtual, that is a UR simulator or physical robot), optimization mode, sensing technology, and control strategy. Gives users a logical and well-organized interface for defining important experiment parameters. makes sure the framework can be adjusted to different user needs and experimental setups.

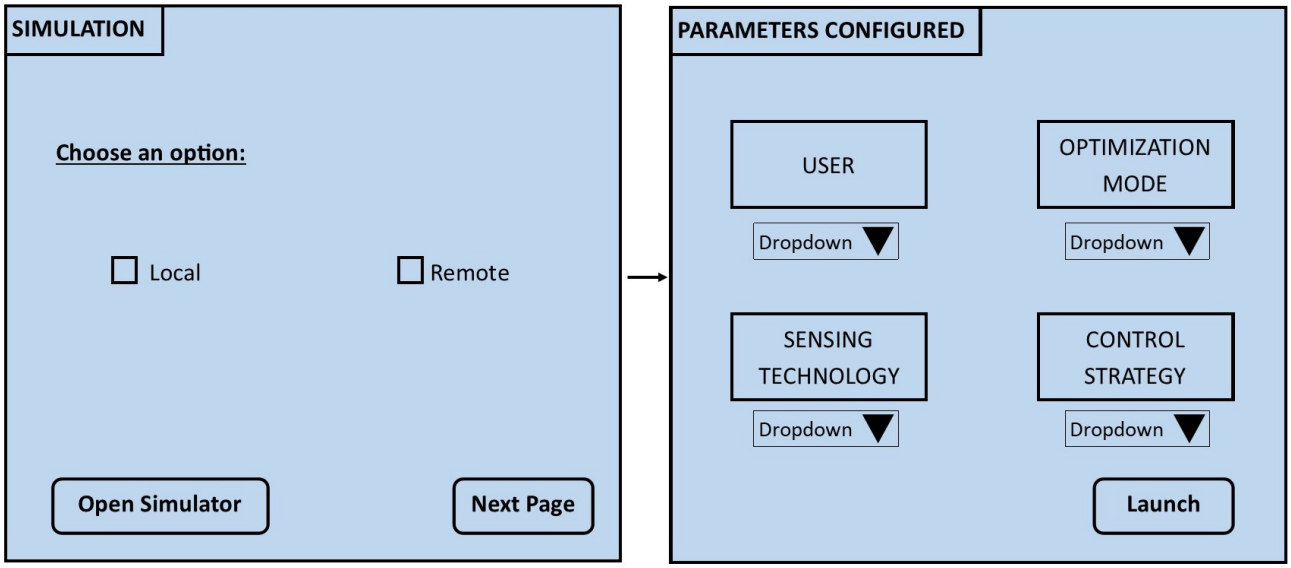


Figure 23: Idealized GUI Mockup

This part has initally idealized in two separate screens as we can see in Figure 23. The first screen gives the user the option of selecting a physical robot or a virtual simulation (UR simulator). This decision is important because it establishes whether the robot will work in a simulated environment or in the real world. Since they don't require actual hardware, simulations are frequently used for training and testing, enabling users to verify the behavior of the robot. To start the UR simulator after choosing the simulation, a dedicated button is provided. Whichever option they initially select, users can move on to the next screen, which displays an extensive list of programmable parameters for the robot's functionality.

The "User Selection" parameter is crucial as it enables users to designate the individuals who will engage with the robot. This parameter makes sure that the behavior and ergonomics of the robot are customized to the particular user using it. Different users may have different ergonomic needs or preferences.

The robot's behavior is affected by the "Optimization Mode" parameter. Users are able to choose between various modes, including passive optimization, classic, CRula or GdRula. The robot's actions and movements can change depending on the mode selected, maximizing for various goals such as productivity, safety, or ergonomics.

The robot needs "Sensing Technology" in order to sense its surroundings. The kind of sensing technology—Xsens, Ergowear, Simulation or Offline test—that is employed can be specified by the user. This parameter controls the robot's data collection and environmental awareness, which is

essential for both safe and efficient operation.

The "Control Strategy" parameter is in charge of specifying the control scheme for the robot's movements and actions. Users can choose from a variety of control strategies according to their needs. This parameter could have choices for velocity control, or impedance that influence the way the robot moves through its surroundings and completes tasks.

- **Launch and Command Execution Options** Two separate options should be available in the application interface: one for starting the framework and another for executing commands with their corresponding parameters directly. While the 'launch' option initializes the ROS server and sets up the Ergoaware framework, the 'run' option executes the framework's main executable, which is normally in charge of directing the control of the robot.

- **User Management Through File Upload** It must be possible for users to upload files with user-specific data to add or edit user profiles. Makes it easier to add and manage user profiles in the Ergoaware framework. Ease of use and scalability depend on this feature. In order to effectively customize the Ergoaware framework for each worker, user management within the application is essential. The user files that have been uploaded include each worker's unique anthropometric measurements. For the purpose of calibrating and fitting the kinematic model for inertial data, whether it comes from Xsens or Ergowear systems, these anthropometric measurements are crucial. Each person has distinct anthropometric dimensions, which are essential for precisely estimating joint angles and segment orientations. This calibration procedure guarantees that the output of the framework closely matches the worker's actual movements and improves the accuracy of ergonomic assessments.

- **Simulator Launch Button** For users who plan to use a simulated robot for their experiments, an integrated button within the interface should make it easier to start a simulator. Provides an easy-to-use method for starting and setting up the simulator, an essential and frequent part of robot experiments. Simplifies user interactions and is consistent with the user-friendliness goal of the framework.

- **Task Productivity and Ergonomic Analysis Page** It is necessary to create an analysis page for real-time monitorization of parameters like "task productivity" and "human ergonomic score." The "ros topic echo" command is used to carry out this assessment. Gives users a thorough analytical tool to track the success of their experiments. The objectives of the framework, which include

increasing productivity and ergonomic safety, depend on the ability to evaluate task productivity and ergonomic scores.

- **Calibration Settings Page** A calibration page where the robot's preprogrammed positions can be set and modified should be part of the interface. To guarantee the robot's accuracy and precision, calibration is essential. Achieving the framework's goals of increasing manufacturing process efficiency and safety depends on this feature.

- **Framework Package Management Page** This functionality entails the creation of a specific page in the Ergoaware interface for installing and configuring necessary framework packages. Options to enable or disable each package must be provided in case these packages are already installed.

  Makes sure the necessary framework packages are easily accessible, installable, and configurable for users, making the Ergoaware framework setup process easier. This makes package management more user-friendly while lowering its technical complexity.

4.2 UI Design for Human-Robot Collaboration

The fundamental design tenets of simplicity, intuitiveness, consistency, and user-centered design form the basis of the **UI** design process. These guidelines are essential for guaranteeing that the user interface is highly functional, aesthetically pleasing, and compatible with the needs and preferences of the target non-technical user base.

The thorough examination of workflow patterns in the context of **HRC** was an important first step. The purpose of this analysis was to learn more about how end users—especially non-technical staff members—interact with the Ergoaware system. It was crucial to identify bottlenecks, user needs, and crucial touchpoints in order to determine the specific areas where **UI** design could have a significant impact on safety, efficiency, and productivity.

The information architecture of the user interface was carefully designed to guarantee an intuitive user experience. To improve overall usability and expedite user interactions, this required meticulous arrangement of data, menus, and navigation elements. To create an interface that is both aesthetically pleasing and easy to use, visual design elements such as color schemes, typography, and visual cues have all been carefully chosen.

In addition, the design considers the framework's future scalability potential, guaranteeing that the

interface can adjust to the changing demands of the manufacturing sector.

4.2.1 Design iterations

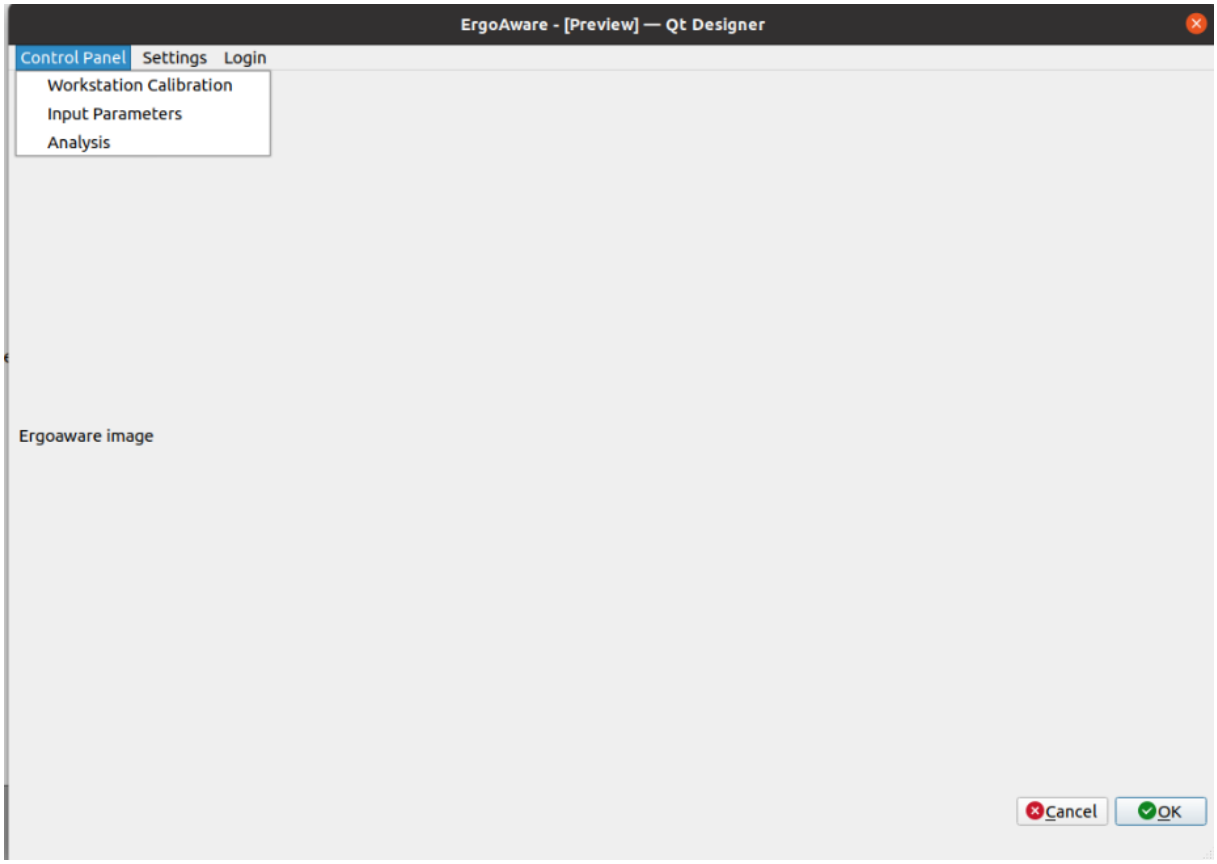


Figure 24: Attempt 1

The interface was minimalistic in its first iteration of design, as we can see in Figure 24. Three categories make up the taskbar's arrangement: Settings, Login, and Control Panel. More options appear when moving the cursor over these categories. Options for Workstation Calibration, Input Parameters, and Analysis are easily accessible through the Control Panel.

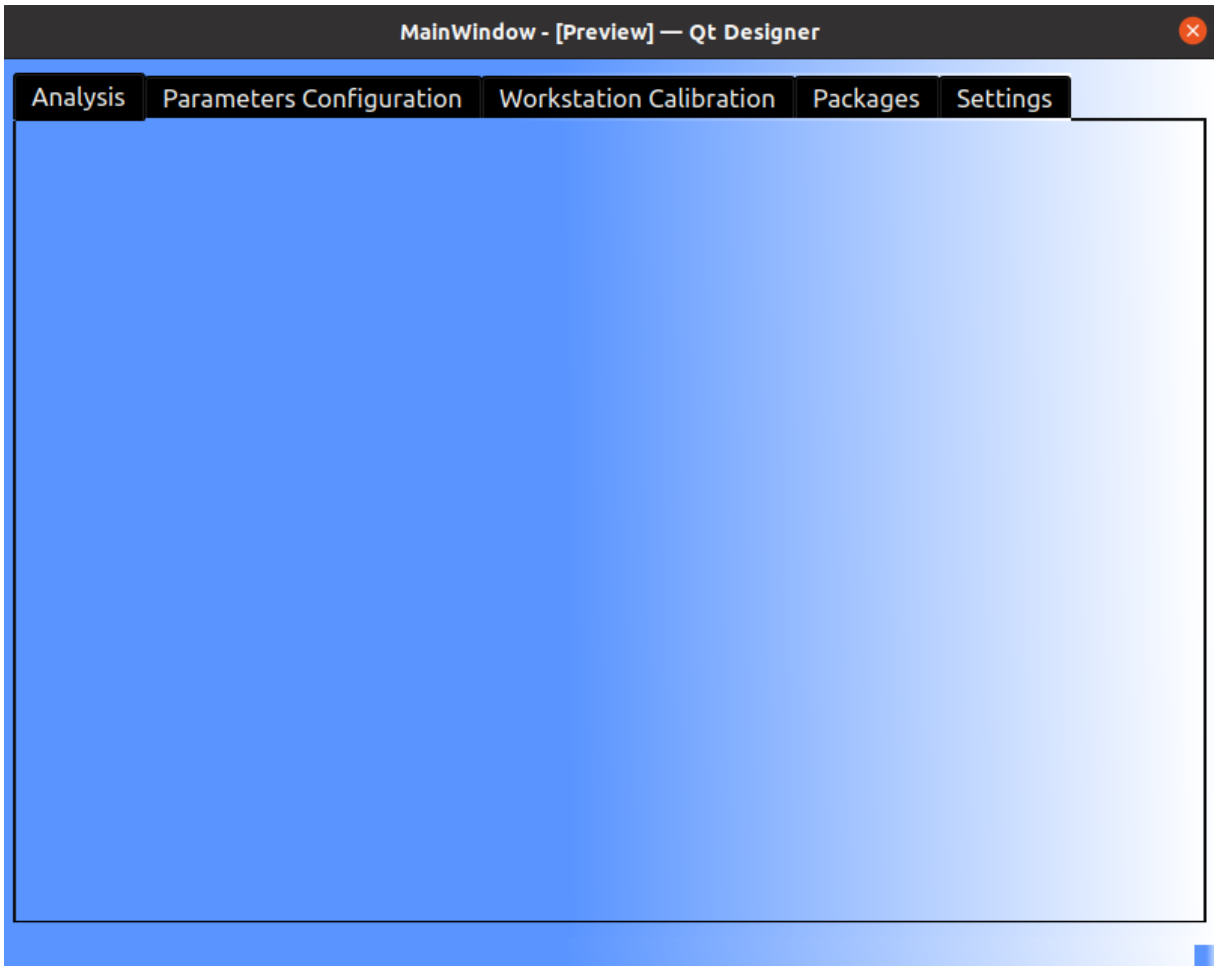


Figure 25: Attempt 2

As shown in figure 25, an interface that is more structured and colorful is introduced in the second design iteration. Navigation is made simpler with the replacement of the taskbar with a more user-friendly tab system. The tabs are now divided into five categories: Analysis, Parameters Configuration, Workstation Calibration, Packages and Settings. Tabs are easy for users to navigate between, and each one links to a different page or feature.

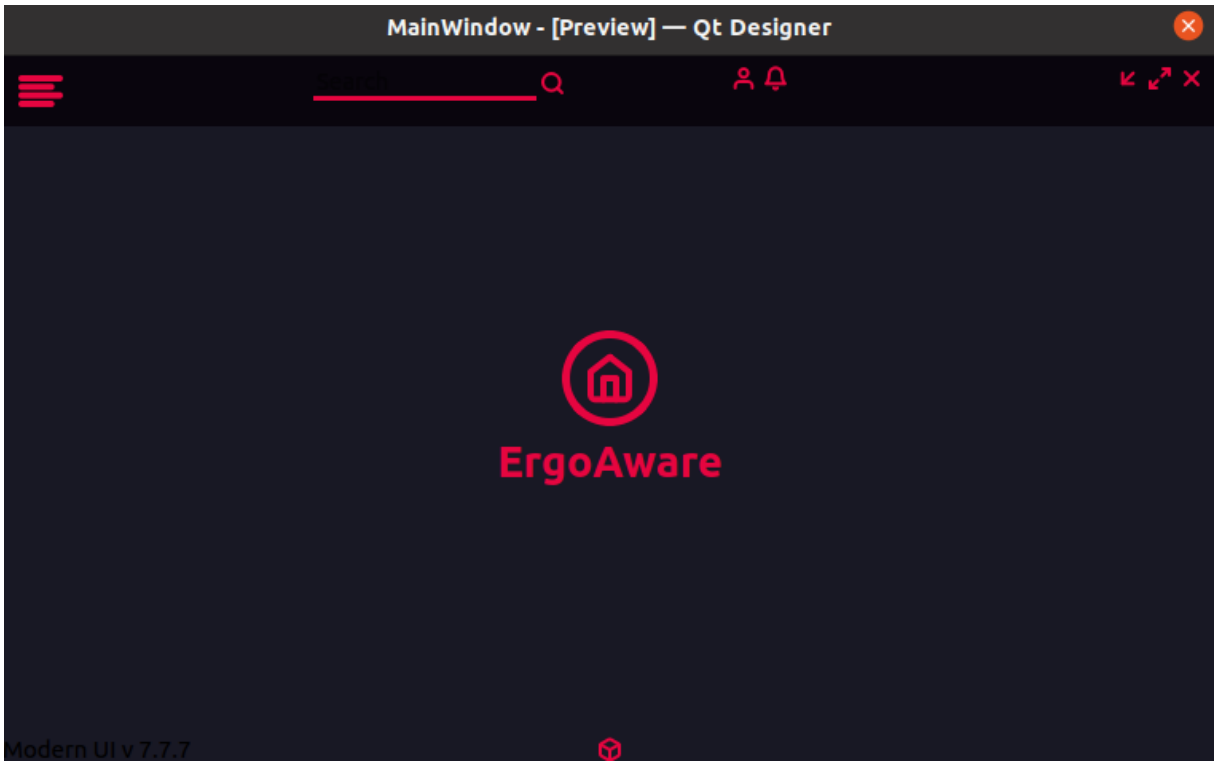


Figure 26: Attempt 3

Figure 26, the third design, adopts a dynamic sidebar menu, which is a major improvement. The background of the interface is changed to a simpler, darker design. To improve user experience, the sidebar menu is added as a primary navigation element. The use of red in the color scheme and the use of different icons creates a more appealing and contemporary visual identity.

Figure 27: Graphical user interface final result

The streamlined, black backdrop of the interface is carried over from the fourth and final design revision (Fig.27). But there's a noticeable change in the color scheme, with elements taking on a calming shade of light blue. This modification improves the user experience by giving the **UI** a feeling of balance and refinement. The light blue hue harmonizes with the entire style, resulting in a unified and aesthetically pleasing interface.

The journey to develop an intuitive and visually appealing front-end for the Ergoaware framework is reflected in these design evolutions. The finished layout offers users a polished and welcoming interface by combining a calm light blue color scheme with a tidy dark background. The sidebar menu's dynamic nature guarantees effective navigation, simplifying the process of accessing essential features and functionalities, and ultimately improving the user experience in general.

4.3 UI Implementation for the Ergoaware Framework

This section will examine the real-world application of the Ergoaware framework's user interface (**UI**), delving deeper into the practical side of the project. With the help of flowcharts, and insights into the inner workings of the framework, this section attempts to give a thorough explanation of the **UI** design and how it interacts with the Robotic Operating System (**ROS**).

The graphical user interface of the Ergoaware front-end was created using Python, utilizing the robust and adaptable PyQT5 library. A powerful Python library called PyQT5 makes it easier to create **UI**s that are both aesthetically pleasing and functional. The team used in-depth tutorials to fully understand this library functionalities.

Qt Designer, a design tool, was used to further streamline the development process. A visual design tool called Qt Designer provides a simple and effective method for creating graphical user interfaces for Python applications. Its workspace includes necessary elements that quicken the design process.

4.3.1 Qt Designer

Figure 28, shows an all-inclusive platform for designing and modifying the Ergoaware front-end's user interface, the Qt Designer workspace.

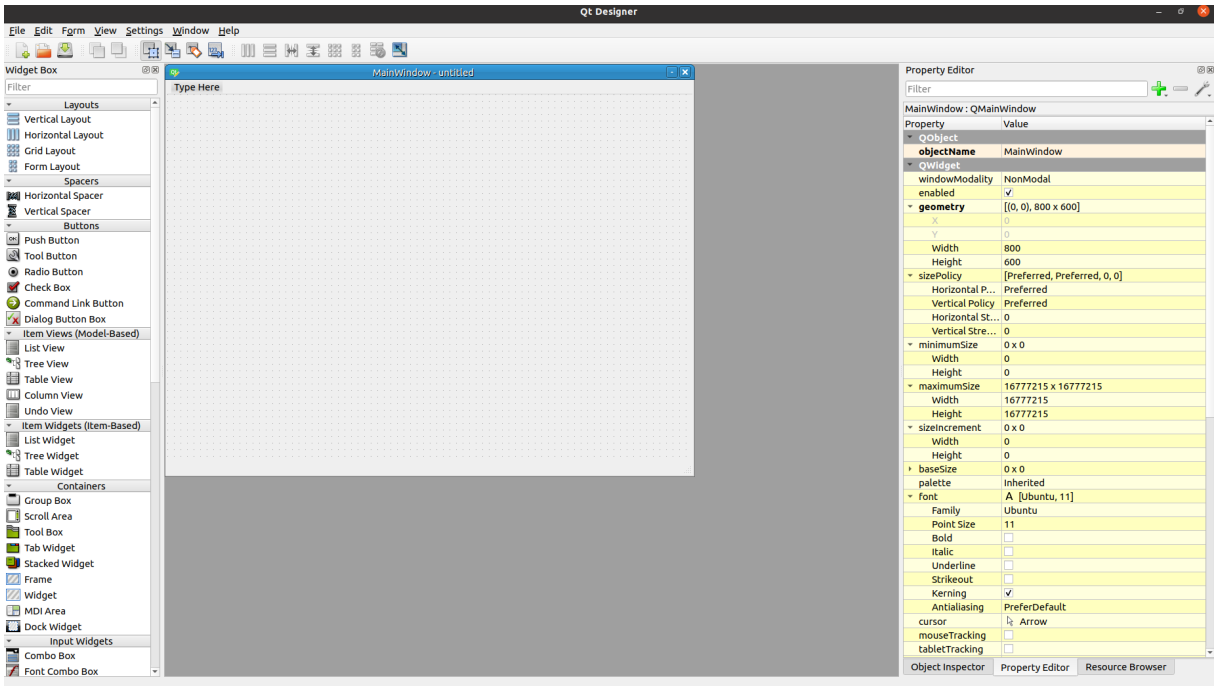


Figure 28: QtDesigner Interface

Widget Box

The Widget Box, which is situated on the workspace's left side, contains a variety of components that can be easily incorporated into the program. This comprises layouts, input widgets, buttons, containers, and more. Addition of these elements to the application's interface is as easy as drag-and-drop.

Property Editor

The Property Editor, which is located on the workspace's right side, provides extensive customization choices for every element. It gives developers the ability to adjust text font, color, size, and other attributes to perfection. Additionally, it permits the use of stylesheets with CSS, which offers a high level of visual customization.

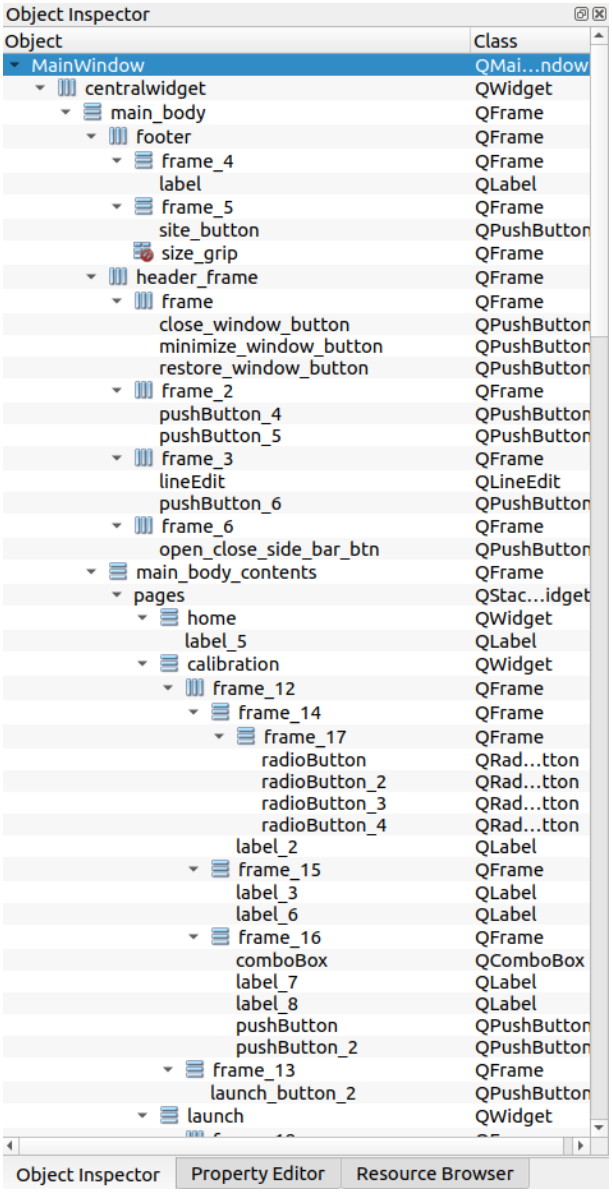


Figure 29: Object Inspector

Object Inspector

The Object Inspector, a valuable tool for developers, permits easy navigation through the hierarchy of widgets in the application, as we can see in Figure 29. This is particularly beneficial when dealing with complex user interfaces with multiple elements and containers.

Qt Designer generates a.ui file automatically, making the process of creating graphical interfaces easier. The "*pyuic5 interface.ui -o interface.py*" command is then used to start the conversion process, converting the .ui file into a Python script (*interface.py*). The graphical elements created in Qt Designer and the logic of the application are connected by this Python script.

The application's central component is the *main.py* file. The generated *interface.py* file is imported, and the complete user experience is coordinated. The *main.py* file defines the flow of the application, button actions, window size adjustments, and other essential functions. Additionally, it manages the application's graceful shutdown, making sure that all related processes—like the simulator and the Ergoaware framework—are closed properly upon the program's exit. Apart from the primary interface, the application integrates additional screens intended to provide guidance to users. These screens are helpful tools that give users detailed instructions and improve their application experience.

The careful integration of PyQT5, Qt Designer, and a strong Python codebase guarantees that the Ergoaware front-end is a highly functional platform that enables ergonomic and productive human-robot collaboration, in addition to being an aesthetically pleasing interface.

It is a remarkable example of the combination of strong development tools, **HRC**, and **UCD** principles, as demonstrated by its extensive **UI**.

The culmination of these components creates an interface that is both sophisticated and user-friendly, enabling users of all skill levels to make effective and efficient use of the application.

4.3.2 Crafting the Command Line: Enabling Ergoaware Framework Access

One essential component of the application is the ability to launch the Ergoaware framework via a command line interface. With the help of this feature, users can precisely configure and launch the framework, which simplifies their interactions with the robotic system. In order to achieve user-friendly and effective human-robot collaboration, this technical component of the application is essential.

Based on user selections from the **UI**, the createCommandLine function is in charge of creating and executing the **ROS** launch command. The process starts by extracting the chosen options from the **UI** elements, including the user, optimization mode, sensing source, and control strategy. It uses "velocity" as the default value for control_type, which is then used in the command. Because position-based control and velocity-based control are the two main control techniques in this system and they are mutually exclusive, the "velocity" control type is set as the default option. The default setting of "velocity" was made because it works effectively for activities requiring flexibility and instantaneous response. Users can adjust this parameter to "position" for jobs like ergonomic posture optimization that need for exact and accurate placement. This default offers flexibility, but it also highlights how crucial it is to choose the right control method depending on the demands of the task.

The default values for the configured parameters are predefined in accordance with the launch files utilized in the command line initiation of the robot. These launch files serve as templates for setting

initial configurations and options. The parameters are automatically initialized with these default values, providing a foundation for users to customize their selections while ensuring a seamless and consistent startup experience. The code determines whether "Choose an option" is selected as the user option. If this is the case, it prevents the command from being launched with insufficient parameters by displaying an error window and exiting the function. It also verifies the sensing, control, and optimization settings. When these options are selected as "Choose an option" or "Default," the code sets these parameters to their default values.

The chosen options are then combined to create the **ROS** launch command in the format needed to start the Ergoaware framework. Lastly, the code launches the created command in a new terminal window by using "subprocess.Popen". For tracking and management, the process is added to the subprocess_list, so then it is possible to close this window by closing the main application.

4.3.3 Interaction with ROS

Throughout this research, there were a number of difficulties encountered when creating the bar graphs for ergonomic scores. Establishing smooth communication between the **UI** and the **ROS** (Robot Operating System) framework, which provided real-time ergonomic score data, was one of the main challenges. It took a thorough understanding of Qt-based user interface development and ROS messaging systems to integrate the incoming data and make sure it was properly visualized in the form of dynamic bar graphs.

Moreover, one major challenge was the intricacy of organizing and parsing the incoming data. The structured data that made up the ergonomic scores had to be properly transformed into a format that could be shown on the bar graph. This involved handling data types and structures carefully, handling errors related to possible data problems, and having a thorough understanding of ROS message types.

Furthermore, a thorough understanding of the QtCharts library and how to dynamically update the charts when new data arrived were required due to the graphical representation of the bar graphs. It was a non-trivial task to manage the chart's appearance, labels, and axis scales to accurately depict ergonomic scores.

The requirement to maintain the **UI**'s responsiveness and aesthetic appeal in the face of continuous data updates further complicated the development process. Careful optimization and performance tuning were needed to strike a balance between real-time data processing and user-friendly interactions.

Overcoming these obstacles was a major accomplishment in the application development process, and it emphasizes how crucial it is for the **UI** and the underlying **ROS** architecture to communicate effectively. The accomplishment of dynamic bar graphs for ergonomic scores is a prime example of the dedication to

offering a clear and easy-to-use interface within the framework of human-robot cooperation.

ROSHandler is a custom class that extends QObject. It's used to handle ROS (Robot Operating System) callbacks for receiving data from **ROS** topics. "signal_update_values", a custom signal defined in the class, was used to emit a signal whenever new data is received from the **ROS** topic of type RULAScoresStamped. The signal is emitted to notify other parts of the application that new data is available. "handle_ros_callback" is the slot method that processes the data received from the **ROS** topic. When data is received, this method is called automatically. Inside the "handle_ros_callback" method, the emit statement triggers the "signal_update_values" signal and sends the received data as its argument. This signal is then used to update the **UI** with the new data. The "handle_ros_callback" method is responsible for receiving and processing data from **ROS** topics and updating the **UI** based on the received data.

This code facilitates the real-time update of the bar graph in response to data from **ROS**, allowing users to visualize and monitor ergonomic scores from the **ROS** topic within the user-friendly interface.

The "subscribe_ros_topic" guarantees that bar graphs such as the Human Ergonomic Score (HES) are updated in real time. To understand its function, let's break it down step by step.

A **ROS** node with the name "barras_node" is started by the first line. Nodes in the **ROS** are independent software components that interact with one another. As it prepares the application to interact with the **ROS** network, this initialization is essential.

Magic happens in the next line, 'rospy.Subscriber'. The program is subscribed to a particular **ROS** topic—in this example, '/human/ergonomic_score.' This topic is where other **ROS** nodes publish data pertaining to the Human Ergonomic Score, usually from the Ergoaware framework. Updates on data in real time can be received by the application by subscribing to this topic.

The callback function 'self.ros_handler.handle_ros_callback' is triggered whenever a new set of data is published to the '/human/ergonomic_score' topic. This callback function, as previously mentioned, analyzes the incoming data and updates the bar graphs appropriately. It is essential to making sure the application stays up to date with the evolving ergonomic data.

This code essentially creates a communication channel with the **ROS** ecosystem so that the application can receive updates on the most recent ergonomic information. This feature is vital for monitoring and improving **HRC** by giving users dynamic and real-time insights into the ergonomic score. The program stays data-driven and responsive thanks to the smooth integration with **ROS**, which increases its usefulness in actual manufacturing situations.

4.3.4 Package installation

The selected python packages are installed using the pip package manager. When a user chooses and authorizes the installation of a particular Python package from the application's frontend, the install_package function is called. It obtains the package name from the front-end input to be installed and it uses the subprocess.run function to run the pip install command inside the try block. With the help of the pip package manager, this command installs the given package.

The command's output is recorded, along with any success or error messages. The stdout=subprocess.PIPE and stderr=subprocess.PIPE parameters ensure that both standard output and standard error are captured. It examines the pip install command's return code. A successful installation is indicated by a return code of 0, which also includes the package name in the success message. It returns any error messages from the stderr along with an error message indicating that the package could not be installed if the return code is not 0. It captures exceptions and returns error messages in the event that something goes wrong during the installation process.

The package installation code checks for installation errors or makes sure the user-selected Python packages are installed so that the user can freely use the framework without having problems with **ROS**. The ability to control the installation of required packages is an essential component of the application's functionality.

4.3.5 Application's primary workflow

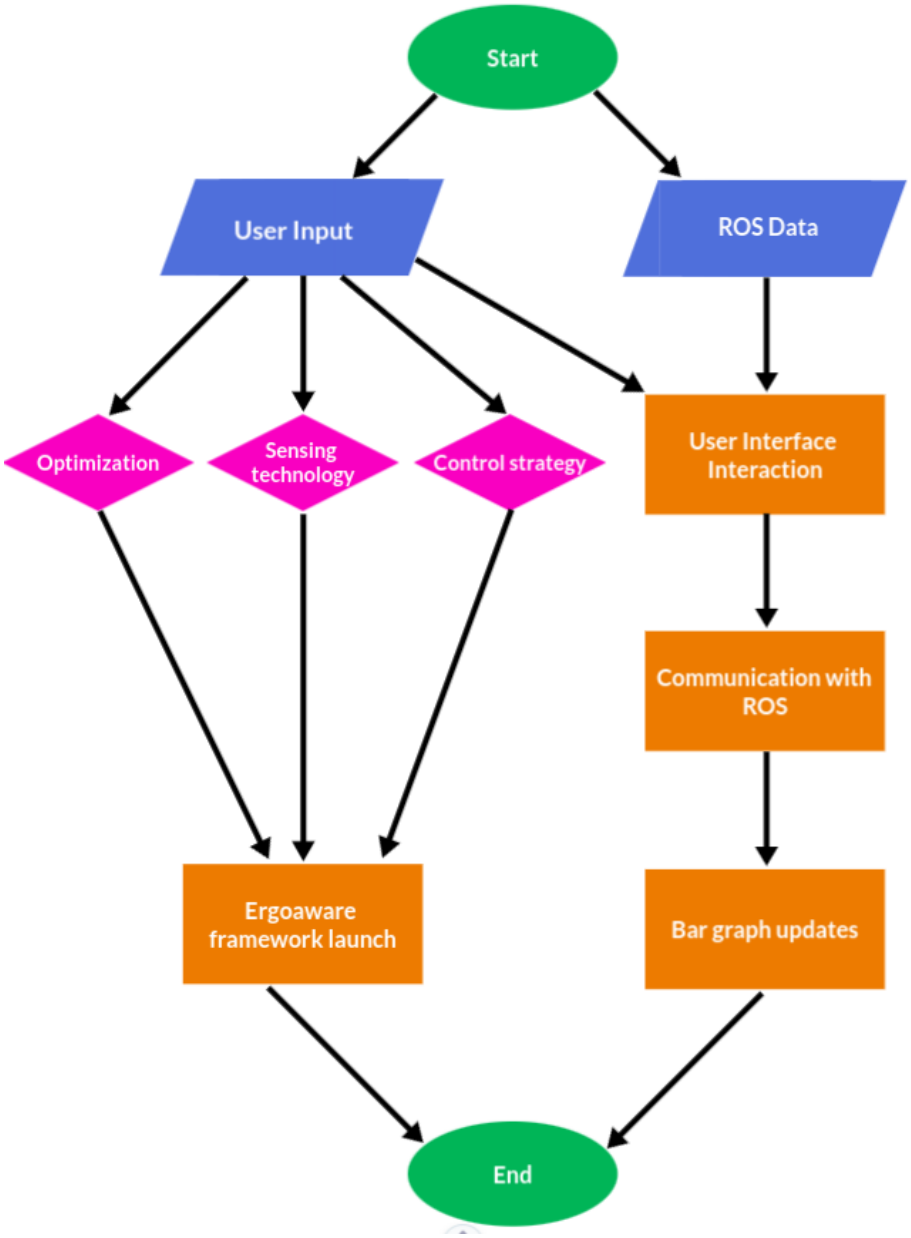


Figure 30: Application's primary workflow

The primary workflow of the application is depicted in this flowchart in Figure 30, with an emphasis on how it interfaces with the **ROS**. The primary goal of the application is to give users an easy-to-use interface for setting parameters and seeing real-time ergonomic score visualizations.

User interactions with the Ergoaware Frontend, such as setting parameters and starting the Ergoaware framework, are reflected in "User Input". The crucial decision of "Optimization Mode" is made when deciding which optimization mode to use. "Sensing Technology" refers to the choice made between

alternative sensing technologies and offline testing. "Control Strategy" refers to choices made for the Ergoaware framework's control strategies. After all the choices made in each of the parameters, these options are reflected in the framework launch.

"**ROS** Data" refers to the information obtained from **ROS**, including ergonomic score data. Input from the user is processed and actions are performed through "User Interface Interaction". The mechanism in charge of connecting to **ROS**, creating communication channels, and sending and receiving data is shown in "Communication with **ROS**". "Bar Graph Updates" describes in detail how the ergonomic score bar graphs are updated using **ROS** data processing.

This flowchart provides a thorough overview of the interactions between the Ergoaware Frontend and **ROS**, allowing users to set up parameters and view ergonomic scores in real time. It describes the decision-making points at which the program adjusts to user selections and interacts with **ROS** in a smooth and educational manner.

The flowchart shows how the Ergoaware Frontend improves ergonomics assessment for **HRC** by streamlining communication with **ROS** and simplifying a complex process and data flow.

Chapter 5

Results and discussion

This section discusses the application's results and discussion, which is critical in ensuring the usability, functionality, and effectiveness of the application. The validation process aims to confirm that the application meets the requirements, is user-friendly, and contributes to the overall goals of enhancing human-robot collaboration in ergonomic and productivity aspects.

Figure 31: Main Screen

Figure 31 shows the main screen that serves as the application's central hub, providing access to all key functionalities. The application includes a menu with options for navigating to the Control Panel (Workstation Calibration, Robot Launch, Analysis) and Settings (Package Installation, Help). This screen is the user's entry point into the application.

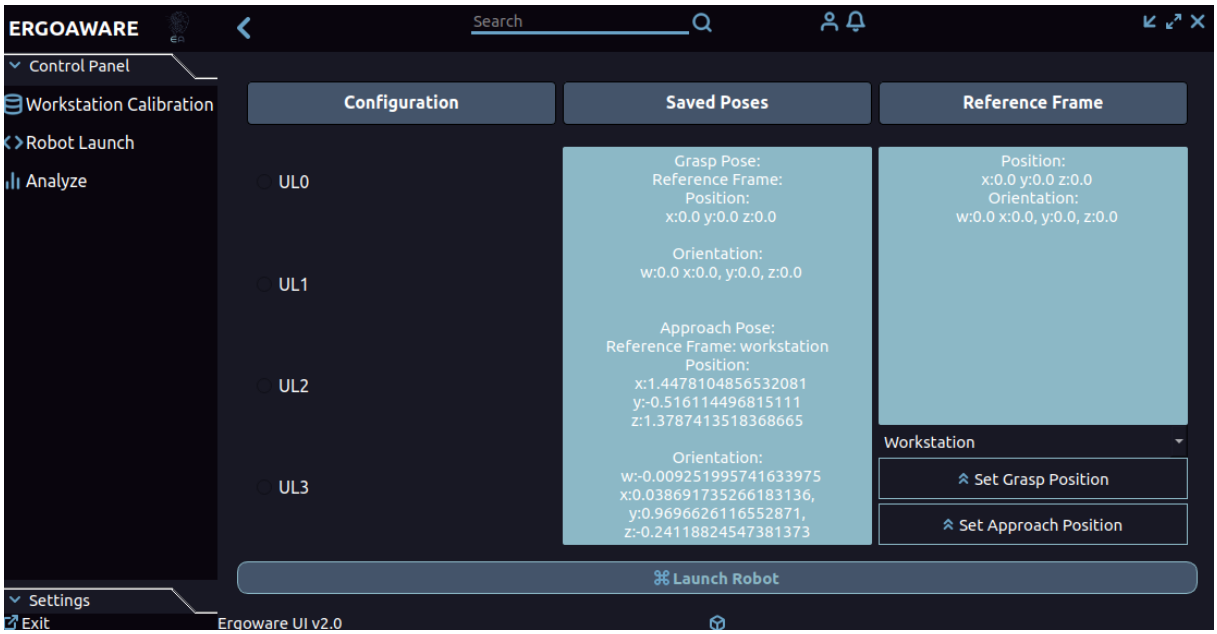


Figure 32: Workstation Calibration

An essential part of the application is the "Workstation Calibration" page, as shown in the Fiure 32, which is used to configure and adjust the robot's workspace and movements. Users can quickly prepare the robot for particular tasks or movements by selecting from a selection of pre-defined robot poses. Determining the grasp and approach positions on this page also ensures that the robot interacts with its surroundings in an efficient manner. After configuring every parameter, users can start the specified task to begin the robot's actions. "Workstation Calibration" optimizes the robot's performance and adaptability for a variety of tasks in **HRC** by streamlining the process of setting up the workspace and poses. With the help of this functionality, users can customize the workspace and behavior of the robot, increasing efficiency and productivity without compromising safety.

Although the robot is a key component of the team, it is not equipped with cameras or visual sensors to identify the exact location of the rails or storage spaces where the parts that need to be delivered to the worker are kept. Because of this limitation, a methodical calibration procedure is required to match the actual positions and orientations of these important components with the robot's understanding of the workstation. Features on the page enable accurate calibration in terms of orientation, position, and quaternions. It guarantees that the robot can navigate and interact with the workstation as precisely as possible by matching the robot's understanding of these parameters with the actual physical arrangement of the rails. The robot can now operate with a high level of accuracy after it has been calibrated. The calibrated robot can pick, place, or manipulate objects from the workstation. It can also adapt to the

changing needs of human-robot collaboration with ease. Accurate calibration enhances safety by lowering the possibility of mistakes and collisions. The robot's accuracy in identifying and interacting with the storage areas and rails improves workplace safety overall and reduces operational disruptions.

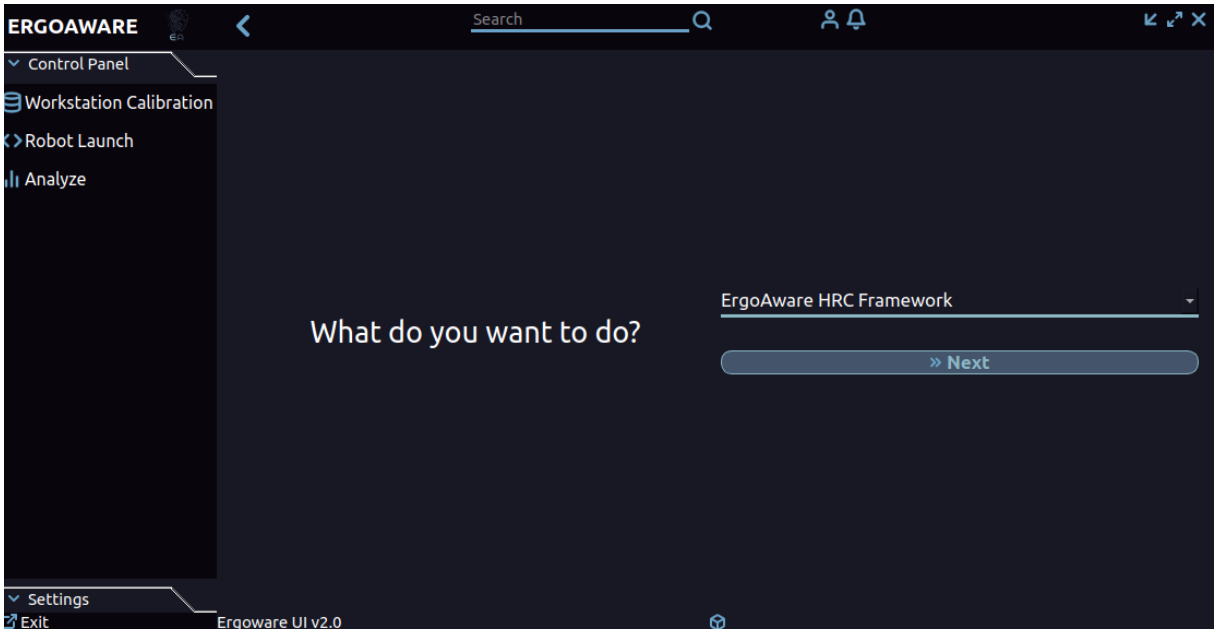


Figure 33: Robot launch menu

In the Figure 33 we can see a page where it is possible to choose which framework the user wants to launch. It has the possibility of selecting different frameworks, however only the Ergoaware HRC Framework option has been developed.

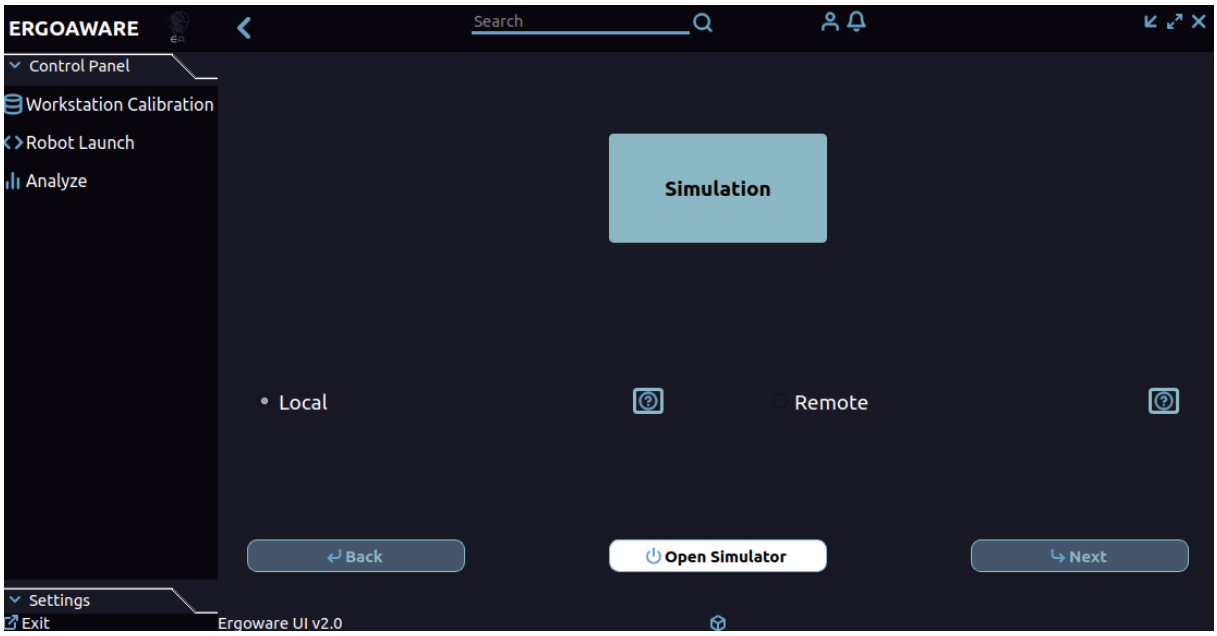


Figure 34: Simulation Page

Users can choose between launching the robot locally or remotely, as we can see in Figure 34. Additionally, it provides a way to launch the simulator app. Before launching the robot, users can select their preferred mode.

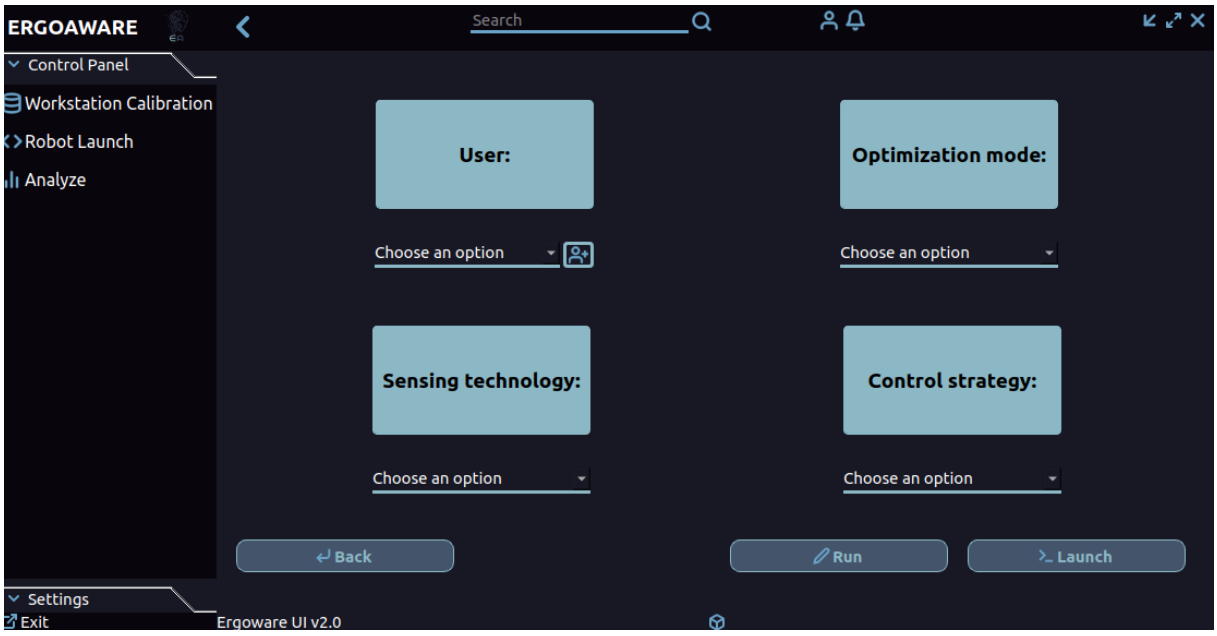


Figure 35: Launch Parameters

Users can set up the necessary parameters in this section in order to launch the robot as we can see in Figure 35. The user selection process, optimization mode selection, sensing technology definition, and

control strategy selection are examples of parameters. The Ergoaware framework is ready for use with these configurations. Figure 36 shows all of the options available for choosing each parameter.

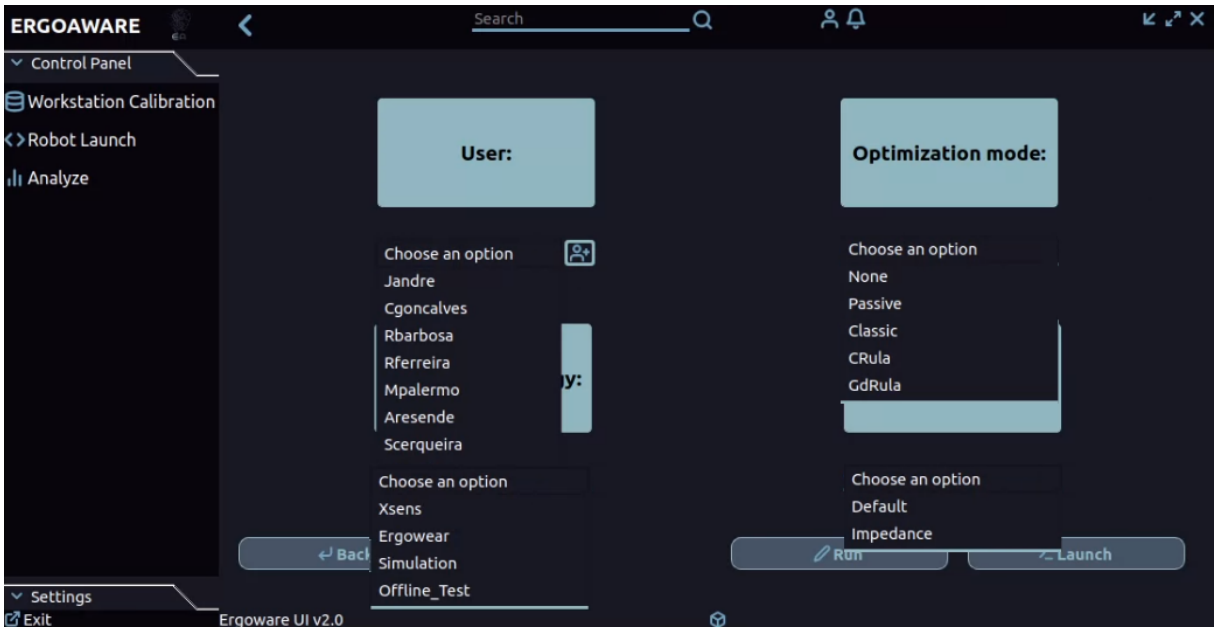


Figure 36: Possible launch parameters

By clicking the button to add a new user, is possible to access the file selection dialog box of the operating system. The process of adding users is made easier by this integration, which results in a clear and simple interface. With this functionality, users can conveniently select and upload a file that contains the anthropometric dimensions and relevant information of the new user.

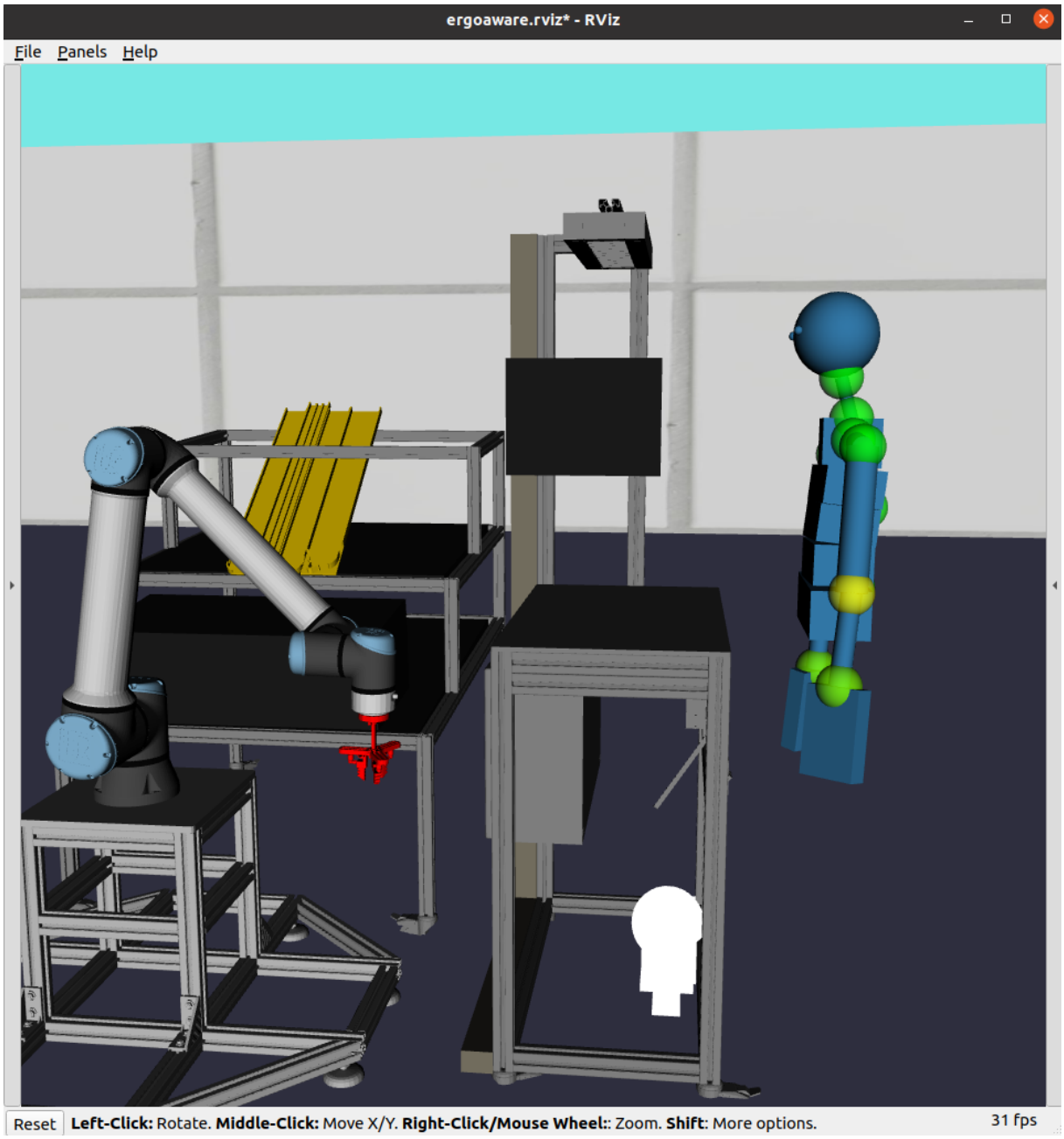


Figure 37: RViz

In Figure 37 we can see the RViz framework that demonstrates the advanced visualization feature that the Ergoaware framework uses to model and track robot behavior when performing tasks involving human-robot cooperation. With the help of RViz, users can view a comprehensive 3D visualization environment and get real-time insights into how robots move and interact with their surroundings. By controlling the robot's simulation in RViz, users can examine the behavior and interactions of the robot in a virtual setting. Before implementing robot operations in the physical workspace, this tool is invaluable for testing and fine-tuning them in a safe and controlled environment.

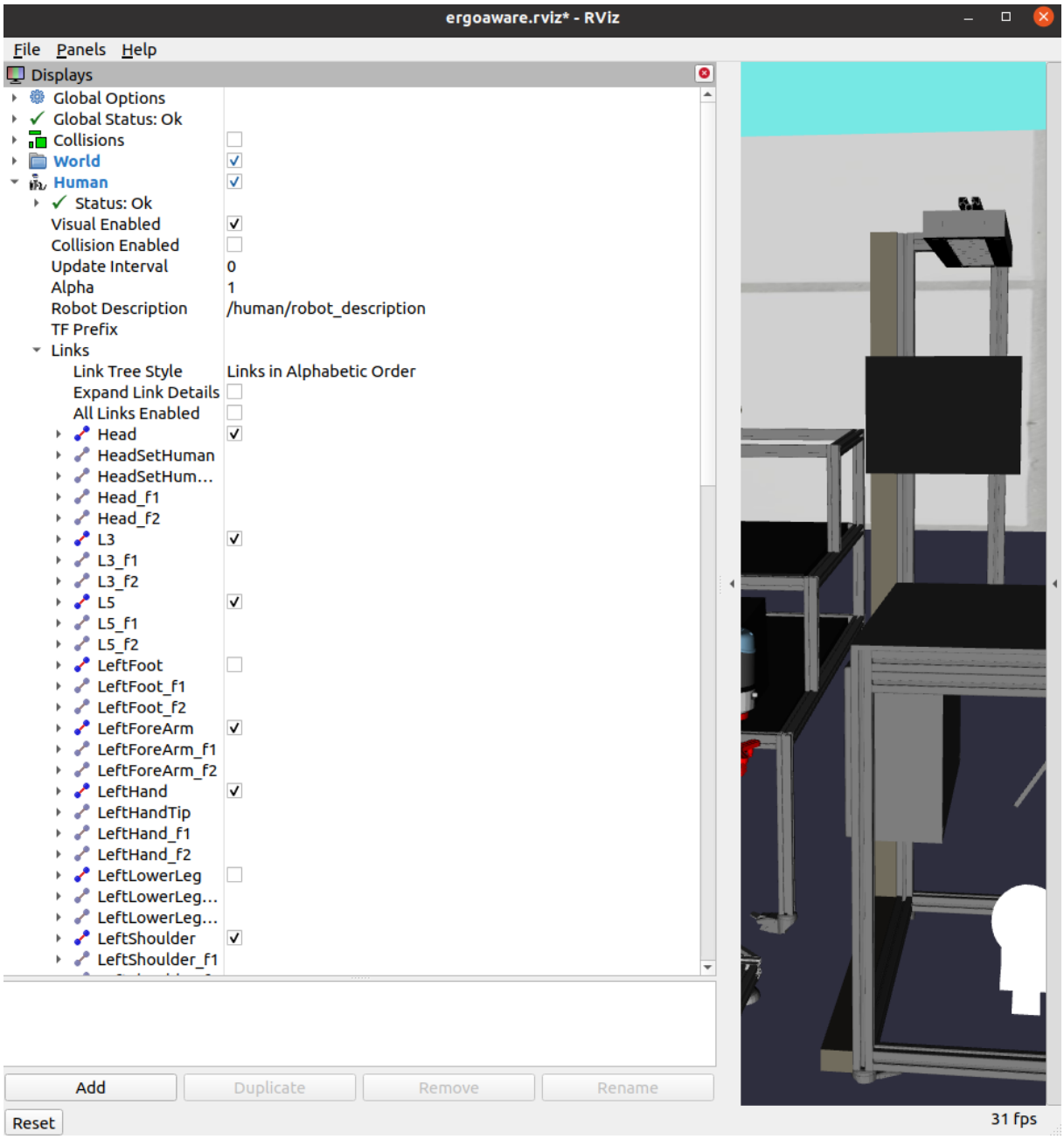


Figure 38: User parameters

Ergoaware's user interface allows users to interact with the system through an easy-to-use side menu (Fig.38). The ability to change and adapt the kinematic model according to the user's unique anthropometric measurements by choosing an user on the parameters selected before the framework launch, is one of the interface's primary features. Users can load their kinematic model into the user interface to make sure the robot behaves in a way that suits their particular physical characteristics and workspace needs.

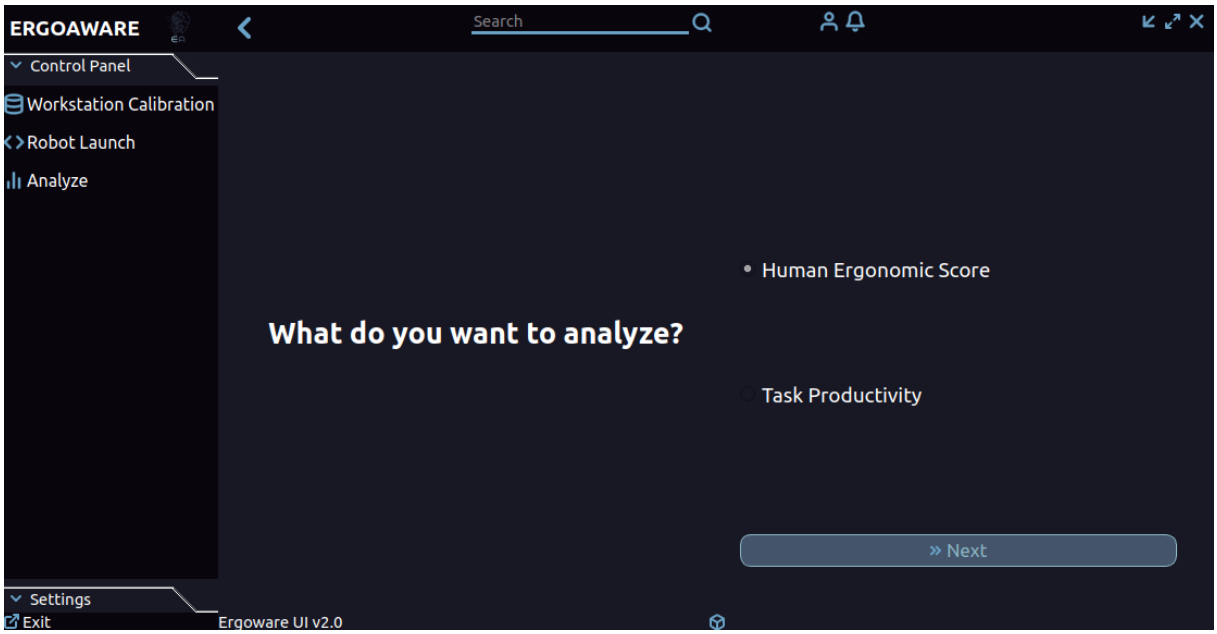


Figure 39: Analyze menu

The main location for key metric analysis is possible to see on the screen in the Figure 39. The "Human Ergonomic Score" and "Task Productivity" statistics are available for users to view, giving them the ability to track and evaluate these important variables. It provides a thorough understanding of the outcomes and functionality of human-robot cooperation.

Figure 40: Human Ergonomic Score

Figure 40 shows that users can choose to view the ergonomic score statistics for the left or right

side of the human body on this page. The Rapid Upper Limb Assessment (RULA) method, a well-known framework for evaluating the ergonomics of repetitive tasks, is the source of this score.

The Ergonomic Score is an all-encompassing indicator that considers the ergonomics of particular joints and body segments involved in human-robot interaction. This score is broken down into separate parts, each of which stands for a unique score connected to a specific joint or body part. The individual scores, denoted as A1, A2, B1, B2 and so on, correlate to particular anatomical regions and offer information about the ergonomic stress that those body parts undergo while performing a particular task. Low scores indicate better ergonomics. These scores show that there are fewer risk factors and that the postures are within reasonable ergonomic bounds.

Table A: Wrist Posture Score

Upper Arm	Lower Arm	1		2		3		4	
		Wrist Twist		Wrist Twist		Wrist Twist		Wrist Twist	
		1	2	1	2	1	2	1	2
1	1	1	2	2	2	2	3	3	3
	2	2	2	2	2	3	3	3	3
	3	2	3	3	3	3	3	4	4
2	1	2	3	3	3	3	4	4	4
	2	3	3	3	3	3	4	4	4
	3	3	4	4	4	4	4	5	5
3	1	3	3	4	4	4	4	5	5
	2	3	4	4	4	4	4	5	5
	3	4	4	4	4	4	5	5	5
4	1	4	4	4	4	4	5	5	5
	2	4	4	4	4	4	5	5	5
	3	4	4	4	5	5	5	6	6
5	1	5	5	5	5	5	6	6	7
	2	5	6	6	6	6	7	7	7
	3	6	6	6	7	7	7	7	8
6	1	7	7	7	7	7	8	8	9
	2	8	8	8	8	8	9	9	9
	3	9	9	9	9	9	9	9	9

Figure 41: Table A: Wrist Posture Score

The ergonomics of the wrist are examined in Table A, Figure 41 of the RULA evaluation. It takes into account the wrist movements' posture, force, and repetition when performing a specific task. Different wrist postures and forces are given scores in the table, which has a direct impact on ergonomic risk.

Elevated scores in Table A suggest a higher risk of ergonomic errors related to wrist movements.

Higher scores may be attributed to elements like excessive force, repetitive wrist motions, and extreme flexion or extension.

Neck Posture Score	Table B: Trunk Posture Score											
	1		2		3		4		5		6	
	Legs		Legs		Legs		Legs		Legs		Legs	
	1	2	1	2	1	2	1	2	1	2	1	2
1	1	3	2	3	3	4	5	5	6	6	7	7
2	2	3	2	3	4	5	5	5	6	7	7	7
3	3	3	3	4	4	5	5	6	6	7	7	7
4	5	5	5	6	6	7	7	7	7	7	8	8
5	7	7	7	7	7	8	8	8	8	8	8	8
6	8	8	8	8	8	8	8	9	9	9	9	9

Figure 42: Table B: Trunk Posture Score

Table B, in Figure 42, is devoted to evaluating upper body or trunk ergonomics. It takes into account a number of variables, such as repetition, force application, and trunk posture. The scores in Table B offer important details regarding the ergonomic risk related to posture and trunk movements.

Elevated scores in Table B indicate an increased risk of ergonomic strain on the trunk. Elevated scores can be attributed to various factors, including significant trunk bending, awkward postures, and repetitive movements.

Figure 43: Left analysis

Figure 43 shows the ergonomic scores that are displayed as a dynamic bar graph that updates in real

time in response to information gathered from ROS. In this case it is possible to see the ergonomic score for the left side. Users are able to evaluate and examine ergonomic scores for different body sides.

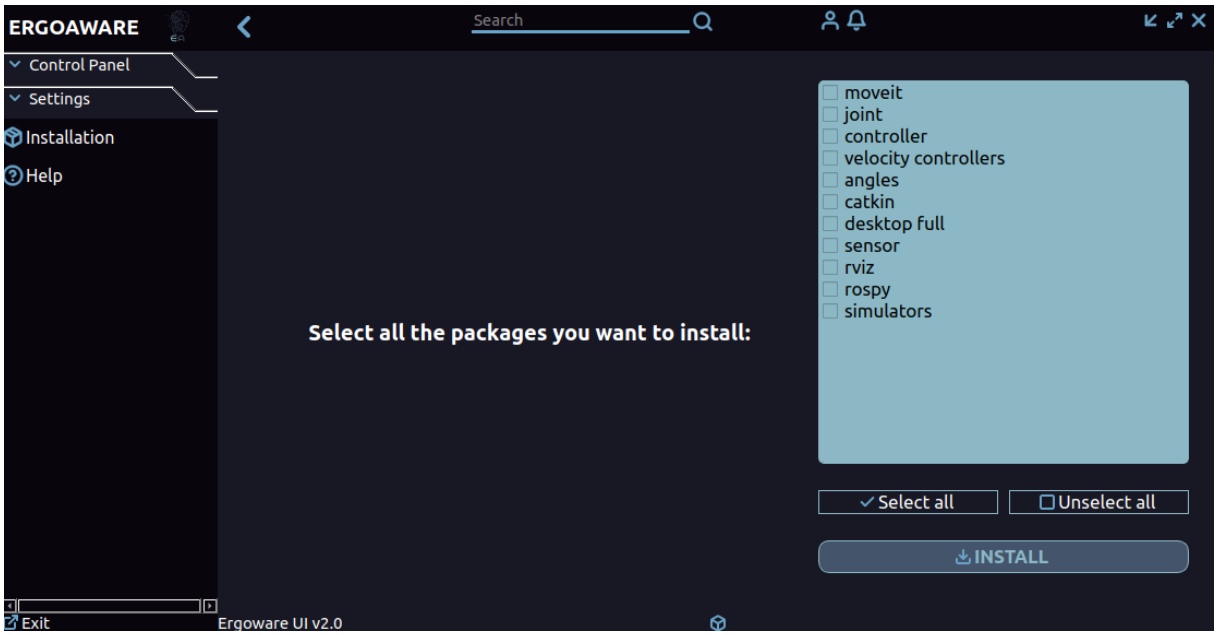


Figure 44: Package instalation

This section, as seen in Figure 44, facilitates the installation of necessary packages for the Ergoaware application. Users can manage the installation of framework-related packages, ensuring that the software is properly configured. It streamlines the installation process for a hassle-free user experience.

Chapter 6

Conclusion

The new era of **HRC** brought about by robotics technology advancements has profound effects across a range of industrial domains. Creating user-friendly interfaces that enable smooth communication between humans and robots is crucial to maximizing the potential of **HRC** systems, especially for non-technical staff. The main focus of this dissertation has been on human ergonomics, productivity, and safety through the development and validation of an intuitive front-end application within the Ergoaware framework.

The main goal of this study was to tackle the difficulties involved in creating an intuitive user interface that can act as a starting point for productive human-robot cooperation. The project took place against the backdrop of the **HRC** framework Ergoaware. The use of Robot Operating System (**ROS**), is essential to the Ergoaware framework. **ROS** has grown to become a vital resource for roboticists and researchers everywhere. It is the perfect fit for the Ergoaware architecture because of its flexibility and modularity, which allow high-level control systems, actuators, and sensors to be seamlessly integrated.

The Ergoaware architecture's modularity and flexibility offer significant potential for enhancing **HRC** in manufacturing industries. Navigating the architecture and backend of the Ergoaware framework, especially when grasping the intricacies of **ROS**, presented challenges. This involved understanding data processing, communication between nodes, and integration with external sensors and actuators.

This dissertation, which emphasizes the role of technology in raising task productivity and ergonomic scores, has offered insightful information about the creation and validation of user-friendly front-ends for **HRC** frameworks. There is still a great deal of room for innovation in human-robot cooperation as technology develops. This potential is exemplified by the Ergoaware framework and its user-friendly interface, which open up new possibilities for the manufacturing sector and beyond.

To sum up, this research has opened the door for more user-centered **HRC** interfaces that have the potential to further transform industrial processes by increasing their productivity, safety, and accessibility for a larger variety of users. Though there is still a long way to go, this application is a major step in the right direction toward seamless **HRC**. The accomplishment of creating and verifying an intuitive front-end

shows that technology can be used to increase task productivity and ergonomic scores while maintaining accessibility for a larger user base.

Bibliography

[1] S. Bevan, “Economic impact of musculoskeletal disorders (MSDs) on work in Europe,” Best Pract. Res. Clin. Rheumatol., vol. 29, no. 3, pp. 356–373, Jun. 2015.

[2] X. T. R. Kong, X. Yang, G. Q. Huang, and H. Luo, “The impact of industrial wearable system on industry 4.0,” ICNSC 2018 -15th IEEE Int. Conf. Networking, Sens. Control, pp. 1–6, 2018.

[3] A. Resende et al., “Ergowear: an ambulatory, non-intrusive, and interoperable system towards a Human-aware Human-robot Collaborative framework,” in 2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), 2021, pp. 56–61.

[4] Ecological interface design effectively reduces cognitive workload – The example of HMIs for speed control. URL https://www.sciencedirect.com/science/article/abs/pii/S136984782030423X

[5] European Commission, ”Horizon 2020, Work Programme 2018 - 2020”

[6] Robotics 2020 Multi-Annual Roadmap For Robotics in Europe Horizon 2020 Call ICT-2016 (ICT-25 ICT-26)

[7] User-centred design scheme. URL https://www.reveall.co/guides/user-centered-design

[8] D. Norman, ”The Design of Everyday Things”, pp. 187-218, 1988

[9] International Organization for Standardization (ISO) Technical Committee TC 159 ”Ergonomics” and Sub-Committee SC 3 ”Human-system interaction”, ”ISO 9241-210:2010, Ergonomics of human-system interaction - Human-centered design for interactive systems”, pp. 3-14, 2010

[10] What Is UX Design? URL https://purplegriffon.com/blog/what-is-ux-design

[11] Principles of User Interface Design. URL https://www.embedded.com/principles-of-user-interface-design/

[12] António Manuel Pereira do Anjo, "Modern Front-End Web Development", pp. 21-37, 2018

[13] Brooke, J., SUS: A "quick and dirty" usability scale. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, & A. L. McClelland (Eds.), Usability evaluation in industry, pp. 189-194, 1996

[14] System Usability Scale Benchmarking for Digital Health Apps: Meta-analysis. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9437782/

[15] Scalefit homepage. URL https://www.scalefit.de/home.html

[16] KUKA homepage. URL https://www.kuka.com/en-de

[17] Vivelab Ergo homepage. URL https://www.vivelab.cloud/

[18] KUKA Android and iOS app download. URL https://www.kuka.com/en-de/products/robot-systems/software/application-software/kuka-hrc-guide-app

[19] Roos, T., Riedmiller, M., Pollefeys, RoboEarth: A world wide web for robots. In 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 75, 2011

[20] R. T. Fielding, "Architectural styles and the design of network-basedsoftware architectures," Ph.D. dissertation, Univ. California, Irvine, 2000.

[21] D. Crockford. (2006, July). RFC4627—The application/json MediaType for JavaScript Object Notation (JSON) [Online]. http://www.ietf.org/rfc/rfc4627.txt?number=4627

[22] ABB's Collaborative Robots homepage. URL https://new.abb.com/products/robotics/robots/collaborative-robots

[23] Rethink Robotics' Baxter and Sawyer homepage. URL https://www.rethinkrobotics.com/sawyer/applications

https://www.fanuc.eu/ch/en/robots/robot-filter-page/collaborative-robots

[24] FANUC homepage. URL https://www.fanuc.eu/ch/en/robots/robot-filter-page/collaborative-robots

[25] FANUC homepage. URL https://www.yaskawa.eu.com/header-meta/news-events/article/yaskawa-s-new-cobot-motoman-hc20dt_n10277

[26] Kawasaki homepage. URL https://kawasakirobotics.com/?gclid=Cj0KCQiAlKmeBhCkARIsAHy7WVuQg2sOlXVQwGS9hHvdZN-IYUZR1gm5oHZUC4XILgPQJbxwNVvDRM0aAtJwcB

[27] Ricardo Tellez, "A History of ROS (Robot Operating System)". URL https://www.theconstructsim.com/history-ros/

[28] Robot Operating System Wikipedia. URL https://en.wikipedia.org/wiki/Robot_Operating_System