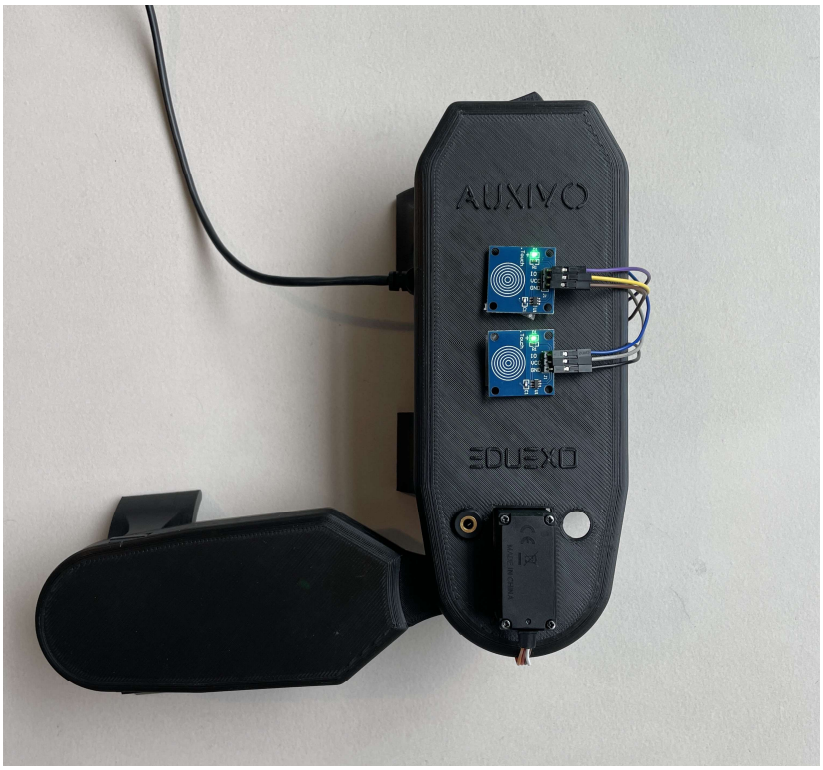
CHALMERS



Exploring and testing of orthopedic exoskeletons

Development of software for an elbow rehabilitation exoskeleton for more dynamic control of movement

Bachelor's thesis of Machine Engineering

Christopher Le Veau
Amir Mahdavi

DEPARTMENT OF MACHINE ENGINEERING

CHALMERS UNIVERISTY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

ACKNOWLEDGEMENTS

This bachelor's degree was conducted by two engineering students in mechanical engineering (180 credits) at Chalmers University of Technology. The work was done under the Department of Electrical Engineering and is done for Chalmers University of Technology.

We would like to start by thanking our examiner and supervisor Emmanuel Dean who helped us with guiding this project and helping us understand the technical aspects of this project.

We would also like to thank Sophia Ha and Fanny Tingberg for their help in understanding how the exoskeleton worked and answering technical, software and hardware questions regarding Arduino and the exoskeleton.

ABSTRACT

Exoskeletons are a growing technology in both industry and healthcare. With the increasing number of cases of musculoskeletal disorders, exoskeletons can be a potential solution for the rehabilitation of people with such disorders.

The department of electrical engineering at Chalmers University of Technology, working with Sahlgrenska University Hospital, wants to develop the understanding and technology of exoskeletons. A previous student group of Chalmers developed an elbow exoskeleton for this task, which used the pre-contracted exoskeleton EduExo lite 2 from AUXIVO.

The goal of this project is to improve the movement of the previous student group's elbow exoskeleton, as its movement system was too unnatural and static. This was done by improving and developing a more dynamic movement system for the elbow exoskeleton, to further develop understanding and research of orthopedic exoskeletons for rehabilitation, without introducing new hardware components or replacing the existing hardware components provided by the group.

Developing the movement system required first testing the learning the response time between the microcontroller and feedback servomotor, as well as the rotation speed and resolution of the feedback servomotor. Spline and polynomial functions were used to better simulate the natural movement of a human arm, as it gives motion a gradual increase and decrease in movement velocity.

The resulting code made it a dynamic movement control system, where movement from any start and end point, results in a smoother and softer motion, where the duration of the movement can also be altered.

The result of this project provides good grounds for potential further development, which can lead to a dynamic modular movement system to provide ways easily adapted to a patient's specific needs.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

TERMINOLOGIES / ABBREVIATIONS

DOF – Degrees of freedom; the number of independent paths or axes of motions of a body, device or object.

PWM – Pulse width modulation; a signal control method in which analog signals are sent in cycles “on” and “off” times, where changing the ratio, or “width”, of “on” time to the total cycle time to dynamically control certain devices.

API – Application Programming Interface; sets of commands and functions used for programming hardware or software which connect and communicates between one another.

IDE – Integrated Development Environment; software applications that provides tools and facilities that assist programmers in developing software code.

I/O – Input/output; Pins or ports used by computers to communicate with other systems, by receiving (input) and sending out data signals.

1
INTRODUCTION

1.1 Background

Exoskeletons are a growing technology in both industry and healthcare. With the increasing number of cases of musculoskeletal disorders, exoskeletons can be a potential solution for the rehabilitation of people with such disorders.

Two students at Chalmers university of technology, Sophie Kimberly Ha and Fanny Tingberg, did a degree project where they developed a pre-engineered exoskeleton for the arm, specifically an AUXIVO EduExo 2 Lite [1]. The focus of the development was on the software so that it can assist with the movement of the user. The goal was to make the system modular and scalable, to ensure future work and to be able to adapt the system to different exoskeleton systems [2].

However, the exoskeleton has limitations with its function. The movement of the exoskeleton during use is currently not suitable for use in physiotherapy. Chalmers (Department of Electrical Engineering) wants to develop a movement method for exoskeletons that can be used in healthcare.

The work is being conducted at Chalmers University of Technology.

1.2 Purpose

The project goal is to improve the previous group's exoskeleton movement.
The angular movement of the exoskeleton elbow is currently quite "unnatural", "robotic" and "linear". If it is possible to change the movement to allow for more control, it would allow it to be more adaptable to different patients' orthopedic injuries.

The group will try to make the exoskeleton's arm move smoothly, so that both acceleration and deceleration are smooth and better adapted to the patient's needs. The movement of the previous exoskeleton did not meet the goal well enough. Starts and stops occur suddenly, which can mean a harmful load for an operated muscle. The new exoskeleton should have smoother and more gentle movements.

The exoskeleton should also be able to react to forces outside the system during its movement.

The pendulum is to make it so that you can easily turn off the exoskeleton and just let it go.

1.3 Limitations

- We are developing an already existing exoskeleton system. With that, we will not replace, manufacture or introduce new hardware or software not provided by the previous group, and that directly affects the performance of the exoskeleton of this project.
- Limitations that were set with the hardware from the previous work will apply to this project as well. [2]
- Tasks in the work that involve programming will be written in Python and C++ code.

- This work is a development of a previous project. The goal of the project was not to create a final product, but to explore and develop. This work follows the same boundary and should not result in a final product, but to develop it so that it can get closer to a final product.
- The exoskeleton that will be developed further only covers the elbow, not any other body part, and the amount of allowed angular movement of the exoskeleton is between 0 and 90 degrees.

1.4 Clarification of the question

- What frequency can the Arduino systems reach by reading, writing, and executing a command?
- How accurately should it be possible to configure the speed of the arm's movement with its dynamic model? What level of user-friendliness do you want to aim for?
- What is the maximum number of angle changes that the Arduino or servo can manage and what can be the minimum (delta t) for each angle change?
- What alternative solutions can be thought of to achieve the goals?
- How should the simulation of force resistance be performed?

2
THEORY

This part of the report provides an overview of the components and the technical background that underpins the orthopedic Exoskeleton system. This part will focus more on the components that are more relevant to the software of the system, and the hardware if relevant to the goals of the work.

2.1 The Exoskeleton

Exoskeletons are wearable robots that are worn externally on the body, which can have different forms depending on the body part they are made for, such as a full-body suit or leg/arm attachments [3]. They are designed to work with their users to assist or improve the functionality of their movable body parts, depending on what is required and the user's specifications [4]. They have a wide range of applications in their uses, such as mobility support for disabled people or improving range of motion to reduce physical stress on a person in areas where physically demanding work needs to be performed [5].

The exoskeleton for this project was the EduExo 2 lite, developed and manufactured by AUXIVO AG, a company that specializes in wearable exoskeleton used for educational purposes and industrial applications [1] [6]. The exoskeleton hardware was pre-assembled by the previous student groups of Chalmers university of Technology bachelor's degree project, which this project seeks to improve [2].

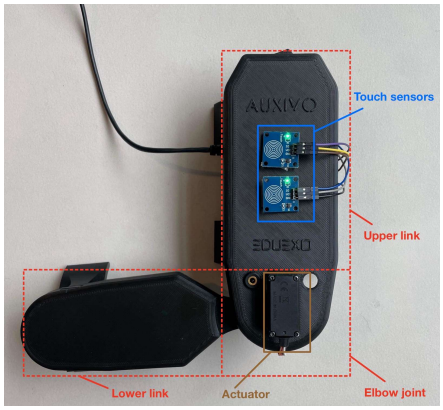


Figure 2.1: Exoskeleton elbow (front) with main components, as provided by the previous group [2].

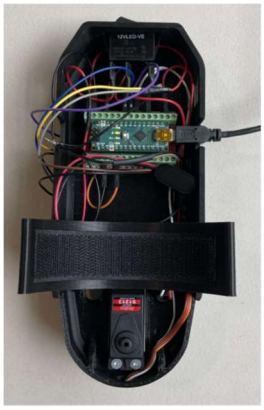


Figure 2.2: Elbow exoskeleton (internal plane) with component layout, as provided by the previous group [2].

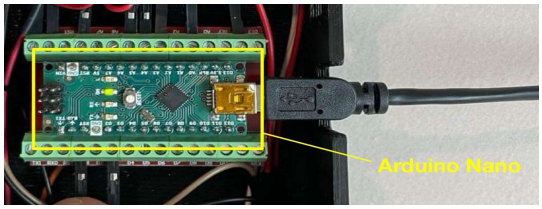


Figure 2.3: Close-up of the microcontroller within the exoskeleton elbow, as provided by the previous group [2].

The assembled exoskeleton main components will be explained in detail further on in this report. These components are:

- **Mechanical structure:** Parts that make up the foundation of the exoskeleton, which includes the frame, links, and joints. Their placements are seen in figure 2.1.
- **Touch sensors:** Sensors that triggers upon touch of a human finger, which for this project, controls which direction the exoskeleton arm rotates along the revolute joint. Attached to the front, outward-facing frame of the exoskeleton as seen in figure 2.1.
- **Arduino nano microcontroller:** A microcontroller attached inside the hollow interior of the exoskeleton frame, seen in figure 2.2 and figure 2.3. It receives and outputs signals and data, which controls the servomotor to change the angular position of the elbow joint, and in turn, the position of the lower link.
- **The analog feedback servomotor:** The part that is responsible for turning electrical energy through the microcontroller to physical motion of the elbow joint, seen in figure 2.2 at the bottom of the frame's interior. Its closed loop properties are key to allowing a method for smooth motion.

2.1.1 Mechanical structure: links and joints

The mechanical structure of an exoskeleton, analogous to the human musculoskeletal system, consists of rigid links connected by joints. Links serve as structural elements that transmit forces and maintain the relative position of joints, while joints enable motion and flexibility in the system.

In this project, the exoskeleton is designed to support elbow motion, which involves a single degree of freedom (DOF) flexion and extension. Thus, a rotational (revolute) joint is implemented to replicate this movement. The system includes two rigid links, representing the upper arm and forearm, connected by the elbow joint. Because of the single joint, it simplifies the kinematic model and allows for direct application of polynomial functions, with relatively minor setbacks.

Material selection for the links prioritizes low weight and high strength, typically using Aluminum or composite materials. Proper joint alignment with the user's anatomical elbow is critical for comfort and functionality, minimizing mechanical resistance and ensuring natural motion.

Overall, the configuration of links and joints directly influences the range of motion, mechanical efficiency, and ergonomic integration of the exoskeleton with the human body.

2.1.2 Touch sensors

Touch sensors are electrical components that capture and record physical touch or embrace on a device or object. They are designed to react and detect capacitive touch of a body, and as such are sensitive to touch, force and pressure. They are used in a wide variety of fields, from smart phones to industrial control panels [7]. They are used to input information about which direction the user wants to move and change the angular position of the exoskeleton's elbow joint.

Figure 2.4: The Touch Sensor model used to control the exoskeleton arm, from the… [8]

The touch sensors work through capacity touch sensing and electrostatic coupling. It operates by an electrode, modeled as two parallel conductive plates, that senses changes in the capacitance relative between the sensor pad and grounding [9]. When a conductive object -like a human finger- approaches or touches the electrode, the capacitance changes and the sensor outputs an electrical signal. A microcontroller connected to the touch can then interpret and process this signal into either an input or output.

2.1.3 Microcontroller: Arduino nano

A microcontroller can be defined as a simplified computer where all components and interfaces are integrated and collected on a single circuit board. They are generally designed to execute a program in repetition at timed intervals [10]. A microcontroller can be defined as a simplified computer where all components and interfaces are integrated and collected on a single circuit board. They are generally designed to execute a program in repetition at timed intervals [10].

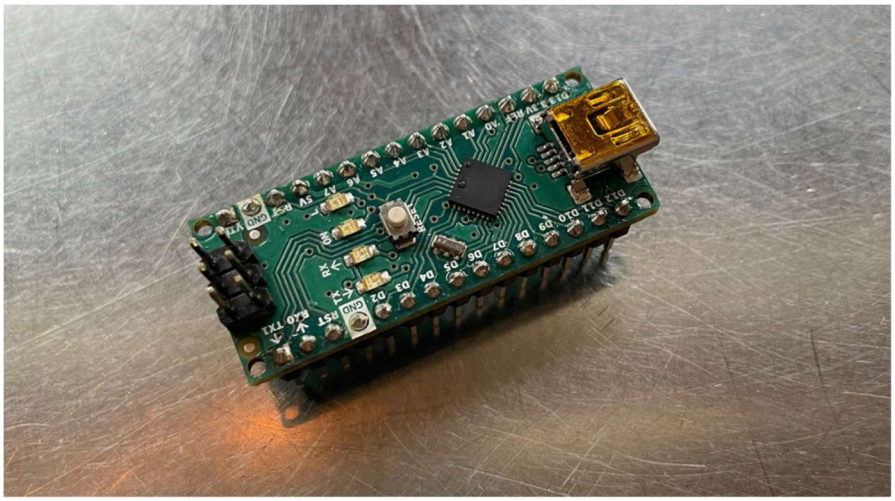


Figure 2.5: The Arduino Nano microcontroller used for the exoskeleton elbow, outside the exoskeleton frame and unconnected [2]

The Arduino Nano microcontroller, as seen in figure 2.5, is a wildly used microprocessor known for being a compact, low-cost design while providing a balance between processing power, memory capacity, low power consumption, and number of I/O pins (input/output). It includes 14 digital input and output pins and 8 analog input pins [11] [12]. These pins allow the microcontroller to control the cozen hardware wired to it, which includes reading sensor data from encoders and generating PWM signals to control servomotors. The board also provides a

Mini-B USB port to allow communication between other electrical components, such as a PC to control or change its programing [13].

2.1.4 Actuator: Analog feedback Servomotor

Actuators are fundamental components in robotic systems, responsible for converting electrical energy into mechanical motion. In exoskeletons, actuators drive the movement of joints to replicate or assist human motion. This project utilizes an analog feedback servomotor, which integrates a DC motor, a control circuit, and a positional feedback mechanism, typically a potentiometer within compact housing.

Figure 2.6: A Batan S1213 Servo

The servomotor used in this system, a Batan S1213 Servo as seen in figure 2.6, operates within a closed-loop control architecture, where the motor's output (angular position) is continuously monitored and corrected based on feedback. When a position command is issued via a PWM signal, the internal controller compares the desired position with the actual position detected by the feedback sensor. The motor adjusts accordingly to minimize the error, allowing for precise and stable angular control [14] [15].

The analog feedback servomotor is particularly suited for low-cost and compact robotic applications due to its simplicity, reliability, and ease of integration. In this context, it is used to actuate the elbow joint of the exoskeleton, providing one degree of rotational motion. Its built-in potentiometer generates an analog voltage corresponding to the shaft angle, which is read by a microcontroller and used for real-time control.

While analog feedback systems may not offer the high resolution and noise immunity of digital encoders, they are advantageous in systems where moderate precision and low system complexity are acceptable. Moreover, the integrated design reduces the number of external components required, simplifying the mechanical and electrical layout of the exoskeleton.
In summary, the analog feedback servomotor provides a cost-effective and efficient solution for controlling joint movement in exoskeletons, enabling responsive and accurate actuation while maintaining system simplicity and compactness [16].

2.1.4.1 Potentiometer Function and Encoder

Accurate motion sensing and position feedback are essential in modern exoskeleton systems, where adaptive interaction between the user and the robotic structure depends on reliable sensor technology. Two core components used for feedback control in such systems are potentiometers and encoders, both of which convert mechanical motion into electrical signals that enable precise regulation.

In the S1213 servomotor, a built-in potentiometer serves as an angular position sensor, measuring the shaft's rotation and providing real-time feedback. The potentiometer functions as a voltage divider with three terminals—supply (V_s), ground (GND), and output (V_{out}). As the motor shaft rotates, the wiper moves along a resistive track, altering the resistance ratio and producing a voltage proportional to the angular position. This voltage is read by a microcontroller via an analog input and used in a closed-loop control system. The motor is controlled by PWM (Pulse Width Modulation) signals, where pulse duration specifies the desired position, as explained in the following figure 2.7. The potentiometer's analog feedback confirms the actual angle, enabling continuous adjustment [17].

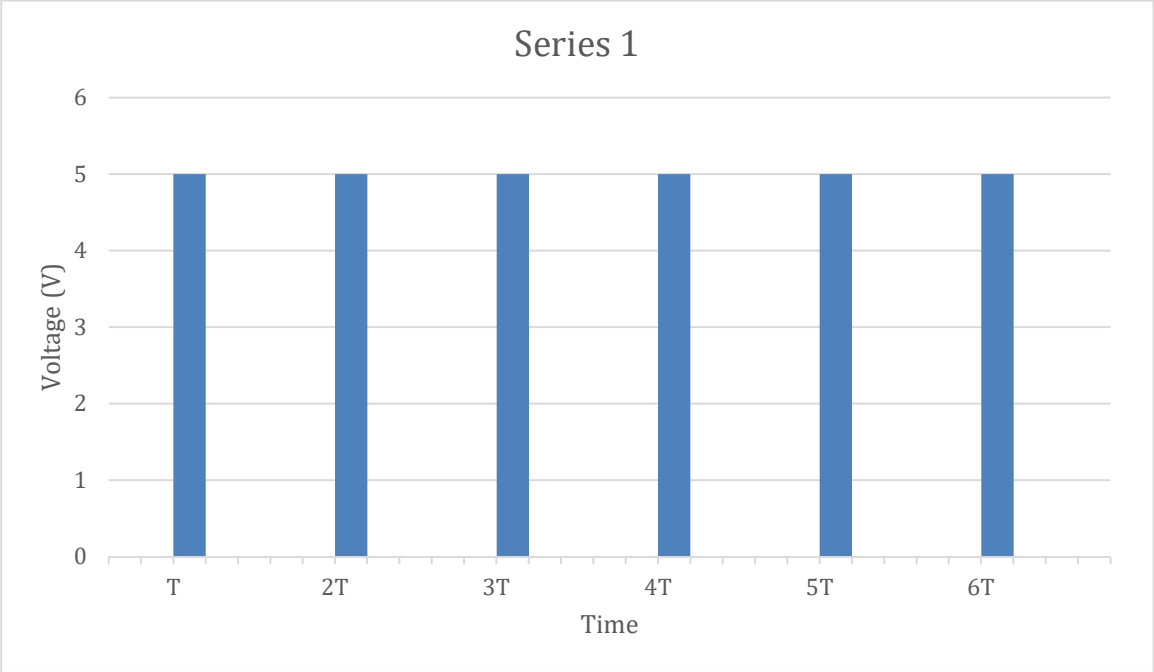


Figure 2.7: Voltage-time graph of how the desired motor angle is encoded using pulse width modulation (PWM) from the Arduino to the servo, based on the voltage-time graph in the AUXIVO EduExo handbook [17].

Technically, the potentiometer is classified as a resistive sensor, since it generates output based on changes in electrical resistance. Although it is not a digital encoder, it performs a similar role to an analog encoder, delivering continuous position feedback. Compared to optical or magnetic encoders, which rely on light or magnetic fields, the potentiometer offers a simpler, more cost-effective, and reliable analog alternative.

Encoders, on the other hand, play a vital role in advanced exoskeleton control. They are categorized as incremental encoders, which measure relative movement, and absolute encoders, which retain positional information even after power loss. Optical encoders are preferred for their high resolution, whereas magnetic encoders are better suited for harsh environments.

These sensors are typically mounted at joint axes or actuator shafts to capture precise information about limb flexion or rotation. The data helps synchronize actuator movement with the user's biomechanics, supporting both ergonomic load distribution and rehabilitative guidance.

In conclusion, sensors such as potentiometers and encoders are fundamental to the operation, safety, and responsiveness of exoskeleton systems. By enabling real-time feedback and accurate motion tracking, they support intelligent control strategies that enhance user experience and biomechanical performance [18].

2.2 Arduino IDE

Arduino IDE is a computer programming software used for developing and uploading codes to the Arduino family of microcontroller boards. With a USB cable connecting the microcontroller and a computer with the software, it enables users to write, edit, compile, and upload codes to Arduino boards. The program is used wildly for educational and prototyping purposes, mainly for its accessibility and ease of use [19] [20].

The programming language for Arduino IDE is known as the Arduino API. The language is based on a simplified version of the programming language C++, in its functions, variables, and structure [21].

The program provides a text editor, uploader, compiler, and a serial monitor. The text editor is used for writing and editing source code (referred as sketches in Arduino), which then can be compiled to create a binary file of the code and uploaded to the microcontrollers flash memory [22]. The serial monitor allows data to be sent between the computer and the microcontroller and can then be used as a debugging tool for the code [23].

2.2.1 Servo Library

Arduino IDE allows users to access and include official, and community-created libraries from the Arduino network. Libraries are C or C++ files that provide sketches that include them with additional functions, which is done by importing the files in Arduino IDE and then writing in the code a #include statement, followed by the header (.h) file name in the library's folder [24].

For this project, it requires the use of the Servo library. The Servo library allows Arduino boards to control a variety of servomotors, including analog feedback servomotors [25]. The library includes six functions, however in this project, only three of them are important to understand.

- ***attach()***
- ***write()***
- ***writeMicroseconds()***

The *attach()* function attaches the servo variable to a pin. The functions *write()* and *writeMicroseconds()* both allow the Arduino microcontroller to write a value to the attached servo to move and set angle of the servo shaft. When using the *write()* function, the input value for the function is an integer angle in degrees [°] between 0° and 180°. The *writeMicroseconds()* function uses and accepts specific pulse width modulated (PWM) microseconds to move the servo shaft. The range of values for *writeMicroseconds()* depends on the servomotor, however it most commonly accepts inputs between 1000 µs and 2000 µs, which corresponds to the degree angles between 0° and 180° [25].

Because the *writeMicroseconds()* functions have a wider range of acceptable input values compared to the *write()* function, *writeMicroseconds()* provides a higher resolution of control of the servomotor.

2.3 MATLAB

MATLAB (matrix laboratory) is a high-level programming software used for engineering, mathematics and scientific applications. It was developed by MathWorks and is designed for numeric computations, algorithm development, and data analysis [26]. The platform provides built-in support for serial communication, which allows real-time reading and writing of data to and from external hardware, such as microcontrollers.

Serial communication is a method of digital data transmission where data is sent sequentially in bites, bit by bit, over a single communications line such as a USB connection. It is commonly used in embedded systems, such as microcontrollers, due to its simplicity, low hardware requirements, and its applicability for long-distance communication [27].

This allows MATLAB to plot data from the Arduino nano microcontroller, such as variables used as command inputs for the servomotor and feedback signals from the encoder. This creates a live visualization of the system that provides insight into its behavior and performance.

MATLAB provides additional toolboxes and add-ons that can be downloaded during or after installation of the program. These toolboxes and add-ons expand the capabilities and functionality of MATLAB and allow for a variety of additional uses of the program. For this project, only the essential base toolboxes were used, without additional add-ons.

2.4 Splines (or polynomial functions)

Splines are piecewise defined functions of polynomial segments that are connected at specific points known as knots. The segments are defined over finite intervals and join adjacent segments that share function value and derivatives to form continuity.
Mathematically, splines $S(t)$ can be expressed as:

$$S(t) = \begin{cases} P_0(t), & t_0 \le t < t_1 \\ P_1(t), & t_1 \le t < t_2 \\ \vdots & \\ P_k(t), & t_{k-1} \le t < t_k \end{cases} \tag{2.1}$$

where $P_i(t)$ is a polynomial function of degree n, defined over variable intervals $[t_{i-1}, t_i]$.
The polynomial function can be defined as:

$$P_n(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n \tag{2.2}$$

with a_i being polynomial coefficient that are determined based on interpolation or boundary conditions of the polynomial [28]. The process of determining these will be covered in the following methodology section.

In robotics or motion control, splines are useful to generate smooth motion trajectories of an object or device that adhere to both positional and dynamic constraints.
In the context of moving an object, these constraints often include

- Position
- Velocity

- Acceleration
- Jerk

The number and type of constraints to adhere to determines the minimum degree of the polynomial function required.

To give some examples, with the coefficients based on boundary conditions at both the start and at the end of the movement path. A 3^{rd}-degree polynomial function allows for continuous position and velocity, while a 5^{th}-degree polynomial function adds continuous acceleration to the motion.

In this project, the splines and polynomial functions are based on a time variable.

3
METHODOLOGY

This chapter of the report describes the methods, techniques and processes used for the development of the exoskeleton elbow.

3.1 Architecture

The architecture shown here is to better understand how the systems are connected to one another, and how they interact and affect one another.

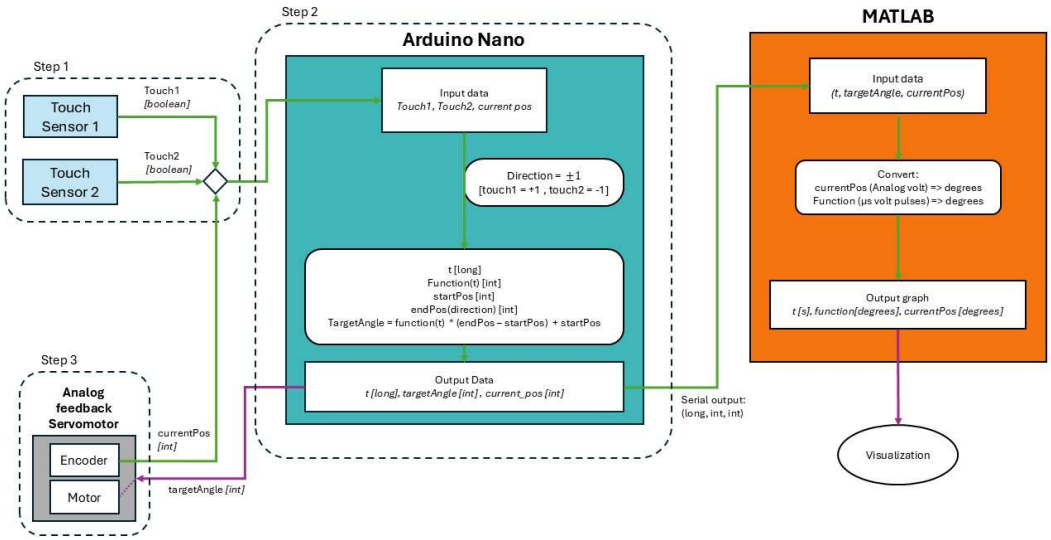


Figure 3.1: Data flow architecture of the exoskeleton, connected to a computer with a USB cable.

Figure 3.1 shows a diagram detailing the connections between Arduino and MATLAB, as well as the electrical components of the exoskeleton. It gives an overview of the data flow within and between components and programs, as well as a simplified description of the processes.

3.2 Software development process

To write, understand, and learn the programming languages and their codes, AI tools like ChatGPT were used for both Arduino API and MATLAB, due to the group's lack of experience with both programming languages.

3.3 Determining Arduinos and the servos resolution

To better understand the Arduino nano microcontroller and the servomotor's capabilities and limitations, several tests were performed to this endeavor.

What needed to be known was:

- The servos shaft maximum rotation speed
- The servos movement resolution (the smallest change of position possible).
- The shortest time for the servo to receive and complete a command

Rotation speed and the shortest time for the servo to complete a command requires a measurement of time. This measurement was done in the Arduino Microcontroller with the

usage of the mills() and micros() functions, which measures the time from the point when the program started.

3.3.1 Determining the servos rotation speed

The project began by identifying the range of motion required for the exoskeleton arm. The current design uses an existing arm with a single degree of freedom, operating between 0 and 90 degrees. The goal is to achieve smooth acceleration at the start of motion and smooth deceleration when stopping.
To measure rotation speed, a test program was developed for Arduino that would send a single command to the servomotor to reach a destination angle. The program allowed the user to press a touch sensor ten times, recording the time required for the servo to move between 0-90 degrees each time. The program then calculated the average time and frequency. Result showed an average movement time of approximately 0.58 seconds which is used for the next step.

3.3.2 Determining the servos resolution

In this project, a program was developed to study the relationship between a servo motor's control signal, given as pulse width in microseconds (µs), and its angular movement from 0–90 degrees. Measurements showed that a pulse of 551 µs corresponded to 0°, and 1471 µs corresponded to 90°. This gave a total range of 920 µs (1471 – 551). In theory, this means the control signal could be changed in 920 steps of 1 µs, giving 920 possible angles.

However, a servo motor's actual resolution is not only determined by the smallest change in the control signal. Mechanical and electronic limits inside the servo decide the minimum pulse change that produces movement. To find this limit, an additional program was created. It increased the pulse width by 1 µs at a time and displayed "yes" if the servo moved and "no" if it did not.

The results showed a repeating pattern: "no, no, no, no, yes." This meant that the servo only moved after a change of 5 µs. Therefore, even though the signal could be changed in 1 µs steps, the servo's effective resolution was 5 µs. Dividing the 920 µs range by 5 gave 184 distinct angle positions, which is about 0.49° per step.

This demonstrates the difference between signal resolution (the smallest step the controller can send) and mechanical resolution (the smallest step the servo can perform). The lower mechanical resolution can be caused by factors such as gear backlash, inertia, and the accuracy of the internal position sensor.

The Arduino Servo library was used to send control signals to the servo. This library controls the timing of PWM (Pulse Width Modulation) signals, sending a HIGH pulse every 20 ms. The length of the HIGH pulse (for example 551–1471 µs) determines the target position. The library handles this process automatically, which allows precise control without the need to write low-level timing code.

Overall, the study showed that while the control signal can be adjusted with high precision, the servo's actual performance is limited by its mechanical and electronic resolution. This must be considered when designing systems that require accurate positioning.

3.3.3 Determining the shortest command time

By using the results from the previous two tests, it was possible to determine the shortest command time. In the first test, the shortest time measured was 0.58 seconds. The second test showed that the servo had 184 distinct angular positions, corresponding to approximately 0.49 degrees per step. By dividing 0.58 seconds by 184 positions, the smallest time required for a single angular change was obtained.

3.4 Defining spline functions

The Splines used for the movement controller requires defining the coefficients of the polynomial functions.
To ease the development of the movement, the spline only uses one polynomial function, which defines the elbow exoskeleton joints angular position. The polynomial function was of a 5th-degree, and integrates boundary constraints for joints positional and dynamic properties:

- angular position $\theta\ [deg°] = P_5(t)$
- angular velocity $\dot{\theta}\ [deg°/s] = \dot{P}_5(t)$
- angular acceleration $\ddot{\theta}\ [deg°/s^2] = \ddot{P}_5(t)$

$$\begin{cases} P_5(t) = \theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \\ \dot{P}_5(t) = \dot{\theta}(t) = 0 + a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 \\ \ddot{P}_5(t) = \ddot{\theta}(t) = 0 + 0 + 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 \end{cases} \quad (3.1)$$

Each of these properties are determined by the movement time $t\ [\mu s]$ of the exoskeleton to complete its motion. Movement time is relative from the initial time $t_i\ [\mu s]$ of the movement (t_i can be treated as being equal to 0 µs), and the time it takes to reach its destination $t_f\ [\mu s]$. The destination time cannot be lower than the minimum time determined in chapter 3.3.1 and is for this project assigned as 3 seconds (3,000,000 µs).

Table 3.1: Table of boundary conditions for the polynomial function.

Boundary conditions	$t_i\ (= 0\ s)$	$t_f\ (= 3\ s)$
Angular position: $\theta(t)$	θ_i	θ_f
Angular velocity $\dot{\theta}(t)$	0	0
Angular acceleration: $\ddot{\theta}(t)$	0	0

With the polynomial function and its boundary constraints defined, define it as the matrix equation:

$$\boldsymbol{P_5 = T * A} \quad (3.2)$$

Where $\boldsymbol{P_5}$ is a vector containing the boundary conditions, $\boldsymbol{T}$ is a matrix containing the function variables, and $\boldsymbol{A}$ is a vector containing the polynomial coefficients.

$$\underset{\boldsymbol{P_5}}{\begin{bmatrix}\theta_i\\ \theta_f\\ 0\\ 0\\ 0\\ 0\end{bmatrix}} = \underset{\boldsymbol{T}}{\begin{bmatrix}1 & t_i & t_i^2 & t_i^3 & t_i^4 & t_i^5\\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5\\ 0 & 1 & 2t_i & 3t_i^2 & 4t_i^3 & 5t_i^4\\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4\\ 0 & 0 & 2 & 6t_i & 12t_i^2 & 20t_i^3\\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3\end{bmatrix}} * \underset{\boldsymbol{A}}{\begin{bmatrix}a_0\\ a_1\\ a_2\\ a_3\\ a_4\\ a_5\end{bmatrix}} \tag{3.3}$$

Determining the coefficients in vector $\boldsymbol{A}$ requires the inverse matrix of T ($= T^{-1}$). Since the $\boldsymbol{T}$ matrix has a size of 6x6, calculations of the inverse matrix were done through the computational knowledge engine Wolfram alpha [29], to simplify the process.

$$\boldsymbol{A = T^{-1} * P_5} \tag{3.4}$$

$$\underset{\boldsymbol{T}\ (t_i = 0)}{\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5\\ 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4\\ 0 & 0 & 2 & 0 & 0 & 0\\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3\end{bmatrix}} \rightarrow \frac{1}{2} * \underset{\boldsymbol{T^{-1}}}{\begin{bmatrix}2 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 2 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ -\frac{20}{t_f^3} & \frac{20}{t_f^3} & -\frac{12}{t_f^2} & -\frac{8}{t_f^2} & -\frac{3}{t_f} & \frac{1}{t_f}\\ \frac{30}{t_f^4} & -\frac{30}{t_f^4} & \frac{16}{t_f^3} & \frac{14}{t_f^3} & \frac{3}{t_f^2} & -\frac{2}{t_f^2}\\ -\frac{12}{t_f^5} & \frac{12}{t_f^5} & -\frac{6}{t_f^4} & -\frac{6}{t_f^4} & -\frac{1}{t_f^3} & \frac{1}{t_f^3}\end{bmatrix}} \tag{3.5}$$

With T^{-1} defined, determine the coefficients with matrix vector multiplication.

$$\begin{cases} a_0 = \theta_i \\ a_1 = 0 \\ a_2 = 0 \\ a_3 = \dfrac{10}{t_f^3} * (\theta_f - \theta_i) \\ a_4 = \dfrac{-15}{t_f^4} * (\theta_f - \theta_i) \\ a_5 = \dfrac{6}{t_f^5} * (\theta_f - \theta_i) \end{cases} \tag{3.6}$$

With determined coefficients, implement them in the polynomial function. When implementing the polynomial coefficients to the polynomial function, and then breaking out variables and simplifying it, you get a movement function with a "Smooth" path that can go to any one point to another.

$$P_5(t) = \theta_i + (\theta_f - \theta_i) * \left(\frac{10}{t_f^3} * t^3 - \frac{15}{t_f^4} * t^4 + \frac{6}{t_f^5} * t^5\right) \tag{3.7}$$

3.5 Testing different polynomial grades

Since the movement of the arm is required to be "smooth", testing different degrees of polynomial functions were essential. The degrees of polynomial were based on three criteria:

- How "smooth" the motion felt.
- Can the Servomotor move with the required precision?
- Can the Arduino microcontroller handle the calculations without difficulties or processing issues?

The tests were done by creating different polynomial grade functions into the Arduino microcontroller through Arduino IDE, testing each of them with the exoskeleton, and comparing the results. To get a general idea of the motion paths, Wolfram alpha [29] was used to plot the motion paths, to better compare them directly.
The experiments and their results are detailed in chapter 4.

3.6 Creating the final control code

With the Spline function defined, the next step is to implement the function into and creating the final code, **"Full control model final"**.

The program begins with the standard setup explained in chapter 3.3, and aligning the servomotor shaft and exoskeleton arm to 0 degrees (straight down).

The exoskeleton then waits for one of the two touch pads to sense register an input. Once an input has been detected, it will check if the button was pressed or held. Afterwards, it will then assign what the next position of the servo joint will be, based on which touch sensors were triggered and whether if it was pressed or held. If one of the touch sensors is pressed, then it will assign the next angular positions to be an increase (touch 1) or decrease (touch 2) of the current position by 30°. If one of the two sensors were held, then the program will assign the next position either 90° (touch 1) or 0° (touch 2).

Once a destination is assigned, a timer starts that counts in microseconds (micros()). When the timer has exceeded the "minimum time interval" (se chapter 3.3.3), it will then use the spline function to calculate the next position the servo is to take at that current time. Once calculations are done, the program sends the next position to the servomotor, and then the program prints out the serial data on what the current time is, the calculated position of the function, and the current position read from the encoder. This process loops until the time arrives or exceeds the defined time of arrival (t_f), which for this project is 3 seconds.
Once the defined time of arrival has been reached, the program sets its current angular position as the initial angle and waits for the next input from one of the touch sensors.

3.7 Real-time graph with MATLAB

This part goes over how the MATLAB code works to create the plot for the exoskeleton movement graph.

Work on the MATLAB code began after the completion of the **"Full control model final"** Arduino code. For the MATLAB real-time plotting code to create graphs based on the output serial data from the microcontroller without overriding the previously uploaded code from Arduino IDE, it only reads the USB serial data from the microcontroller.

While Arduino IDE code **"Full control model final"** is uploaded to the microcontroller, and the board is connected to a USB port with the MATLAB software download, run the real-time

graph plotter code in MATLAB. When running the MATLAB code, it will launch a MATLAB figure window with the x-axis representing time and y-axis representing the angle in degrees.

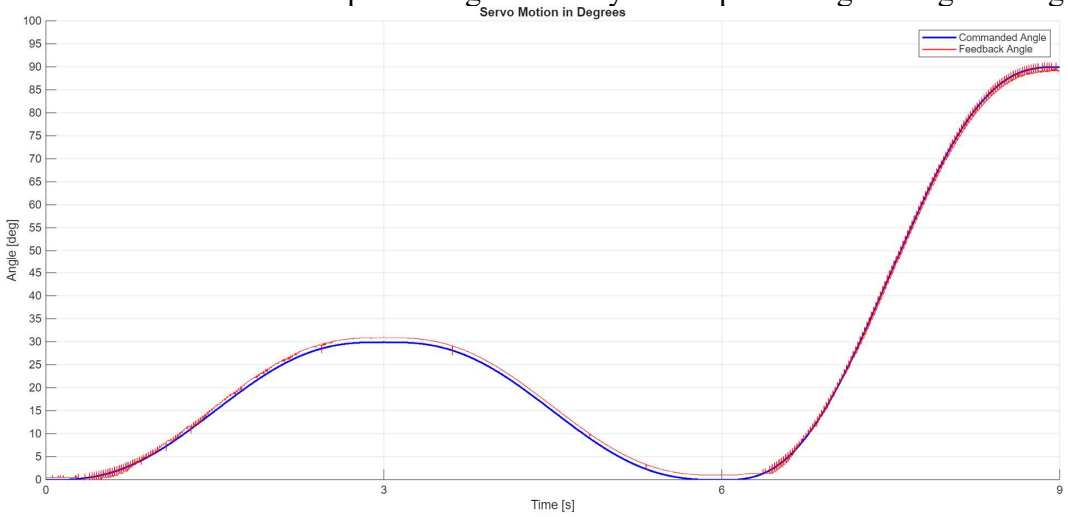


Figure 3.2: MATLAB plot of the exoskeleton motion, using the code "Continous_plotter_V4".

With the figure window open, seen in figure 3.4, MATLAB will then wait for serial data output from the microcontroller to read. Once the microcontroller detects serial data from the board, MATLAB will use the values read to plot out the graph in real time. The data used to plot out the graph is the polynomial functions result value sent to the servomotor (the Command Angle line in figure 3.4 colored blue), and the encoder reading (the Feedback Angle line in figure 3.4 colored red), at the respective timeframe they were read based on the time variable from the output data of the Arduino microcontroller. Once a motion is completed and the data is plotted in the graph, the MATLAB code stops updating the graph and waits for new data to be read from the exoskeleton. The x-axis was set to include time intervals of 9 seconds, done to limit the size of the plot for readability. In figure 3.4, this means when it reads another output from the exoskeleton, the graph will remove plotting between 0 and 3 seconds and update the graph to visualize the plot between 3-12 seconds as an example.

4
EXPERIMENTAL EVALUATION AND RESULTS

4.1 Encoder feedback angle filter

An issue encountered with the potentiometer encoder was its susceptibility to electrical noise. While readings form encoder whenever the exoskeleton was still giving relatively accurate value with little noise, readings during the motion of the exoskeleton arm resulted in an increase of electrical noise, as demonstrated by the following figure 4.5

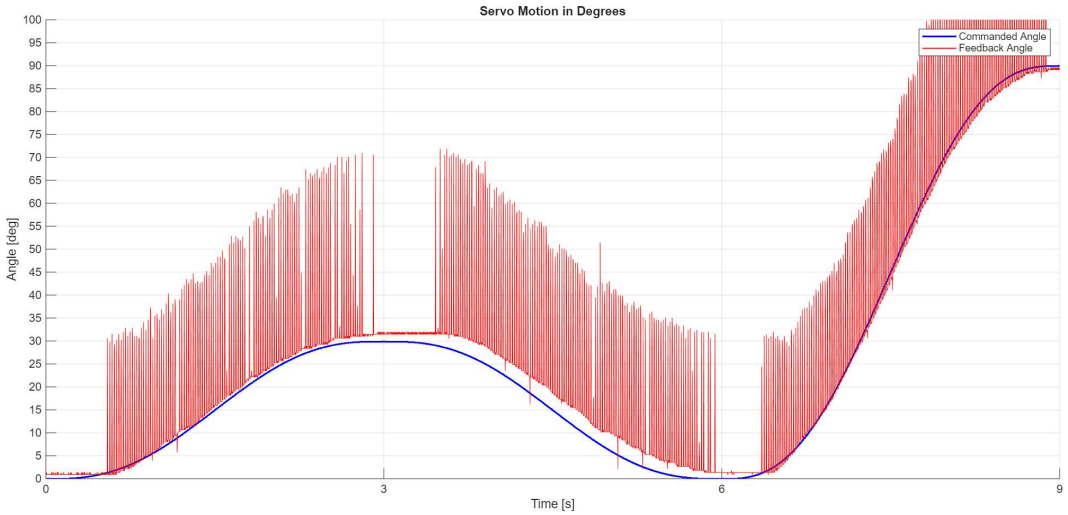


Figure 4.1: MATLAB plot of the exoskeleton motion without encoder filter, using the code "Continous_plotter_V4_raw_encoder".

This created difficulties with tasks where performance is dependent on encoder readings, such as time measurement of the exoskeleton movement between two points as an example, since the program would require a method to know when it arrived at its destination. These noises were due to electrical noise around the potentiometer feedback encoder, of which it is susceptible to. Usually, this problem can be remedied by implementing and altering components that filter electrical noise, however due to the limitations imposed, tests were made to implement filters within the uploaded Arduino code and MATLAB code.

For the Arduino code, the resulting experimental filters were unable to solve the noise issue without compromising essential parts of the task that it was used in, namely time measurement tasks. An example of these proposed filters was a function performing multiple readings in each instance instead of only once, and then returning either the mean, median, or typical value for that instance. While it would slightly reduce noise, the extra performance required of these functions resulted in inaccurate time measurements. Adding manual delays to each reading in the function would return more accurate results, however this naturally delayed the code itself and would return inaccurate time measurements.

A filter code for MATLAB was decided to be implemented, because the raw values read reduced the readability of the plot, as seen in figure 4.1. Since the encoder value aligned relatively well with the command angle despite the noise, the filter implemented only removed values that offset the command value with more than one percent. Refer to figure 3.4 for the resulting graph with the implemented filter.

4.2 Polynomial degree mathematical test

The test done here were to determine the most fitting polynomial degree to get the "smoothest" possible movement path.

To evaluate this, the group made polynomial functions of 3rd-, 5th-, and 7th-degree polynomial functions with the same method described in section 3.4 of this report. The difference between the 5th-degree and both the 3rd- and 7th-degree functions is with the boundary conditions, where 3rd-degree abstains the boundary conditions for angular acceleration, while the 7th-degree add boundary conditions to jerk motion, with it being the third derivative of angular position and set to 0 $[deg°/s^3]$ at the beginning and at the end of the motion.

Calculated with wolfram alpha, the computation tool also the creation of plots as seen in the following figure 4.1.

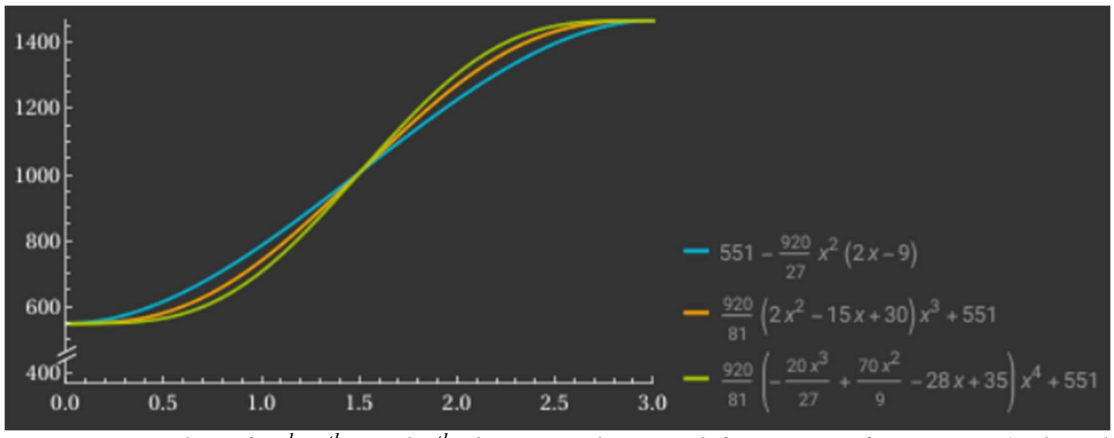


Figure 4.2: Plot of 3rd, 5th, and 7th degree polynomial functions of x = time (colored blue, orange, and green respectively), calculated and plotted in Wolfram Alpha [29]

The plots were made to get a simplified overview of the motion, to better assess the appropriate motion of the arm. Through observation, the group concluded that the function is best suited for 5th and 7th degree function, as the 3rd degree motion path was too linear for the project goal, in theory. The following experiment was a practical application of the functions calculated in wolfram would later be used in the following test.

4.3 Splines function practical test

To better evaluate our observation, the next step was to implement the different degrees of polynomial functions into the main control loop code **"Full control model final"**, running the code for each polynomial degree (by having it go from 0 to 90 degrees), and then evaluate the resulting motion based on how well they fulfill the goals of the project. The method was done by observing the motion of the exoskeleton, as seen in the following figures 4.6, 4.7, and 4.8, comparing them to the resulting MATLAB graphs of them, and taking note of the Arduino and the servomotors performances.

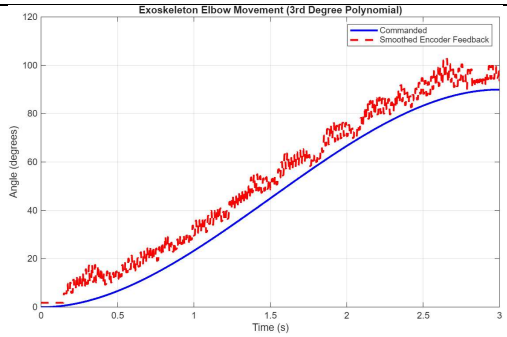


Figure 4.3: Exoskeleton movement graph of the Third-degree polynomial function in MATLAB (Blue = function value, red = encoder reading)

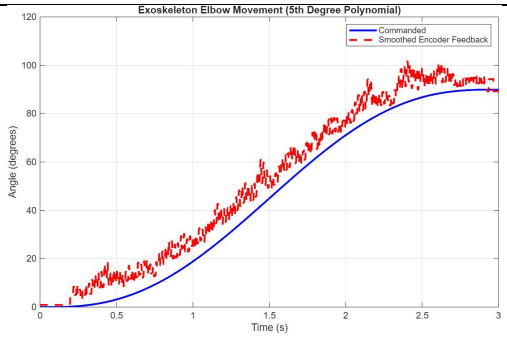


Figure 4.4: Exoskeleton movement graph of the Fifth-degree polynomial function in MATLAB (Blue = function value, red = encoder reading)

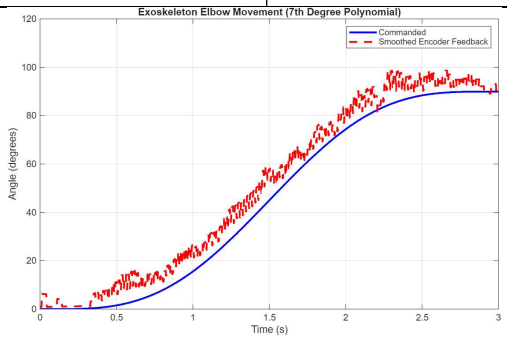


Figure 4.5: : Exoskeleton movement graph of the Seventh-degree polynomial function in MATLAB (Blue = function value, red = encoder reading)

Figure 4.6: First part of the exoskeleton motion. Pressing and holding the touch sensor to initiate the 0-to-90-degree motion of the exoskeleton.

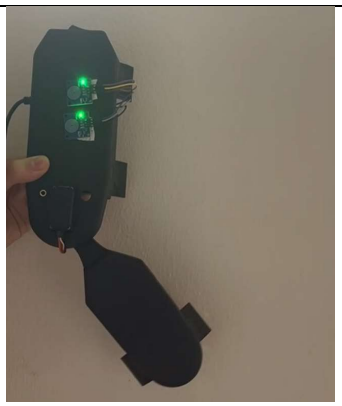


Figure 4.7: Second part of the exoskeleton motion. The movement of the exoskeleton's elbow joint from 0 to 90 degrees.

Figure 4.8: Third part of the exoskeleton motion. The arm reaching the desired joint angle of 90 degrees.

Through observation, the group confirmed their evaluation of the 3rd degree polynomial function, as its movement compared to the other degrees were lacking in comparison. The test of the 7th degree polynomial exposed a few issues with its implementation to the exoskeleton.

Arduino IDE function to calculate exponentials has difficulties in outputting values with high exponential values, and function is susceptible to returning overflow values (when the value of a variable exceeds its maximum value limit). Since the function required for the project requires the usage of microseconds in the function for better control, these factors together made it difficult to implement while allowing altering the movement time. The group would eventually find a solution to the issue, detailed in the "Full model code final" in the code repository (link in appendix). Calculating the 7th degree function for the microcontroller requires more processing power, which increases the time it takes for it to calculate and send signal commands to the servomotor.

The practical test results of the exoskeleton movement with 7th degree polynomial function does improve theoretically and graphically, according to the figure 4.5, to the 5th degree motion. However practically, the differences between the 7th and 5th degree function movements were observed to not have any noticeable difference by the group.
The results of the test led to the group using a 5th degree polynomial function for the project.

5
CONCLUSIONS AND DISCUSSIONS

In this project, we began by learning about Arduino and ROS2. However, during the work, we discovered that ROS2 would not be useful for our purposes. We ended up learning a lot of things beyond what was needed, as we tested and worked with many different methods and codes that turned out to be unnecessary for solving the main problem.

The goal of the project was to find a solution that would make the arm move smoothly at the start and stop points. To achieve this, we needed to determine the minimum signal transmission time from the Arduino to the servo. We came up with many different ideas for how this could be done, which led us to write several complex and sometimes overly complicated programs to measure the frequency and minimum time. Unfortunately, these tests did not produce precise or consistent results, as the measurements varied significantly each time.

Eventually, we arrived at a simple but effective solution: we measured the time when the arm began to move until it stopped and then divided this time by the number of angles changes the servo could handle in microseconds.
We're glad that we went through all these trials, as we learned a great deal along the way—things we would not have had the opportunity to discover otherwise. At the same time, we'd wish we had come up with the final idea earlier so that we could have saved time.

For the next group, we'd recommend that, if they want to determine the minimum time and frequency, they should first gain a deeper understanding of how Arduino and programming work or consult someone experienced. This would allow them to write more efficient code and reach a solution faster. Personally, we believe that our ideas for measuring frequency and time were good, but we needed more time and knowledge to reach a definitive answer.

BIBLIOGRAPHY

[1] AUXIVO. Exoskeleton | EduExo Lite. [Online].; 2024 [cited 2025 Maj 12. Available from: https://www.auxivo.com/eduexo-lite.

[2] Ha S, Tingberg F. Arduino/ROS2 Control Software Development for an Elbow Exoskeleton. Slutrapport. Göteborg: Chalmers Tekniska högskola, Elektroteknik; 2025.

[3] Gopura RARC, Bandara DSV, Kiguchi K, Mann GKI. Developments in hardware systems of active upper-limb exoskeleton robots: A review. In Robotics and Autonomous Systems.; 2016. p. 203-220.

[4] Gull MA, Bai , Bak T. A Review on Design of Upper Limb Exoskeletons. 2020 February; 9(1): 16.

[5] Bai S, Virk GS, Sugar TG. Wearable exoskeleton systems: Design, control and applications.: Institution of Engineering and Technology; 2018.

[6] AUXIVO AG. "About us"-auxivo.com. [Online].; 2019 [cited 2025 August 7. Available from: https://www.auxivo.com/about.

[7] Farnell. Touch Sensors. [Online]. [cited 2025 Juli 29. Available from: https://se.farnell.com/sensor-touch-sensor-cap-res-technology.

[8] Osepp. "Touch sensor modul" Osepp.com. [Online]. [cited 2025 August 13. Available from: https://osepp.com/electronic-modules/sensor-modules/80-touch-sensor-module.

[9] Baharava Z, Kakaralab R. Capacitive touch sensing: signal and image processing algorithms. Computational Imaging IX, Proc. SPIE 7873,; 2011.

[10] RS Components. "A Complete Guide to Microcontrollers". [Online].; 2023 [cited 2025 juni 24. Available from: https://uk.rs-online.com/web/content/discovery/ideas-and-advice/microcontrollers-guide.

[11] Arduino. Arduino Store. [Online]. [cited 2025 07 31. Available from: https://store.arduino.cc/products/arduino-nano?srsltid=AfmBOopuYYB2hFO71_TzBCxo3ry6dtS98iLqUHXCVnSD4l3rcw9K1A6f.

[12] Arduino. Nano. [Online].; 2008 [cited 2025 juli 30. Available from: https://docs.arduino.cc/resources/datasheets/A000005-datasheet.pdf.

[13] EmmaAshely. "What is Arduino Nano? A Getting Started Guide", RS-DESIGNPARK. [Online].; 2021 [cited 2025 juli 31. Available from: https://www.rs-online.com/designspark/what-is-arduino-nano-a-getting-started-guide.

[14] Firgelli Automations. "Actuators - what is an actuator?"-firgelliauto.com. [Online]. [cited 2025 juli 31. Available from: https://www.firgelliauto.com/pages/actuators?srsltid=AfmBOoriYxRfcPB_taXzB32xApvt8Dc_TVdBxcjzKXhmlfWnAdOwUxLe&utm.

[15] Firgelli automations. "What is a Rotary Actuator and how are they used" - firgelliauto.com. [Online].; 2023 [cited 2025 Augusti 01. Available from: https://www.firgelliauto.com/blogs/actuators/what-is-a-rotary-actuator-and-how-are-they-used?srsltid=AfmBOoobP2fFYo9m568tg2l4BqY5e5lCG1fjr1pye9bnAXQTr2L5QY7V.

[16] Earl B. Adafruit. [Online].; 2025 [cited 2025 05 13. Available from: https://learn.adafruit.com/analog-feedback-servos?view=all.

[17] Auxivo AG. EduExo - the robotic exoskeleton kit handbook and tutorial. 3rd ed.: AUXIVO; 2022.

[18] Cook JS. arrow. [Online].; 2019 [cited 2025 August 9. Available from: https://www.arrow.com/en/research-and-events/articles/encoder-vs-potentiometer-how-to-choose.

[19] Arduino. Arduino Software. [Online].; 2022 [cited 2025 August 6. Available from: https://www.arduino.cc/en/software/.

[20] Arduino. "ArduinoDocs" - docs.arduino.cc. [Online]. [cited 2025 augusti 1. Available from: https://docs.arduino.cc/software/ide/.

[21] Söderby K. "Getting Started with Arduino"-docs.arduino.cc. [Online].; 2024 [cited 2025 August 4. Available from: https://docs.arduino.cc/learn/starting-guide/getting-started-arduino/.

[22] Arduino , Bagur J, Chung T. "Arduino Memory Guide"-docs.arduino.cc. [Online].; 2023 [cited 2025 August 4. Available from: https://docs.arduino.cc/learn/programming/memory-guide/.

[23] Söderby K. "Using the Serial Monitor tool"-docs.arduino.cc. [Online].; 2024 [cited 2025 August 4. Available from: https://docs.arduino.cc/software/ide-v2/tutorials/ide-v2-serial-monitor/.

[24] Arduino. "Arduino Libraries" Arduin Docs. [Online].; 2024 [cited 2025 August 13. Available from: https://docs.arduino.cc/retired/hacking/software/Libraries/.

[25] Arduino. "Servo" - Arduino docs. [Online].; 2024 [cited 2025 August 13. Available from: https://docs.arduino.cc/libraries/servo/#Usage/Examples.

[26] MathWorks. "What is MATLAB"-mathworks.com. [Online]. [cited 2025 Augusti 1. Available from: https://mathworks.com/discovery/what-is-matlab.html.

[27] Axelson JL. Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems. 2nd ed. Madison: Lakeview Research LLC; 2007.

[28] Burden RL, J. FD. Numerical Analysis. 9th ed. Julet M, editor. Boston: Cengage Learning; 2010.

[29] Wolfram. "Wolfram|Alpha". [Online].; 2009 [cited 2025 August 8. Available from: https://www.wolframalpha.com/.

[30] Arduino.cc. Arduino Documentation. [Online].; 2025 [cited 2025 april 14. Available from: https://docs.arduino.cc.

[31] Siciliano B, Sciavicco L, Villani L, Oriolo G. Robotics: Modelling, Planning and Control (Advanced Textbooks in Control and Signal Processing). 1st ed. Michael J. G, Professor Michael A. J, editors. London: Springer; 2009.

[32] Texas Instruments. Capacitive Sensing Basics. [Online].; 2017 [cited 2025 Juli 30. Available from: https://software-dl.ti.com/msp430/msp430_public_sw/mcu/msp430/CapTIvate_Design_Center/1_83_00_08/exports/docs/users_guide/html/CapTIvate_Technology_Guide_html/markdown/ch_basics.html#.

[33] Earl B. "Analog Feedback Servos" - learn.adafruit.com. [Online].; 2025 [cited 2025 juli 31. Available from: https://learn.adafruit.com/analog-feedback-servos/about-servos-and-feedback.

[34] Firgelli Automations. "Actuators - what is an actuator?"-firgelliauto.com. [Online]. [cited 2025 july 31. Available from:

https://www.firgelliauto.com/pages/actuators?srsltid=AfmBOoriYxRfcPB_taXzB32xApvt8Dc_TVdBxcjzKXhmlfWnAdOwUxLe&utm.

APPENDIX

Link to video demonstration of the elbow exoskeleton demo: https://youtu.be/RSOuFtCxUaA

Link to complete code repository: https://git.chalmers.se/veau/smartexo_smoothmotion.git

DEPARTMENT OF MACHINE ENGINERRING
CHALMERS UNIVERISTY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

CHALMERS