

Virtual Gravity

Simulating Weight Sensation in Virtual Reality by Rendering

Gravity using Haptic Interfaces

Sindja LIKA

Marta RÓŻYCKA

Promotor: Maria Torres Vega
Co-promotor: Jit Chatterjee

Master's Thesis submitted to obtain the degree
of Master of Science in Electronics and ICT
Engineering Technology

Academic Year 2024 - 2025

Virtual Gravity

Simulating Weight Sensation in Virtual Reality by Rendering Gravity using Haptic Interfaces

Różycka Marta, Lika Sindja

Master in Electronics and ICT , Faculty of Engineering Technology, Campus GROUPT Leuven, Andreas Vesaliusstraat 13, 3000 Leuven, Belgium

Promotor: Maria Torres Vega

Electronics and ICT , Faculty of Engineering Technology, Campus GROUPT Leuven, Andreas Vesaliusstraat 13, 3000 Leuven, Belgium, <maria.torresvega@kuleuven.be>

Co-promotor: Jit Chatterjee

FIIW Campus GroepT, A. Vesaliusstraat 13, 3000 Leuven, Belgium, <jit.chatterjee@kuleuven.be>

ABSTRACT

The growing accessibility and capabilities of Virtual Reality (VR) have made it an increasingly valuable tool for training applications. Enabling multimodality, beyond purely visual feedback, enhances the realism and effectiveness of training. This is especially relevant for astronaut training conducted in environments that differ from those on Earth. Incorporating haptic devices can introduce tactile feedback in VR, but the current research focuses mainly on devices meant for users' hands. In comparison, this work explores the integration of tactile sensations specifically in the forearm region. The aim is to investigate whether exoskeletons primarily used in the medical field can be implemented to simulate weight sensation for astronaut training in VR. Gravity is rendered through torque control based on the angular position of the forearm. The system is designed with a focus on its portability and accessibility. Initial results indicate that the technical implementation is promising for simulating varying gravitational conditions. By introducing a way to simulate weight sensation, this research contributes to the underexplored area of multimodal VR experiences, particularly in tactile feedback on the user's forearm.

1 INTRODUCTION

Over the years, Virtual Reality (VR) has become more affordable due to the technological developments. As a result, VR has gained popularity in a wide range of applications [1; 2]. VR offers the possibility to simulate diverse environments and scenarios in a safe setting. This makes VR a promising technology for training applications [3]. For example, training astronauts for demanding space missions with VR simulations has become an emerging application field with the revived interest in manned space exploration [4]. Astronaut training requires realism because the simulated conditions extensively differ from those on Earth. One way of enhancing the sense of realism of VR simulations is by incorporating multimodal experiences [5]. However, in the current state of the art, VR remains predominantly a visual and auditory experience. Integrating haptic devices into a VR experience can overcome this constraint, but research into devices that effectively simulate tactile sensations is limited and focuses mainly on hands. However, in certain applications, including astronaut training, it is preferred that the user's hands remain unobstructed. Additionally, considering the forearm sensation, alongside the hand sensation, helps achieve a more realistic experience since the entire upper limb is involved in perceiving weight [6].

This Master Thesis proposes a novel approach to incorporating weight perception into VR experiences through torque-based actuation. We designed and developed an exoskeleton device to operate on a forearm and simulate weight sensation in various gravity conditions. The focal points of design are the system's accessibility and portability. Both of these aspects are the limitations in the current state of the art, where similar devices used in the medical field are priced at thousands of euros or significantly limit user's movement reducing the applicability in VR training scenarios [7]. Results of the technical measurements demonstrate that an accessible weight simulator designed with off-the-shelf components is capable of accurately simulating weight. Moreover, preliminary user evaluation shows that the simulation evokes a realistic perception of weight and the differences in gravitational conditions can be identified.

The remainder of this paper is distributed as follows. Chapter 2 details the design requirements. Chapter 3 presents relevant findings from the related literature. Chapter 4 describes the design and implementation processes. Chapter 5 showcases findings from evaluation of the developed prototype. The qualitative implications of these results are further discussed in Chapter 6. Chapter 7 summarises the work, highlighting its main contribution.

2 REQUIREMENTS

The key difference between haptic devices and other robotic systems is their direct attachment to the human body. This creates a two-way physical interaction, where the device responds to and influences the human movement. Designing haptic devices without clearly defined system requirements can lead to negative consequences for the users [8; 9]. Therefore, it is important to define the requirements of the system that are related to its dynamic interaction with the user's body and evaluate the developed prototype against them.

The requirements of the system are:

- **Comfort of Use, Portability and Weight**

Comfort of use refers to how seamlessly the device integrates with the human arm without causing strain. High discomfort decreases the system usability and user satisfaction, potentially leading to abandonment of the device [10]. Portability describes the system's ability to allow free movement without being constrained by fixed infrastructure. Weight refers to the total mass of the device. These three factors are interconnected. A heavier wearable device reduces comfort and increases fatigue, especially during extended use. Portability further influences comfort because an unportable and bulky system restricts natural arm motion, reducing immersion and usability. Weight can be quantitatively assessed with direct mass measurements. Portability and comfort, being more subjective, are best evaluated through user feedback.

- **Power Consumption and Heat Management**

Power consumption defines the energy required by the system during both active and idle states. Heat management addresses how effectively the system handles thermal losses and dissipates the generated heat. Power and heat are closely related because greater power consumption typically leads to higher heat generation. Moreover, minimising power consumption supports the portability requirement by allowing battery operation. Effective heat dissipation is also crucial to maintaining user comfort. Power consumption can be calculated using voltage and current measurements. Heat management can be assessed by monitoring the temperature of key components during operation.

- **Accessibility**

Accessibility defines how easy it is to reproduce, maintain or further develop a system. A cost analysis can help determine how accessible the components and production process are.

- Latency

Latency in VR systems is the delay between user input and system response. It includes both hardware (signal and actuation delays) and software latency (processing delays). Latency interacts indirectly with comfort, safety, and even control accuracy because delays over 25-100 ms are typically perceivable and can cause cybersickness [11; 12; 13]. Timestamp logging and watchdog functions can be implemented to measure software and hardware response times.

- Control Accuracy

Control accuracy refers to how precisely the device can replicate desired physical interactions, in this case, the weight simulation. Inaccurate control not only diminishes realism and comfort but could also affect user safety if unintended forces are applied. Accuracy can be measured by comparing performance of the device with what is theoretically expected.

- Safety of Use

Safety refers to the system operating within the voltage and current limits suitable for human contact and that the device does not exhibit unexpected behaviour [9]. Safety is influenced by nearly all other requirements. Poor control accuracy, excessive heat or discomfort due to device misalignment may pose direct risks to the user. Measuring voltage and current while performing fault testing is a way to evaluate the device safety.

3 RELATED WORK

3.1 Haptics and VR training applications

The main aim of this work is to incorporate multimodality into the VR experience. Research shows that touch plays an important role in interacting with Virtual Environments (VE) because it involves direct physical interaction between a person and virtual objects [14]. Tactile feedback significantly enhances authenticity and immersion of the VR experience [14; 15]. Moreover, according to [15], haptic feedback can be even more important than visual and auditory cues because it provides real-time control and feedback, which training applications depend on. In [16], the authors further emphasise that haptic feedback enables the closing of the action-confirmation loop, reduces cognitive load and increases user performance. Despite these benefits, the commercial VR controllers typically provide only basic haptic cues [17].

One specific area of tactile feedback that has received attention is the simulation of weight sensation [18; 19]. As

an important aspect of object manipulation, weight sensation enhances the sense of realism in VE. The ability to feel the weight of a lifted object is particularly valuable in fields that demand manual precision. Human weight perception involves complex mechanisms that rely heavily on both skin sensation and force feedback from voluntary muscles [18]. The latter plays a dominant role in creating a convincing illusion of weight.

To enable simulation of physical interactions within the VE, active haptic feedback, such as applying force on the user's body, is required [20]. Specifically, simulating the feeling of lifting an object involves mimicking the gravitational force acting on the limb. Gravity generates torque about the elbow joint, and to replicate this force, actuators must apply a similar torque to the user's arm [21]. In [22], the authors underline the importance of torque-based feedback in recreating movement under non-Earth gravity conditions.

With renewed global interest in space exploration, astronaut training in VR has become an emerging research focus [4]. Traditional astronaut training relies on specialised infrastructure, such as neutral buoyancy pools to replicate microgravity and full-scale simulators that mimic the ISS modules [23]. While effective, these methods are costly, time-consuming and inflexible. In contrast, VR-based solutions offer a lower-cost, safer and more adaptable approach to creating extraterrestrial conditions, as well as shorten training time while maintaining realism.

3.2 Exoskeletons, Exosuits and End-Effectors

The forearm remains an underexplored but important area of VR haptics research. Upper limb dynamics play a crucial role in daily life and weight perception [24]. As shown in [6], restricting elbow motion affects overall weight sensation, confirming that accurate feedback requires engaging the full upper limb. Despite this, most force-feedback interfaces focus on the hands [18]. Targeting the forearm addresses the gap in research while also enabling hands-free interactions, which are beneficial in many contexts, including manipulating virtual tools. Focusing on the forearm also allows for a manageable project scope and system complexity, which is particularly important in prototyping stages.

Since there exists limited work focused on force-feedback devices for the forearm in VR, the rehabilitation field is explored for inspiration. In contrast to VR, rehabilitation has seen extensive development of wearable assistive devices. Three main types of such devices, as seen in Figure 3.1, can be distinguished, mainly end-effectors, exoskeletons and exosuits.

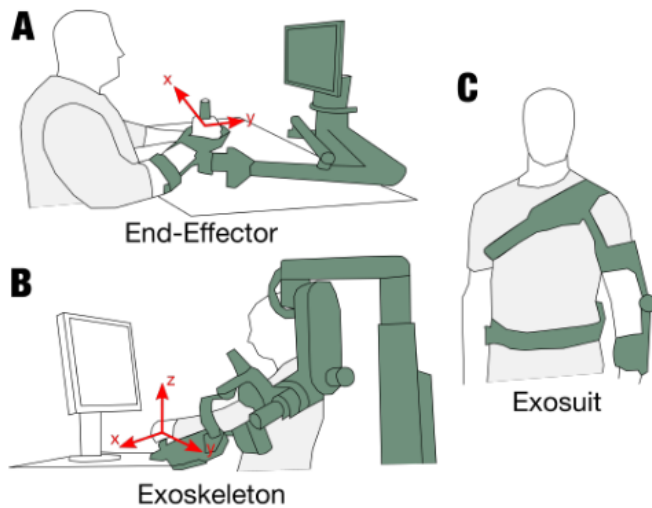


Figure 3.1: Types of wearable robotic structures for upper-limb assistance, adapted from Proietti et al. [25]. (A) End-effector: connects at a single point (e.g., hand) to guide motion. (B) Exoskeleton: rigid frame aligned with joints. (C) Exosuit: soft, textile-based design for portable assistance.

End-effectors are connected to the human body only at one point, which means that the structure's joints do not match with the human joints. Force generated at the interface changes the positions of other joints simultaneously, making the generation of isolated movement of a single joint difficult [26; 27]. In comparison, exoskeletons resemble human limbs because their joint axes match the human joint axes. As a result, in these types of structures, restricting the movement of specific muscle groups is possible [26]. Despite exoskeletons being more complex systems, their ability to control the forearm movement makes them a preferred choice for this application [28].

When comparing exoskeletons and exosuits, the main difference lies in the material type. Exoskeletons are typically constructed of rigid and metallic components, while exosuits are designed using soft and flexible materials. The soft structure, despite its flexibility, low weight and low power consumption, suffers from complexity in adaptation to the human body [29]. Even the smallest misalignment can induce discomfort when forces are applied incorrectly [25]. Other issues that exosuits might face due to their soft structure are failing to deliver the required torque and limited placement options for actuators and power sources. A strategic arrangement can lead to improvements in weight distribution and, in some cases, a reduction in power consumption [30]. Exoskeletons, due to their rigid structure, do not exhibit these issues and as a result are selected as the most suitable structure.

3.2.1 Exoskeletons

Exoskeletons are categorised into passive, active and semi-active devices based on how they generate and transmit force [31]. Passive exoskeletons use the principle of energy storage and release, typically through load-bearing elements such as springs and dampers. In contrast, active exoskeletons use external energy sources such as electrical, hydraulic or pneumatic actuators. Semi-active exoskeletons use a combination of both passive and active elements [32].

Passive exoskeletons weigh less and offer more flexibility compared to active ones. This is due to the lack of a power source and the requirements that actuators put on the systems. However, passive exoskeletons lack the precision of control, which is essential for applications that require close coordination with human movement. Passive devices use full mechanical systems, which leads to the inability to implement a closed-loop control, resulting in limited accuracy and adaptability. Conversely, active exoskeletons can follow human movements seamlessly because they use sensors that detect user's movements, forming a closed-loop control [33]. Although heavier, active exoskeletons can dynamically generate various torque profiles, unlike the passive ones, which depend on manual pre-stress settings [31]. When it comes to semi-active exoskeletons, they offer a balance between the simplicity of passive structures and the functionality of active ones. However, since the focus of this application is limited to a single limb segment, a hybrid solution can introduce unnecessary complexity without significant benefits. Therefore, the active type is chosen as the most appropriate solution.

3.3 Actuators

Actuators are the core components of active exoskeleton systems [33]. Their function is to convert energy into mechanical motion. As mentioned previously, actuators used in exoskeletons are based on electric motors, hydraulic systems or pneumatic mechanisms. The key difference between them relates to the energy source used to generate mechanical power. The choice of a power source is essential as it influences the system's mechanical structure, weight, control accuracy and forces it can generate [33; 34].

Hydraulic actuators are known for outputting high forces. However, these actuators are heavy due to the additional mechanical components, which makes them less suitable for portable applications. Moreover, there is a risk of hydraulic fluid leakage, which could negatively affect the user's safety. Due to these limitations, hydraulic actuators

are used mostly in lower limb and industrial exoskeletons [34; 33]. Figure 3.2 presents a typical example.

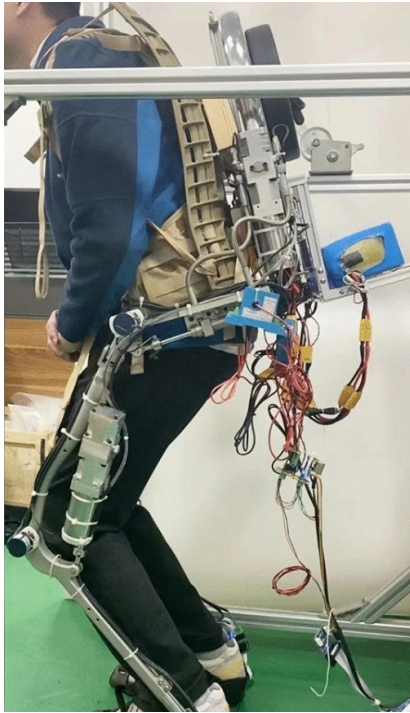


Figure 3.2: Hydraulic actuation system for lower-body exoskeletons. Adapted from Sun et al. [35].

Regarding pneumatic actuators, these are driven by compressed air, making them lighter than hydraulic ones. However, they lack precision, are noisy and need a bulky air supply system. Moreover, their force output is small, which makes them more suitable for finger and wrist rehabilitation [33]. Figure 3.3 presents a typical example.

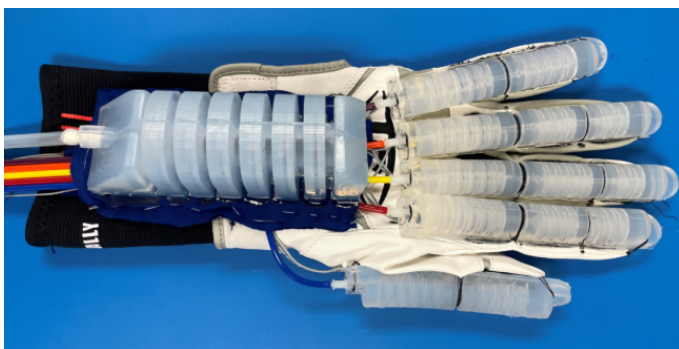


Figure 3.3: Pneumatically actuated soft robotic exoskeleton for hand and wrist rehabilitation. Adapted from Ridremont et al. [36].

As for electric actuators, these are often used for the upper-limb exoskeletons because they have high precision, and are fast and easier to control [25; 37]. Figure 3.4 presents a typical example. The types of electric motors used in exoskeletons are direct current (DC), servo and stepper motors. All of them are lightweight and compact.

However, the addition of the gears and controllers makes them bulky, which can cause tiredness and discomfort to the user.



Figure 3.4: Electrically actuated exoskeleton for upper-limb assistance. Adapted from Myomo Inc. [38].

Nonetheless, in comparison to the hydraulic and pneumatic actuators, electric motors are the most used in active exoskeleton designs [34]. This is due to their portability and high precision. Electric motors are also characterised by the highest responsiveness, which is essential for use in VE. Electric actuation is selected because it offers the optimal balance between control and performance.

To control the position and orientation of the exoskeleton, the number of degrees of freedom (DoF) must be defined. DoF is the summation of all independent movements, including linear displacements and rotations, that can be performed in the joints of the design. The design complexity increases with the increasing number of DoF [34]. The focus of this work is limited to the elbow joint, meaning reduction of complexity to 1 DoF (flexion-extension), also because of this consideration [39].

4 DESIGN AND IMPLEMENTATION

The developed exoskeleton is intended to operate in synchronicity with the human movement. Its purpose is to oppose the movement of the user's forearm while lifting an object in the VR environment. Lifting of an object is done by a rotational movement of the forearm as shown in Figure 4.1 .

As mentioned in the related work section, the tendency of a force to rotate an object about an axis is measured by the torque T [40]. In other words, torque needs to be rendered in order to simulate weigh sensation on user's forearm. Therefore, in order for the actuator/motor to deliver the right amount of torque, a control system is needed. The fully designed control system schematic for this par-

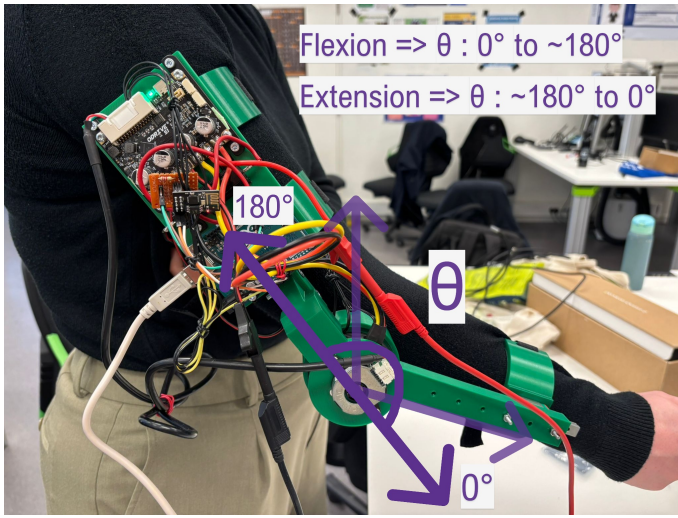


Figure 4.1: Exoskeleton attached to the user's arm, enabling rotational movement of the forearm through its full range of flexion and extension.

ticular case is shown on Figure 4.2. As seen, the control system has the angular position of the forearm as the input and the torque value as the output. In addition to angular position as an input, the controller block also receives input from the VR application. This way, the system is linked to the user's movements within the virtual environment.

The designed control system has aspects of both hierarchy-based and physical parameters-based control systems [30]. The high-level logic tasks are handled by the microcontroller, which processes sensor input to calculate the desired torque. This value is then transmitted to the motor controller, which acts as the low-level controller by applying the torque command, as the physical parameter, directly to the motor.

The implementation of the system is based closely on the flow of the control loop. It starts by calibrating the motor with the encoder using the Odrive graphical user interface (GUI) [41]. Next, the serial communication is set up between Odrive S1 [42] and Arduino Uno [43]. This is to set the ODrive in torque control mode. Then, another serial communication is established between the ESP-01 WiFi module [44] and Arduino Uno. Once the system is ready, the VR application (Unity) [45] communicates with it through a series of string-based commands sent over WiFi via the ESP-01. Figure 4.2 includes the two serial communication links and the WiFi connection using Hypertext Transfer Protocol (HTTP). The implementation of each step, as well as the individual blocks of the full control loop in Figure 4.2, are discussed in more detail in the following sections.

4.1 Actuator

It is important to select an actuator based on its ability to generate the required torque. In this application, the chosen object whose weight is simulated across different gravitational fields is an apple. The apple is chosen not only due to its symbolic association with Newton's law of gravitation, but also because it is a familiar object whose weight is easily perceived. Moreover, the mass of an apple is approximately 0.1 kg which is sufficient for simulating its weight across all the different planetary environments of the solar system.

The torque required to simulate lifting an apple is calculated using equation:

$$T = m \times g \times r \quad (4.1)$$

where $m = 0.1$ kg is the mass of the apple, g is the gravitational acceleration specific per planet, and $r = 0.3$ m is the distance from the elbow to the center of mass of the object held by the hand. The value of r is based on general anthropometric adult data, indicating that the average length from elbow to fingertip is approximately 43.7 cm for males and 40.7 cm for females [46]. Assuming the center of mass of the object is near the midpoint of this length, an estimate of 30 cm is used for r . This general value serves as a baseline. However it has to be mention that this value needs to be adjusted per user's specific limb measurements to ensure accuracy and comfort.

Table 4.1. shows the torque required to lift a 0.1 kg apple on each planet in the solar system, ranked from the lowest to the highest gravity [47].

Table 4.1: The torque required to lift a 0.1 kg apple on each planet in the solar system presented in order of increasing gravitational acceleration.

Planet	Gravity (m/s ²)	Torque (Nm)
Mars	3.70	0.111
Mercury	3.71	0.111
Uranus	8.72	0.262
Venus	8.90	0.267
Saturn	8.99	0.270
Earth	9.81	0.294
Neptune	10.99	0.330
Jupiter	23.1	0.693

As mentioned previously, the type of actuator chosen to be used in this application is the electric motor. In [34], authors detail the most used types of electric motors being brushed DC motors, brushless DC motors (BLDC) and series elastic actuators. In this case, a brushless DC motor is chosen. It requires less maintenance compared to the brushed DC motor while holding the same desirable characteristics. It also has a higher power-to-rate ratio due

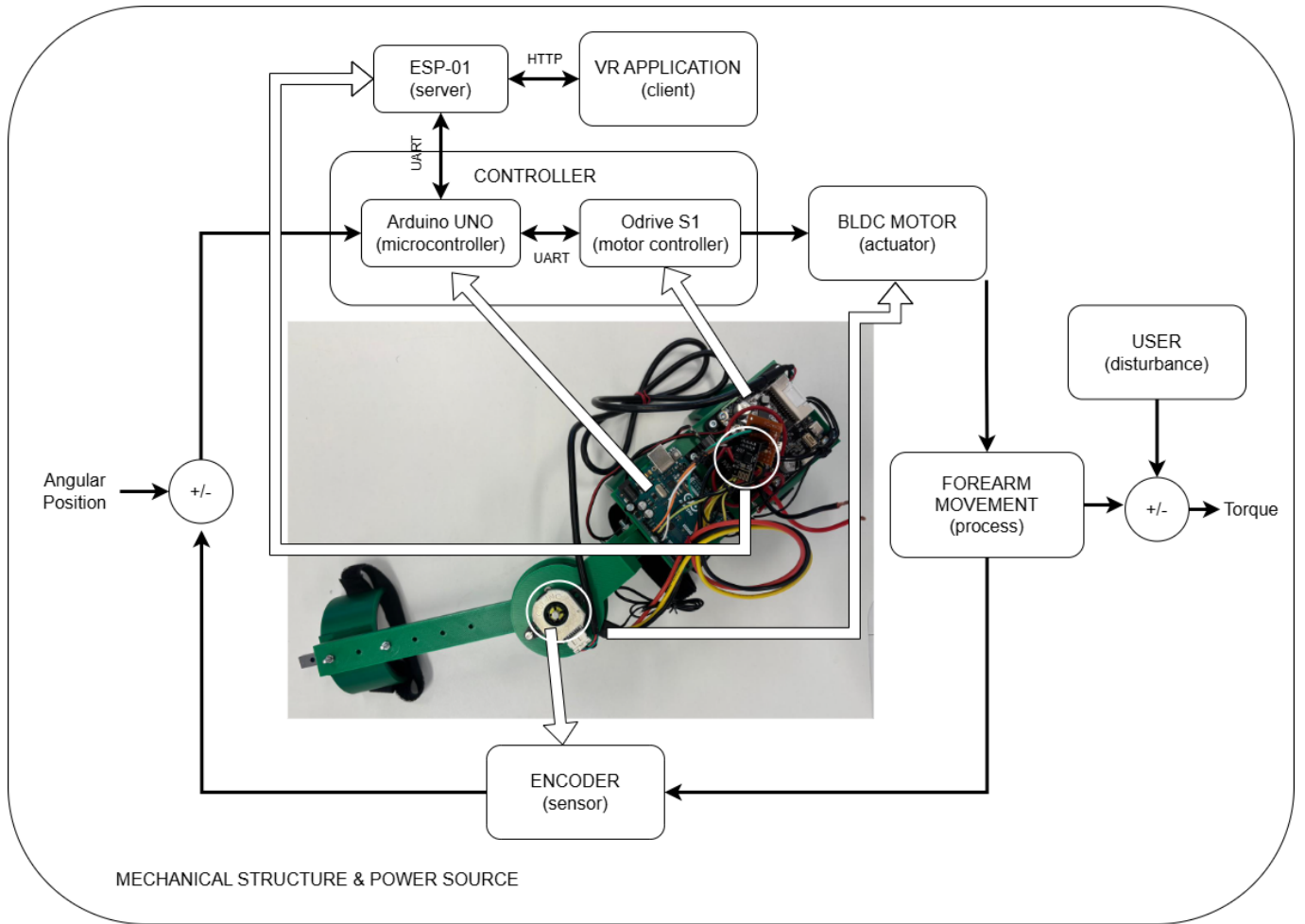


Figure 4.2: Control system schematic of the exoskeleton, using angular position input and VR commands to compute and apply torque via a feedback loop, enabling weight simulation on the user's forearm.

to the absence of brushes. In addition, there is no need to make the structure overly complex by using the string elastic actuators.

The torque values in Table 4.1 indicate that the BLDC motor must be capable of generating up to 0.693 Nm of torque corresponding to Jupiter. Based on this key factor, D5312s 330KV is selected [48]. In addition to meeting the torque demand, this motor offers several other important advantages. Its lightweight design enhances the portability of the exoskeleton, it provides ease of control for precise torque application, and it operates with low power consumption. Moreover, its cost supports the objective of keeping the overall system affordable. These combined features make the D5312S 330KV a suitable choice for this application. The motor specifications are summarised in Table 4.2.

Based on the motor specifications, the theoretical maximum continuous and peak torque values are calculated using the torque-current relationship:

$$T = I_q \times k_t \quad (4.2)$$

Table 4.2: Specifications of D5312s 330KV brushless DC motor [48]

Specification	Value	Unit
Speed Constant	330	RPM/V
Torque Constant	0.025	Nm/A
Speed / Torque Gradient	730.83	RPM/Nm
Pole Pairs	7	–
Phase Resistance	37	mΩ
Phase Inductance	23	μH
Continuous Current	30	A
Peak Current	60	A
Thermistor	NTC 10k 3435	–

where T is the torque in Nm, I_q is the current in A, and $k_t = 0.025 \text{ Nm/A}$ is the torque constant of the motor.

As shown in Table 4.3, the motor can sustain a maximum continuous torque of approximately 0.75 Nm under passive cooling. For short bursts, it can reach up to 1.5 Nm of peak torque, which is suitable for high-load, short-duration tasks. This confirms that the selected actuator is capable of delivering above the required 0.693 Nm for this applica-

tion.

Table 4.3: Theoretical torque values based on current ratings

Current Type	Current (A)	Torque (Nm)
Continuous	30	0.75
Peak	60	1.5

To control the torque in brushless DC motors, it is essential to supply the appropriate phase current, as torque is directly proportional to current. To calculate the required current to produce the given torque, the motor's torque constant k_t is used from Table 4.2. In order to reach a torque of 0.693 Nm, the motor phase current needs to be:

$$I_q = \frac{T}{k_t} = \frac{0.693}{0.025} = 27.72 \text{ A}$$

Delivering such a high and variable current requires more than a standard power supply. This is where a motor controller becomes essential when using a BLDC motor. As it not only amplifies current beyond what typical batteries or power supplies can provide, but more importantly, it dynamically regulates the current based on the user's forearm position to enable smooth and precise torque rendering. The selected motor controller is described in more detail in Section 4.3.1.

While torque in BLDC motors is directly proportional to current, supplying sufficient voltage is essential to overcome the back electromotive force (back-EMF), which is proportional to the motor's speed, and the resistive losses from the motor windings.

In practice, a supply voltage of approximately 16 V is recommended for this motor when used with a motor controller [42].

4.2 Sensor

As previously mentioned, accuracy and responsiveness of the system are critical in exoskeletons. To achieve this, a feedback loop is essential for effective control of the BLDC motor. The sensor in the loop enables the system to continuously monitor and adjust the motor's output based on real-time measurements. In this application, the motor's rotation directly corresponds to the user's forearm angular position. Therefore, the sensor used to provide this position feedback and enable precise control is chosen to be the AMT212B-OD absolute rotary encoder [49]. This particular encoder uses capacitive sensing, which makes it more resistant to dust, vibration, and temperature changes.

4.3 Controller

The controller block in the control system schematic consists of an Arduino Uno and an Odrive S1 [43; 42]. The Arduino Uno handles high-level decision-making and communication, while the ODrive S1 is responsible for direct motor control. Communication between the two is established via the Universal Asynchronous Receiver/Transmitter (UART) protocol [50; 51].

4.3.1 Odrive S1

The ODrive S1 is chosen for this application due to its compatibility with the D5312S 330KV BLDC motor's voltage and current requirements [42]. The ODrive S1 accepts a DC-bus voltage of 12–48 V. As previously mentioned, although, the motor requires 27.7 A phase current, this does not directly translate to the current drawn from the power supply. This is because the ODrive S1, like most motor controllers, performs current amplification via its inverter stage. As a result, the power supply current can be significantly lower than the motor current. The value of this current is calculated considering the forearm's angular velocity. Kinematic data for healthy adults report an average of forearm's angular velocity of 406–428°/s ($\approx 7\text{--}7.5$ rad/s) during activities of daily living [52]. Using $\omega = 8$ rad/s as a representative value gives the mechanical power:

$$P_{\text{mech}} = T \omega = 0.75 \times 8 = 6.0 \text{ W} \quad (4.3)$$

Running the Odrive at the lowest recommended bus voltage, $V_{\text{bus}} = 16$ V, the required supply current is

$$I_{\text{bus}} = \frac{P_{\text{mech}}}{V_{\text{bus}}} = \frac{6.0}{16} \approx 0.38 \text{ A} \quad (4.4)$$

Odrive S1 is also chosen due its ability to perform precise torque control. Unlike generic Electronic Speed Controllers, which typically lack advanced feedback mechanisms and operate in open-loop configurations, ODrive offers a cascaded control loop [53]. As shown in Figure 4.3, it allows independent control of torque, velocity, and position. This is crucial for rendering realistic force feedback in a controlled and responsive manner. Additionally, the controller is relatively lightweight and compact, making it suitable for this application.

Figure 4.3 presents a diagram of the cascaded control loops used by ODrive S1.

Moreover, ODrive S1 includes a GUI that allows real-time monitoring of torque, position, velocity, and current, as well as motor calibration [41]. This is important when it comes to monitoring and measuring the power consumption and

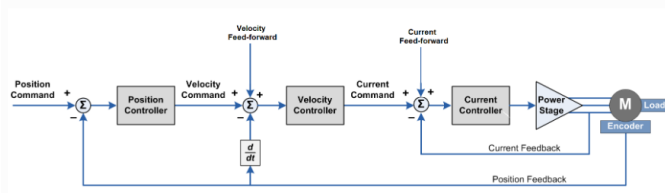


Figure 4.3: ODrive S1 Control Loop Overview [42]

accuracy of the system. It also supports UART used in this setup as a communication interface with the Arduino Uno [50].

For implementation, the ODrive is run from a lab power supply set to 31.5 V with the current limited to 3 A. Enough voltage for the controller to work while keeping the supply and motor safely within limits.

Before the system starts working, the motor must be calibrated with the encoder. The chosen encoder has been customized with ODrive-compatible firmware to ensure faster update rates. This ensures that the position and velocity of the motor is obtained accurately later on. The calibration is done using the ODrive GUI. To avoid repeating the calibration process through the GUI every time, the ODrive is programmed using the Python-based `odrivetool`. This allows saving the motor and encoder settings for automatic calibration at startup.

After calibration, the ODrive S1 is set to torque control mode in a closed-loop state. Closed-loop control is important because it uses feedback from the encoder to apply the correct torque dynamically based on the user's movement.

Listing 4.1: Odrive S1 setup

```
1 odrv.axis0.controller.config.  
  control_mode =  
  CONTROL_MODE_TORQUE_CONTROL  
2 odrv.axis0.requested_state =  
  AXIS_STATE_CLOSED_LOOP_CONTROL
```

4.3.2 Arduino Uno

To implement a closed-loop control system, a microcontroller is required to handle logic-level operations based on the sensor data. In this case, its primary role is reading the encoder values, VR application data and translating them into torque commands. In other words, this control unit acts as the interface between the high-level system logic and the low-level motor controller, the Odrive S1. The Arduino Uno (version : Rev3 SMD) is selected as the microcontroller board for this setup [43]. It is not only simple, reliable, and widely supported, but also recommended by

the ODrive developers for UART-based communication in prototyping environments [50].

The torque calculation logic is executed directly on the Arduino Uno as mentioned previously. It is calculated as followed:

$$T = mgl \left[1 - \sin\left(\min\left(\theta(t), \frac{\pi}{2}\right)\right) \right] \quad (4.5)$$

where m is the mass of the object whose weight is simulated, predefined in the Arduino code. The gravitational constant g is a dynamic value received from the Unity application, allowing the system to simulate different planetary environments. The lever arm length l represents the distance from the elbow joint to the hand (0.3m). The angle $\theta(t)$ corresponds to the user's forearm position. It is estimated as such:

$$\theta(t) = \theta_0 + \int_{t_0}^t \omega(\tau) d\tau, \quad (4.6)$$

where $\omega(t)$ is the measured angular velocity (rad s^{-1}), and θ_0 is the reference angle at time t_0 .

As shown from formula 4.6, the angular velocity is read from the motor's encoder and integrated over time. Since the user's forearm naturally moves within a limited range of motion and does not complete full rotations, the angular position is constrained within $\pm 180^\circ$ (± 0.5 revolutions), ensuring accurate tracking and preventing integrator drift over time. Direct position readings are avoided due to limited resolution. Limited resolution causes small forearm movements not to be detected accurately, leading to incorrect torque output. In contrast, velocity feedback is always fractional and stable, allowing smooth and precise estimation of motion. This is essential for rendering a continuous and natural-feeling torque response.

The applied torque depends on the angle of the user's forearm, following the physics of gravitational torque. The calculated angle is then clamped to a maximum value of $\theta_{\max} = \frac{\pi}{2}$, corresponding to 90° . The expression $1 - \sin(\theta)$ is used to produce a smooth reduction in torque. As the arm approaches the user's body, gravity has less effect on it, so less torque is needed [54]. The choice of clamping the angle at 90° is arbitrary and primarily serves to demonstrate how the torque decreases smoothly toward zero as the forearm moves closer to the body. This does not imply that all users reach exactly 90° during flexion, nor that the system assumes a fixed anatomical limit. In reality, the angular range is highly dependent on the chosen reference angle and individual biomechanics. For some users, the point at which the forearm contacts the upper body, representing full flexion may occur beyond 90° , potentially approaching 180° . Figure 4.4 done in Desmos illustrates

the theoretical relationship between torque (Nm) and angular position (radians) for the forearm exoskeleton under Jupiter's gravitational conditions, with a maximum torque of 0.693 Nm [55]. As shown, the torque decreases sinusoidally as the angle goes from 0 rad to 1.5 rad (90° or $\frac{\pi}{2}$).

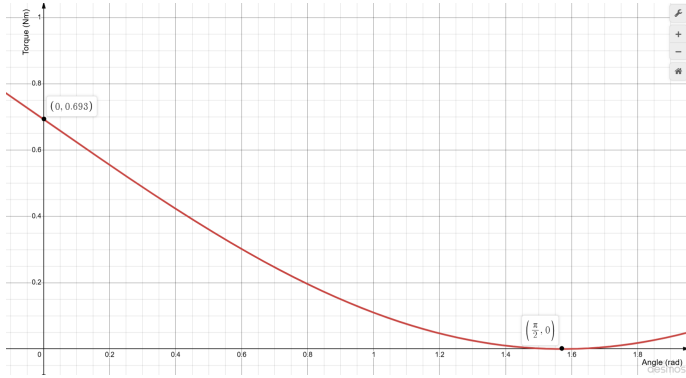


Figure 4.4: Theoretical torque (Nm) versus angular position (rad) for gravity compensation in the case of Jupiter when maximum torque is 0.693 Nm

The Arduino Uno communicates serially with both the ODrive and the ESP-01 WiFi module as shown in Figure 4.2. It sends real-time torque values to the ODrive and receives gravity values and control commands from Unity via the ESP-01. However, since Arduino Uno has only one hardware UART port, two software-based serial libraries are used to handle both connections simultaneously.

- `AltSoftSerial` is used for the ODrive connection [56]. The ODrive transmits 32-bit floating-point values at 19,200 baud several times per second, which cannot be reliably captured with the normal `SoftwareSerial`. Therefore, using `AltSoftSerial` ensures no data loss or glitches in encoder feedback. This library is interrupt-driven and uses a hardware timer, allowing it to handle high-frequency communication reliably even while the main loop performs floating-point calculations.
- `SoftwareSerial` is used for the ESP-01 module [57]. The ESP only sends short, infrequent ASCII strings such as "objectHeld", "systemShutdown", and a numeric gravity value at 9,600 baud rate. Therefore, a polling-based serial interface is sufficient for this low-bandwidth communication.

This dual-serial setup allows the Arduino to react to user interactions in Unity while maintaining precise and uninterrupted motor control via the ODrive.

4.3.3 ESP-01 WiFi Module

Since the Arduino Uno does not have built-in WiFi capabilities, an external module is required to handle wireless communication. The ESP-01 WiFi Module (ESP8266 Serial WiFi Module) is chosen for this purpose due to its compatibility with the Arduino environment and its ability to act as a lightweight web server [44]. It is a widely used solution that supports standard HTTP communication. HTTP is an application-layer protocol that relies on Transmission Control Protocol at the transport layer to ensure reliable communication between clients and servers [58]. This makes it well-suited for receiving commands from the Unity application over a local network and forwarding them to the Arduino.

To enable real-time communication between Unity and the physical exoskeleton system, the ESP-01 WiFi module is configured as an HTTP server. This setup allows Unity to send commands such as:

- "objectHeld" – starts torque rendering
- "systemShutdown" – stops torque rendering
- number – sets a new gravity value

over a local WiFi network. These commands are transmitted as URL parameters using standard HTTP GET requests, which are easy to construct and parse on both the Unity and ESP-01 sides.

Listing 4.2: Setting up ESP-01 as a WiFi server

```
1 ESP8266WebServer server(80);
2 WiFi.mode(WIFI_STA);
3 WiFi.begin(ssid, password);
4 while (WiFi.status() !=
   WL_CONNECTED)
5     delay(500);
6 Serial.println(WiFi.localIP());
7 server.begin();
```

The ESP-01 first connects to the local WiFi network and prints its IP address to the serial monitor. This IP is then used by Unity to target the ESP with HTTP requests. The Unity side is managed through a `NetworkManager` object, which uses a script called `ESPHttpClient`.

Listing 4.3: Unity Method for Sending an HTTP Command

```
1 public void SendCommand(string
   command) {
2     StartCoroutine(
       SendCommandCoroutine(
         command));
3 }
4
5 private IEnumerator
   SendCommandCoroutine(string
   command) {
```

```

6 |     string url = $"http://{
    |         espIPAddress}:{port}/
    |         command?msg={command}";
7 |     UnityWebRequest request =
    |         UnityWebRequest.Get(url);
8 |     yield return request.
    |         SendWebRequest();
9 | }

```

This script exposes a method `SendCommand(string command)`, which is later used by user interface (UI) elements in the Unity application to send HTTP requests to the ESP-01 over port 80. Commands are put in the URL and sent to the ESP.

4.4 Virtual Reality application

The VR application is developed in Unity 6 to enable integration of the exoskeleton within an immersive VR experience [45]. Unity was selected for this project due to its extensive documentation and tutorials. It has also been employed in academic projects involving exoskeleton systems, for example, render visual feedback and communication between submodules [59; 60].

To stream the Unity application, a head-mounted display (HMD) is required. The Meta Quest 2 set is selected as sufficient for this application [61]. It facilitates hand tracking, wireless operation, and is affordable and easy to use.

4.4.1 System Architecture and Functionality

There are a multitude of Unity application architecture styles suited for different requirements. This project uses an architecture style based on `MonoBehaviour` classes, which is a typical style for prototype projects [62]. In this architecture style, the core logic is tied to the `GameObject`s, which are the fundamental building blocks. Functionality is added to `GameObjects` by attaching different C# scripts.

The required application functionality can be divided into three main tasks: rendering objects, user interaction and communication with hardware. Objects can be modeled in Unity or Blender software. They can also be added to the project as pre-modeled assets from the Unity Asset Store [63]. When it comes to user interaction, it is typically handled by controllers or hand gestures. In this work, the user interacts with the VR environment via hand tracking because the use of controllers may skew the user's perception of weight. Moreover, using hand tracking makes the experience more realistic because, when grabbing and lifting, the user performs gestures similar to those in real life. As for communication with hardware, WiFi is used. The application detects user's activity and sends commands to

the ESP-01 module. Further details are explained in the subsection 4.4.3.

4.4.2 Virtual Environments

To show the range of torques possible to render with the exoskeleton, Jupiter, Mars and Earth environments are selected. The gravity constant on Jupiter is the largest among all planets in the Solar System, whereas on Mars it is the smallest. Earth is included in the simulation because its gravity has an intermediate value and is an environment that users are familiar with. User switches between the environments through the menu scene. The application is designed to facilitate interaction with objects. Therefore, in each of the three environments, the user finds a setup with an apple which the user can grab from the table and lift. Additionally, the application is extended with an experimental environment, which is prepared for the evaluation of the exoskeleton. Figure 4.5 presents the design of the virtual environments.

Each scene of the VR application has a similar structure. The main components of the scene hierarchy are:

- Terrain, lighting, and interactive objects;
- VR interaction manager;
- Hand gesture-compatible UI;
- Custom managers for dynamic changes (skybox, planet rotation, scene switching).

4.4.3 Integration with Hardware

The aim of the software-hardware communication is twofold. Firstly, when the user selects an environment in the menu scene, the corresponding gravitational constant must be communicated to the hardware to calculate the adequate torque value. Secondly, the software detects when the object is grabbed and released and, based on that, commands the initiation and termination of the actuation. As mentioned before, the WiFi communication is implemented via the HTTP protocol, in which the VR application acts as a client. On the other side of the two-way communication channel is the ESP-01 WiFi module, which acts as the server. To enable the HTTP client functionality of the VR application, an HTTP client script is attached to each UI element and to each apple. The script uses the `UnityWebRequest` class to initiate the GET request [64].



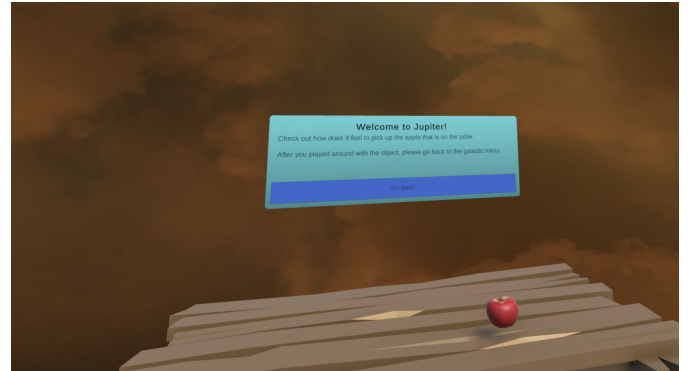
(a)



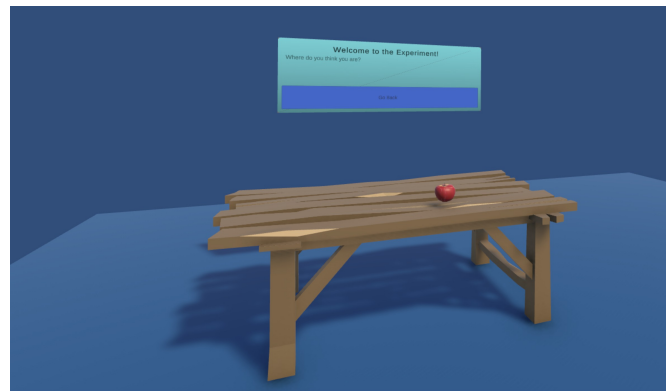
(b)



(c)



(d)



(e)

Figure 4.5: Environments of the menu scene, Earth, Mars, Jupiter and experiment respectively from the user's point of view in VR

4.5 Mechanical structure

4.5.1 Objectives and requirements

The primary goals of the mechanical structure are to provide structural support and establish an interface between the electronic components and the human body. The structure design is driven by the requirements defined in chapter 2, mainly ease of manufacturing and cost-effectiveness.

As mentioned in chapter 3, the design has to facilitate 1 DoF movement in the elbow joint and allow for interaction with the virtual objects through grabbing and lifting. More-

over, the design must be ergonomic and accommodate human use safely and comfortably. The design should be adjusted to the average human forearm dimensions. Modularity is also an important aspect, as it facilitates further customization and adaptability of exoskeletons.

4.5.2 Material considerations

When it comes to material selection, exoskeletons are traditionally made of metals like aluminium and steel [65]. While these materials offer high structural strength, they significantly increase the device's weight. Therefore, the

use of composite materials such as carbon fibre-reinforced polymers is prevalent [33]. To improve the adaptability to different body types, there is a growing trend toward 3D-printed exoskeletons.

4.5.3 Conceptual design

Figure 4.6 depicts a sketch of the initial idea. The device is strapped onto the forearm and actuated with two motors in parallel. However, there are multiple issues with this design. First, placing the actuator on the forearm increases the required torque output, as its mass adds to the load being moved. Secondly, driving two motors in parallel is challenging due to, for example, potential electrical mismatch and the need for individual control loops, which increases the control complexity. The selected motor, D5312s provides sufficient torque output, so another structure that incorporates only one motor can be considered.

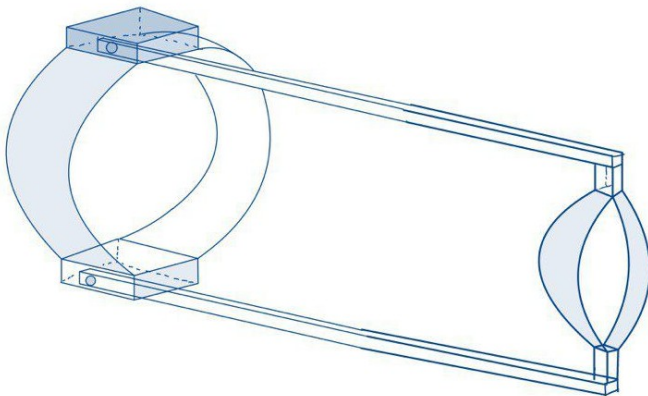


Figure 4.6: First design iteration

A different approach proposed by Auxivo AG, a spin-off company of ETH Zurich, fulfills the design requirements defined in section two [66]. Auxivo AG's digital open-source exoskeleton EduExo Maker is chosen as a baseline for this design [67]. The design is shown in Figure 4.7. In comparison with the initial design, the structure is fastened to both the forearm and arm of the user, so the actuator is held by the upper arm and, therefore, all the produced torque can be translated into resisting the user's forearm movement to render gravity.

The EduExo Maker design was adapted for implementing the D5312s motor, ODrive S1 and Arduino UNO boards. Figure 4.8 shows the first design iteration modeled in On-Shape [69].

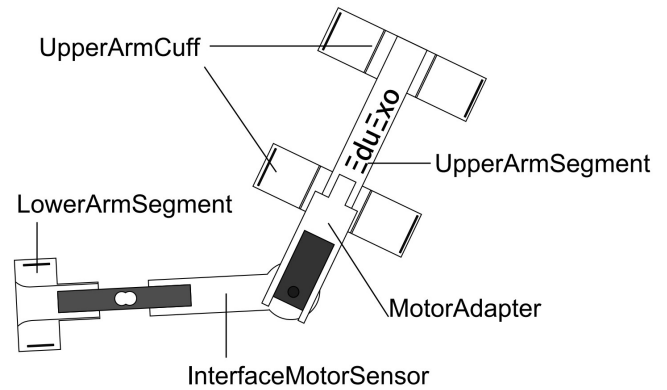


Figure 4.7: EduExo Maker Robotic Exoskeleton Design [68]

4.5.4 Manufacturing and Assembly

The mechanical structure was then 3D printed in Polylactic Acid (PLA), which contributes to its lightweight design. The device was assembled by screwing the 3D prints and electronic components together. The motor was connected to both the forearm and upper arm segments, and the electronics boards were positioned on the upper arm segment. Figure 4.9 shows the complete assembly.

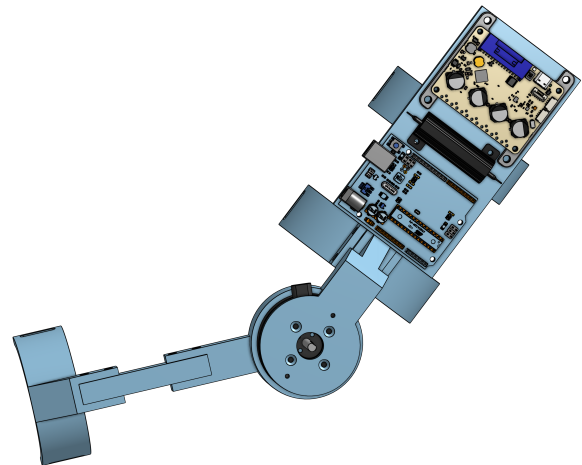


Figure 4.8: CAD model of the device

Initial tests of the assembly confirmed that the joint range of motion was within the expected limits. However, two main issues with the first prototype were identified. Firstly, the device was sliding down under the weight of the electronics boards and was not gripping well to the user's body. Secondly, during the flexion and extension movement, the motor wasn't always aligned with the elbow joint. To enhance the grip on the user's skin or clothing, a layer of neoprene rubber was added to the cuffs. The second issue required more careful consideration and research.

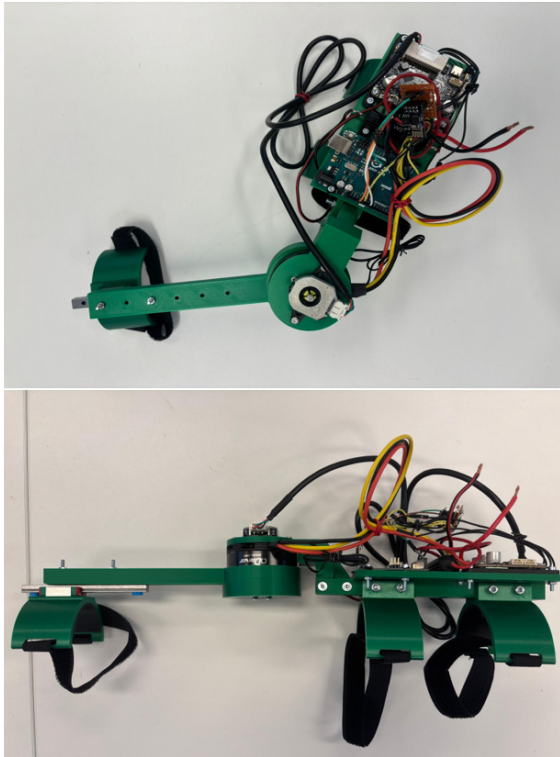


Figure 4.9: Assembled device

4.5.5 Misalignment

As mentioned before, exoskeleton devices are designed to operate in synchronicity with the human body. A common issue that can cause asynchrony is the misalignment of the rotation axis of the human joint and the device joint. There are two main causes of this issue. Firstly, the axis of rotation is difficult to precisely locate because it is covered by muscles and tissue [70]. Secondly, the center of rotation can migrate due to the human movement and change of angle [70]. Accounting for these issues leads to a better translation of the produced torque and increases the comfort and safety of the device. To mitigate the misalignment, a slider mechanism is introduced to the forearm segment. Its function is to decouple the motion of the exoskeleton from the user's arm. Figure 4.10 presents the CAD model of the second version of the prototype with the implemented slider.

5 EVALUATION

The system is evaluated both quantitatively and qualitatively based on the requirements defined in chapter 2. First, the power consumption of the system is evaluated based on current and voltage measurements. The thermal performance is assessed by conducting temperature measurements. The latency is measured to quantify the delay caused by the processing and between-module communication. The accuracy of actuation is quantified to assess

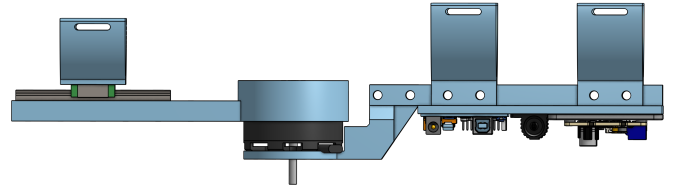


Figure 4.10: Second iteration of the model with slider implemented on the forearm segment to improve on the misalignment issue

the applied torque against the theoretically modeled one. To evaluate the robustness of the system, current is measured while applying a high load to the exoskeleton arm. To validate the accessibility and cost-effectiveness of the system, a cost analysis is performed. As for the qualitative evaluation, a 12-participant pilot user study is performed to gain insights into the user perception of the simulated weight, comfort and portability of the device.

5.1 Power Consumption

The power consumption of the system is evaluated by measuring the current and voltage drawn by each component during two distinct operating conditions: idle (I), where 0 Nm of torque is applied, and active (A), where maximum torque of 0.75 Nm is applied continuously. The maximum achievable torque value of 0.75 Nm is used to test the system's highest possible power usage. The full VR simulation lasts approximately 15 minutes. During this time, the system is only active when the user grabs and lifts the virtual apple, for the rest of the session, it remains in idle mode. As a result, the motor is active for roughly 1 minute in a typical session.

Current measurements for ESP-01 are performed using an oscilloscope equipped with a current probe to capture dynamic behaviour and instantaneous consumption. In parallel, steady-state current values are read directly from the power supply's built-in monitoring interface. Voltage measurements are taken using a digital multimeter across the supply lines of each component.

The resulting electrical power values are calculated using

formula:

$$P = V \times I \quad (5.1)$$

and are summarized in Table 5.1. This includes both the idle and active power draw for each system component. The total system power consumption is then calculated by summing the contributions of all components in each state.

Table 5.1: Measured power characteristics of each system component during idle (I) and active (A) conditions.

Component	Current (A)	Voltage (V)	Power (W)
Motor + ODrive S1	0.05	31.5	1.58 (I)
	1.14	31.5	35.91 (A)
ESP-01	1.68e-4	3.28	6e-4 (I)
	0.163	3.28	0.53 (A)
Arduino Uno	0.03	5.01	0.15 (I)
	0.12	5.01	0.60 (A)
Total			3.46 (I)
			37.04 (A)

As shown in Table 5.1, the combination of ODrive and motor accounts for the majority of the system's power consumption, drawing 35.91W when active. This brings the total power draw in the active state to 37.04W. When the system is idle, the power consumption drops significantly to 3.46 W. Given that the actuator is only active for about 1 minute during a typical 15-minute VR simulation, the average energy consumed per session is approximately 1.42 Wh (watt-hours). This low energy requirement makes the system well-suited for portable use with compact lithium-polymer (Li-Po) batteries.

5.2 Temperature

To evaluate the thermal performance of the system, the surface temperature of each component is measured with an infrared camera. The measurements are taken after 15 minutes of continuous simulation of maximum torque. The results are shown in Table 5.2.

Table 5.2: Measured temperature of each system component after 15 minutes of operation.

Component	Temperature (°C)
Arduino Uno	37.7
Break Resistor	25.8
ODrive S1	40.8
Motor	32.2
Encoder	31.5
ESP-01	43.9

The values in Table 5.2 are below the thermal shutdown thresholds [42]. This is expected, as the actuator is only

active for around 15 seconds during a typical session, limiting heat buildup. Therefore, no active cooling is required in this use case. Additionally, the electronics are not in direct contact with the user but mounted on the 3D-printed structure, minimizing any risk of thermal discomfort. Overall, the system's thermal performance is safe and consistent with findings from similar upper-limb exoskeletons in the literature [32].

5.3 Latency

To evaluate the system's real-time responsiveness, the total latency from a Unity command to torque actuation on the motor is measured. The measurement covers the full communication and processing pipeline, which Figure 5.1 details.

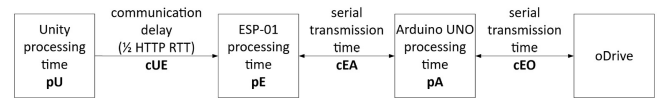


Figure 5.1: System latency measurement pipeline with intermediate elements

Each segment is timestamped using platform-specific clocks, `Time.realtimeSinceStartup` for Unity and `micros()` for Arduino. ESP-01 forwards the full timing message to Arduino, which reconstructs and prints the total end-to-end latency. The measurements are taken by sending a sequence of three commands: (1) a gravity value is sent, (2) the system is turned on and torque is rendered, and (3) the system is shut down. This pattern is repeated five times to capture the full range of system states and processing loads across 15 measurements. The collected timing results and the averages are summarised in Table 5.3.

Table 5.3: Latency measurements in milliseconds from Unity input to ODrive S1 torque output across 15 command cycles

	pU	cUE	pE	cEA	pA	cAO	Total (ms)
	3	54	1	14	10	1	83
	5	65	1	19	18	5	113
	6	21	1	28	18	5	79
	2	14	1	14	10	1	42
	3	34	1	19	20	5	82
	5	51	1	14	18	5	94
	3	59	1	14	10	1	88
	5	59	1	19	18	5	107
	5	33	1	14	17	5	75
	3	62	1	14	9	1	90
	6	10	1	19	17	5	58
	5	5	1	14	20	5	50
	2	27	1	14	10	1	55
	4	39	1	19	17	5	85
	5	40	1	14	17	5	82
Avg	4.1	38.0	1.0	16.9	15.3	3.7	78.1

As seen in Table 5.3, the measured end-to-end latency averaged 78.1 ms, with values ranging from 42 ms to 113 ms across the 15 command cycles. The largest variability occurred in the Unity-to-ESP-01 communication (cUE) and Arduino processing (pA). This is expected due to WiFi transmission delays (HTTP overhead, network interference) and the varying complexity of commands, capturing both idle and active states. Despite this, the average latency falls within the 25–100ms range commonly accepted in haptic and VR systems [11].

5.4 Torque Control Accuracy

Control accuracy determines how closely the system's torque output during a lifting motion follows the expected values over time and as a function of position. The current in the active state is recorded to evaluate the control accuracy. These measurements are performed on the partial system, including only the oDrive. The current values are then translated into the applied torque values using the equation 4.2. Figure 5.4 shows the resulting torque values, as well as the theoretical torque values, plotted against time and position for all VE. The expected torque values are calculated with the control equation 4.5, for which the position is modelled with a single cosine cycle, which resembles the forearm flexion-extension movement.

As seen in Figure 5.4 (a, c, e), the applied torque values reach the desired minimum and maximum values for all planet environments and the overall torque profile corresponds to the theoretically modelled one. However, there are mismatches between the applied and the theoretical

torque trajectories. The applied torque graph is not symmetrical and there is a slope mismatch between the theoretical and applied torques. These mismatches are related to the variability in the system introduced by the user. The slopes are related to the speed with which the user is moving their arm; the steeper the slope, the faster the movement. Since the user performs the flexion or extension parts of the movement at a different pace, the graph becomes non-symmetrical, and the slopes differ. The extension movement tends to be slower than the flexion movement, which can be concluded from the measurements. Moreover, the user tends to stay in the flexed position longer than what is modeled by the cosine cycle.

When it comes to the torque as a function of position graphs, in Figure 5.4 b, d and f, the applied torque closely follows the theoretical curve. However, as seen in the applied torque graphs, the user tends to go beyond the designed 90° in the flexion motion. For those values, the torque is clipped to 0 in firmware. As for the negative angular position values, these are recorded in the final part of the extension movement. These can be explained by the user going beyond the initial position, which is kept as a reference in the measurements. For these negative values, the torque should be clipped as well, to ensure the torque beyond the maximum value is not applied.

In the user evaluation, described in section 5.8, the whole system is used. This included the ODrive S1, motor, encoder, Arduino Uno, ESP-01 module linked to the Unity application. For this case, the applied and theoretical torque profiles, both as a function of time and position, are presented in Figure 5.3 and Figure 5.4b. The values are measured for two scenarios: one where torque is rendered over a 40° angular range, and another over a 90° range.

As seen from Figure 5.3 the applied torque closely follows the theoretical profile at lower angles (40°), but significant deviations occur during larger movements like 90°. These discrepancies are not due to encoder error. The encoder performs accurately when tested directly with the ODrive, as shown in Figure 5.4. Instead, the issue likely comes from polling delays on the Arduino Uno. Since the Uno cannot run tasks in parallel, alternating between reading from the ODrive and the ESP via software serial introduces timing issues. This affects how smoothly velocity is read and integrated, especially during longer motions, leading to sudden drops or early clipping in the torque output. As shown in Figure 5.4b, these polling delays cause sharp drops in the computed angle and torque, deviating from the expected smooth curve. At 40°, the motion is short enough that this effect is minimal, which is why that range was chosen for user testing.

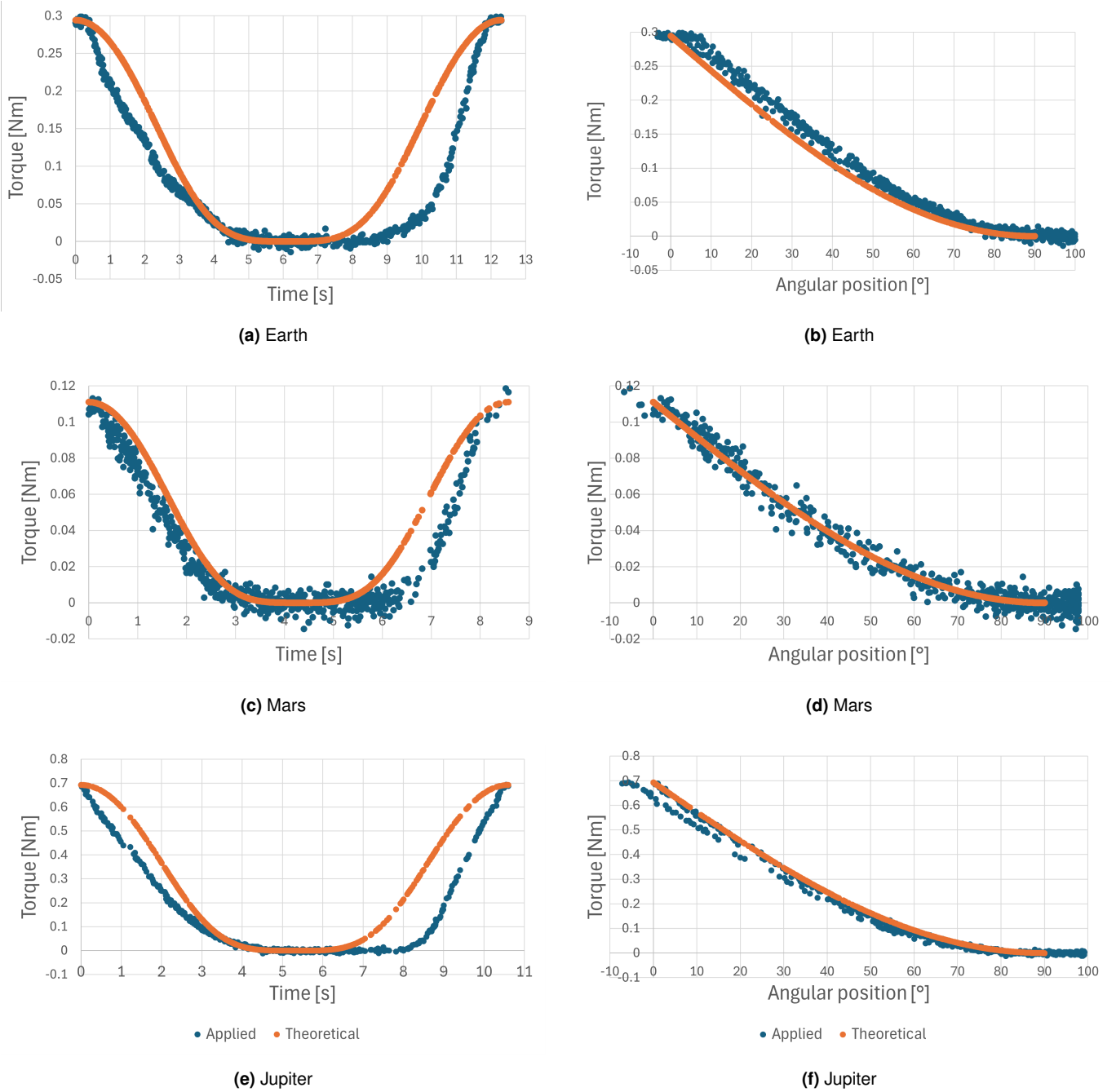


Figure 5.2: Applied and theoretical torque values as a function of time and position. These measurements were performed on the partial system, not including only the VR application and ESP-01.

5.5 Loading Test

To evaluate the exoskeleton's robustness, a loading test is performed, where a weight instead of the user's arm is applied. The exoskeleton set to generate the maximum torque of 0.75 Nm can hold a load of 424g. Figure 5.5 shows the current measurement of approximately 28 A, pulled from the motor, during the test. The current pulled from the power supply averages around 1.45A. Afterwards, a larger load of 1kg is applied to the structure. During this test, the exoskeleton with torque set to 0.75Nm was not able to hold the load. Additionally, when slowly re-

leasing the load, the power supply showed a large spike in the current reading. The system was pulling 1.6A from the power supply while the motor current was kept consistent at approximately 28 A. If kept continuously, this would make the Odrive S1 shut down from intensive current pulling. These results suggest that simulating heavier objects (above 0.42 kg) would require either a more powerful motor or mechanical adjustments, such as gear reduction, to achieve the necessary holding torque.

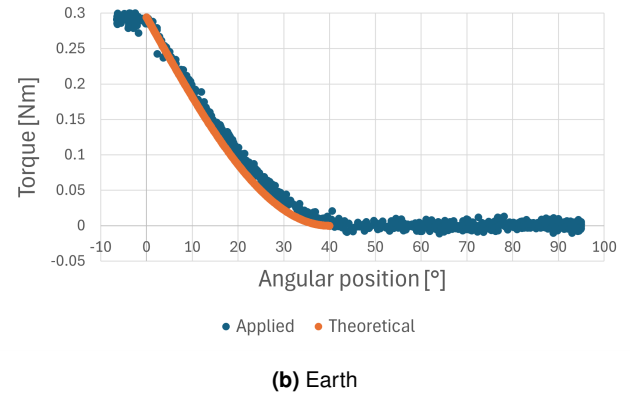
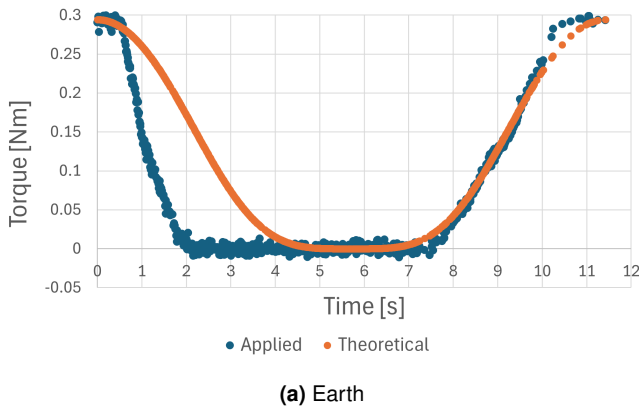


Figure 5.3: Applied and Theoretical Torque as a Function of Time and Position Using the Full System Setup (ODrive, Arduino, ESP-01, Unity); 0 Nm Torque Clipped at 40°, Representing the Range Used in User Evaluation.

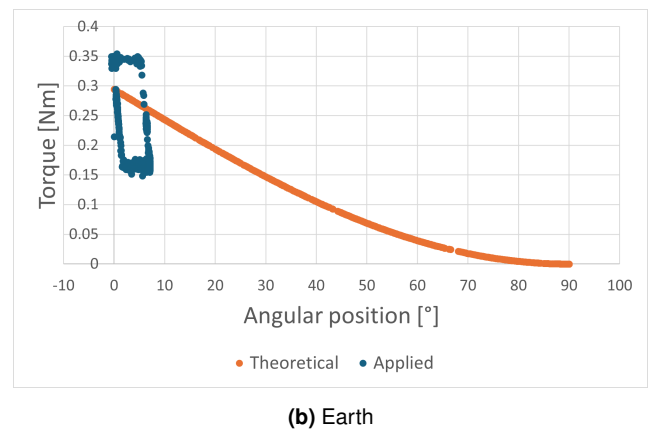
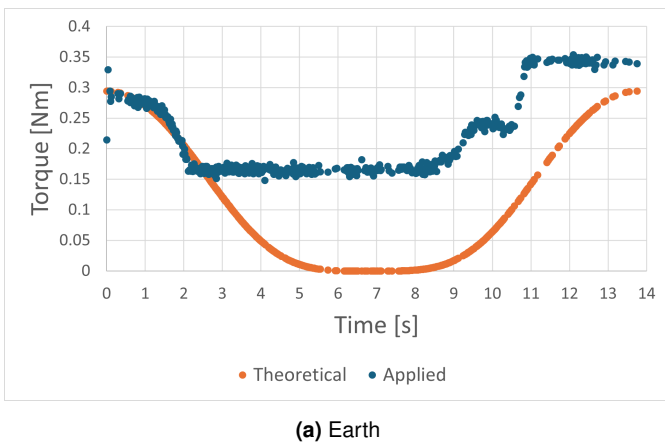


Figure 5.4: Applied and Theoretical Torque as a Function of Time and Position Using the Full System Setup (ODrive, Arduino, ESP-01, Unity); 0 Nm Torque Clipped at 90°, Highlighting Deviations Due to Communication Delays

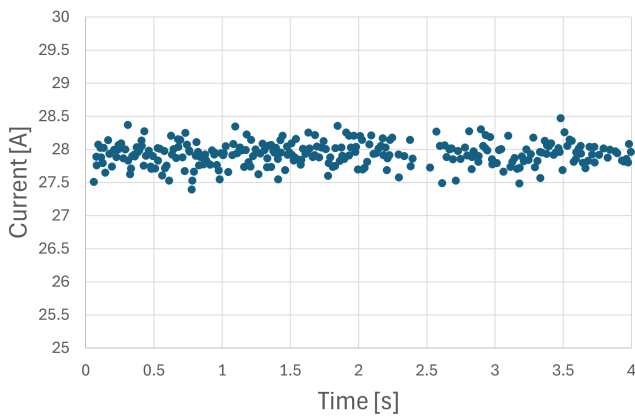


Figure 5.5: Motor current during the loading test at maximum torque of 0.75 Nm.

5.6 Cost analysis

A bill of materials 5.4 is compiled to evaluate the economic accessibility of the prototype. Every listed item is an off-the-shelf component that can be sourced from electronics and hardware distributors. The mechanical struc-

ture and the VR application were designed with Onshape and Unity, respectively, both of which are free for accredited academic use [71; 45]. The mechanical structure was fabricated by 3D printing, which is a globally available and low-cost manufacturing technique. The total hardware cost is calculated.

Table 5.4: Cost calculation

Component	Price (€)
Arduino Uno	29.3
Break Resistor	2.38
ODrive S1	149
Motor D5312S	59
Encoder	49
3D printing	30.84
ESP-01	2.66
Slider	8.21
Neoprene rubber	12.99
Total	343.38

The total hardware cost of 343.38 € is significantly lower

than that of commercial upper-limb exoskeletons, often priced in the thousands of euros [7; 72]. This makes the system not only affordable but also well-suited for academic research or prototyping purposes, where budget constraints are a common concern.

5.7 Safety

When the system is powered on but not actively rendering torque (idle state), the motor draws only :

$$I_{\text{idle}} = 0.081 \text{ A}$$

from the Odrive S1. This idle phase current is measured at a bus voltage of approximately 31.5 V and remains well below 1 A. This low and stable current confirms that no unexpected current spikes occur during idle operation, ensuring safe standby conditions. It is important to note that this is the motor's phase current, not the current drawn by the ODrive motor controller from the power supply.

For additional safety, the ODrive S1 controller includes support for a brake resistor [73]. A properly configured external brake resistor is used in this setup. This safety feature helps dissipate excess regenerated energy, which can occur during deceleration or backdriving, preventing voltage spikes or controller faults.

5.8 User study

As a final analysis, a pilot user study comprising 12 participants was performed to evaluate the user's perception of the simulated weight sensation, as well as the comfort of use and portability of the device. To support these factors, the exoskeleton's weight was measured to be 793 g.

During the experiment, data is collected through an online form. At the beginning, the participant is asked to complete their demographic data and is explained the purpose of the evaluation and the testing procedure. To assess if they identified the correct planet in the experimental environment, the participant's name is recorded. Each planet was equally represented as the experimental environment across participants. The user's arm length is measured because it is identified as a factor that may affect the comfort of use.

To perform the test, the user wears the exoskeleton and the HMD. Inside the simulation, the user is prompted to first explore the Earth, Mars, and Jupiter environments, which constitute the training phase of the evaluation. In each environment, the user is asked to pick up the apple with their arm straightened out and slowly perform the flexion movement, during which they are supposed to focus on the weight sensation. The user is supposed to move on

to the next planet environment once familiarized with the sensation. After familiarizing themselves with all three environments, the user enters the experimental environment and is asked to identify which planet they are on based only on the physical sensation that the exoskeleton creates. This concludes the simulation part of the evaluation. And afterwards, the user is asked to complete the second and third parts of the questionnaire, which relate to the sensation and comfort of use. Lastly, with the exoskeleton still on their arm and disconnected from the power supply, the user is prompted to move around the room for 2 minutes. Afterwards answer the questions regarding the device's portability. The duration of the user test is about 25 minutes. Figure 5.6 shows one of the users participating in the evaluation at the moment of interacting with the UI, while wearing the exoskeleton and the HMD.



Figure 5.6: User participating in the evaluation wearing the exoskeleton and the HMD

Eleven out of twelve participants responded that they were feeling as if they were picking up an apple. Everyone felt that the apple was the heaviest in the Jupiter environment, and ten out of twelve participants felt it was lightest in the Mars environment. The remaining two participants reported that this was the case in the Earth environment. The sensations experienced on Earth and Mars are more comparable due to the smaller difference in the applied torque than with Jupiter. In six out of twelve tests, the users correctly identified the planet in the experimental environment.

Figure 5.7 shows the six aspects related to the comfort of use (a-d) and portability (e-f) of the device that the users were asked to evaluate. The majority (83%) of the participants reported low to medium discomfort related to the stresses exerted by the device during actuation. Similarly,

most participants (84%) reported a low to medium fatigue after going through the testing simulation. When it comes to the ease of putting on and taking off, nine out of twelve participants rated it as easy to moderate. What is more, the majority (84%) reported that the exoskeleton fits their arm well or very well. When it comes to the portability evaluation, the majority of users reporting high to moderate ease of arm movement and carrying while wearing the exoskeleton.

6 DISCUSSION AND FUTURE WORK

For this particular use case, the evaluation of the proposed exoskeleton system showed that it performs efficiently in terms of power consumption, maintains safe temperature levels, and delivers effective torque rendering. However, a number of system-level and user-related limitations were also identified. These findings create key areas for improvement in future iterations, ranging from hardware upgrades and software enhancements to mechanical adjustments and user-specific adaptations. The following discussion outlines these aspects in detail and proposes potential solutions to address them.

As described in Chapter 5, the average energy consumption for a 15-minute VR simulation is approximately 1.42 Wh. This level makes it feasible to use Li-Po batteries instead of a power supply. Existing literature on upper-limb exoskeletons supports the practicality of using compact Li-Po batteries, with several systems employing similar battery types. [74; 75; 76]. While limited to four planets, results from the Jupiter test demonstrate that the system can be scaled to simulate all planets and celestial bodies in the Solar. Extending the simulation to cover these scenarios would increase the total power consumption. However, Li-Po batteries remain a viable solution, which is supported by previous long-duration studies of upper-limb exoskeletons.

For the current application, the system's thermal performance remains within safe limits. This is expected given the short actuation time. However, if the simulation duration were to be extended, the internal heating of the electronic components would increase. Under such extended operation conditions, integrating a thermistor into the motor system is recommended for real-time temperature monitoring and protection. ODrive S1 supports this feature and can automatically reduce current to prevent overheating [73].

In terms of system responsiveness, the measured end-to-end latency falls within the generally accepted range for VR and haptic feedback systems. The largest variability in latency occurred in the WiFi communication and Arduino

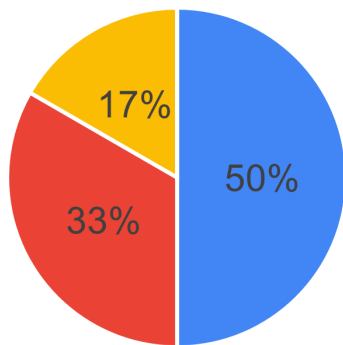
processing stages. This is expected, as WiFi transmission introduces delays due to network interference and HTTP overhead. Additionally, when the system is active and rendering torque, the Arduino is tasked with managing multiple operations. Such as continuously listening to the ESP-01, communicating with the ODrive S1, and processing torque control logic. This leads to higher processing delays compared to the system's idle state. Despite the timing inconsistencies, the average latency is still low enough, demonstrating real-time responsiveness suitable for VR simulations involving haptic feedback, especially given that the system is not visually linked to the user.

It is important to note that the latency measurements were obtained using software-based timestamping. While this approach provides a useful estimation, it is not the most accurate method for capturing precise delay metrics. More reliable measurements could be achieved using external hardware tools capable of capturing real-time signals. Nevertheless, the current software setup offers a reasonable approximation that helps identify system bottlenecks.

The bottlenecks related to latency and Arduino processing limitations directly impact torque accuracy. When only the Odrive S1 is linked to the Arduino, the applied torque generally follows the theoretical profile. Some slope mismatches and asymmetries occur due to user movement variability. Such as differences in speed between flexion and extension or the user holding the flexed position longer than modelled. While some variability is expected in user-driven motion, the emphasis here is on system-level behaviour. From a software-hardware integration perspective, the situation changes significantly when the ESP-01 is introduced. With both the ODrive S1 and ESP communicating over software serial, the Arduino's inability to handle parallel tasks introduces polling delays. Although the encoder was verified to perform accurately in isolation, the Arduino's limited bandwidth causes timing issues that affect velocity integration. This results in clipped or distorted torque output, particularly during longer or faster motions. All this points to the Arduino Uno as the key bottleneck affecting real-time torque accuracy in more complex scenarios. To address this, replacing the Arduino with a more capable microcontroller that supports multithreading or interrupt-driven communication is advised. Such an upgrade would allow the system to handle torque rendering, sensor input, and wireless communication more efficiently and in parallel. This would significantly improve overall system performance.

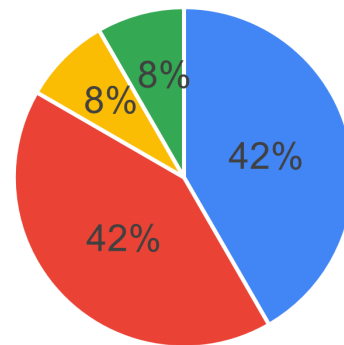
As mentioned previously, the variability in user-driven motion introduces another disturbance to the system. Since the user's movement is voluntary and unpredictable, relying on a single sensor to measure motion introduces limitations. As a result, the system currently applies torque

Did you feel any abnormal stresses when the exoskeleton applied force on your arm? From 1 to 10, how uncomfortable was it?



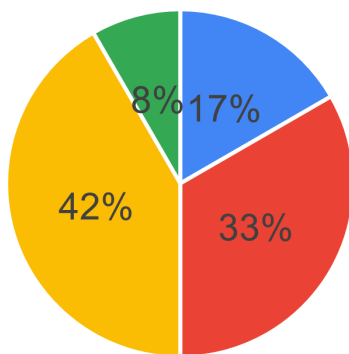
(a)

Is your arm tired after having the device on for the duration of the test? If so, rate it from 1 to 10.



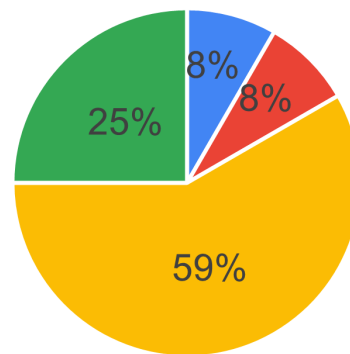
(b)

How would you rate the ease of putting the device on and taking it off from 1 to 10?



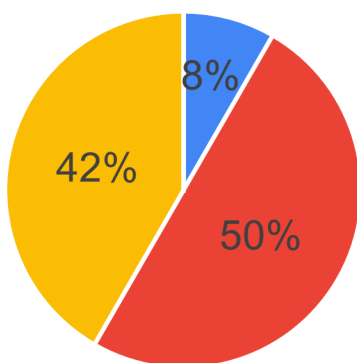
(c)

How well from 1 to 10 did the exoskeleton fit your arm?



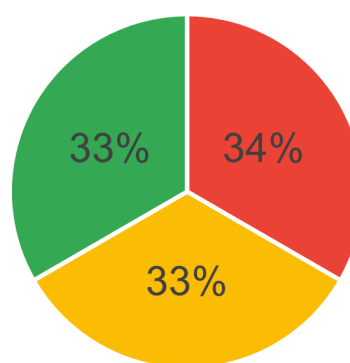
(d)

From 1 to 10 how easy was to move your arm around when the exoskeleton was on?



(e)

From 1 to 10, how easy was to carry the exoskeleton when walking?



(f)

■ 1-3 ■ 4-6 ■ 7-8 ■ 9-10

Figure 5.7: Results of the comfort and portability related parts of the questionnaire

in only one direction. Simply relying on the velocity sign to determine direction is unreliable, as small fluctuations or noise in the signal can lead to incorrect or unstable torque application. To address this, integrating an additional sensor that detects the end of a flexion or extension cycle would allow the microcontroller to automatically reverse the torque direction and correctly adjust the rendering range (from 0° to full flexion or extension). Since this range varies between users, such a sensor would improve realism and eliminate the need for manual tuning per individual.

From a mechanical design perspective, the current exoskeleton is lightweight and generally comfortable, but its load-bearing capacity is limited. To support the simulation of heavier objects, two solutions are proposed: either replace the current motor with one that offers higher torque (lower KV rating) or introduce a gear reduction mechanism to amplify the holding torque. To help manage the added weight that both approaches introduce, without negatively affecting portability, a shoulder brace that supports the exoskeleton's upper structure can be included. This would distribute the load more effectively and enhance user comfort during extended or higher-force simulations.

User testing reinforced the system's comfort and usability. Eleven out of twelve participants reported that the simulation felt realistic, particularly in the Jupiter environment, where the perceived heaviness was highest. Users identified the planets correctly in most cases, supporting the torque rendering's effectiveness. Furthermore, a majority reported low to medium discomfort and minimal fatigue, confirming that the device is suitable for sessions of 15 minutes on average. Since the device is working in synchronicity with the user's body, the evaluation depends as well on the physical characteristics of the participant, such as arm length. For example, one of the participants had a much shorter arm compared to the rest of the participants. They reported a higher discomfort related to the stresses applied by the exoskeleton and more difficulty with mobility and arm mobility while wearing the exoskeleton. This suggests that future iterations should account for anthropometric variability, possibly by allowing for adjustable segment lengths. Another important factor contributing to user comfort is joint misalignment, which was addressed through mechanical design. The slider mechanism in the forearm segment allowed the exoskeleton to move more naturally with the user's arm, reducing alignment issues and improving comfort and torque simulation.

In addition to the mechanical solutions, further improvements on the software side are possible. Currently, the applied torque assumes a fixed lever arm length, which may not accurately reflect different users' forearm dimensions or the position of the object being held. Future versions

of the system could include a configurable interface that allows users to specify their arm length. This would enhance the realism and precision of torque rendering while reducing the need for manual calibration.

Finally, the portability feedback was also promising. After disconnecting the power supply, users could walk wearing the exoskeleton without significant discomfort. This suggests that integrating Li-Po batteries in a location with proper weight distribution could make the system fully portable.

7 CONCLUSION

This Master's thesis presented the development of a forearm exoskeleton designed to simulate weight sensation under varying gravitational conditions. The proposed simulator introduces a novel approach to haptic feedback in VR, enabling the perception of weight through the application of variable torque. This technique contributes to the emerging field of embodied VR interactions, particularly for training scenarios.

Quantitative evaluations demonstrated that the device is capable of reliable torque control while maintaining low power consumption and acceptable latency levels. A preliminary user study provided encouraging insights into the system's ability to evoke realistic weight perceptions and distinguish between different simulated gravitational conditions. These findings support the feasibility of enhancing VR immersion through physical interaction at the forearm level.

The work also emphasises affordability and accessibility, showcasing that effective haptic feedback systems can be developed using commercially available components and electric actuator-based control, a method already familiar in the medical field. While the current prototype is functional, there remains significant potential for further improvements, particularly in enhancing portability, adaptability, and dynamic responsiveness for a wider range of users.

Overall, this thesis demonstrates that integrating weight sensation into VR experiences is both technically and economically feasible, paving the way for more immersive and physically engaging virtual environments.

ACKNOWLEDGEMENTS

The authors would like to thank the promotor and copromotor for guidance in this research, as well as study participants, friends and family, who contributed to the project.

LIST OF SYMBOLS

Symbols

T	Torque
m	Mass
g	Gravitational acceleration
r	Distance
P	Power
V	Voltage
I	Current
k_t	Torque Constant
θ	Angle
ω	Angular velocity
τ	Time

Acronyms

BLDC	Brushless Direct Current Motor
DoF	Degrees of Freedom
GUI	Graphical User Interface
HMD	Head-Mounted Display
HTTP	Hypertext Transfer Protocol
RTT	Round Trip Time
UART	Universal Asynchronous Receiver/Transmitter
UI	User Interface
VE	Virtual Environments
VR	Virtual Reality

BIBLIOGRAPHY

- [1] S. Zhang and W. Li, "Applying extended reality(xr) technology in commerce, management, and business applications: A survey," in *2024 4th International Conference on Computer, Control and Robotics (IC-CCR)*, pp. 108–113, 2024.
- [2] M. Holly, J. Pirker, S. Resch, S. Brettschuh, and C. Gütl, "Designing vr experiences – expectations for teaching and learning in vr," *Educational Technology Society*, vol. 24, no. 2, pp. 107–119, 2021.
- [3] M. C. Howard, M. B. Gutworth, and R. R. Jacobs, "A meta-analysis of virtual reality training programs," *Computers in Human Behavior*, vol. 121, p. 106808, 2021.
- [4] V. Garg, V. Singh, and L. Soni, "Preparing for space: How virtual reality is revolutionizing astronaut training," in *2024 IEEE International Conference for Women in Innovation, Technology Entrepreneurship (ICWITE)*, 2024.
- [5] D. Martin, S. Malpica, D. Gutierrez, B. Masia, and A. Serrano, "Multimodality in vr: A survey," *ACM Comput. Surv.*, vol. 54, 2022.
- [6] J. Huang, T. Tanino, K. Wada, and N. Tomokuni,

- "Measurement of the relationship between weight perception and upper limb movement," in *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1123–1128, 2012.
- [7] D. Pinto, M. Garnier, J. Barbas, S. H. Chang, S. Charlifue, E. Field-Fote, C. Furbish, C. Tefertiller, C. K. Mummidisetty, H. Taylor, A. Jayaraman, and A. W. Heinemann, "Budget impact analysis of robotic exoskeleton use for locomotor training following spinal cord injury in four sci model systems," *Journal of NeuroEngineering and Rehabilitation*, vol. 17, no. 1, pp. 1–10, 2020. Accessed: May 19, 2025.
- [8] A. D. André and P. Martins, "Exo supportive devices: Summary of technical aspects," *Bioengineering*, vol. 10, no. 11, 2023.
- [9] S. Massardi, D. Rodriguez-Cianca, D. Pinto-Fernandez, J. C. Moreno, M. Lancini, and D. Torricelli, "Characterization and evaluation of human–exoskeleton interaction dynamics: A review," *Sensors*, vol. 22, no. 11, 2022.
- [10] A. Baldassarre, L. G. Lulli, F. Cavallo, L. Fiorini, A. Mariniello, N. Mucci, and G. Arcangeli, "Industrial exoskeletons from bench to field: Human-machine interface and user experience in occupational settings and tasks," *Frontiers in Public Health*, vol. Volume 10 - 2022, 2022.
- [11] T. Muender, M. Bonfert, A. V. Reinschluessel, R. Malaka, and T. Döring, "Haptic fidelity framework: Defining the factors of realistic haptic feedback for virtual reality," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, (New York, NY, USA), Association for Computing Machinery, 2022.
- [12] A. Geris, B. Cukurbasi, M. Kilinc, and O. Teke, "Balancing performance and comfort in virtual reality: A study of fps, latency, and batch values," *Software: Practice and Experience*, vol. 54, no. 12, pp. 2336–2348, 2024.
- [13] H. Oh and W. Son, "Cybersickness and its severity arising from virtual reality content: A comprehensive study," *Sensors*, vol. 22, no. 4, 2022.
- [14] "A touch of virtual reality," *Nature Machine Intelligence*, vol. 5, no. 6, pp. 557–557, 2023.
- [15] Y. Shi and G. Shen, "Haptic sensing and feedback techniques toward virtual reality," *Research*, vol. 7, p. 0333, 2024.
- [16] E. Pezent, A. Israr, M. Samad, S. Robinson, P. Agarwal, H. Benko, and N. Colonnese, "Tasbi: Multisensory squeeze and vibrotactile wrist haptics for augmented and virtual reality," in *2019 IEEE World Haptics Conference (WHC)*, pp. 1–6, 2019.
- [17] P. Strohmeier and K. Hornbæk, "Generating haptic textures with a vibrotactile actuator," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, (New York, NY, USA), p. 4994–5005, Association for Computing Machinery, 2017.
- [18] W. N. Lim, K. M. Yap, Y. Lee, C. Wee, and C. C. Yen, "A systematic review of weight perception in virtual reality: Techniques, challenges, and road ahead," *IEEE Access*, vol. 9, pp. 163253–163283, 2021.
- [19] A. Adilkhanov, M. Rubagotti, and Z. Kappasov, "Haptic devices: Wearability-based taxonomy and literature review," *IEEE Access*, vol. 10, pp. 91923–91947, 2022.
- [20] D. Monteiro, H.-N. Liang, X. Wang, W. Xu, and H. Tu, "Design and development of a low-cost device for weight and center of gravity simulation in virtual reality," in *Proceedings of the 2021 International Conference on Multimodal Interaction*, ICMI '21, (New York, NY, USA), p. 453–460, Association for Computing Machinery, 2021.
- [21] A. M. Okamura, "Lecture 7: Kinesthetic haptic devices: Design and kinematics." <https://web.stanford.edu/class/me327/lectures/lecture07-prerecorded.pdf>, 2020. ME 327: Design and Control of Haptic Systems, Stanford University.
- [22] D. Verdel, S. Bastide, F. Geffard, O. Bruneau, N. Vignais, and B. Berret, "Reoptimization of single-joint motor patterns to non-earth gravity torques induced by a robotic exoskeleton," *iScience*, vol. 26, no. 11, p. 108350, 2023.
- [23] National Aeronautics and Space Administration, "Training for space: Astronaut training and mission preparation." <https://www.nasa.gov/centers/johnson>, 2006. Fact Sheet FS-2006-03-011-JSC, Johnson Space Center.
- [24] C. Liang, F. Jiang, D. Kawaguchi, and X. Chen, "A biomechanical simulation of forearm flexion using the finite element approach," *Bioengineering*, vol. 11, 12 2023.

- [25] T. Proietti, E. Ambrosini, A. Pedrocchi, and S. Micera, "Wearable robotics for impaired upper-limb assistance and rehabilitation: State of the art and future perspectives," *IEEE Access*, vol. PP, pp. 1–1, 01 2022.
- [26] S. Lee, G. Park, D. Cho, H. Kim, J.-Y. Lee, K. Suyoung, S.-B. Park, and J.-H. Shin, "Comparisons between end-effector and exoskeleton rehabilitation robots regarding upper extremity function among chronic stroke patients with moderate-to-severe upper limb impairment," *Scientific Reports*, vol. 10, 02 2020.
- [27] H. S. Lo and S. Q. Xie, "Exoskeleton robots for upper-limb rehabilitation: State of the art and future prospects," *Medical Engineering Physics*, vol. 34, no. 3, pp. 261–268, 2012.
- [28] P. Maciejasz, J. Eschweiler, K. Gerlach-Hahn, A. Jansen-Troy, and S. Leonhardt, "A survey on robotic devices for upper limb rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, vol. 11, no. 1, p. 3, 2014.
- [29] D. Chiaradia, M. Xiloyannis, M. Solazzi, L. Masia, and A. Frisoli, *Comparison of a Soft Exosuit and a Rigid Exoskeleton in an Assistive Task: Proceedings of the 4th International Symposium on Wearable Robotics, WeRob2018, October 16-20, 2018, Pisa, Italy*, pp. 415–419. 01 2019.
- [30] A. D. André and P. Martins, "Exo supportive devices: Summary of technical aspects," *Bioengineering (Basel, Switzerland)*, vol. 10, no. 11, p. 1328, 2023.
- [31] A. Voilqué, J. Masood, J. Fauroux, L. Sabourin, and O. Guezet, "Industrial exoskeleton technology: Classification, structural analysis, and structural complexity indicator," in *2019 Wearable Robotics Association Conference (WearRAcon)*, pp. 13–20, 2019.
- [32] K. Rykaczewski, "Thermophysiological aspects of wearable robotics: Challenges and opportunities," *Temperature*, vol. 10, no. 3, pp. 313–325, 2023.
- [33] D. M. G. Preethichandra, L. Piyathilaka, J. H. Sul, U. Izhar, R. Samarasinghe, S. D. Arachchige, and L. C. de Silva, "Passive and active exoskeleton solutions: Sensors, actuators, applications, and recent trends," *Sensors (Basel, Switzerland)*, vol. 24, no. 21, p. 7095, 2024.
- [34] M. Tiboni, A. Borboni, F. Vèrité, C. Bregoli, and C. Amici, "Sensors and actuation technologies in exoskeletons: A review," *Sensors*, vol. 22, no. 3, 2022.
- [35] M. Sun, X. Ouyang, J. Mattila, H. Yang, and G. Hou, "One novel hydraulic actuating system for the lower-body exoskeleton," *Chinese Journal of Mechanical Engineering*, vol. 34, no. 1, p. 31, 2021.
- [36] T. Ridremont, I. Singh, B. Bruzek, A. Jamieson, Y. Gu, R. Merzouki, and M. B. J. Wijesundara, "Pneumatically actuated soft robotic hand and wrist exoskeleton for motion assistance in rehabilitation," *Actuators*, vol. 13, no. 5, p. 180, 2024.
- [37] R. Gopura, D. Bandara, K. Kiguchi, and G. Mann, "Developments in hardware systems of active upper-limb exoskeleton robots: A review," *Robotics and Autonomous Systems*, vol. 75, pp. 203–220, 2016.
- [38] Myomo Inc., "Myomo – myopro powered arm brace." <https://myomo.com/>, 2024. Accessed: May 20, 2025.
- [39] M. A. Gull, S. Bai, and T. Bak, "A review on design of upper limb exoskeletons," *Robotics*, vol. 9, no. 1, 2020.
- [40] R. A. Serway and J. W. Jewett, *Physics for Scientists and Engineers*. Cengage Learning, 10 ed., 2018.
- [41] ODrive Robotics, "Odrive gui interface documentation." <https://docs.odriverobotics.com/v/latest/interfaces/gui.html>, 2025. Accessed 14 May 2025.
- [42] ODrive Robotics, "Odrive s1 datasheet." <https://docs.odriverobotics.com/v/latest/hardware/s1-datasheet.html>, 2025. Accessed 14 May 2025.
- [43] Arduino AG, "Arduino uno rev3 smd." <https://store.arduino.cc/products/arduino-uno-rev3-smd>, 2025. Accessed 14 May 2025.
- [44] Ai-Thinker, "Esp8266 serial esp-01 wi-fi module datasheet." https://components101.com/sites/default/files/component_datasheet/ESP8266%20Datasheet.pdf, 2018. Accessed: May 16, 2025.
- [45] U. Technologies, "Unity engine." <https://unity.com/products/unity-engine>, 2025. Accessed 14 May 2025.
- [46] B. K. and C. B., "Height estimation of an individual from forearm length in medical students," *Asian Journal of Medical Sciences*, vol. 13, no. 5, pp. 90–93, 2022.

- [47] NASA Goddard Space Flight Center, “Nasa planetary fact sheet.” <https://nssdc.gsfc.nasa.gov/planetary/factsheet/>, 2025. Accessed 14 May 2025.
- [48] ODrive Robotics, “D5312S 330KV Motor Specifications,” 2024. Accessed: 2025-05-14.
- [49] CUI Devices, “AMT212B-V Absolute Rotary Encoder.” <https://www.digikey.co.uk/en/products/detail/same-sky-formerly-cui-devices/AMT212B-V/8537386>, 2025. Accessed: May 15, 2025.
- [50] ODrive Robotics, “Arduino uart communication guide.” <https://docs.odriverobotics.com/v/latest/guides/arduino-uart-guide.html>, 2025. Accessed 14 May 2025.
- [51] Rohde & Schwarz GmbH & Co. KG, “Understanding uart communication.” https://www.rohde-schwarz.com/nl/products/test-and-measurement/essentials-test-equipment/digital-oscilloscopes/understanding-uart_254524.html, 2025. Accessed 14 May 2025.
- [52] H. A. Rahman, Y. C. Fai, and E. L. M. Su, “Analysis of human hand kinematics: Forearm pronation and supination,” *Journal of Medical Imaging and Health Informatics*, vol. 4, no. 2, pp. 245–249, 2014.
- [53] H. V. Rahul, R. Rachana, L. G. Vaishnavi, V. Nersara, G. K. Savyasachi, and A. V. D’Souza, “Electronic speed controllers: A review,” *International Journal of Engineering Research & Technology (IJERT)*, vol. 10, Nov. 2022. ICEI – 2022.
- [54] M. Paterna, S. Magnetti Gisolo, C. De Benedictis, G. G. Muscolo, and C. Ferraresi, “A passive upper-limb exoskeleton for industrial application based on pneumatic artificial muscles,” *Mechanical Sciences*, vol. 13, no. 1, pp. 387–398, 2022.
- [55] Desmos Inc., “Torque-speed curve visualization.” <https://www.desmos.com/calculator/1bw85mchnu>, 2025. Accessed 14 May 2025.
- [56] Arduino Team, “Altsoftserial - arduino documentation.” <https://docs.arduino.cc/libraries/altsoftserial/>, 2025. Accessed: May 21, 2025.
- [57] Arduino Team, “Softwareserial - arduino documentation.” <https://docs.arduino.cc/learn/built-in-libraries/software-serial/>, 2025. Accessed: May 21, 2025.
- [58] ScienceDirect Topics, “Hyper-text transfer protocol - an overview.” <https://www.sciencedirect.com/topics/computer-science/hyper-text-transfer-protocol>, 2025. Accessed: May 16, 2025.
- [59] P. Blazek, J. Bydžovský, R. Griffin, K. Mls, and B. Petersen, *Obstacle Awareness Subsystem for Higher Exoskeleton Safety*, pp. 59–71. 01 2021.
- [60] R. Dizor, A. Raj, and B. Gonzalez, “Deep reinforcement learning control of an avatar and exoskeleton,” in *2024 IEEE Research and Applications of Photonics in Defense Conference (RAPID)*, pp. 1–2, 2024.
- [61] M. P. Inc., “Meta quest — mr & vr headsets & accessories.” <https://www.meta.com/be/en/quest/>, 2025. Accessed 14 May 2025.
- [62] U. Technologies, “Monobehaviour class — unity manual.” <https://docs.unity3d.com/6000.1/Documentation/Manual/class-MonoBehaviour.html>, 2025. Accessed 14 May 2025.
- [63] U. Technologies, “Unity asset store.” <https://assetstore.unity.com/>, 2025. Accessed 14 May 2025.
- [64] U. Technologies, “Unitywebrequest class — unity scripting api.” <https://docs.unity3d.com/ScriptReference/Networking.UnityWebRequest.html>, 2025. Accessed 14 May 2025.
- [65] D. M. G. Preethichandra, L. Piyathilaka, J. H. Sul, U. Izhar, R. Samarasinghe, S. D. Arachchige, and L. C. de Silva, “Passive and active exoskeleton solutions: Sensors, actuators, applications, and recent trends,” *Sensors (Basel, Switzerland)*, vol. 24, no. 21, p. 7095, 2024.
- [66] Auxivo AG, “About auxivo.” <https://www.auxivo.com/about>, 2024. Accessed: 2025-05-04.
- [67] Auxivo AG, “Eduexo maker.” <https://www.auxivo.com/eduexo-maker>, 2024. Accessed: 2025-05-04.
- [68] Auxivo AG, *EduExo – The Robotic Exoskeleton Kit. Handbook and Tutorial Maker Edition*. Schwerzenbach, Switzerland: Auxivo AG, 2022. For educational use only. Contact: info@auxivo.com.
- [69] O. Inc., “Onshape — product development platform.” <https://www.onshape.com/en/>, 2025. Accessed 14 May 2025.

- [70] M. B. Näf, K. Junius, M. Rossini, C. Rodriguez-Guerrero, B. Vanderborght, and D. Lefeber, "Misalignment compensation for full human-exoskeleton kinematic compatibility: State of the art and evaluation," *Applied Mechanics Reviews*, vol. 70, p. 050802, 02 2019.
- [71] Onshape Inc., "Onshape - product development platform." <https://www.onshape.com/en/>, 2024. Accessed: 2025-05-18.
- [72] R. Cano-de-la Cuerda, A. Blázquez-Fernández, S. Marcos-Antón, P. Sánchez-Herrera-Baeza, P. Fernández-González, S. Collado-Vázquez, C. Jiménez-Antona, and S. Laguarda-Val, "Economic cost of rehabilitation with robotic and virtual reality systems in people with neurological disorders: A systematic review," vol. 13, no. 6, p. 1531, 2024. Accessed: May 19, 2025.
- [73] ODrive Robotics, "Odrive hardware configuration documentation." <https://docs.odriverobotics.com/v/latest/manual/hardware-config.html>, 2025. Accessed: May 18, 2025.
- [74] A. Sattar, "Hardware design and development for an upper-body exoskeleton framework," Master's thesis, California State University, Northridge, January 2025. Accessed: May 16, 2025.
- [75] A. González-Mendoza, I. Quiñones-Urióstegui, S. Salazar-Cruz, *et al.*, "Design and implementation of a rehabilitation upper-limb exoskeleton robot controlled by cognitive and physical interfaces," *Journal of Bionic Engineering*, vol. 19, pp. 1374–1391, 2022. Accessed: May 16, 2025.
- [76] A. Tyagi, "Design and development of a low-cost upper limb exoskeleton for rehabilitation and strength augmentation: A portable, arduino-controlled system with emg integration for assistive motion and human augmentation," April 2025. Accessed: May 16, 2025.

APPENDICES

A	GitHub repository	A-1
----------	------------------------------------	------------

APPENDIX A : GITHUB REPOSITORY

The full source code and supporting files for this project are available on GitHub.

- **Repository:** <https://github.com/martarozyczka/Virtual-Gravity---Master-Thesis>
- **Contents:**
 - /Arduino – Arduino and ESP code for motor control and subsystem communication
 - /Unity – Unity project files for the VR test environment