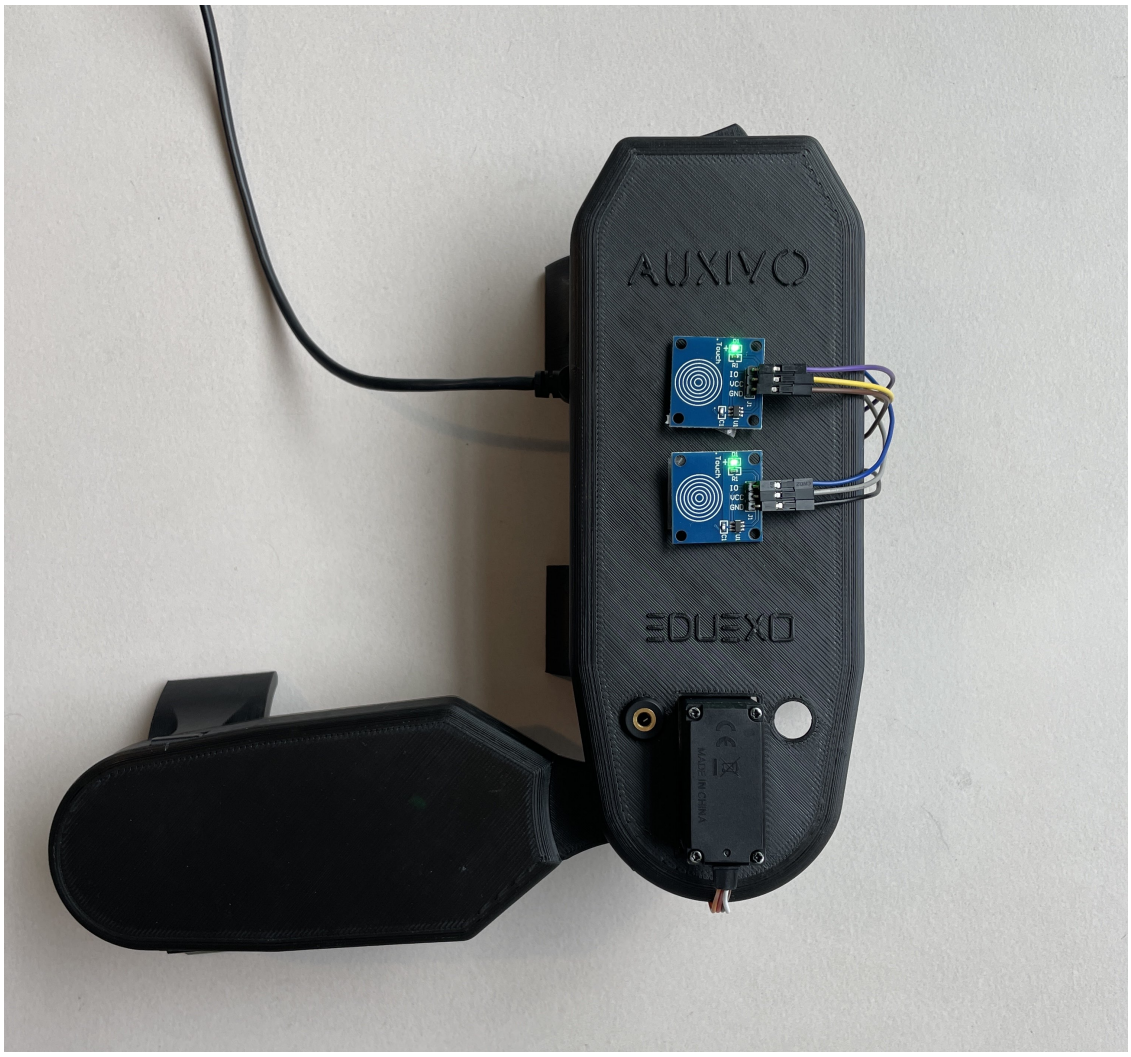
CHALMERS
UNIVERSITY OF TECHNOLOGY



Arduino/ROS2 Control Software Development for an Elbow Exoskeleton

The development of the exoskeleton software contributes to the modularity and scalability of the system

Degree project report in Mechatronics Engineering

Sophie Ha
Fanny Tingberg

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Degree project report 2025

This Report Outlines the Development of Arduino/ROS2 Control Software for an Elbow Exoskeleton.

The development of the exoskeleton software contributes to the modularity and scalability of the system

Sophie Ha
Fanny Tingberg

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2025

Arduino/ROS2 Control Software Development for an Elbow Exoskeleton
The development of the exoskeleton software contributes to the modularity and scalability of the system
Sophie Ha and Fanny Tingberg

Supervisor: Emmanuel Dean, Department of Electrical Engineering, Chalmers University of Technology
Examiner: Emmanuel Dean, Department of Electrical Engineering, Chalmers University of Technology

Degree project report 2025
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: The elbow exoskeleton set to 90 degrees.

Typeset in LaTeX
Gothenburg, Sweden 2025

Abstract

The following work presents the development of control software for an elbow exoskeleton, utilizing Arduino and ROS2. The exoskeleton is modeled as a 1-degree-of-freedom system with the primary goal of providing support for the user and assisting with movements. Additionally, the system is designed to be modular and scalable, allowing for future adaptations.
To achieve these objectives, research questions were created to more precisely understand what needed to be done to achieve the goals. The questions were also based on the limitations of the projects that were set to make the project feasible since it could easily become very complex. The limitations included using a preexisting frame of the orthosis to focus on software development rather than mechanical design, focusing on development for one arm with a range of motion of 0-90 degrees, and using a basic user interface based on touch sensors.

The development process began with a preliminary study to identify suitable components and pre-existing orthosis. The hardware configuration and system architectures were then established to prepare the hardware for the software implementation and to facilitate easy identification and debugging. The software was developed separately for ROS2 and Arduino. The Arduino software was developed to interface with hardware components (touch sensors, encoder, and motor) while enabling serial communication with ROS2. ROS2 was structured into two primary nodes, one for the trajectory and the other for serial communication. Finally, the system underwent testing, beginning with partial tests to verify individual components, followed by full system testing to ensure functionality and seamless interaction.

The results are validated through experimental testing, demonstrating that the exoskeleton provides assisted movement when the sensors are pressed. The system is adjustable to user preference, with movement data visualized in a graph. Additionally, ROS2 allows the user to change to the preferred range of motion. The modular architecture allows for the creation of additional ROS2 nodes for specific tasks.

Challenges encountered during the development, such as actuator torque limitations, are discussed, along with potential future improvements. These include changing to a more powerful actuator to provide better assistance or to fully compensate for the loss of upper arm muscle functions.
In summary, this report presents the development of a modular and scalable elbow exoskeleton using ROS2 and Arduino to facilitate assisted movements, offering opportunities to enhance performance and expand its functionalities.

Keywords: Robotics, ROS 2, Exoskeleton, Arduino, Elbow

Acknowledgements

We would like to thank our supervisor and examiner, Emmanuel Dean, for the opportunity to work on this project, as well as for his patience and support throughout the process. His consistent guidance regarding the project and the report has been greatly appreciated. With both our curiosity and Dean's expertise, we have thoroughly enjoyed working on this project, continuously learning and gaining valuable experience in programming and problem-solving.

Sophie Ha and Fanny Tingberg, Gothenburg, February 2025

Contents

List of Figures

1
Introduction

At present, the increasing number of musculoskeletal disorders has emphasised the critical need for innovative solutions in the field of rehabilitation [1]. These conditions significantly impact the quality of life for individuals and require solutions beyond traditional rehabilitation approaches [2].

In this work, we further develop a pre-designed elbow exoskeleton with a focus on the software to provide mobility. The goal is to make the system both modular and scalable, ensuring that it can be adaptable for other exoskeletons and meet the requirements of future works.

1.1 Background

The electrical engineering department at Chalmers has collaborated on a project with Sahlgrenska University Hospital to develop an autonomous elbow orthosis.
The orthosis is designed to provide support to patients with musculoskeletal disorders, where upper arm muscle functions are limited. The exoskeleton is used to compensate for the lack of muscle functions and should, therefore, primarily assist with movements. The goal is to allow the user to adjust the position of the arm by controlling the orthosis and changing the angle of the elbow joint, which has a range of motion (ROM) of 0-90 degrees. It should also assist in maintaining the position of the arm while providing comfort and support during the specific task.

Today, there are a significant number of developed exoskeletons. However, not many are designed to offer support for patient rehabilitation. Among the companies that target medical exoskeleton, AUXIVO's Eduexo is one notable example [3]. This project focuses on developing a system where the user can control the coordinates and the movement of the exoskeleton to meet their specific needs and rehabilitation tasks.

Wearable robotics that have been designed to offer support to the user has to have a specific design and focus on features such as portability, lightweight, and comfort, as well as being safe and precise. Furthermore, the solution must have an aesthetic appeal. There are challenges to the development of exoskeletons and this project aims to address some of these challenges to contribute to the development of new exoskeletons.

To enhance the portability and comfort of the user, lightweight yet durable material for the physical parts will be used for the orthosis. To make a precise and safe system, control algorithms will be developed, including motion limits. Additionally, making the system modular and scalable is a goal, allowing the software to be adaptable to other exoskeletons.

1.2 Purpose

The primary objective is to develop a safe and user-friendly exoskeleton for the elbow to be able to compensate for the muscle functions of a person without upper limp muscle functionality. The long-term goal is to allow the user to regain independence by providing and enhancing mobility for everyday tasks like gripping an object. An initial step to achieve this is to focus on developing Arduino and ROS2 software for real-time interaction to interpret sensor data, allowing for control, and providing real-time feedback for movement.

Additionally, the development also aims to enhance the system's modularity and scalability, allowing for future adaptation and integration of other exoskeletons.
Ultimately, this project aims to provide insight into the main objective and, in the long-term, contribute to the goal of a richer life quality and a more active daily life for users.

1.3 Limitations

The development of an elbow exoskeleton, particularly one with a focus on the software and real-time interface, is complex and ambitious. Therefore, there need to be limitations for the project to be feasible. This section describes these limitations.

- The frame construction and tests will be made on a pre-existing orthosis design. There will be no custom design of the exoskeleton itself. This is because the project will focus on software development and the exoskeleton's behavior, not the mechanical design, CAD, or its aesthetics.
- Since the project will be a work for future development, the aim is not to create a finished product but rather to explore and research. This goes in hand with the limited time frame.
- The construction will be focused on one arm, in our case, the right arm. This will reduce complexity and provide a manageable scope within the time frame.
- The Robot's Range Of Motion (ROM) will be between 0 and 90 degrees. This range effectively covers the range for daily tasks such as gripping or reaching for objects.
- The user interface will be limited to basic control functions using touch sensors. This is simple to integrate and provides the control user interface needed. Future work could explore a wider range of control options such as EMG-sensors.

1.4 Research Questions

The project focuses on developing software for the control of an elbow exoskeleton, utilizing Arduino and ROS2. To achieve this goal, several important questions need to be addressed. This section lists these questions.

- What user interface will be used?
- Will the system be able to provide accurate position based on user input?
- Is it possible to create an interface for the hardware system and the ROS system?
- How can we create a modular software architecture?

2
Theory

This section provides an overview of the components and the technical background needed to form the basis of the elbow exoskeleton system. These elements are crucial in the development of the exoskeleton and provide an understanding of how they work together to create a functional system.

2.1 Exoskeleton

Exoskeletons are a type of wearable robotic device that is external to the body and can take various forms such as full-body suits or a leg/arm attachment [4]. They are designed to interact with the user to effectively provide assistance or improve functionalities of the moving parts of the body, to the user's specific needs [5]. The application covers a wide range of domains such as movement aids for disabled people or improving range of motion to reduce physical burden in environments where physically demanding work needs to be executed [6].

Figures 2.1, 2.2, and 2.3 illustrate what an elbow exoskeleton can look like and what components the exoskeleton can be made up of, including its joints, links, electronic components, and frames of reference.

Similar to other robots, robotic exoskeletons are made out of various components that work together to create the functionalities of the system. They are also essential to achieve the desired level of assistance and support. Below, the key components are presented and can be categorized into:

- **Mechanical structure:** The base for the exoskeleton includes its rigid frame, links, and joints.

- **Robot Modeling:** Modeling of the structure which includes kinematic- and dynamic modeling as well as control. This is vital to be able to understand the behaviour of the robotic system and to the development of the control system.

- **Electrical components :** Physical components that are essential to provide the control for mobility assistance. These include components such as sensors, actuators, microcontrollers, and a control system.

- **Software:** Set of algorithms and instructions that runs the system including

reading and interpreting sensor data, the control and trajectory planner

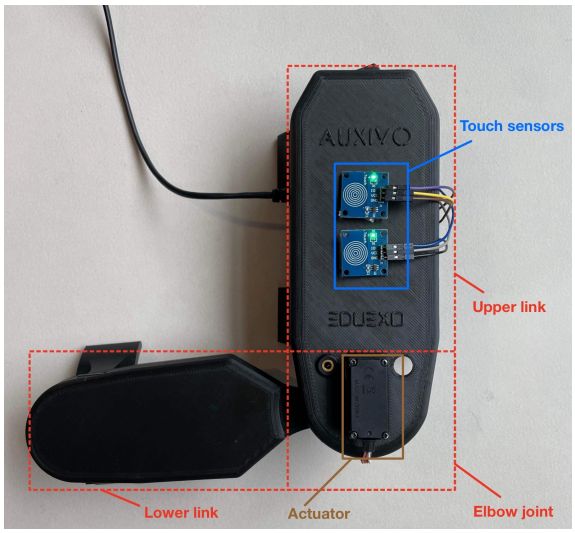


Figure 2.1: Elbow exoskeleton (front view) with its key components.

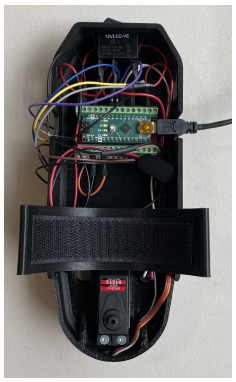


Figure 2.2: Elbow Exoskeleton (internal view) with component layout.

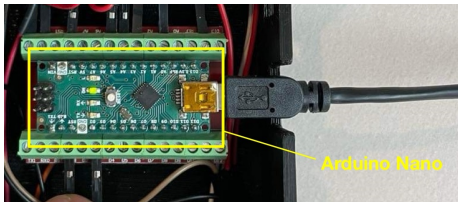


Figure 2.3: Close up of the microcontroller within the Elbow Exoskeleton.

2.1.1 Mechanical structure

Links

Similar to how we humans have joints and bones, the upper limb exoskeletons have joints and links. Links are the rigid bodies that connect the segments of the exoskeleton by the joints [7]. As seen in Figure 2.1, the upper limb has two links connected to an elbow joint.

Joints

To mimic the human elbow, the upper limb exoskeletons typically have an elbow joint [8]. The joint allows for the links to move or rotate in the elbow within its range of motion. The revolute joint, as shown in Figure 2.4, has one degree of freedom (DOF) where the rotation is on the z-axis and the joint variable θ represents the displacement from joint i to i-1.

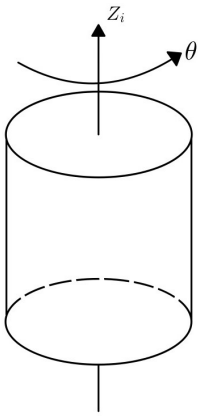


Figure 2.4: Revolute joint.

2.1.2 Robot Modeling

In robot analysis, a crucial step includes modeling the robot and its kinematic behaviour. For an elbow exoskeleton with a single degree of freedom (DOF) based on the elbow joint, the model can be defined as pictured in Figure 2.5.

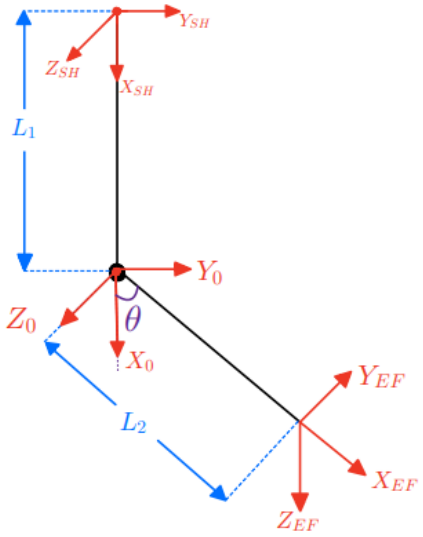


Figure 2.5: Kinematic model of the upper limb with one DOF

Figure 2.5 visualizes the kinematic model. The system consists of one joint at the elbow and represents a system moving in one plane. The model includes parameters defined as:
$\boldsymbol{L_1}$: Length of the upper arm (the segment from the shoulder to the elbow)
$\boldsymbol{L_2}$: Length of the under arm (the segment from the elbow to the end-effector)
$\boldsymbol{\theta}$: Joint angle at the elbow, representing the rotation of the under arm relativ to

the upper arm
$\boldsymbol{x}_{\text{SH}}, \boldsymbol{y}_{\text{SH}}, \boldsymbol{z}_{\text{SH}}$: World coordinate frames denoted as (SH) for the shoulder.
$\boldsymbol{x}_{0}, \boldsymbol{y}_{0}, \boldsymbol{z}_{0}$: Coordinate frame for the elbow joint
$\boldsymbol{x}_{\text{EF}}, \boldsymbol{y}_{\text{EF}}, \boldsymbol{z}_{\text{EF}}$: Coordinate frame for the end-effector (EF).

Kinematic modeling

The kinematic modeling is fundamental in describing the robot's movements and usually involves two sub-problems [9]:

- **Forward kinematics (FK):** This describes the position and orientation (pose) of the robot's end-effector (the end of a robot's arm), with respect to the joint variable. In simpler terms, for a specific set of joint angles, **FK** can help determine the exact pose of the end-effector.

- **Inverse kinematics (IK):** This problem is generally more complex than the **FK** [10]. As previously stated, in the **FK** problem the exact pose of the end-effector could be determined based on a unique set of joint angles. The **IK** problem on the other hand involves finding the joint variables based on the known pose of the end-effector and is essentially the inverse problem of **FK**. Since a robot can have more than one DOF, meaning it has more than one joint angle, the solutions to an **IK** problem could have multiple solutions.

To perform the **FK** analysis, it is common to use the Denavit-Hartenberg (DH) convention [10]. The DH convention involves selecting the reference frames attached to each robot link. These are then used to create homogeneous transformations which describe the relationship between two adjacent links. These conventions are defined using the product of four parameters (θi, ai, di, αi) that are associated with the link i and the joint i. These parameters represent the joint angle (θ), link length (ai), link offset (di), and link twist (αi) of joint i.

When the DH parameters are computed, they can be applied to obtain both the relative and the absolute Homogeneous Transformation (HT) matrices. The relative HT matrix is necessary to describe the transformation from one link to another, specifically from the previous link (i-1) to the current link (i), H_{i-1}^{i}. The absolute HT matrix describes the pose and the transformation of the end-effector with respect to the base frame. These matrices represent the spatial relationship between the links in the manipulator. Through matrix multiplication, where all relative HT matrices (H_{i-1}^{i}) are multiplied sequentially, the absolute transformation matrix can then be obtained, denoted as H_{ef}^{0}
For an upper limb exoskeleton, the equation could be expressed as:

$$T_{EF}^{SH} = \begin{bmatrix} c(\theta) & -s(\theta) & 0 & L_2 c(\theta) \\ s(\theta) & c(\theta) & 0 & L_2 s(\theta) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 2.6: Forward kinematic model of the upper limb with one DOF

In order to determine the velocity and direction of movement of the end-effector based on the elbow joint's rotation, the derivation of the Jacobian matrix has to be done. This is called differential kinematics. The Jacobian is calculated by determining the partial derivatives of the end-effector's pose with respect to the elbow joint's angle. This involves:

- **Positional Derivative**: Calculating the partial derivative of the end-effector's position with respect to the elbow joint angle, which shows how changes in this angle affect the end-effector's position along the x, y, and z coordinates.
- **Orientation Derivative**: Determining how the elbow joint's rotation influences the orientation of the end-effector, represented by the rotation axis vector of the elbow joint [10].

Dynamic modeling

As previously stated, kinematics describes the motion of the robot. It doesn't consider external forces or torques acting on it. In contrast, the dynamic modeling of the system deals with that, in fact, the dynamic equations explicitly describe the relationship between force and motion[10]. The Euler-Lagrange methodology is often used when deriving the dynamics (equations of motion) of the system. For a one DOF system, the model has the form [11]:

$$\mathbf{M}(q, \dot{q})\ddot{q} + \mathbf{C}(q, \dot{q})\dot{q} + \mathbf{B}(q)\dot{q} + \mathbf{G}(q) = \tau \tag{2.1}$$

Here,
$\boldsymbol{M}(q, \dot{q})\ddot{q}$ represents the inertia matrix.
$\boldsymbol{C}(q, \dot{q})\dot{q}$ represents the centripetal and Coriolis effect matrix.
$\boldsymbol{B}(q)\dot{q}$ represents the viscous friction matrix.
$\boldsymbol{G}(q)$ represents the vector of gravitational torque.
$\boldsymbol{\tau}$ represents the input joint torque. $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are vectors that represent the joint position, velocity, and acceleration.

The equation of motion is important to describe the required torque to move the joint. It also describes the dynamic behavior including the motion of the system with respect to external forces. This model is important for the design of the control algorithms that are used for following a trajectory or position control [10].

Control

One of the main goals in modeling and controlling the exoskeleton is to get the behavior of the manipulator to match the desired motion. This could be that it

should accurately maintain and follow the intended trajectory or hold a position. An ideal system should be able to dynamically compensate for any deviations or disturbances that occur during the process. Utilizing feedback is one of many methods that could be implemented to reach the goal. Another common technique is, for example, the use of PID control in the system to tune errors and adjust its actions. Using PID control, the errors are adjusted based on the actual values of the outputs with respect to the desired value [12].

2.2 Electrical components

In this section, the electrical components used in the project will be listed and described to provide background knowledge and understanding of their functions and roles within the project.

2.2.1 Touch Sensor

Touch sensors are also called tactile sensors and are sensitive to touch, force, or pressure [13]. They are electronic components that react and detect body capacitive touch and are used in various applications ranging from smartphones to industrial control panels [14]. The touch sensors are used to detect information like the user's intended movement and changing the joint position.

Figure 2.7: Touch sensor.

Capacity touch detection is based on the technology behind capacitive sensing and capacitive coupling, where information and energy are transferred between two electric circuits or two conductive layers [13], [15]. The layers are separated by a small

space that creates a conductive path when the capacitive coupling senses and detects conductive objects. It could be a fingertip or anything that has a different dielectric property than that of air. The capacitance is then converted to an electrical signal which can then be interpreted in the component's controller or microcontroller where the signal is processed into appropriate input or output [15]. By utilizing this principle, the surface of the screens on the sensors can sense and detect touch without any actual force being applied to them [13].

2.2.2 Microcontroller: Arduino Nano

A microcontroller can be defined as a simplified computer where all the components and interfaces are integrated and contained on a single board. It is generally designed to execute a program repeatedly on a timed loop [16]. The Arduino Nano is a compact microcontroller board with integrated USB, specifically, it uses a Mini-B USB port [17]. It can be used for controlling hardware components such as reading sensor data, controlling actuators, and communication between other electronic components. The Arduino Nano is a good microcontroller for projects requiring less memory space and it has fewer GPIO pins [18]. The pins include 14 digital input and output pins, six analog input pins, and two pins for serial communication [17].

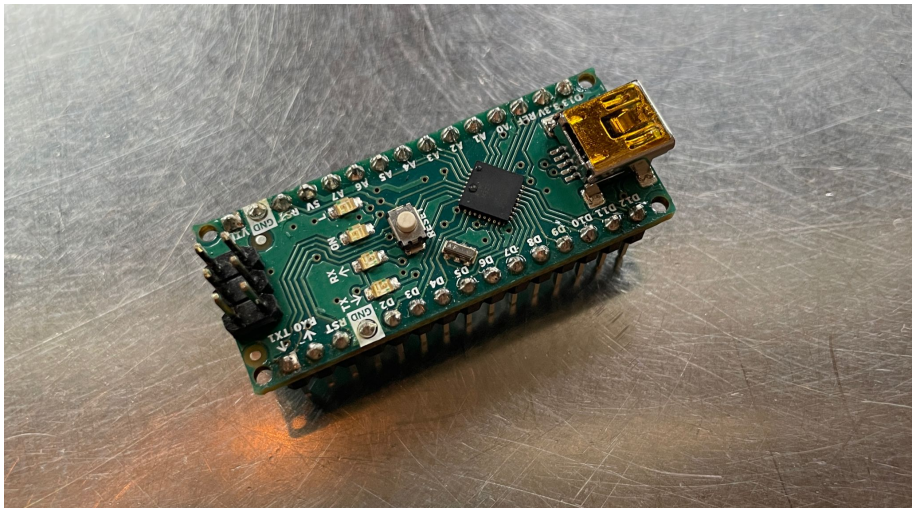


Figure 2.8: Picture of an Arduino Nano.

2.2.3 Actuator: Analog feedback Servomotor

An analog feedback servomotor is a closed-loop control system where the position of the output provides feedback to the microcontroller [19]. In a closed-loop system the feedback signal from the encoder or position sensor can be sent to the control unit to, for example, control and adjust to maintain desired speed or position.

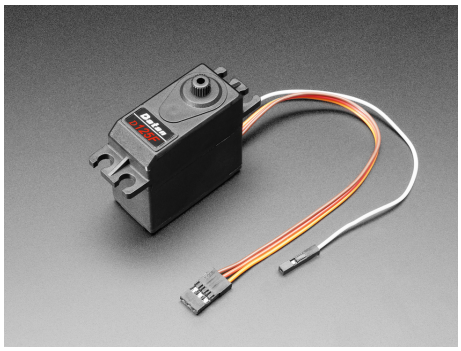


Figure 2.9: Picture of an Analog feedback Servomotor.

In total, the servomotor has four wires. The red and brown wires are used for the power supply where the red is connected to positive polarity and the brown to negative polarity or ground. Orange is an input for the control signal to the servomotor. This wire is connected to the microcontroller to generate a pulse width modulated (PWM) signal to control the position or movement of the servomotor [3]. Lastly, the white wire is the position feedback signal wire that connects to the output for the position or angle signal from the integrated encoder. This wire can read an analog input such as those on an Arduino Nano and send a signal after reading the position or angle from the servomotor and forward the signal to the control system. This information can also be used to record and replay the motions of the servo [19].

2.2.4 Absolute Encoder

An encoder is a type of position sensor and is a device that is essential to get precise information for the control of a manipulator, such as a robotic arm. It also provides the controlled object, such as a servomotor to be able to get feedback about its current position or speed [20].

There are two types of of encoders: incremental and absolute. An incremental encoder does not know its absolute position and works by finding a zero point or a reference position, which it has to return to upon startup or reset. When it has returned to its reference point, the encoder resets and starts to track movements from there [21].

Absolute encoders on the other hand can provide information about the absolute position. Most absolute encoders are composed of a light emitter diode, a transparent plastic covering the lens, a disc code pattern mounted on an encoder shaft, a Charge Coupled Device (CCD) matrix sensor, and the processor unit with code memory [20]. Relative to when the encoder shaft rotates, the CCD sensor captures the images of the pattern of the disc. The processor then reads the binary code on the pattern disc to return an angle of the shaft and then stores the information in the memory. For most absolute encoders there is only a single bit changing during the process of changing from one position to another [20].

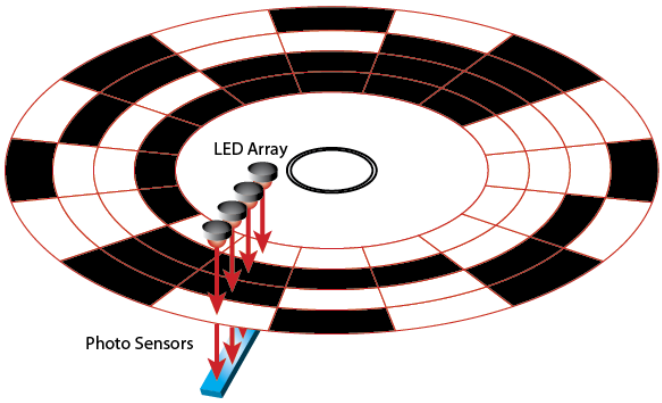


Figure 2.10: Code pattern disc of absolute encoder.

Figure 2.10 illustrates what an absolute encoder disc could look like. As the disk rotates, the sensors will output high or low, depending on whether they detect black or transparent sections. This generates a unique binary code at each position.

2.3 Robot Operating System (ROS2)

Robot Operating System (ROS) 2 is an open-source framework for robot software development that provides tools and libraries for building robot applications [22]. It is commonly used in robotic applications and allows for modularity and reusability through its key features. The features include the concept of creating nodes which are computational units that can perform a specific task. The nodes communicate and exchange information through messages that typically are data types, such as integers, floating points, booleans, strings, etc [23]. The nodes use a publish and subscribe message system, allowing the nodes to transmit and receive the messages by publishing to a topic and subscribing to a topic of interest [24].

To facilitate organized and collaborative development, the ROS software system allows for the creation of packages. These ROS packages contain XML files that describe relevant information such as the package's name and other dependencies that make it unique for the intended purpose. They can also contain other essential files such as launch files and source code files.

The packages consist of different software for specific tasks to contribute to the whole robotic system. For example, one package could be for sensor drivers while another package could be for the control logic. Essentially packages allow for manageability and flexibility.

For development, the framework can be programmed with the most common programming languages including Python and C++ [25].

2.4 AUXIVO

AUXIVO is a company that focuses on developing and manufacturing exoskeletons. The goal for the company is to create products that can support industry workers and users in their daily activities and provide them with the right support, for example when they have to conduct physically demanding tasks. This is to reduce the risk of injury and improve their well-being [26].
At AUXIVO, their team consists of designers, researchers, and engineers who have the experience and the right expertise in the development of biomechanics and wearable exoskeletons. Today, they have developed a considerable amount of exoskeletons ranging from occupational exoskeletons, which are designed for different types of tasks, to educational ones, which are designed to facilitate learning. Examples of the occupational ones are the LiftSuit 2 which is a lightweight exoskeleton that supports the muscles in the hip and back when conducting lifting motions from below hip level. One of the educational exoskeletons they have created is the EduExo 2 which is an elbow exoskeleton. It comes as a kit that contains all parts needed to assemble the exoskeleton including the physical parts of the exoskeleton and the hardware. A handbook is also provided to introduce the exoskeleton's technology as well as to give guidance in the assembly.

Figure 2.11: EduExo 2 by AUXIVO.

3
Methodology

This section will provide the approach for the project, describing each step taken to design and develop the hardware and software of the elbow exoskeleton.

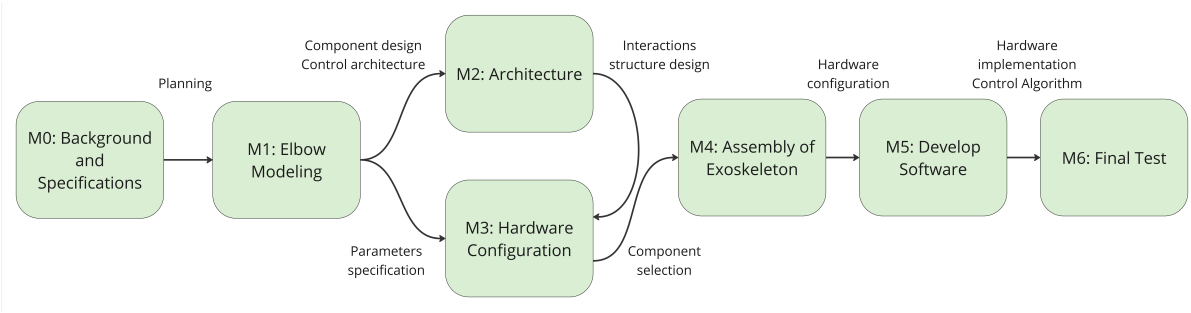


Figure 3.1: Block diagram of the project structure

3.1 Background and Specifications

To ensure a systematic and structured project, the first step was to create a hypothetical work-time flow with different milestones, each with specific tasks and a specific time frame. This was done in Chalmers GitLab and was created to get information on what steps in the process had been completed and what steps were needed to move forward toward the end goal. Figure 3.1 presents a block diagram of the milestones depicting how the work is connected and the most important tasks required to move on to the next milestone.

3.2 Architectures

The architectures were created to understand and visualize the connections within the system, as well as to facilitate easier identification of debugging and collaboration. From the calculations and designs made above, the architectures as follows were created.

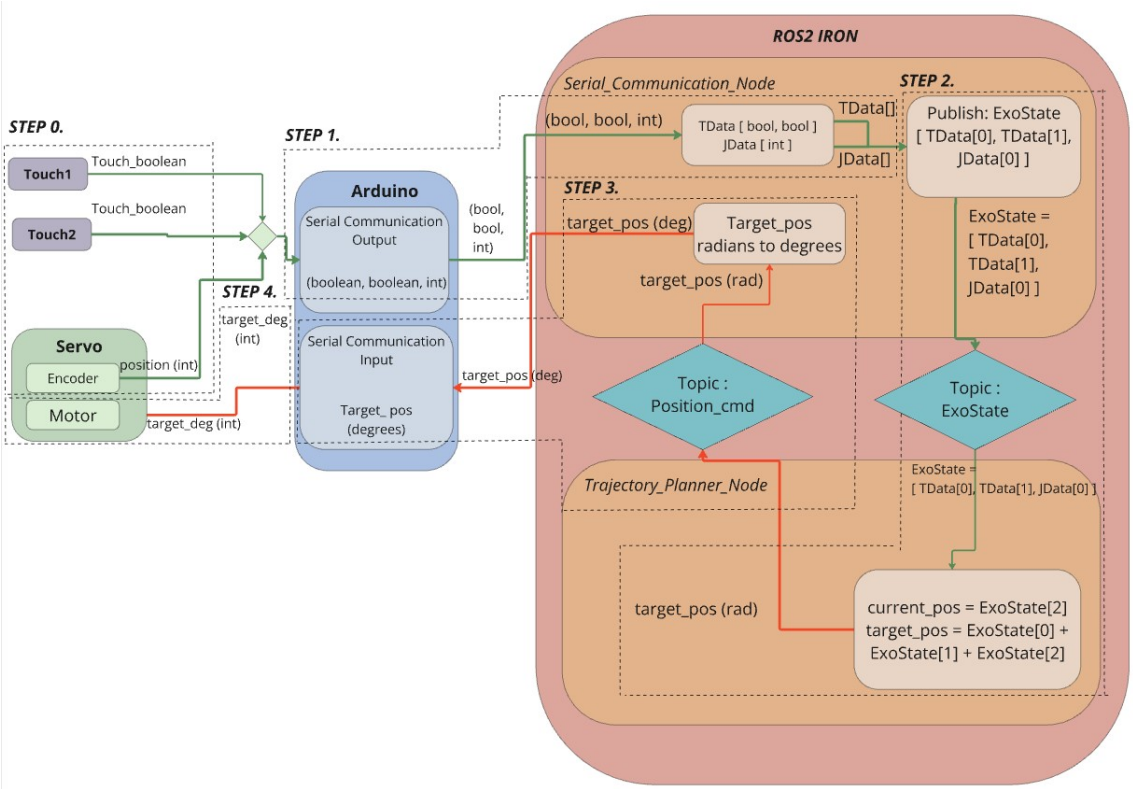


Figure 3.2: Framework architecture.

Figure 3.2 provides a detailed diagram of the system's communication between the ROS2 architecture and the hardware components. It illustrates the data flow and interaction patterns between the ROS2 nodes, the microcontroller, and other hardware elements. The framework architecture provides a clear overview of the system and its structure.

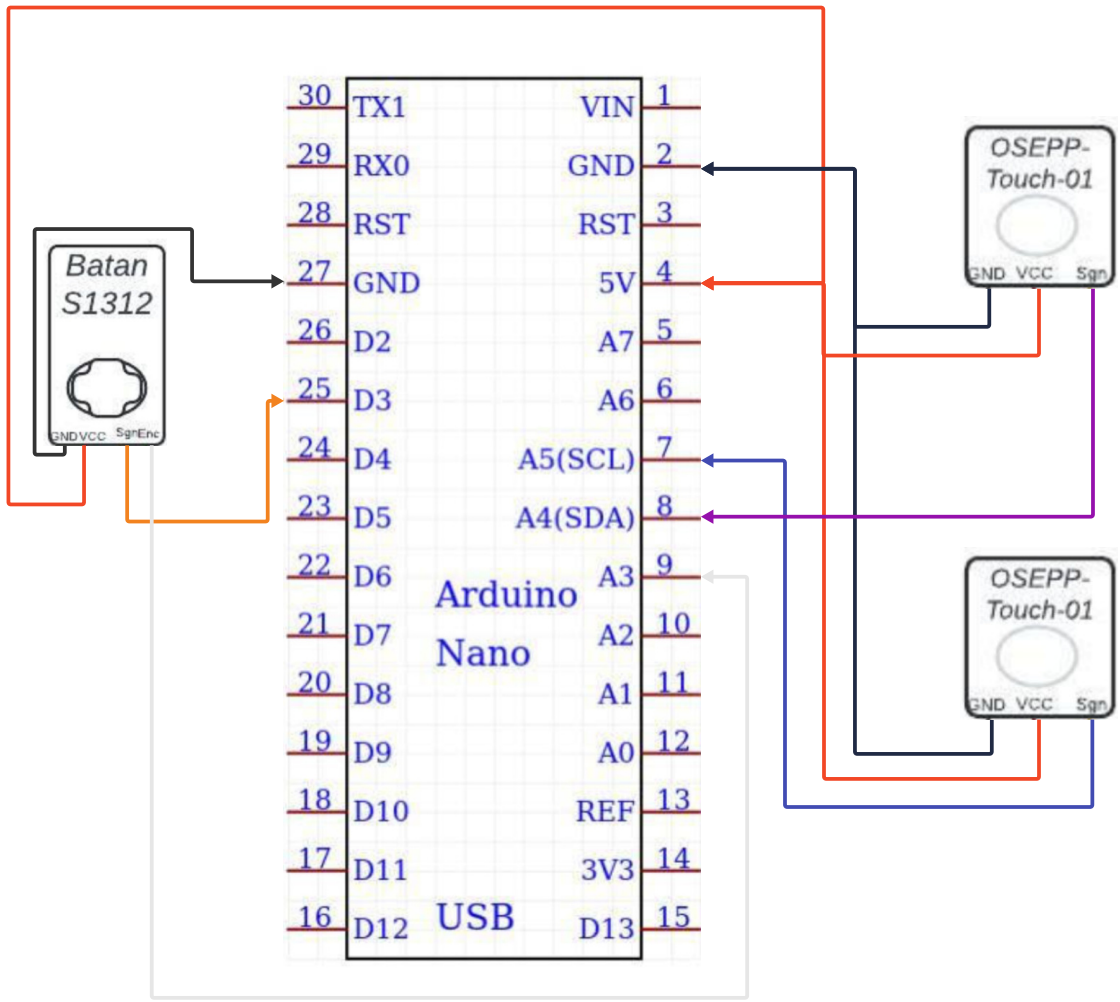


Figure 3.3: The electrical architecture for electronic components.

Figure 3.3 illustrates the wiring schematic that details the connections made between the microcontroller (Arduino Nano), touch sensors (OSEPP-Touch-01), and actuator (Batan S1312). It also highlights the specific pin assignments for each component. The electrical architecture provides a clear view of how the hardware components interact to provide functionality.

3.3 Hardware selection and configuration

The design of the exoskeleton hardware was based on the AUXIVO EduExo 2, which was bought pre-made. The decision to buy a pre-made exoskeleton was to be able to focus more on the software and to get the communication between the ROS2 and Arduino to function, rather than focusing on the design and software, which would take a longer time. The electronic components for the project were primarily selected based on compatibility with the pre-made exoskeleton as well as performance, cost, and set requirements. To be able to create a lightweight, non-bulky, and highly functional system, the components mentioned in the theory chapter were selected.

For the microcontroller it had to be able to get programmed to read the signals from the encoder that was inside the actuator's housing, and for motor control. The Arduino Uno, Arduino Nano, and Raspberry Pi were considered. After researching the differences between the three of them and considering the functionalities, the Arduino Nano was selected because of its communication interface, its processing power, and the fact that it had sufficient pins needed while still maintaining a compact size and relatively cost-effective. The other microcontrollers had the properties of the Arduino Nano, however, the size and the price were the main deciding factors. Therefore, the Arduino Nano was deemed most suitable.

The servomotor was pre-included in the package with the hardware for the exoskeleton. The motor featured a built-in encoder and the component was considered suitable as the exoskeleton was for an upper limb, therefore already providing the suitable strength.

Since the movements depended on user input, two simple yet functional tactile sensors were selected to facilitate this interaction. Their simple design and suitable functionalities made them easy to integrate into the system. The tactile sensors were also chosen because of their ability to be responsive and precise. Since they were also quite lightweight and compact, positioning them could be done easily and comfortably.

3.4 Assembly

The assembly could be separated into two sections; first, the assembly of electronic components, and second, the exoskeleton.

3.4.1 Electronics

The electronics assembly consisted of soldering to allow the connection of all electronic components. The first step was to connect the servomotor to the Arduino through the terminal adapter, which provides connectors for the wires to be inserted and connected through it. Using wires with the same color as the cables on the servomotor (brown, white, orange, and red wire). The wires were then inserted into the terminal adapter. The following describes the connections made for the motor to the terminal adapter:

- White wire to terminal adapter port A3.
- Orange wire to terminal adapter port D3.
- Red wire to terminal adapter port 5V.

When the motor wires were in place, the next step was to solder the wires to the two touch sensors, three wires each. The following describes the connections made for the touch sensors to the terminal adapter:

- Touch sensor 1 to terminal adapter port A4.
- Touch sensor 2 to terminal adapter port A5.

The motor and the sensors' ground wires were soldered together and lastly connected to the GND terminal adapter port.
With the motor and sensors connected, the terminal adapter was then placed inside the upper arm cover and screwed to place using two M3x8 screws. For a better overview of the connections made, see Figure 3.3 for the electrical architecture of electronic components. See figure 3.3 for visualization of the connections.

Finally, the Arduino Nano was plugged into the terminal adapter in such a way that the numbers on the Arduino matched with the terminal adapter. The two touch sensors were taped to the upper arm cover for better management. Arduino was then connected to the PC through a mini-B USB cable, powering the Arduino and allowing for program uploading.

3.4.2 Exoskeleton

The assembly of the exoskeleton was first constructed with all the pre-made mechanical parts that consisted of five plastic parts: three plastic cuffs with textile cuffs on the inside, and a forearm- and upper arm cover. These were assembled to create the exoskeleton.
The first step was to use six threaded inserts and one M3x8 screw together with a 2mm hex key. The threaded inserts were placed in the upper arm cover and then fully embedded in the plastic covers using the M3x8 screw and the hex key. This was done to ensure that the motor and other electronic components could be installed on the cover. The installment of the servomotor was done with four M3x8 screws. The motor was placed in the upper arm cover where its wires were placed outside, and the shaft was oriented to the long end of the cover. Then it was fixed onto the cover with the screws.
With the motor installed, we then proceeded to assemble the forearm cover with the upper arm cover, forming the exoskeleton. This assembly can be seen in Figure 2.1. For this part, four M2 wood screws and an included round motor adapter were used along with a screw for the servo. The motor adapter was first connected to the forearm cover using the four M2 wood screws. Then, it was connected to the forearm cover using the included screw. The cuffs were clipped on and off in the covers depending on when they were needed. The final results of the assembly can be seen in Figure 2.1 and Figure 2.2.

3.5 Software development

The following section involves the development process of the software for the performance and functions of the exoskeleton.
The software development was divided into two parts: the first part was developing the software for the Arduino Nano, using the Arduino Integrated Development Environment (Arduino IDE), which utilizes the C++ programming language. This software was responsible for low-level hardware interactions, such as reading sensor inputs and controlling the servomotor.
The second part focused on creating software in ROS2 Iron, a version of the Robot

Operating System (ROS). This part was developed to manage the high-level functionalities such as the communications between the system's components and coordinating data flow.

3.5.1 Arduino framework

All the hardware components including the touch sensors, the encoder, and the servomotor, were declared at the beginning of the code.

The program began by setting the servo's starting position to align with the upper arm in its vertical position, ensuring proper alignment. Movements had to be restricted to 90 degree flexion for the right arm to implement a limit of range of motion.

Once the system was initialized, programming of the serial communication setup could get started to facilitate data exchange between the Arduino and ROS2 environment. Serial communication was programmed to read the input data from the touch sensors and the current position from the servomotor to send to ROS2. The data was then received and processed in ROS2 where, depending on the input of the sensors, a new commanded position was calculated. The new position was sent back to the microcontroller, allowing the servomotor to either increment (touch sensor 1) or decrement (touch sensor 2) its position. The servomotor was then moved to the new commanded position accordingly. The final position was controlled through comparison to the current position. An example of this process can be seen in the example depicted in Figure 3.4.

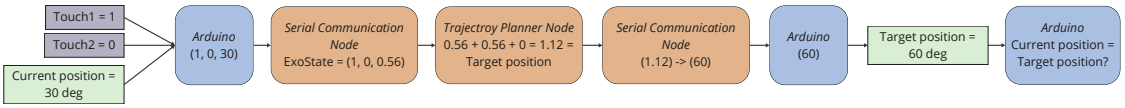


Figure 3.4: Example data for an increased joint position from 30° to 60°.

3.5.2 ROS2 framework

The ROS2 communication framework was structured into two primary nodes, each with its own tasks: one for serial communication and the other for trajectory generation. Figure 3.2 illustrates the system's communication architecture, providing a clearer representation of how the nodes interact within the ROS2 system. Below is a description of how each node communicates and integrates within the system.

Serial Communication Node

The serial communication node was the first developed node and was responsible for interfacing with the serial port. Its primary function was to facilitate seamless data exchange via serial communication, handling data transmission to and from the Arduino Nano. It served as a bridge, managing communication between the ROS2 system and the hardware (Arduino).

After testing and verifying the node's capability to send and receive information from the Arduino and ensuring reliable communication, the node was configured to operate in a limited mode, focusing on its key tasks. The serial communication node was configured as a publisher, tasked with processing the information received from the Arduino and packaging the information into a topic named $exo - state$, to then publish it into the ROS2 network. The processed data contained the data of the current position of the servomotor and the input from the two tactile sensors.

The topic was then sent and received by the Trajectory Planner Node, configured as a subscriber, and used this information for trajectory calculations and planning. Before the Serial Communication Node publishes the data, the node performs several critical functions. These included data conversion to convert the position data from degrees to radians to ensure compatibility with ROS2 conversions for angular measurements, and input processing, where it evaluated the input from the tactile sensors to determine whether to increment or decrement the joint position.

From the Arduino serial communication output, the Serial Communication Node received the data from the tactile sensors as boolean values and the current position data from the servomotor encoder as an integer. These values were structured into Tdata $[x, y]$ representing the tactile sensor inputs, and Jdata [z] representing the current position. This processed data was then packaged and published in the topic called $exo - state$ as an array $[x, y, z]$, to be read in the Trajectory Planner Node. In addition to publishing $exo - state$, the Serial Communication Node subscribes to the $position - cmd$ topic published by the Trajectory Planner Node. This topic contains the new target position denoted by $target - pos$ for the exoskeleton in radians. Upon receiving a target position, the node converts the position data from radians back to degrees before being transmitted via the serial communication input port to the Arduino. The Arduino then processed the information and sent the final desired position command denoted as $target - pos$ to the servomotor, updating the actuators in real-time, resulting in the intended movement.

Trajectory planner Node

The Trajectory planner node was configured as a subscriber to the $exo - state$ topic. Initially, the position was set to 0. Upon receiving data from $exo - state$, the node updates the current position based on the received value and converts the position from degrees to radians denoted as $JData$. The target position was then updated by adding the tactile sensor inputs, denoted as $TData$, which either incremented or decremented the joint position ($target - pos = JData + TData$).
Once the desired target position was determined, the information was published on the topic named $position - cmd$. This data was then received by the Serial Communication Node. These messages dictated the desired trajectory for the exoskeleton. The primary function of this node was to create a predictable, linear trajectory for the exoskeleton's movement. For example, transitioning from 0 degrees to 90 degrees in defined increments or decrements.

3.6 System testing

Testing was an essential part of the development process. It ensured that each component functioned correctly both individually and as part of the integrated system. This section outlines the tests made for the hardware, software, and their integration.

3.6.1 Mechanical and Electronics testing

After completing the assembly of the EduExo, an alignment check was performed. This was simply done by putting the exoskeleton on the arm and aligning it with the elbow joint to avoid misalignment issues. Then, a range of motion test was done by flexing and extending to ensure that it felt smooth, confirming that the mechanical components were properly assembled and functioning without resistance.

To test the electronic components, simple Arduino programs were written to verify functionality. The first program was used to test the servomotor. The program was written to read the position feedback from the servomotor's potentiometer. The output was used to measure the elbow joint angle of the exoskeleton. This was to make sure that it was correctly coupled and installed.

A separate program was written to test the tactile sensors. The touch sensors were tested individually, with the program outputting a 1 when touched and a 0 when untouched, confirming correct functionality.

3.6.2 Hardware verification and Serial Communication Node testing

The hardware components (tactile sensors, servomotor, and Arduino) were tested individually using Arduino programs to confirm their functionalities without ROS2. When correct operations were confirmed, the ROS2 system could be developed.
The Serial Communication Node was the first ROS2 component to be tested, as it served as a bridge for communication between the hardware and the ROS2 system.

3.6.3 ROS2 node testing

The software testing phase focused on validating the functionalities of the ROS2 nodes and their interactions with the hardware. First, the test involved sending a simple command from ROS2 to the Arduino to verify functionality.
Moving forward, after confirming basic functionalities, the ROS2 nodes were developed and tested. The serial communication node was tested on its ability to publish the $exo - state$ topic. The node's subscription to the $position - cmd$ topic was verified by sending different target position commands and confirming the executions.
Following, a trajectory planner node was developed and tested. It was first tested on its ability to subscribe to the *ExoState* topic and compute desired positions based

on different data inputs from the tactile sensors. Then, its ability to publish the accurate position command to the Serial Communication Node was verified.

3.6.4 System Validation

The final phase of testing involved validating the entire system to ensure that it met the desired performance. Due to regulations, the test was done with a user wearing the device. It was tested to ensure that it could reach the desired range of motion between 0 degrees (full extension) to 90 degrees (flexion) in defined increments or decrements, and provide support for the user.

4

Experimental Validation and Results

The ROS2-based control system demonstrates serial communication capabilities, handling data transmission between the Arduino Nano and the ROS2 nodes. The experimental validation confirms that the system can respond to user input and provide an accurate movement to the desired positions.

The hardware components, including the tactile sensors and actuator, and Arduino operated accordingly under coordinated software commands. The system provided basic support for the elbow flexion/extension within a 0-90 degree range, validating the integration between the software and hardware.

This section presents the results of the developed elbow exoskeleton system, highlighting its performance and the efficacy of software and hardware integration through experimental validation.

4.1 System communication

The final communication architecture of the system is divided into two parts and consists of five sequential steps. This structure provides a clear view of the workflow for the communication, starting from step 0 and progressing to step 5, integrating both the hardware components and the ROS2 system.
Figure 4.1 illustrates the architecture of the framework, showing the process and structure of the system, including the interactions between hardware (Arduino Nano, tactile sensors, and actuator) and software (ROS2 nodes and topics). The full Arduino and ROS2 code is provided in the appendix for reference.

4.2 Active test

To observe the behavior and ensure the system's functionality, an active test is performed. The purpose of the test is to collect data from the inputs of the capacitive sensors to visually see the measured response, testing if the functionality is accurate. This is to be able to evaluate its performance.

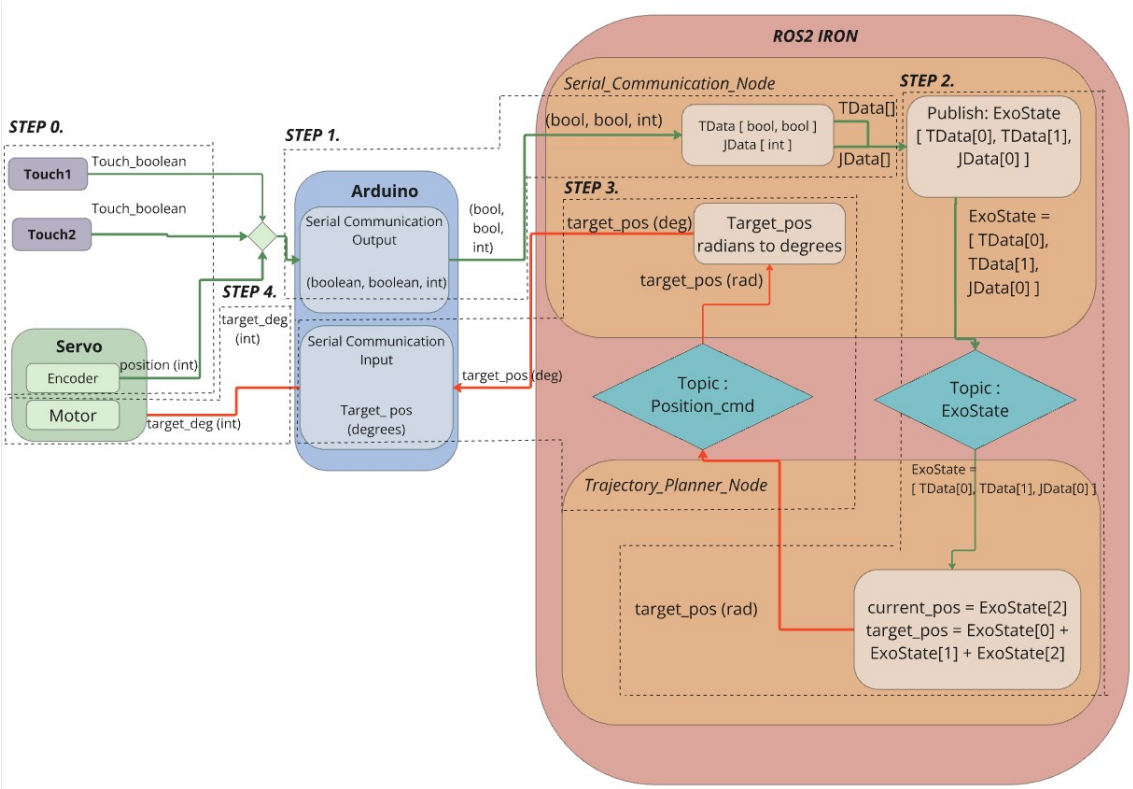


Figure 4.1: Communication architecture.

During the active test, incremental movements are executed to flex the elbow from an extended position. The upper tactile sensor, labeled *Active increase* in Figure 4.2, is activated by user input. The movement is performed in increments of 30 degrees, allowing the system to achieve a range of motion between 0 and 90 degrees, and as illustrated in Figure 4.2, 30 to 90 degrees. Decremental movements are illustrated in Figure 4.3, where the lower tactile sensor, labeled *Active decrease* is activated by user input. The range of motion is from 90 to 0 degrees, or as illustrated, 60 to 0 degrees with decrements of 30 degrees.
A more comprehensive demonstration of the movement, including both incremental and decremental actions, is provided in the video linked in the appendix. This video offers a complete view of the system's functionality in action.

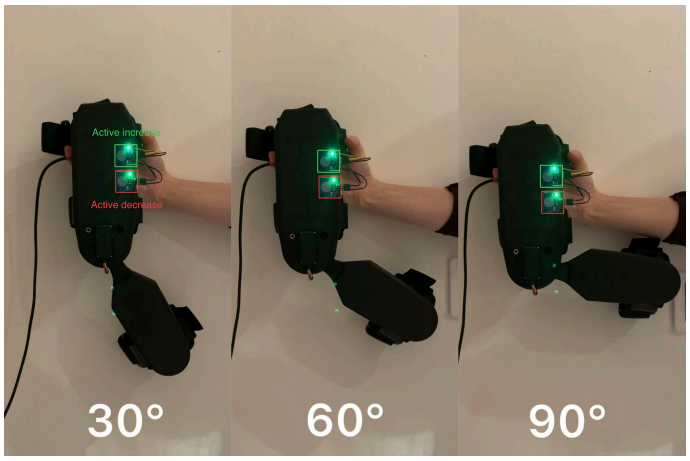


Figure 4.2: Incremental elbow flexion from 30 to 90 degrees in 30 degree steps.

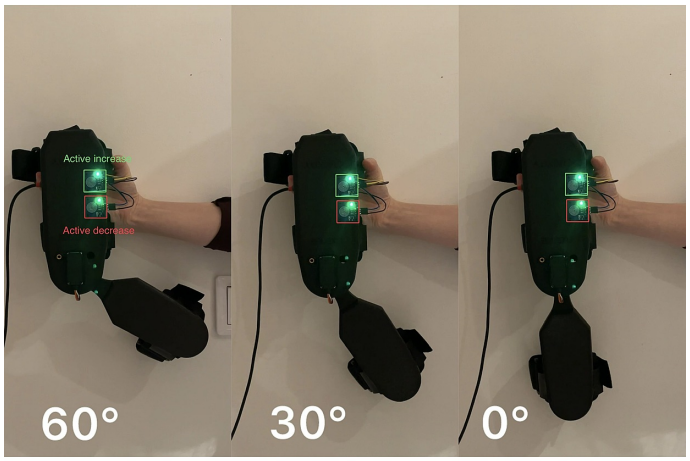


Figure 4.3: Decremental elbow extension from 60 to 0 degrees in 30 degree steps.

Figure 4.4 illustrates the relationship between the joint position command (position-cmd), tactile sensor inputs (touch-inc and touch-dec), and joint position (current-position). The figure consists of four subplots, each representing different aspects of the system's response. The first subplot represents Touch Sensor 1, responsible for incrementing the position, and is denoted by *tdata*. The second subplot rep-

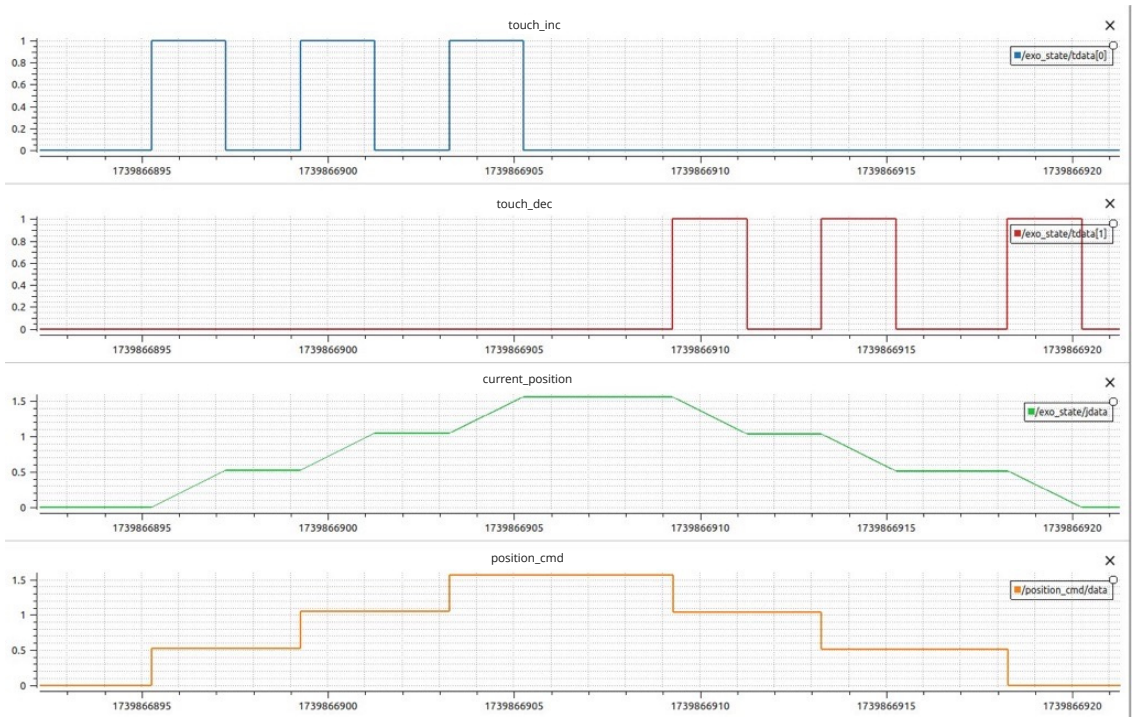


Figure 4.4: Plots of the data from active test.

resents Touch Sensor 2, responsible for decrementing the position, also denoted by *tdata*. The third subplot displays the current joint position, denoted by *jdata*, which changes depending on the tactile input. The fourth subplot shows the commanded position denoted by *data*, which follows the trend of *jdata*.

At a moment where the first subplot (touch-inc) for tdata remains inactive (0) while the second subplot (touch-dec) shows that tdata is active (1), a decrement action is triggered. This results in a linear decrease in the third subplot (current-position) *jdata* starting immediately after touch-dec is triggered, and a simultaneous decrease in the fourth subplot (position-cmd) data following touch-dec with a slight delay. Similarly, when touch-inc is set to active (1) instead, an increment action occurs where both current-position and position-cmd increases. This reflects the system's response to the increment command. This confirms that position-cmd closely follows the data of the exo-state data, adjusting it in response to the tactile sensor inputs, ensuring predictable movement.

5
Discussion

In this section, the challenges encountered during the project will be discussed, along with potential improvements that could enhance the system in future work.

5.1 Challenges

The most significant challenge, and the one that took the most time, was coding using Python and C++ while developing software using the ROS2 framework. This was all new to us, as we had no prior experience with these tools. ROS2 proved to be more complex than we had anticipated. There were many challenges as we developed the ROS2 system. For example, creating packages with nodes and integrating nodes within the same package was achievable, but enabling communication between all of them was more difficult. As a result, we had to scale back the work and focus on two functioning nodes. Despite the setback, the process was educational, though time-consuming.

Due to the servomotor's torque limitations, the system's ability to provide significant support was restricted, and it could only assist with basic movements. This limitation was identified late in the development process. While a stronger motor was purchased to try and solve the issue, integrating it required significant changes to the code. This was due to the new motor using PWM (Pulse Width Modulation) for control, while the initial motor operated based on position through angle, making the transition more complex. Additionally, integrating the new motor would not guarantee that the desired range of motion would be achieved. Given these challenges and the time constraints, we decided to set the new motor aside and continue using the initial one.

The project deviated from the initial plan that was established at the beginning due to a shift in focus during the design of the exoskeleton. Originally, the goal was to design and fabricate the physical parts of the exoskeleton. However, to focus on software development over the mechanical design, a decision was made to purchase a pre-built exoskeleton. This resulted in more time-consuming work than initially anticipated.

The change in scope resulted in a more software-based project. This enables more focus on learning and working with new programming languages, such as Python, and C++. However, this also introduced additional challenges with additional tasks,

such as the implementation of a modular and scalable ROS2 communication framework.

5.2 Potential improvements

A potential improvement to the exoskeleton would be the addition of enhanced safety features. Currently, the servomotor is too weak to be of any significant danger, but incorporating additional safety measures would make the system more secure. At present, the primary safety mechanism in the system is from the mechanical parts, where the lower link is physically restricted from moving further than 90 degrees. However, further improvements could be implemented to enhance the safety. For example, software-based safety measures could be added to prevent unintended actions triggered by the touch sensors. Additionally, instead of only relying on the software, a physical emergency stop could be integrated to make the motor stop when activated, for example, a power switch.

Another potential improvement could involve redesigning the exoskeleton. As previously described, the current design of the exoskeleton has a limitation on the range of motion, restricting the motor to a maximum of 0-90 degrees for arms. By altering the design of the exoskeleton, it could be possible to achieve a wider range of motion, such as 0-180 degrees, thereby enhancing its functionality.

Currently, the system is powered via a USB cable connected to a laptop. To achieve portability, the exoskeleton could utilize a safe power supply, such as a battery. A rechargeable battery would be an even better solution.

6
Conclusion

In this project, we defined the following research questions:

- **What type of user interface will be used?** The project utilizes two tactile sensors as its user interface, allowing the user to decide whether to flex or extend the arm. Interaction is achieved by pressing the sensors, and visual feedback is provided, as shown in Figure 4.3. The interface between the hardware system and ROS2, detailed in Figure 4.1, functions effectively using serial communication and ROS2 nodes.
- **Can the system provide accurate positioning based on user input?** The project has resulted in a safe and user-friendly system capable of providing accurate positioning based on user input. The system operates smoothly and accurately within the 0 to 90-degree range. It can assist with basic movements of the upper limb, incrementing or decrementing as the user prefers.
- **Is it possible to create an interface between the hardware system and the ROS system?** An interface was successfully implemented using a communication bridge, specifically the serial communication node. Testing confirmed seamless data exchange and reliable communication between the hardware and ROS2 system. The integration between the Arduino with ROS2 demonstrated the system's capability to synchronize tasks effectively, with hardware components, including sensors and actuators, responding accurately to the software commands.
- **How can a modular software architecture be designed?** The modular design of the system was validated through the integration of separate nodes within the ROS2 architecture. This modularity facilitates easier maintenance and upgrades, allowing for future enhancements without disrupting existing functionalities. The system is scalable in the way that additional hardware can be integrated into the system without disrupting existing functionalities, provided the new components follow the existing communication framework.

To conclude, this project provides a foundation for assisting users with basic movements and has the potential to improve mobility for individuals with limited upper limb functionality. The servomotor's limited torque ensures minimal risk of harm, as the user can stop the movements manually, and a mechanical stop can prevent it from going beyond the 90-degree limit. Position data is collected in ROS and can be plotted for verification, providing feedback to improve the system's performance.

However, while the system provides basic support, fully compensating for muscle

function would require further improvements. Enhancements such as a motor with higher torque and additional safety features would need to be implemented to enhance performance and address muscle fatigue.

Bibliography

[1] World Health Organization (WHO). "Musculoskeletal health". Accessed: May 10, 2024. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/musculoskeletal-conditions

[2] M. H. Rahman et al., “Development of a whole arm wearable robotic exoskeleton for rehabilitation and to assist upper limb movements”, Robotica, vol. 33, no. 1, pp. 19–39, 2015. doi:10.1017/S0263574714000034

[3] Auxivo AG, EduExo - the robotic exoskeleton kit handbook and tutorial, 3rd ed., 2022. Available: https://www.eduexo.com/

[4] R.A.R.C. Gopura, D.S.V. Bandara, K. Kiguchi, and G.K.I. Mann, "Developments in hardware systems of active upper-limb exoskeleton robots: A review," Robotics and Autonomous Systems, vol. 75, no. Pt B, pp. 203–220, 2016, doi: 10.1016/j.robot.2015.10.001.

[5] M. A. Gull, S. Bai, and T. Bak, "A Review on Design of Upper Limb Exoskeletons," Robotics, vol. 9, no. 1, p. 16, 2020. doi: 10.3390/robotics9010016.

[6] S. Bai, G. S. Virk, and T. Sugar, Wearable Exoskeleton Systems: Design, Control and Applications. London, UK: Institution of Engineering and Technology, 2018. doi: 10.5555/3265106.

[7] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, Robotics: Modelling, Planning and Control. London: Springer, 2009, doi: 10.1007/978-1-84628-642-1.

[8] A. Stevens, "Forward Kinematics," in Modeling, Motion Planning, and Control of Manipulators and Mobile Robots, Y. Wang, Ed., A. Lunia, A. Stevens, C. Holt, R. Morgan, J. Norris, S. Poyyamozhi, and Y. Zhong, Eds. Clemson University.

[9] J. Iqbal, R. ul Islam, and H. Khan, "Modeling and Analysis of a 6 DOF Robotic Arm Manipulator", Canadian Journal on Electrical and Electronics Engineering, vol. 3, no. 6, pp. 300, July 2012.

[10] M. W. Spong, S. Hutchinson, and M. Vidyasagar, "Robot Modeling and Control", John Wiley Sons, 2006.

[11] N. Paredes-Acuña, N. Berberich, E. Dean-León, and G. Cheng, "Tactile-Based Assistive Method to Support Physical Therapy Routines in a Lightweight Upper-Limb Exoskeleton," IEEE Transactions on Medical Robotics and Bionics, vol. 4, no. 3, pp. 541–549, Aug. 2022, doi: 10.1109/TMRB.2022.3188429

[12] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, Robotics: Modelling, Planning and Control, 1st ed., Advanced Textbooks in Control and Signal Processing. London, UK: Springer-Verlag, 2009. doi: 10.1007/978-1-84628-642-1.

[13] Farnell, "Touch sensors". Accessed: May 16, 2024. [Online]. Available: https://se.farnell.com/sensor-touch-sensor-cap-res-technology

[14] OSEPP, "Touch Sensor Module". Accessed: May 16, 2024. [Online]. Available: https://www.osepp.com/electronic-modules/sensor-modules/80-touch-sensor-module

[15] Z. Baharav and R. Kakarala, "Capacitive touch sensing: signal and image processing algorithms", Proc. SPIE 7873, Computational Imaging IX, vol. 7873, no. 78730H, Feb. 2011, doi: 10.1117/12.876714.

[16] RS Components, "A Complete Guide to Microcontrollers". Accessed: May 15, 2024. [Online]. Available: https://uk.rs-online.com/web/content/discovery/ideas-and-advice/microcontrollers-guide

[17] Arduino.cc, "Arduino Nano". Accessed: May 15, 2024. [Online]. Available: https://store-usa.arduino.cc/products/arduino-nano

[18] E. Ashley. "What is Arduino Nano? A Getting Started Guide". RS-online.com. Accessed: May 15, 2024. [Online]. Available: https://www.rs-online.com/designspark/what-is-arduino-nano-a-getting-started-guide

[19] B. Earl. "About Servo and Feedback". Adafruit.com. Accessed: May 15, 2024. [Online]. Available: https://learn.adafruit.com/analog-feedback-servos?view=all

[20] Dziwiński, T., "A Novel Approach of an Absolute Encoder Coding Pattern," IEEE Sensors Journal, vol. 15, no. 1, pp. 397-401, Jan. 2015, doi: 10.1109/JSEN.2014.2345587.

[21] Wang, H., Wang, J., Chen, B., Xiao, P., Chen, X., Cai, N., Ling, B. W. K. (2015). Absolute optical imaging position encoder. Measurement, 67, 42-50. doi.org/10.1016/j.measurement.2015.02.028

[22] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," Science Robotics, vol. 7, May 2022.

[23] J. Iqbal, M. U. Islam, and H. Khan, "Modeling and analysis of a 6 DOF robotic arm manipulator," Canadian Journal on Electrical and Electronics Engineering, vol. 3, pp. 300-306, 2012.

[24] X. He, H. Sato, Y. Okumura, and T. Azumi, "TILDE: Topic-Tracking Infrastructure for Dynamic Message Latency and Deadline Evaluator for ROS 2 Application," in Proceedings of the 2023 IEEE/ACM 27th International Symposium on Distributed Simulation and Real Time Applications (DS-RT), Singapore, 2023, pp. 1–9, doi: 10.1109/DS-RT58998.2023.00010.

[25] L. Joseph, Learning Robotics Using Python: Design, Simulate, Program, and Prototype an Autonomous Mobile Robot Using ROS, OpenCV, PCL, and Python, 2nd ed. Birmingham, U.K. Packt Publishing, 2018. [Online]. Available: https://research.ebsco.com/c/lu54te/search/details/gd5cc2wfff?limiters=FT1%3AY&q=ros%20and%20python

[26] Auxivo, "About us". Accessed: May 17, 2024. [Online]. Available: https://www.auxivo.com/about

A

Appendix

Link to complete code repository: `https://github.com/Fanny1337/SmartElbowExo.git`

Link to comprehensive demonstration of the exoskeleton functionalities: `https://youtube.com/shorts/U9X3MglCPU8?si=SUSVzhdu2F3j3Wpc`

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY