CHALMERS
UNIVERSITY OF TECHNOLOGY

Sensorized Exoskeleton For Therapy Gamification

A Framework For Engaging Knee Rehabilitation Through Data-Driven Serious Games

Bachelor's thesis in Electrical Engineering
(EENX16-VT25-04)

Ludwig Andersson, Teodor Franzell, Kaspar Lifmark,
Jeremiah Terdin, Sebastian Wang, Pontus Åkerman

DEPARTMENT OF ELECTRICAL ENGINEERING
DIVISION OF SIGNAL PROCESSING AND BIOMEDICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

BACHELOR'S THESIS 2025

Sensorized Exoskeleton For Therapy Gamification

A Framework For Engaging Knee Rehabilitation Through Data-Driven Serious Games

Ludwig Andersson, Teodor Franzell, Kaspar Lifmark,
Jeremiah Terdin, Sebastian Wang, Pontus Åkerman

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2025

Sensorized Exoskeleton For Therapy Gamification
A Framework For Engaging Knee Rehabilitation Through Data-Driven Serious Games
Ludwig Andersson, Teodor Franzell, Kaspar Lifmark, Jeremiah Terdin, Sebastian Wang, Pontus Åkerman

Supervisor: Fabian Just and Emmanuel Dean, Division of Systems and Control, Electrical Engineering
Examiner: Emmanuel Dean, Senior Researcher Division of Systems and Control, Electrical Engineering

Bachelor's thesis 2025
Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Group member playing serious game "Rehab Flappy" controlled by sensorized exoskeleton

Typeset in LaTeX
Gothenburg, Sweden 2025

Sensorized Exoskeleton For Therapy Gamification
A Framework For Engaging Knee Rehabilitation Through Data-Driven Serious Games
Ludwig Andersson, Teodor Franzell, Kaspar Lifmark, Jeremiah Terdin, Sebastian Wang, Pontus Åkerman
Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology

Abstract

Rehabilitation of knee injuries is a demanding process, often hindered by repetitive exercises and subjective progress assessments, leading to patient demotivation and prolonged recovery. This bachelor's thesis addresses these challenges by proposing a sensorized exoskeleton integrated with a serious-game-like application. The aim is to show the potential for future development of therapy gamification of knee injuries.

The project combines wearable sensor technology with a serious-game-like application to simulate rehabilitation. A lightweight, 3D printed exoskeleton prototype equipped with an angular encoder, an IMU, and EMG sensors was developed to collect real-time kinematic data. This data is processed via micro-ROS and ROS 2 for joint angle and movement analysis, as well as muscle activity. It is then transmitted to a Unity game. In this environment, knee angle measurements drive the primary "Exo-Slide" therapy game, while EMG threshold crossings control the "Rehab Flappy" adaptation game. Gamification elements such as visual feedback and goal-oriented tasks were incorporated to motivate patients and ensure adherence to prescribed exercises.

Results showed that it is possible to use a sensorized exoskeleton to accurately control a serous-game-like application, which is promising for the development of actual serious gaming integration. Future work includes gamifying more rehab exercises, expanding sensor types, and refining the gamification framework for broader applications.

This project highlights the transformative role of interdisciplinary engineering in healthcare, wearable robotics, and bridging biomedical innovation with patient-centered game design to redefine modern physiotherapy.

Keywords: Serious gaming, wearable sensors, exoskeleton, ROS 2, Unity, physiotherapy, patient engagement.

Acknowledgements

We would like to express our deepest gratitude to everyone who contributed to the successful completion of this thesis. First and foremost, we extend our sincere thanks to our supervisors Fabian Just and Emmanuel Dean, for their guidance, constructive feedback, and expertise throughout the project.

We are grateful to Sahlgrenska Hospital for their collaboration and insights into the practical challenges of physiotherapy and rehabilitation. Special thanks to Anders Björk and Niklas Larsson for taking their time to discuss the medical parameters and procedures in rehabilitation. Their real-world perspectives helped us align our technical solutions with the needs of medical professionals and patients alike.

AI-usage

Usage of Artificial intelligence was limited to improving, reformatting and rephrasing part of the thesis. As well as assisting in finding relevant sources and code debugging.

Ludwig Andersson, Teodor Franzell, Kaspar Lifmark, Jeremiah Terdin, Sebastian Wang and Pontus Åkerman
Gothenburg, May 2025

Nomenclature

Below is the nomenclature featured in the report:

ACL	Anterior Cruciate Ligament is a strong ligament that connects your thigh bone to your shinbone.
Baud rate	Defines the speed of data transmission in bits per second between devices.
EMG	Electromyography measures muscle response or electrical activity in response to nerve stimulation of the muscle. The stimulation is measured by a pair of electrodes, called a channel, that are placed centred along the desired muscle. A reference sensor is placed on a bony part of the skin to reduce noise.
Encoder	An encoder converts angular position to a digital or analogue value. By using algorithms, this can accurately display angle and angular velocity.
Fusion 360	A cloud-based computer-aided design (CAD) 3D-modeling software.
GameObject	A fundamental entity in game engines that represents any object in a scene, such as characters, items, or cameras.
GUI	Graphical User Interface is a type of user interface that a user can interact with through, for example, graphical icons or visual indicators.
IMU	Inertial Measurement Unit. IMUs consist of accelerometers, gyroscopes, and sometimes magnetometers to measure rotation and movement.
Microcontroller,	A microcontroller is a single-chip computer that can handle signal processing like motor control or sensor measurements.

Node	A modular process in ROS that performs a specific task such as sensor processing and communicates with other nodes via topics.
PCB	A Printed Circuit Board uses layers with conductive materials to connect electrical circuits on a flat surface.
PlatformIO	An open source system for embedded development that simplifies coding for microcontrollers.
PlotJuggler	A open-source time-series visualization tool for robotics and engineering, allowing users to plot data.
ROS	Robot Operating System is a collection of software libraries and tools that help build robot applications.
Serial data	Data that is sent one bit at a time over some communication channel
Serious Game	A game where entertainment isn't the main purpose, but rather rehabilitation or research purposes.
Topic	A named communication channel in ROS used for a one-way data exchange between nodes via publish-subscribe messaging.
UI	User interface, collection of visual elements for communication between a user and the computer.
Unity	A multiplatform game engine used to develop many types of games.

Contents

List of Figures

List of Tables

1
Introduction

Rehabilitation plays a critical role in recovery after injuries, particularly when it comes to restoring mobility and function in joints such as the knee. During rehabilitation, patients may struggle with the process due to repetitive exercises and a lack of motivation. This often leads to low commitment, longer recovery times, and less-than-ideal results. One study found that compliance in post-operative rehabilitation programs is a key factor in rehabilitation outcomes [1].

Recent advances in technology offer new ways to address these challenges. One promising approach is the integration of gamification into rehabilitation, using game elements to make therapy more stimulating and interactive.

This thesis presents the development of a system that combines both concepts: a sensor-equipped exoskeleton paired with a serious game for knee rehabilitation. The system is modeled around patients recovering from knee ligament injuries, specifically anterior cruciate ligament injuries, which often require long and demanding rehabilitation programs.

With the collaboration of Sahlgrenska University Hospital in selecting exercises for knee injuries, a basis for developing relevant input data and a basis for a game. By combining sensor data with gamification, this project aims to contribute to the evolving field of technology-assisted rehabilitation, making therapy more effective, data-driven, and more engaging for patients.

1.1 Background

A major obstacle in physical therapy is sustaining patient interest throughout the rehabilitation process. Although standard programs are clinically effective, they depend heavily on patient discipline and subjective feedback. This can lead to inconsistent participation and make it difficult for clinicians to monitor improvements or adapt treatment strategies effectively. Active involvement of the patient in the setting of goals and the choice of exercises to perform has been shown to improve patient outcomes [2].

A good example of this is Anterior Cruciate Ligament (ACL) injuries. Improving rehabilitation protocols for patients with ACL tears is important due to the high rate of injury and long-term consequences. In Sweden, approximately 8,000 people

suffer an ACL tear each year, with about 4,000 undergoing surgical reconstruction. However, 20% of operated patients require additional operation or operations due to complications or insufficient recovery. Without proper rehabilitation, many young patients and others risk suffering long-term instability and are at risk of developing knee osteoarthritis within 10 to 15 years. Optimized rehab protocols can help reduce repeat injuries, improve quality of life, and minimize societal healthcare costs [3].

To address these challenges, researchers and developers have increasingly explored gamification as a way to improve motivation and participation. Serious games, a specialized form of gamification, have shown considerable promise in healthcare, particularly within rehabilitation programs. Studies have shown that game-based therapy can promote active participation, produce immediate feedback, and offer a more enjoyable experience for patients [4]. For adults, these interactive tools have been found to enhance feelings of progress, minimize boredom, and increase willingness to continue with therapeutic exercises.

From a systems and technology perspective, the integration of intelligent and responsive digital tools into healthcare practice is part of a larger shift toward more adaptive and patient-centered care models. The development of serious games for rehabilitation is supported by advances in sensors, computing platforms, and software architectures that enable real-time data collection and personalized feedback [5]. These tools contribute not only to a more engaging patient experience but also to more accurate data-driven decision-making for therapists and clinicians.

1.2 Purpose

Exploring the possibility of combining sensor-based data and games, establishes a foundation for improved motivation during exercises and the overall rehabilitation experience for individuals recovering from knee injuries. This project seeks to explore the possibilities of increasing patient engagement by developing a system that enhances interactivity, feedback, and the overall experience of rehabilitation exercises.

Using sensors to capture real-time movement data and incorporate it into interactive game mechanics, the system aims to encourage more active participation and create a sense of progress and achievement during therapy.

1.3 Goals

Developing a prototype that tracks user movements during rehabilitation exercises, demonstrating the possibilities of gamification within rehabilitation. This involves creating a system that captures real-time movement data using sensors and converts said data into interactive input within a game-based application developed in Unity. The system is intended to simulate a rehabilitation experience that could possibly promote user participation and help visualize progress through interactive feedback.

1.4 Delimitations / Demarcations

There were several constraints that influenced the scope and possible outcomes. The project was carried out with a limited budget, resulting in compromises with respect to hardware selection and software development. Due to ethical and logistical considerations, the system would not be tested on real patients; instead, validation was performed using simulated movements. Although this approach provided preliminary information, it may not fully reflect the performance of the system in clinical settings or with diverse patient conditions.

The scope of rehabilitation exercises incorporated into the system was narrowed due to time limitations, neglecting certain exercises that could be relevant for comprehensive therapy. Furthermore, the project did not assess long-term usability, patient adherence, or adaptability to different stages of rehabilitation. Hardware limitations, such as sensor accuracy and robustness during prolonged use, were also not thoroughly investigated.

In order to delimit the project, it only focuses on the rehabilitation of a specific knee injury, ACL injuries, or tears. Sensors and games were limited by rehab exercises for ACL injuries, recommended by Sahlgrenska Hospital [6]. For the full exercise list, see the Appendix A.

These limitations highlight areas for future refinement, including expanded clinical testing, integration of a larger exercise library, and extensive collaboration with medical professionals to enhance practical applicability.

2
Theory

Diving into the theoretical foundation of the technologies and methods used in the project. Exploring the role of gamification in improving patient motivation and increasing their commitment to rehabilitation programs. Discussing the use of exoskeletons to support recovery through sensor feedback and showing the importance of selecting an appropriate sensor sampling frequency that ensures accurate motion tracking and data. Together, these components form the basis for a data-based rehabilitation system that combines physical support with interactive game therapy.

2.1 Gamification in Rehabilitation

Implementing a game can be very enjoyable and increase patient motivation to continue rehabilitation. This can be achieved with elements such as points, challenges, and other forms of data, which promote a sense of improvement. The following study compares the literature that explores game elements used in the healthcare industry [7]. Common game elements, which were mentioned earlier, were shown to improve patient engagement and adherence to treatment. The implementation of this is explained in Section 3.2.

However, several challenges were also identified, including privacy concerns, declining user interest over time, and opportunities for cheating. There is room for improvement, and therefore, it is necessary to make technological improvements to possibly address these issues. When creating a game, it is necessary to take into account the risk that not all patients have the same gaming experience, physical ability, or cognitive skills. Designing the game to be too complex or visually intensive could potentially overwhelm the user.

2.2 Measuring Data from ACL Exercises

The first part of the project was to decide what exercises it should encompass, as well as how they could be implemented to control a game. The exercises selected were the ACL injury rehabilitation exercises recommended by Sahlgrenska Hospital [6]. Exercises 1 and 2 involve muscle contractions without moving the leg. Exercise 1 can be measured through a two-channel EMG (Electromyography) sensor placed on the calf: one channel for extension and one for flexion. Exercise 2 can be measured with a single channel placed on the thigh, see Section 3.1.3. Exercises 3 and 4 involve knee movements and can be measured by the encoder, see Section 3.1.1.

Exercises 5-7 involve straight-legged movements and can be measured by the IMU, see Section 3.1.2. Exercises 3-5 were used to control "Exo-Slide", and exercise 2 was used to control "Rehab Flappy". See Section 3.2 for more details. For a full exercise list, see Appendix A.

2.3 Exoskeletons in Rehabilitation

One of the main advantages of knee exoskeletons is the ability to integrate sensor-based feedback loops, allowing real-time monitoring of joint angles and motion quality [8]. The design of the exoskeleton is described in Section 3.1.5. When these systems are combined with an EMG, exoskeletons can interpret user intent more directly. This allows the device not only to assist movement but also to react in sync with the patient's muscle activation. This creates a more natural rehabilitation experience and can stimulate neuroplasticity.

2.4 Sampling & Frequency for Sensors

In this project, an exoskeleton is equipped with an encoder, an IMU, and an EMG sensor to monitor the patient during rehabilitation exercises. The accuracy and reliability of the data captured from these sensors depend on the sampling frequency chosen for the system. According to the Nyquist–Shannon sampling theorem, to reconstruct a signal without loss of information, the sampling rate must be at least twice the highest frequency present in the original signal [9]. See Section 3.1 for sensor implementation.

Applying this principle to rehabilitation means that the exoskeleton's sensors must sample at a rate high enough to capture the full dynamics of the patient's knee movement, including slow, controlled motions and faster, more abrupt changes in speed or direction. If the sampling rate is too low, critical information about the motion may be lost or misrepresented, which could affect both the real-time feedback delivered through the connected game and the quality of the data available for physiotherapist evaluation.

A higher sampling frequency not only enables smoother and more responsive game controls but also ensures that the recorded motion data accurately reflect the true joint behavior. This allows the physiotherapist to make well-informed clinical decisions based on objective measurements rather than subjective observations. Therefore, selecting an appropriate frequency, aligned with the expected range of patient movement speeds, is essential for the reliability and effectiveness of both the rehabilitation process and the game-driven engagement system.

A study by Fong underlines this, showing that robotic exoskeletons, when recording data at higher sampling rates (around 60 Hz compared to 30 Hz for external sensors), can capture more precise movement information, which is essential to understand joint performance and compensatory behavior during rehabilitation. They

also note that high-frequency data is useful for developing patient-specific exercises and tracking therapy progress over time. However, the study also highlights that the accuracy of the sensor depends not only on frequency but also on the correct calibration and physical alignment between the exoskeleton and the patient's body [10].

2.5 Robot Operating System (ROS)

The decision to use Robot Operating System (ROS) for this project is due to it being a widely used system for robotics middleware and a recommendation from our examiner. ROS is not an actual operating system but an open-source middleware framework for robotics. Providing a standardized set of software libraries and tools that help developers integrate sensors, actuators, and algorithms in robot applications [11]. During the past decade, ROS has become a reference platform in robotics, widely adopted in research and industry due to its rich ecosystem of reusable packages [12]. See Section 3.3 for the implementation of ROS in the system.

ROS has two main versions: ROS1 (the original release) and ROS2 (a major redesign). ROS2 was developed to overcome the limitations of ROS 1, especially the lack of real-time performance and the difficulty in multi-robot systems [11]. This project was developed with ROS 2 Jazzy.

2.5.1 General Concepts in ROS

In a ROS system, the computation is divided into independent modules called nodes. Each node is a separate program (process) that performs a specific task, such as reading a sensor, running a control algorithm, or commanding an actuator [13]. Nodes communicate by sending or receiving messages. Messages are typed data structures (for example, a velocity command or a sensor reading) that are published on named topics. A topic acts like a named bus or channel. One node can publish a message to a topic, and other nodes can subscribe to that topic to receive the data. This publish–subscribe model decouples the sender and the receiver. The publishing node does not need to know which nodes are listening, only the topic name [11]. The implementation of ROS is explained in Section 3.3.

2.5.2 Micro-ROS Implementation

For very resource-limited devices (such as simple microcontrollers on wearable sensors or actuators), even ROS2 can be too heavy. Micro-ROS is a variant of ROS2 designed specifically for tiny embedded systems. Bridging the gap between resource-constrained microcontrollers and larger processors in robotics applications [13]. In other words, micro-ROS puts the ROS2 stack on microcontrollers. This fits the project with the need to process sensor data in some sort of light and portable processing unit.

From a developer's point of view, a micro-ROS node behaves much like a regular ROS2 node (you write code that publishes and subscribes to topics), but under the hood, it is designed for an embedded environment [13]. This is very suitable for wearable exoskeletons or rehabilitation robots, where sensors and motors might be controlled by microcontrollers: micro-ROS allows those controllers to communicate seamlessly with the higher-level ROS2 system. It is very practical in this project because this means that the data is integrated into the ROS environment very early, which creates a clear and structured data architecture. The implementation of micro-ROS is explained in Section 3.3.2.

3
Methodology

The workflow followed a methodology inspired by the principles of Agile Project Management [14]. The project was initialized with a planning phase in which the path to the goal was defined by dividing it into modules and breaking it down into actionable tasks.

The workload was split into 3 subgroups: Hardware, Game Development, and Data Integration. By simulating dummy input game controls, the game could be developed in isolation from the rest of the project. This enabled the simultaneous and independent development of the hardware, ROS, and the games. These parts were integrated into the final phase of the project. Figure 3.1 shows the workflow concerning time. Note that the direction of time and data flow through the three subgroups' work does not align. The ROS system architecture was developed throughout the project but only implemented as the glue between the sensorized exoskeleton and the serious game application.

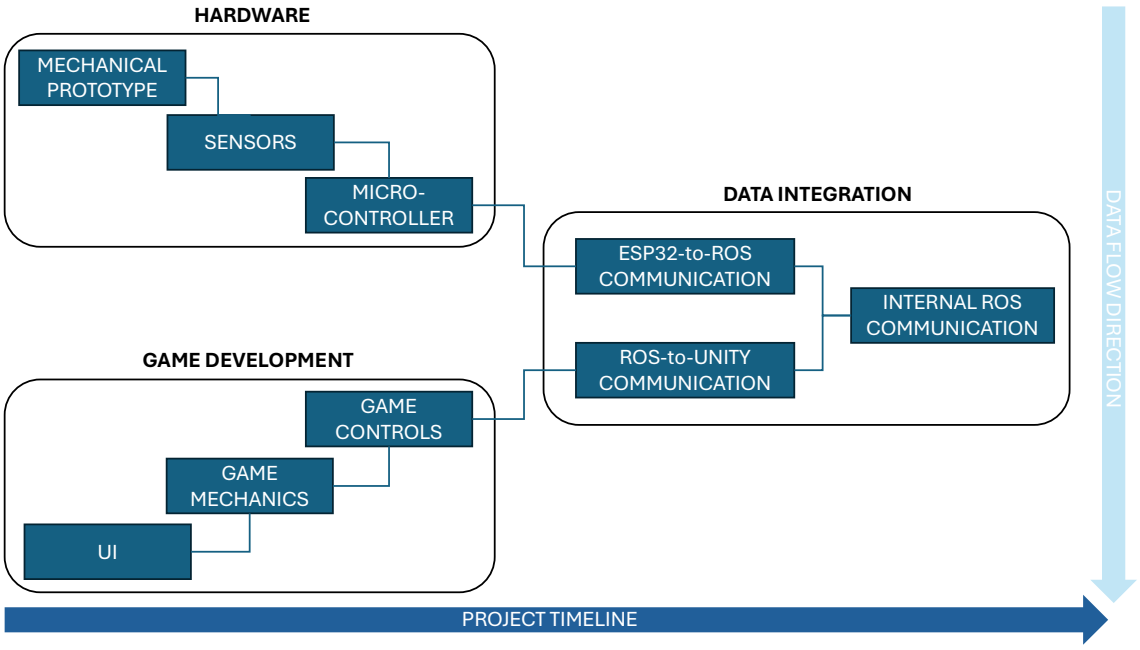


Figure 3.1: Project work across three subgroups: Hardware, Game Development, and Data Integration. The flowchart describes the workflow concerning time when observing it from left to right. However, when observing it from top to bottom, the flowchart describes how data travels through the final system of the project.

Figure 3.2 shows a simplified flow chart that describes the data flow in the final end-to-end system. This will be described in detail throughout this chapter.

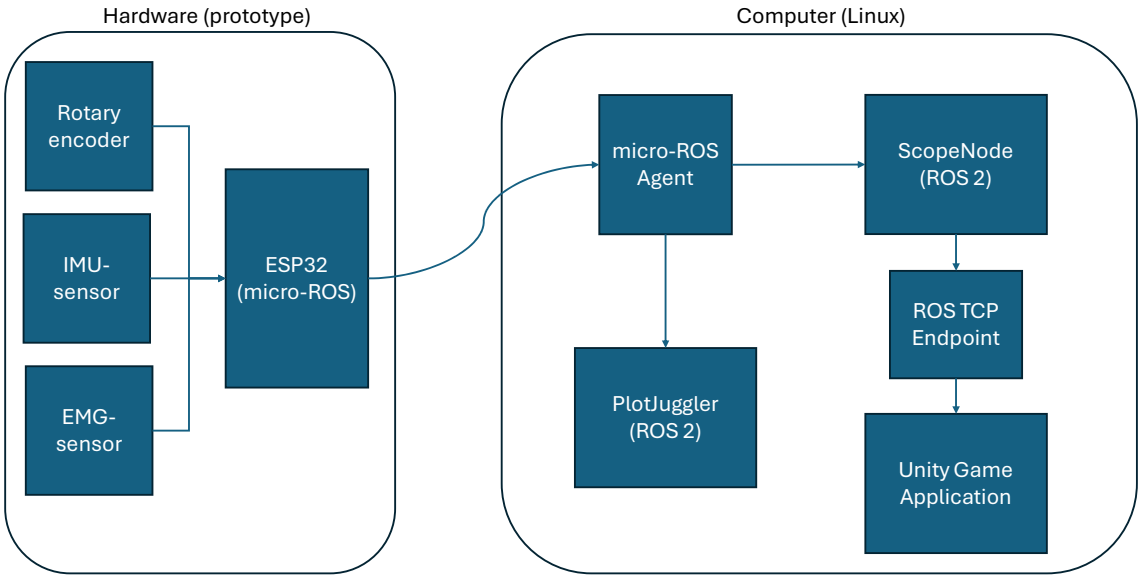


Figure 3.2: End-to-end system data architecture. The flowchart describes how the data flows through the system. It starts at the sensors that sense the information, the ESP32 processes the sensor data and sends it to the computer via a USB cable. Inside the computer, the data goes to the Unity application, which controls the games, or PlotJuggler, where the data can be visualized and analyzed.

3.1 Hardware Setup

The core of the hardware setup is the ESP32 WeMos LOLIN32. It is a microcontroller chosen for its flexibility and performance in handling real-time sensor data. It was programmed using PlatformIO and connected to the host computer through a micro-USB cable. This ensured that there was stable and consistent communication without the need to configure wireless protocols, which could introduce additional issues during initial testing.

3.1.1 Encoder Implementation

For exercises requiring knee movements, an AMT103-V rotary encoder was selected to measure the knee angle. The sensor was tested and calibrated by comparing the angle output with the measured angles. See Section 3.3.1.1 for more details. The AMT103-V was selected because of the good balance of cost and function.

3.1.2 IMU Implementation

For exercises requiring leg movements with a straight knee, data were collected with an Adafruit 9-DOF IMU. Providing complementary orientation data by combining accelerometer, gyroscope, and magnetometer readings. It was primarily used to

measure the calves' angle compared to the gravity of the Earth. Laterally and anteriorly. The sensor was tested by comparing the angle output with the measured angles. See Section 3.3.1.1 for more details.

3.1.3 EMG Sensor Implementation

In addition to the sensors mentioned above, a pair of MikroElektronika EMG Click, provided by our Examiner, was used to measure muscle signals, using EMG (Electromyography) from a maximum of 2 channels. For each channel, there are 3 electrodes. One is a reference sensor, placed on a bony part like the knee, and the other two are placed centered along the desired muscle. The acquired data was filtered, rectified, and later thresholded, so that it would send a signal after a high enough amplitude was measured. See Section 4.2 for more details.

3.1.4 Custom Designed PCB

In order to mount these components in a clean and stable way, Printed Circuit Boards (PCB) were designed and connected via flat cables. This was made to ensure that the electronics had a lower risk of breaking or disconnecting.

3.1.5 Construction of the Exoskeleton

The design of the physical components of the exoskeleton was designed using Fusion 360, a cloud-based CAD modeling software [15]. Then 3D printed using PLA plastic with Bambu lab A1-mini 3D printer, for printing specifications, see Table 3.1.

Table 3.1: Printing specifications for exoskeleton parts, such as infill, outer layers, and the material used.

Parts	Infill [%]	Outer layers	Material
Rotational parts	30	5	PLA
Mounts	30	3	PLA
Connections	15	3	PLA
Rotational Axis	50	5	PLA

The components were later put together using M5 bolts and nuts, creating a highly modular design, which made it easy to replace components if needed. The exoskeleton has one degree of freedom located above the knee joint, mimicking the movements of the actual knee. The exoskeleton rotates using two separate parts, one representing the lower leg and the other the thigh. These are connected using four circular plastic layers (two per rotating part) interlinked to form a stable joint with a single degree of freedom. For easier visualization, see Figure 3.3.

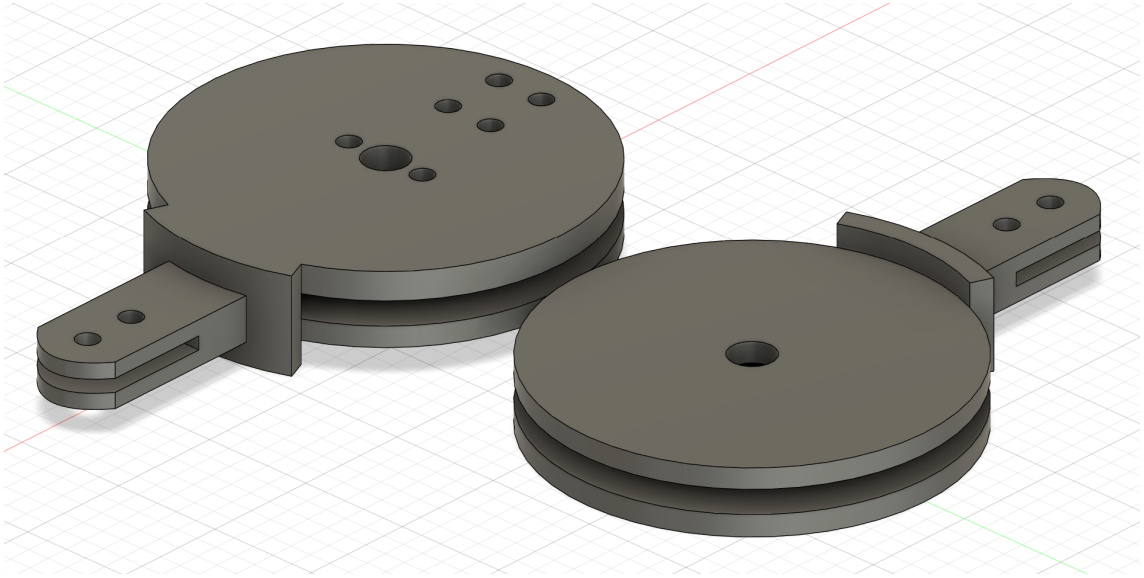
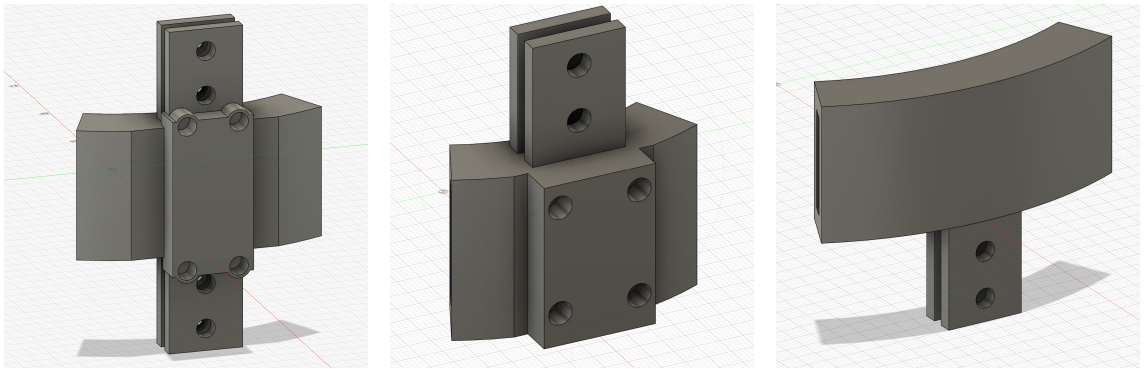


Figure 3.3: Lower and upper rotational CAD parts. Together, they create a single joint mechanism that moves together with the knee. They are connected using the rotational axis.

In addition, parts were added for easier mounting on the leg itself; the thigh, mid calf, and ankle, see Figure 3.4. These parts were hollowed out during the printing process to allow the Velcro tape to fit through, allowing it to attach to the leg. All components also contain holes for M3 threaded sleeves that match the dimensions of the PCB design, making it possible to attach the PCB to the prototype, see Figure 3.5.

The overall design philosophy revolved around keeping it simple, as the exoskeleton at this stage does not need to support an injured leg. It serves simply as a mounting place for the relevant sensors. However, it should still be comfortable enough and portable enough to allow testing and gamification to take place.



a Mid-calf mount. **b** Ankle mount. **c** Thigh mount.

Figure 3.4: Mounts for electronics as well as loops for Velcro straps. They connect to the rotational parts and each other with the help of the connectors.

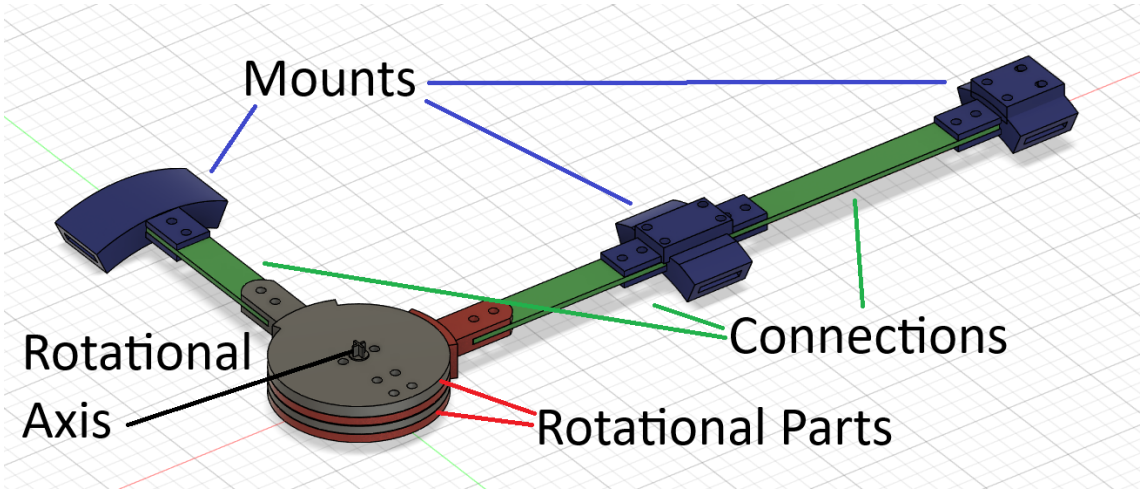


Figure 3.5: Fully assembled CAD model of the exoskeleton prototype. Here you can see how all the parts connect to each other.

3.2 Game Development

The name of the primary game is "Exo-Slide", and this part will go over the game development process. The game was developed in the free community version of the Unity game engine, which is relevant when the goal is to integrate robotics, as there are many assets that simplify communication between the two.

Beforehand, the decision was made to keep the game simple, as determining the time required to develop a complete game proved difficult. Therefore, the game was developed around a single input from a sensor, and based on the prototype and exercises, the angle input was selected, making it ideal to have a game with some type of slider movement that could mimic a difference in angle when moving a leg, for example. The fully integrated game is tested in Section 4.1.1.

3.2.1 Game Objectives

The objectives in the game are for the player to avoid as many obstacles as possible without hitting any of them, while simultaneously collecting coins. The concept of avoiding obstacles was chosen because it could work as a good measure of a player and their control over their own body. An additional incentive has been shown to improve engagement [7]; therefore, coins were added to the game. A point system and feedback help with motivation in games, to add this in a fun and frequently used way, coin collection was implemented. More about this can also be read in Section 2.1.

There are three different types of obstacle, each signifying a certain difficulty, and each has a different color, but also a different gap, making it harder for the player to completely avoid hitting it, without having good control and precision. Figure 3.6 shows each of the different difficulties in the game, where each color signifies the difficulties: green (easy), blue (medium), red (hard). Since this is only a proof-of-

concept, the difficulties are not tested but are only for showcasing.

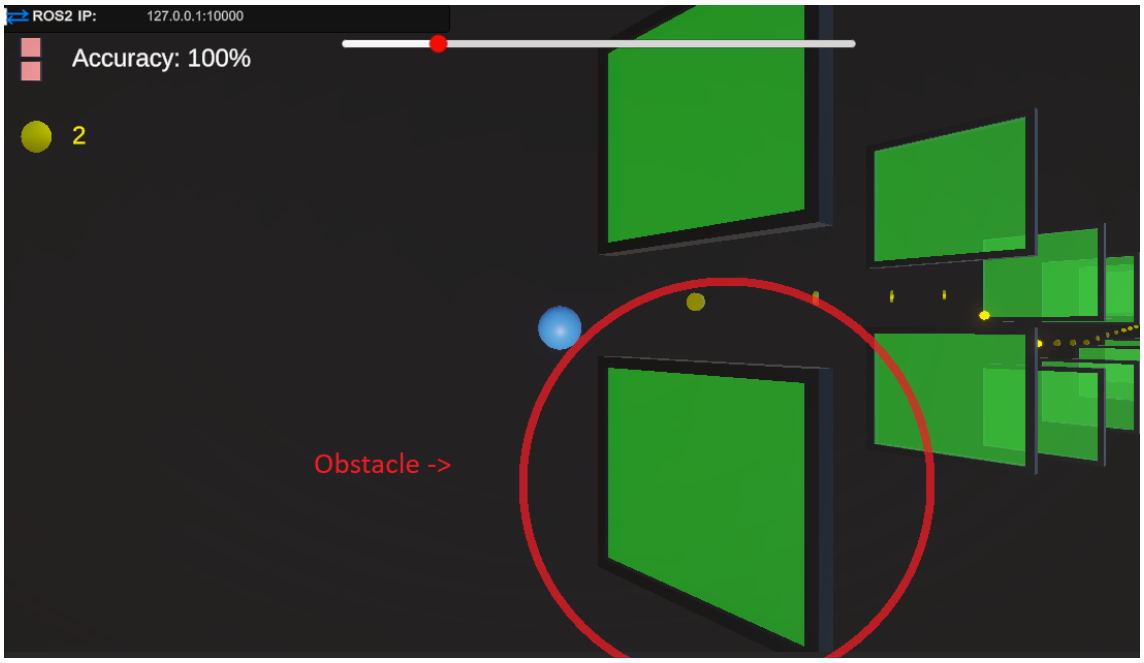


Figure 3.6: This is an obstacle for the easy difficulty in the game. That means the gap is at its largest and is the easiest for the player to avoid.

3.2.1.1 Obstacle Functionality

The obstacles are composed of two parts, the upper wall and the lower wall, and each spawn location is calculated with a preset gap depending on the difficulty that the obstacle should have. The number of obstacles and their respective locations are determined by a list of values, which would signify an imported data set for a plot or graph, for example. It does not matter what values each data point in the list has, since the script will calculate a corresponding normalized value between the max and min that the obstacles can spawn between.

3.2.1.2 Coin Functionality

Collectible coins appear between the gaps of each obstacle. They represent the straightest path between each gap. If the player moves well, they will catch many coins, but if their movement is erratic, they will catch fewer.

Each coin has a constant rotation put upon it to make it spin and feel more alive, and when hit by the player object, the coin will be destroyed, and the collected coin count will increase by one. Figure 3.7 shows the coins that are placed in the game in the gap between the obstacles and between each obstacle instance.

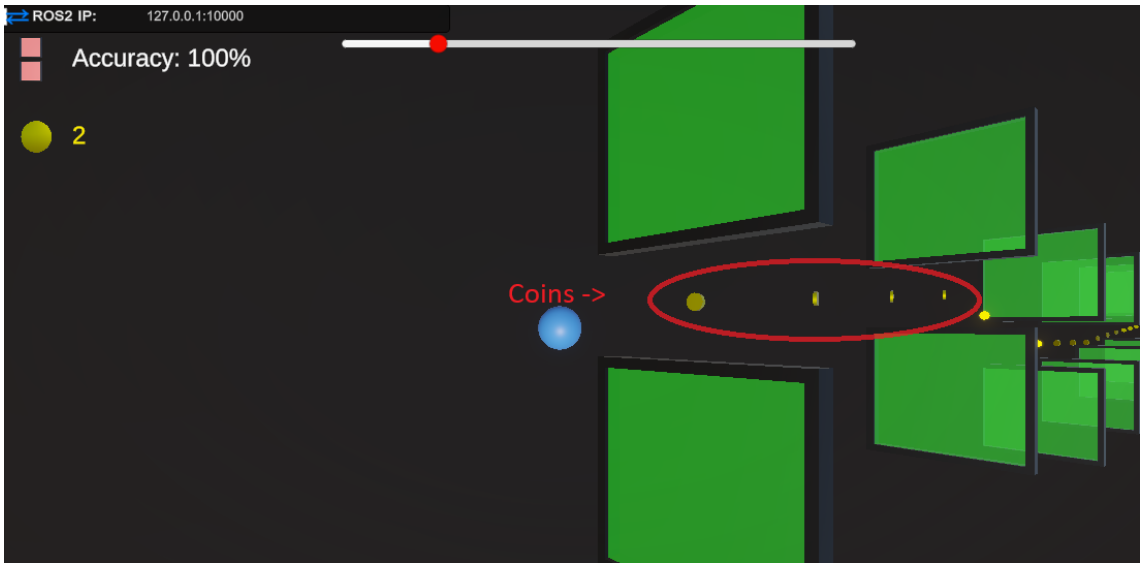


Figure 3.7: Coins are placed in the desired path of the player. When they are collected, the counter on the top left of the screen will increase by one. If the player catches very few coins, it could be the result of erratic movement during the game.

3.2.1.3 Player Functionality

Figure 3.8 shows the object of the player, which is what the player will control. The directions of movement are restricted to only up and down within a set interval. The shape of the sphere is chosen as a placeholder and can be changed for future applications.

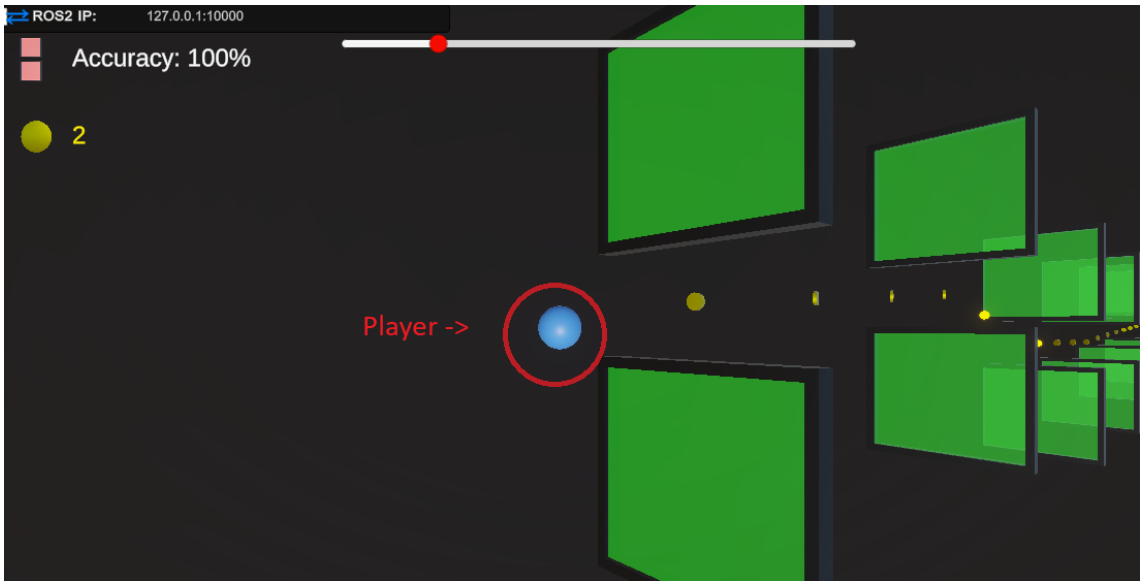


Figure 3.8: Player object in the game. It moves up and down based on the input. The player object will also blink when it hits an obstacle to signal a hit.

3.2.2 Designing the User Interface

When designing the the Exo-slide, it was very important to display meaningful information in the game at the same time while the game felt engaging and fun for the player. The UI (User interface) in the game was implemented to show the current accuracy of the player for the obstacle that was just passed or was just hit and display a percentage of how much of the player did not hit the obstacle; this is calculated to give feedback to both the player and the medical staff of how much control they have over their movement and reflexes. The other UI element is showcasing the amount of coins the player has currently collected in the game, which is to motivate the player to try and collect as many coins as they can.

Figure 3.9 shows the options menu in the game. Here, you can specify the minimum and maximum angle of motion that you are going to be using for your specified exercise. The difference in value written in the input fields will be used to normalize the input movement between the game bounds to make sure that the player does not go outside the desired range.

There are also three different choices for game speed; this is intended to give the game a bit more variety through speeding up the game in case it is important to measure the player's reaction speed, making the game more difficult.

Figure 3.9: Options menu for the game. At the top of the screen, you input the movement range of the knee. Underneath, you select the speed you want the obstacles to move at. When the desired settings are selected, you can start the game with the button furthest down.

Figure 3.10 shows the user interface while playing the game, also known as the Graphical User Interface (GUI), which shows the current accuracy of the player for the obstacle it just hit or passed, as well as the amount of coins that have been collected so far. This was accomplished by using the latest percentage of accuracy after passing an obstacle, and another larger collider was added to the player object to detect if the player passed by an obstacle without hitting it to display an accuracy of 100%.

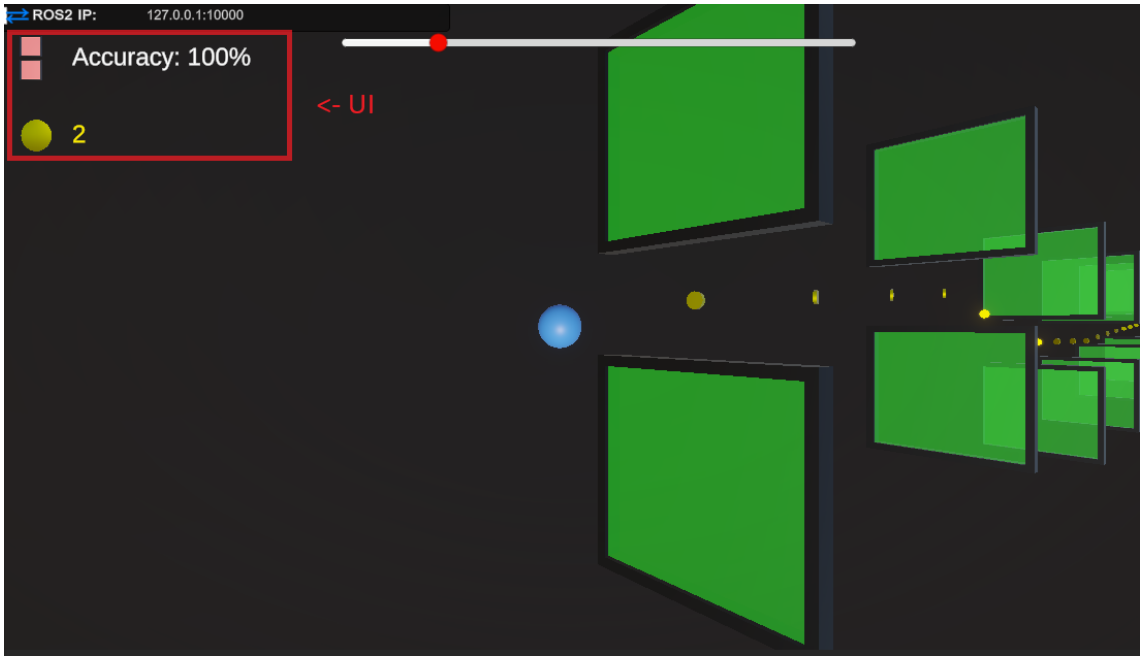


Figure 3.10: The user interface in the game is shown in the top left of the screen. It shows how accurately the player avoids obstacles as well as the number of coins collected.

To show how the player performed during the game, a game summary screen was placed. Displaying the average accuracy for the whole game and for each of the difficulties, as well as the amount of coins collected compared to the total amount. This was implemented primarily as an example of information that could be useful for medical professionals to see which part of the game the player found the hardest and to see how the player has been improving over game iterations.

The total average accuracy in the game is calculated by adding up the accuracy for all the obstacles that the player has passed in the game, and then taking the average value and displaying it. For each of the difficulty accuracies, it was calculated by first having each difficulty be one-third of the total amount of obstacles in the game, and then summarizing the accuracies for each third of the total amount of obstacles, and then taking the average for each third.

Other functions added to the game summary are the ability to restart the game, to go back to the options menu to change the speed of the game or to change the range of motion, or to go back to the start menu, depending on what action is desired.

Figure 3.11 shows the game summary screen and how the different accuracy and coin elements are displayed.

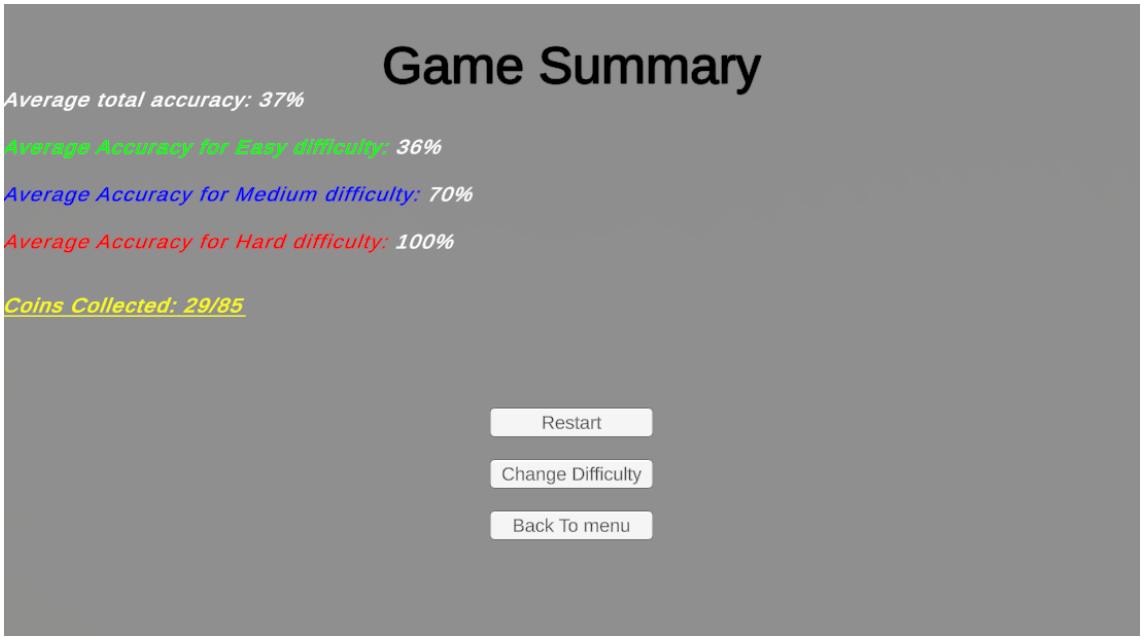


Figure 3.11: The game summary that pops up after every run. From top to bottom, it displays the average total accuracy, as well as the accuracy for each difficulty. It also displays how many of the total number of coins were collected. From here, you can either restart the game, change your settings, or return to the start menu.

3.2.3 Adaptation of Existing Game

In addition to the main game developed for this project, an adaptation of popular games on the market was developed in Unity with the help of a tutorial video with open source assets [16]. For this project, it is called "Rehab Flappy". This prototype serves as a proof-of-concept to demonstrate that an existing, engaging 2D game could be integrated with a rehabilitation device. The movement of the character is based upon a single true or false input that tells the game to either make the character flap or not. This was put in place since the game originally is based on a single button press to function. The fully integrated game is tested in Section 4.2.

To establish the role of 'Rehab Flappy' as a serious game and improve user data logging and sampling, a new menu screen was developed in the UI, including options for both new and existing users. The contrast levels of the background, the pipe obstacles, and the bird were optimized to improve visibility and support accessibility. Input fields for player names, a score display in the top left corner, and a leaderboard were implemented to support tracking and motivate user engagement.

The game is intended for rehabilitation exercises such as calf raises or static quadriceps contraction. It is currently functioning as a proof-of-concept, where "Rehab

Flappy" assigns Easy, Medium, or Hard modes based on the specified heel-raise height or contraction strength, see Figure 3.12 and 3.13 for game menus.

This approach highlights the flexibility and potential for a wider application, showing that it can support a variety of existing game environments for rehabilitation purposes.

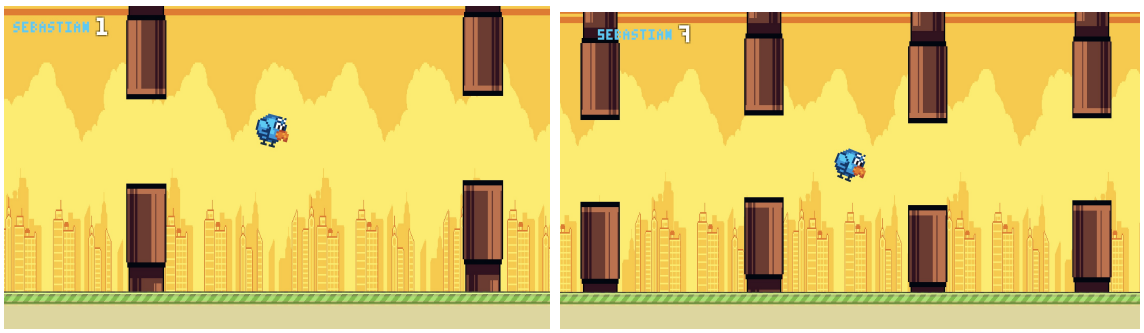
Figure 3.12: This is the Menu Screen, the first thing the player will see in the game. Here, you can choose to either create a new patient profile or select an already existing one.

Figure 3.13: This is the menu for creating a new patient profile. You enter a name as well as a manually entered value for max heel raise height, which sets the game difficulty and was intended for future IMU implementation.

3.2.4 Difficulty Settings

When it comes to in-game aspects, such as difficulty, Figure 3.14 (a) shows the easy difficulty in the game where the pipe obstacles have a long spacing between to give the player a long period before a reaction is needed to avoid them. Figure 3.14 (b)

instead shows the hardest difficulty in the game, where the spacing is much shorter between the pipe obstacles and requires a much faster reaction speed and control to avoid them.



a) Easy mode. **b)** Hard mode.

Figure 3.14: Comparison of Rehab Flappy game modes: a) Easy and b) Hard. When the difficulty increases, the intervals between obstacles decrease, leading to a higher frequency of muscle activations required to progress through the game.

3.2.5 Leaderboard Interface

After the player has hit an obstacle and thereby ended the game, a leaderboard screen appears as shown in 3.15, where the player can see a leaderboard placement and comparisons to old results to measure if there has been an improvement over time.

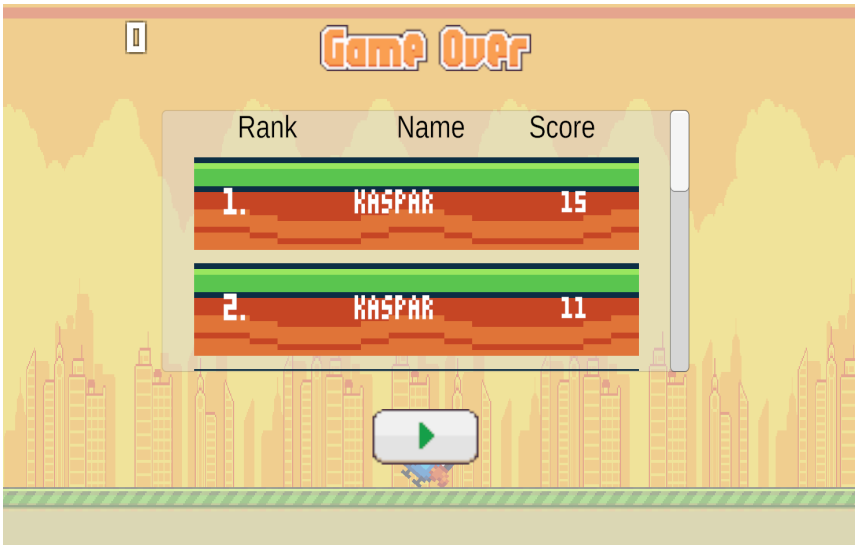


Figure 3.15: This is the leaderboard screen at the end of the game, where the player can view their highest scores achieved during the game. It provides a summary of personal bests to track progress over time.

3.3 Data Integration

The two previous sections described the hardware behind the sensorized exoskeleton prototype and the development of serious gaming applications. This section will focus on how the work of these two project segments communicates with each other, see Figure 3.16. The data architecture of this project is built on ROS. For more information on ROS, see Section 2.5.1.

Figure 3.16: Data from hardware runs through a ROS2 architecture to reach the game application. All communication between the hardware and the Unity game is built on ROS2. Hardware includes all sensors as well as the microcontroller.

ROS2 with Jazzy Jalisco distribution was used for the data integration in this project, more specifically. The ESP32 was configured with micro-ROS (see Section 2.5.2), allowing it to publish sensor data directly to ROS2 topics. These messages included joint angle readings from the encoder and motion orientation from the IMU, as well as EMG activity. Data were sent at a consistent rate, and topics were managed through ROS nodes on the host computer. All relevant sensor data were packaged into a single ROS-message, to reduce the number of unnecessary headers, simplify topic management, and improve synchronization.

This setup enabled seamless communication between the hardware and the higher-level applications. Sensor data could be visualized in real-time using tools such as PlotJuggler, helping with both debugging and movement analysis. The ROS2 environment also laid the groundwork for integration with Unity, as it served as the central hub where all data was collected and processed.

3.3.1 ESP32-ROS Communication

The ESP32-ROS communication pipeline was implemented using a WeMos LOLIN32 microcontroller programmed via PlatformIO, using the micro-ROS framework to create a bridge with sensor data and the ROS2 system, as seen in Figure 3.17. This setup was chosen because of its balance of real-time performance and development flexibility.

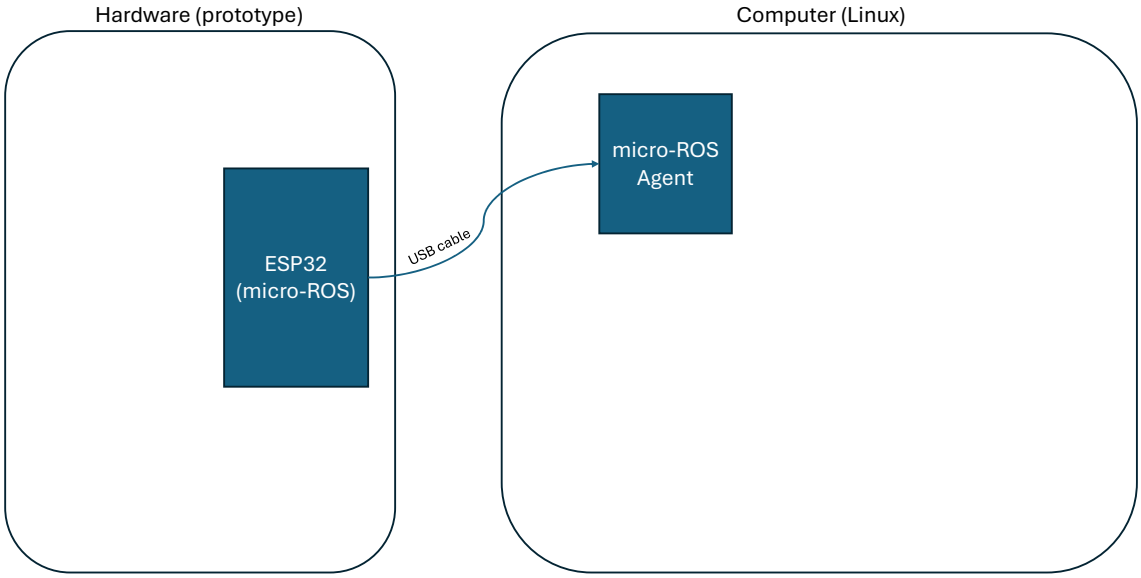


Figure 3.17: Sensor data that gets processed in the ESP32 microcontroller goes via a USB-cable into a computer. The microcontroller script integrates micro-ROS to be able to send the data in a standardized ROS message right into the computer.

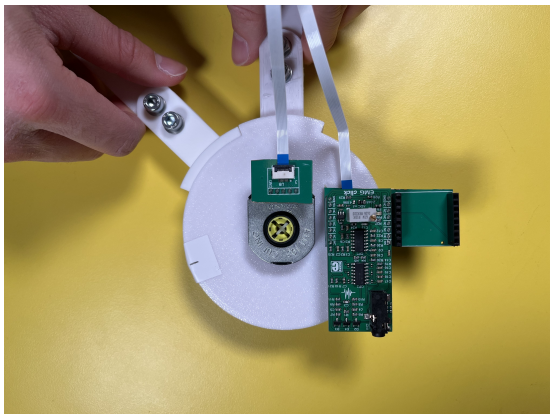
Initial prototyping with a Seeit ESP32 revealed inconsistent USB-C connectivity and firmware upload failures, which severely slowed development progress. After systematic testing, we traced these issues to an unstable USB connection on the microcontroller development board. As a countermeasure, we transitioned to the more reliable LOLIN ESP32 variant, which solved connectivity problems and provided a stable foundation for sensor integration. The final hardware configuration used a micro-USB connection at a baud rate of 921600 for data transmission, deliberately avoiding wireless protocols during this phase to eliminate potential latency variables and simplify debugging.

In the setup, we used PlatformIO for the integration with an included micro-ROS library and defined a custom message. The message contained several variables for all relevant sensor data outputs, which were compiled in both the ESP32 firmware and the host ROS2 workspace. This ensured consistent message formatting across the entire data pipeline. In the ESP32 firmware, we implemented a publishing node that was populated with our sensor readings and pushed to a topic. The message population occurred within the timer callback function, where processed sensor values were assigned to their respective fields before being published on the topic. A hardware timer configured with a 2-ms period triggered the publishing routine, ensuring consistent data transmission at roughly 170Hz. We believe that this number can be further improved for better communication and results.

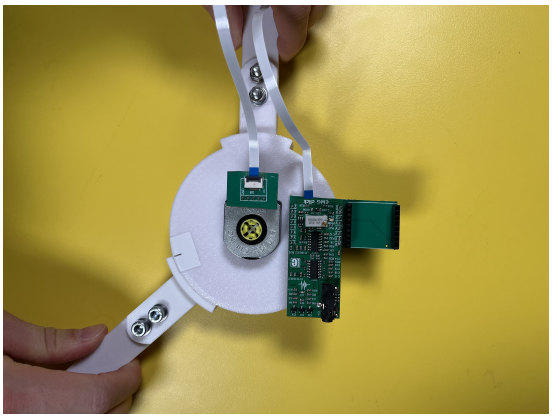
The validation of the communication pipeline was performed using standard ROS2 tools. The ROS2 topic echo command provided immediate verification of successful data transmission, while PlotJuggler allowed us to see different variables sent from the topic as graphs.

3.3.1.1 Sensor Validation

Experimental tests were performed in order to validate the data given by each sensor. The goal was to assess how accurately the devices performed under simple movement conditions. To validate the encoder, an angle of 0 to 90 degrees was measured using a contractor and marked on the prototype. The prototype was manually rotated back and forth to this mark, which can be seen in Figures 3.18 (a) and 3.18 (b). The encoder readings were then plotted, revealing a close match with minor deviations, demonstrating a generally reliable performance throughout the tested range, which can be seen in Figure 3.19. These tests ensure that the encoder provides reliable results before performing the test for the "Exo-Slide" game, shown in Section 4.1.1.



(a) Prototype joint at reference angle.



(b) Prototype joint 90° from reference.

Figure 3.18: Encoder-angle test setup to verify the accuracy of the encoder angle. The prototype joint was rotated between the two positions above, which were measured to be 90° apart. The movement was performed while the prototype was connected to the computer, and the data was plotted and captured in PlotJuggler.

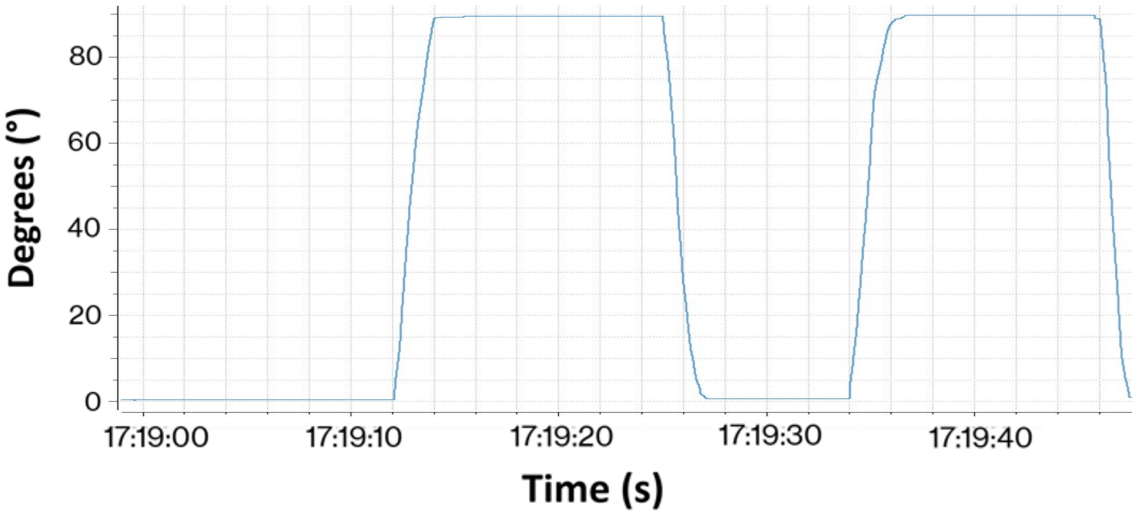
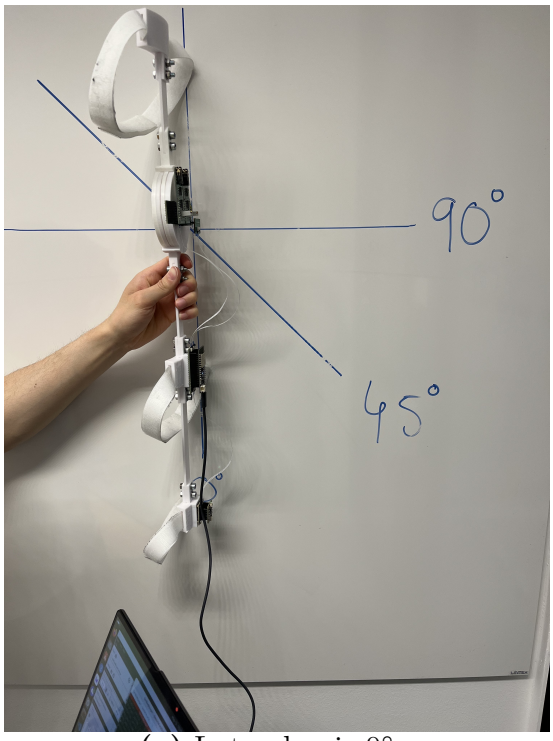
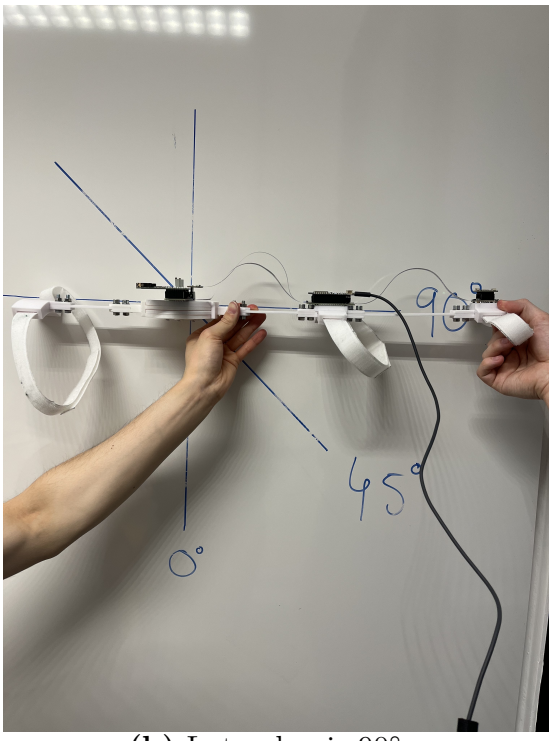


Figure 3.19: Encoder-angle data test results. As visualised by the graph, the encoder measures a movement between approximately 0° and 90°. This matches the physical measurements done in Figure 3.18.

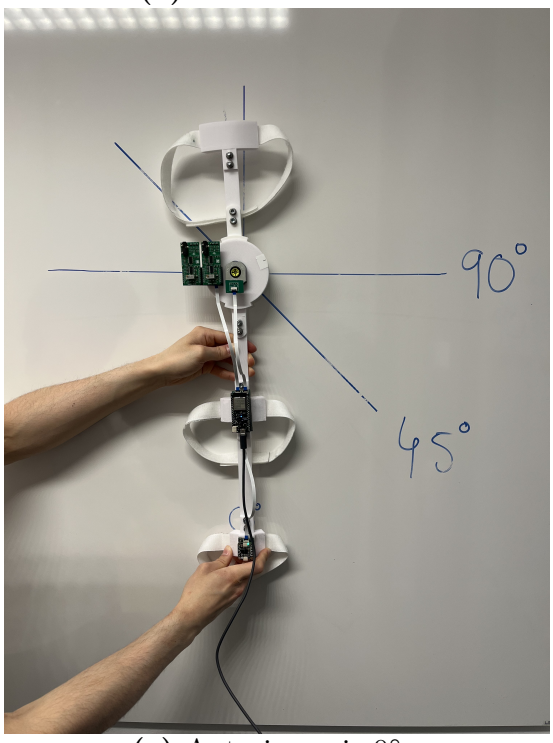
To verify the accuracy of the IMU tilt measurements, we performed manual prototype movement tests across the lateral and anterior axes. The evaluation spanned the range of 0° to 90°, which can be seen in Figure 3.20. Lateral and anterior tilt tests followed an identical protocol, but required special attention to mounting alignment. Using a contractor and a spirit level, we were able to draw lines corresponding to the desired tilting level and follow these lines as a reference for validation. It is important to acknowledge that multiple factors collectively contribute to the observed deviations, but the goal is to confirm the IMU's fundamental angle-sensing capability.
The IMU angle measurements demonstrate reliable tracking of the anterior and lateral tilts at positions 90 and 0 degrees, as shown in Figures 3.21 and 3.22. The lateral measurements follow the expected trend and show some variation, mainly due to human error during movement transitions. This validation approach provides sufficient verification for our target application.



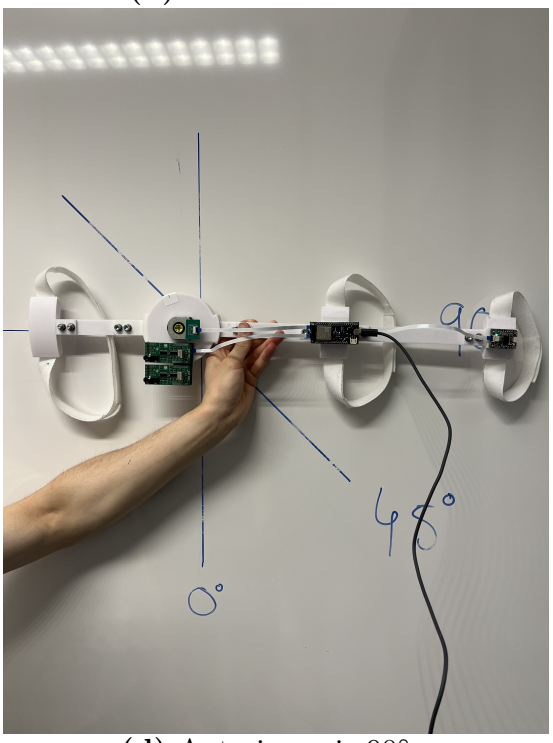
(a) Lateral axis 0°.



(b) Lateral axis 90°.



(c) Anterior axis 0°.



(d) Anterior axis 90°.

Figure 3.20: Angle test to verify IMU accuracy, both laterally and anteriorly.

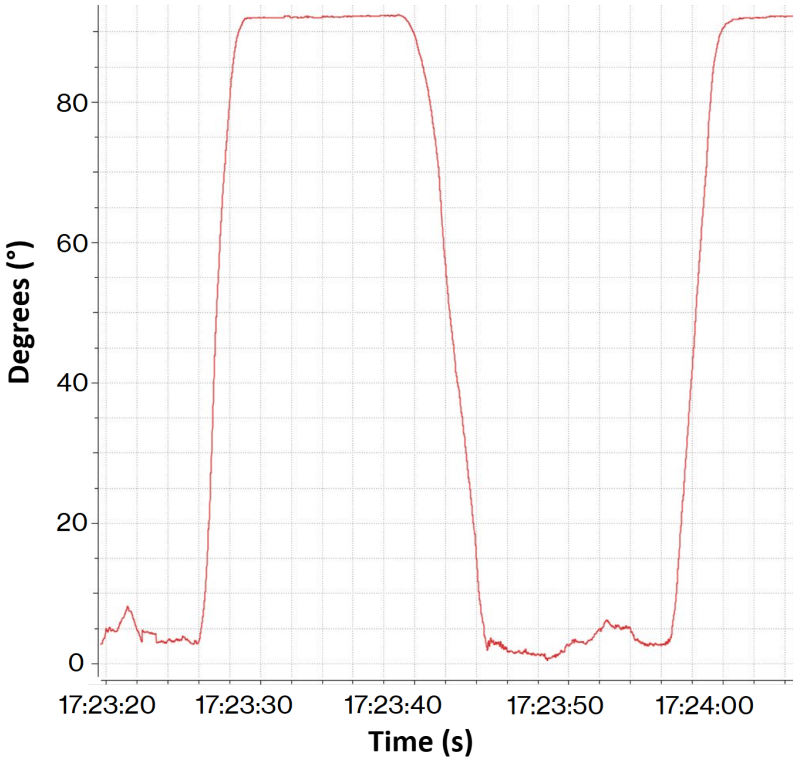


Figure 3.21: IMU-angle test plot across lateral axis. As visualized in the graph, the angles move between around 0° and 90°, which matches the movements in Figure 3.20 (a) and 3.20 (b).

Figure 3.22: IMU-angle test plot across anterior axis. As visualized in the graph, the angles move between around 0° and 90°, which matches the movements in Figure 3.20 (c) and 3.20 (d).

3.3.2 Internal ROS Communication

The ROS2 data path was designed with just a single node to efficiently bridge the ESP32 micro-ROS output with Unity's input requirements. The data path can be seen in Figure 3.23. At its core, this system relied on two custom message types, one for the communication between ROS and the microcontroller and the other between ROS and Unity. This made it very easy to create a simple structure and made it easy to swap desired variables sent to Unity.

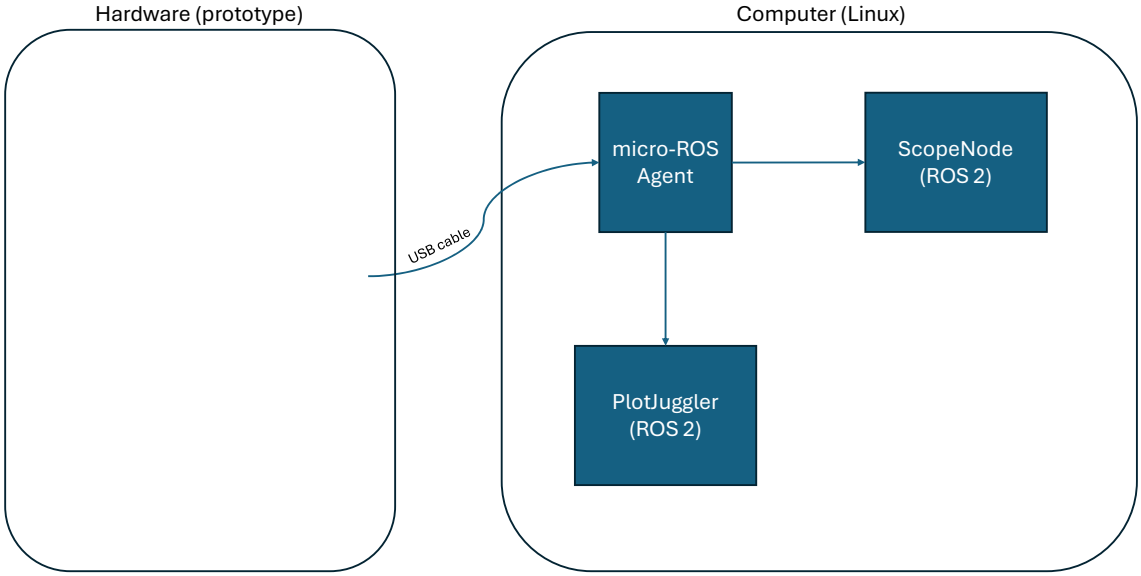


Figure 3.23: A micro-ROS Agent opens up the USB port in the computer and lets the hardware talk to the incoming topic. Inside the computer, the data is used for two things: to control the game (ScopeNode) or to be analyzed by being visualized in the ROS2 compatible plotting tool PlotJuggler.

The communication flow began with the micro-ROS agent receiving serial data from the ESP32 via USB and republishing it as an ROS2 topic. The custom node subscribed to the raw data topic that carried the complete message, while also publishing a filtered topic containing only the essential variables for Unity. This design decision to strip data from the message in the ROS-node rather than doing it in Unity or PlatformIO was easier. It allowed for the development of the hardware and game software in parallel, minimizing the need for frequent changes during development.

During the initial phase of ROS environment configuration, we implemented a dummy package to simulate sensor data streams before hardware integration was complete. This approach allowed parallel development of the game interface and data processing while waiting for the sensor-data configuration to be finished. For practical deployment, nodes were launched manually using ROS2 run commands during development. This allowed for rapid iteration and debugging. The entire stack from micro-ROS agent through to Unity's ROS-TCP-Endpoint ran on a single machine during testing, with all components using the default ROS2 middleware without special configuration.

3.3.3 ROS-UNITY Communication

To enable communication between a robotic system, running ROS2, and a simulation environment in Unity, the ROS-TCP-Endpoint framework is used, as seen in Figure 3.24. This integration allows real-time data exchange between ROS2 nodes and Unity GameObjects, allowing virtual testing and development of robotic applications. It is particularly valuable in scenarios where hardware is unavailable or when algorithms need to be tested in a safe, repeatable environment.

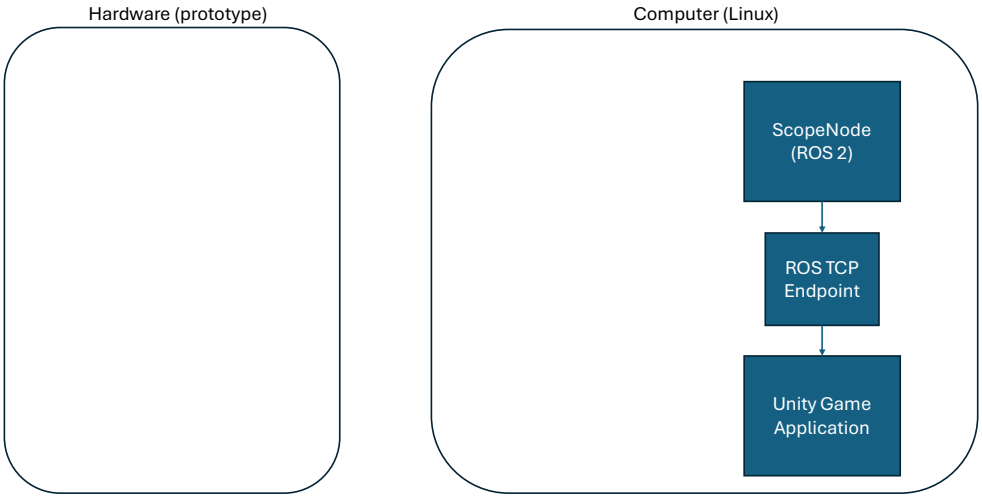


Figure 3.24: Data meant to control the game gets filtered in the ScopeNode and goes from the ROS environment to the Unity Game via a ROS-TCP-Endpoint. This is a ROS and Unity-compatible server that creates a bridge between the different systems.

The connection consists of two core components. On the ROS side, the ROS-TCP-Endpoint package provides a TCP server node that listens for incoming connections from Unity. It handles the serialization and routing of ROS messages, services, and actions. On the Unity side, the ROS-TCP-Connector package provides a ROS Connection component that establishes a client connection to the ROS-TCP-Endpoint. Through this connection, Unity can publish to ROS topics, subscribe to data, or request services.

For message compatibility, ROS message definitions (such as .msg files) are converted into equivalent C# classes using Unity's message generation tool. This ensures that both ROS2 and Unity understand the message structure, allowing accurate serialization and deserialization of data. Once configured, this setup enables Unity to act as a real-time interface for robotic systems, supporting both control commands and sensor feedback.

This method of integration is widely used in robotics research and development because it supports flexible system architectures. Developers can prototype and debug robotic behaviors in Unity using real sensor data from ROS or simulate virtual sensors and send data back to ROS, all over a single TCP connection. See Figure 3.24 for the flow of data.

4

Experimental Validations and Results

Presentation of the experimental validations conducted to assess the core functionality and precision of our sensitized exoskeleton prototype and the resulting hardware, system architecture, and game design. As a proof-of-concept project, the focus is on establishing basic technical performance, system integration, and usability metrics to guide future development.

Because two games were developed based on two different sensor data, this chapter will be split into two full integration sections, showing the respective sensor with its respective game. The two different games share the same general data flow architecture, which is visualized in Figure 4.1.

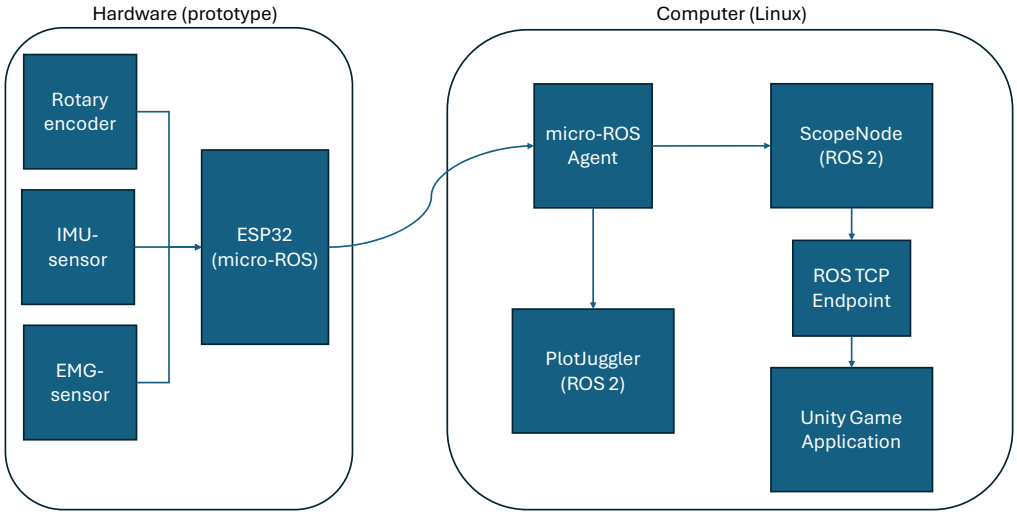


Figure 4.1: End-to-end system data architecture. The flowchart describes how the data flows through the system. It starts at the sensors that sense the information, the ESP32 processes the sensor data and sends it to the computer via a USB cable. Inside the computer, the data goes to the Unity application, where it controls the games, or it goes to PlotJuggler where the data can be visualized and analyzed.

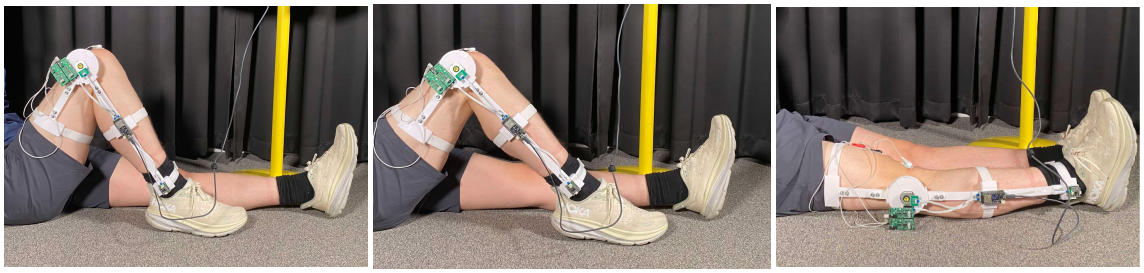
The complete source code, including documentation and version history, is available in the Gitlab repository of the project [17]. To demonstrate the practical application and user interface of the tool, a short demonstration video has been created and can be found online at [18].

4.1 Full Integration Using Angle Data

The first full integration of the final prototype was conducted in two segments, where the first one is based on angle data collected from the encoder that is put on the prototype's joint. The other segment explores the possibility of controlling the game using IMU angle data, which are based on its orientation to the ground instead.

4.1.1 Game Display Using Encoder Angle Data

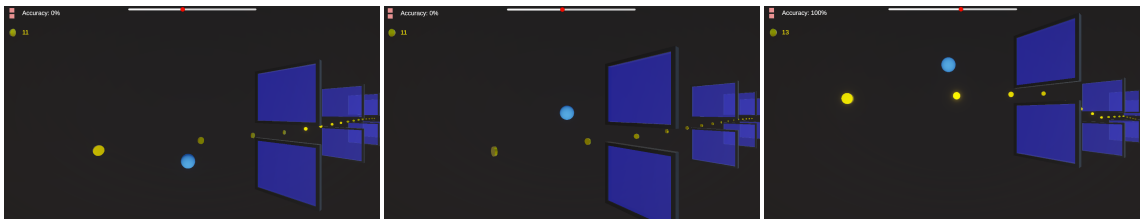
When using the encoder to get angle data, the data is coming from the angle of the knee flexion, which is then input into the "Exo-slide" game. The character moves according to the angle of knee flexion, as seen in Figure 4.2.



(a) Fully retracted knee slide. **(b)** Semi-extended knee slide. **(c)** Fully extended knee slide.

Figure 4.2: This figure shows the range of motion for exercise 3, see Appendix A. Each of these positions has a different knee angle, meaning they will have different positions within the game, see Figure 4.3.

The output of this movement can be shown in Figure 4.3. The update frequency is relatively high at around 170 Hz, and the character does not go outside bounds because of the already established range of movement that would be used. This result is therefore considered successful.



(a) Bottom position. **(b)** Middle position. **(c)** Top position.

Figure 4.3: Gameplay showing the blue obstacles indicating the medium difficulty in the Exo-Slide game. The game is being played with the input from the prototype while imitating the knee slide rehabilitation exercise, and the player's position is going between the highest and lowest range, reflecting the range of motion for the exercise. Each player position corresponds to the knee angles in Figure 4.2 (a) to Figure 4.2 (c).

4.1.2 Takeaways for the Angle Game Test

The game has a sampling rate of 170 Hz, and the player movement matches the angle of the knee, proving that the integration into Unity from the sensors is working well and that the encoder angle is a valid input for playing the game.

During the game test, if the angle exceeded or went below the lowest specified angle in the game, the character would end up out of bounds. However, this did not affect the game experience for the player as it was still possible to avoid obstacles, and the player's accuracy could still be gauged. The reason for the exceeded angles is that the prototype was not designed with an angle lock on it, as it is a placeholder and is only meant to show how it works.

4.1.3 IMU Angle Data

The use of the IMU sensor angle data instead of the encoder is something that would allow different types of movement, for example, without flexing the knee, but rather when raising the leg, shown in Figure 4.4. This shows that different sensors that produce similar types of data can be used to play the game, allowing the user to perform different exercises and still play the same game. However, this exercise was not adapted to the current game but could be adapted with IMU-sensor feedback.

Figure 4.4: Testing with the prototype strapped to the leg, while performing Exercise 8 in the rehab protocol, see Appendix A.

The IMU was used to verify the quality of the movements. During measurement of the encoder angle, the IMU could measure the lateral angle of the calf to verify if the knee is tilting during movement. Figure 4.5 shows the measurements of an exercise in which the IMU detects a tilt between 5 and 15 degrees. This is relevant for therapists who want to quantify the quality of movement.

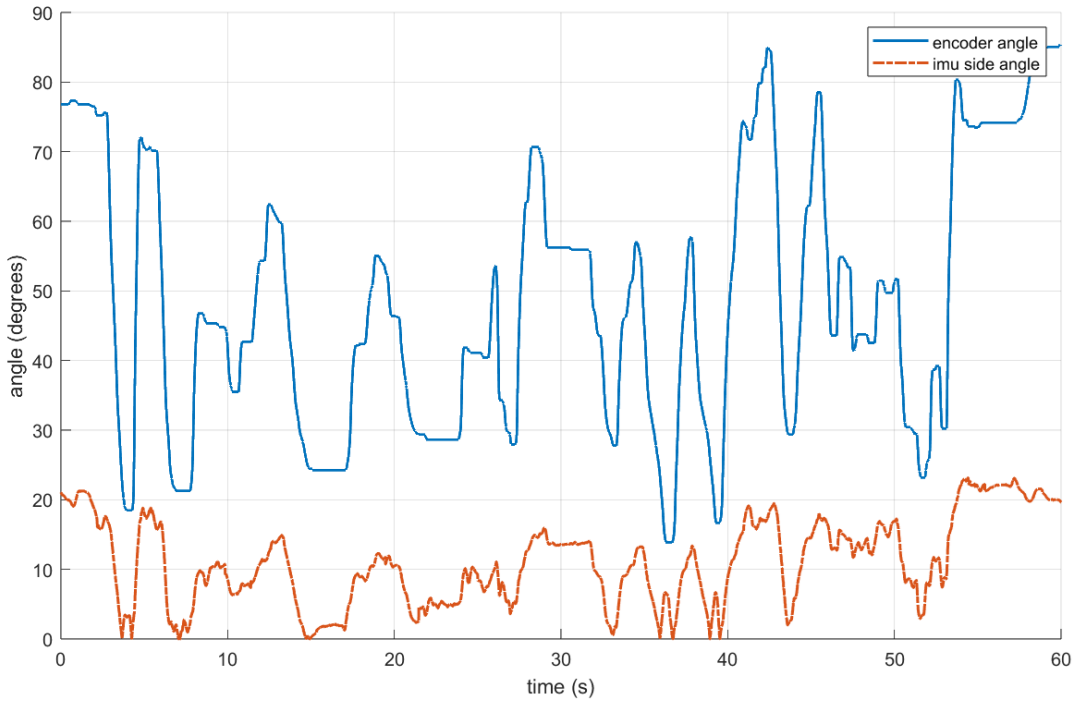


Figure 4.5: Data during "Exo-Slide" game. Encoder angle in blue, indicating the current angle of the encoder. IMU side angle in red, indicating how straight the sensor is compared to the floor.

4.2 Full Integration Using Muscle Tensions

This integration is meant to show that many types of sensors or data can be used to create fun and motivating games. At the same time, it also shows that already established games on the market could potentially be used in rehabilitation without having to make a completely new game for a certain exercise or input data.

As mentioned in Section 3.1.3, when the EMG signal passes a certain threshold, the code sends an integer value of 1 or 0, which will be used in "Rehab Flappy" to determine if the player object flaps or not. This can be seen in Figure 4.6, where the user is creating tension in their leg to send a value that can later be incorporated into the game.

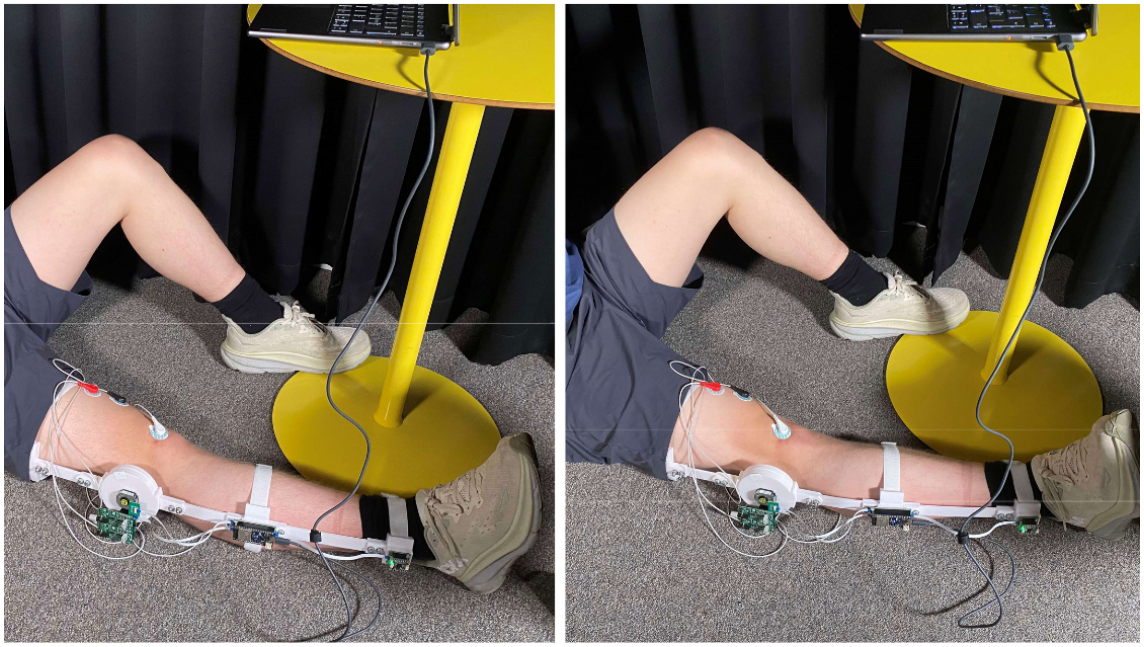


Figure 4.6: Isometric quad contraction rehabilitation exercise. To do the exercise, the leg muscles contract, which activates the EMG sensor positioned on the thigh in the figure.

4.2.1 Game Display Using EMG Activation Data Game Test

Figures 4.7 and 4.8 illustrate the integration of EMG control within "Rehab Flappy" during a game test. Figure 4.7 shows an in-game screenshot captured during active play, where the bird's movement is triggered by muscle activation.

Figure 4.7: Rehab Flappy controlled by isometric quad contractions using the EMG sensor (in medium difficulty).

Figure 4.8 complements this by presenting the corresponding EMG signal data from the test run. In the bottom graph, the blue dotted line represents the recorded EMG amplitude over time, while the orange line indicates the threshold. Whenever the EMG amplitude is above the threshold, it gives a Boolean output in the top graph. Together, these graphs demonstrate how muscle activations in a patient can be used to trigger outputs within a game.

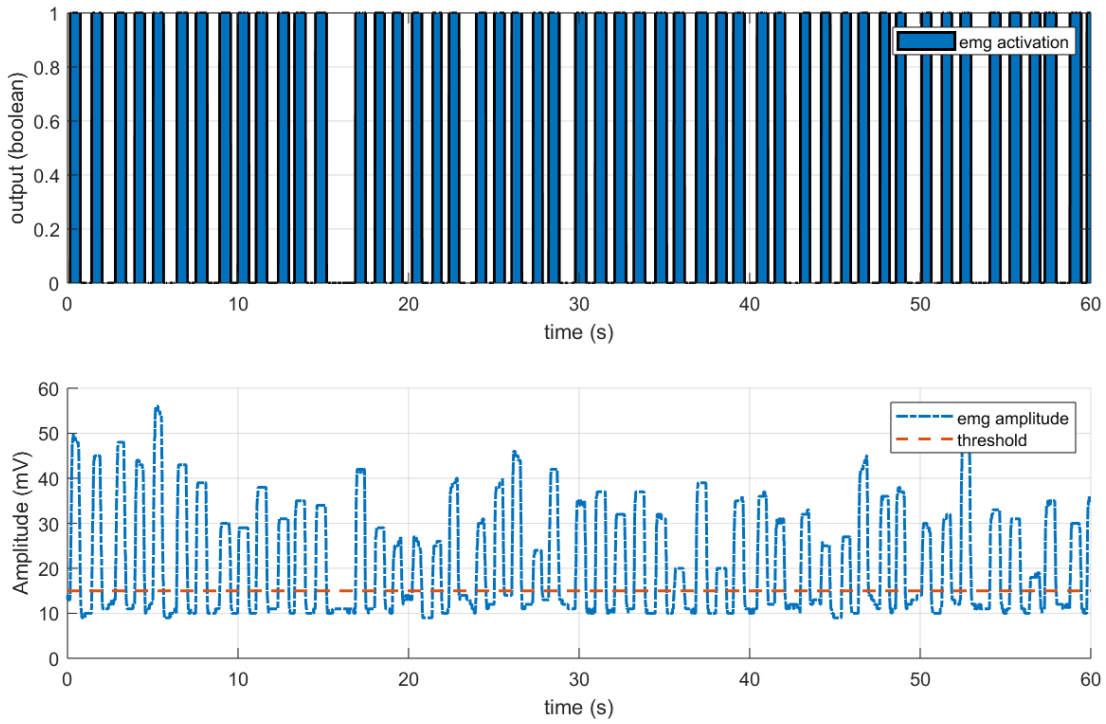


Figure 4.8: Data from Rehab Flappy test run. The bottom graph shows the EMG amplitude and the threshold value. The resulting output is seen in the top graph. Whenever the amplitude is above the threshold, it results in a boolean output in the top graph.

4.2.2 Takeaways for the EMG Game Test

During testing of the EMG sensors in the game, it became evident that if the game remained at the same difficulty as playing with a keyboard, it would be extremely difficult for the user to consistently flex their muscles at an adequate speed to be able to avoid the obstacles. However, the measurements and thresholds worked as intended, and since clinical relevance was out of the scope of this project, this was deemed sufficient.

4.3 Prototype Wearability

While wearing the prototype, an uninjured user was able to perform all rehabilitation tasks without significant pain. Details of the exoskeleton are explained in Section 3.1.5. The mounts, depending on the user, sometimes felt uncomfortable

when attached. The joint followed the movements of the user without significantly inhibiting or assisting the user during the movements. During testing, the connections between mounts were not altered in size, making them either too large or too small for some individuals. The final prototype, including sensors and excluding the USB cable and EMG cables, weighed 366.2 g. Since the development of a physical prototype lies largely outside the scope of the project, the wearability was deemed sufficient. See Figure 4.9

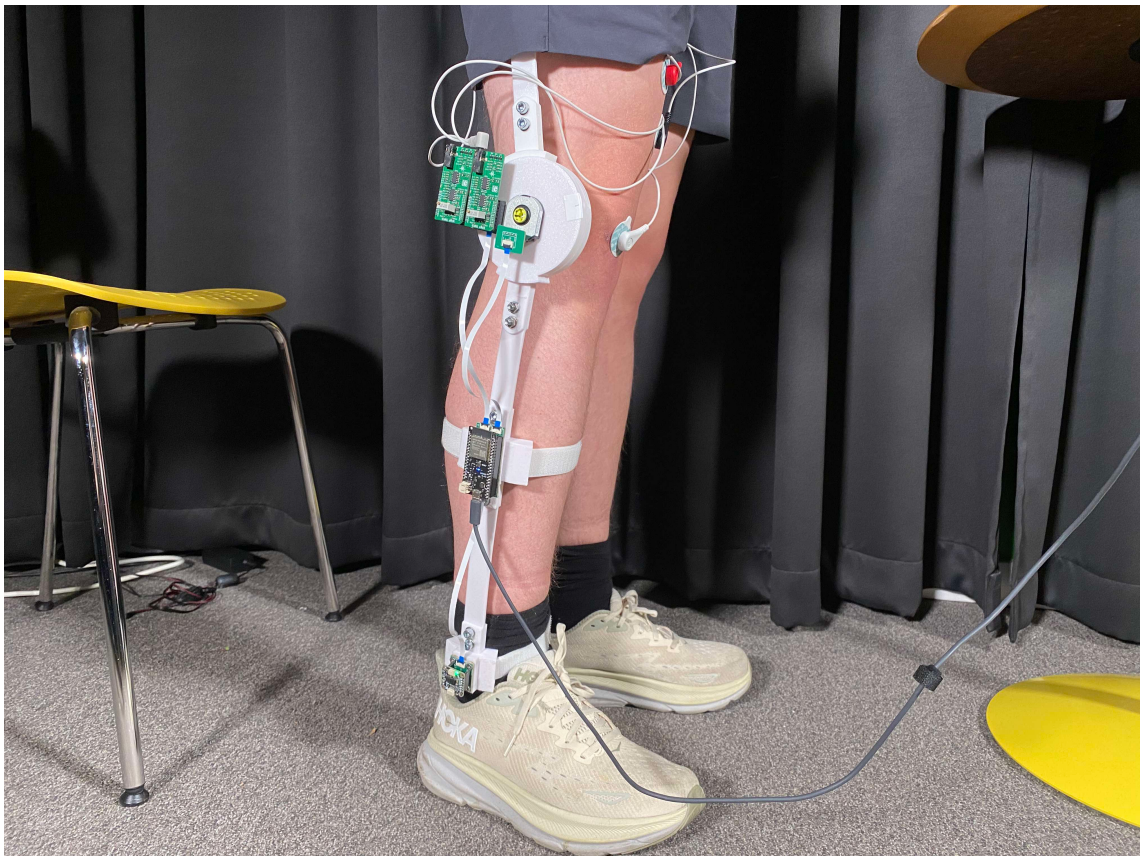


Figure 4.9: Demonstrating the Wearability of the mechanical prototype. It has three fasteners to keep it in place on the leg, one around the thigh, one around the calf, and one around the ankle. The figure shows all the sensors from the IMU at the bottom, the microcontroller in the middle, the EMG, and the angle encoder at the rotating knee joint.

5
Conclusion

In conclusion, this project shows that it is possible to send a usable signal from the body to an external device. An exoskeleton was constructed with sensors and microcontrollers to measure quantitative information from bodily movements during actual ACL rehab exercises. The information was processed to be able to be plotted as well as used as input for a game. Two games, "Exo-slide" and "Rehab Flappy", were created, which showcased this data. The games were designed in such a way that when you play the game, you have to perform one of the exercises recommended by Sahlgrenska Hospital [6], see Appendix A. In the games, there is feedback being shown to motivate the player in the form of coins and accuracy, or current score. This prototype is the first step in gamifying physiotherapy for ACL injuries. By integrating more exercises and fine-tuning the games, a further developed prototype version could have the potential to be of clinical relevance. Although the larger project has a long way to go before it is clinically relevant, this first prototype was deemed a success.

6

Limitations and Future Development

This project has been a long and iterative process, with many trials and errors that shaped the final result. This chapter presents the different key decisions made during the course of the project, highlighting what worked well and the challenges faced along the way, as well as additional potential areas for future work.

6.1 Changes Regarding Prototype and Sensors

During the project, the selection of what type of sensors to apply changed. The original selection of sensors included a torque sensor for measuring forces acting on the joint, which allows measurements of the patients' strength during rehabilitation. However, when other options were found that could be delivered on time and fit the budget, additional problems arose. Options were discussed to alter the physical design of the prototype to allow cheaper options available with Swedish retailers. This would add another layer of complexity to an already multifaceted project, so the decision was made together with the supervisor to not include a torque sensor. However, the possible clinical benefit of directly measuring the patients' relative strength during rehabilitation remains. Since the prototype is, at its core, a sensorized gamification of the rehabilitation exercises, the torque sensor was deemed unnecessary to achieve the goals of the project. Due to the structure of the ROS architecture, implementation of additional sensors should be rather simple, however the limitations lie in the physical prototype and the PCB design. With additional time, a torque sensor would prove the most useful improvement regarding the prototype and sensors.

Throughout most of the project, a Seeit ESP32-DEV-38P microcontroller was used. Unfortunately, it had problems when communicating via micro-ROS, so it was scrapped in favor of the WeMos LOLIN32.

The physical design of the prototype i.e, the 3D-printed parts, was limited mainly due to time constraints; the main focus was on providing mounting for the sensors. The choice of materials was limited due to the budget and time-constraints, some components, such as the rotational axis, would preferably be made of metal or a more durable material rather than plastic. With extended use, the PLA-plastic that the rotational axis was made of could break. Since the prototype was not tested

for prolonged use or between individuals to a great extent, possible issues remain unidentified. The issue with the uncomfortable mount could be resolved by altering the dimensions to better suit the current user.

Additionally, the rotational parts of the prototype are in their design, quite stiff. When the knee moved, plastic layers rubbed against each other, creating friction in the movement. This small additional resistance does not significantly affect the rehabilitation movements. Since the rehabilitation movements were tested on healthy users, the effect of the resistance might be overlooked. Because the physical design is not made for injured users at this stage, this design flaw can be safely overlooked at this stage.

6.2 Possible ROS Optimisation

The prototype leveraged ROS2 as the middleware for managing sensor data flow between the ESP32 microcontroller and the Unity-based game application. While this setup successfully demonstrated real-time data translation, several optimizations could enhance scalability, flexibility, and precision in future iterations.

In the current implementation, the ESP32 communicated with the host computer via a USB cable, which limited mobility and introduced practical constraints during demonstrations. A logical improvement would be to utilize the ESP32's Wi-Fi capabilities to transmit sensor data wirelessly to an ROS-TCP-Endpoint hosted on a local server. This would enable a distributed architecture where one computer could handle data aggregation while another renders the game, reducing computational load and enabling remote monitoring. ROS2's native support for DDS (Data Distribution Service) over Wi-Fi could further streamline this process, as it is designed for decentralized systems.

The prototype's sensor data flow frequency was constrained by hardware limitations and USB serial bandwidth. While standard displays refresh at 30–60 Hz, increasing the sensor data frequency (e.g., to 200–500 Hz) would not visibly make the game look smoother but could enhance motion-tracking accuracy for clinical analysis. For rehabilitation, higher-frequency data would better capture micro-movements or deviations in joint kinematics, aiding in progress tracking. Future work could explore this trade-off, prioritizing data fidelity where medically relevant while optimizing for game performance.

The current ROS nodes were centralized on a single machine, which simplified debugging but limited scalability. Deploying ROS nodes on edge devices (e.g., Raspberry Pi) or cloud servers could allow for real-time analytics, long-term data storage, or multi-user applications. For instance, a clinician's interface could subscribe to processed data topics while the game subscribes to low-latency streams. ROS2's flexible middleware supports such configurations, though this would require additional benchmarking to balance latency and reliability.

In summary, while the prototype validated the core concept, transitioning to wireless communication, optimizing data frequency for clinical use, and decoupling ROS nodes would address key limitations in preparation for real-world deployment.

6.3 Changes During Unity Development

The choices made in Unity were mostly towards game design, as well as what information should be displayed and how. The first version of the "Exo-Slide" also used coins and obstacles, but the player moved left or right instead of up or down. However, this is not very natural since the motion of the player does not match the movement in the game. Therefore, it was decided that the game would be redesigned with movement in the same axis as the player is moving to make it more comfortable to play the game with the intended exercises.

6.3.1 Previous Game Design

The following Figure 6.1 shows how the first instance of the game looked using horizontal movement instead of vertical. In this instance, coins could be collected, and a single obstacle could be avoided. In this version, obstacles hit would be counted and displayed in the top left corner. The game would not show any information about exactly how accurate the player was during "Exo-Slide" or how close to the obstacle it was when it passed or hit an obstacle.

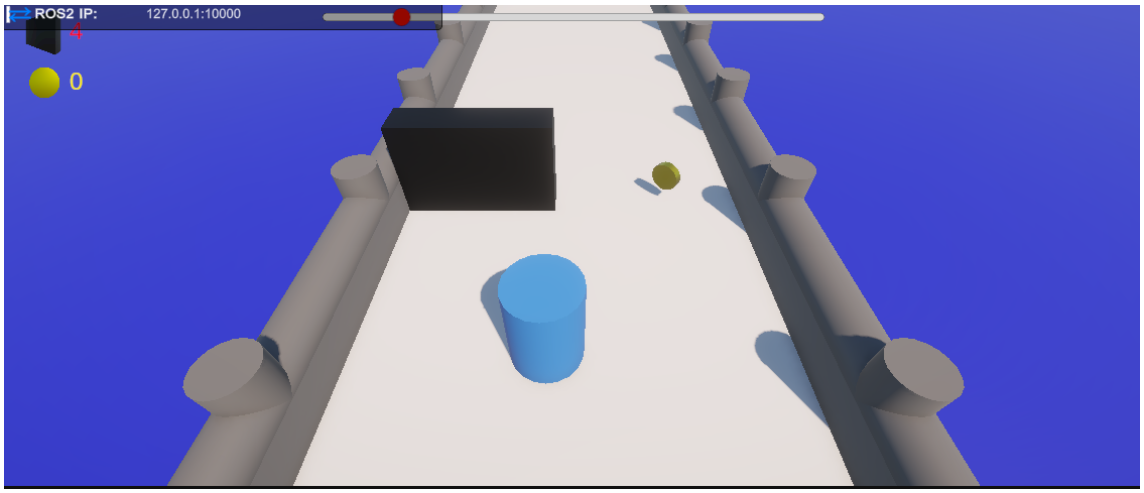


Figure 6.1: This is the first iteration of the game "Exo-Slide", where the player (the blue cylinder in the middle) moves from side to side, rather than up and down in the final version. When the player hits an obstacle and or coin, the counters in the top left corner will increase by one.

Worth noting is that this game concept could be used for a different exercise design that utilizes horizontal movement rather than vertical, with the use of, for example, the IMU sensor that measures the angle on a different axis.

6.3.2 Improving Exo-Slide

The decision to display the accuracy of the player avoiding or hitting obstacles was a recommendation from the examiner, as it would show how much control the player has over their leg during the exercises. If this information could be saved for later, it could be used to display the improvement of the player over a longer period.

The "Exo-Slide" game is developed from the base of the previous game, but the game is instead rotated 90 degrees, meaning that the input currently is still based on the x-axis, but the game movement looks like y-axis movement. This was made as it was unclear how long it would take to redevelop the game. However, for future developments, it would be recommended to change the axis to the y-axis to avoid confusion during game development. This change should be rather straightforward in the game's current version.

6.3.2.1 Alterations to Game Difficulty

The difficulty settings in the game were first put in the options menu, where you currently choose the speed of the game; it would instead be set to easy, medium, or hard difficulty. This was later changed to incorporate all difficulties in the "Exo-Slide" at once to get three different perspectives on the players' performance from one game session.

6.3.2.2 Limitations for the Game Input

The game is only using a single input from the ROS2 topic currently. This is because, for the chosen exercise, the angle data seemed the most important and easy to implement into the game. Although many more inputs could be utilized, it was decided to narrow the scope of the project to be sure that everything could be completed.

6.3.3 Rehab Flappy changes

Altering the game parameters to suit a rehabilitation movement proved difficult. The games' speed, gravitational coefficient, and EMG threshold could be altered to either make the game easier or more difficult. To alleviate this issue, game difficulties were added: easy, medium, and hard. However, alterations of the games' difficulty still occurred, tailored to the current player. Giving players even greater control over the games' difficulty level by allowing them to manually alter game speed, gravitational coefficient, and EMG-threshold value, could possibly solve this issue entirely.

6.3.3.1 Issues with Calibration

Calibration of Rehab Flappy proved difficult, as threshold values between individuals varied greatly and were manually adjusted between sessions. An automated calibration sequence would possibly solve this issue by calculating an optimal threshold

by measuring the maximum and minimum EMG outputs. This calibration was however, not implemented, but acts as a future improvement area.

6.4 Future Development

Future developments should aim to significantly strengthen both the technical foundation and clinical relevance of the platform. For rehabilitation exercise 1, we can extend EMG control to a dual setup that detects both ankle extension and flexion, enabling additional game actions and ensuring the user performs the exercise correctly. For exercises 5 and 6, the IMU angle data from the IMU can be used to control a game similar to or identical to "Exo-Slide". For exercise 7, height could be measured with position coordinates from an IMU. Adding a torque sensor to the mechanical exoskeleton would provide direct measures of joint strength, enhance adaptive game-difficulty algorithms, and offer clinicians objective progress metrics beyond kinematics and EMG thresholds. Finally, linking the ESP32 with ROS2 to a wireless connection such as Wi-Fi would remove cable constraints and improve usability.

Bibliography

[1] Della Villa F, Andriolo L, Ricci M, *et al.*, "Compliance in post-operative rehabilitation is a key factor for return to sport after revision anterior cruciate ligament reconstruction - PubMed," *Knee surgery, sports traumatology, arthrosopy; offical journal of the ESSKA*, Aug. 2019.

[2] J. S. Sagen, I. Kjeken, A. Habberstad, *et al.*, "Patient Involvement in the Rehabilitation Process Is Associated with Improvement in Function and Goal Attainment: Results from an Explorative Longitudinal Study," *Journal of Clinical Medicine*, vol. 13, no. 2, 2024, ISSN: 20770383. DOI: 10.3390/jcm13020320.

[3] Hansson F, Moström EB, Forssblad M, Stålman A, and Janarv PM., "Svenska korsbandregistret Årsrapport 2022.," Swedish, *The Swedish national knee ligament registry*, Aug. 2022. [Online]. Available: https://www.aclregister.nu/media/uploads/Annual%20reports/rapport2022.pdf.

[4] M. Randriambelonoro, C. Perrin, A. Blocquet, *et al.*, "Hospital-to-Home Transition for Older Patients: Using Serious Games to Improve the Motivation for Rehabilitation – a Qualitative Study," *Journal of Population Ageing*, vol. 13, no. 2, pp. 187–205, Jun. 2020, ISSN: 1874-7884. DOI: 10.1007/s12062-020-09274-7.

[5] P. A. Rego, P. M. Moreira, and L. P. Reis, "Architecture for Serious Games in Health Rehabilitation," in 2014, pp. 307–317. DOI: 10.1007/978-3-319-05948-8{_}30.

[6] *Knä – Korsband - Främre*. [Online]. Available: https://mellanarkiv-offentlig.vgregion.se/alfresco/s/archive/stream/public/v1/source/available/sofia/su3187-637809942-600/surrogate.

[7] S. Al-Rayes, F. A. Al Yaqoub, A. Alfayez, *et al.*, "Gaming elements, applications, and challenges of gamification in healthcare," *Informatics in Medicine Unlocked*, vol. 31, p. 100 974, 2022, ISSN: 23529148. DOI: 10.1016/j.imu.2022.100974.

[8] M. Mathew, M. J. Thomas, M. Navaneeth, S. Sulaiman, A. Amudhan, and A. Sudheer, "A systematic review of technological advancements in signal sensing, actuation, control and training methods in robotic exoskeletons for rehabilitation," *Industrial Robot: the international journal of robotics research and application*, vol. 50, no. 3, pp. 432–455, Apr. 2023, ISSN: 0143-991X. DOI: 10.1108/IR-09-2022-0239.

[9] R. G. Baraniuk, T. Goldstein, A. C. Sankaranarayanan, C. Studer, A. Veeraraghavan, and M. B. Wakin, "Compressive Video Sensing: Algorithms, architectures, and applications," *IEEE Signal Processing Magazine*, vol. 34, no. 1, pp. 52–66, Jan. 2017, ISSN: 1053-5888. DOI: 10.1109/MSP.2016.2602099.

[10] J. Fong, V. Crocher, D. Oetomo, Y. Tan, and I. Mareels, "Effects of robotic exoskeleton dynamics on joint recruitment in a neurorehabilitation context," in *2015 IEEE International Conference on Rehabilitation Robotics (ICORR)*, IEEE, Aug. 2015, pp. 834–839, ISBN: 978-1-4799-1808-9. DOI: 10.1109/ICORR.2015.7281306.

[11] *Overview of Robot Operating System (ROS)*. [Online]. Available: https://se.mathworks.com/help/ros/gs/robot-operating-system-ros-basic-concepts_mw_85271ff3-e5c4-4998-9fbd-7eee81be97d9.html.

[12] *ROS 2 (Robot Operating System): overview and key points for robotics software*. [Online]. Available: https://robotnik.eu/ros-2-robot-operating-system-overview-and-key-points-for-robotics-software/#:~:text=For%20over%20a%20decade%2C%20ROS,and%20developers%20around%20the%20world.

[13] *Understanding ROS Nodes*. [Online]. Available: https://wiki.ros.org/ROS/Tutorials/UnderstandingNodes.

[14] *Agile Project Management: Explained for Beginners*. [Online]. Available: https://businessmap.io/agile/project-management.

[15] *Autodesk Fusion | Molnbaserad programvara för 3D-CAD, CAM, CAE och PCB | Autodesk*. [Online]. Available: https://www.autodesk.com/se/products/fusion-360/overview?term=1-YEAR&tab=subscription.

[16] *How to make Flappy Bird in Unity (Complete Tutorial) - YouTube*. [Online]. Available: https://www.youtube.com/watch?v=ihvBiJ1oC9U.

[17] Ludwig Andersson, Jeremiah Terdin, Kaspar Lifmark, Jeremiah Terdin, Pontus Åkermark, and Teodor Franzell, *Files · main · phric / AURORAPhi / AURORAExoSkeleton / Supervision / Bachelors / 2025 / EENX16-VT25-04_SensorGameExo / EENX16-VT25-04_SensorGameExo · GitLab*, Jan. 2025. [Online]. Available: https://git.chalmers.se/phric/aurora_phi/aurora_exoskeleton/supervision/bachelors/2025/eenx16-vt25-04_sensor_game_exo/eenx16-vt25-04_sensor_game_exo.

[18] *Full demonstration of Exo-Slide and Rehab Flappy (Sensorized Exoskeleton For Therapy Gamification)*. [Online]. Available: https://www.youtube.com/watch?v=SpC7a2safC8.

A

Appendix A

SAHLGRENSKA UNIVERSITETSSJUKHUSET VGR

Knä – Korsband - Främre
Hemträningsprogram

Programmet ska utföras 3 – 5 gånger per dag

 	1) Kraftiga fottramp upp och ner. *Cirka 3 minuter flera gånger per dag eller vid svullnad.*
	2) a) Spänn lårmuskeln ”pumpa” 20 – 25 gånger *Upprepa 3 gånger* b) Spänn lårmuskeln maximalt och håll kvar i 10 sekunder *Upprepa 5 gånger*
	3) Böj och sträck i knäleden. Låt foten glida mot underlaget. *2 x 20 gånger*
 	4) Med <u>hela</u> fotsulan i golvet. Böj och sträck i knät så att foten glider så långt fram du kan – så långt bak du kan *10 gånger*
 	5) a) Sitt på en bordskant. Vinkla upp foten och sträck maximalt i knäleden. Lyft därefter benet ca 10 cm från golvet och långsamt tillbaka igen. Vila *3 x 10 gånger* Förmåga att bibehålla en god knäkontroll i övningen styr när du kan stegra träningen. Detta gör du genom att: b) Sätta dig på en stol c) Sätta dig på golvet

Sahlgrenska Universitetssjukhuset
Arbetsterapi och Fysioterapi
sahlgrenska.se

SAHLGRENSKA UNIVERSITETSSJUKHUSET VGR

2025-03-07 Sida | 2

	6) Stående tyngdöverföring. Spänn lårmuskeln på det opererade benet och lägg över tyngden på rakt ben. *20 gånger*
	7) Stående. Tåhävning, på två ben. *2 x 20 gånger*

De första dagarna ska du ta det lugnt och ha benet i högläge så mycket som möjligt.
Använd kylförbandet som smärtlindring i perioder om 10 – 20 minuter åt gången.
Pumpa med fötterna ofta och jobba på med lårmuskelaktiveringen för att minska svullnad i benet/underbenet och risken för trombos (blodpropp).
Du ska använda kompressionsstrumpa dygnet runt i 1 vecka och därefter 1 vecka dagtid. Strumpan bör bytas och tvättas regelbundet (för att behålla sin elasticitet).

De första dagarna ska du använda kryckor när du går. Du får belasta till smärtgränsen på det opererade benet. Försök att gå så normalt som möjligt. Utomhus bör du gå med kryckkäppar fram till återbesöket till fysioterapeut cirka 2 veckor efter operationen.
Använd en påse el dylikt som skydd för såret när du duschar.

Vid kraftig vadsmärta eller om knät svullnar upp onormalt mycket och blir värmeökat bör du kontakta akutmottagningen för kontroll.

Boka tid hos Fysioterapeut om 2 veckor för start av fortsatt träning.

.....

.....

sahlgrenska.se 1177.se

Programmet är kopplat till din behandling som du fått i samband med ditt besök på sjukhuset

www.sahlgrenska.se

Sahlgrenska Universitetssjukhuset
Arbetsterapi och Fysioterapi
sahlgrenska.se

Figure A.1: Full list of ACL rehab exercises (1-7), recommended by Sahlgrenska Hospital [6]. The use of these are explained in section 2.2

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

CHALMERS
UNIVERSITY OF TECHNOLOGY