Development of a Wearable Embedded System providing Tactile and Kinesthetic Haptics Feedback for 3D Interactive Applications

Roumeliotis Michael

January 2021

School of Electrical and Computer Engineering,
Technical University of Crete

Thesis Committee
Aikaterini Mania, Associate Professor, Thesis Supervisor
Konstantinos Balas, Professor
Konstantinos Kalaitzakis, Professor

Abstract

Over the last few decades, there has been growth in the Virtual Reality market, which created the demand for more sophisticated systems that simulate the human senses. Even though the audio and vision technologies have improved, the haptic technologies were mainly used for mobile phones and gaming controllers. However, nowadays more and more researchers start focusing on designing wearable haptic interfaces in order to provide tactile and kinesthetic feedback to users of interactive 3D applications, either mobile, in Virtual Reality, or Augmented Reality. This thesis focuses on the implementation of such a haptic interface, which provides both tactile and kinesthetic feedback in 3D interactive applications. Specifically, it consists of an exoskeleton, which was designed in 3D CAD modeling software. This exoskeleton is placed on the top of the hand, and with five servo motors provide the kinesthetic feedback. The servo motors are controlled with a PWM/Servo driver board, which is connected with the Arduino microcontroller. Furthermore, the haptic interface provides tactile feedback to the user through fifteen coin vibration motors, where one was placed on each fingertip, one on the bottom part of each finger, five on the top part of the palm, and one on the right side of the palm. These motors are also controlled from the Arduino microcontroller and offer three different levels of vibration. In order to present these features, a 3D multimodal and interactive application was created with the Unity gaming platform, where the user was able to feel and interact with the virtual objects. The interactive application implemented could be easily ported as a 3D or gaming application for Virtual or Augmented Reality headsets or mobile platforms without major software modifications. Apart from the visual and haptic feedback, this application also provides coordinated audio feedback at the time when an interaction with the virtual object occurred. Last but not least, the hand was tracked and represented in this virtual environment through the device called Leap Motion Controller.

Keywords: Embedded, Haptics, Tactile Feedback , Kinesthetic Feedback, Leap Motion, Arduino, Unity3D

Acknowledgements

First of all, i would like to thank my supervisor, Associate Professor Aikaterini Mania for her continuous support on my thesis.

I would also like to thank Professor Aggelos Bletsas and Mr. Georgios Vougioukas for helping me print the circuit board.

Finally, i would like to express my love to my family and friends, who supported me all the years.

Table of Contents

List of Figures

List of Tables

1 Introduction

Even though humans have 5 senses, the market mainly consists of electronic devices that provide auditory and visual feedback. However, over the last few years, there has been growth in the development of haptic technology. Haptic technology refers to any technology that simulates the sense of touch and the interaction with objects.

This technology has been used in applications like teleoperation, mobile phones, and virtual reality. Specifically, in teleoperations, the users can remotely control a system with a haptic interface and interact with the environment from a safe distance. In mobile phones, the users receive haptic feedback in the form of vibrations when the screen is touched. As for virtual reality, the haptic feedback is applied to the user through controllers.

However, hands are unique on every person(size and shape) and can perceive a wide range of feedback, which makes the development of haptic devices more complex. Until very recently, these devices were grounded, which did not offer wearability to the user. Therefore, the motivation of this thesis was to develop a wearable system, that provides a more realistic sense of touching and interacting with virtual objects.

1.1 Contribution

This thesis presents a wearable embedded system, that can provide both tactile and kinesthetic feedback in 3D interactive applications. The purpose of this interface is to simulate the touching and manipulating of virtual objects more realistically. The following are the capabilities of the implemented design:

- **15 different points of tactile stimuli**

  The embedded system provides vibrotactile feedback on the user's hand in 15 different points, one on every fingertip, one on the bottom part of every finger, five on the top part of the palm, and one on the side of the palm. On top of that, each motor is enabled independently from the others, thus not all motors vibrate together when the object is touched by the index only for example.

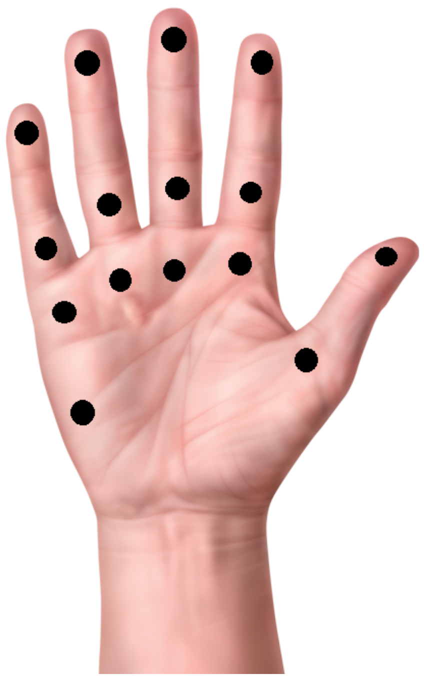


Figure 1: Position of vibration motors on the hand

- **3 different levels of vibration intensity**

The user can distinguish the power of the contact with the virtual object by adding a low level of vibration for light touching, a medium level of vibration for normal touching, and a high level of vibration for hard touching.

- **Exoskeleton for the kinesthetic feedback**

  This interface provides the user with kinesthetic feedback by adding one servo motor on every finger. The exoskeleton will be controlled by the servo motors and when there is contact detected, the movement of the fingers will stop.

- **Wearability by adding a Printed Circuit Board**

  By using a printed circuit board instead of a breadboard for the connection of the electronic components, the total weight and bulkiness of the device are diminished. Thus, the wearability of the haptic device is improved.

Then, this wearable embedded system is connected with a 3D application, which was created in Unity, via serial communication from a USB cable. The representation of the hand in the application is accomplished with a commercial solution, the Leap Motion.

Furthermore, when there is a collision between the hand and the object in the virtual environment, some characters are sent to the microcontroller(Arduino Mega 2560) via the serial port, and the Arduino, depending on which characters are sent, enable the appropriate vibration motor. The intensity of these vibrations depends on the power of the contact. Additionally, to provide kinesthetic feedback, when the user grabs a virtual object some other characters are sent to the Arduino, which rotates the servo motors(which turn the 3D printed part) in order to block the finger movement. The servo motors are controlled from the PCA9685, which is a PWM driver and is capable to drive a lot of servos at the same time.

The haptic feedback device is presented in figures 2,3 and its components are explained below:

- In figure 2, [1] is the exoskeleton, which was 3D printed, and the 5 servo motors that are connected with PCA9685.

- In figure 2, [2] is the PCA9685 and the printed circuit board, which consists of 15 n-type MOSFETs, 100pF capacitors, 50KOhm resistors, and Schottky diodes. The PCB is connected with the pair of cables of every vibration motor, the PWM pins, the 5V, and GND from Arduino. Furthermore, the PCA9685 is connected with the 5 servo motors. In order to power these motors, an external power of 5V and 2.5A was used.

- In figure 2, [3] is the Arduino Mega 2560 Rev3, which is powered with 5V from the USB port. Also, it powers and sends information to the PCA9685 board through the 5V, GND, SDA, SCL pins.

- In figure 3, the Coin Vibration Motors are presented, which were attached on the bottom side of the glove with glue. Every motor has two cables, which are connected to the printed circuit board.

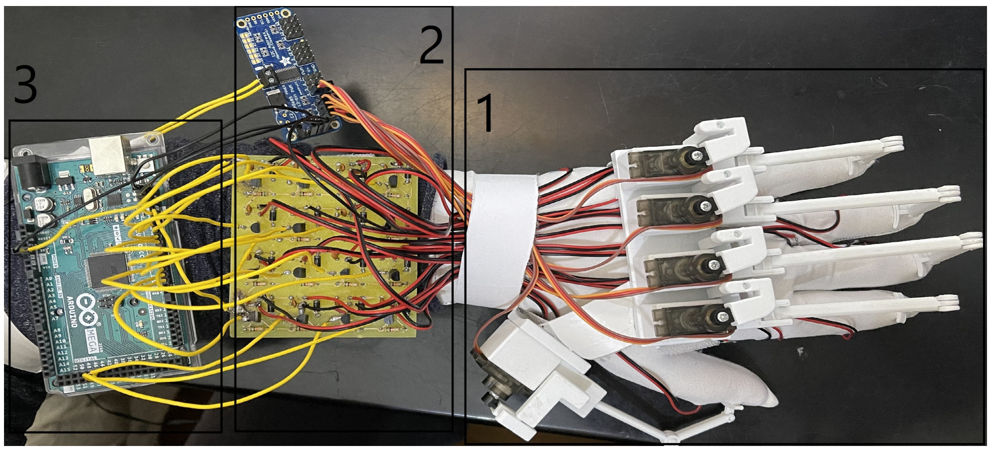


Figure 2: The top view of the haptic device

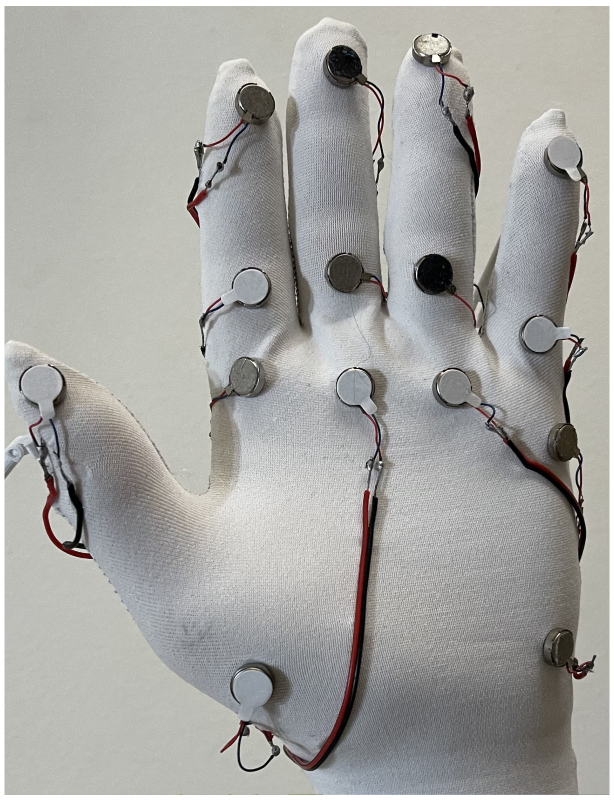


Figure 3: Bottom view of the haptic device

1.2 Thesis Structure

The thesis structure consists of the following 7 Chapters:

- The first Chapter is the Introduction, where the motivation for this thesis is explained. Then, the implementation of this thesis' haptic feedback device was presented along with its features.
- The second Chapter is the Existing Work and Background, which was the base for this thesis. Specifically, it presents the types of haptic feedback, the hardware components that are mainly used in haptics, the tracking technologies for the representation of the hand in virtual environments, some related work in haptic devices, and the previous implementations that were made in the Technical University of Crete.
- The third Chapter is the Glove Design and Implementation, where the whole embedded system was presented. Specifically, it analyzes the program that was used to create the 3D parts of the exoskeleton and the process of how it was done. Then, it explains why each part was chosen and presents a circuit for the vibration motors. Moreover, it analyzes the

program that was used to design the printed circuit board and how it was accomplished. Also, it presents how the microprocessor was programmed to offer haptic feedback.

- The fourth Chapter is the 3D Demo Application. This chapter presents the programs that were used to create the application's 3D models and the application itself. Then, it explains how the hand is detected in the application and how the collision's info is sent from the application to the microcontroller. Furthermore, it analyzes the application's environment and how haptic feedback was provided to the player through it.
- The fifth Chapter is the Evaluation of the application, where a questionnaire was given to the users. Then the results of this questionnaire were analyzed and presented in graphs.
- The sixth Chapter is the Conclusion and Future Work. This chapter presents the conclusions and the problems that were detected after implementing this thesis' haptic device. Then, some ideas were suggested in order to improve this design with some further implementation.
- The seventh Chapter is the References and Bibliography of this thesis.

2 Existing Work and Background

2.1 Types of Haptic Feedback

In order to implement haptic technology, researchers create special devices that provide the appropriate feedback to the user. The type of feedback that can someone find in these devices is **kinesthetic** and **tactile**. Tactile feedback refers to the things you feel in your fingers—vibration, softness, hardness, warmth, while kinesthetic is associated with the things you feel from sensors in your muscles, joints, tendons—weight, stretch, shape, etc.

2.2 Hardware Components used in Haptics

2.2.1 Tactile Feedback

Due to their small size and enclosed vibration mechanism, coin vibration motors are a popular choice for many applications. They are mainly used in applications like smartwatches and other wearable devices to provide tactile feedback. These vibration motors divide into two categories—**ERM**(Eccentric Rotating Mass) and **LRA**(Linear Resonant Actuator).

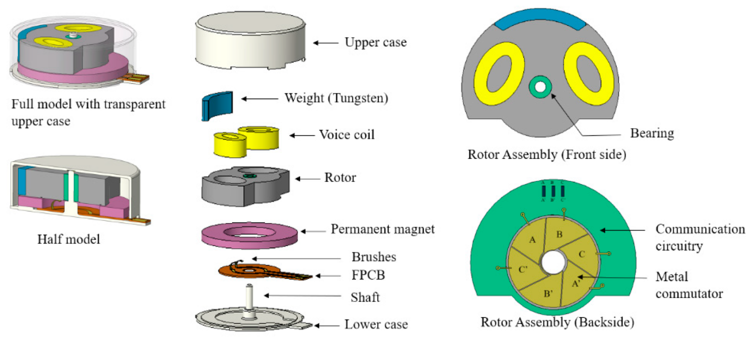


Figure 4: ERM Coin Vibration Motor

Eccentric Rotating Mass Motors [1] are small and lightweight DC motors that are widely used in mobile phone technology. They consist of an offset non-symmetric mass that is attached to the motor shaft. When this mass rotates, its centripetal force is asymmetric, resulting in a centrifugal force that displaces the motor. This displacement of the motor causes vibration.

The vibration is a Driven Harmonic Motion, which means that an external power causes it and the movement of the vibration could be represented as a sinusoidal wave. The frequency of the wave is the frequency of the vibration which results from the applied voltage. The equation for the centrifugal force is:

$$F = m\omega^2 r$$

From the centrifugal equation we could derive that if the voltage is increased, the vibration frequency (ω) will increase along with the vibration amplitude(F).

Linear Resonant Actuators [2] is a recently developed alternative to ERM. These motors consist of a movable mass, permanent magnet, voice coil, and spring. The voice coil creates a magnetic field, which forces the magnet to move, thus compresses or releases the spring (attached on top of the magnet). However, they require an AC signal to switch the magnetic field direction to produce the vibration.

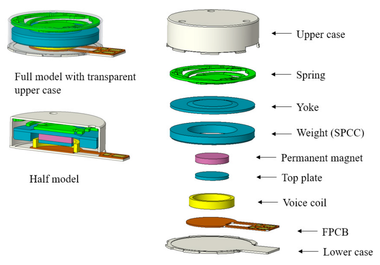


Figure 5: LRA Coin Vibration Motor

Furthermore, AC signals can have an amplitude and a frequency. The vibration output is affected by the amplitude. Although, these motors have to work on the resonant frequency because their efficiency and performance drop instead.

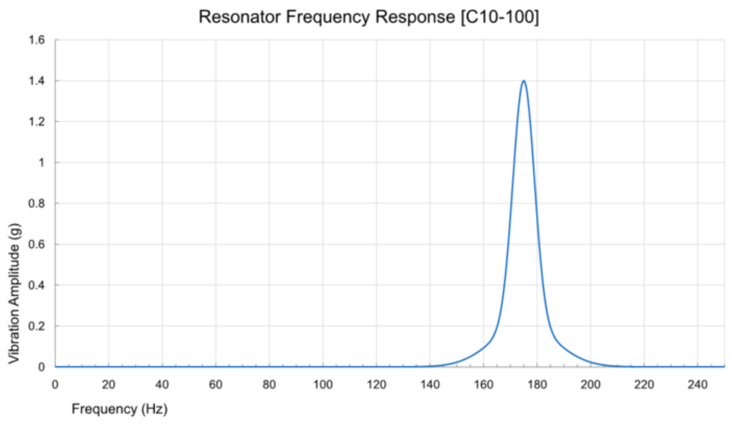


Figure 6: Performance of LRA in frequency

This thesis used ERM motors to achieve tactile feedback because they are more available in the market, they are powered with DC, and the circuit to drive them is not as complex as LRA's.

2.2.2 Kinesthetic Feedback

The haptic applications that use kinesthetic feedback, look to provide force to hands or fingers to recreate the virtual object. The most popular way that this is accomplished is with electric motors(servos).

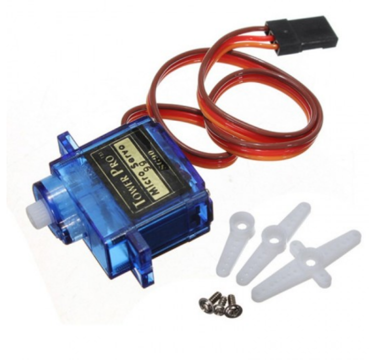


Figure 7: Servo Motor

Servo Motors are small-sized and energy-efficient components that have been around for a long time. The circuit of the servo is built inside the motor unit, which gives motion to the shaft that is mounted on top of the motor.

The basic components that the servo is consisted of are a small DC motor, a potentiometer, and a control circuit. When the DC motor rotates, the potentiometer's resistance changes. Then, the control circuit can regulate the movement and its direction. Once the shaft reaches the desired position, the power supplied to the motor is stopped. The control of servos is achieved by sending an electrical pulse and the movement is limited to only 180^o.

2.3 Tracking Technologies

In haptic interfaces, a very important aspect is to detect the position and rotation of the user's hand in space. This could be accomplished in various ways like optical tracking solutions, commercial solutions, IMU tracking systems, flex sensors, etc.

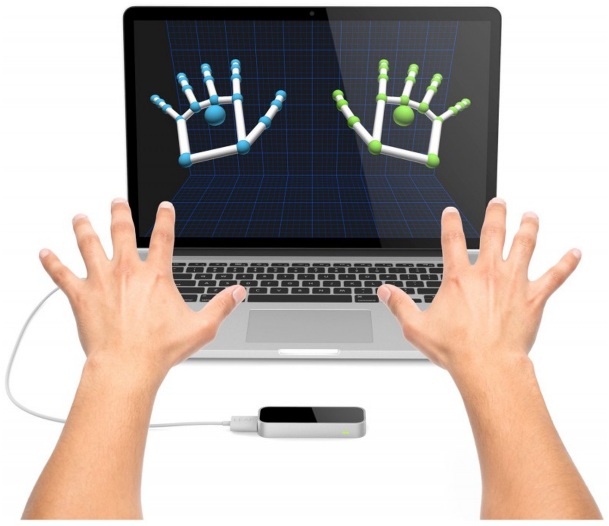


Figure 8: Leap Motion Controller

Leap Motion Controller [3] is an optical tracking solution, that can offer all 27 Degrees Of Freedom of the hands. This device works with two cameras and three infrared LEDs, that track infrared light with a wavelength of 850 nanometers. Also, the controller is capable to track hands up to 60cm above the device and within 140^o field of view. Then, the data is sent via USB to the Leap Motion Software, which processes the images and then reconstructs a 3D representation of what the device sees.

Figure 9: Oculus Touch Controller

Oculus Touch [4] is a commercial solution and provides 3/5 Degrees Of Freedom for every hand. Also, it consists of two handheld units, which have an analog stick, three buttons, and two triggers. Each unit has a ring on the upper side of the device, that contains infrared LEDs. Those LEDs are the ones responsible for the tracking of the controller's position.

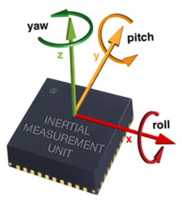


Figure 10: Inertial Measurement Unit

Inertial Measurement Units(IMUs) [5] are electronic devices, that measure a set of factors with the usage of some tools inside them and offer up to 10 Degrees Of Freedom. These tools could be an accelerometer to measure velocity and acceleration, a gyroscope to measure rotation and rotational rate, and sometimes a magnetometer to establish the direction.

This thesis used the Leap Motion Controller to track the hand and the fingers movement because it does not need to be placed on the hand, which would increase the weight and the total complexity of the design.

2.4 Related Work

Over the recent years, the field of haptics is developing and has various applications in other fields—video games, mobile devices, virtual reality, robotics, simulators, medicine, etc. This section reviews the literature on haptic feedback devices.

To begin with, Paccheriotti et al. [6] provided an in-depth review of haptic systems focusing on their ability to be wearable. At the start, they categorized them based on the type of tactile interaction they provide to the wearer, the area they apply these stimuli, and their mechanical properties. Then, they presented four requirements that every wearable device should meet, which are small form factor, low weight, no limits at the motion of the wearer, and comfortability. At last, they reviewed and explained the mechanism of various devices dividing them into those used on the fingertip and those used on the whole hand.

The first haptic device that will be presented in this thesis is made by **HaptX** [7]. They implemented a haptic glove that can offer both tactile and kinesthetic feedback. For the tactile stimuli they use a flexible array of pneumatic actuators and for the kinesthetic stimuli, they designed a lightweight exoskeleton, which provides 5 pounds of force per finger. Also, they use magnetic motion tracking with 6 DOF per finger. They power the glove with a control console, that makes zero noise. However, the whole design is heavy and the precision at small objects is off.

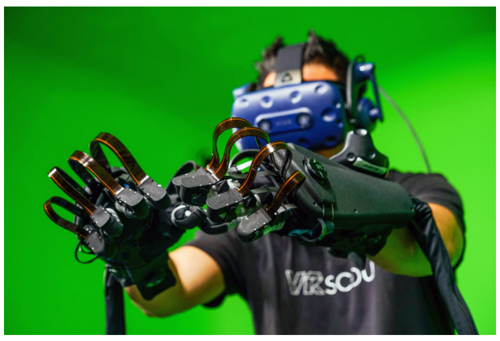


Figure 11: Haptx Glove

One of the few haptic devices made for medical training is by **FundamentalVr** [8], where in order to offer tactile sensations, they provide a grounded device. The device is the **Geomagic Touch**, which has 6 DOF positional sensing, 3 DOF force feedback, and uses a stylus to interact with the virtual environment. One of the main advantages of this design is that offers real-time feedback and can be useful for simulating the injection of a syringe. However, it provides small resistance to the hand, thus making the simulation less realistic.

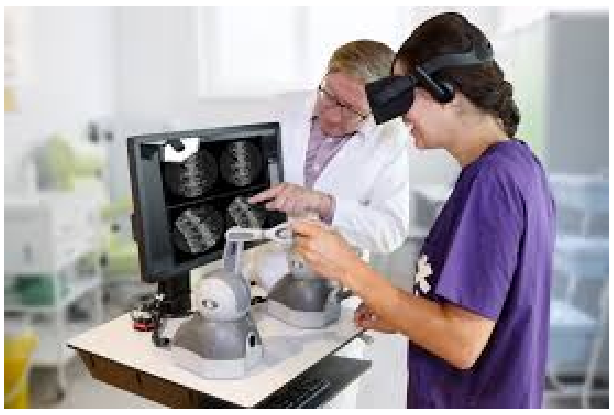


Figure 12: Fundamental VR

Another haptic device made for medical training is by **LinkDyn Robotics** [9], where they offer a grounded robotic arm, named **Optimo Arm**. The electromagnetic actuators come with a gear transmission, that amplifies force, which disables the precise force control. On top of that, it increases the inertia of the device, making it difficult to be back-driven and results in dangerous forces. To avoid this drawback, they developed a series of elastic actuator configurations. Specifically, the geared motor makes use of a load sensor at the actuator output. In the actuator, a spring replaces the load cell and delivers a force comparable to the human arm.

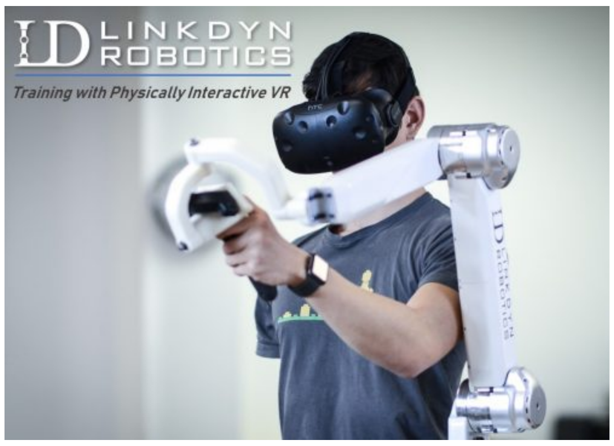


Figure 13: LinkDyn Robotics Implementation

Last but not least, an important problem is that during virtual reality experiences, enhanced haptic feedback incongruent with other sensory cues can reduce realism, producing an **uncanny valley of haptics** [10]. In this work, they conducted 3 experiments with two controllers(linked) to probe the influence of haptic stimuli.

The first was a passive experiment(user not moving arms) and the aim was to elicit the uncanny valley of haptics. During the simulation, the participants placed a virtual stick(which they held with the controllers) in a marked area—that takes the form of a cloud. When the stick enters this area, the participants receive haptic feedback.

The second was a dynamic experiment and is aimed at exploring if the active motion could help avoid the uncanny valley. Inside VR, the users held the same stick, however in this case they received the haptic experience only when they moved the stick up and down repetitively.

The third was a causal experiment and they explored the power of top-down prediction to reduce the uncanny valley. Inside this simulation, the participants placed a stick in the marked area like in the passive experiment. Similarly, they received the haptic feedback, but in this experiment, the marked area was moving, giving the impression of randomness.

In the three experiments, they rendered four different haptic conditions. The conditions were no haptic feedback, generic haptic feedback, spatialized feedback(different vibration amplitudes), and visual + spatialized feedback. Participants in each experiment underwent all these conditions and they completed a questionnaire.

From the results of the questionnaire, the researchers came to the conclusions, that to avoid the uncanny valley of haptics, the feedback should be compatible with the kind of stimuli, all feedback cues should be aligned(visual, auditory, haptic) and providing a cause can explain why the feedback cues occur(ex. press a button to cause something).

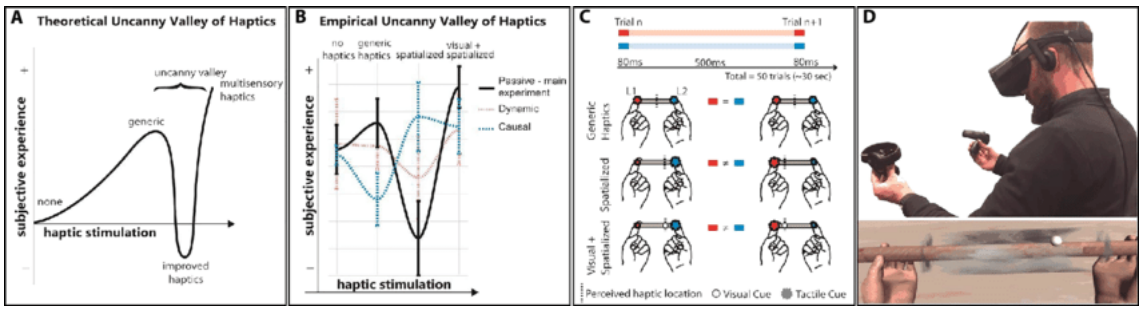


Figure 14: The uncanny valley of haptics

2.5 Previous Haptic Glove Implementations

2.5.1 A Haptic Glove Prototype for Tactile Feedback in 3D Interactive Applications

This thesis on haptics was conducted at the School of Electrical and Computer Engineering, Technical University of Crete(Busho, 2018) [11] and implemented a haptic feedback interface with the ability to provide vibration in 3D interactive applications.

The haptic interface Busho implemented, uses 10 ERM Coin Vibration Motors to provide tactile feedback. Five of these motors were placed at the tip of the finger and the other five at the bottom of the finger. Furthermore, these motors were programmed by a microcontroller(Arduino UNO) to have 3 different levels of vibration, to offer light touch, average touch, and heavy touch. Moreover, Busho designed a 3D game in Unity and when the user interacts with virtual objects(the information is sent with the Leap Motion to the microcontroller), only the vibration motors that collide with the object will vibrate.

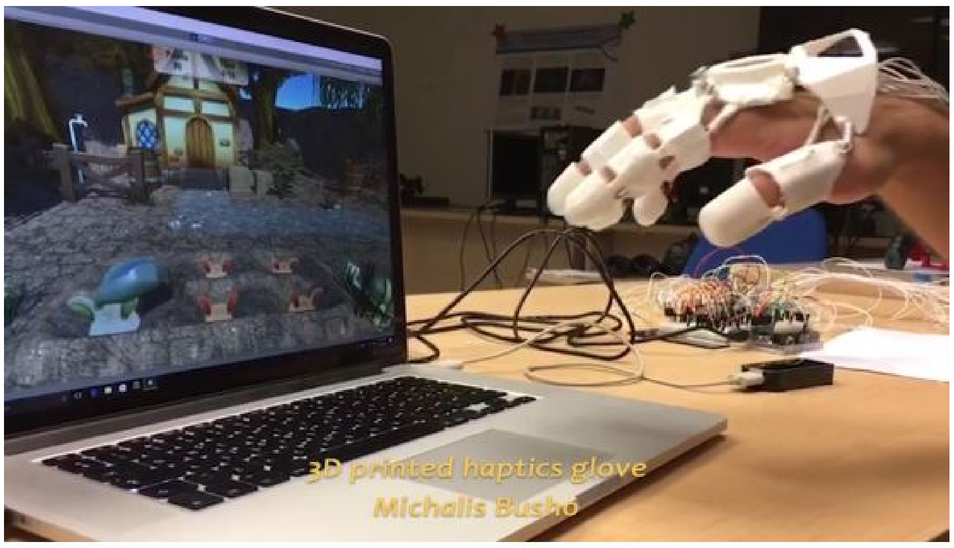


Figure 15: A Haptic Glove Prototype for Tactile Feedback in 3D Interactive Applications

2.5.2 Wireless Embedded System on a Glove for Hand Motion Capture and Tactile Feedback in 3D Environments

This thesis on haptics was also conducted at the School of Electrical and Computer Engineering, Technical University of Crete(Effraimidis, 2019) [12] and implemented a wireless haptic feedback interface to provide vibration in 3D interactive applications and a handmade tracking method for the hands.

In his thesis, Effraimidis used 5 ERM Coin Vibration Motor, for the tactile feedback, and he placed them on the fingertips. Moreover, he placed 5 flex sensors to track the finger rotation and a gyroscope-accelerometer for the pitch and roll rotation of the hand. All these devices are programmed by the microcontroller(Raspberry Pi). Also, Effraimidis created a device to detect the hand in the space, by combining a USB Camera with 3 IR LEDs.

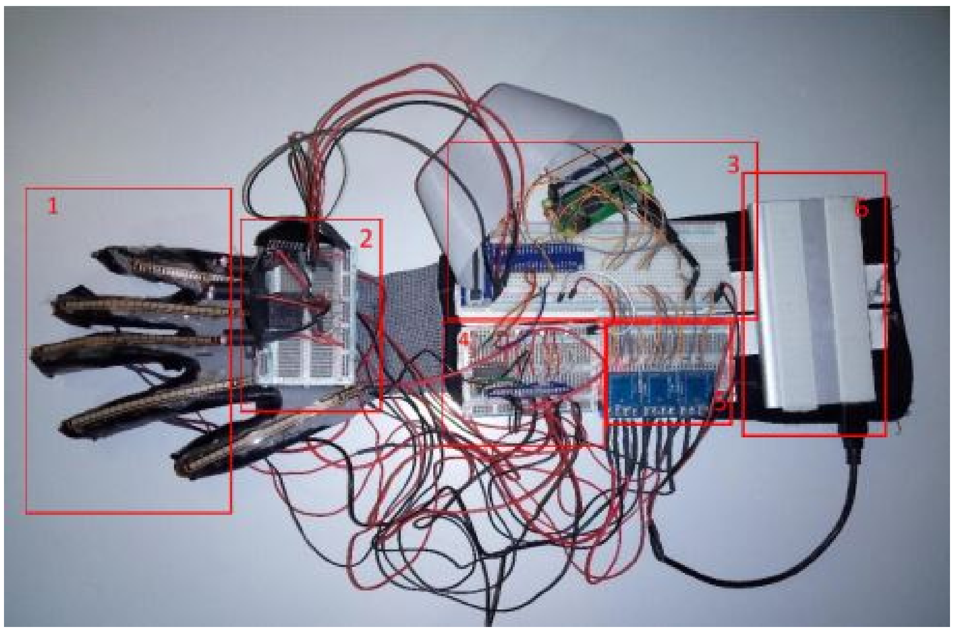


Figure 16: Wireless Embedded System on a Glove for Hand Motion Capture and Tactile Feedback in 3D Environments

2.5.3 Similarities and Differences with the previous implementations

The haptic glove that is presented in this thesis has used some basic ideas, from both of the above theses, but the main difference is that it provides kinesthetic feedback to the fingers.

Specifically, all three haptic devices use ERM coin vibration motors, that interact individually or together, depending on the contact, with the virtual environment. Also, the environment in every implementation was created in Unity Engine and the programming of the devices was made by microcontrollers.

However, this thesis used Arduino Mega 2560, instead of Arduino UNO (Busho) or Raspberry Pi Zero W(Effraimidis). Also, an important difference is that this glove has 15 vibration motors, which are driven with a circuit, that contains an N-MOSFET transistor(2N700), a Schottky diode to protect against voltage spikes, a resistor to keep the MOSFET fully off, and an EMI suppression capacitor. While Busho's has 10 motors, that are controlled by an NPN Darlington pair transistor, and Effraimidis implemented 5 motors, that are driven with a Dual Motor Driver. Furthermore, Effraimidis used a noncommercial solution to track the hand movement, while in this thesis and Busho's a commercial solution is being used(Leap Motion). Additionally, one difference is that in this thesis a printed circuit board was created to replace the breadboard, thus making the whole implementation less bulky and less heavy.

Finally, the major difference with the previous implementations, is that this thesis' haptic interface offers kinesthetic feedback. This type of feedback is achieved with the usage of 5 servo motors(one for every finger), that are driven with a 16-Channel 12-bit PWM Driver(PCA9685). Therefore, if the user grabs an object in the virtual environment, the collision info will be sent to the servos and they will lock a 3D printed gear. Then, the user will not be able to close his hand anymore, simulating the feeling of grabbing an object.

3 Glove Design and Implementation

In order to accomplish this thesis, the use of different programs was necessary. To design the haptic interface, solid modeling CAD software was used to design the 3D-printed exoskeleton. Then, electronic design automation(EDA) software was used to create the printed circuit board.

3.1 Hardware Assembly

The haptic feedback device is presented in figures 2,3 and its components are explained below:

- In figure 17, [1] is the exoskeleton, which was 3D printed, and the 5 servo motors which are connected with PCA9685.
- In figure 17, [2] is the PCA9685 and the printed circuit board, which consists of 15 n-type MOSFETs, 100pF capacitors, 50KOhm resistors, and Schottky diodes. The PCB is connected with the pair of cables of every vibration motor, the PWM pins, the 5V, and GND from Arduino. Furthermore, the PCA9685 is connected with the 5 servo motors. In order to power these motors, an external power of 5V and 2.5A was used.
- In figure 17, [3] is the Arduino Mega 2560 Rev3, which is powered with 5V from the USB port.Also, it powers and sends information to the PCA9685 board through the 5V, GND, SDA, SCL pins.
- In figure 18, the Coin Vibration Motors are presented, which were attached on the bottom side of the glove with glue. Every motor has two cables, which are connected to the printed circuit board.

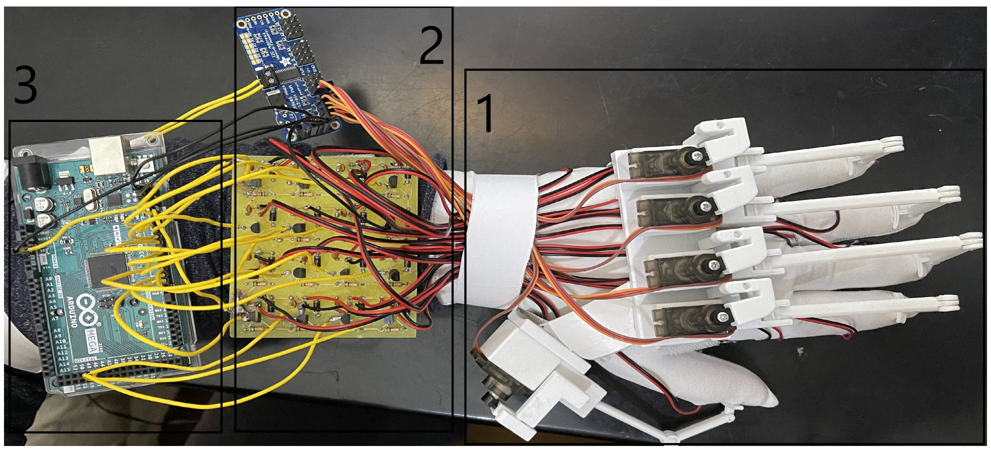


Figure 17: The top view of the haptic device

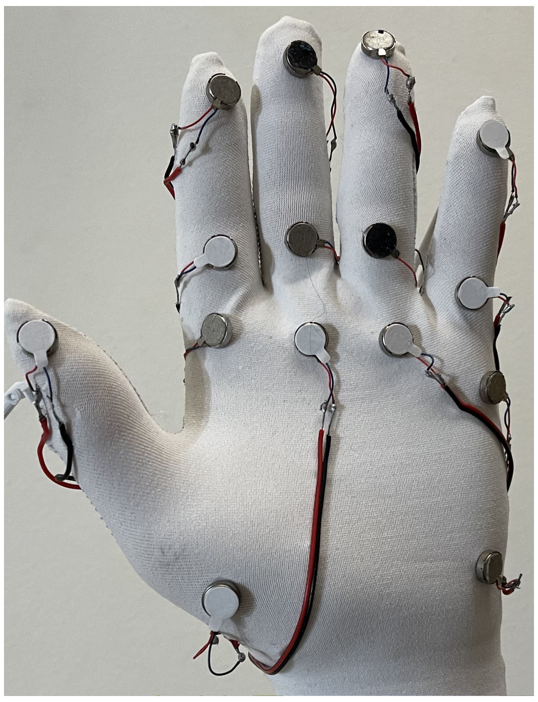


Figure 18: Bottom view of the haptic device

3.2 Design of the 3D Parts

The software, which was used in this thesis to create the 3D parts for the exoskeleton, was SolidWorks. The program has two panels. In the right panel, the user can create or modify the part. In the left panel of the window, the program contains the FeatureManager Design Tree(shows all the parts of the design), the PropertyManager(shows all the properties in every part the user selects), and the ConfigurationManager(shows all the configurations of the parts). The top side of the window contains a toolbar, that is called CommandManager.

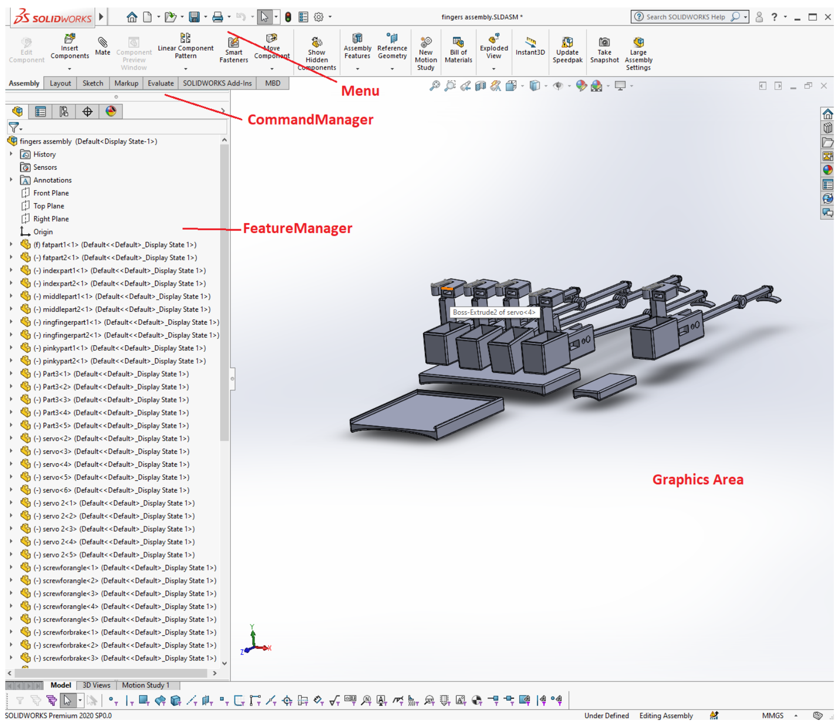


Figure 19: SolidWorks Window

The exoskeleton's design consists of 16 different parts and each part was designed from scratch. Then, all parts were combined and created the 3D assembly, which is the whole exoskeleton. This assembly could be divided into the following two sections:

The first section is the finger part of the exoskeleton, which is consisted of these three parts(the following process is repeated five times, one for every finger):

- **The bottom half of the finger**

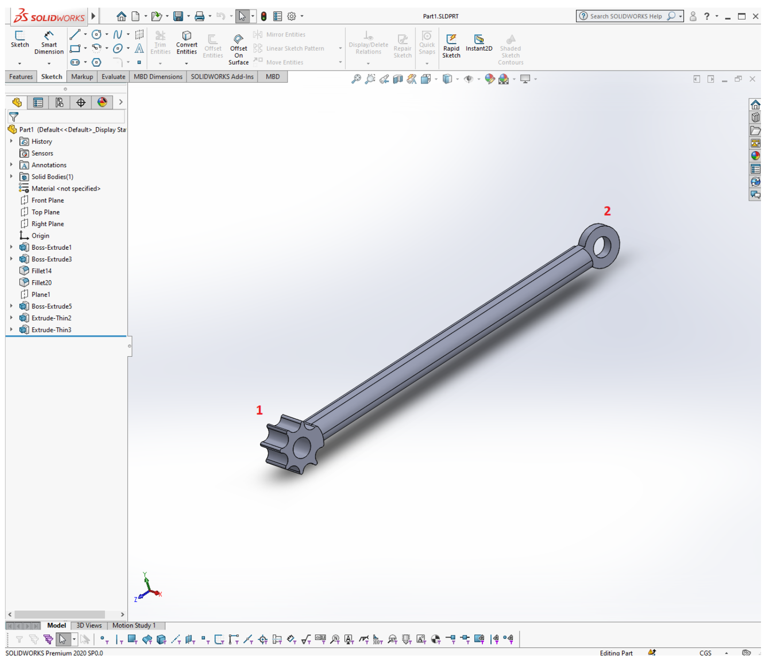


Figure 20: Bottom half of the finger 3D part

  This part is placed at the base of the finger and simulates the movement of the proximal phalanx bone of the hand. The left edge of the part will be connected with the braking mechanism, while the right edge will be connected with the part that simulates the medial and distal phalanges. To design this part, the first step was to create a 2D sketch(Sketch from CommandManager → Sketch) in a rectangular shape. Then this sketch is extruded(Features from CommandManager → Extruded Boss/Base), thus creating the middle section of this part. The next step is to fillet it(Features → Fillet) in the corners to make it round. Then, on the right edge, two circles are sketched(one is inside the other) and when they are extruded, a disc is created with a hole in the middle with the radius of the smaller circle. Similarly, two circles are sketched and extruded on the left edge, but this time at the edge of the bigger circle, seven semicircles are sketched to create the shape of a gear. The length of this part varies, depending on the finger, because every finger has a different length.

- **The top half of the finger**

  This part is placed at the top half of the finger and simulates the movement of the medial and distal phalanx bones of the hand. The left edge of the part will be connected with the part that simulates the proximal phalanx, while the right will be connected with the fingertip part. The first step to designing this part was to create a rectangular sketch. The second step was to extrude this sketch at a certain length depending on the finger and to fillet it to smooth the edges. After that, on the left edge, two circles are created on each side. Then, these circles are extruded until

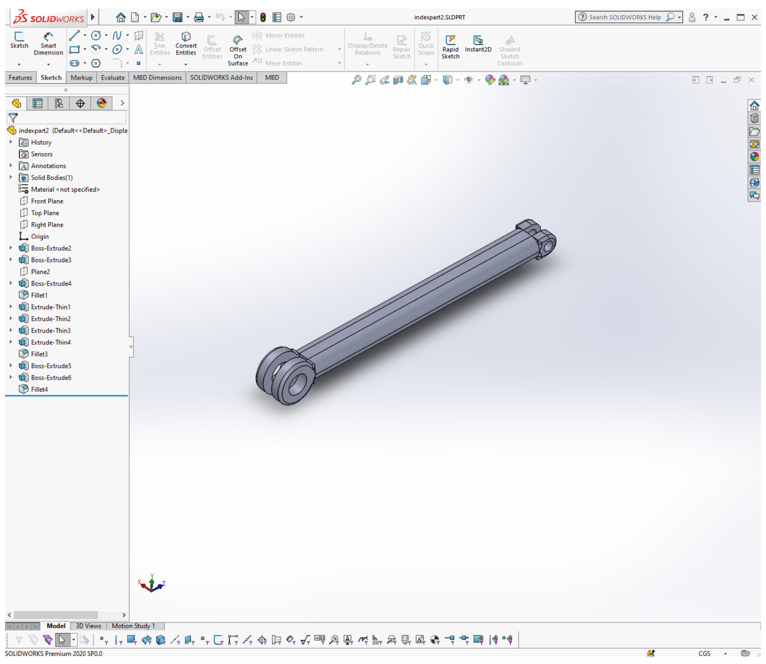


Figure 21: Top half of the finger 3D part

the gap between them has the same width as them. Finally, the same process is repeated on the right edge, but this time the circles are smaller.

- **The fingertip**

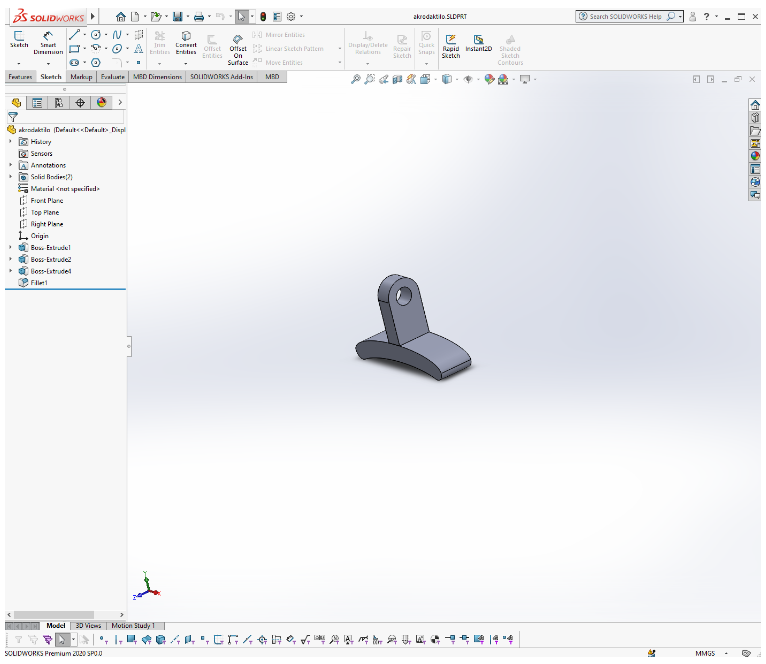


Figure 22: The fingertip 3D part

This part is placed at the fingertip of the user. The upper section of this part will be connected with the part that simulates the medial and distal phalanges, while the lower one will be connected with the finger. The first step to create the upper section was to draw a rectangular sketch and extrude it. Then, two circles(one inside the other) were created at the top of the extruded sketch(see figure 22). Finally, to design the lower section, two half ellipses were sketched, extruded, and filleted.

The second section of the exoskeleton is the servo motor part, which is consisted of the following three parts(this design of this section of the exoskeleton is repeated five times, one for every finger):

- **The base for the servo**

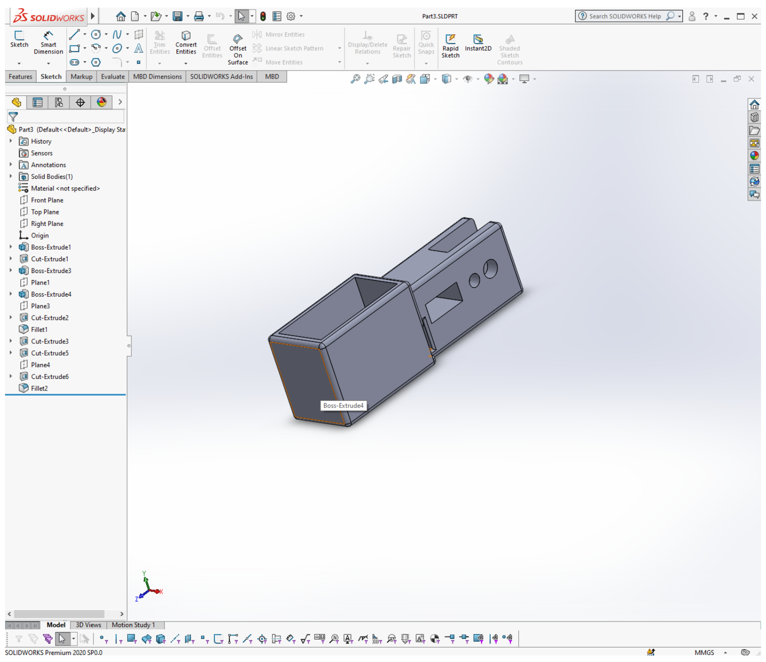


Figure 23: Base of the servo 3D part

This part is placed at the back of the hand near the knuckles and it consists of two sections, the rear, and the front. The rear section is, where the servo motors are placed and in order to design it, a rectangular sketch was created and extruded. Then, a new rectangular sketch with smaller dimensions was created on top of the bigger sketch and with the extruded cut tool(Features → Extruded Cut), a hole was made that was big enough to fit the servo. As for the front section, it consists of a rectangular sketch that was created and extruded at the front area of the rear section. Then, a hole with a rectangular shape was made in this sketch. Moreover, a sketch with the shape of "U" was created and extruded. However, the space between the two sides had to be wide enough to fit the bottom half of the finger(3D part) that was mentioned before. Finally, two round holes were made on each side of the "U" in order for the parts the be connected.

- **Component of the braking mechanism**

  This part is a component of the braking mechanism of the finger movement and it is placed at the left side of the servo's base. In order to create this design, a rectangular sketch had to be made. Then, a smaller rectangular sketch was created at the top right corner of the initial sketch and was extruded cut to create an "L" shape. Additionally, three new sketches were made. The first one was a circle, that was placed at the top side of the component and then was extruded to be connected with the servo horn 3D part. The second one was a rectangular sketch that was extruded to be connected with the hole in the servo's base. Last but not least, another circle was sketched and placed next to the rectangle and is used to block the rotation of the gear in the bottom finger part when needed.

- **The servo horn connector**

  This part is placed at the servo horn. To create this part, a rectangular sketch was made and extruded with a small rectangular hole inside it to insert the servo horn. Then, a second rectangle, vertical to the first one, was created and extruded. Finally, one last rectangle, vertical to the

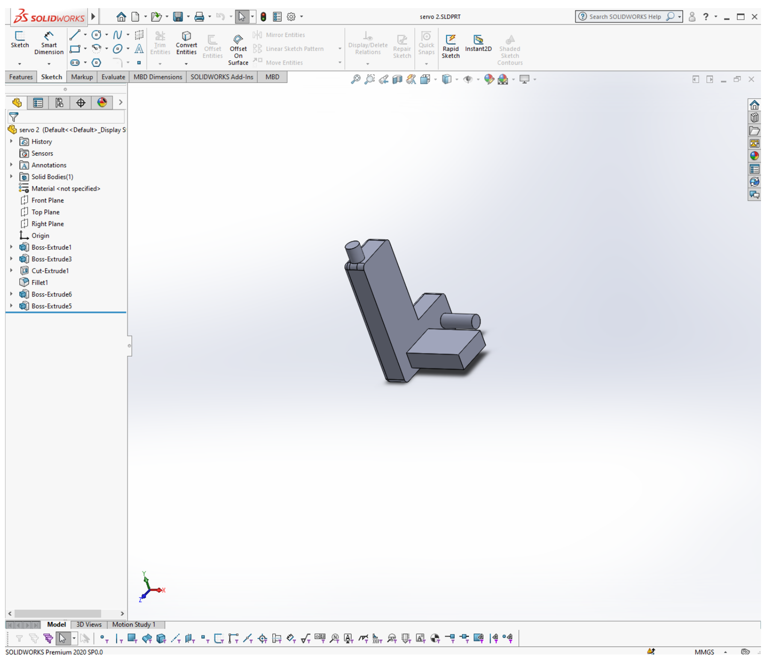


Figure 24: A component of the braking mechanism 3D part

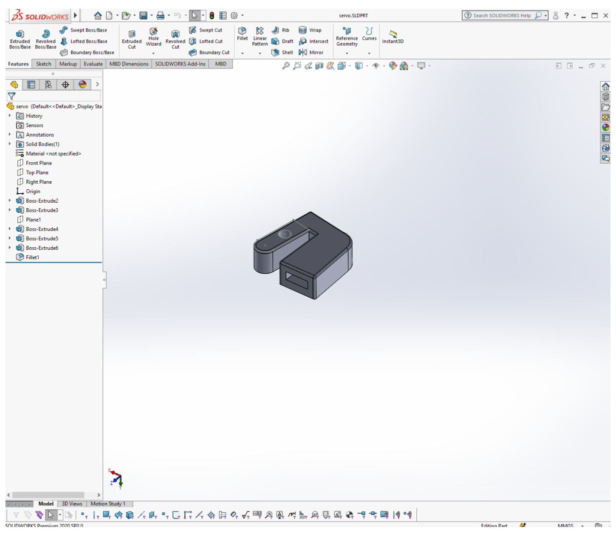


Figure 25: The servo horn connector 3D part

second one and parallel to the first one, was sketched and extruded with a circular hole in the middle to connect it with the part of the braking mechanism.

When the process of designing the CAD model of the exoskeleton finished, it had to be 3D printed. There are many materials, that are being used in the 3D printing process, each one offering some pros and cons [13]. However, the three main materials that were considered in this thesis were ASA, PLA, and ABS.

ABS(Acrylonitrile Butadiene Styrene) is a thermoplastic polymer and a very durable general use material. Specifically, the polybutadiene monomer makes the material flexible and strong enough to withstand impacts, while the styrene makes it harder and acrylonitrile holds everything together. Finally, this material dissolves in solvents like acetone, which allows easy smoothing.

ASA(Acrylonitrile Styrene Acrylate) is another 3D printable plastic with properties similar to ABS. The difference with ABS is that the Acrylate is more UV resistant than Butadiene, making ASA less brittle to sunlight. ASA is also known for high impact resistance and is commonly used in outdoor applications. However, an important problem, is that it is likely to wrap under stress during 3D printing and it is quite expensive.

PLA(PolyLactic Acid) is one of the most environmentally friendly materials because it is sourced from natural crops such as sugarcanes and corns. On top of that, PLA is renewable and most importantly biodegradable. Last but not least, it is harder and stiffer than ABS, thus making it easier to break cause it is not bendable.

Considering all the above and the limited budget, the choice was PLA, because of its stiffness and general cost.

3.3 Components used for the Haptic Glove

In this section, the components of this thesis haptic interface were analyzed. The main components are the Arduino Mega 2560 Rev3, the vibration motors, the servo motors, and the PCA9685-Servo Driver.

3.3.1 Arduino Mega 2560 Rev3

To control all the components, the use of a microcontroller was necessary. A microcontroller is a small computer, that contains a processor, a memory, and some programmable input/output peripherals. The hardware selected for this thesis is the Arduino Mega 2560 Rev3. The Arduino Mega is the most advanced board of the Arduino technology and is suggested for complicated designs that demand larger memory and more inputs/outputs. The Mega 2560 model is the most popular and is compatible with many sensors and shields. Also, it is based on the same architecture as the Uno board, but their main differences are the memory capacity and the number of inputs/outputs for connection with outside devices

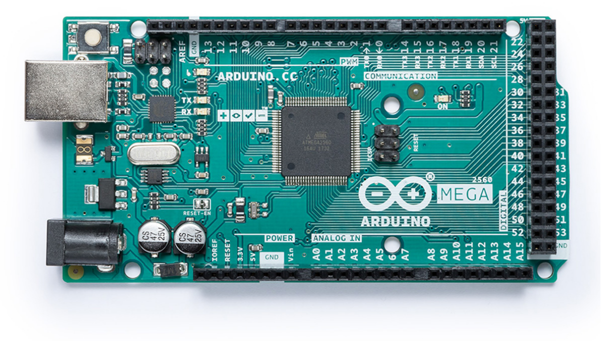


Figure 26: The Arduino Mega 2560 Rev3

Furthermore, Arduino Mega 2560 Rev3 is based on Atmel's microcontroller ATmega2560 and it is a board that contains whatever is necessary to be programmed and operated by simply connecting a USB cable into the computer or by an external power supply. Specifically, the board is equipped with 54 digital inputs/outputs (15 of them can be used as PWM outputs), 16 analog inputs, 4 serial ports, a USB port(type B), a power supply port, a 16 MHz crystal oscillator, an ICSP header, and a reset button.

Despite the circuit board, Arduino also offers an Integrated Development Tool(IDE). The Arduino IDE is a cross-platform application that is used to write and upload code to the board using a USB cable, therefore an external programmer is not required. Also, it supports a simplified version of C++, which makes it easier to use.

Arduino uses the name sketch for a program and it contains the code. Every sketch consists of two special functions: setup() and loop(). The setup() is called once when the sketch starts and it is used to initialize variables, inputs, outputs, and libraries. The loop() function is called repeatedly and it controls the board until the board is powered off.

Moreover, some other important functions used in this thesis, were pinMode(), delay(), analogWrite(), etc. With the pinMode() function, the user configures a pin to work either as an input or an output. The syntax for this function is pinMode(pin, mode) with the pin being the number of the Arduino pin and mode being either INPUT or OUTPUT. This function is often called in the setup(). The analogWrite() is used to write an analog value, which could be a PWM signal, to a pin. Specifically, the analogwrite() function takes as parameters the name of the pin and an integer value(0-255) that sets the duty cycle(0 fully off and 255 fully on). As for the delay() function, the user can pause the program for a specific amount of time. The syntax is delay(ms), with ms being the number of milliseconds, the program is paused.

3.3.2 Coin Vibration Motors

One of the most important parts of this thesis was finding a way to provide tactile stimuli. This was achieved through 15 coin vibration motors. The options for the vibration were either Linear Resonant Actuators or Eccentric Rotating Mass motors, each having its advantages and disadvantages [14].

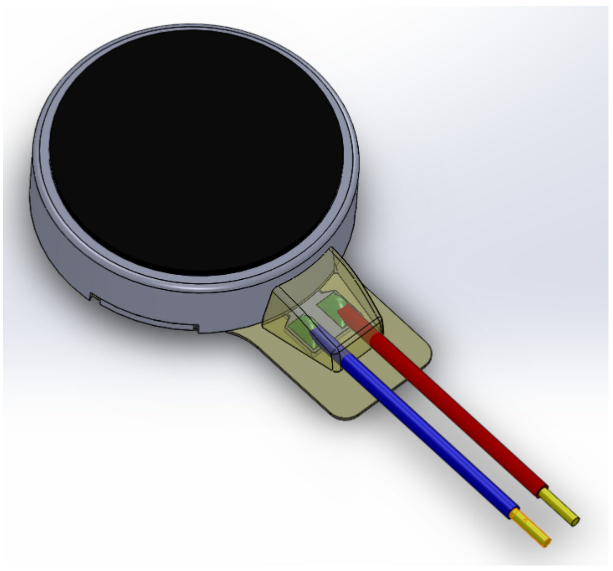


Figure 27: The coin vibration motors

One of the LRA's advantages is that it has 50% less power consumption than ERM motors. Also, the fact that LRA's amplitude is independent of frequency, means that they can provide more complex waveforms from ERM motors. Besides, LRA makes less noise and they have a longer life span.

However, LRAs work only with Alternative Current(AC), thus the need for a complex driving circuit is mandatory. Also, LRAs are more expensive and less available in the market.

In conclusion, despite the advantages, LRAs offer, the fact that the budget was limited and the need for AC signal to the motors the choice of ERM motors was the most reasonable.

The following table shows the specifications of the chosen ERMs.

Specification	Value
Voltage [V]	3
Frame Diameter [mm]	10
Body Length [mm]	3.4
Weight [g]	1.2
Voltage Range [V]	2.5 – 3.8
Rated Speed [rpm]	12000
Rated Current [mA]	75
Start Voltage [V]	2.3
Start Current [mA]	85
Terminal Resistance [Ohm]	75
Vibration Amplitude [G]	0.8

Table 1: The coin vibration motor specifications

3.3.3 WS-MG90S Micro Servos

Another important part of this thesis was finding a way to provide kinesthetic stimuli. This was achieved with 5 WaveShare MG90S micro servo motors.

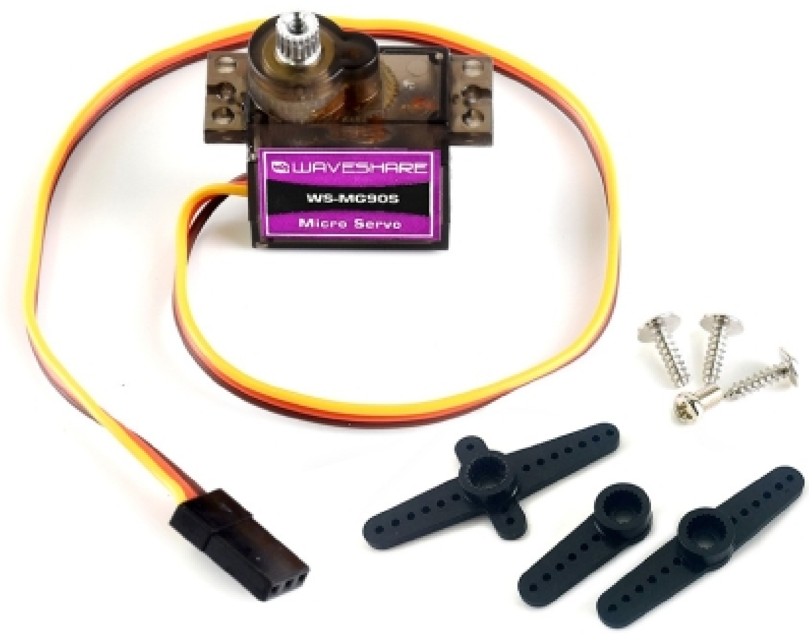
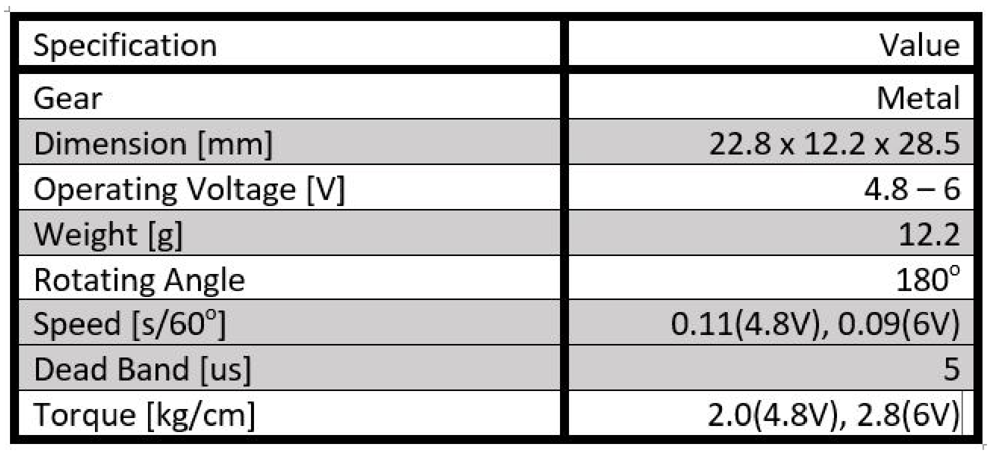


Figure 28: The MG90S servo motor

These motors were chosen because they are small and lightweight devices, capable to provide enough power to block the finger's movement.

The following table shows the specifications of the servo motors.



Specification	Value
Gear	Metal
Dimension [mm]	22.8 x 12.2 x 28.5
Operating Voltage [V]	4.8 – 6
Weight [g]	12.2
Rotating Angle	180°
Speed [s/60°]	0.11(4.8V), 0.09(6V)
Dead Band [us]	5
Torque [kg/cm]	2.0(4.8V), 2.8(6V)

Table 2: The MG90S servo motor specifications

3.3.4 PCA9685-Servo Driver

To drive the servo motors, Arduino offers the Servo library, but each servo will consume a pin and some Arduino processing power. This thesis proposed the solution of the PCA9685 16-Channel 12-bit PWM/Servo Driver for driving these motors. The PCA9685 will drive up to 16 servos over I2C with only 2 pins and with the on-board PWM controller it will happen simultaneously.

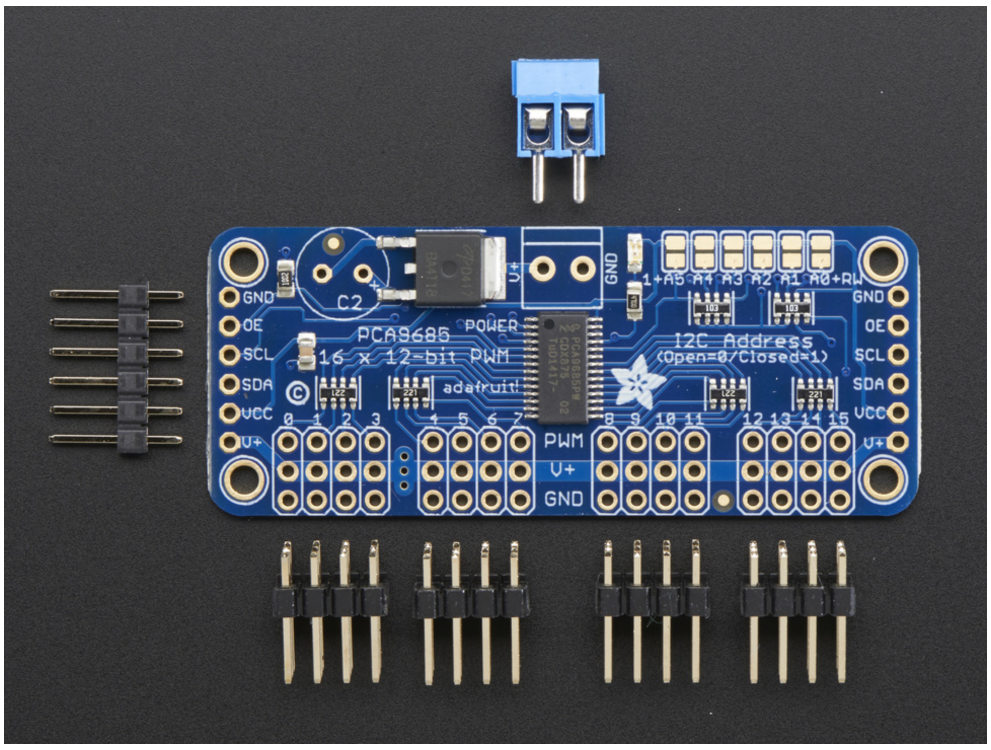


Figure 29: The PCA9685 16-Channel 12-bit PWM/Servo Driver

As shown in Figure 27, each side of the chip contains the control input pins. This happens because the chip gives the user the freedom to connect another PCA9685, for more than 16 servos, by chaining them side-by-side.
The control input pins are the following:

- **GND**, which is the power and signal ground pin.
- **OE**, which is the output enable and can be used to quickly disable all outputs(if this pin is high)

- **SCL**, which is the I2C clock pin and is connected with the microcontrollers I2C clock line.
- **SDA**, which is the I2C data pin and is connected with the microcontrollers I2C data line.
- **VCC**, which is the logic power pin.
- **V+**, which is the optional power pin that will supply the servos. It can be left disconnected because it is substituted from the blue terminal in the middle of the chip.

Moreover, there are 16 output ports at the bottom side of the chip. Each port has 3 pins: V+, GND, and the PWM output. Even though each PWM is independent, it must have the same PWM frequency as the others.

The next step after soldering the pin headers into the marked position was to hook up the Arduino with the PCA9685 in the following way:

- **GND → GND**
- **5V → VCC**
- **SDA → SDA**
- **SCL → SCL**

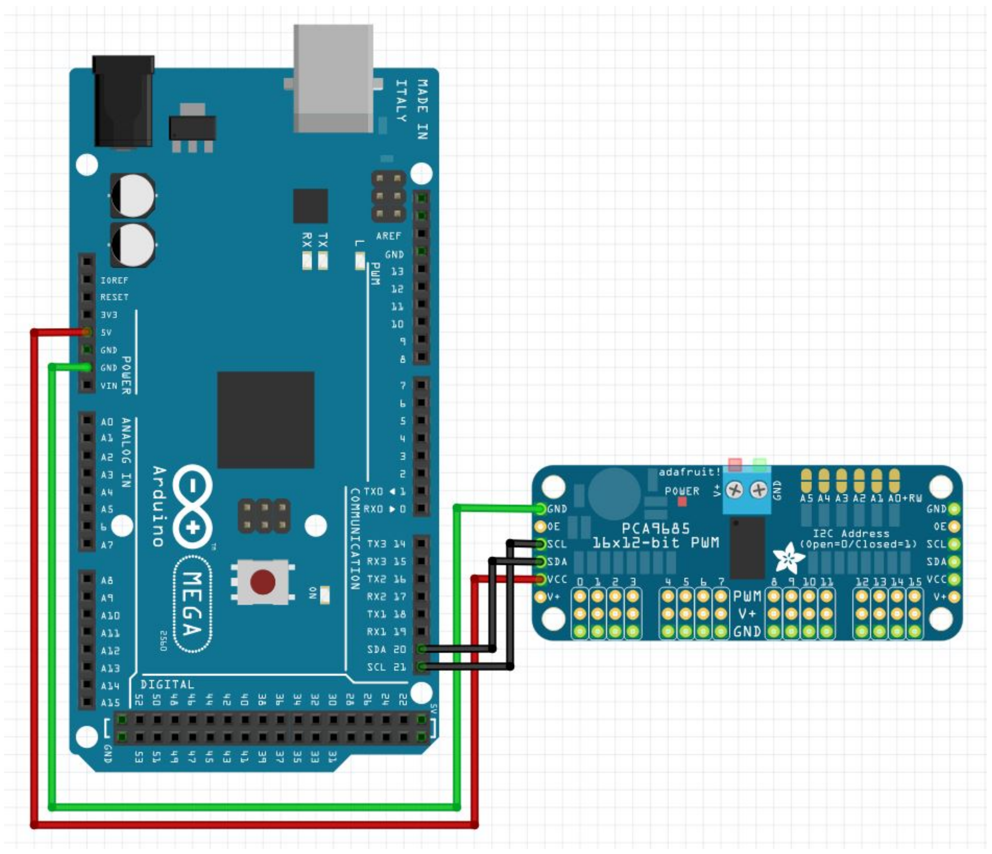


Figure 30: The wiring between Arduino and PCA9685 in Fritzing

The connection in figure 28 is only used to power the breakout board, but to power the servos, an external 5V connected on the blue terminal is necessary.

The servos are connected on the output ports at the bottom side of the board, with the brown wire going to GND, the red to V+, and the orange going to PWM. If the user wants to connect more than 16 servos, he should chain more PCA9685(up to 62), but he needs to assign each board with a unique address. This is accomplished by soldering to bridge the address jumpers on the upper right edge(board 0: 00000, Board 1: 000001, etc).

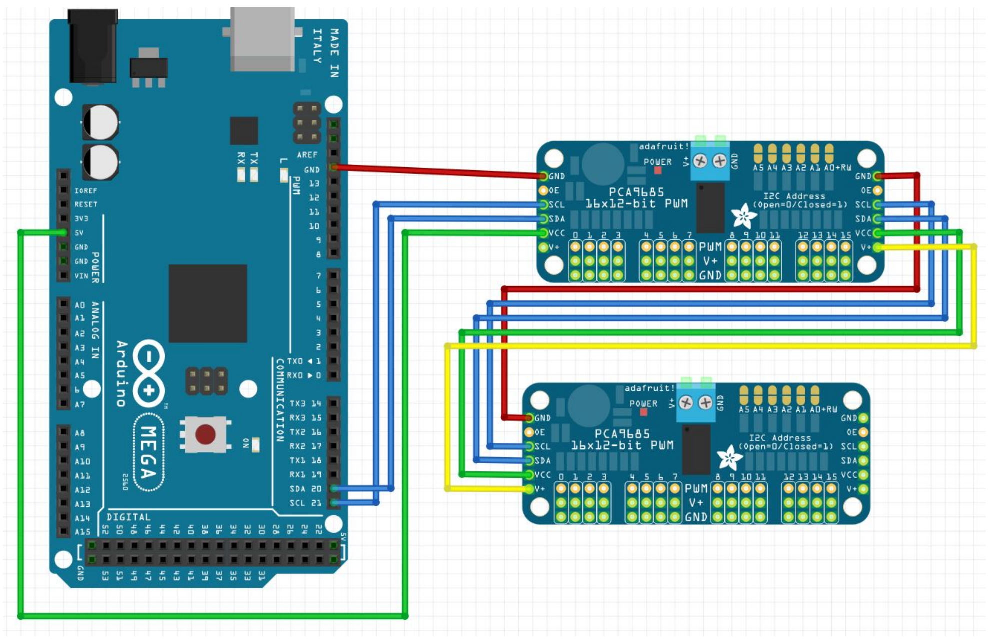


Figure 31: Chaining PCA9685 made in Fritzing

3.4 Motor Driver Circuit

The ERM motors used in this thesis have a start current draw of 85mA and an operating current draw of 75mA, which is greater than the output current from each pin in Arduino(up to 40mA). Thus a driver circuit is required to overpass this limitation and the major component in this driver circuit was a transistor. The transistor is a semiconductor device used for the switching(they can turn the motor on and off very quickly) or the amplification(they can amplify the low power from the Arduino to a higher one in the ERM motor) of electronic signals. There are two different types of transistors, the Bipolar Junction Transistor(BJT) and the Field Effect Transistor(FET).

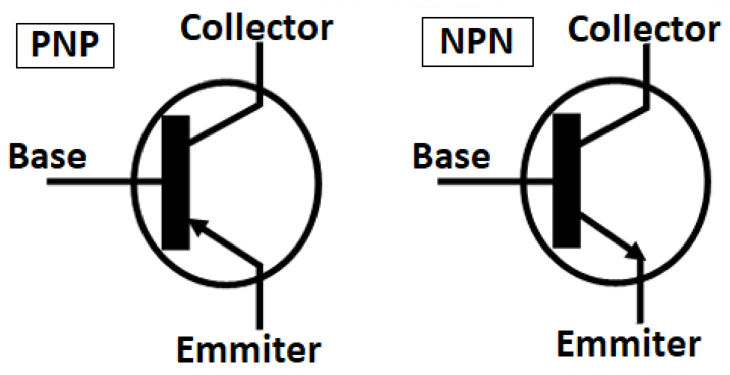


Figure 32: Bipolar Junction Transistor

Bipolar Junction Transistor [15] has three terminals, the Emitter, the Base, and the Collector. It has two types of configurations NPN and PNP, where NPN is made by placing a p-type material between two n-type materials and PNP by placing an n-type material between two p-type materials. Moreover, BJT is a current controlled device, thus when the biasing current flows to the Base, the controlled current goes from Collector to Emitter(NPN) or from Emitter to Collector(PNP).

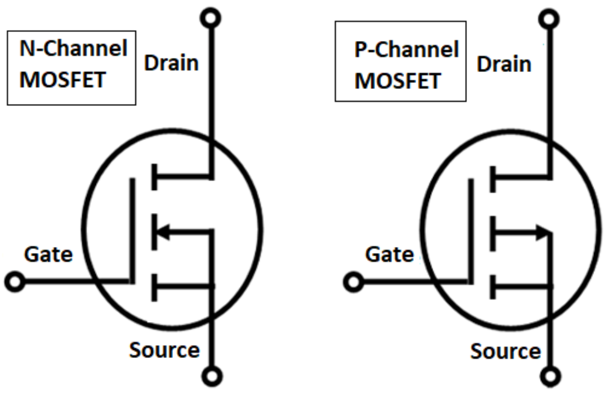


Figure 33: Metal Oxide Semiconductor Field Effect Transistor

Field Effect Transistor also has three terminals, the Source, the Gate, and the Drain. The two main types of FETs are JFET and MOSFET. MOSFETs are divided into two categories, the N-Channel MOSFET and the P-Channel MOSFET. N-MOSFET is made with a p-type body with two regions of n-type material(Source and Drain) adjacent to the Gate, while P-MOSFET is made with the n-type body with two regions of p-type material adjacent to the Gate. Additionally, they are voltage-controlled devices, thus when voltage is applied at the Gate it controls the current between Source and Drain(if the voltage is negative, the MOSFET is on depletion mode, but if the voltage is positive, the MOSFET is on enhancement mode).

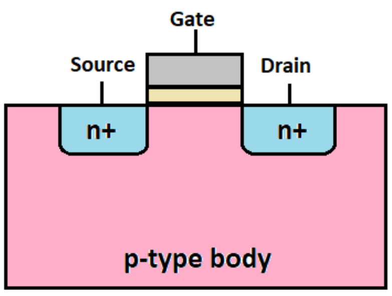


Figure 34: N-type MOSFET

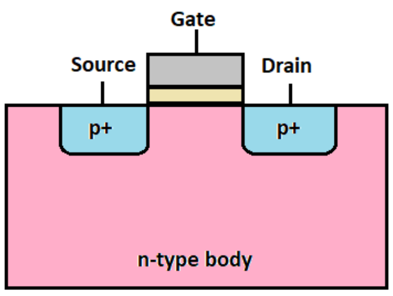


Figure 35: P-type MOSFET

In this thesis, the choice was between BJTs and MOSFETs. However, BJT transistors are less efficient and require heat-sinking when high currents pass through them, thus the type of transistor used was n-type MOSFET(in its saturated mode) and more specifically the 2N7000. An N-channel MOSFET was selected because it works better with 2V and higher(higher Vgs gate turn-on-voltage).

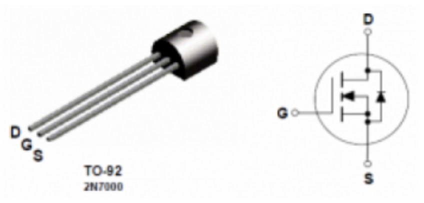


Figure 36: The 2N7000 N-Channel MOSFET

After the selection of the NMOS, the circuit was not complete and it required some extra components in order to work properly. More specifically:

- For the protection of the MOSFET against voltage spikes from coils, a Schottky diode was placed in parallel across the motor's terminals. A diode is a two-terminal electronic component that allows the current to pass only from anode to cathode. The diode that was used in this thesis was the 1N5817, which has a maximum DC blocking voltage of 20V and maximum average forward rectified current 1A.

Figure 37: The Schottky Diode 1N5817

- To keep the MOSFET fully off when there is no signal, a pull-down resistor was used. The resistor that was used in this thesis was 50k Ohm and was sufficient enough to reduce the quiescent current.
- For the reduction of the high-frequency electromagnetic noise generated by the motor, an EMI suppression capacitor was used. For this thesis, any capacitor from 10 to 100pF is suitable, because it is small enough not to interfere with the PWM signal but large enough to limit the voltage spikes. An important note is to place the capacitor as close to the motor's terminal as possible in order to work better.

The complete circuit that is used in this thesis to drive the ERM motors looks like this:

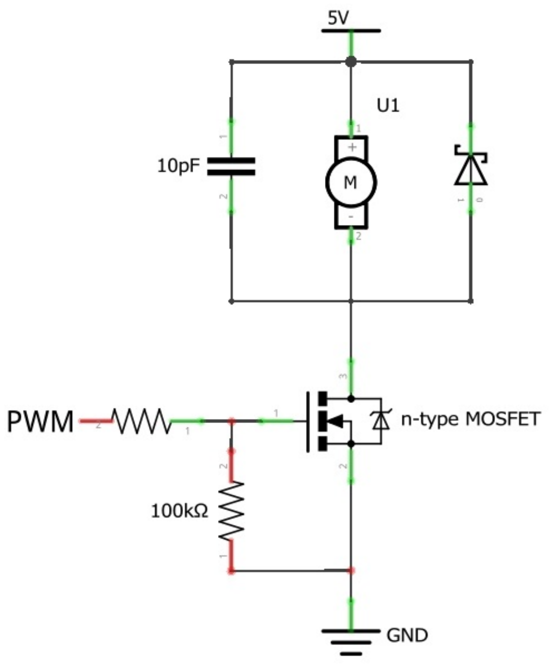


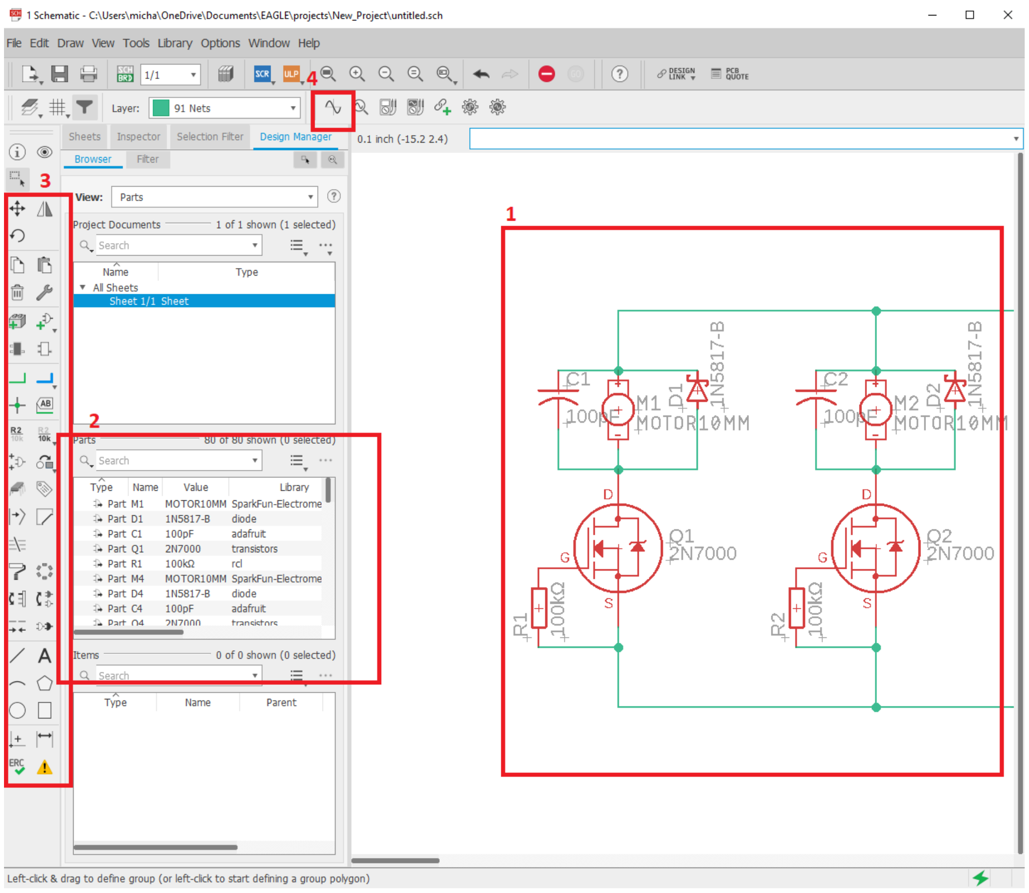
Figure 38: The Motor Driver Circuit

3.5 Design of the Printed Circuit Board

After completing the idea of the driver circuit, it had to be moved from the prototyping breadboard to a printed circuit board in order to reduce the size of the system.

The electronic design automation software used in this thesis is the Autodesk Eagle. Eagle belongs to the category of software tools, that are used for designing electronic systems such as printed circuit boards. Specifically, this program contains a schematic editor and a PCB layout editor.

In the schematic editor, the user can create a schematic diagram, by inserting components. These will be represented in the biggest window(see [1] in figure 39) and with the form of a list on the left side of the window(see [2] in figure 39). The user can insert these components with the tools he finds on the left side of the window(see [3] in figure 39) and then quickly validate their performance with SPICE simulation methods(see [4] in figure 39). Also, with a complete set of electronic rule checks, the user can validate his schematic design. The files from this editor are stored with the .SCH extension.


1 Schematic - C:\Users\micha\OneDrive\Documents\EAGLE\projects\New_Project\untitled.sch
File Edit Draw View Tools Library Options Window Help
Layer: 91 Nets
Sheets Inspector Selection Filter Design Manager
Browser Filter
0.1 inch (-15.2 2.4)
1
2
3
4
C1
100pF
M1
MOTOR10MM
D1
1N5817-B
Q1
2N7000
R1
100kΩ
C2
100pF
M2
MOTOR10MM
D2
1N5817-B
Q2
2N7000
R2
100kΩ
D
G
S
Left-click & drag to define group (or left-click to start defining a group polygon)

View: Parts

Project Documents — 1 of 1 shown (1 selected)

Name	Type
All Sheets	
Sheet 1/1	Sheet

Parts — 80 of 80 shown (0 selected)

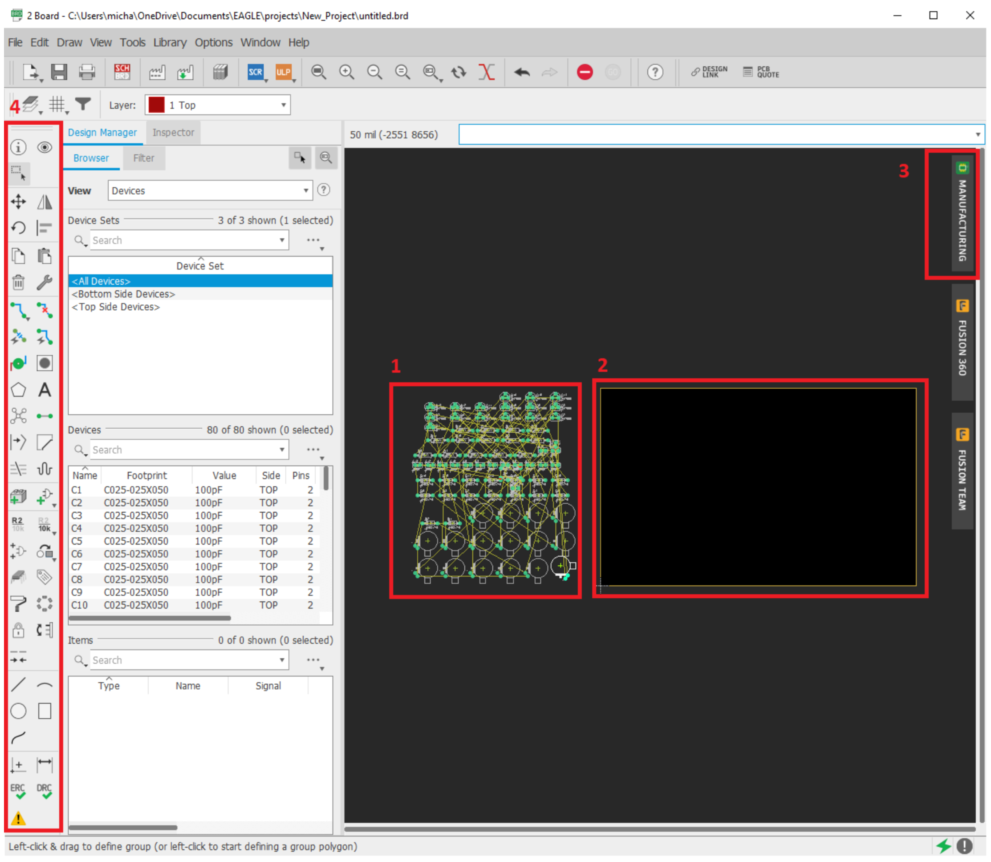
Type	Name	Value	Library
Part	M1	MOTOR10MM	SparkFun-Electrome
Part	D1	1N5817-B	diode
Part	C1	100pF	adafruit
Part	Q1	2N7000	transistors
Part	R1	100kΩ	rcl
Part	M4	MOTOR10MM	SparkFun-Electrome
Part	D4	1N5817-B	diode
Part	C4	100pF	adafruit
Part	Q4	2N7000	transistors

Items — 0 of 0 shown (0 selected)

Type	Name	Parent

Figure 39: Schematic Editor

In the PCB layout editor, the user can place the components(see [1] in figure 40), that he added in the schematic editor, in a specific area(see [2] in figure 40) in any way he desires. Then, he can use the tools on the left side of the window(see [4] in figure 40) to route the wires, move the parts in space, add new holes, etc. Moreover, he can preview how the PCB will look like by pressing the manufacturing button in the top right corner of the window(see [3] in figure 40). Finally, the user can control his design flow and avoid unexpected problems with the customizable PCB design rules and constraints. The files from this editor are stored with the .BRD extension.


2 Board - C:\Users\micha\OneDrive\Documents\EAGLE\projects\New_Project\untitled.brd

Layer: 1 Top

50 mil (-2551 8656)

View: Devices

Device Sets — 3 of 3 shown (1 selected)

Device Set
<All Devices>
<Bottom Side Devices>
<Top Side Devices>

Devices — 80 of 80 shown (0 selected)

Name	Footprint	Value	Side	Pins
C1	C025-025X050	100pF	TOP	2
C2	C025-025X050	100pF	TOP	2
C3	C025-025X050	100pF	TOP	2
C4	C025-025X050	100pF	TOP	2
C5	C025-025X050	100pF	TOP	2
C6	C025-025X050	100pF	TOP	2
C7	C025-025X050	100pF	TOP	2
C8	C025-025X050	100pF	TOP	2
C9	C025-025X050	100pF	TOP	2
C10	C025-025X050	100pF	TOP	2

Items — 0 of 0 shown (0 selected)

Type	Name	Signal

Left-click & drag to define group (or left-click to start defining a group polygon)

Figure 40: PCB Layout Editor

The first step to create the PCB was to pass the circuit design(see figure 38) into the Schematic Editor and duplicate it 15 times. Then, all the GND and 5V were connected to a single GND line and a single 5V line. After pressing the generate/switch to board button, the PCB Layout Editor popped and the components, that were inserted through the Schematic Editor, were carefully placed in the black area. The result is shown in figures 41 and 42.

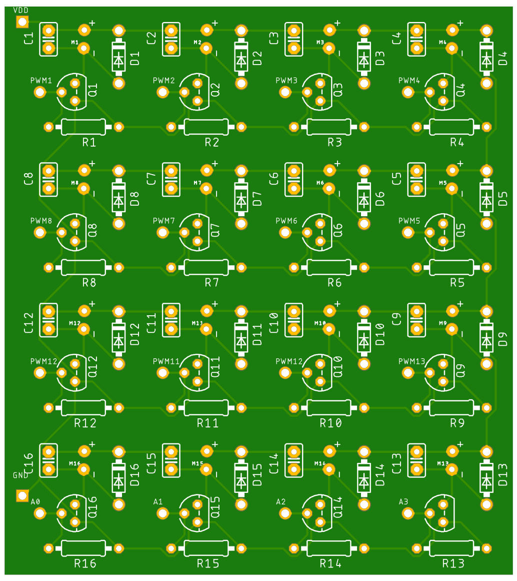


Figure 41: Preview of the PCB from Manufacturing Button

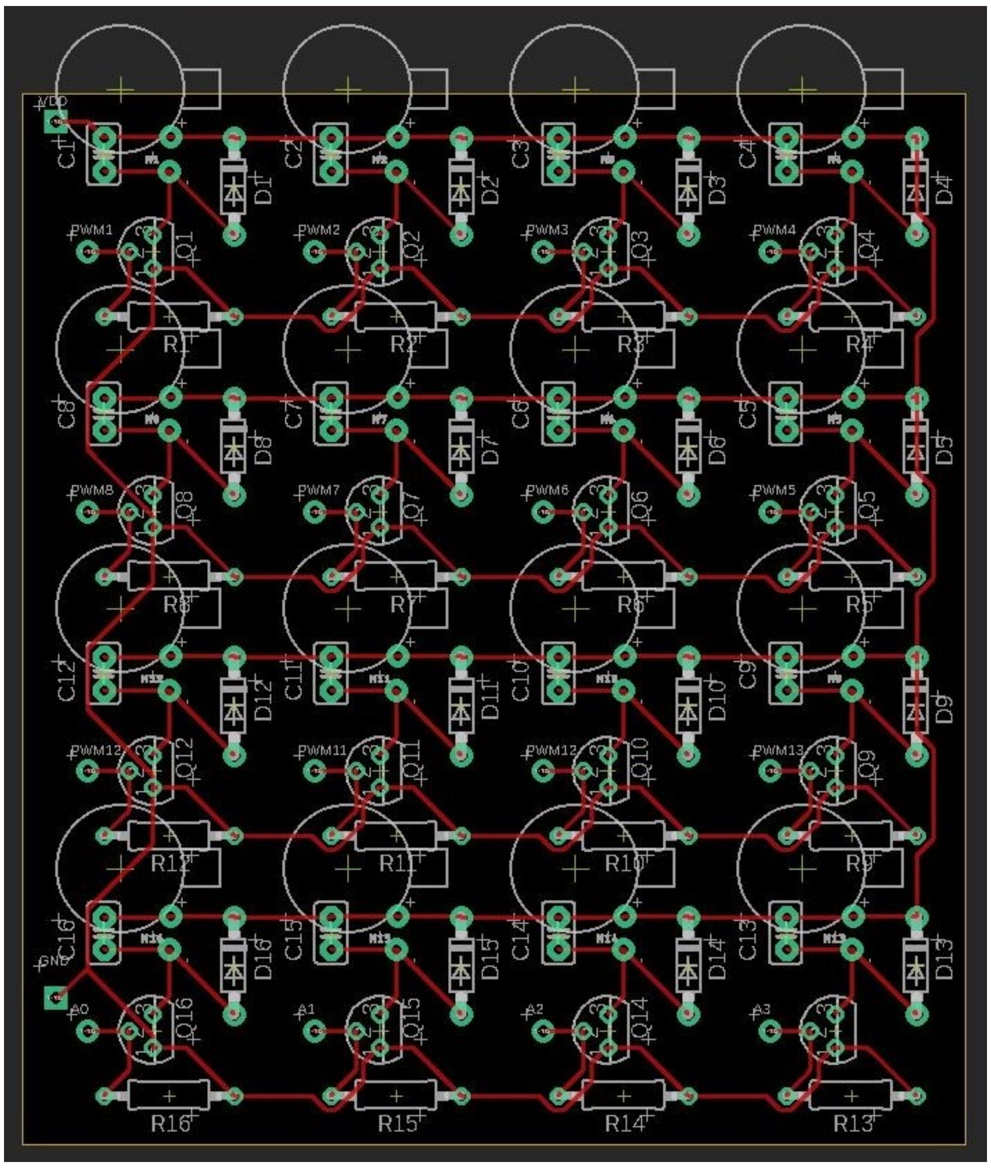


Figure 42: PCB Design and Wiring

3.6 Programming Arduino

After creating the 3D-printed exoskeleton for the kinesthetic feedback and the ERM's driver circuit for the tactile feedback, the next step was to program the Arduino. The Arduino was programmed to turn the servo motors in specific degrees and to enable or disable the coin vibration motors depending on the characters it received from Unity.

Before explaining the code, it is necessary to give further information on how the kinesthetic feedback is provided in this thesis. More specifically, if the Arduino receives a specific character, the servo turns (See 1 in figure 43), which moves the braking mechanism 3d part(see figure 20 and 2 in figure 43). Then, the extruded cylinder, that is on the bottom side of the braking mechanism 3d part, is plugged into the hole of the base of the servo 3d part and blocks the movement of the gear(see 3 in figure 43), thus the user is not able to move the finger.

To begin with, because the PCA9685 Servo Driver is controlled over I2C, it is important to install the Adafruit_PWMServo and the Wire libraries. Then, the next step was to configure the pins, that are responsible for the vibration motor's power, as outputs with the function pinMode() and initialize them as zero with the function analogWrite(). Furthermore, the servo motors are initialized with the function setPWM(). This function sets the start and the end of the high part of the PWM pulse on a specific channel.

In order to receive the data from Unity, this implementation used the func-

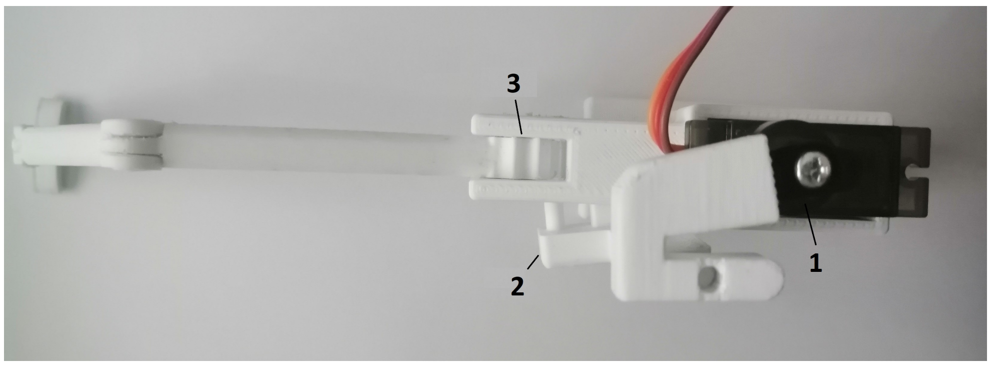
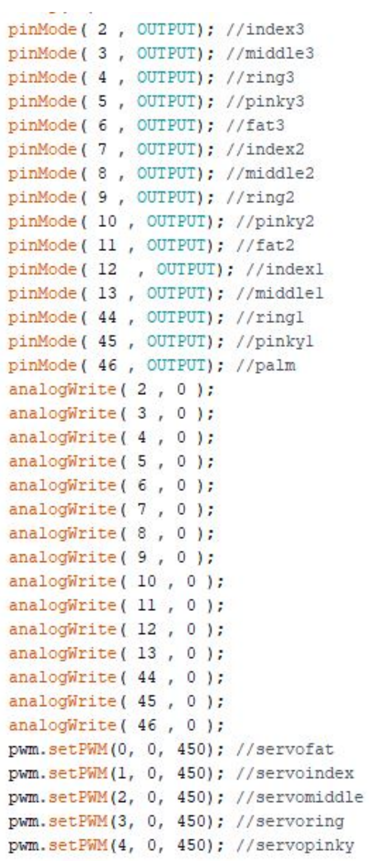


Figure 43: Exoskeleton's part with Servo Motor

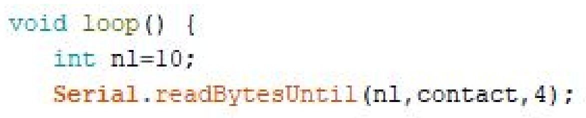
tion Serial.begin(9600) to open the serial port and set the data rate at 9600 bits per second. Then, the data were read with the function Serial.readBytesUntil (character, buffer, length), where character is the character to search for, buffer is the buffer to store the bytes in, and length is the number of bytes to read.



```
pinMode( 2 , OUTPUT); //index3
pinMode( 3 , OUTPUT); //middle3
pinMode( 4 , OUTPUT); //ring3
pinMode( 5 , OUTPUT); //pinky3
pinMode( 6 , OUTPUT); //fat3
pinMode( 7 , OUTPUT); //index2
pinMode( 8 , OUTPUT); //middle2
pinMode( 9 , OUTPUT); //ring2
pinMode( 10 , OUTPUT); //pinky2
pinMode( 11 , OUTPUT); //fat2
pinMode( 12  , OUTPUT); //index1
pinMode( 13 , OUTPUT); //middle1
pinMode( 44 , OUTPUT); //ring1
pinMode( 45 , OUTPUT); //pinky1
pinMode( 46 , OUTPUT); //palm
analogWrite( 2 , 0 );
analogWrite( 3 , 0 );
analogWrite( 4 , 0 );
analogWrite( 5 , 0 );
analogWrite( 6 , 0 );
analogWrite( 7 , 0 );
analogWrite( 8 , 0 );
analogWrite( 9 , 0 );
analogWrite( 10 , 0 );
analogWrite( 11 , 0 );
analogWrite( 12 , 0 );
analogWrite( 13 , 0 );
analogWrite( 44 , 0 );
analogWrite( 45 , 0 );
analogWrite( 46 , 0 );
pwm.setPWM(0, 0, 450); //servofat
pwm.setPWM(1, 0, 450); //servoindex
pwm.setPWM(2, 0, 450); //servomiddle
pwm.setPWM(3, 0, 450); //servoring
pwm.setPWM(4, 0, 450); //servopinky
```

Figure 44: Initialize Arduino's and PCA9685's pins

Specifically, this function reads characters from the serial buffer and saves them into an array. The number of characters that were read was four and was set from the third parameter. Also, the buffer is declared as an array and was named as contact in the second parameter. Moreover, the character to search for the termination process was set as 10 in ASCII code, which is the newline.

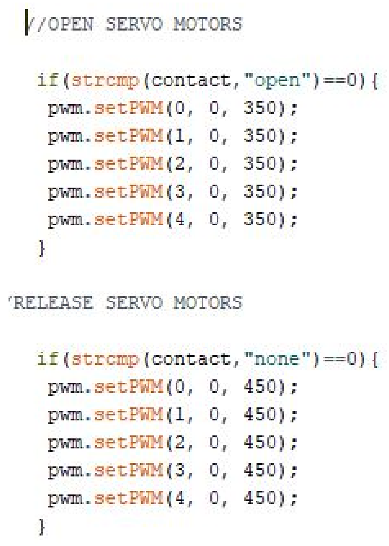


```
void loop() {
  int nl=10;
  Serial.readBytesUntil(nl,contact,4);
```

Figure 45: Read 4 characters and save them in contact

After reading the characters from the contact array, they were compared with the motors enabling/disabling strings using the function strcmp(). This function is used to compare two strings, that are passed through parameters. If these strings are equal, the function returns zero.

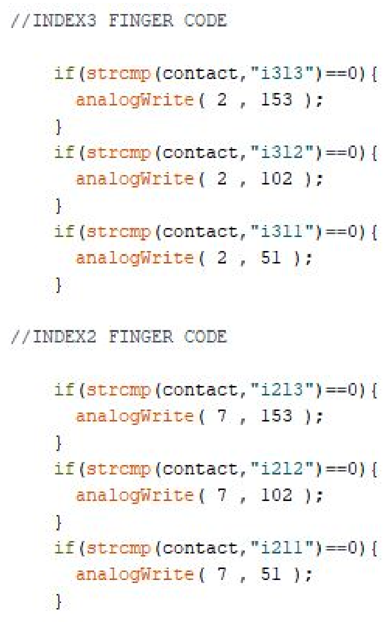
As was mentioned before, the servo motors provide kinesthetic feedback by turning at specific degrees. When the appropriate string is sent through the buffer, the motors are programmed to block the movement of the finger by turning at 54^o, which in the sketch this number was 350(ticks), and unblock the movement by turning 18^o, which in the sketch this number was 450(ticks).



```
//OPEN SERVO MOTORS

  if(strcmp(contact,"open")==0){
   pwm.setPWM(0, 0, 350);
   pwm.setPWM(1, 0, 350);
   pwm.setPWM(2, 0, 350);
   pwm.setPWM(3, 0, 350);
   pwm.setPWM(4, 0, 350);
  }

'RELEASE SERVO MOTORS

  if(strcmp(contact,"none")==0){
   pwm.setPWM(0, 0, 450);
   pwm.setPWM(1, 0, 450);
   pwm.setPWM(2, 0, 450);
   pwm.setPWM(3, 0, 450);
   pwm.setPWM(4, 0, 450);
  }
```

Figure 46: The code to block and unblock the finger movement

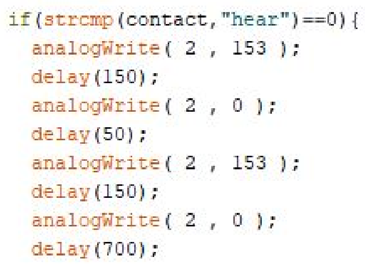
On the other hand, the vibration motors provide the tactile feedback with 3 different levels of intensity. This was achieved by connecting them to pins that provide PWM. PWM is a digital square wave, where the frequency is constant, but the duty cycle(from on to off) can be between 0(0V) and 255(5V). On top of that, by using PWM another issue was solved in this implementation. Because Arduino is supplied with 5V and the motors rated voltage is 3V, it was important to use PWM to reduce the applied voltage. This was accomplished with the function analogWrite() and setting the second parameter(duty cycle) to 3/5(153). Additionally, the other two levels of intensity were set at 2/5(102) and 1/5(51) of the duty cycle.



```
//INDEX3 FINGER CODE

    if(strcmp(contact,"i3l3")==0){
      analogWrite( 2 , 153 );
    }
    if(strcmp(contact,"i3l2")==0){
      analogWrite( 2 , 102 );
    }
    if(strcmp(contact,"i3l1")==0){
      analogWrite( 2 , 51 );
    }

//INDEX2 FINGER CODE

    if(strcmp(contact,"i2l3")==0){
      analogWrite( 7 , 153 );
    }
    if(strcmp(contact,"i2l2")==0){
      analogWrite( 7 , 102 );
    }
    if(strcmp(contact,"i2l1")==0){
      analogWrite( 7 , 51 );
    }
```

Figure 47: The code to enable or disable the vibrators

Last but not least, this thesis also takes advantage of the PWM to provide the feeling of heartbeat through vibrations. The following figure shows a part of the code.



```
if(strcmp(contact,"hear")==0){
  analogWrite( 2 , 153 );
  delay(150);
  analogWrite( 2 , 0 );
  delay(50);
  analogWrite( 2 , 153 );
  delay(150);
  analogWrite( 2 , 0 );
  delay(700);
```

Figure 48: The code for the heartbeat

4 3D Demo Application

In this section, a 3D demo application is presented, which was made to demonstrate the tactile and kinesthetic feedback of this thesis embedded system. In order to create this application, a game engine was used and more specifically the Unity3D. Also, most of the 3D objects inside the application were modeled using the Blender software.

4.1 Design of 3D Models

Blender is a free and open-source 3D computer graphics software used for creating 3D models, visual effects, animated films, interactive applications, etc. Furthermore, Blender is accessible for a wide range of users, from a beginner to a more advanced graphics designer.

Moving on to the interface of Blender, there are plenty of windows that should be explained.

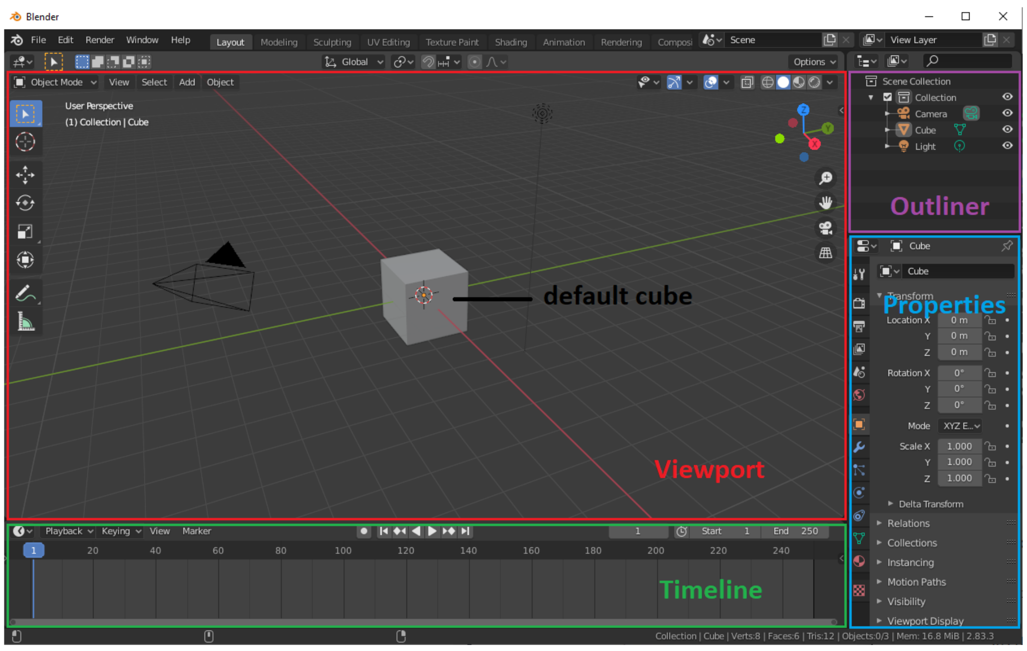


Figure 49: The Blender Interface

- **Viewport** is the largest area, where the 3D models are visible. When Blenders starts, a default cube is in the middle of Viewport. Generally, in Viewport the user can add details to the model and transform it(move, rotate, scale).
- **Timeline** is the bottom area, where the user by pressing the spacebar, will notice that a blue line will start to move from left to right. If there was any animation, it would start playing.
- **Outliner** is the top right area, where the user can find all the objects, including camera and lights that may be in the scene.
- **Properties** is the bottom right area, where the user can add materials to the objects, manipulate them with modifiers, etc.

Apart from the interface, there are some modeling basics(shown in figure 45) that should be explained.

- **Current Mode** - By pressing TAB, the user switches between Edit Mode and Object Mode. In Edit Mode the user can manipulate a model's vertices, edges and faces.

- **Vertices** - these are points in 3D space.
- **Edges** - these are the lines that connect the vertices
- **Faces** - these fill the spaces between edges and vertices

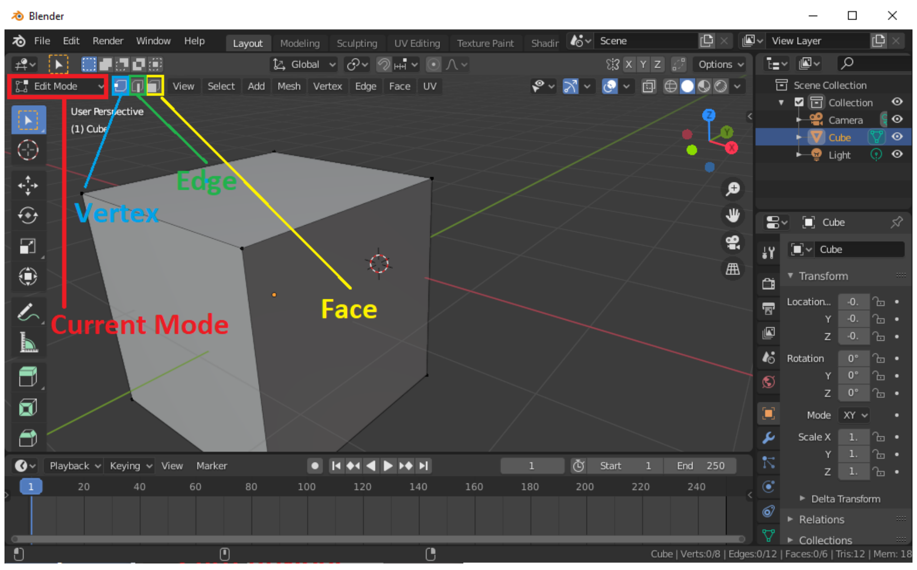


Figure 50: Vertex, Edge, Face in Edit Mode

In the following figures, some 3D models, that were made in Blender for the Unity application, will be presented.

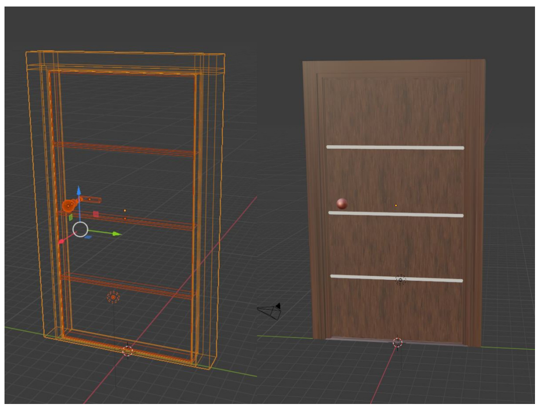


Figure 51: The room door 3D model in Wireframe and Object Mode

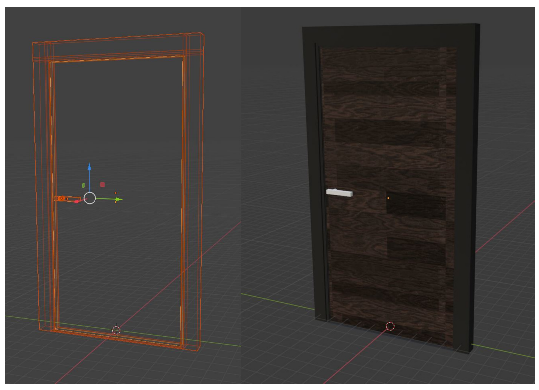


Figure 52: The room door 3D model in Wireframe and Object Mode

The two types of door that are shown in figure 46 and 47 are created so that the user can open them by interacting with the virtual environment. After the interaction is successful, the opening process is made with an animation that was set frame by frame(shown in figure 48).

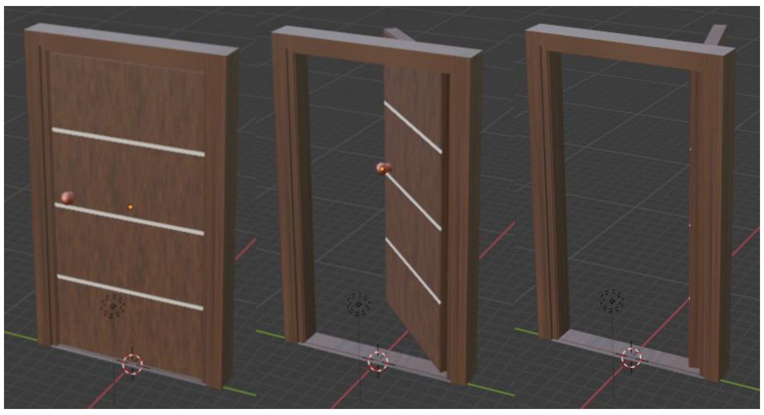


Figure 53: The room door 3D model Animation

To make the application more appealing, some other 3D models were created. Some of them were a balcony door with animation, a lamp, a bookcase with books and vases, a table, one small couch, one big couch, etc.

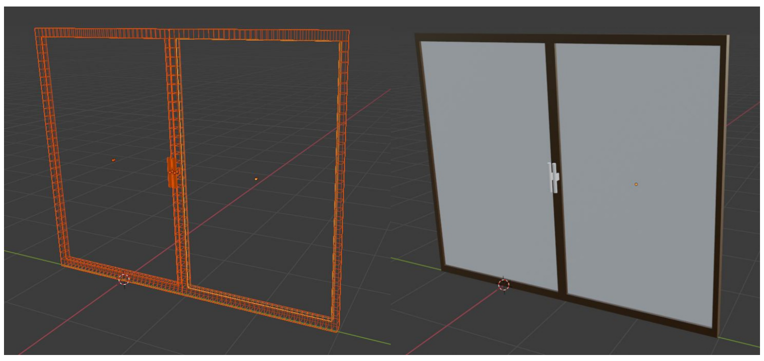


Figure 54: The balcony door 3D model in Wireframe and Object Mode

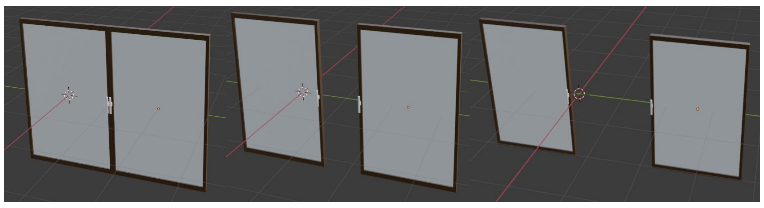


Figure 55: The balcony door 3D model Animation

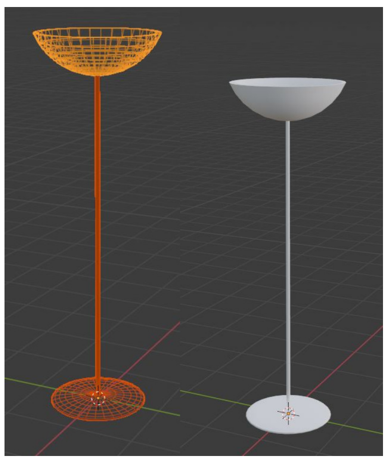


Figure 56: The lamp 3D model in Wireframe and Object Mode

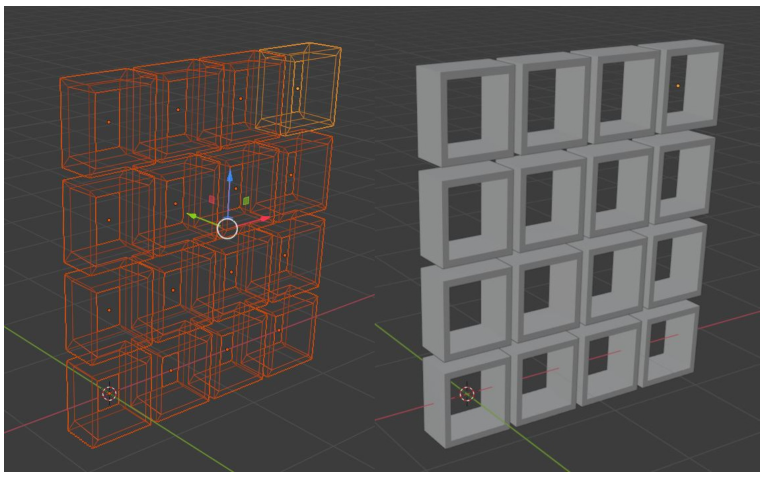


Figure 57: The bookcase 3D model in Wireframe and Object Mode

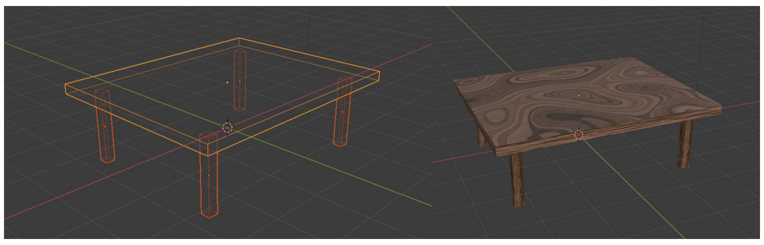


Figure 58: The table 3D model in Wireframe and Object Mode

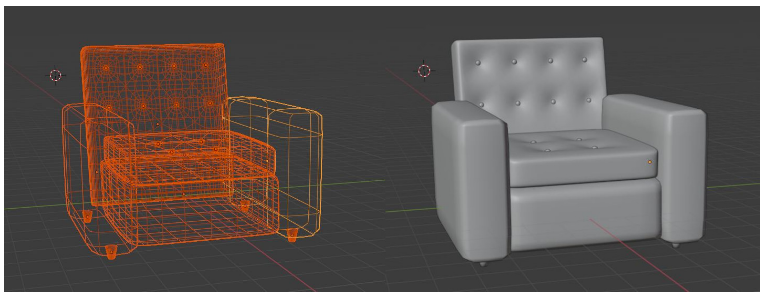


Figure 59: The small couch 3D model in Wireframe and Object Mode

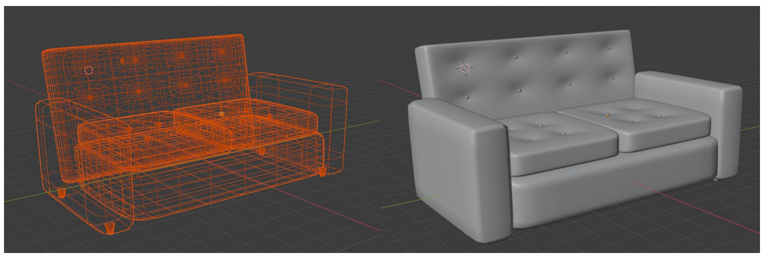
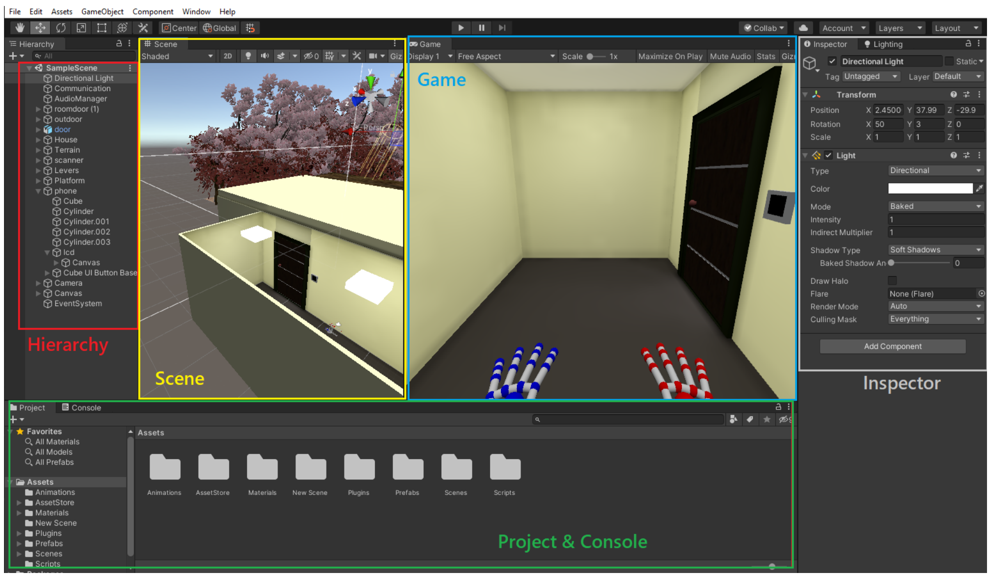


Figure 60: The big couch 3D model in Wireframe and Object Mode

4.2 Design of the Application

As was aforementioned the 3D application was made with Unity3D. Unity 3D is a game engine, that gives the users all the appropriate tools to create both 2D and 3D gaming environments. Additionally, it includes an Asset Store, which is a place where developers can purchase 3D objects amongst other things for their applications. Another advantage of Unity is that is cross-platform and a project can be moved from one platform to another. Some of the supported platforms are Windows, Mac, Android, iOS, PlayStation, Xbox, etc. Moreover, Unity has a big community and there is a plethora of online courses and tutorials, that the user can watch to learn the basics of this platform, thus making Unity the perfect choice.



Directional Light
Tag Untagged Layer Default
Transform
Position X 2.4500 Y 37.99 Z -29.9
Rotation X 50 Y 3 Z 0
Scale X 1 Y 1 Z 1
Light
Type Directional
Mode Baked
Intensity 1
Indirect Multiplier 1
Shadow Type Soft Shadows
Baked Shadow An 0
Flare None (Flare)
Render Mode Auto
Culling Mask Everything

Figure 61: The Unity Interface

In the Unity interface, the user will see the following windows:

- **Hierarchy** is on the left side of the interface and it shows the user a list with all the GameObjects in the active scene. From there the user can easily locate any GameObject and change its properties.
- **Scene** is in the middle of the interface and it shows the user a view of the scene that he is currently working with. In this window, the user can move, rotate, scale the GameObjects.
- **Game** is also in the middle of the interface and it shows the user a view of the scene, but this time from the perspective of the camera and he will not be able to move anything this time. Also, this is where the game is played when the play button is pressed.
- **Inspector** is on the right side of the interface and it shows the user the components of the selected GameObject.
- **Project & Console** is on the bottom side of the interface. In Project, the user can find all the files that consist of the application and in Console, he can see information from Unity, like errors and warnings.

Every GameObject contains components and some of them will be presented below:

- **Transform Component**

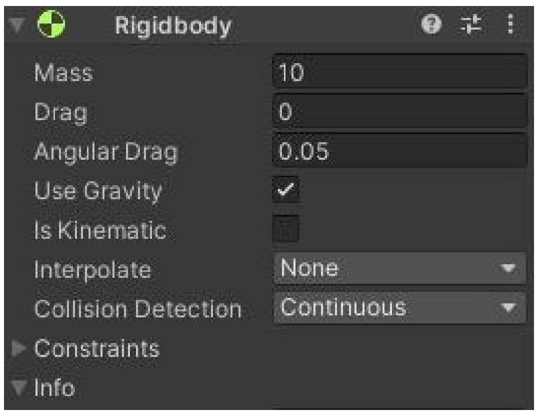
  This is the most important component because without it the GameObject can not exist. With this, the user can change the position, rotation, and scale of the GameObject. One important note is that the transform values of the object are relative to the parent's transform. In mesh renderer, the user can control if the object will cast shadows among other things.

- **Mesh Component**

  Mesh is the shape of the GameObject and it consists of vertices, edges, faces, normal maps(show how light bounces on the object), and UV maps(define how textures are placed on the object). The Mesh Filter contains this information and passes them to Mesh Renderer, which renders the GameObject in the scene.

- **Physics Component**

  Unity provides the users the ability to simulate physics in their application with a built-in physics engine(NVIDIA PhysX). More specifically, by using the Rigidbody component, the object is subjected to the physic's laws(it receives mass, drag, etc).



Rigidbody

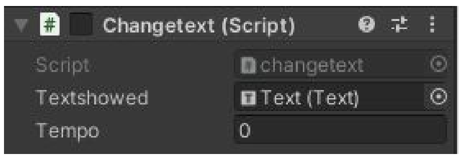
Mass	10
Drag	0
Angular Drag	0.05
Use Gravity	✓
Is Kinematic	
Interpolate	None
Collision Detection	Continuous
Constraints	
Info	

Figure 62: Rigidbody component

  With the Collider component users can have interactions, like collisions, between GameObjects. The collider has a similar shape with the object, but the user is unable to see it. The colliders can have a spherical shape (Sphere Collider), cuboid shape(Box Collider), cylindrical shape (Capsule Collider), etc.

- **Script Component**

  Unity offers to the users the ability to create scripts that control the GameObjects. The programming language used for scripting in Unity is C#. Every script contains two main functions. The first one is the **Start** function, which is called in the first frame and is used to initialize values. The second one is the **Update** function, which is called on every frame and is used for the main coding of the application, like triggering actions. Last but not least, the user can create his functions, or use functions that pass in the object through other components(like OnCollisionEnter, etc) to control the objects.



Changetext (Script)

Script	changetext
Textshowed	Text (Text)
Tempo	0

Figure 63: Script component

- **Audio Components**

  Audio is an important aspect when creating a game, thus Unity provides the users with Audio components. Some of these components are the

Audio Listener, which is usually attached to the Camera and receives all the sounds inside the game(works like a microphone), and the Audio Source, which plays a sound(source of the audio).

4.2.1 Virtual Hand and Collision Detection

To detect the hand and its movement in the virtual environment, a commercial solution was selected, the Leap Motion. Leap Motion is easily imported into Unity and with some modules, it can send the hand data, that are read from the controller, into the application. Additionally, it provides some hand prefabs for the virtual representation of the hand. More specifically, in this thesis, the Leap Motion Controller prefab was used, which contains the LeapServiceManager script(sets if the device is on head-mounted or desktop mode). Then the Hand Models GameObject was added(see figure 64), which has as children the right and left CapsuleHand(virtual representation of the hands), and the right and left RigidRoundHand(rigidbodies and colliders of the hands).

Each RigidRoundHand consists of five fingers and the palm. Then these fingers are divided into three areas, the bones. Furthermore, every bone(plus the palm) has a rigidbody and a capsule collider(palm has a box collider), so that the hand can interact with the virtual world. When the hand touches a GameObject with Collider and Rigidbody components, the function OnCollisionEnter is called and when this collision stops, the function OnCollisionExit is called.

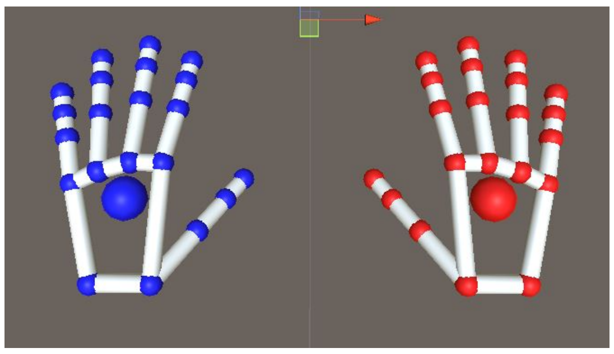
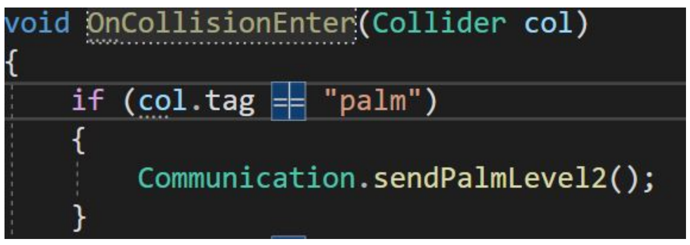


Figure 64: Hand Models in Unity

After Leap Motion is successfully imported, a serial port script was made. This script opens a serial port so that Unity and Arduino can communicate. This port took as parameters the name of the port and the data rate, which was 9600 bits per second.

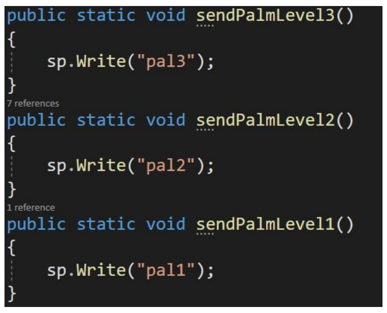
Additionally, when the virtual hand collides with a virtual object, the object checks which bone collided and calls the appropriate function(through the OnCollisionEnter or the OnCollisionExit functions).



```
void OnCollisionEnter(Collider col)
{
    if (col.tag == "palm")
    {
        Communication.sendPalmLevel2();
    }
```

Figure 65: OnCollisionEnter calls the SendPalmLevel2 function

This function is located in the serial port script and its job is to write characters in the serial buffer. These characters symbolize the bone that collided and the intensity of the collision happened. In the following figure ”pal3” means palm with intensity 3, etc.



```
public static void sendPalmLevel3()
{
    sp.Write("pal3");
}
7 references
public static void sendPalmLevel2()
{
    sp.Write("pal2");
}
1 reference
public static void sendPalmLevel1()
{
    sp.Write("pal1");
}
```

Figure 66: Characters send for Palm with the different intensities

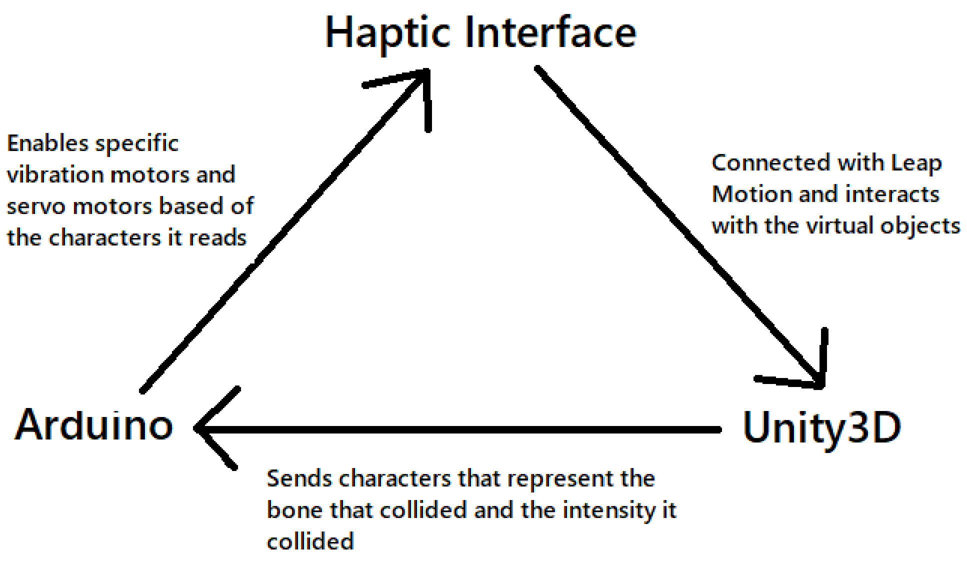

Haptic Interface
Enables specific vibration motors and servo motors based of the characters it reads
Connected with Leap Motion and interacts with the virtual objects
Arduino
Unity3D
Sends characters that represent the bone that collided and the intensity it collided

Figure 67: Representation of the communication

4.2.2 Application's Environment and Concept

To point out the abilities of the embedded system that was created in this thesis, a 3D game application was implemented. The game consists of three scenes, the main menu, the main game, and the ending.

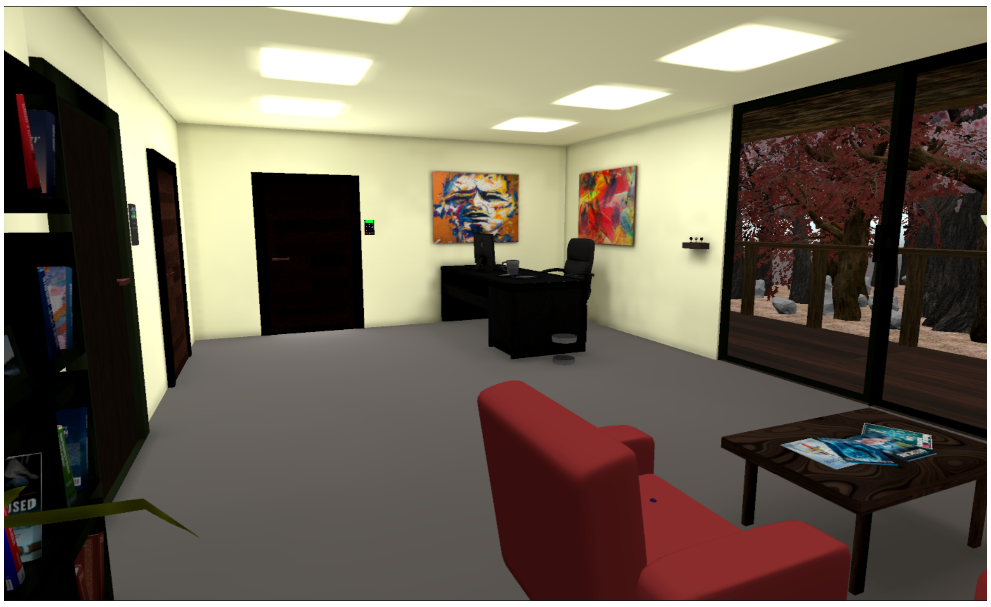
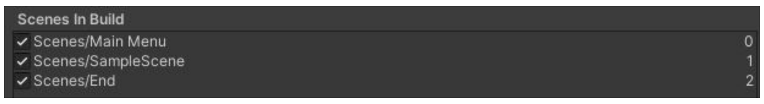


Figure 68: 3D application's environment

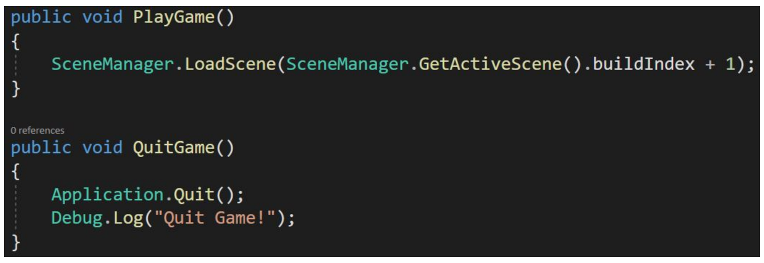
The main menu is a canvas with two buttons, the Play, which starts the main game, and the Quit, which quits the application. With the PlayGame()

function the next scene in Build Settings is loaded, while with the QuitGame() function the running application is closed. These functions are executed from the canvas script and they are called when the OnClick event of each button is triggered.



```
Scenes In Build
✓ Scenes/Main Menu      0
✓ Scenes/SampleScene    1
✓ Scenes/End            2
```

Figure 69: Build Settings



```
public void PlayGame()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
}

0 references
public void QuitGame()
{
    Application.Quit();
    Debug.Log("Quit Game!");
}
```

Figure 70: Main Menu script

After the Play button is pressed, the Main Game Scene is loaded. This is an escape room type of game, which takes place in a doctor's office. The main goal of the game is to unlock the doctor's office door, but the player has to complete four tasks first. The player can move forward by pressing the left mouse button and backward by pressing the right mouse button. Also, he can look by moving the mouse around.

At the start of the game the player is placed outside of the office and the first task is to unlock the front door to get in. To achieve that, he has to put his hand inside the black square, which works like a scanner and provides the player with tactile feedback.

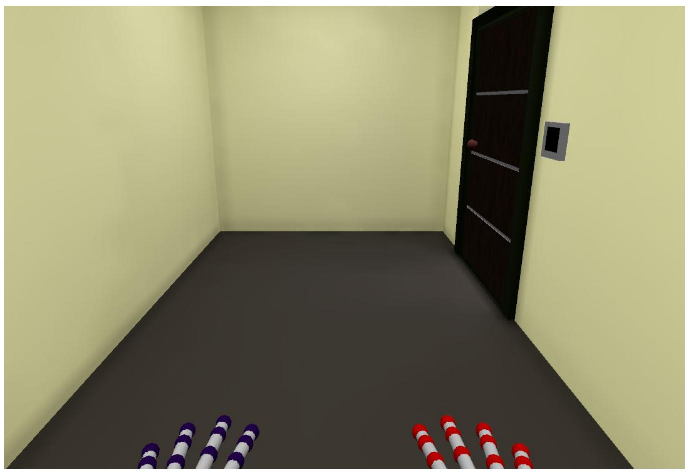
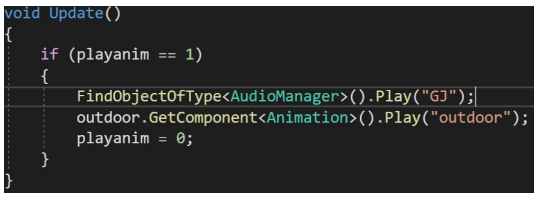


Figure 71: The front door task

The collider of the scanner is marked as a trigger so that the player can insert his virtual hand. When the whole hand is in and then pulled out, the animation of the door and a congratulations sound are played. Besides, with the OnTriggerEnter() function, every bone that enters the scanner will call the appropriate function in the serial port script, which will send the corresponding characters to the Arduino(through the serial buffer). More specifically, whichever part of the hand enters the scanner, its equivalent motor will vibrate. Furthermore, with the OnTriggerExit() function the motor, that is linked with the bone that exits, will stop vibrating.

After the front door is opened, the player enters the doctor's office waiting room. In this area, he will find some 3D objects like two couches, a small



```
void Update()
{
    if (playanim == 1)
    {
        FindObjectOfType<AudioManager>().Play("GJ");
        outdoor.GetComponent<Animation>().Play("outdoor");
        playanim = 0;
    }
}
```

Figure 72: Script that plays an audio and an animation

table, a bookcase, a desk with a chair, two doors, a keypad, and three levers next to the balcony door. These three levers are the second task and provide the player with both kinesthetic and tactile feedback.

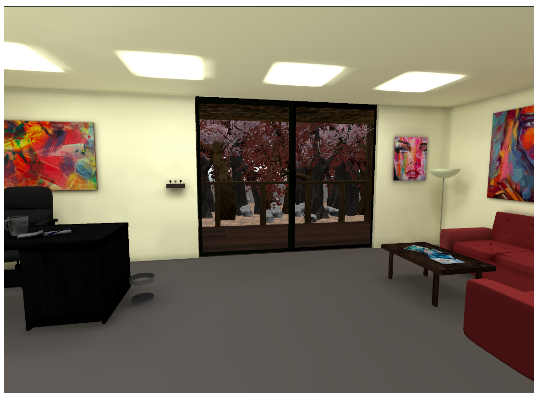
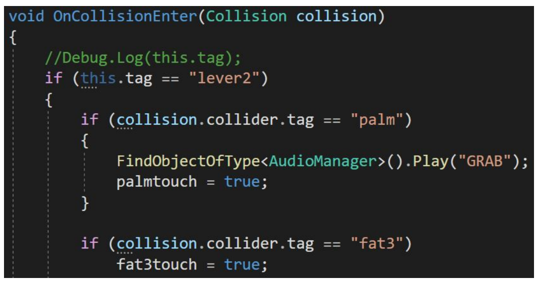


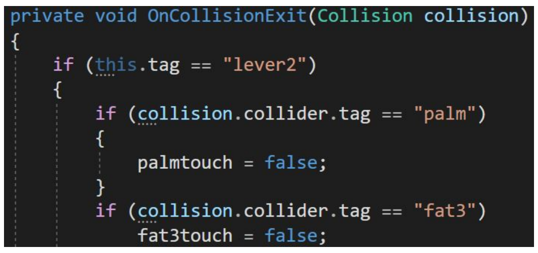
Figure 73: The Levers task

In this task, the player needs to pull three levers to open the balcony door. Every lever has a configurable joint component, which allows the lever to move only back and forth, and a grablever script. Specifically, the script has an OnCollisionEnter() function and if both the palm and the fingers collide with the object, a grabbing sound is played and a testvalue parameter is set to 1. This process is done by setting some boolean values to true when a collision happens. In the Update() function, if the value for the palm and the values for the fingers are true, the testvalue is set to 1. However, with the OnCollisionExit() function, when collision exits, these values are set to false, thus the testvalue now is set to 0.



```
void OnCollisionEnter(Collision collision)
{
    //Debug.Log(this.tag);
    if (this.tag == "lever2")
    {
        if (collision.collider.tag == "palm")
        {
            FindObjectOfType<AudioManager>().Play("GRAB");
            palmtouch = true;
        }

        if (collision.collider.tag == "fat3")
            fat3touch = true;
```

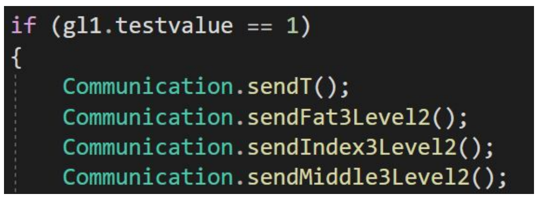
Figure 74: OnCollisionEnter sets values that collide as true



```
private void OnCollisionExit(Collision collision)
{
    if (this.tag == "lever2")
    {
        if (collision.collider.tag == "palm")
        {
            palmtouch = false;
        }
        if (collision.collider.tag == "fat3")
            fat3touch = false;
```

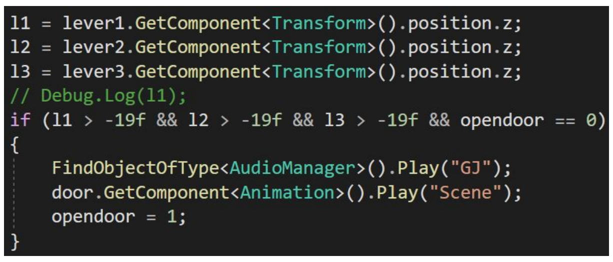
Figure 75: OnCollisionExit sets values that stop colliding as false

Then, the parent object, which contains the three levers, has a script. This script checks if the value from every lever is set to 1 and if so it calls the appropriate function from the serial port script to send the characters to Arduino. Then, Arduino reads these characters, enables the vibration of the motors, and turns the servo motors to block the hand's movement. Moreover, the parent's script takes the position of the three levers and if they are pulled by the player at a specific position, a congratulations sound and the animation of the balcony door are played(part of the script is shown in the figures below).



```
if (gl1.testvalue == 1)
{
    Communication.sendT();
    Communication.sendFat3Level2();
    Communication.sendIndex3Level2();
    Communication.sendMiddle3Level2();
```

Figure 76: Checks if the value from each lever is 1 and calls the serial port's functions



```
l1 = lever1.GetComponent<Transform>().position.z;
l2 = lever2.GetComponent<Transform>().position.z;
l3 = lever3.GetComponent<Transform>().position.z;
// Debug.Log(l1);
if (l1 > -19f && l2 > -19f && l3 > -19f && opendoor == 0)
{
    FindObjectOfType<AudioManager>().Play("GJ");
    door.GetComponent<Animation>().Play("Scene");
    opendoor = 1;
}
```

Figure 77: Checks position to play an audio sound and the animation

After opening the balcony door, the player is able to go to the balcony, where he will find a platform with two spheres. This platform with the spheres is the third task and provides the player with both tactile and kinesthetic feedback.

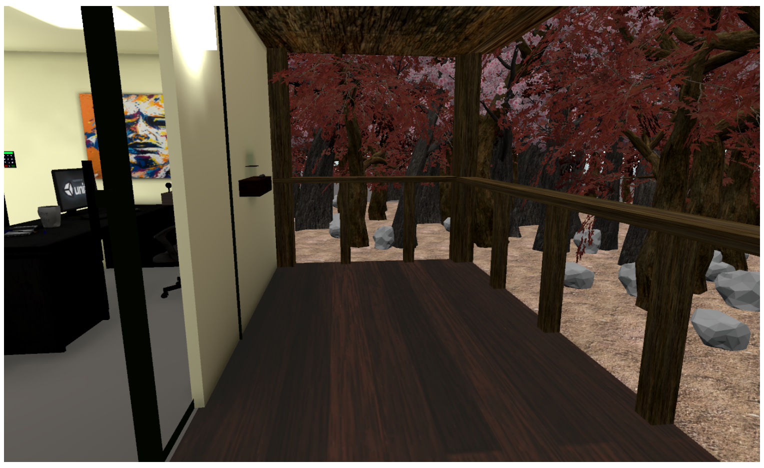


Figure 78: The Platform task

In this task, the player needs to place these two spheres on top of the platform, which works like a scale. If both are on, the scale will touch the ground a congratulation sound and a UI will pop up with information about the code for the doctor's office door(by pressing the Escape key the user can close this pop-up window). The tactile and kinesthetic feedback is provided in the same way as in the levers task. Specifically, each sphere contains a script, wherewith the OnCollisionEnter() function if both the fingers and the palm collide with it, a grabbing sound will be played and a testvalue parameter will be set to 1. With the OnCollisionExit this value will be set to 0.

Then, the parent's script will check if this value is set to 1 and will call the serial port script to send characters to Arduino. These characters will enable the corresponding vibration motors and the servo motors.

After completing the third task, the player can now enter the password on the keypad, which is next to the doctor's office door. This task provides the player with only tactile feedback.

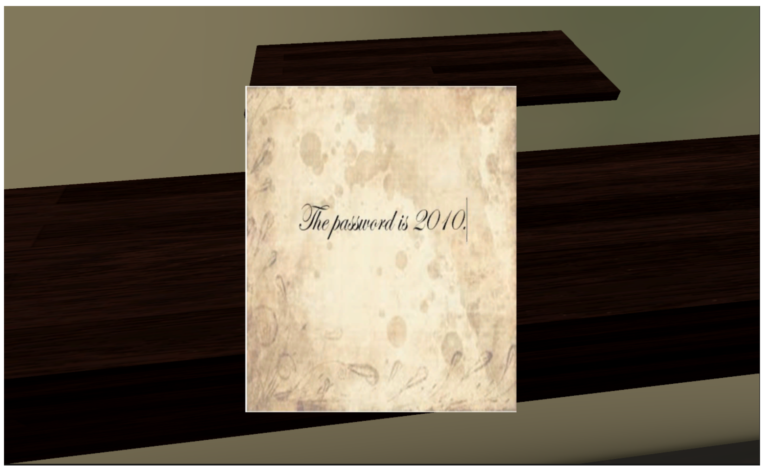
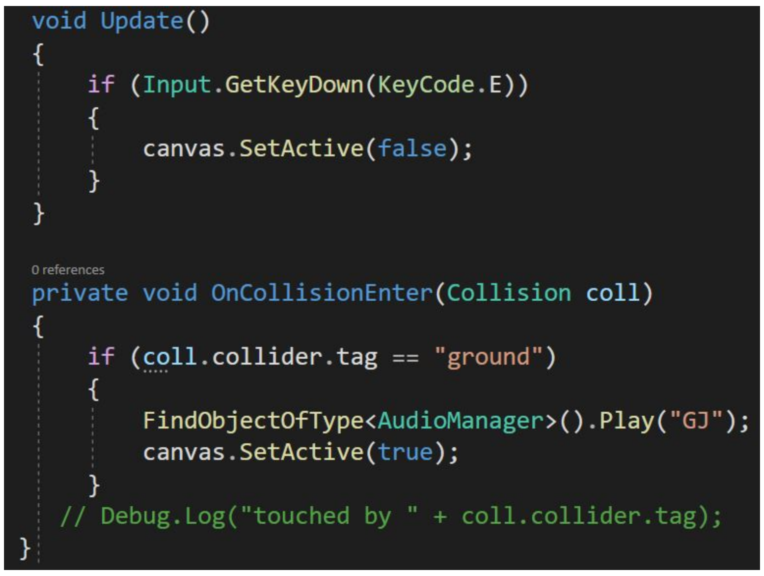


Figure 79: The Password pop-up window



```
void Update()
{
    if (Input.GetKeyDown(KeyCode.E))
    {
        canvas.SetActive(false);
    }
}

0 references
private void OnCollisionEnter(Collision coll)
{
    if (coll.collider.tag == "ground")
    {
        FindObjectOfType<AudioManager>().Play("GJ");
        canvas.SetActive(true);
    }
  // Debug.Log("touched by " + coll.collider.tag);
}
```

Figure 80: A part of the Platform task code

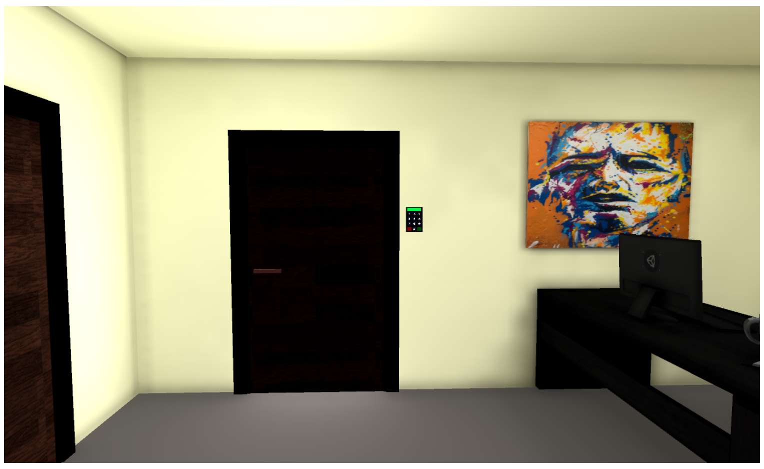
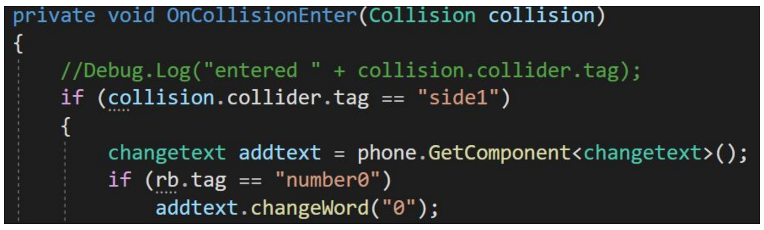


Figure 81: The Password Task

In this task, the user has to input the password, he acquired from the platform task, to open the doctor's office door. Every key on the keypad has a configurable joint component to simulate the movement of a button. Also, it has a script, which calls the changeWord() function from the parent object with the name of the key, that was pressed, as a string parameter. Moreover, the keypad has an LCD, which visualizes the keys that are pressed.

The parent's script is connected with a text object and has the function changeWord(), which changes the text in the following way.

- If the name of the key that was pressed is "Back", then a click sound is

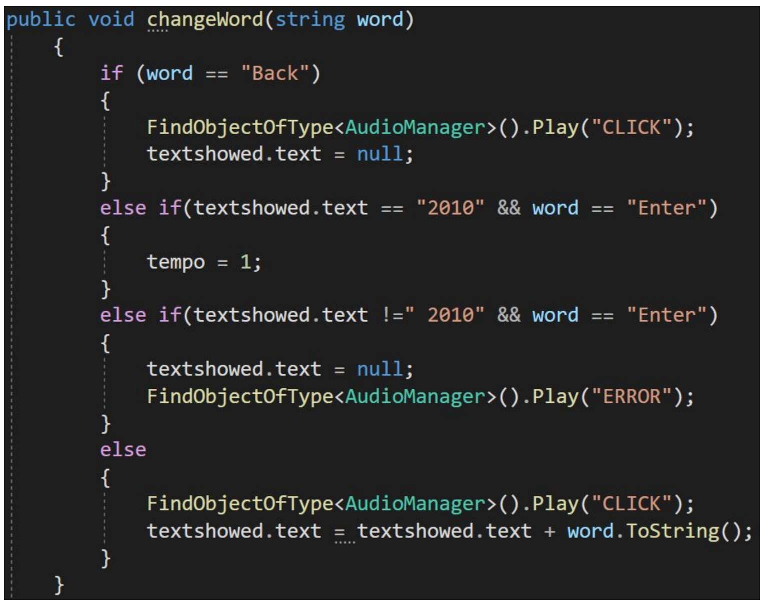


```
private void OnCollisionEnter(Collision collision)
{
    //Debug.Log("entered " + collision.collider.tag);
    if (collision.collider.tag == "side1")
    {
        changetext addtext = phone.GetComponent<changetext>();
        if (rb.tag == "number0")
            addtext.changeWord("0");
```

Figure 82: The collision calls the changeWord

played, and the string in the text object is set to null.

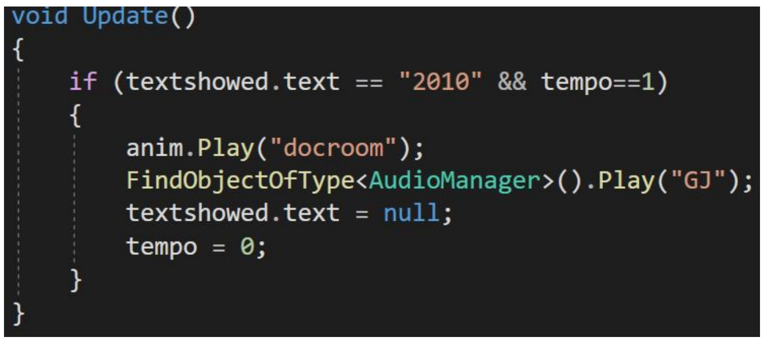
- If the name of the key that was pressed is anything from 0 to 9, a click sound is played and the number is added to the string.
- If the name of the key that was pressed is "Enter" and the string is not "2010", then an error sound is played and the string in the text object is set to null.
- If the name of the key that was pressed is "Enter" and the string is "2010", then a value, named tempo, is set to 1.



```
public void changeWord(string word)
    {
        if (word == "Back")
        {
            FindObjectOfType<AudioManager>().Play("CLICK");
            textshowed.text = null;
        }
        else if(textshowed.text == "2010" && word == "Enter")
        {
            tempo = 1;
        }
        else if(textshowed.text !=" 2010" && word == "Enter")
        {
            textshowed.text = null;
            FindObjectOfType<AudioManager>().Play("ERROR");
        }
        else
        {
            FindObjectOfType<AudioManager>().Play("CLICK");
            textshowed.text = textshowed.text + word.ToString();
        }
    }
```

Figure 83: The changeWord() function

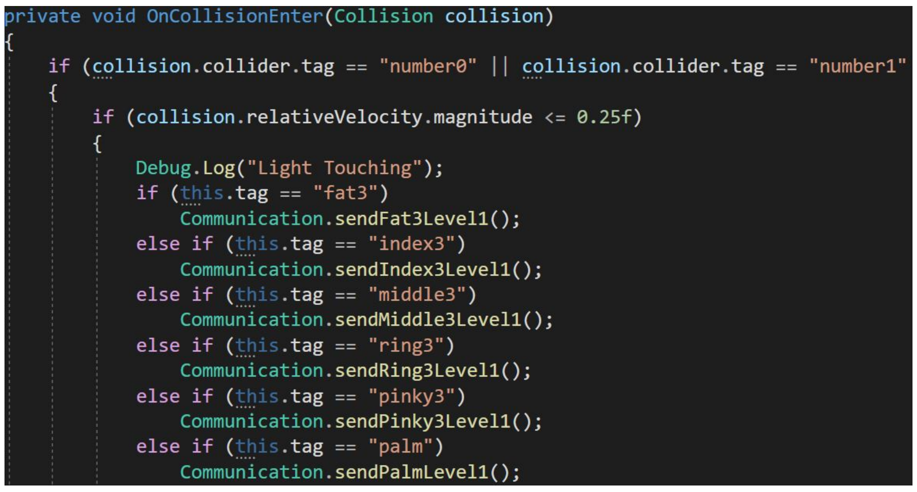
Then, in the Update() function if the tempo is 1, a congratulation sound, and the animation of the door are played.



```
void Update()
{
    if (textshowed.text == "2010" && tempo==1)
    {
        anim.Play("docroom");
        FindObjectOfType<AudioManager>().Play("GJ");
        textshowed.text = null;
        tempo = 0;
    }
}
```

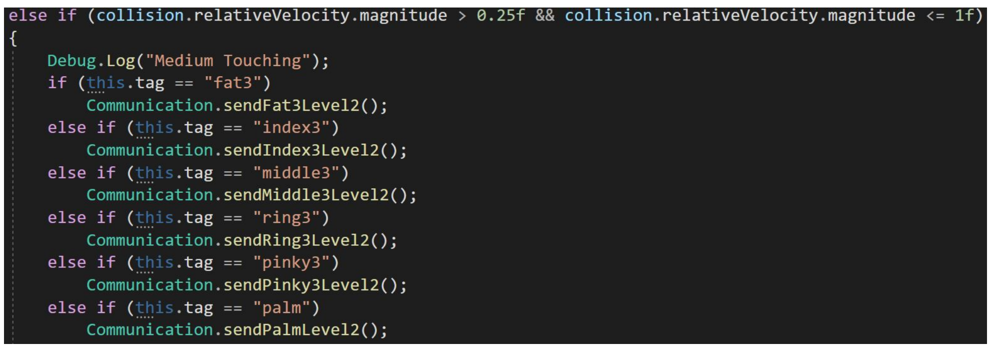
Figure 84: The script with the doctor's office door animation

Additionally, this application provides the player with three different intensities while pressing the buttons on the keypad. By using the relativeVelocity.magnitude, this application can detect the power of the impact between the fingers and the virtual object.



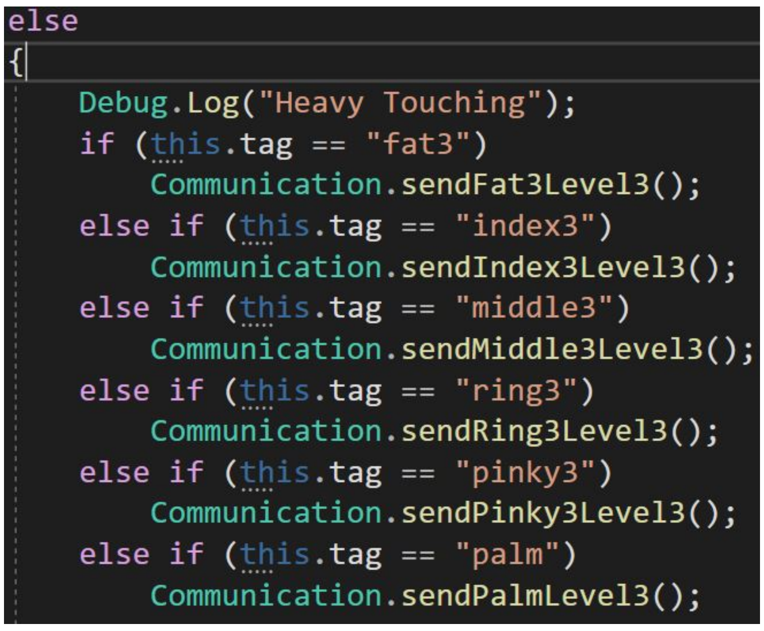
```
private void OnCollisionEnter(Collision collision)
{
    if (collision.collider.tag == "number0" || collision.collider.tag == "number1"
    {
        if (collision.relativeVelocity.magnitude <= 0.25f)
        {
            Debug.Log("Light Touching");
            if (this.tag == "fat3")
                Communication.sendFat3Level1();
            else if (this.tag == "index3")
                Communication.sendIndex3Level1();
            else if (this.tag == "middle3")
                Communication.sendMiddle3Level1();
            else if (this.tag == "ring3")
                Communication.sendRing3Level1();
            else if (this.tag == "pinky3")
                Communication.sendPinky3Level1();
            else if (this.tag == "palm")
                Communication.sendPalmLevel1();
```

Figure 85: Light touching buttons code



```
else if (collision.relativeVelocity.magnitude > 0.25f && collision.relativeVelocity.magnitude <= 1f)
{
    Debug.Log("Medium Touching");
    if (this.tag == "fat3")
        Communication.sendFat3Level2();
    else if (this.tag == "index3")
        Communication.sendIndex3Level2();
    else if (this.tag == "middle3")
        Communication.sendMiddle3Level2();
    else if (this.tag == "ring3")
        Communication.sendRing3Level2();
    else if (this.tag == "pinky3")
        Communication.sendPinky3Level2();
    else if (this.tag == "palm")
        Communication.sendPalmLevel2();
```

Figure 86: Medium touching buttons code



```
else
{
    Debug.Log("Heavy Touching");
    if (this.tag == "fat3")
        Communication.sendFat3Level3();
    else if (this.tag == "index3")
        Communication.sendIndex3Level3();
    else if (this.tag == "middle3")
        Communication.sendMiddle3Level3();
    else if (this.tag == "ring3")
        Communication.sendRing3Level3();
    else if (this.tag == "pinky3")
        Communication.sendPinky3Level3();
    else if (this.tag == "palm")
        Communication.sendPalmLevel3();
```

Figure 87: Heavy touching buttons code

During the main game, there is a pause menu, which is activated when the user is pressing the Escape key on his keyboard. The pause menu consists of three buttons, the Resume, the Main Menu, and the Quit.
Specifically, when the pause menu is activated, the player is not able to move or rotate the camera.

In the Pause Menu the player can perform the following actions

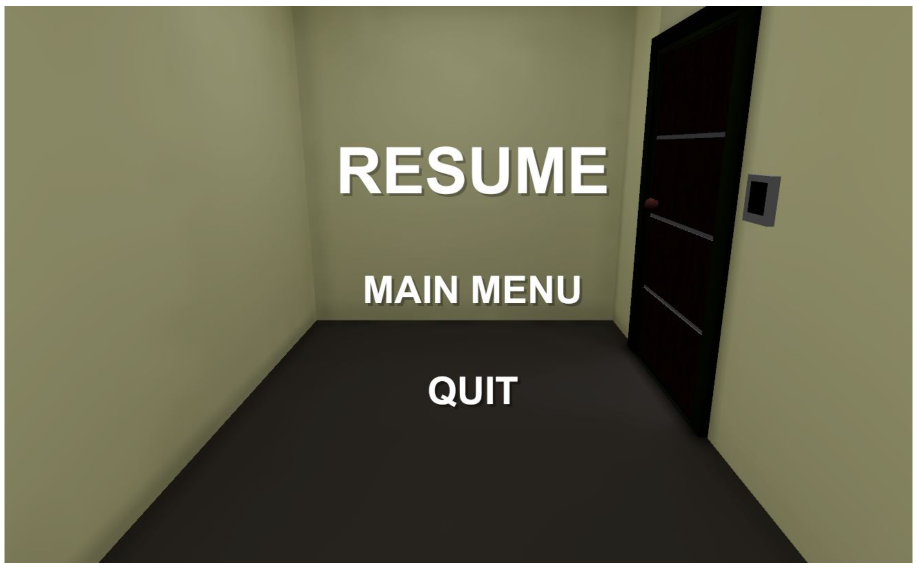
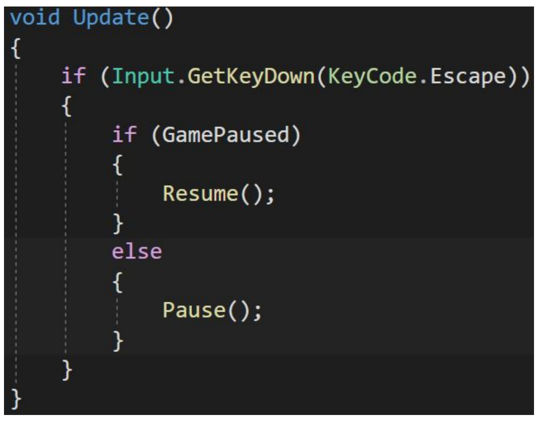
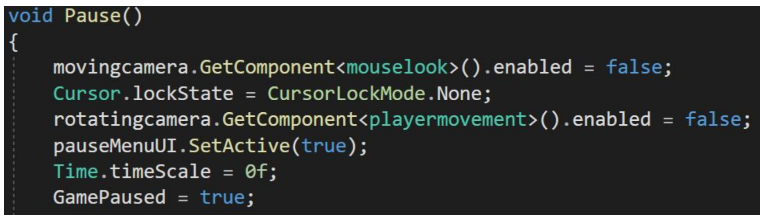


Figure 88: Pause Menu



```
void Update()
{
    if (Input.GetKeyDown(KeyCode.Escape))
    {
        if (GamePaused)
        {
            Resume();
        }
        else
        {
            Pause();
        }
    }
}
```

Figure 89: Pause Menu Update Script



```
void Pause()
{
    movingcamera.GetComponent<mouselook>().enabled = false;
    Cursor.lockState = CursorLockMode.None;
    rotatingcamera.GetComponent<playermovement>().enabled = false;
    pauseMenuUI.SetActive(true);
    Time.timeScale = 0f;
    GamePaused = true;
```

Figure 90: Pause Script

- If he presses the Resume button, the Pause Menu will close and he will be able to move in the virtual environment again.
- If he presses the Main Menu button, he will be sent to the Main Menu Scene.
- If he presses the Quit button, the application will close.

After the door opens, a white light appears and the player can enter, which will load the ending scene.

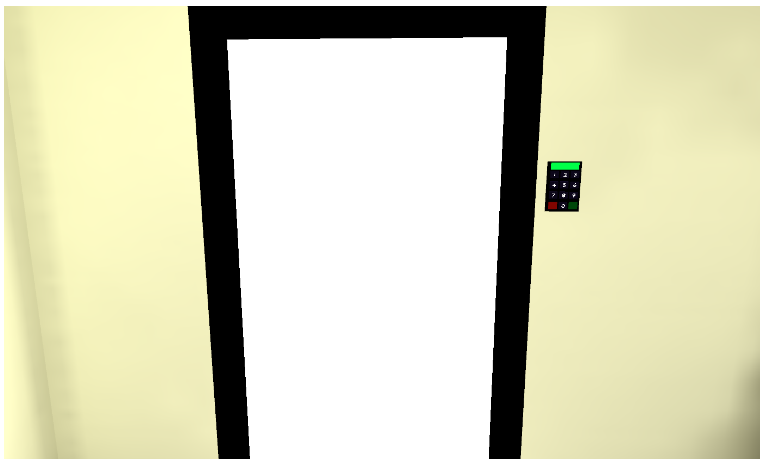
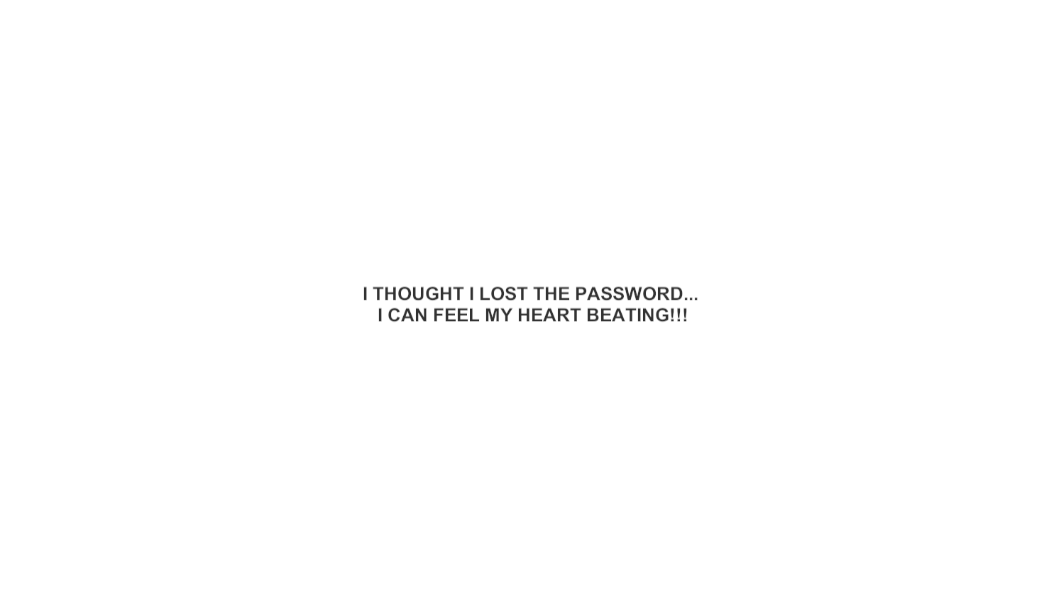


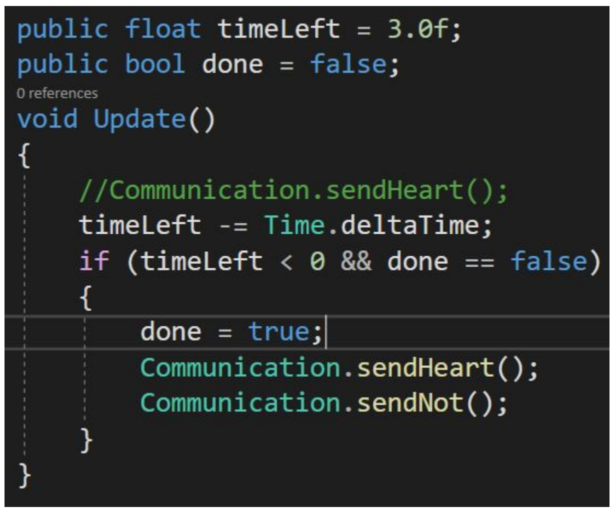
Figure 91: Doctor's office door opened

When he enters, a white canvas will appear with a message. The message is representing the player's thought. The message says that "I thought I lost the password... I can feel my heart beating!!!". Then the player is feeling a heartbeat in his hand by the vibration motor and the game ends.



I THOUGHT I LOST THE PASSWORD...
I CAN FEEL MY HEART BEATING!!!

Figure 92: The End Game scene



```
public float timeLeft = 3.0f;
public bool done = false;
0 references
void Update()
{
    //Communication.sendHeart();
    timeLeft -= Time.deltaTime;
    if (timeLeft < 0 && done == false)
    {
        done = true;
        Communication.sendHeart();
        Communication.sendNot();
    }
}
```

Figure 93: The script for the heartbeat

5 Evaluation

5.1 General Procedure

In order to measure the sense of presence in the virtual environment an evaluation of the glove was conducted. Specifically, the participants wore the haptic device, which is presented in this thesis, and started playing the demo application that was described in the previous chapter. After they finished the game, they gave their opinion on the embedded system and completed a questionnaire.

5.2 Questionnaire

The questionnaire that was used for the evaluation is Version 3 of the Presence Questionnaire [17]. This questionnaire consists of 32 questionS and is divided into four categories, which influence the sense of presence in the virtual environment. These categories are :

- **Involvement**

  The questions to check how much the participants are involved in the game are the following:

  1) How much were you able to control events?

  2) How responsive was the environment to actions that you initiated (or performed)?

  3) How natural did your interactions with the environment seem?

  4) How much did the visual aspects of the environment involve you?

  6) How natural was the mechanism which controlled movement through the environment?

  7) How compelling was your sense of objects moving through space?

  8) How much did your experiences in the virtual environment seem consistent with your real world experiences?

  10) How completely were you able to actively survey or search the environment using vision?

  14) How compelling was your sense of moving around inside the virtual environment?

  17) How well could you move or manipulate objects in the virtual environment?

  18) How involved were you in the virtual environment experience?

  29) How easy was it to identify objects through physical interaction, like touching an object, walking over a surface, or bumping into a wall or object?

- **Immersion**

  The questions to check how much the participants are immersed in the game are the following:

  9) Were you able to anticipate what would happen next in response to the actions that you performed?

  20) How quickly did you adjust to the virtual environment experience?

21) How proficient in moving and interacting with the virtual environment did you feel at the end of the experience?

24) How well could you concentrate on the assigned tasks or required activities rather than on the mechanisms used to perform those tasks or activities?

25) How completely were your senses engaged in this experience?

30) Were there moments during the virtual environment experience when you felt completely focused on the task or environment?

31) How easily did you adjust to the control devices used to interact with the virtual environment?

32) Was the information provided through different senses in the virtual environment (e.g., vision, hearing, touch) consistent?

- **Sensory fidelity**

  The questions to check the sensory fidelity are the following:

  5) How much did the auditory aspects of the environment involve you?

  11) How well could you identify sounds?

  12) How well could you localize sounds?

  13) How well could you actively survey or search the virtual environment using touch?

  15) How closely were you able to examine objects?

  16) How well could you examine objects from multiple viewpoints?

- **Interface Quality**

  The questions to check how much the participants are distracted from the haptic device are the following:

  13) How well could you actively survey or search the virtual environment using touch?

  22) How much did the visual display quality interfere or distract you from performing assigned tasks or required activities?

  23) How much did the control devices interfere with the performance of assigned tasks or with other activities?

5.3 Questionnaire's Results

After listening to the participants' opinions, when they completed the game and reviewing their answers from the questionnaire, it is concluded that their experience was generally good. Specifically, as they said, in the virtual environment the interaction with the virtual objects felt the same way as interacting with real ones. Also, they praised the implementation of the heartbeat feeling, when they completed the game, and the fact that they could feel different vibration while touching the objects aggressively or softly. However, some of them pointed out some negatives of the haptic interface and the application, like the weight of the whole system, the restriction of the hand's movement with so many components, a problem with Leap Motion, which sometimes does not detect the hand properly, and the lack of information about the steps they have to follow in order to finish the game. Last but not least, the participants' answers for every question are presented in the following pages, along with the total results from every one of the four categories that influence the sense of presence.

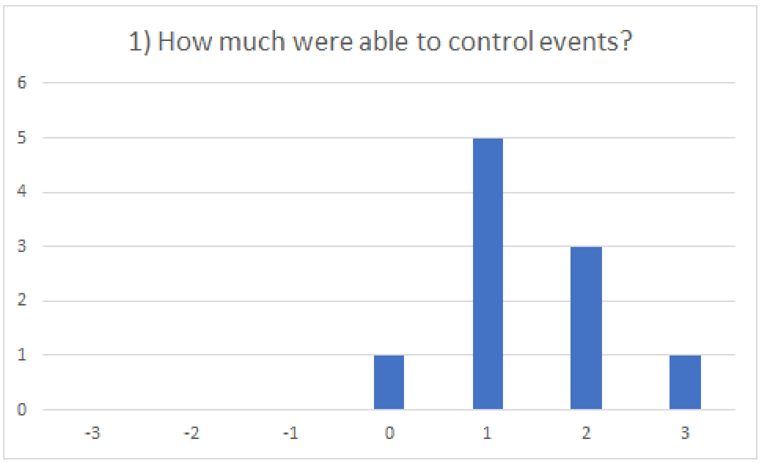


Figure 94: The participants' answers to Question 1

From the participants' answers to the first question, it is concluded that they were able to control things fairly well and good.

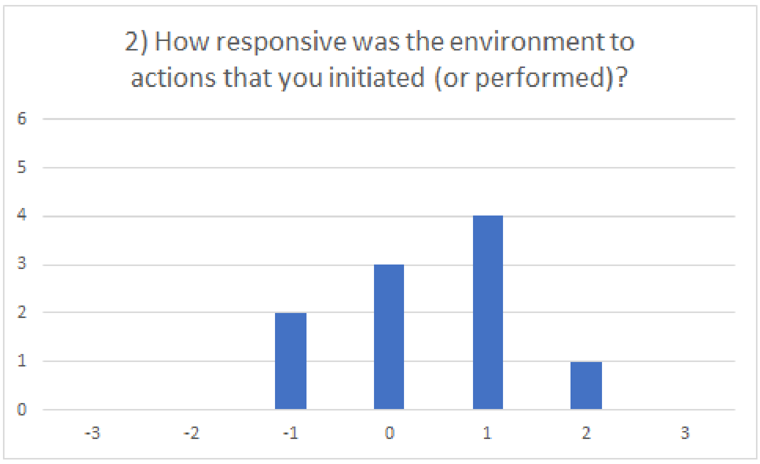


Figure 95: The participants' answers to Question 2

In the second question, the participants' answers differentiated, because of the Leap Motion, which sometimes did not work properly.

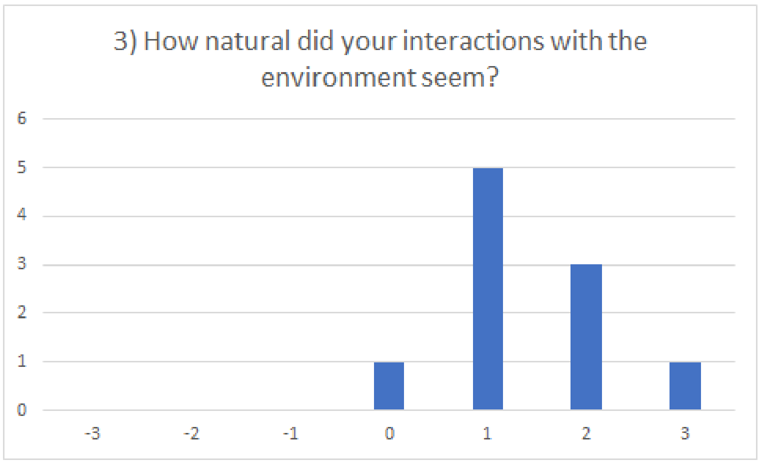


Figure 96: The participants' answers to Question 3

In the third question, the participants mainly agreed that the interactions with the environment seemed natural.

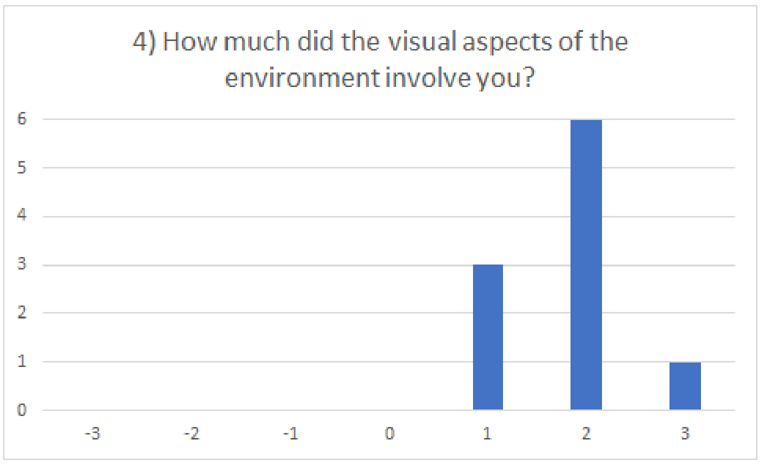


Figure 97: The participants' answers to Question 4

In the fourth question, it is concluded that the visual aspects of the game involved the participant in the environment.

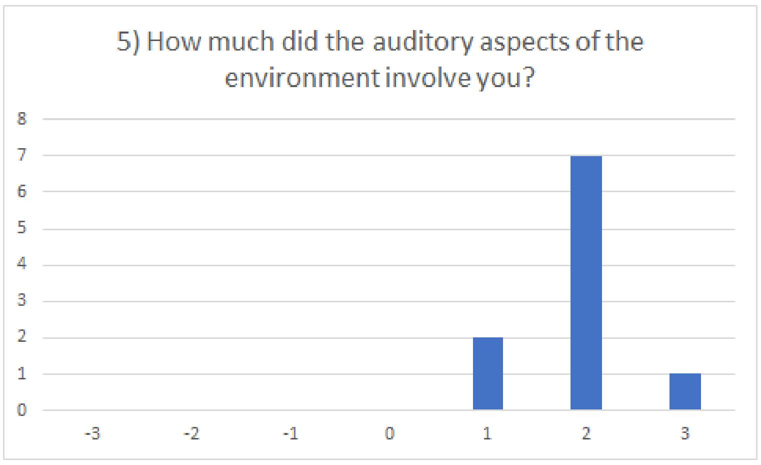


Figure 98: The participants' answers to Question 5

In the fifth question, it is also concluded that the auditory feedback of the game involved them in the environment.

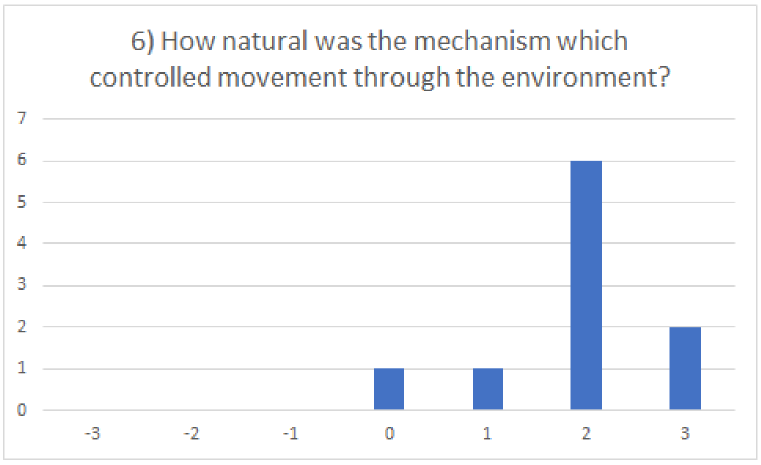


Figure 99: The participants' answers to Question 6

In the sixth question, the participants found the movement of the game very natural.

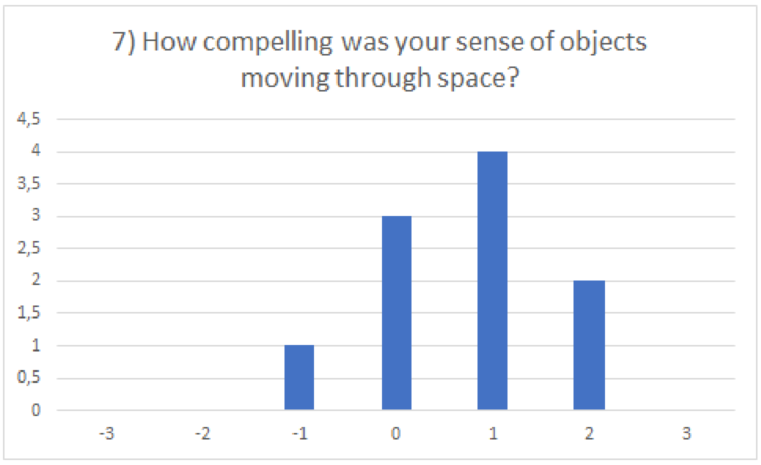


Figure 100: The participants' answers to Question 7

In the seventh question, some of the participants found the sense of objects moving quite good and one of them found it bad.

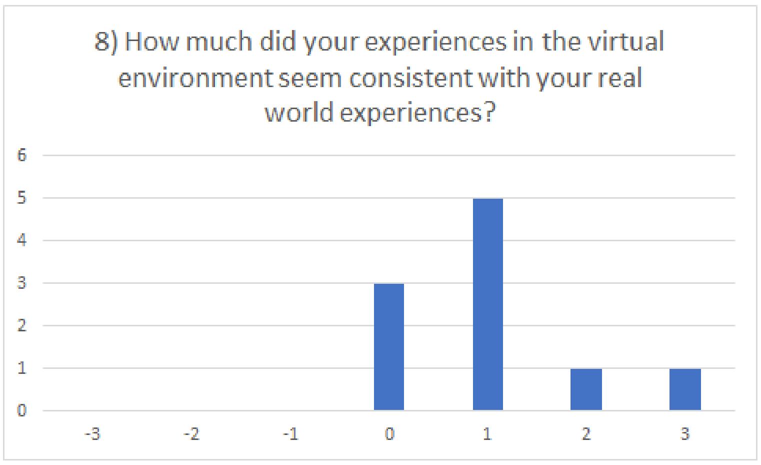


Figure 101: The participants' answers to Question 8

In the eighth question, it is concluded that the participants' experience in the virtual environment seemed fairly consistent with the real world.

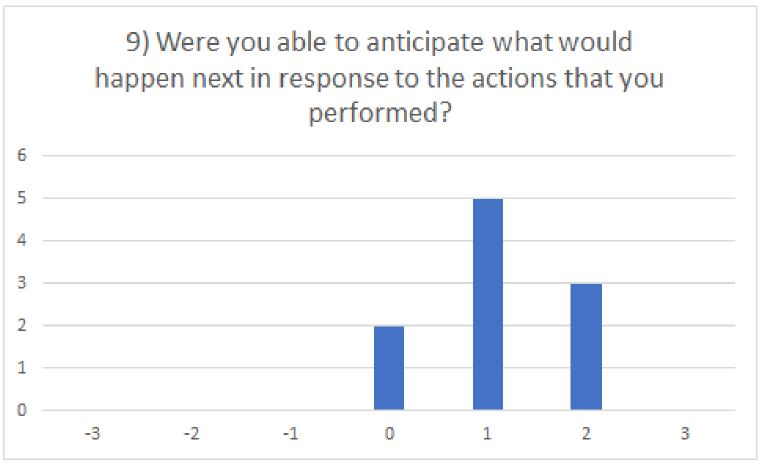


Figure 102: The participants' answers to Question 9

In the ninth question, the participants anticipated what would happen with their actions in the virtual environment.

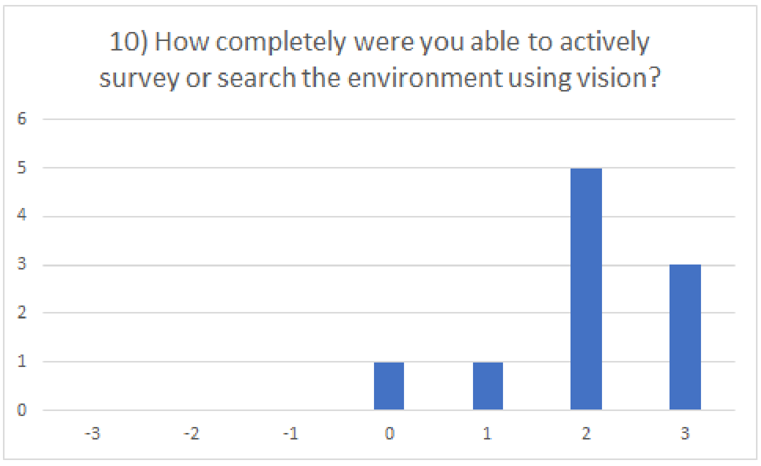


Figure 103: The participants' answers to Question 10

In the tenth question, the participants found the search of the area through vision very good.

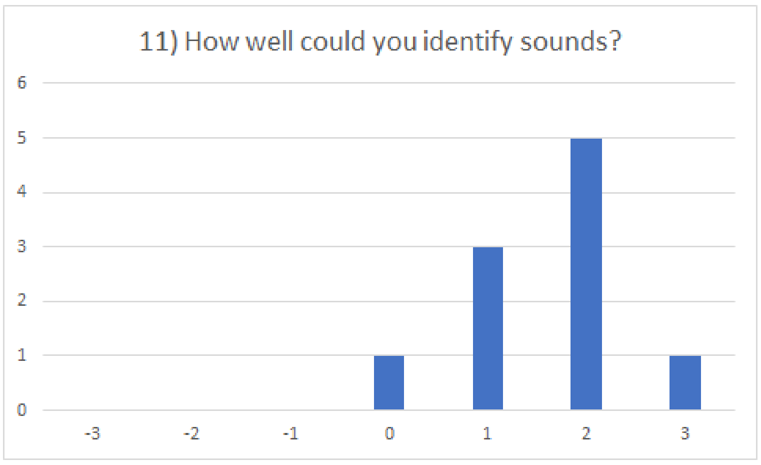


Figure 104: The participants' answers to Question 11

In the eleventh question, the participants were able to identify the sounds quite well.

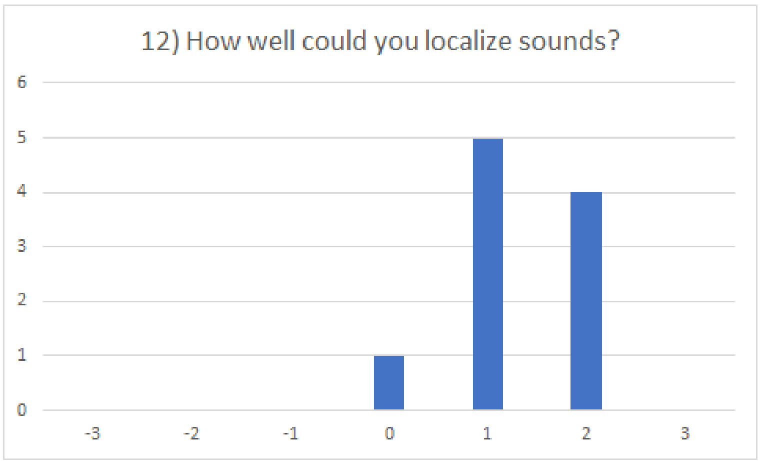


Figure 105: The participants' answers to Question 12

In the twelfth question, the participants were able to localize the sounds quite well.

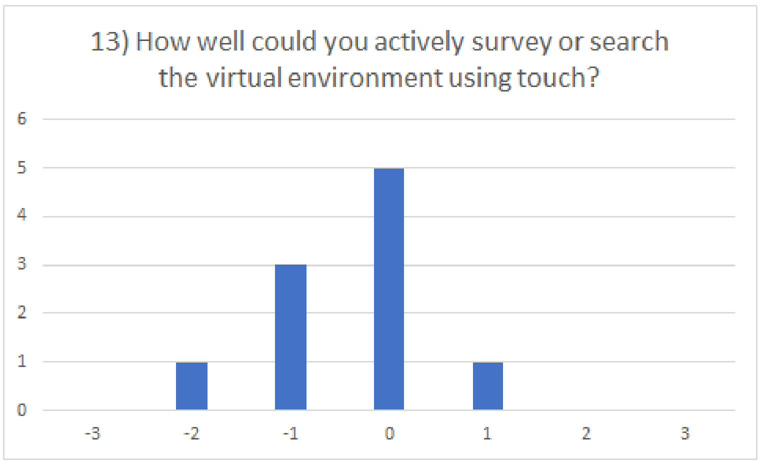


Figure 106: The participants' answers to Question 13

In the 13th question, the participants could partially survey the environment using touch, because only the objects that were part of the task were interactable.

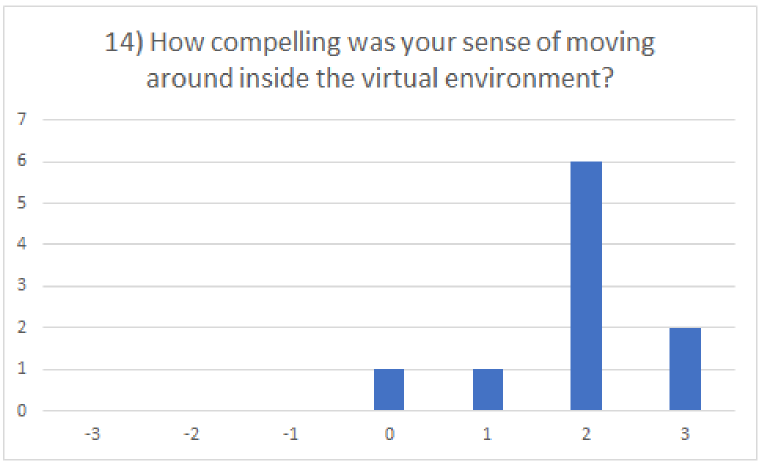


Figure 107: The participants' answers to Question 14

From the 14th question, it is concluded that the participants found the sense of moving in the virtual environment compelling.

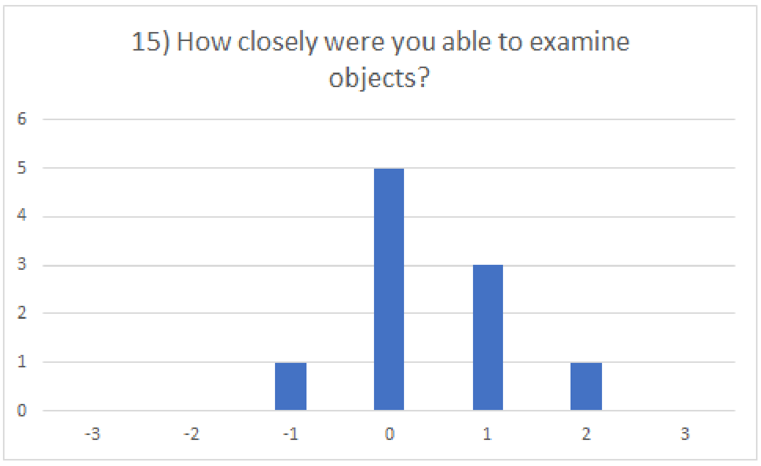


Figure 108: The participants' answers to Question 15

In the above question, the participants' answers were mainly neutral.

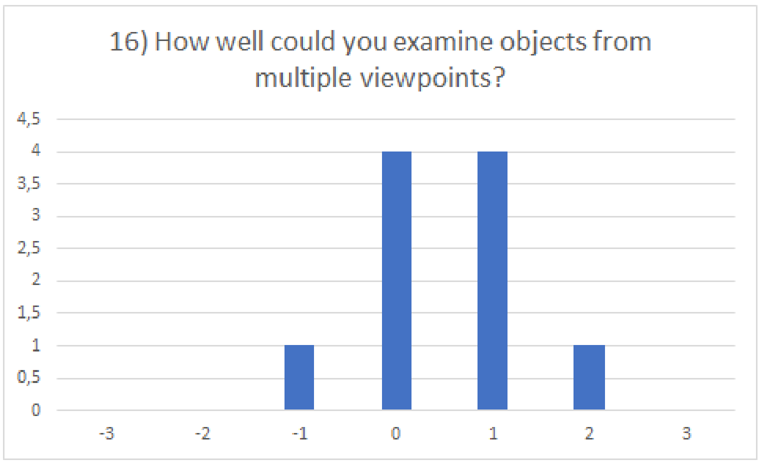


Figure 109: The participants' answers to Question 16

In the 16th question, the participants said that they could examine the objects normally.

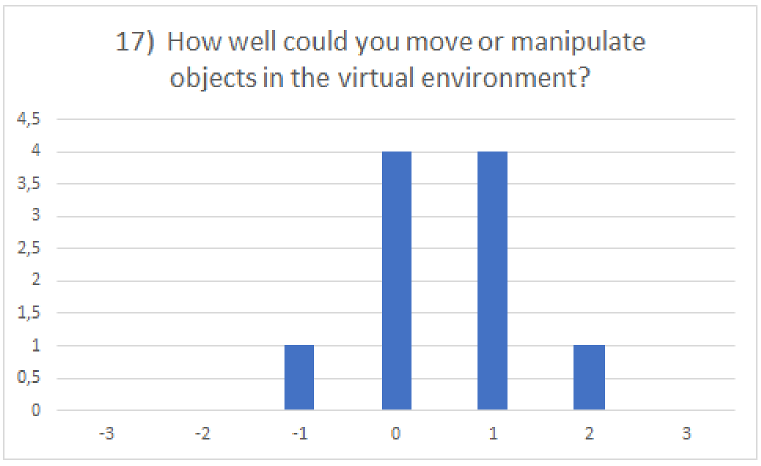


Figure 110: The participants' answers to Question 17

In the 17th question, they could move the objects quite well, except on some occasions, where the Leap Motion is inconsistent.

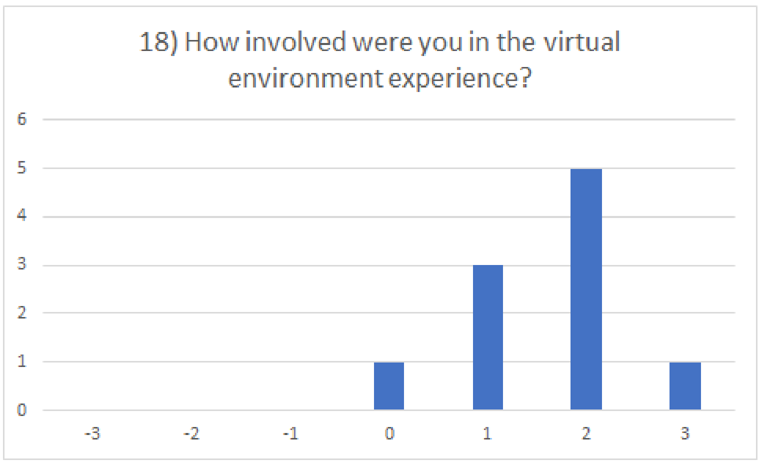


Figure 111: The participants' answers to Question 18

In the 18th question, the participants said that they were very involved in the game.

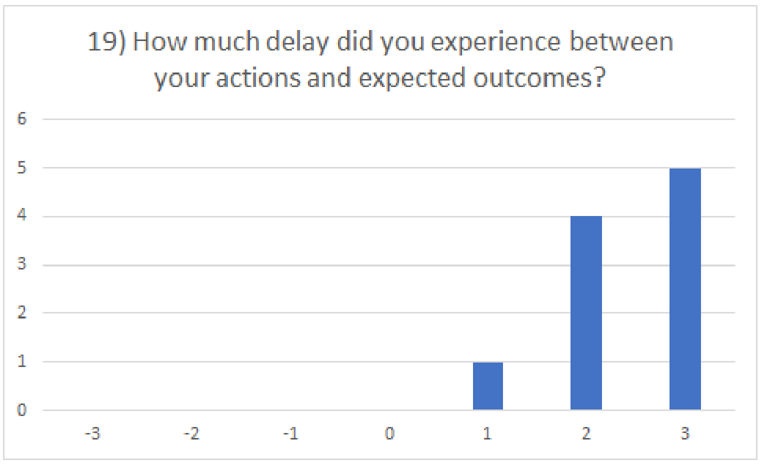


Figure 112: The participants' answers to Question 19

In this question, they noted that they did not feel any delay between their actions and the outcome.

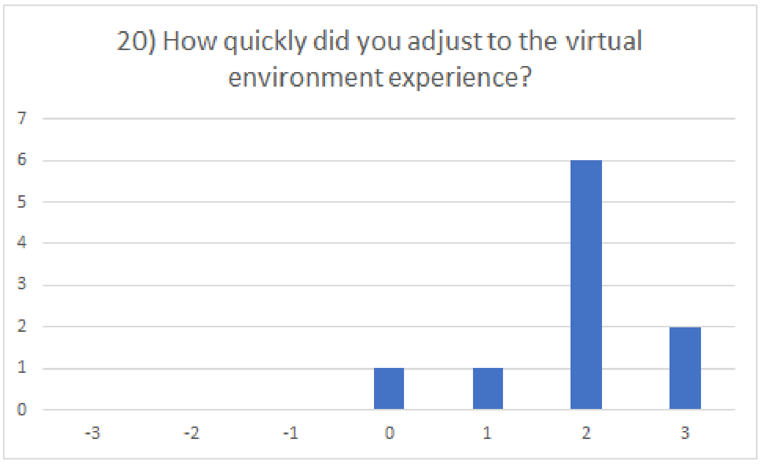


Figure 113: The participants' answers to Question 20

In the above question, it is clear that they were adjusted to the virtual environment quite well.

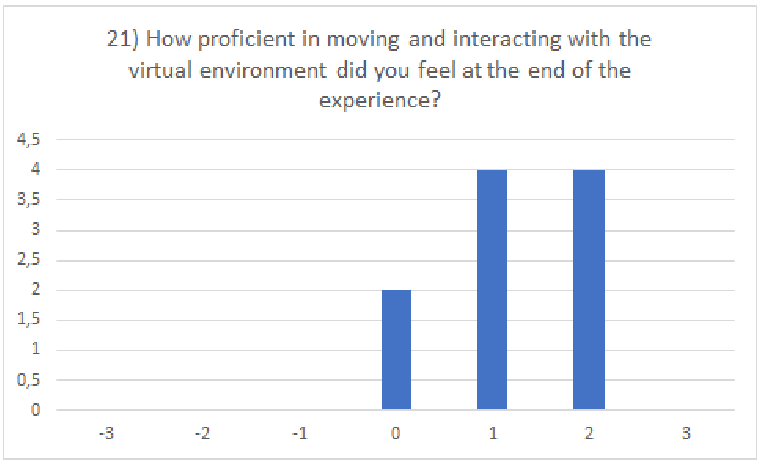


Figure 114: The participants' answers to Question 21

In the 21st question, the participants became quite good at moving and interacting with the environment very fast.

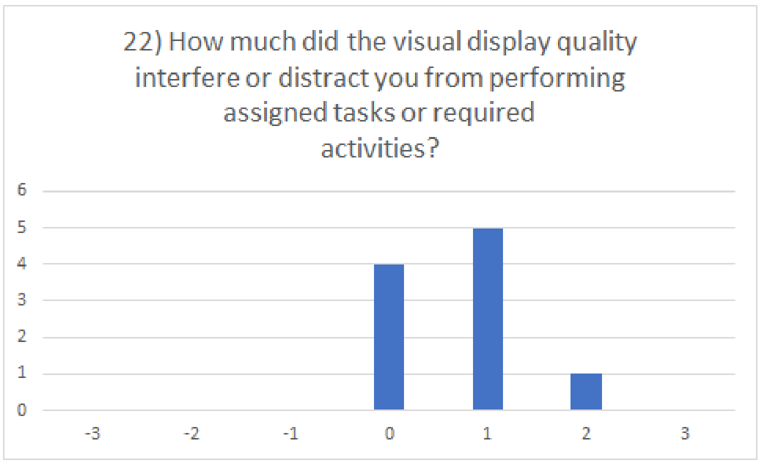


Figure 115: The participants' answers to Question 22

In the 22nd question, the participants seemed neutral.

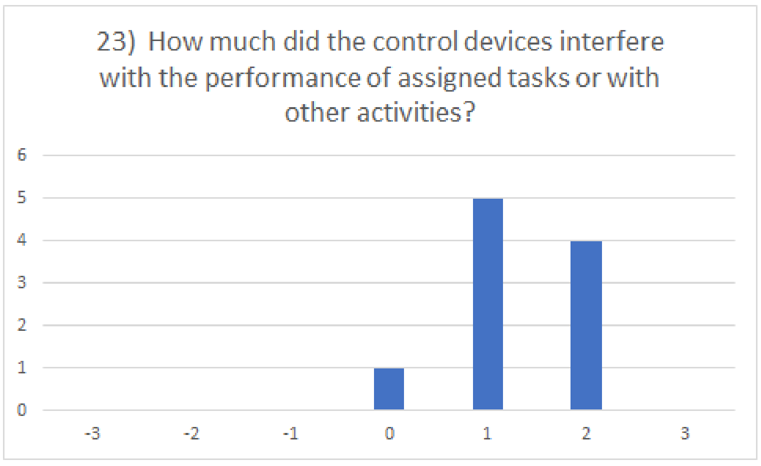


Figure 116: The participants' answers to Question 23

In the 23rd question, the participants answered that the haptic glove interfered with the performance of the tasks.

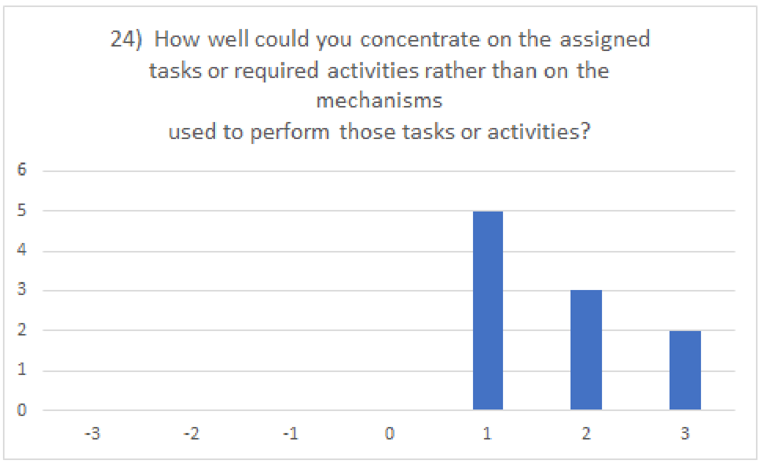


Figure 117: The participants' answers to Question 24

In this question, it was clear that the participants could focus on the tasks quite well.

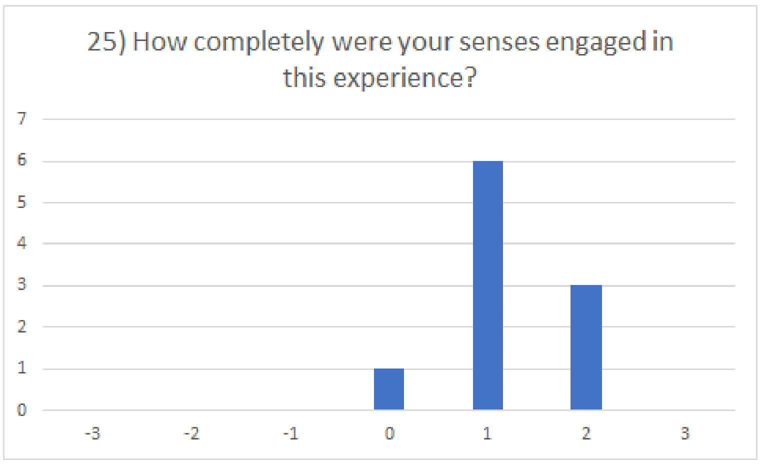


Figure 118: The participants' answers to Question 25

In the 25th question, the participants said that their senses were somehow engaged in this experience.

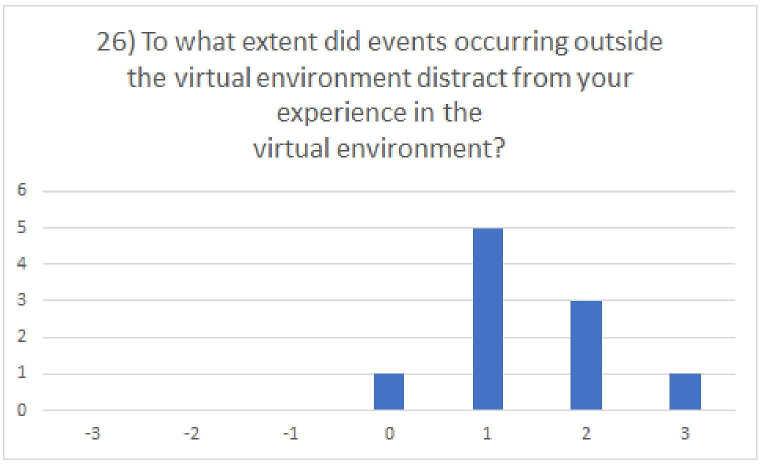


Figure 119: The participants' answers to Question 26

In this question, participants said that did not get distracted from outside events and were focused on the application.

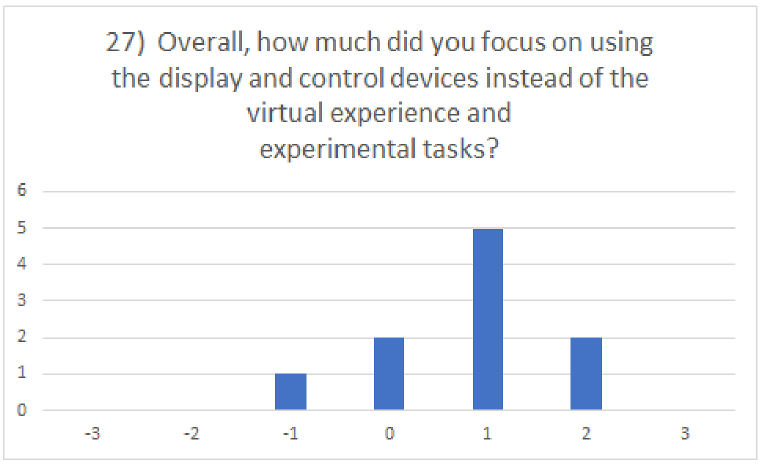


Figure 120: The participants' answers to Question 27

In this question, the participants said that they focused more on the experience than on the devices.

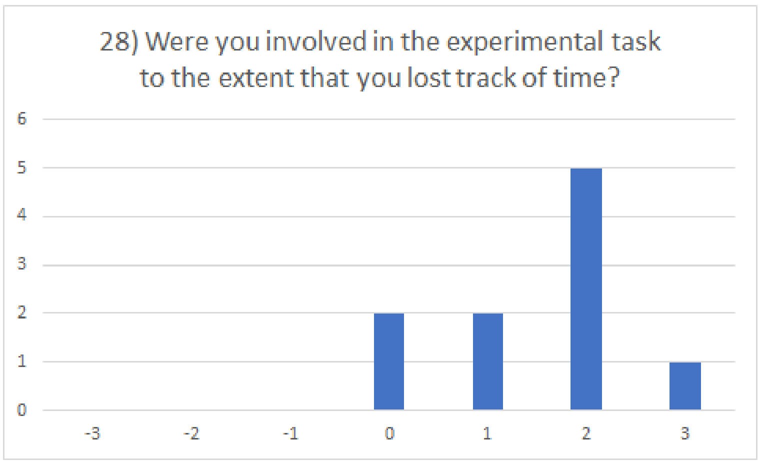


Figure 121: The participants' answers to Question 28

In this question, most of the participants said that they did not focus on time, while they were playing.

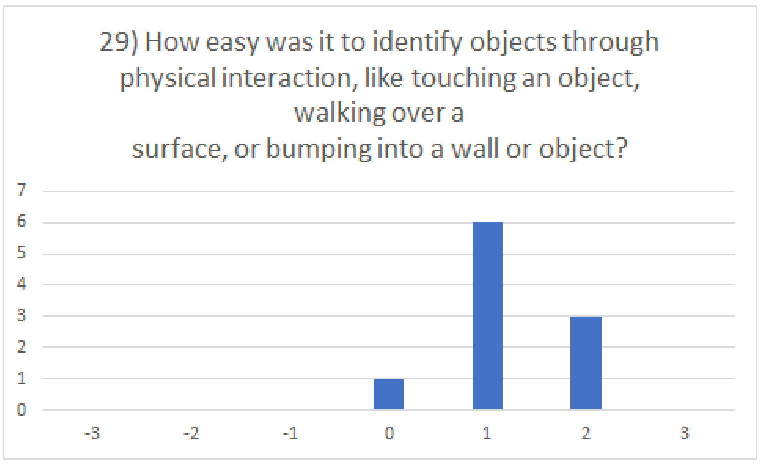


Figure 122: The participants' answers to Question 29

In the 29th question, the participants said that they could identify some objects very well and some others not so much, that why they selected the fairly good option.

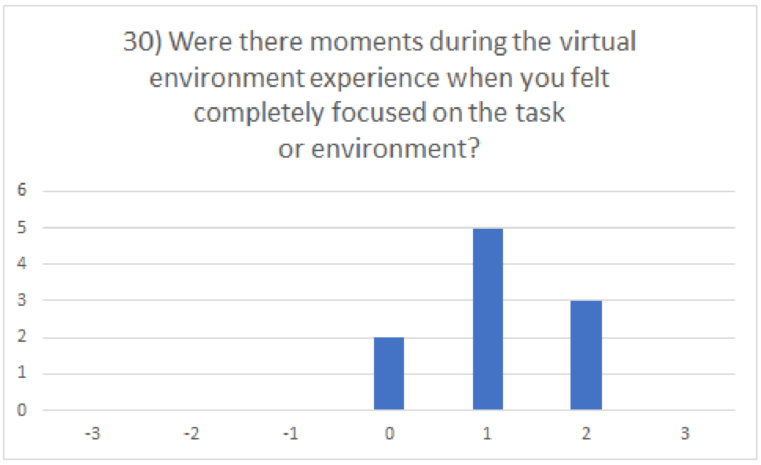


Figure 123: The participants' answers to Question 30

During the experience, some of the users felt quite focused on the task, while others not so much.

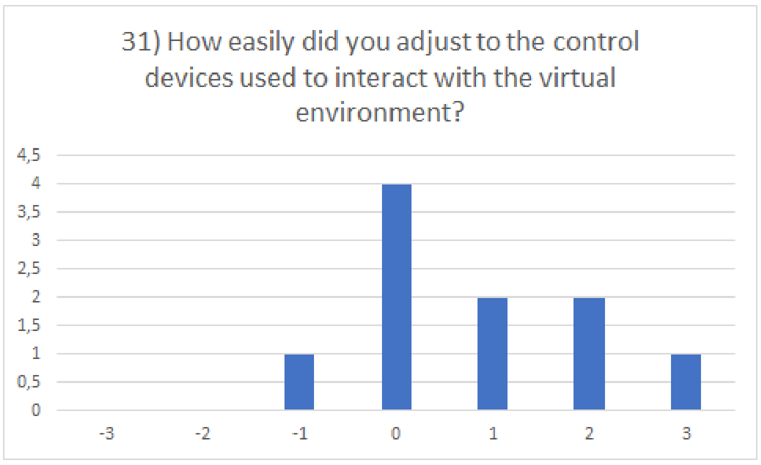


Figure 124: The participants' answers to Question 31

In this question, the answers were ranged from -1 to 3, because some participants were able to adjust to the control of the devices easily, while others not so much.

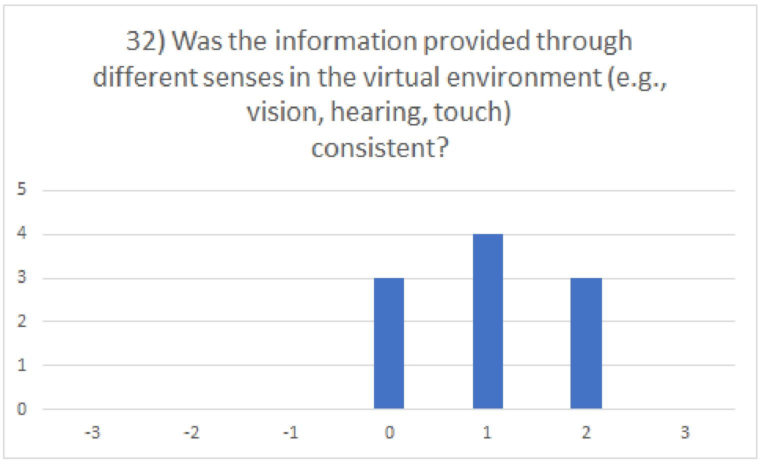


Figure 125: The participants' answers to Question 32

In the last question, the participants agreed that the information provided through different senses was quite consistent.

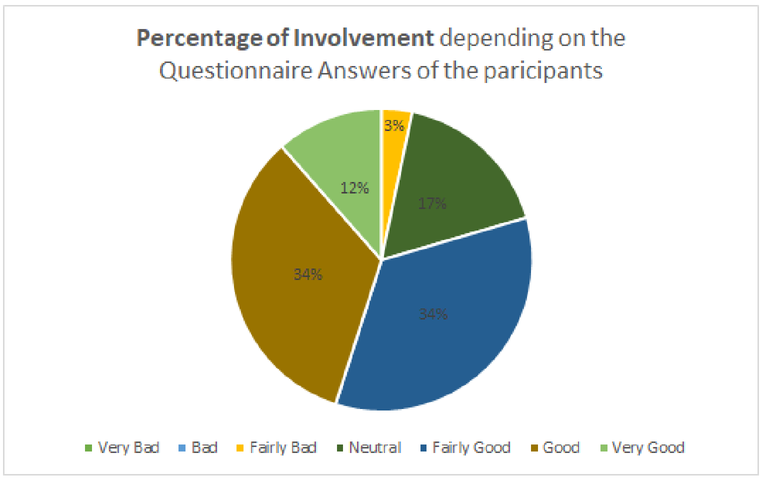


Figure 126: Percentage of Involvement

In the above chart, it is shown how involved were the participants in the environment. The data for this chart was the participants' answers to the questions that belonged to the involvement category of the questionnaire.

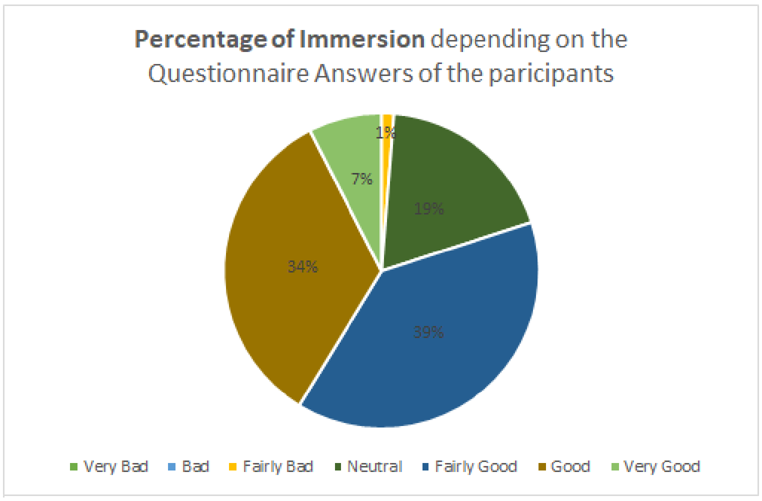


Figure 127: Percentage of Immersion

In the above chart, it is shown how immersed were the participants in the environment. The data for this chart was the participants' answers to the questions that belonged to the immersion category of the questionnaire.

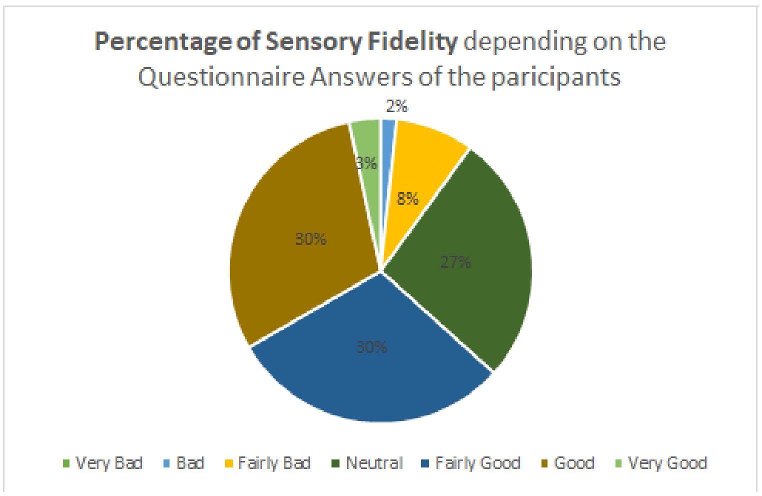


Figure 128: Percentage of Sensory Fidelity

In the above chart, it is shown how accurate the participants think the sensory feedback was. The data for this chart was the participants' answers to the questions that belonged to the sensory fidelity category of the questionnaire.

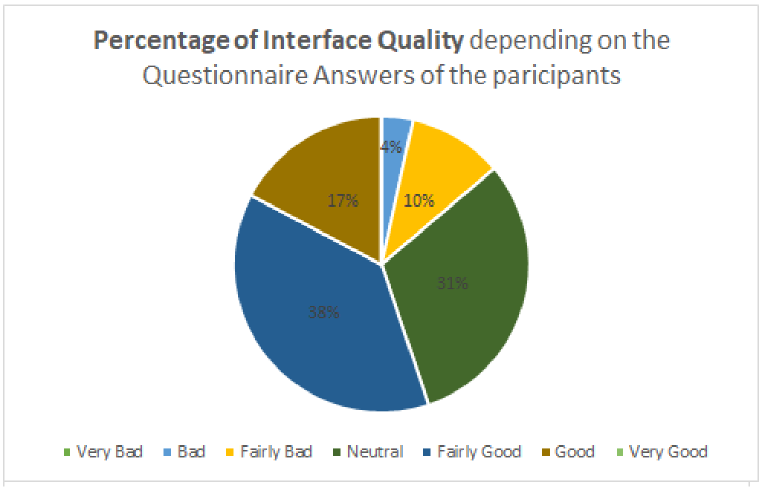


Figure 129: Percentage of Interface Quality

In the above chart, it is shown how good the participants think the quality of the interface was. The data for this chart was the participants' answers to the questions that belonged to the interface quality category of the questionnaire.

6 Conclusion and Future Work

6.1 Conclusion

Even though the design in this thesis successfully provided tactile and kinesthetic feedback to the user, the implementation of such a system came with some challenges. To begin with, it was important to study haptics and how they are applied to the user in virtual environments. This study helped this thesis to find the perfect options for each type of haptic feedback.

Specifically, for the tactile feedback, 15 coin vibration motors were used and the options considered, were either ERM or LRA. However, due to the limited budget and the circuit complexity of the LRAs, ERMs were picked. These motors were placed on the user's hand in 15 different points, one on every fingertip, one on the bottom part of every finger, five on the top part of the palm, and one on the side of the palm. Also, they are enabled independently and provide the user with three different levels of intensity.

For the kinesthetic feedback, at the start, it was necessary to become accustomed to 3D CAD software in order to create the CAD models for the exoskeleton. Initially, this process seemed easy, but the conceiving of the idea for the braking mechanism delayed it. Therefore, it took a fair amount of time before it was ready for 3D printing. Then, all the parts were connected and composed the exoskeleton. After this, the components to provide the kinesthetic feedback to the exoskeleton were 5 servo motors, one on each finger.

Furthermore, these motors required a system that could enable/disable them and the choice was the Arduino Mega Rev3. However, Arduino did not provide sufficient current for the vibration motors, thus a circuit with an N-type MOSFET, an EMI suppression capacitor, a Schottky diode, and a resistor was used for driving each one of them. This circuit at the start was made in a breadboard, but in order to reduce the system's weight and portability, a PCB was designed. This process was also difficult because it required extensive knowledge of electronic design automation(EDA) software. Additionally, Arduino does not have the required power to control 5 servo motors at the same time, thus a PWM/Servo driver was used. It was connected with the Arduino to receive signals and was powered through an external 5V power supply.

After the assembly of the whole system, a few more difficulties aroused. To begin with, due to the limited time and budget, the cable management could not be integrated into the exoskeleton and the vibration motors cables were very fragile and some sudden moves could destroy the connections. Also, the interface is heavier than it was thought, thus an extended usage time was exhausting. Finally, due to the extended number of components on top of the hand, Leap Motion sometimes was inefficient to track it correctly.

6.2 Future Work

The limitations that aroused during the process of designing and using the haptic interface, can be surpassed with some future improvements. These improvements could be the following:

- Using a better filament like ASA could increase the durability of the exoskeleton, making it less likely to break under pressure.
- Redesigning the exoskeleton or designing a new way to provide kinesthetic feedback, in order to decrease its size and weight. A new way to provide kinesthetic feedback could be through strings that will be pulled when a collision in the virtual environment occurs.

- Replacing the Leap Motion with a new system, that will be added on the haptic interface, so that the tracking of the hand, its representation in the 3D environment, and the haptic feedback are all provided from the same device.
- Adjusting the stiffness of the servo motors in order to provide feedback on the softness or hardness of the objects. Specifically, if the user touches a flexible object, his fingers could feel the object's deformation, but if he touches a rigid body, his finger movement will be blocked completely.
- Adding a moving platform at the fingertip to provide information for the curvature and the surface of the virtual object. This could be accomplished by creating 3D printed legs that with the help of a motor would move the platform.
- Making it wireless so that the user will not be constrained by the cables.

7 References and Bibliography

References

[1] Eccentric Rotating Mass. Precision Microdrives. https://www.precisionmicrodrives.com/content/ab-004-understanding-erm-vibration-motor-characteristics/

[2] Linear Resonant Actuator. Precision Microdrives. https://www.precisionmicrodrives.com/vibration-motors/linear-resonant-actuators-lras/

[3] Leap Motion. (2016). Ultraleap. https://www.ultraleap.com/datasheets/Leap_Motion_Controller_Datasheet.pdf

[4] Oculus Touch. (2020). Wikipedia. https://en.wikipedia.org/wiki/Oculus_Touch

[5] Inertial Measurement Unit. (2020). Wikipedia. https://en.wikipedia.org/wiki/Inertial_measurement_unit

[6] C. Pacchierotti, S. Sinclair, M. Solazzi, A. Frisoli, V. Hayward and D. Prattichizzo, "Wearable Haptic Systems for the Fingertip and the Hand: Taxonomy, Review, and Perspectives" in IEEE Transactions on Haptics, vol. 10, no. 4, pp. 580-600, 1 Oct.-Dec. 2017.

[7] Haptx. (2018). Haptx. https://haptx.com/

[8] FundamentalVr. (2018). FundamentalVr. https://www.fundamentalvr.com

[9] LinkDyn. (2018). LinkDyn Robotics. https://linkdyn.com

[10] Berger, Christopher and Gonzalez-Franco, Mar and Ofek, Eyal and Hinckley, Ken. (2018). The uncanny valley of haptics. Science Robotics. 3. eaar7010. 10.1126/scirobotics.aar7010.

[11] Busho, M. (2018). A Haptic Glove Prototype for Tactile Feedback in 3D Interactive Applications.

[12] Effraimidis, M. (2019). Wireless Embedded System on a Glove for Hand Motion Capture and Tactile Feedback in 3D Environments

[13] Know Your Stratasys FDM Materials: ABS, ASA, and PLA. Computer Aided Technology. https://www.cati.com/blog/2018/07/know-stratasys-fdm-materials-abs-asa-pla/

[14] Solutions for ERM and LRA Actuators. Texas Instruments. https://www.ti.com/lit/pdf/sszb151

[15] Bipolar Junction Transistor. (2020). Wikipedia. https://en.wikipedia.org/wiki/Bipolar_junction_transistor

[16] MOSFET. (2020). Wikipedia. https://en.wikipedia.org/wiki/MOSFET

[17] Witmer, B. G., Jerome, C. J., & Singer, M. J. (2005). The factor structure of the presence questionnaire. Presence: Teleoperators and Virtual Environments, 14, 298-312.