Next-Step: Tech Feasibility

Jett Koele
Benjamin Huntoon
Naima Ontiveros
Kendall Callison

Sponsored by:
Jack R Williams & Zachary Lerner

Mentored by:
Scott Larocca

2. Table of Contents

3. Introduction

In the United States, over 7.5 million individuals face difficulty walking and engaging in physical activity due to conditions that impair mobility. For these individuals, traditional treatment approaches such as physical therapy (PT) and passive leg braces are commonly prescribed. However, despite PT's potential benefits, maximizing the amount and effectiveness of therapy can be challenging.

The Biomechatronics Lab at Northern Arizona University(NAU), led by Dr. Zachary Lerner and Dr. Jack Williams, is working to address these challenges by developing a fully open-source exoskeleton system known as OpenExo. The current system is designed to help overcome barriers to the widespread use of wearable exoskeletons by making them more accessible. OpenExo, combined with an open-source Python API, enables real-time operation and assessment of the exoskeleton. Still, a key challenge remains: how to keep users engaged during rehabilitation to ensure they get the maximum benefit from using the device.

Our capstone project focuses on addressing this engagement challenge. We aim to develop a gamified rehabilitation training tool that interfaces with the OpenExo system through its already-built Python API. This component we will add to the Python API will communicate with Bluetooth sensors and built-in sensors to collect real-time data and provide interactive, game-like feedback to patients as they use the exoskeleton for different exercises. By transforming rehabilitation into an engaging experience, we hope to improve patient motivation and enhance the overall effectiveness of therapy. To make the tool more accessible and encourage collaboration, we are making it open-source. This will let developers, therapists, and researchers customize and improve it to fit different rehabilitation needs. With an open-source approach, we hope to build a community that keeps the tool innovative, flexible, and available to everyone.

We are building on an existing program that was developed by the Biomechatronics Lab at NAU to enhance its functionality for rehabilitation robotics applications. This includes expanding the core modules, such as chart_data.py, exoData.py, exoDeviceManager.py, exoTrial.py, openPythonApi.py, and realTimeProcessor.py. Additionally, we are integrating new functionalities within the provided GUI.py and BioFeedback.py files to create a more user-friendly interface and improve real-time biofeedback capabilities. Our goal is to create an interactive tool that can support both research and practical applications in rehabilitation therapy.

In this Technological Feasibility Analysis, we will explore the main technological challenges of the project, including integrating Bluetooth sensors, processing real-time data, and building a gamified feedback system. We will analyze possible solutions and demonstrate the feasibility of our approach, ensuring that our final product is effective, engaging, and useful for any researcher working with wearable robotics.

4. Technological Challenges

Creating an open-source project is a challenge within itself. Thinking about all the possible ways to develop and design software that is open-source and then narrowing it down to fit the project can be a struggle. Some problems we have encountered are usability for users and developers, any game getting inputs from the exoskeleton, expandability of the software, and Bluetooth sensor support. These problems are all directly related to our open-source idea as we constantly have to keep these in mind while developing our project.

With creating an open-source game controlled by rehabilitation prosthetics, there are several technical challenges that must be addressed. First, we are not only developing games for users but developing a platform for outside developers to also contribute games. With the project being open source, we want to easily invite other developers to help give users new, unique experiences. To do this, we will create scripts that standardize and present only the necessary biometric data for controlling a game. On the user end, easy-to-navigate menus will allow you to simply choose and play any downloaded games made for the program. The heart of this idea being that with a standardized method of collecting user input, future developers can make a wider variety of games for users to play while they engage with their exercises. Ideally, this would add a “plug and play” aspect to the games users can play while wearing the brace as opposed to a single game, keeping the rehabilitation process engaging and exciting.

When creating the actual games, our team must look at a few different challenges. To start we must compare and contrast different Python libraries, such as Pygame and Pyglet, in order to find the most suitable fit for us. The libraries must provide adequate support for rendering graphics, handling game mechanics, and ensuring smooth gameplay for the user.

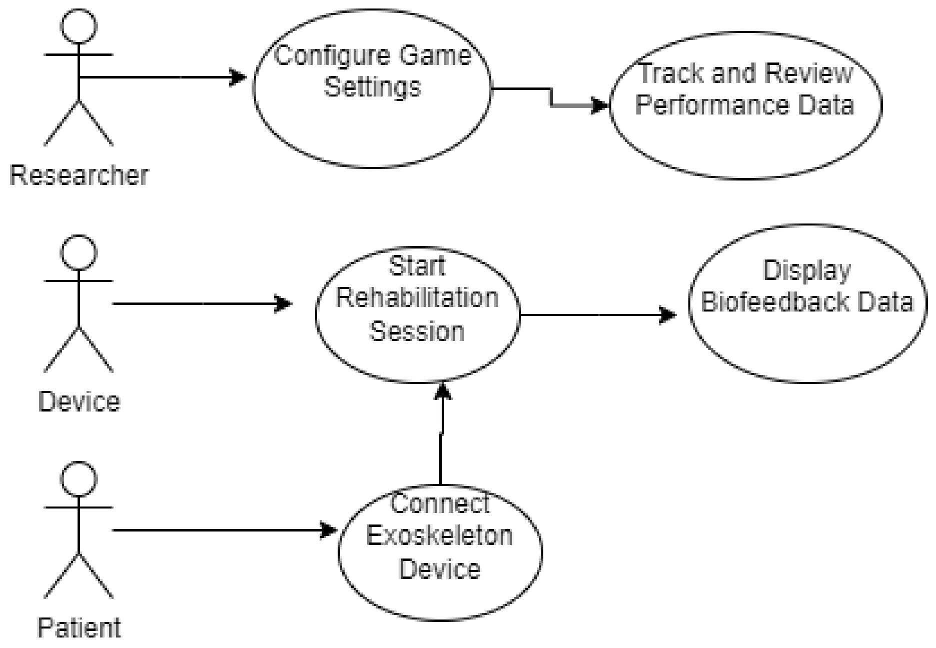
Like any other video game, we must consider all possible game states and logic. Using a standardized system for "Start", "Pause", and "End" will help us and other developers create a more desirable user experience. One of the largest contributors to user experience is the game's logic, or how the user interacts with the game's environment. This aspect is key to making the games fun and rewarding for those using it. The back-end involves having a standardized control system that reacts to real-time data from the Bluetooth sensors. This will involve ensuring that games respond dynamically to user inputs and sensor data without causing delays or crashes. On the front-end, the user interface must provide intuitive immediate feedback to players based on sensor input. This means that Python libraries must support real-time data handling, ensuring that any game event tied to rehabilitation movements reflects immediately in the game.

Finally, the game itself must put the user’s rehabilitation first as the primary experience before the game. While our project is to develop an engaging experience for the user, our game must never take away from the prescribed exercises and we must therefore develop around that concept. Moreover, the games should center on the specific exercises users will have to perform. These exercises are created by us developers but will allow the user to set unique goals for themselves, ensuring they can be used on anyone's path to rehabilitation.

5. Technology Analysis

We need to develop a game for a rehabilitation brace so that wearers will be more engaged with their rehabilitative therapy. The game should help make the exercises motivating by including rewards and challenges. The game also needs to be adjustable to different types of exoskeleton devices so that it works for a variety of treatments.

Use Case Diagram for Rehabilitation Game


Configure Game Settings
Track and Review Performance Data
Researcher
Start Rehabilitation Session
Display Biofeedback Data
Device
Connect Exoskeleton Device
Patient

It shows how the patient, therapist, and exoskeleton device interact to run the game, provide feedback, and track progress.

5.1 Desired Characteristics:

Our project will be built upon the existing code to implement a game a user can play while wearing the leg brace for rehabilitation exercises. The largest request from the clients was to ensure that the game remains open source as the entire mission of the biomechatronics lab at NAU is to create an open-source form of mechanized rehabilitation for people in need. Our game should focus specifically on the leg braces, however, the game should be able to be configured with different settings to support various types of devices designed for different parts of the body. The game should run smoothly and without delay while engaging the user for the duration of their exercise. This aspect of the project is critical to ensure that while the user is engaged they are not distracted by poor game performance while they work on their exercises. Finally, the game should be easy to maintain and adjust to each user. We want to ensure that after we have created our project, future developers are not bogged down by trying to decipher what we have implemented and we want to make sure that if the researchers would like to make certain tweaks to the code that modification of the game should be quick and easy.

5.2 Game Development:

In terms of game development, we were looking at using:
Pygame:

It is a Python library for building simple 2D games. It helps developers easily manage graphics, sound, and user input, making game development beginner-friendly. Pygame focuses on keeping things simple while giving developers control over their code.

Pyglet:

Pyglet is another Python library designed for building games and multimedia applications. It supports both 2D and lightweight 3D development. It has features such as OpenGL graphics, sound, and video playback. Pyglet is fully cross-platform and requires no external dependencies, which makes it better for performance-focused projects.

Unity:

Unity is a game engine that will make it easy for game developers to design and develop games for the future of this project. Generally speaking, it is able to do all of the previously mentioned features and more since it is such a big and very well-sourced game engine. The only limitation would be the computer the game would be running on since hardware-demanding games can be developed and designed on this platform. The main issue with this direction is to simulate a controller input using feedback from the exoskeleton as an input.

GUI:

In our current project, Tkinter is being used to manage the Graphical User Interface (GUI), particularly in BioFeedback.py. As we move forward, we plan to build on this existing code and use Tkinter with the game we are developing. Tkinter is a built-in Python library that helps us create windows, buttons, labels, text boxes, and other simple interfaces. It’s essentially a control panel to our program that users can click on or interact with.

5.3 Analysis:

We decided on implementing a “Wario-ware” style game of a few minigames meant to engage the user. The minimum viable project would include one game of good detail while a better game would include multiple types of games to play. In terms of development, we don’t have a lot of experience with game development and agreed we need to familiarize ourselves with the libraries. BLEAK will also be used to collect data from the Bluetooth sensors from the brace.

We used a few different criteria to evaluate and compare the alternatives. These include familiarity and learning curve, Bluetooth data collection compatibility, and performance and multimedia support.

Criteria 1: Familiarity and Learning Curve

- Evaluation: Since we’re new to game development, we focused on libraries that are easy to learn.
    - Pygame: Beginner-friendly with lots of tutorials and community help.
    - Pyglet: Faster but harder to learn, which could slow us down.
    - Unity: Powerful but takes time to master due to its complexity.

Conclusion: We prefer Pygame because it is easier to learn with our limited time.

Criteria 2: Bluetooth Data Collection

- Evaluation: We need to use BLEAK to collect data from Bluetooth sensors on the rehab brace.
    - Pygame and Pyglet work well with Python libraries like BLEAK.
    - Unity would be harder to set up for Bluetooth since it’s a different platform.

Conclusion: Pygame and Pyglet are better options because they easily support BLEAK integration.

Criteria 3: Performance and Multimedia Support

- Evaluation:
    - Pyglet: Faster and supports OpenGL, good for games with high performance and multimedia needs.
    - Pygame: Slightly slower but still works well for simple 2D games.
    - Unity: Great for big 3D games, but overkill for our small minigames.

Conclusion: If we need higher performance, Pyglet is the better choice, but Pygame is still good for our 2D minigames.

5.4 Chosen approach:

For our project, we opted to choose PyGame as our chosen game development library. While Pyglet is faster and offers more GUI elements, a core component of our project is keeping code open source and easy to understand. This means that PyGame's ease of use is a key factor in our decision and it is easy to incorporate within our project. The largest drawback of this decision is that we will have to rework the way the existing GUI is handled within the existing code as PyGame is not compatible with that aspect of the project currently. Picking the chosen game library is the central decision with regard to our project as other Python libraries like tkinter and BLEAK are already present within the code and must therefore be incorporated into the future project as we begin game development.

Overall, Pygame offers a better approach for developing the rehabilitation game due to its simplicity and user-friendly nature. Its straightforward API allows for quick development, making it easier to focus on core game mechanics and integration with the exoskeleton device. Compared to more complex game engines like Unity, Pygame provides a more manageable learning curve, which is advantageous for our team, given our limited experience with game development in Python. Additionally, Pygame's compatibility with Python libraries (such as Tkinter for the GUI and biofeedback charts) ensures that development remains efficient and free from unnecessary overhead.

Comparison of Game Libraries

Pyglet	**Pygame**	**Unity**
Pyglet is much faster as compared to Pygame⭐⭐⭐⭐⭐	Pygame is slower as compared to Pyglet⭐⭐⭐	Highly optimized for complex projects ⭐⭐⭐⭐⭐
Pyglet uses OpenGL⭐⭐⭐⭐⭐	Pygame uses SDL libraries and does not require OpenGL⭐⭐⭐	Uses a custom, advanced rendering engine based on OpenGL/DirectX/Vulkan ⭐⭐⭐⭐⭐
It is rich with GUI	GUI using Pygame is not	Extensive GUI capabilities

elements⭐⭐⭐⭐	compatible⭐⭐⭐	with native tools ⭐⭐⭐⭐⭐
3D projects are supported in Pyglet⭐⭐⭐	Only 2D projects are supported ⭐⭐⭐⭐	Full 3D project support and physics engine ⭐⭐⭐⭐⭐
Pyglet supports music, video, and images in all formats⭐⭐⭐⭐⭐	Pygame supports a few formats ⭐⭐⭐	Supports all major multimedia formats ⭐⭐⭐⭐⭐
No external installation requirements ⭐⭐⭐⭐⭐	Few module installations are required ⭐⭐⭐⭐	Requires Unity Hub and external dependencies ⭐⭐⭐
Not beginner friendly ⭐⭐	Very beginner-friendly and easier to implement ⭐⭐⭐⭐⭐	More complex, suitable for large projects ⭐⭐⭐

As shown above in the table we listed each major pro and con between the three development engines and ranked each aspect. From these sub-rankings, we then gave each engine an overall ranking based on what we thought would make game development the most efficient for our project. To make each ranking we voted on how many stars each subcategory got between the engines and then decided as a group which library we preferred overall. Although Pyglet outranked Pygame overall, we did not need all the extra features Pyglet provided to begin with. All in all, after taking in all the pros and cons, we decided on mostly developing within Unity. The Unity engine is able to produce faster and more intensive experiences, while also providing all graphical displays right out of the box.

Since we are only building a simple 2D game, all the other overhead is unnecessary and it is much more efficient to choose a library that everyone can use right away since it is more beginner-friendly.

Although we have picked out our game engine of choice, other developers might want differently. Thus, using the python library vGamepad, we can allow for interaction across almost all video game engines. Our approach invisions scaling the raw exoskeleton data into gamepad button inputs. Using vGamepad, one is able to create a virtual gamepad that mimics a real gamepad device. We are then able to use the transformed exoskeleton data as real time input for the virtual gamepad. Since almost all game engines contain built in controller/gamepad support, game development for the exoskeleton can take place on almost any platform.

5.5 Proving feasibility:

To prove feasibility we have begun to create diagrams to illustrate where our project will sit within the existing code. We are working on a package diagram as well as a flow control diagram to illustrate the layout and overall data passage when a user is using the brace for rehabilitation.

Our immediate plans for testing and validating our design choices will include designing smaller "walking-based" games meant to encourage the user to move around. Some initial ideas we have discussed were a hiking-based game or a delivery-based game where the user must walk to a location to deliver an item to win the game. The core idea behind these games is they allow the user to engage with the rehabilitation exercises while also allowing researchers the ability to collect valuable data on the users as they exercise. To create a demo game, we intend on creating simulated data meant to mimic the biofeedback a real brace would put out as a user moves around. This would allow us to collect input for our game so we can simulate a real user experience.

6. Technology Integration

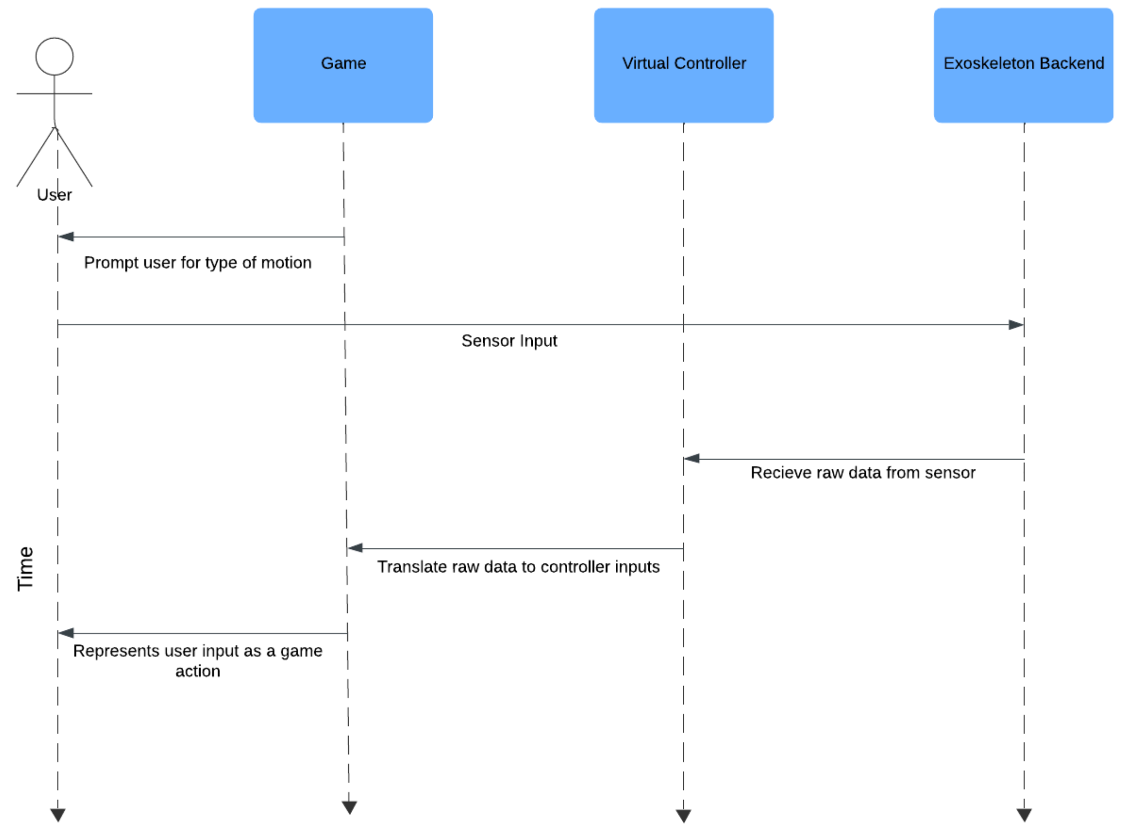
Exoskeleton Brace	Collects and transmits movement data for the user, providing real-time feedback to guide rehabilitation exercises.
Bleak Python Library	Handles Bluetooth low-energy connections to ensure seamless communication between the brace and the game software. This will allow us to track the user's movements in real time and translate them into gameplay.
Pygame Library	A versatile game development library for Python that will enable us to create engaging visuals, sound effects, and smooth gameplay. It also provides tools to build interactive elements, such as timers and score counters, which will enhance the user's experience.
Python	Serves as the primary programming language, helping us integrate various components and libraries. Python's simplicity and versatility

	will make it easier to prototype, test, and iterate on new ideas.

vGamepad	A python library that serves as communication between the exoskeleton brace and the games created for it. This allows us to simulate a real gamepad within python. In turn, allowing game development on any engine. Unity, PyGame, etc.

We are building on an existing program developed by the Biomechatronics Lab at NAU, integrating new features that align seamlessly with the established architecture. The foundational files—such as chart_data.py, exoData.py, exoDeviceManager.py, exoTrial.py, openPythonApi.py, and realTimeProcessor.py—provide the core structure for handling data, managing devices, and processing real-time feedback. Our additions in GUI.py and BioFeedback.py build on these components to enhance user interaction and biofeedback capabilities.

By incorporating Bleak for Bluetooth communication, Pygame for game development, Python as the main language, and vGamepad for virtual controller integration, we ensure that our solution smoothly connects the hardware (exoskeleton brace) with the game software. This setup allows for real-time data flow, accurate feedback, and an immersive, therapeutic experience, all while preserving and expanding upon the existing architecture.


Game
Virtual Controller
Exoskeleton Backend
User
Prompt user for type of motion
Sensor Input
Recieve raw data from sensor
Translate raw data to controller inputs
Time
Represents user input as a game action

7. Conclusion

This project will require a good amount of experience with game development. Once we get familiar with the libraries we need we can begin discussing software architecture and game concepts to work towards implementing. Our project is built upon existing code and will not need to replace any of the existing code to the best of our knowledge. The game or games we develop will focus on leg movements, but will ideally be usable with other kinds of braces meant for rehabilitation.

Rehabilitation games are essential to help users stay motivated and stick with their therapy. In this project, we've planned to build on existing code by adding new mechanics to promote leg movement, while keeping the games flexible enough for other types of braces. We've also highlighted the importance of learning the tools, planning the structure, and using feedback to improve the experience.

Next, we'll focus on building prototypes, testing them, and making adjustments to ensure the games are both useful and enjoyable. With a clear plan in place, we're confident this project will deliver meaningful results and help users on their rehabilitation journey.