uc3m | Universidad Carlos III de Madrid

University Degree in Computer Engineering

2022-2023

Bachelor Thesis

"Development of a Virtual Reality Experience with Wireless Instrument Connectivity"

Juan Carlos Cebrián Peñuela

Tutor: Alejandro Rey Lopez

Leganes, Madrid, September 2023

Abstract

In recent years, there has been a considerable increase in interest in Virtual Reality and Mixed Reality applications. Such immersive experiences can be a great way for the user to learn or improve skills. With headsets being more affordable, virtual reality usage and demand for applications has increased.

This project uses Virtual Reality to create a musical virtual reality experience using Unity 3D where the user can use an external device to play music in the experience. This device will be custom built for the experience, implementing instrument communication protocols to communicate wirelessly with the headset.

Said experience leverages the immersion and visual feedback of virtual reality to help the user learn to play music with the use of hints that appear over a representation in 3D of the device.

This document describes the development process of both the Virtual Experience and the physical device, detailing the different phases involved such as the design, implementation and testing of the systems.

Keywords: Unity 3D, Virtual Reality, VR, Wireless protocols, MIDI, Bluetooth, Bluetooth Low Energy, ESP32, Arduino.

Index

Table Index

Figure Index

CHAPTER 1

1 Introduction

In this chapter, a brief introduction to the project is found, declaring the motivation behind the project, its objectives and finally the structure of this document.

1.1 Motivation

In recent years, especially after the COVID-19 pandemic, the Virtual Reality space has been experiencing a great boom with a great surge of experiences, users and more importantly, it has brought the attention of big enterprises such as Meta (formerly Facebook) [1] and Apple [2].

Virtual Reality can have many different uses, but one notable field is for education and teaching new skills. Virtual Reality can be used to enable educational experiences thanks to the immersion and interactivity that the headset provides [3]. One of such skills is the ability to play instruments. Using Virtual Reality, a person joining the music world can have interactive piano lessons [4] or play games that makes the user follow a certain rhythm, allowing the user to gradually increment their skill [5].

Virtual Reality experiences might allow the use of additional hardware to assist or improve the experience. For musical games, usually electronic instruments can be used for connection with the experience. But usually, they are at a steep price for a newcomer to the hobby with prices starting at around 200€, without considering the price of the VR headset.

Given the previous stated points, the aim of this project is to design a Virtual Experience where a user can learn or improve their musical skills. Additionally, an external device to the Virtual Reality headset will be designed and build, to be used as an instrument, that will be able to connect and be used to play music through the experience. To achieve this communication between headset and instrument, the different instrument communication protocols such as MIDI and Open Sound Control will be studied and compared with the intention to implement one of them in the system, with a focus on wireless communication.

The motivation for this project is to demonstrate the expandability that virtual experiences can have thanks to peripherals and wireless connectivity.

The development of this project is useful to improve the understanding of instrument control protocols, wireless technologies and the creation of virtual experiences, designing game mechanics and 3D modelling.

1.2 Objectives

This project can be divided in three main objectives.

- The study and analysis of the different instrument communication protocols such as MIDI and Open Sound Control. The comparation of the different wireless technologies like Bluetooth, Bluetooth Low Energy or Wi-Fi as transport layer for the chosen communication protocol.

    - The chosen protocol must be able to send signals representing musical notes and analogic values from sliders and potentiometers.
    - The wireless protocol should be easy to implement, flexible and with no big latency.

- Design and construction of a physical peripheral using the chosen communication protocol and wireless technology.

    - The device should be able to send multiple notes at a time.
    - The device can send analog values from different potentiometers that can be used as pitch or volume for example.
    - The device must be able to be tracked inside Virtual Space.

- Desing and implementation of an interactive virtual experience that uses said documentation protocol. The main objective of the virtual experience is to play the sounds from the physical device and show hints telling the player what note to play.

    - The experience shows a virtual representation of the instrument, that can be tracked inside virtual space when moved.
    - The virtual experience can retrieve musical note data from a music file and show hints on the device representation of what note to play.
    - The players hands are shown and tracked in real time in 3D space.

1.3 Document Structure

The document is divided into chapters containing the different parts of the document. Here you can find a list of chapters and a brief description of them.

- **Chapter 1: Introduction.** In this chapter, a brief introduction to the project is found, declaring the motivation behind the project, its objectives and finally the structure of this document.
- **Chapter 2: State of the Art.** During this chapter, an in-depth discussion on the current available VR products and their relationship with the music world is performed. The different protocols for instrument communication are also studied, comparing the characteristics of each one and the ease of implementation.
- **Chapter 3: Analysis.** In this section, the analysis of the project will be detailed, explaining the user and software requirements, the use cases and traceability matrix. These steps of the analysis are fundamental for the design and development of the application.
- **Chapter 4: Design and Implementation.** This section specifies all design aspects of the instrument device and the VR experience. Later, detailing how it is implemented and the result of the implementation.
- **Chapter 5: Evaluation.** This chapter will explore the evaluation of the implementation, making sure it is conforming with respect to the specified system requirements. To achieve this, a test procedure will be defined and carried out, making sure each requirement is met.
- **Chapter 6: Project Planning and Budget.** This chapter reflects on the planification process for the project, detailing the different phases of the project and finally, discussing the project budget.

- **Chapter 7: Legal Framework.** The next section, the different legal concerns, regulations that might apply and licensing of third-party software will be discussed.
- **Chapter 8: Socio-economic Environment.** In this chapter the reader can find a description of the impact that the project can have and its usefulness on the current social and economic environments.
- **Chapter 9: Conclusion.** Finally, in this chapter a retrospective of the project is done, reviewing the fulfilled objectives and detailing future lines of work.

CHAPTER 2

2 State of the Art

During this chapter, an in-depth discussion on the current available VR products and their relationship with the music world is performed. The different protocols for instrument communication are also studied, comparing the characteristics of each one and the ease of implementation.

2.1 Virtual Reality and the World of Music

Thanks to the immersion that Virtual Reality provides, since its inception, experiences that exploits the sensorial and motor engagement of the user to bring them musical experience. One of these experiences, and perhaps the best known, originated in 2016 is Beat Saber as a simple demo of a rhythm game for Virtual Reality [6]. Beat Saber is a rhythm game, where the user must cut boxes that comes to the user at the beat of the song. The huge success of its release caused the creation of several other rhythm games such as Pisto Whip [7] and Synth Riders [8], all of them being rhythm games with high intensity movement.

Educational games also exist, Smash Drums [9] allows the user to play a virtual set of drums in the rhythm of included songs, just using the controller. It shows a representation of what drum the user must hit with the controllers. Similarly, an application that allows the user to play an air guitar exists [10].

Apart from these experiences, Virtual Reality Musical Instruments (VRMI) exist as instruments built within Virtual Reality. VRMI originated in 1992 as a demo of VPL's EyePhone and DataGlove virtual reality equipment. This demo showed a person playing fictional instruments by moving their hands and doing gestures [11]. Virtual Reality Musical Instruments are digital instruments that do not necessarily have a physical form. This virtual instrument doesn't necessarily copy real life instruments but instead tries to implement new instruments making use of the Virtual Reality and user capabilities [12].

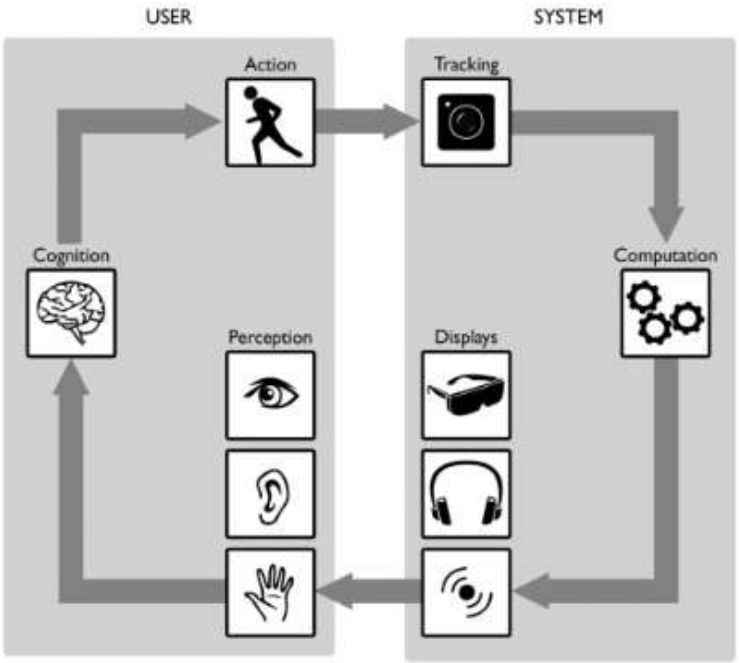


Figure 1. Interaction modalities between VRMI systems and Users [12].

Additionally, there exists experiences that makes use of instrument control protocols to connect to instruments, so the user can interact with them in virtual space. One example of such application is PianoVision [13] which allows the user to improve their piano skills by showing a mixed reality overlay on the piano with indication of notes to play and also connects using the wired MIDI communication in order to reproduce sound and confirm that the user is pressing the correct keys.

In a similar fashion, experiences where the Headset controller is configured as a MIDI controller for external synthesizers or applications also exists, allowing the user to configure VRMIs and play them, sending the information over MIDI to the synthesizer such as the application Modulia Studio [14].

2.2 Instrument Communication Protocols

There exists a broad variety of communication protocols designed for instruments and music. In this section a study and comparation between MIDI, ZIPI and Open Sound Control (OSC) is detailed.

2.2.1 MIDI – Musical Instrument Digital Interface

MIDI was created as an attempt to standardize the synchronization of musical instruments that were using different proprietary communication protocols as they were manufactured by a variety of companies [15].

The MIDI standard was first proposed in 1981 and subsequently discussed and modified by representatives of the most popular instrument builders [16]. The first version of the MIDI protocol would be launched on 1983 as MIDI 1.0 [15].

MIDI is a real-time protocol for transmitting information of playback of a piece of music. MIDI defines a set of messages that are sent in real-time [17] when the target synthesizer must play the sounds.

A MIDI message consists in a status byte and two data bytes. The status byte represents the designated channel of the message in its four low order bits and the message type on the three remaining bits. Message types are divided into system messages and channel messages.

Channel messages are specific for an individual channel and are used to instruct the instrument to play sound, toggle notes and alter the configuration of the channel modifying the sound.

With channel messages, a MIDI device can address up to 16 midi channels with 128 notes.

System messages are not channel specific, instead are listened by the whole MIDI device and are used for device configuration, clock synchronization and control of MIDI sequences.

A MIDI instrument can be connected directly to a sequencer or computer. If more devices need to be connected, they must be installed in series, where each device can broadcast to a MIDI channel.

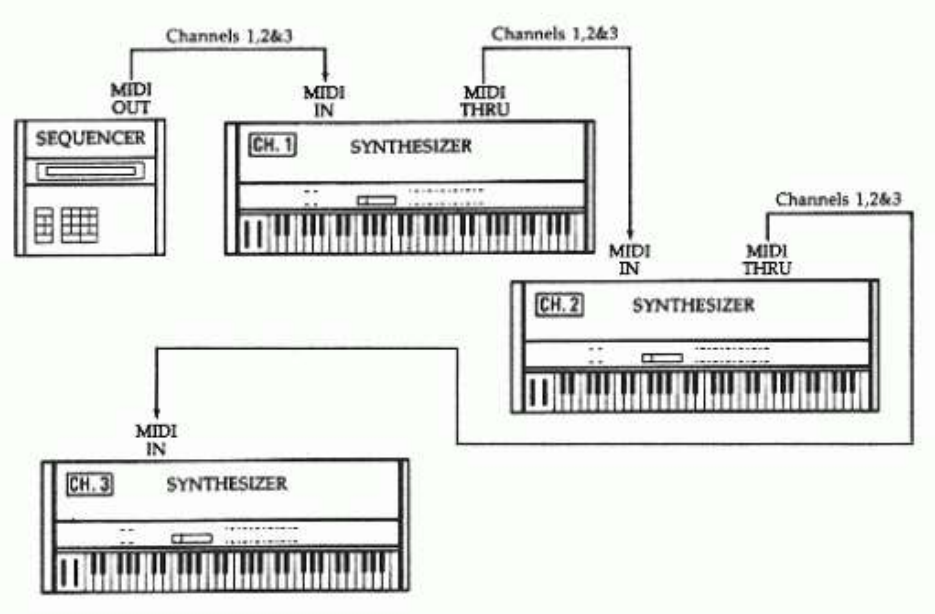


Figure 2. Midi Devices connected in series [18]

Midi is one of the most popular instrument communication protocols used in Digital Instruments. Being this popular allows inter-compatibility between instruments and software, allowing users of the ecosystem to have a better compatibility with other and older hardware.

Wireless MIDI

MIDI can be embedded in different transport layers. There are two notable MIDI wireless implementations, RPT-MIDI (AppleMIDI) [19] and MIDI over Bluetooth low energy [20].

RPT MIDI

RPT-MIDI was created to send MIDI messages over the Real-Time Transport Protocol (RPT) using Ethernet or Wi-Fi networks. This protocol is commonly used for connection of MIDI instruments with Apple devices such as iPads but can also be used in non-apple devices such as Arduino [21], Windows [22], and Android [23].

MIDI Bluetooth Low Energy

The specification of MIDI for Bluetooth Low Energy is supported by most devices with Bluetooth connectivity. Bluetooth Low Energy is a Bluetooth specification that reduces overhead and helps save battery life for devices that don't stream data continuously (like MIDI controllers). MIDI BLE has seen high usage both in instruments and in mobile devices [24], with a lot of application that allow smartphones connect to instruments and receive notes.

Implementation exists for most of the common platforms such as Android [25] and open hardware like as Arduino [26] and Raspberry Pi [27] through open-source libraries.

2.2.2 ZIPI – Zeta Instruments Processor Interface

The Zeta Instruments Processor Interface was originally published in 1994 as a next generation alternative to MIDI by Zeta Instruments and the University of California, Berkeley. ZIPI is designed to run as a start network where all devices are connected directly to a shared switch, which prevents daisy chaining of devices, improving overall speed [28].

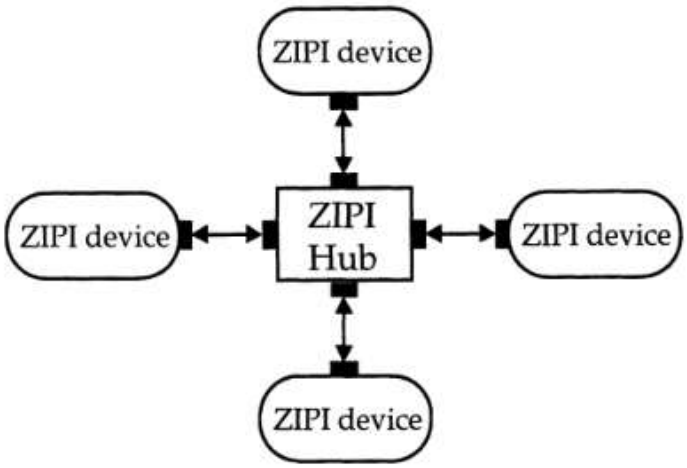


Figure 3. ZIPI devices connected to hub [28].

It has a network speed of up to 250 kBaud while having a small load on the processor, as it only interrupts the processor when receiving a message. Devices connected to a network can send frames directly to any other device on the ring or to all the networks. The network can have up to 253 devices connected.

ZIPI does not differentiate channels, instead it divides the note addresses into 63 families of 127 instruments each, with 127 notes per instrument. This allows great granularity and personalization when playing sounds.

ZIPI defines a variety of synthesis messages and controller messages. Some of these messages come directly from the MIDI specification, but with different names that aim to be more intuitive and less ambiguous.

After its initial publication in 1994, ZIPI did not see much success and is rarely supported by any device.

2.2.3 OSC – Open Sound Control

Open Sound Control is another synthesizer connection protocol for electronic instruments, computers and different multimedia devices that is usually used for music synthesis and show control. It was published in 2002.

As an alternative to MIDI, Open Sound Control aims to provide better resolution and broader parameter space. OSC uses a dynamic and open symbolic nomenclature to address parameters, in the style of URI [29]. OSC can be transmitted over the internet, over local subnetworks through UDP or Ethernet, or over USB serial encapsulated using the SLIP protocol.

Messages in the OSC protocol consists of an address pattern like URIs, a data type tag and the arguments of the message. An example of an OSC message could be:

/instrument/6/pitch ,fi, 0x3f5c28f6

In this message, the value 0x3f5c28f6 as a float-32 (fi tag) is assigned to the pitch of instrument six.

Open Sound Control has achieved extensive usage since its creation and have been built into several commercial products, although mostly experimental. It is also implemented in various synthesizer software.

Wireless OSC

As OSC natively supports UDP/IP transport, it is easy to use over Wi-Fi for communication between instruments and software on Windows, Android, and Arduino [30] devices. As it uses UDP, OSC over Wi-Fi does not include error-checking [31] for messages.

2.3 Comparison and Conclusion

Of the three protocols discussed, Open Sound Control and MIDI are the two most notable protocols. Both have wide acceptability and have seen implementation in various devices. OSC stands out with its addressable URI style nomenclature, that allows for easy differentiation between components. MIDI has the advantage of been really stablished with lots of documentation and compatible devices.

For this project, due to the ease of implementation, amount of documentation and high compatibility, MIDI will be used as a communication protocol. Specially, MIDI over Bluetooth Low Energy due to widespread availability and compatibility with multiple devices.

2.4 Selected technology

In this section, after choosing MIDI Bluetooth Low Energy as the instrument communication protocol, is included the definition and comparation of the different technologies needed for each of the parts of the project. But first, the MIDI Bluetooth low Energy protocol will be defined in depth, as all remaining technologies depend on the implementation of this protocol.

2.4.1 MIDI Over Bluetooth Low Energy

As said previously, MIDI is the most common communication standard for instrument communication and synchronization. The MIDI specification [32] allows to exchange musical information such as music notes, program changes, expression control, etcetera.

MIDI uses messages of multiple bytes for communication. This message consists of a Status byte followed by one or two data bytes. The status byte is used for identification of the type of message.

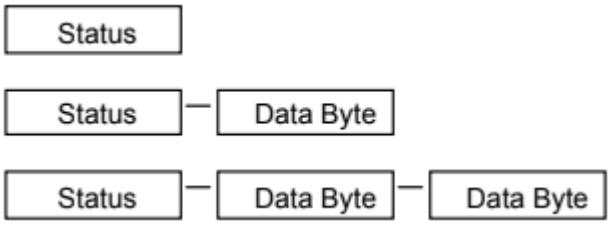


Figure 4. Structure of a MIDI message [32].

MIDI devices can implement a receiver and a transmitter. The receiver is useful for accepting MIDI messages and executing them. A transmitter broadcasts the MIDI messages produced by the instrument. A MIDI device might contain both elements or only a transmitter or receiver so that the device only assumes one of the two functionalities.

Midi differentiates between two main groups of MIDI messages.

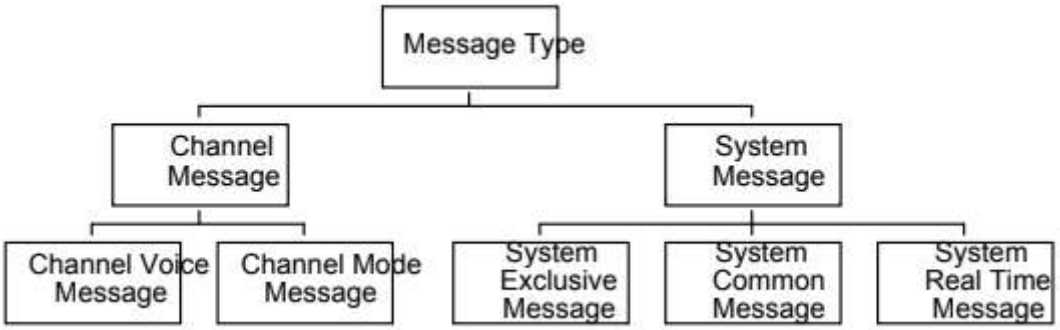


Figure 5. Types of MIDI messages [32].

System Messages

System messages are transmitted to the whole MIDI system, this means, they are not channel specific. Usually used for timing synchronization and setup information. System Messages can be divided in three main groups:

System Common Messages

Common messages that are listened from all devices in the MIDI network regardless of channel.

Status Byte	System Message	Number of Data bytes
F1	Midi Timing Code	1
F2	Song Position Pointer	2
F3	Song Select	1
F6	Tune Request	None

Table 1. System Common Messages [32].

System Exclusive Messages

Exclusive messages may contain any number of data bytes, until an End of Exclusive (EOX) is found. They are usually used for devices identification or manufacturer identification.

Figure 6. Structure of System Exclusive Messages [32].

System Real Time Messages

Real Time messages are used for the synchronization of system clocks. They don't contain any data bytes.

Status Byte	System Message
F8	Timing Clock
FA	Start Sequence
FB	Continue Sequence
FC	Stop Sequence
FE	Active Sensing
FF	System Reset

Table 2. System Real Time messages [32].

Channel Messages

Channel messages are only broadcasted to a particular channel instead to the whole MIDI system. It uses the last four bits of the status byte to designate one of the sixteen MIDI channels. Channel messages can be divided into two groups.

Channel voice messages

Voice messages are used to set notes on and off and assign or modify the sound of notes.

Status Byte	Voice Message	First Data Byte	Second Data Byte
8n	Note Off	Note Number	Note Off Velocity
9n	Note On	Note Number	Note On Velocity
An	Polyphonic Key Pressure	Note Number	Amount of Pressure
Bn	Control Change	Controller Number	Controller Value
Cn	Program Change	Program Number	None
Dn	Channel Pressure	Pressure Value	None
En	Pitch Bend	Most Significant Bit	Least Significant Bit

Table 3. Channel Voice Messages [32].

The letter “n” represents the channel number.

Channel mode messages

Channel mode messages determine how the device will interpret voice messages. They are a special case of the Control Change message (Bn), with the same status byte but with first data byte ranging from 79 to 7F (121 to 127).

First Data Byte	Description	Second Data Byte
79	Reset all controllers	None
7A	Local control	0 -> On 127 -> Off
7B	All notes off	None
7C	Omni mode off	None
7D	Omni mode on	None
7E	Mono mode on (Poly off)	Channel number
7F	Poly mode on (Mono off)	None

Table 4. Channel Mode Messages [32]

MIDI Bluetooth Low Energy

The MIDI Bluetooth Low Energy specification allows the implementation of the MIDI standard encapsulated in Bluetooth. It defines a method for encoding and decoding MIDI messages. It prepends timestamps of millisecond resolution to the messages as the messages are sent in batch at the configured Bluetooth frequency. Theses timestamps allows for a comparable jitter to the USB MIDI specification [20], although it can suffer unexpected delays due to wireless interference.

BLE Packets in the MIDI Specification

Bluetooth packets in the MIDI specification consists in various parts. First, a header byte must be present at the beginning of the packet. It contains two configuration bits and the six most significant bits of the timestamp of the first MIDI message.

Header Byte	
bit 7	Set to 1.
bit 6	Set to 0. (Reserved for future use)
bits 5-0	*timestampHigh*:Most significant 6 bits of timestamp information.

Figure 7. Bluetooth MIDI header byte [20].

Next, a timestamp byte is included that contains the seven least significant bits of timestamp of the first MIDI message.

Timestamp Byte	
bit 7	Set to 1.
bits 6-0	*timestampLow*: Least Significant 7 bits of timestamp information.

Figure 8. Bluetooth MIDI timestamp [20].

After the timestamp byte, the packet contains the MIDI messages. They are encoded in two ways depending on if the message is a Running Status Message or if it is any other MIDI message.

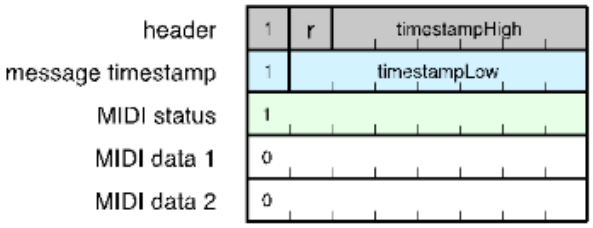


Figure 10. Packet with standard MIDI Message [20].

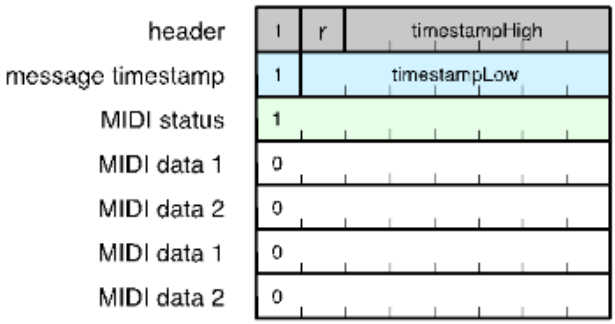


Figure 9. Packet with Running Status Message [20].

These two message encodings can be mixed in the same BLE packet:

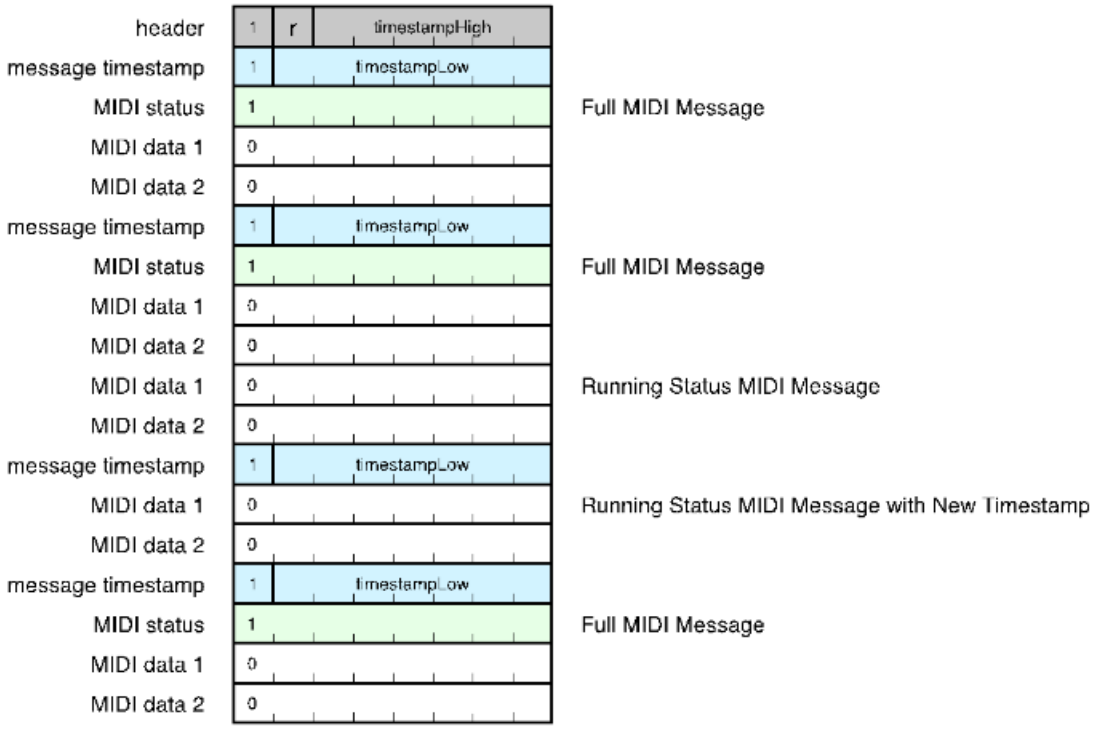


Figure 11. Packet with multiple MIDI messages [20].

Finally, System Exclusive messages can expand various packets, until the end EOX byte is read.

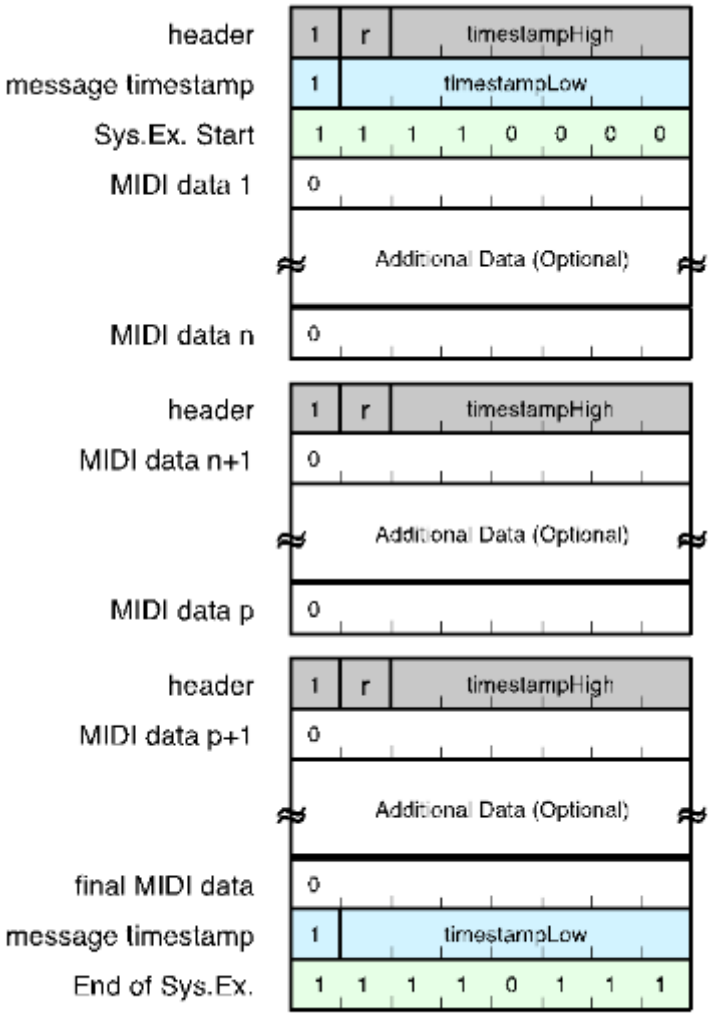


Figure 12. System Exclusive Message across multiple packets [20].

Timestamps

The BLE MIDI specification requires the use of timestamps for each MIDI message sent in a Bluetooth package. These timestamps are of millisecond resolution and are encoded in 13-bit values [20].

This timestamp is divided in two parts, a timestampHigh that it is included in the header of the package contains the 6 most significant bits. The second part is included as a timestamp byte in the package that contains the seven lower bits of the timestamp as timestampLow.

Timestamps must be sent by the transmitter sequentially; they are in the transmitter clock domain.

2.4.2 Instrument Hardware Microcontroller

One part of this project is to create a musical device that implements the MIDI Bluetooth protocol to communicate the notes being played. In this section, some of the different microcontrollers available on the market that can be used for this purpose are detailed.

Arduino Nano 33 BLE

The Arduino Nano 33 BLE is a compact microcontroller highly used in both maker and professional spaces. It has 1MB of CPU flash memory and includes a Bluetooth 5 Low Energy module with internal antenna. This Arduino board has 14 digital Input/Output pins of which 5 of them are PWM pins. It also has 8 analog input pins. You can program it using the Arduino language or Micro-Python. This Arduino board cost around 30€.

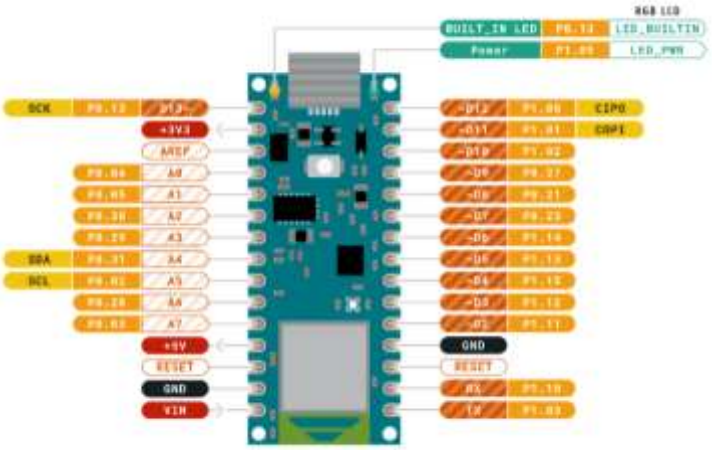


Figure 13. Arduino Nano 33 BLE pinout

Raspberry Pi Zero 2 W

The Raspberry Pi Zero W is the wireless version of the Raspberry Pi Zero family. It allows the execution of custom operating systems (for example Linux). It provides a 40-pin header with 27 GPIO pins, USB and HDMI connectivity and a wireless module that supports Wi-Fi 802.11 b/g/n and Bluetooth 4.2 with BLE capabilities. As it runs custom operating systems, it can run any executable compiled for the board regardless of source language.

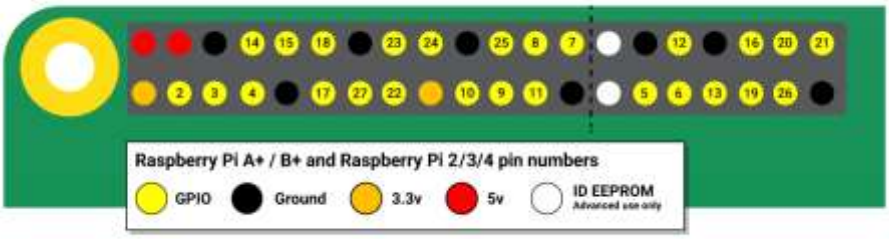


Figure 14. Raspberry Pi 40-pin header pinout.

The Raspberry Pi Zero 2 W cost is around 19€.

ESP32

The ESP32 is a low power System on a Chip (SoC) microcontroller. It provides lots of functionalities at a low price. It comes with a wireless module that allows Wi-Fi 802.11 b/g/n and Bluetooth 4.2 with BLE capabilities. ESP32 also allows for 34 programable GPIO pins, 2 eight bits digital to analog converters. The ESP32 is Arduino compatible and can be programmed using the Arduino IDE in Arduino language or using Micro Python.

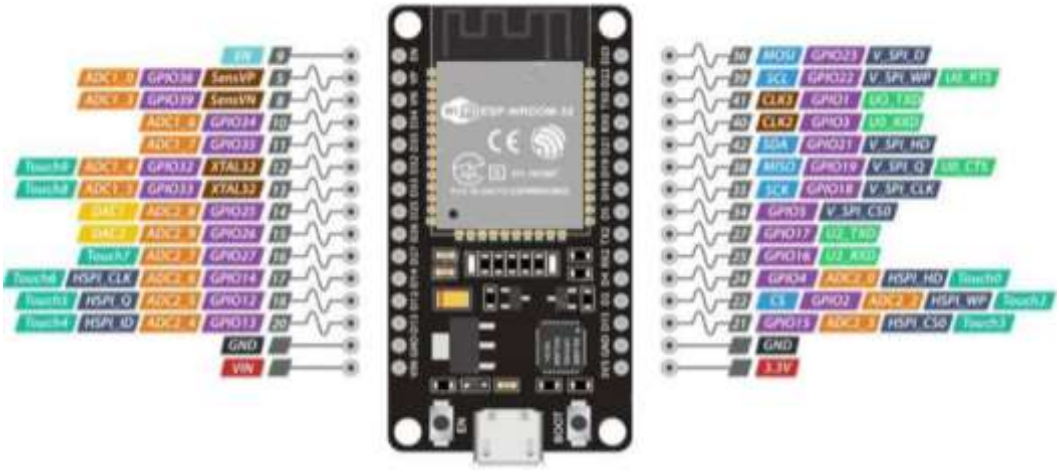


Figure 15. Pinout of ESP32 WROOM 32 version

The ESP32 chip can be found in a variety of boards with a wide range of prices, starting from 5€ up to 20€ or more. These boards usually differ on the number of pins provided or the quality of different board components like DACs.

All these three families of boards provide Bluetooth connectivity with BLE capabilities at a low price.

	Wi-Fi	Bluetooth	I/O	
Arduino Nano 33 BLE	No	Bluetooth 5 BLE	14 Digital pins 8 Analog pins	30€
Raspberry Pi Zero 2 W	Wi-Fi 802.11 b/g/n	Bluetooth 4.2 BLE	40-pin header 27 GPIO pins	19€
ESP32	Wi-Fi 802.11 b/g/n	Bluetooth 4.2 BLE	Up to 34 GPIO pins	5€ - 30€

Table 5. Microcontroller boards comparison

During this project, a board based on the ESP32 SoC is used due to low price, compatibility with Arduino IDE without the need of installing an operating system and having enough pins.

For implementation of the Bluetooth Low Energy MIDI protocol open-source libraries such as ESP32-BLE-MIDI [33], a library that implements most MIDI functionality encapsulated over the Bluetooth Low Energy protocol for ESP32 SoC boards.

2.4.3 Virtual Reality Experience

Headset

The virtual reality experience of the project will run on the Meta (previously known as Oculus) Quest 2 Headset. This headset is a standalone headset based on Android that can run virtual reality games and experiences. This standalone headset provides Wi-Fi and Bluetooth wireless connectivity. The device provides inside out tracking through an array of cameras. These cameras can calculate the position of the headset in the room with six degrees of freedom as well as track the two controllers. This headset can also track the position of your hands and fingers.

Game Engines compatibility

For development of virtual experiences, Meta provides different developer tools like the Meta Quest Developer Hub and extensions that implement compatibility with different game engines, and extensive documentation.

Meta officially supports development on Unity 3D though the Oculus XR plugin [34] that can be downloaded from the Unity Asset Store, which includes the Meta Oculus SDK allowing the implementation of the full functionalities of the headset, including XR passthrough, hand tracking and VR renderer.

In a similar fashion, Meta also officially supports developing on Unreal Engine 5 with the use of the MetaXR plugin for UE5 [35] that can be downloaded from the Meta Quest developer page, or by obtaining a custom version (fork) of Unreal Engine with the Meta VR capabilities built in. Both methods implement the full Meta Quest capabilities.

For the development of the Virtual Experience for this project, the Unity 3D game engine will be used due to it having detailed documentation and having previous personal knowledge of the game engine.

Using Unity 3D for our project has a small caveat. The Oculus XR plugin for Unity 3D requires additional configuration for the Bluetooth and Bluetooth low energy to work. This is caused because this plugin automatically remove the Android permissions related to Bluetooth functionality. For Bluetooth to work, a modification of the 1.4 version of the Oclus plugin needs to be performed by removing one line of code.

In the file OculusBuildProcessor.cs inside the Oculus XR plugin, the line 651 with the following content must be removed:

```
RemoveNameValueElementInTag(manifestDoc, nodePath, "uses-permission", "android:name",
"android.permission.BLUETOOTH");
```

After removing this line of code, Bluetooth permissions will be preserved intact after building the application. In the AndroidManifest.xml the following location and Bluetooth permission, and feature flag must be included:

```
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
<uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>

<uses-feature android:name="android.hardware.bluetooth_le" android:required="true" />
```

CHAPTER 3

3 Analysis

In this section, the analysis of the project will be detailed, explaining the user and software requirements, the use cases and traceability matrix. These steps of the analysis are fundamental for the design and development of the application.

3.1 User Requirements

This chapter defines the user requirements. User requirements defines what functionality the user can expect from the system.

3.1.1 User Requirement Template

Identifier		**Name**	
Description			
Priority		**System Element**	

Table 6. User Requirement Template

Next, detailed is the meaning of the fields of the template.

Identifier

This field is used to identify each requirement. It must be unique to each requirement. This identifier has the following structure: UR-XX

- XX corresponds to the number of the requirement.

Name

The name field is descriptive of the requirement but does not include any details.

Description

Detailed description of the requirement.

Priority

The implementation priority of the requirement. It can be:

- High priority.
- Medium priority.
- Low priority.

System Element

This field represents to which system element the requirement is for. It can take two values:

- Device: Requirements for the physical instrument device.
- VR: Requirements for the Virtual Reality Experience.

3.1.2 User Requirements

Device Requirements

Identifier	UR-01	**Name**	MIDI Compatible
Description	The instrument must be MIDI standard compatible, sending information using this protocol and able to communicate with other MIDI devices.		
Priority	High	**System Element**	Device

Table 7. UR-01 Definition

Identifier	UR-02	**Name**	Use Bluetooth Low Energy
Description	The device must be able to connect to other devices using Bluetooth Low Energy. The devices must be discoverable to other devices on boot up.		
Priority	High	**System Element**	Device

Table 8. UR-02 Definition

Identifier	UR-03	**Name**	16 Note Buttons
Description	The instrument must have sixteen buttons for playing musical notes.		
Priority	High	**System Element**	Device

Table 9. UR-03 Definition

Identifier	UR-04	**Name**	Variable Note Interval
Description	The user can select in which 16 notes interval is playing.		
Priority	High	**System Element**	Device

Table 10. UR-04 Definition

Identifier	UR-05	**Name**	Multiple notes at once
Description	The physical device must allow for the user to play multiple notes at once. Allowing playing any possible combination of notes on the current note interval.		
Priority	High	**System Element**	Device

Table 11. UR-05 Definition

Identifier	UR-06	**Name**	3D Trackable
Description	The instrument must be trackable in 3D space for later integration with the VR experience.		
Priority	High	**System Element**	Device

Table 12. UR-06 Definition

Identifier	UR-07	**Name**	Volume Control
Description	The user can select the sound volume from the device.		
Priority	High	**System Element**	Device

Table 13. UR-07 Definition

Identifier	UR-08	**Name**	Pitch Control
Description	The instrument allows the user to change the Pitch of the sound.		
Priority	High	**System Element**	Device

Table 14. UR-08 Definition

Identifier	UR-19	**Name**	BPM Control
Description	The instrument allows the user to change the BPM of hints reproduction.		
Priority	High	**System Element**	Device

Table 15. UR-19 Definition

Identifier	UR-09	**Name**	Bluetooth Low Energy Connectivity
Description	The virtual experience must be able to discover and connect to other Bluetooth Low Energy Devices.		
Priority	High	**System Element**	VR

Table 16. UR-09 Definition

Identifier	UR-10	**Name**	MIDI Compatible
Description	The experience must be able to receive and understand MIDI messages from connected devices.		
Priority	High	**System Element**	VR

Table 17. UR-10 Definition

Identifier	UR-11	**Name**	Play received notes
Description	The VR experience must implement a synthesizer that is able to play the received MIDI notes. This synthesizer must be able to play multiple notes at once.		
Priority	High	**System Element**	VR

Table 18. UR-11 Definition

Identifier	UR-12	**Name**	Device Representation
Description	The experience must show a 3D model that represents the physical device virtually. This representation must track in 3D space the device and the position of its buttons.		
Priority	High	**System Element**	VR

Table 19. UR-12 Definition

Identifier	UR-13	**Name**	Show notes to play
Description	The experience must show on the 3D representation of the device what note to play next from the music data retrieved.		
Priority	High	**System Element**	VR

Table 20. UR-13 Definition

Identifier	UR-14	**Name**	Start / Restart song
Description	The experience must allow the user start or restart the note hints.		
Priority	High	**System Element**	VR

Table 21. UR-14 Definition

Identifier	UR-15	**Name**	3D Hand Representation
Description	The experience shows a live representation in 3D of the user hands, tracking their position and gestures.		
Priority	High	**System Element**	VR

Table 22. UR-15 Definition

Identifier	UR-16	**Name**	Change Volume, BPM and Pitch
Description	The experience is able to change volume, BPM and pitch from information received.		
Priority	High	**System Element**	VR

Table 23. UR-16 Definition

Identifier	UR-17	**Name**	Unity 3D
Description	The system must be developed in Unity 3D		
Priority	High	**System Element**	VR

Table 24. UR-17 Definition

Identifier	UR-18	**Name**	Meta Quest 2
Description	The experience must be able to be executed in Meta Quest 2 as a standalone application.		
Priority	High	**System Element**	VR

Table 25. UR-18 Definition

3.2 Use Cases

In this section, the Use Cases for the device and experience are detailed. Uses cases explains what interactions or actions a user or actor can perform with each system.

3.2.1 Use Case Template

Use cases will be represented first with a Use Case diagram and later they will be expanded and defined in a use case table for each use case. The table will have the following structure:

Identifier	
Name	
System	
Actor	
Purpose	
Preconditions	
Typical Course	
Alternative Course	
Postconditions	

Table 26. Use Case Template

Field definitions

- **Identifier:** Unique identifier for each use case. Has the following structure: UC_YYYY_XX where YYYY corresponds to the system (VR or DEVICE) and XX corresponds to the number of the use case.
- **Name:** Descriptive name of the use case.

- **System:** The system to which the use case is for. It can be VR experience or Instrument Device.
- **Actor:** The actor that interacts with the system. Can be VR experience or Instrument Device.
- **Purpose:** Small description of the use case, describing what the actor can achieve with it.
- **Preconditions:** Conditions to be met before the use case is executed.
- **Typical Course:** Basic logic flow between the actor and the system.
- **Alternative Course:** Alternative flow that branches from the typical course execution.
- **Postconditions:** The result of the execution of the use case.

3.2.2 Use Cases

In this section, the different use cases will be defined. As the project consists of two system, two sets of use cases are defined, one for the Instrument Device and another one for the VR experience.

Instrument Device System Use Cases

The instrument device can be interacted by two actors:

VR Experience	As an external system, it represents the virtual reality experience.
User	The user that will use the VR headset

Table 27. Definition of Use Cases Actors for the instrument device system

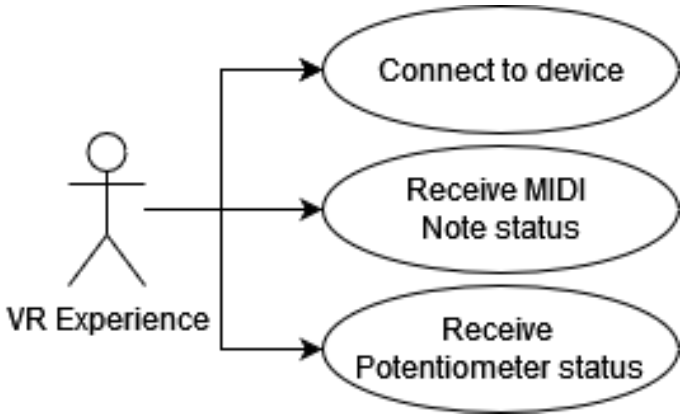


Figure 16. Use Cases – VR Experience Actor on Instrument Device System

Identifier	UC-DEVICE-01
Name	Connect to Device
System	Instrument Device
Actor	VR Experience
Purpose	Wirelessly connect to the musical instrument
Preconditions	The headset has not been connected to anything
Typical Course	1. The virtual experience enters connection mode broadcasting it is available. 2. Wait for instrument device to accept connection.
Alternative Course	None
Postconditions	The virtual experience is connected to the musical instrument device and ready to receive messages.

Table 28. Description of UC-DEVICE-01

Identifier	UC-DEVICE-02
Name	Receive MIDI note status
System	Instrument Device
Actor	VR Experience
Purpose	Receive the status of the Instrument Device notes
Preconditions	The Headset and device are paired, and a note has been pressed or released in the instrument device.
Typical Course	When a button is pressed on the instrument: • The instrument device sends a MIDI Note On that corresponds to the pressed button on the current note interval. • The VR experience receives a MIDI Note On. When a button is released on the instrument: • The instrument device sends a MIDI Note Off that corresponds to the pressed button on the current note interval. • The VR experience receives a MIDI Note Off.
Alternative Course	None
Postconditions	The virtual experience has received the correct MIDI Note message.

Table 29. Description of UC-DEVICE-02

Identifier	UC-DEVICE-03
Name	Receive potentiometer status
System	Instrument Device
Actor	VR Experience
Purpose	Receive the status of the instrument device potentiometers
Preconditions	The Headset and device are paired, and a potentiometer has changed in the instrument device.
Typical Course	1. The instrument device sends a MIDI message when a potentiometer has been changed. 2. The VR experience receives a message with the status of the potentiometer.
Alternative Course	None
Postconditions	The virtual experience has received the status of the potentiometer.

Table 30. Description of UC-DEVICE-03

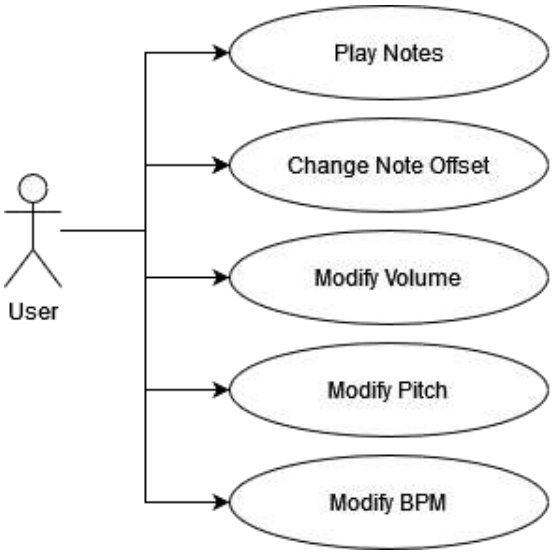


Figure 17. Use Cases – User Actor on Instrument Device System

Identifier	UC-DEVICE-04
Name	Play Notes
System	Instrument Device
Actor	User
Purpose	Notes are played when the user press buttons.
Preconditions	The Headset and device are paired.
Typical Course	1. The user presses a button. 2. A MIDI Note ON message is sent of the note in the current note interval. 3. The user releases the button. 4. A MIDI Note OFF message is sent of the note in the current note interval.
Alternative Course	None
Postconditions	The device has sent the correct messages.

Table 31. Description of UC-DEVICE-04

Identifier	UC-DEVICE-05
Name	Change Note Offset
System	Instrument Device
Actor	User
Purpose	The user can select what note interval he is playing in.
Preconditions	The Headset and device are paired.
Typical Course	1. The User modifies the note interval potentiometer. 2. The device internally selects the note interval of 16 notes mapped to the potentiometer position.
Alternative Course	None
Postconditions	The device will use the new note interval when sending notes.

Table 32. Description of UC-DEVICE-05

Identifier	UC-DEVICE-06
Name	Modify Volume
System	Instrument Device
Actor	User
Purpose	The user can modify the volume notes play at.
Preconditions	The Headset and device are paired.
Typical Course	1. The User modifies the volume potentiometer. 2. The device sends a MIDI message with the new volume information.
Alternative Course	None
Postconditions	The device sets the new volume information.

Table 33. Description of UC-DEVICE-06

Identifier	UC-DEVICE-07
Name	Modify Pitch
System	Instrument Device
Actor	User
Purpose	The user can modify the pitch notes play at.
Preconditions	The Headset and device are paired.
Typical Course	1. The User modifies the pitch potentiometer. 2. The device sends a MIDI message with the new pitch information.
Alternative Course	None
Postconditions	The device sends the new pitch status.

Table 34. Description of UC-DEVICE-07

Identifier	UC-DEVICE-08
Name	Modify BPM
System	Instrument Device
Actor	User
Purpose	The user can modify the BPM song hints play at.
Preconditions	The Headset and device are paired.
Typical Course	1. The User modifies the BPM potentiometer. 2. The device sends a MIDI message with the new BPM information.
Alternative Course	None
Postconditions	The device sends the new BPM status.

Table 35. Description of UC-DEVICE-08

The virtual experience can be interacted by two actors:

Instrument device	As an external system, it represents the physical instrument device.
User	The user that will use the VR headset

Table 36. Definition of Use Cases Actors for the instrument device system.

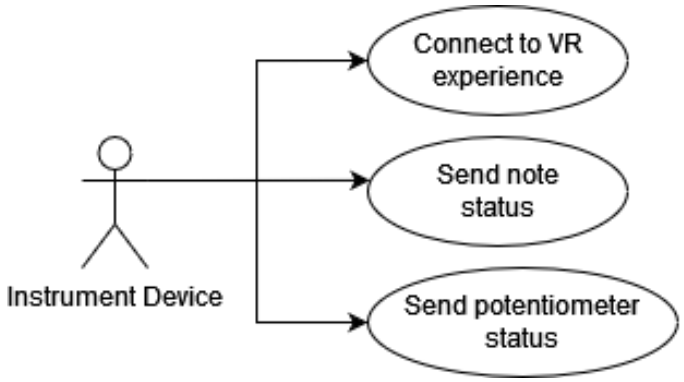


Figure 18. Use Cases – Instrument Device actor on VR Experience System

Identifier	UC-VR-01
Name	Connect to VR experience
System	VR Experience
Actor	Instrument Device
Purpose	Wirelessly connect to the VR headset
Preconditions	The device is not connected to anything
Typical Course	1. The device enters pairing mode, broadcasting it is available for pairing. 2. Wait for second device to accept connection.
Alternative Course	None
Postconditions	The device is connected to the Virtual Experience and ready to send messages.

Table 37. Description of UC-VR-01

Identifier	UC-VR-02
Name	Send note status
System	VR Experience
Actor	Instrument Device
Purpose	Send note status when a note button is pressed or released.
Preconditions	The device is connected to a recipient device and a button has been pressed or released.
Typical Course	When a button is pressed: • The instrument sends a NoteOn MIDI message of the note assigned to that button in the current interval. When a button is released: • The instrument sends a NoteOff MIDI message of the note assigned to that button in the current interval.
Alternative Course	None
Postconditions	A message with the correct note status is sent.

Table 38. Description of UC-VR -02

Identifier	UC-VR-03
Name	Send potentiometer status
Actor	Instrument Device
System	VR Experience
Purpose	Send potentiometer status for current note interval, volume, bpm, and pitch.
Preconditions	The device is connected to a recipient device
Typical Course	When one of the significant potentiometers is changed, the value is sent in a MIDI message.
Alternative Course	None
Postconditions	The potentiometer value is updated.

Table 39. Description of UC-VR-03

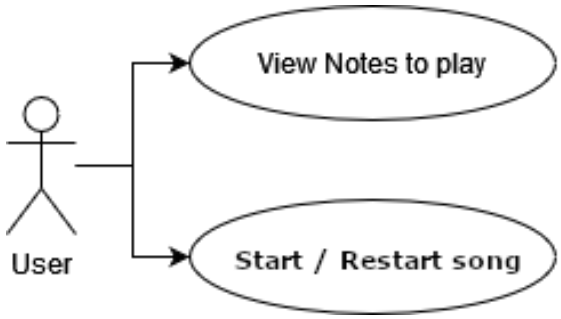


Figure 19. Use Cases – User actor on VR Experience System

Identifier	UC-VR-04
Name	View Notes to paly
Actor	User
System	VR Experience
Purpose	Display notes representation of the notes to play.
Preconditions	The device is connected to a recipient device. A song is playing.
Typical Course	When a song is playing, a note representation showing what button to press on the device will appear for the user to press.
Alternative Course	None
Postconditions	Notes are displayed.

Table 40. Description of UC-VR-04

Identifier	UC-VR-05
Name	Start / Restart Song
Actor	User
System	VR Experience
Purpose	The user can start / restart the song hints.
Preconditions	The device is connected to a recipient device.
Typical Course	1. The user presses the start button. 2. Song hints start appearing. 3. The user presses the start button again. 4. The song starts from the beginning.
Alternative Course	None
Postconditions	The song hints start from the beginning.

Table 41. Description of UC-VR-05

3.3 System Requirements

System requirements describes how the system must be built to satisfy the user requirements.

3.3.1 System Requirements Template

For the definition of the system requirements, the following table has been used.

Identifier			
Name			
Description			
Priority		**System Element**	
Dependencies			

Table 42. System Requirements template

Next, detailed is the meaning of the fields of the template.

Identifier

This field is used to identify each requirement. It must be unique to each requirement. This identifier has the following structure: SR-YY-XX

- YY corresponds to the type of system requirement, can be:
  - FR: Functional Requirement.
  - NFR: Non-functional requirement.
- XX corresponds to the number of the requirement.

Name

The name field is descriptive of the requirement but does not include any details.

Description

Detailed description of the requirement.

Priority

The implementation priority of the requirement. It can be:

- High priority.
- Medium priority.
- Low priority.

System Element

This field represents to which system element the requirement is for. It can take two values:

- Device: Requirements for the physical instrument device.
- VR: Requirements for the Virtual Reality Experience.

Dependencies

The User Requirements that this system requirement aims to meet.

3.3.2 System Requirements

Device System Requirements

Functional Requirements

Identifier	SR-FR-01		
Name	MIDI BLE implementation		
Description	The system shall implement the MIDI BLE protocol for sending messages wirelessly to another device.		
Priority	High	**System Element**	Device
Dependencies	UR-01, UR-02		

Table 43. SR-FR-01 Definition

Identifier	SR-FR-02		
Name	Bluetooth discoverability		
Description	The system shall enter Bluetooth discoverability mode when it is not connected to any other device.		
Priority	High	**System Element**	Device
Dependencies	UR-02		

Table 44. SR-FR-02 Definition

Identifier	SR-FR-03		
Name	Note Buttons		
Description	The system shall have sixteen note buttons. When a button is pressed or release, the correspondent MIDI Note message with the note in the current note interval is sent.		
Priority	High	**System Element**	Device
Dependencies	UR-03, UR-05		

Table 45. SR-FR-03 Definition

Identifier	SR-FR-04		
Name	Select Note Interval		
Description	The system shall have a slider that allows to select the current 16 note interval over the total 127 notes the device is playing in.		
Priority	High	**System Element**	Device
Dependencies	UR-04		

Table 46. SR-FR-04 Definition

Identifier	SR-FR-05		
Name	3D Tracker		
Description	The system shall have a tracker that allows the VR headset to know in what position in 3D space the device is in.		
Priority	High	**System Element**	Device
Dependencies	UR-06		

Table 47. SR-FR-05 Definition

Identifier	SR-FR-06		
Name	Send Note interval potentiometer value		
Description	The system must send the current note interval potentiometer value for representation of the potentiometer in the Virtual Experience.		
Priority	High	**System Element**	Device
Dependencies	UR-04		

Table 48. SR-FR-06 Definition

Identifier	SR-FR-07		
Name	Send Pitch Value		
Description	The system must send the value of the pitch potentiometer so that the pitch can be modified in the VR experience.		
Priority	High	**System**	Device
Dependencies	UR-08		

Table 49. SR-FR-07 Definition

Identifier	SR-FR-08		
Name	Send Volume Value		
Description	The system must send the value of the volume potentiometer so that the volume can be modified in the VR experience.		
Priority	High	**System Element**	Device
Dependencies	UR-07		

Table 50. SR-FR-08 Definition

Identifier	SR-FR-22		
Name	Send BPM Value		
Description	The system must send the value of the BPM potentiometer so that the reproduction BPM can be modified in the VR experience.		
Priority	High	**System Element**	Device
Dependencies	UR-19		

Table 51. SR-FR-22 Definition

Identifier	SR-NFR-01		
Name	MIDI Protocol		
Description	The system must use the MIDI protocol to send messages to external devices.		
Priority	High	**System Element**	Device
Dependencies	UR-01		

Table 52. SR-NFR-01 Definition

Identifier	SR-NFR-02		
Name	Bluetooth low energy device		
Description	The system must use Bluetooth low energy as transport layer for MIDI messages.		
Priority	High	**System Element**	Device
Dependencies	UR-02		

Table 53. SR-NFR-03 Definition

Identifier	SR-NFR-03		
Name	Multiple notes at a time		
Description	The system shall allow the user to press any number of notes at a time.		
Priority	High	**System Element**	Device
Dependencies	UR-05		

Table 54. SR-NFR-03 Definition

Identifier	SR-NFR-04		
Name	Pitch Message		
Description	The system shall send pitch information as a MIDIPitchWheel message at MIDI channel 0.		
Priority	High	**System Element**	Device
Dependencies	UR-08		

Table 55. SR-NFR-04 Definition

Identifier	SR-NFR-05		
Name	Volume Message		
Description	The system shall send volume information as a MIDIControlChange message at MIDI channel 0 and controller 1.		
Priority	High	**System Element**	Device
Dependencies	UR-07		

Table 56. SR-NFR-05 Definition

Identifier	SR-NFR-06		
Name	Note Interval Message		
Description	The system shall send note interval information as a MIDIControlChange message at MIDI channel 0 and controller 0.		
Priority	High	**System Element**	Device
Dependencies	UR-04		

Table 57. SR-NFR-06 Definition

Identifier	SR-NFR-19		
Name	BPM Message		
Description	The system shall send BPM information as a MIDIControlChange message at MIDI channel 0 and controller 2.		
Priority	High	**System Element**	Device
Dependencies	UR-19		

Table 58. SR-NFR-19 Definition

Functional Requirements

Identifier	SR-FR-09		
Name	MIDI BLE implementation		
Description	The system shall implement the MIDI BLE protocol for receiving messages wirelessly from another device.		
Priority	High	**System Element**	VR
Dependencies	UR-09, UR-10		

Table 59. SR-FR-09 Definition

Identifier	SR-FR-10		
Name	Bluetooth discoverability		
Description	The system must look for available Bluetooth Low Energy devices for connection when it is not connected to any device. Once a close device is found, it will connect to it.		
Priority	High	**System Element**	VR
Dependencies	UR-09		

Table 60. SR-FR-10 Definition

Identifier	SR-FR-11		
Name	Synthetiser		
Description	The system shall implement a simple note synthetiser that allows playing different notes as sound. It must be able to play multiple notes at once.		
Priority	High	**System Element**	VR
Dependencies	UR-11		

Table 61. SR-FR-11 Definition

Identifier	SR-FR-12		
Name	MIDI Note Status recognition		
Description	The system must be able to differentiate between MIDI Note ON and OFF, adding and removing notes from the synthetisers note buffer.		
Priority	High	**System Element**	VR
Dependencies	UR-11		

Table 62. SR-FR-12 Definition

Identifier	SR-FR-13		
Name	Device 3D model		
Description	The system must implement a 3D model of the musical instrument device.		
Priority	High	**System Element**	VR
Dependencies	UR-12		

Table 63. SR-FR-13 Definition

Identifier	SR-FR-14		
Name	Button status representation		
Description	The system must show what buttons are pressed down when a note is active.		
Priority	High	**System Element**	VR
Dependencies	UR-12		

Table 64. SR-FR-14 Definition

Identifier	SR-FR-15		
Name	Potentiometer representation		
Description	The system shall show the position of the volume, bpm, pitch and Note Interval potentiometers in the 3D representation.		
Priority	High	**System Element**	VR
Dependencies	UR-12		

Table 65. SR-FR-15 Definition

Identifier	SR-FR-16		
Name	Read MIDI music files		
Description	The system must be able to read MIDI music files.		
Priority	High	**System Element**	VR
Dependencies	UR-13, UR-14		

Table 66. SR-FR-16 Definition

Identifier	SR-FR-17		
Name	Show Notes to play		
Description	The system must be able to show the next notes of the current playing song read from the MIDI file in the 3D representation of the musical device.		
Priority	High	**System Element**	VR
Dependencies	UR-13		

Table 67. SR-FR-17 Definition

Identifier	SR-FR-18		
Name	Start / Restart song		
Description	The system must be able to start or restart the song hints.		
Priority	High	**System Element**	VR
Dependencies	UR-14		

Table 68. SR-FR-18 Definition

Identifier	SR-FR-19		
Name	Hand Tracking		
Description	The system must show a 3D representation of the user hands and gestures in 3D virtual space.		
Priority	High	**System Element**	VR
Dependencies	UR-15		

Table 69. SR-FR-19 Definition

Identifier	SR-FR-20		
Name	Modify sound volume		
Description	The system must be able to change the play volume depending on values retrieved from the instrument.		
Priority	High		VR
Dependencies	UR-16		

Table 70. SR-FR-20 Definition

Identifier	SR-FR-21		
Name	Modify sound pitch		
Description	The system must be able to change the sound Pitch depending on values retrieved from the instrument.		
Priority	High	**System Element**	VR
Dependencies	UR-16		

Table 71. SR-FR-21 Definition

Identifier	SR-FR-23		
Name	Modify BPM		
Description	The system must be able to change the BPM of the note hints reproduction.		
Priority	High	**System Element**	VR
Dependencies	UR-16		

Table 72. SR-FR-23 Definition

Identifier	SR-NFR-07		
Name	Sound initiated by user		
Description	The system must only reproduce musical notes when the action has been initiated by the user through the physical device hardware.		
Priority	High	**System Element**	VR
Dependencies	UR-11		

Table 73. SR-NFR-07

Identifier	SR-NFR-08		
Name	MIDI protocol		
Description	The system shall use the MIDI protocol for communication with external devices.		
Priority	High	**System Element**	VR
Dependencies	UR-09, UR-10		

Table 74. SR-NFR-08

Identifier	SR-NFR-09		
Name	MIDI over Bluetooth		
Description	The system shall communicate with the physical device using Bluetooth Low Energy using the MIDI protocol.		
Priority	High	**System Element**	VR
Dependencies	UR-09, UR-10		

Table 75. SR-NFR-09

Identifier	SR-NFR-10		
Name	Unity 3D		
Description	The system is developed using Unity 3D Professional edition.		
Priority	High	**System Element**	VR
Dependencies	UR-17		

Table 76. SR-NFR-10

Identifier	SR-NFR-11		
Name	Quest 2 Compatible		
Description	The system shall run on Meta Quest 2.		
Priority	High	**System Element**	VR
Dependencies	UR-18		

Table 77. SR-NFR-11

Identifier	SR-NFR-12		
Name	Virtual representation		
Description	The system must show a virtual representation of the physical device.		
Priority	High	**System Element**	VR
Dependencies	UR-12		

Table 78. SR-NFR-12

Identifier	SR-NFR-13		
Name	Pitch Message		
Description	The system must be able to retrieve pitch information received as OnMidiPitchWheel on channel 0.		
Priority	High	**System Element**	VR
Dependencies	UR-16, UR-12		

Table 79. SR-NFR-13

Identifier	SR-NFR-14		
Name	Volume Message		
Description	The system must be able to retrieve volume information received as OnMidiControlChange on channel 0 and controller 1.		
Priority	High	**System Element**	VR
Dependencies	UR-16, UR-12		

Table 80. SR-NFR-14

Identifier	SR-NFR-15		
Name	Note Interval Message		
Description	The system must be able to retrieve note interval information received as OnMidiControlChange on channel 0 and controller 0.		
Priority	High	**System Element**	VR
Dependencies	UR-12		

Table 81. SR-NFR-15

Identifier	SR-NFR-16		
Name	Start / Restart button		
Description	The system must use trigger button on the controller to change song.		
Priority	High	**System Element**	VR
Dependencies	UR-14		

Table 82. SR-NFR-16

Identifier	SR-NFR-17		
Name	Oculus XR		
Description	The system must use Oculus XR Unity plugin for integration with VR, controller tracking and hand tracking.		
Priority	High	**System Element**	VR
Dependencies	UR-18, UR-17, UR-15, UR-12		

Table 83. SR-NFR-17

Identifier	SR-NFR-18		
Name	Store Position Button		
Description	The system must use the Grip button on the attached controller for storing the current position of the board.		
Priority	High	**System Element**	VR
Dependencies	UR-12		

Table 84. SR-NFR-18

Identifier	SR-NFR-20		
Name	BPM Message		
Description	The system must be able to retrieve BPM information received as OnMidiControlChange on channel 0 and controller 2.		
Priority	High		VR
Dependencies	UR-16, UR-12		

Table 85. SR-NFR-20

3.4 Traceability Matrix

To check the consistency of the requirements, a traceability matrix is going to be used to ensure that the system requirements are consistent with the user requirements. A traceability matrix correlates each user requirement with one or more system requirement showing if it is being met.

The traceability matrix in Table 86 shows the User Requirements on the horizontal axis and the system requirements on the vertical axis. As shown in the traceability matrix, every user requirement has one or more system requirement, ensuring requirement consistency.

	UR-01	UR-02	UR-03	UR-04	UR-05	UR-06	UR-07	UR-08	UR-09	UR-10	UR-11	UR-12	UR-13	UR-14	UR-15	UR-16	UR-17	UR-18	UR-19
SR-FR-01	■	■																	
SR-FR-02		■																	
SR-FR-03			■		■														
SR-FR-04				■															
SR-FR-05						■													
SR-FR-06				■															
SR-FR-07								■											
SR-FR-08							■												
SR-FR-09									■	■									
SR-FR-10									■										
SR-FR-11											■								
SR-FR-12											■								
SR-FR-13												■							
SR-FR-14												■							
SR-FR-15												■							
SR-FR-16													■	■					
SR-FR-17													■						
SR-FR-18														■					
SR-FR-19															■				
SR-FR-20																■			
SR-FR-21																■			
SR-FR-22																			■
SR-FR-23																■			
SR-NFR-01	■																		
SR-NFR-02		■																	
SR-NFR-03					■														
SR-NFR-04								■											
SR-NFR-05									■										
SR-NFR-06				■															
SR-NFR-07											■								
SR-NFR-08									■	■									
SR-NFR-09									■	■									
SR-NFR-10																	■		
SR-NFR-11																		■	
SR-NFR-12												■							
SR-NFR-13												■				■			
SR-NFR-14												■				■			
SR-NFR-15												■							
SR-NFR-16														■					
SR-NFR-17												■			■		■	■	
SR-NFR-18												■							
SR-NFR-19																			■
SR-NFR-20												■				■			

Table 86. User and System requirement traceability matrix.

CHAPTER 4

4 Design and Implementation

This section specifies all design aspects of the instrument device and the VR experience. Later, detailing how it is implemented and the result of the implementation.

4.1 Physical Device Design and Implementation

The physical instrument must meet the previously proposed system requirements. This means, in summary, that the device must:

- Have sixteen buttons.
- At least 4 potentiometers.
- Must be able to be tracked in 3D space.
- Must connect wirelessly using Bluetooth and send MIDI messages.

In order to meet these requirements, the following 3D model has been designed.

4.1.1 Physical Device Hardware design

Case Design

A case design has been made using SketchUp that meets all requirements.

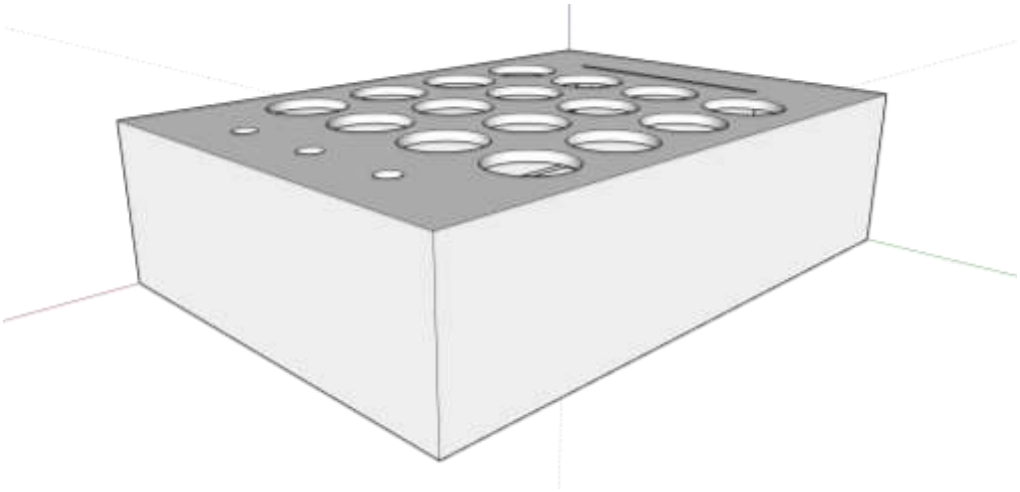


Figure 20. General view of device case

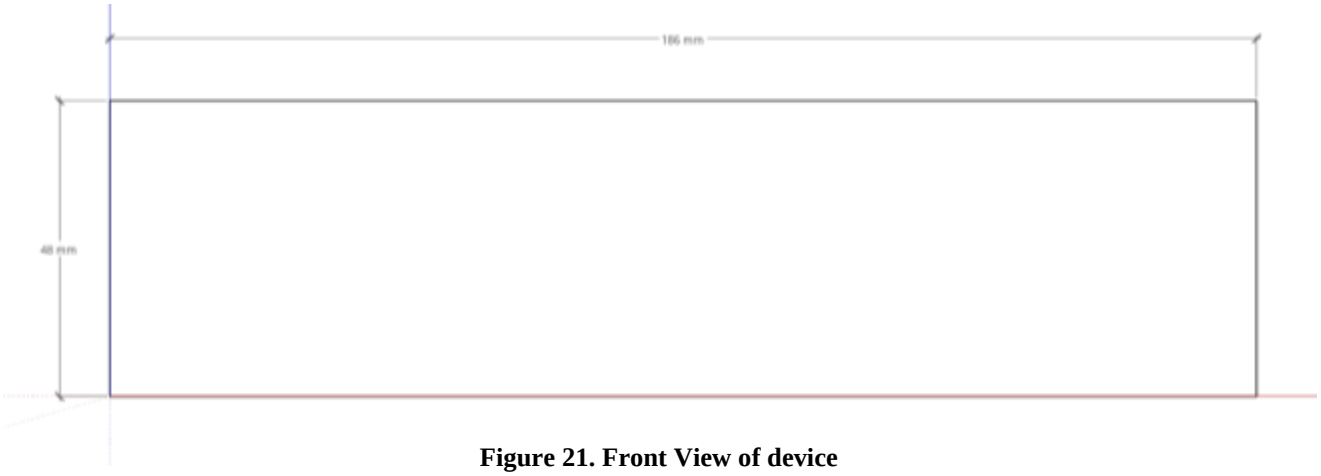


Figure 21. Front View of device case

Figure 22. Side View of device case

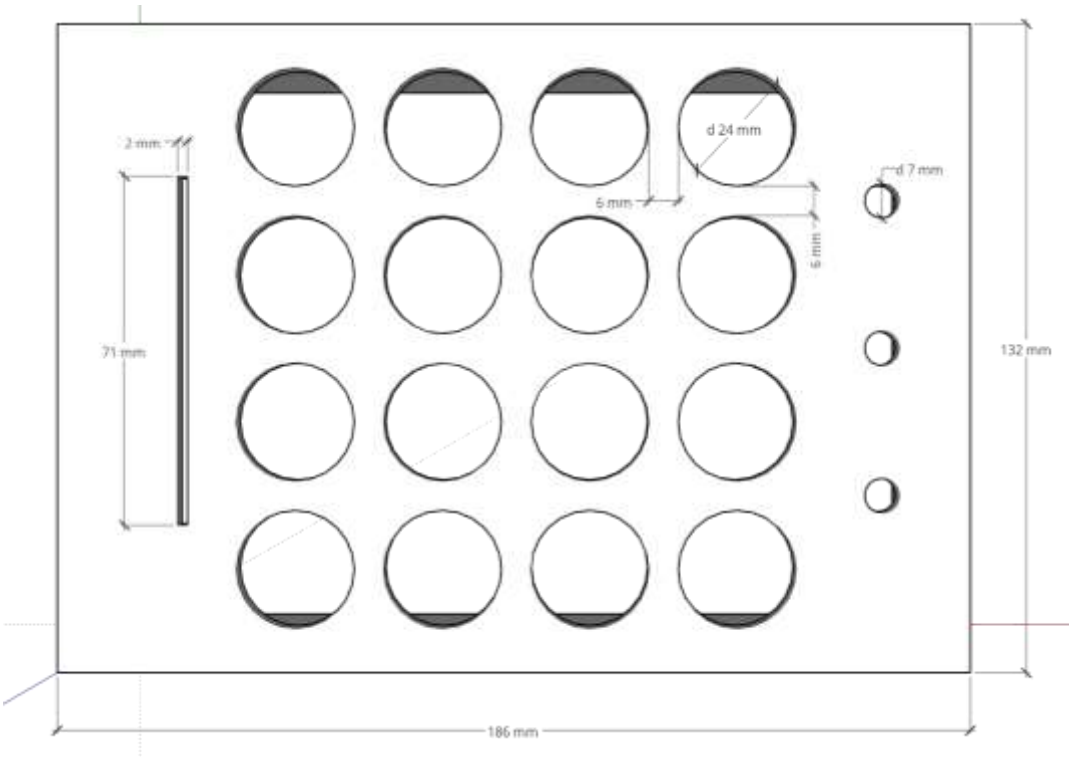


Figure 23. Top view of device case

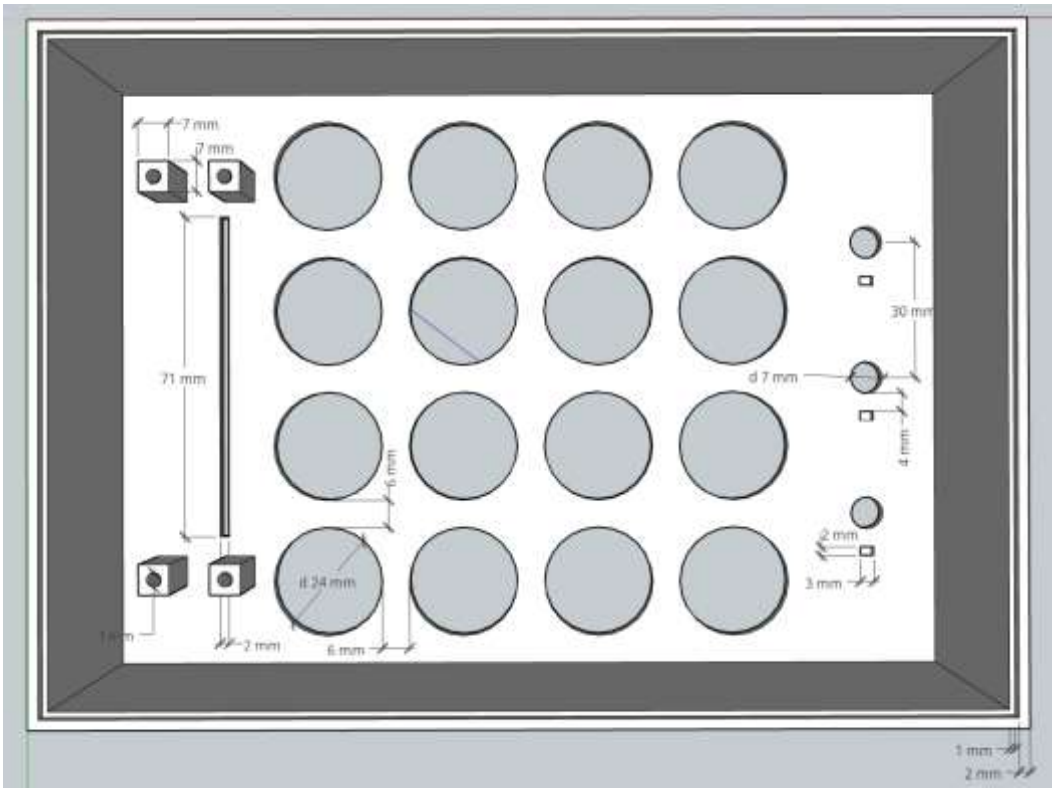


Figure 24. Underside view of device case

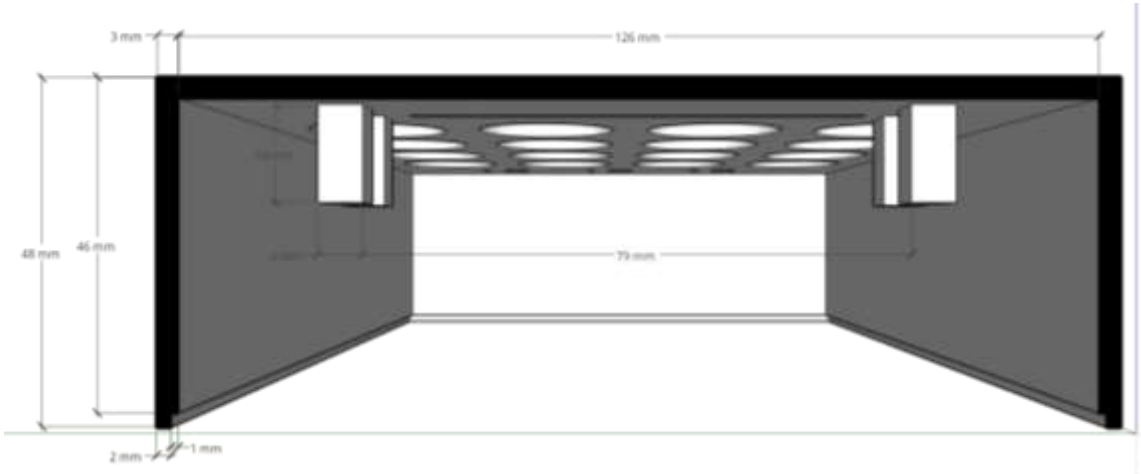


Figure 25. Section view of the side of the device case

In addition, a bottom cover has been designed in order to close the bottom part.

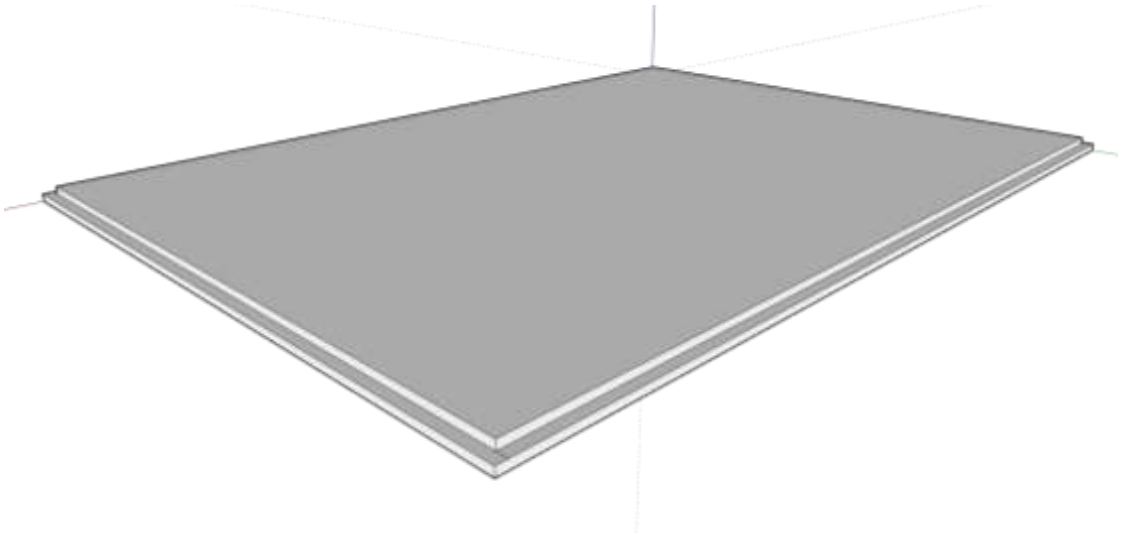


Figure 26. General view of the device back cover

Figure 27. Front View of device back cover

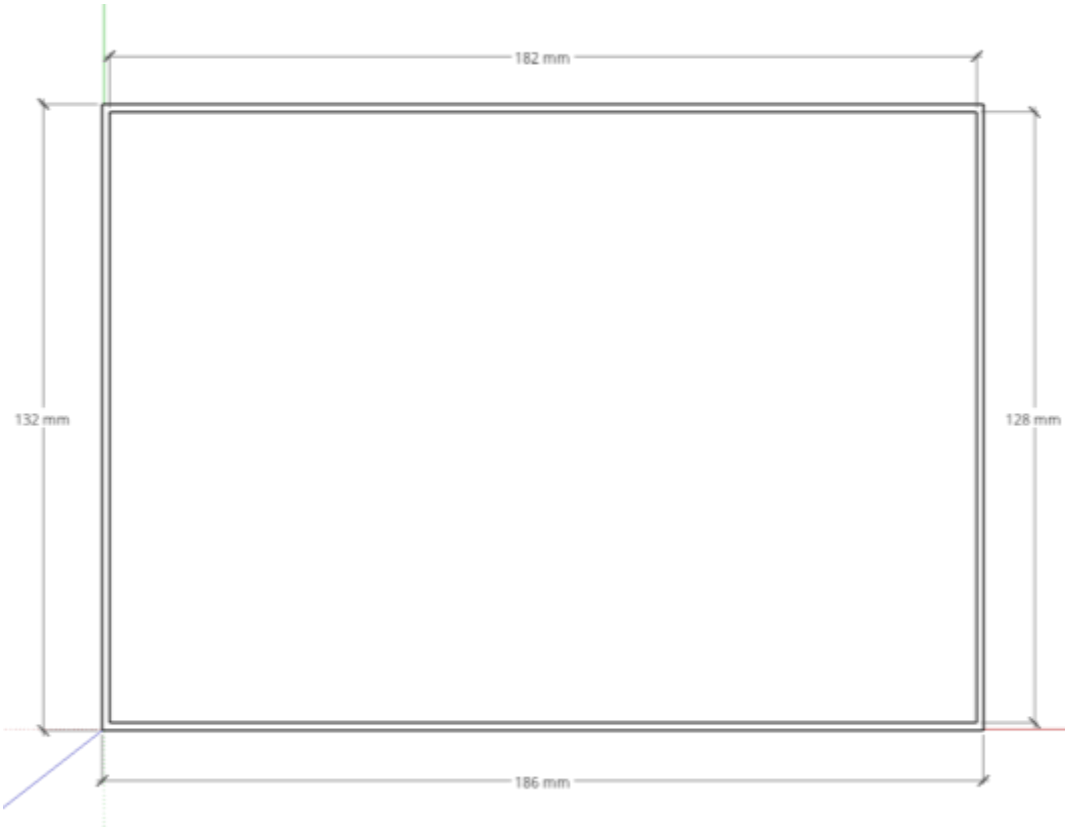
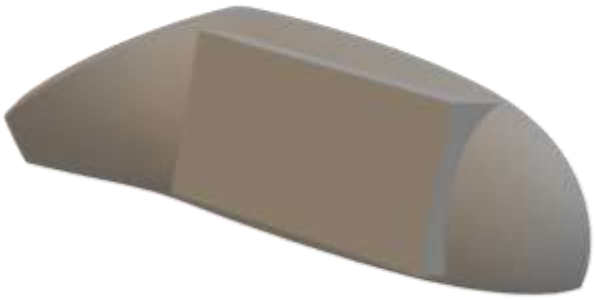


Figure 28. Top View of device back cover

This case design meets all the requirements allowing for 16 buttons, three rotational potentiometers and one sliding potentiometer. Finally, for tracking the device, one of the VR controllers will be used, so that the headset is able to locate the controller and the device. To do so, a modified battery cover for the controller has been created.

Controller Adapter design

This battery cover consists of an open licensed 3D model modified with a flat surface that allows easy attachment to the side of the device.



(a) Battery cover external side with attachment surface.

(b) Battery cover internal side.

Figure 29. Controller Adapter.

For attachment to the instrument case, a hook-and-loop fastener can be used. This would allow for easy attachment and detachment from the device, as well different positioning that might be more comfortable to the user.

Circuit Diagram

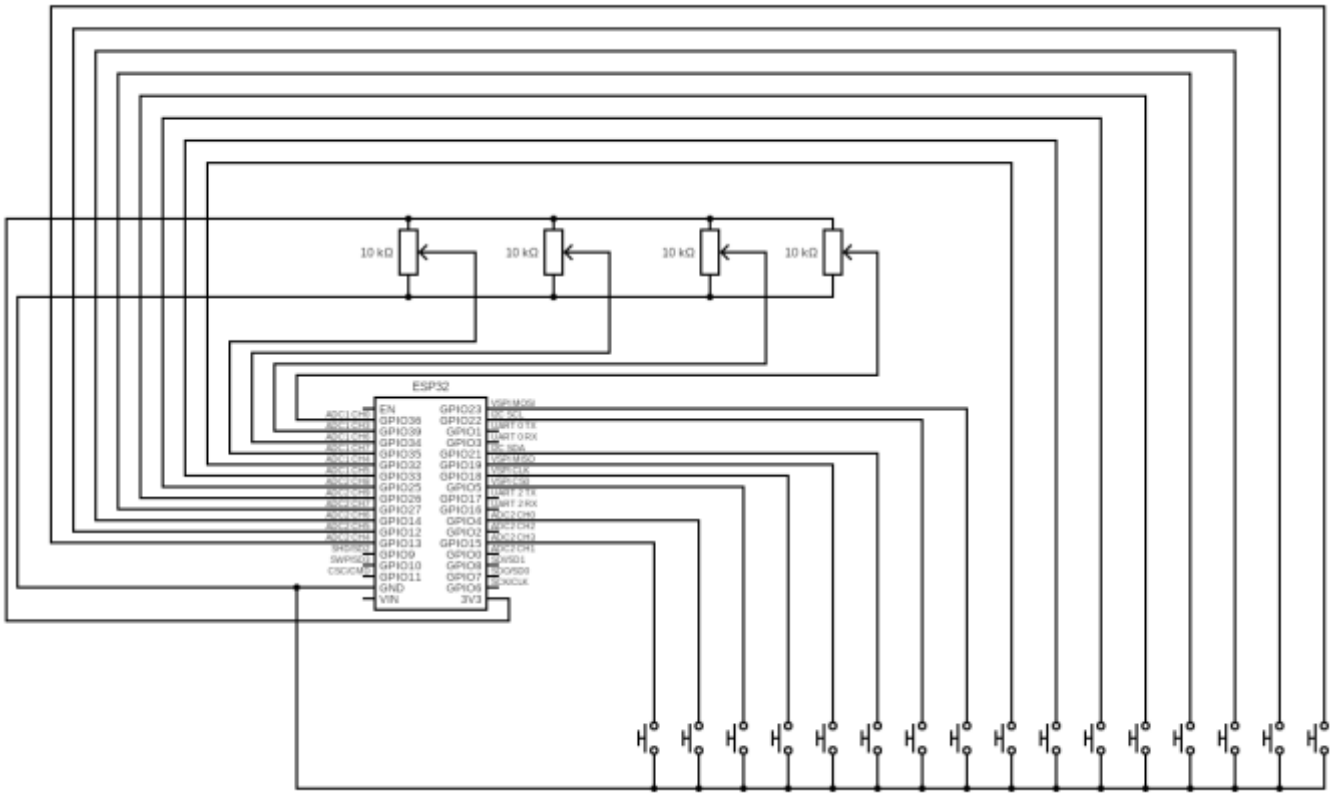


Figure 30. Circuit Diagram of instrument hardware

The circuit diagram in Figure 29 shows how the 16 buttons are connected each one to a pin of the ESP32. These pins provide pulldown - pullup resistors, allowing for easy connection without the need of adding resistors for each button. Finally, 4 potentiometers will be used (3 rotational and a slider potentiometer).

4.1.2 Physical Device Software Design

The software on the physical device consists in mainly two modules. First, the MIDI_Controller is the main Arduino class, that holds all the device logic and communicates with the second module the BLEMidiServer (part of the open-source library) which oversees implementing the Bluetooth Low Energy MIDI protocol and connect to the external device.

Sequence Diagram

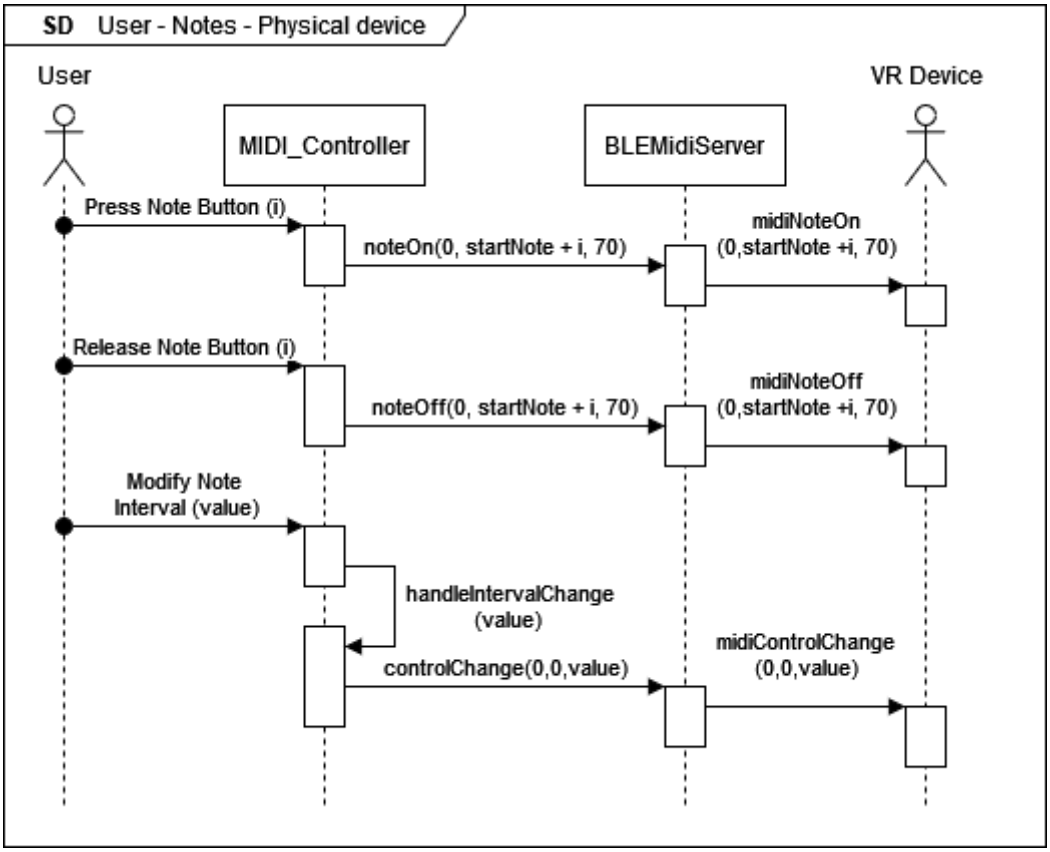


Figure 31. Sequence Diagram of note handling for the physical device

In this sequence diagram, the User and the VR device interact with the physical instrument, and detailed is how the user can press and release notes and how it is communicated to the VR device. The diagram also shows how the user can set the note interval, which is stored in device and the value of the potentiometer is sent to the VR experience for visual representation.

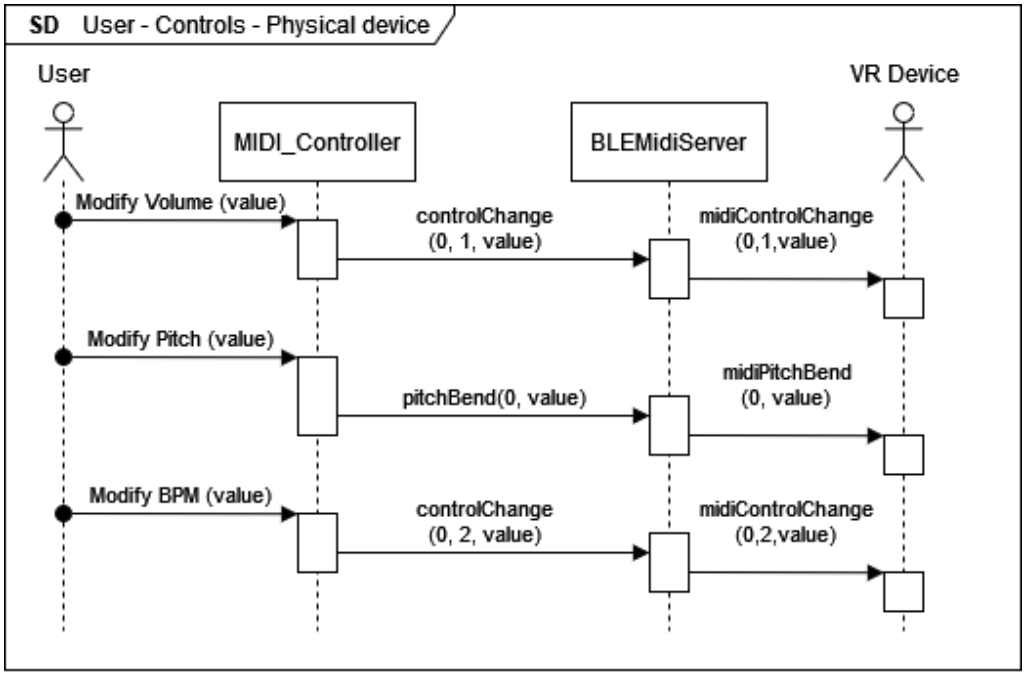


Figure 32. Volume and Pitch handling for the physical device

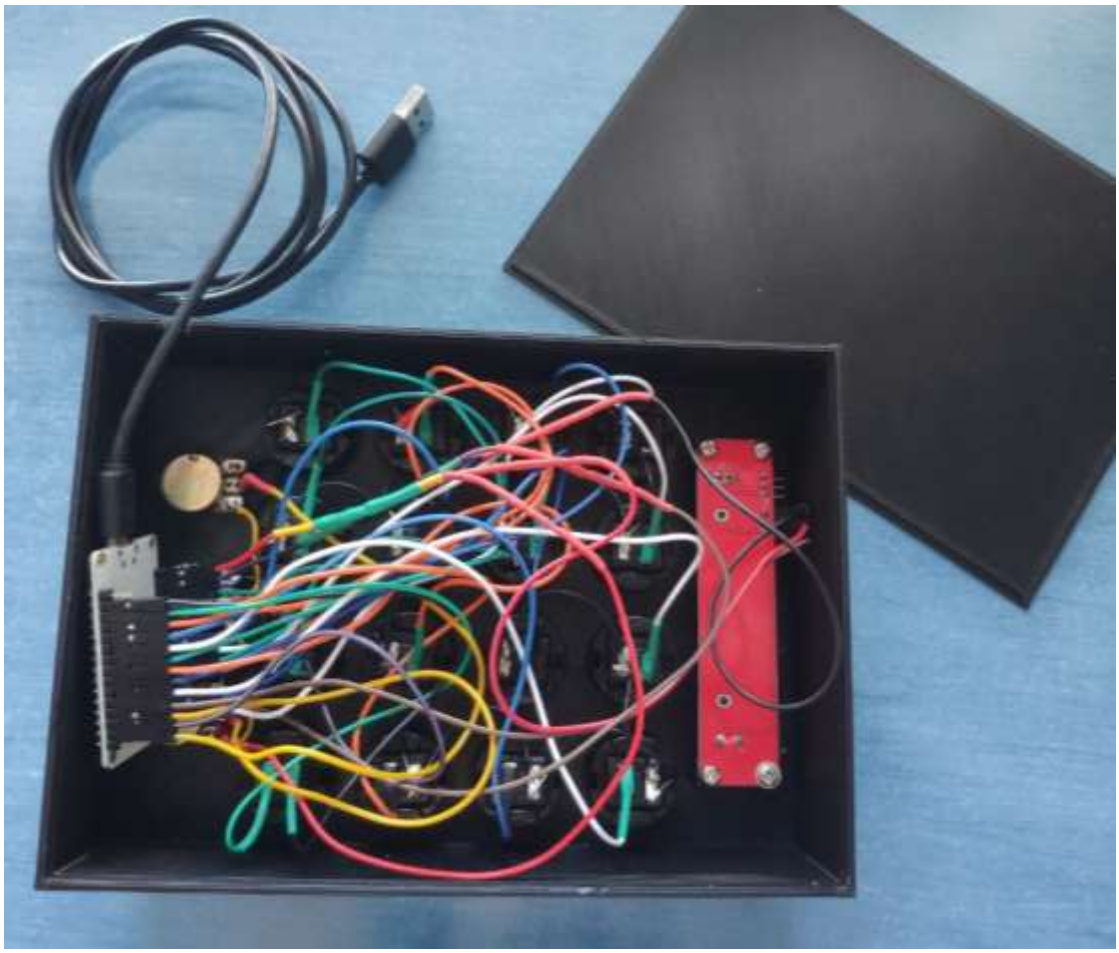
Finally, the user can modify volume, BPM and pitch, the MIDI_controller forwards the calls to the BLEMidiServer which is in charge of communicating these calls to the VR experience.

4.1.3 Physical Device Implementation

With the physical device software and hardware design completed, the implementation can be achieved. The device's case has been 3D printed in high density PLA through a local printing service. Once the case is printed all components have been mounted and connected using Figure 29 connection diagram. The source code for the ESP32 implementation can be found in GitHub:

https://github.com/juancarloscp52/TFE-MusicalVirtualRealityGame/tree/main/Device

(a) Instrument device front face.



(b) Instrument Device back side.

Figure 33. Instrument Device Implementation

Shown in Figure 32.a is the final implementation of the device button board with the Meta Quest 2 controller attached. This attachment is done thanks to the custom battery cover and is joined to the case with a hook-and-loop fastener. This kind of pad allows for easy positioning and detaching of the controller. The controller attachment is showed in detail in Figure 33 In Figure 32.b the backside of the device is shown, inside connection between the ESP32 micro controller and the rest of the components is found. Finally, a USB connection to the microcontroller is used for power. The device can receive power from common USB chargers or USB batteries.

(a) Controller battery cover with attachment pad.

(b) Controller battery cover inside.

(c) Controller with battery cover.

Figure 34. Custom controller battery cover

4.2 Virtual Experience Design and Implementatiton

Similarly, the virtual experience must satisfy the system requirements defined in the Analysis section. In summary, the requirements to meet are:

- Bluetooth device discoverability.
- Midi listener.
- Functional synthetiser.
- 3D device visualization and tracking.
- Notes hints.
- Hand tracking.
- Modify volume, bpm, and pitch.

The VR experience will be developed using Unity 3D and will use additional open-source libraries for MIDI and Bluetooth compatibility. These libraries are: smflite [36] for MIDI file compatibility, and BLE-MIDI-for-android [25] that enables Bluetooth MIDI compatibility on Android devices and Unity 3D.

4.2.1 Virtual Experience Software Design

Component Description

The virtual experience is composed by a set of components that where pre-existing (provided by 3D engine or external libraries). The identified components are:

- **Unity Engine:** Contains all components provided by Unity 3D. Is in charge of rendering, physics, audio, animation handling and much more functionality.
- **OculusXR:** Unity plugin provided by Meta [34] for integration of virtual reality headsets on the Unity Engine. It handles hand tracking, controller tracking and VR rendering.
- **SMFLite:** Open-source library used for MIDI file reading and retrieving MIDI sound events from a song. The plugin consists of a C# class that handles file loading and sequencing. It is located at the authors GitHub page [36].
- **BLEMidi:** BLE-MIDI-for-Android is another open-source library [25] that implements the MIDI protocol over Bluetooth low energy. It allows the user to send most kinds of MIDI messages over Bluetooth and handles device discoverability.
- **Button Board:** Handles general button board logic like updating button and potentiometer status on the 3D model and displaying note hints.
- **Synthesizer:** Simple sin wave synthesizer that allows playing multiple notes at a time.

In Figure 34 a Component Diagram of the VR experience is shown. Components with dark grey backgrounds corresponds to the Unity Engine and Unity add-ons. Components shown in light grey are open-source libraries.

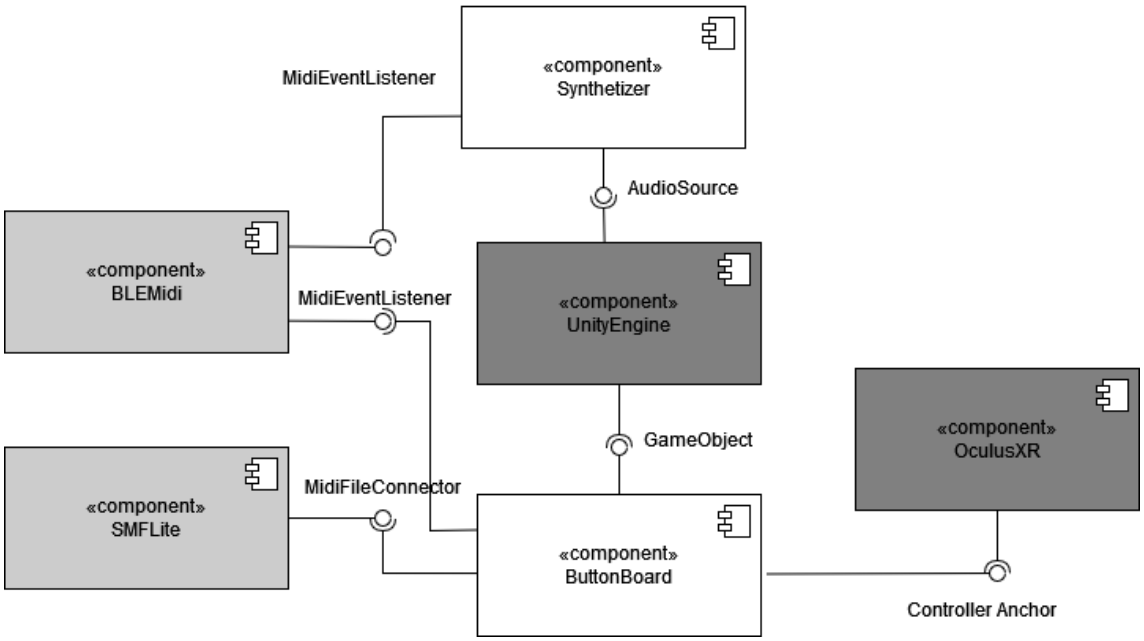


Figure 35. Sequence Diagram for VR experience.

Class Diagram

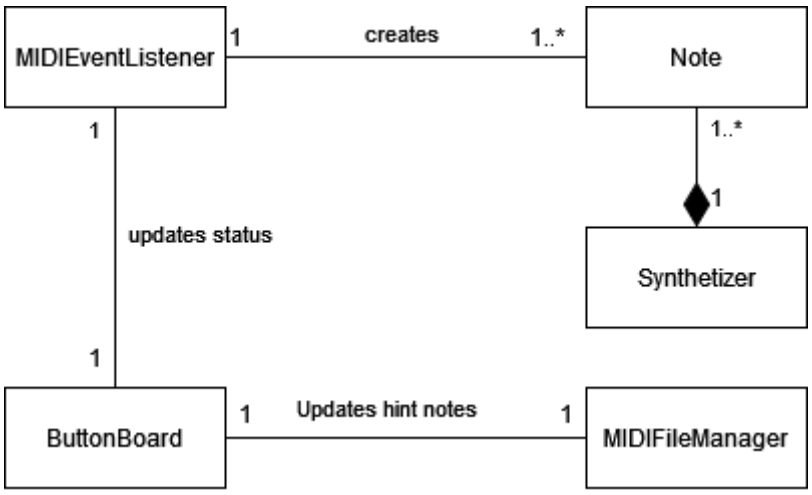


Figure 36. VR experience class diagram.

The above class diagram represents the logic behind the VR experience system handling the button board representation, note hints management, receiving MIDI notes and playing them. The rest of the logic such as hand tracking and the tracking of the VR controller (attached to the board) is handled directly in engine by the Oculus XR plugin.

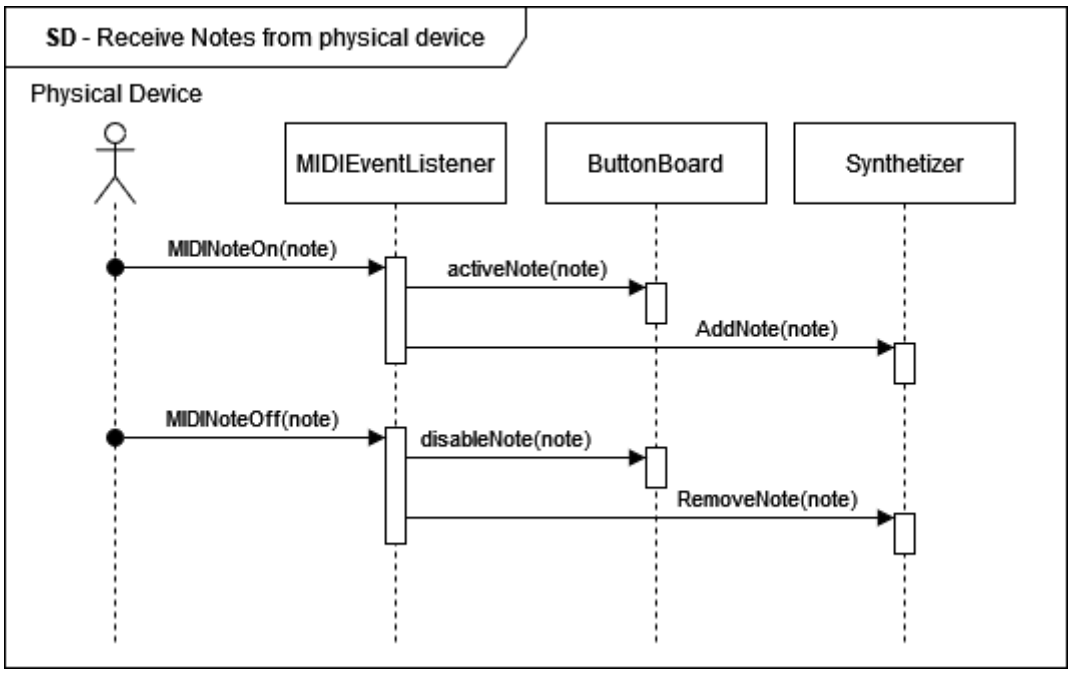


Figure 37. Sequence diagram of the Physical Device actor sending notes to the VR experience.

This sequence diagram describes how the physical device sends MIDI note messages to the virtual device, where they are received in the MIDIEventListener. This listener oversees forwarding the event and calling the correct function. For note events, it sends the button board the note status so it can display the button press. Additionally, it adds or removes the received note to the synthesizer so the note can be played.

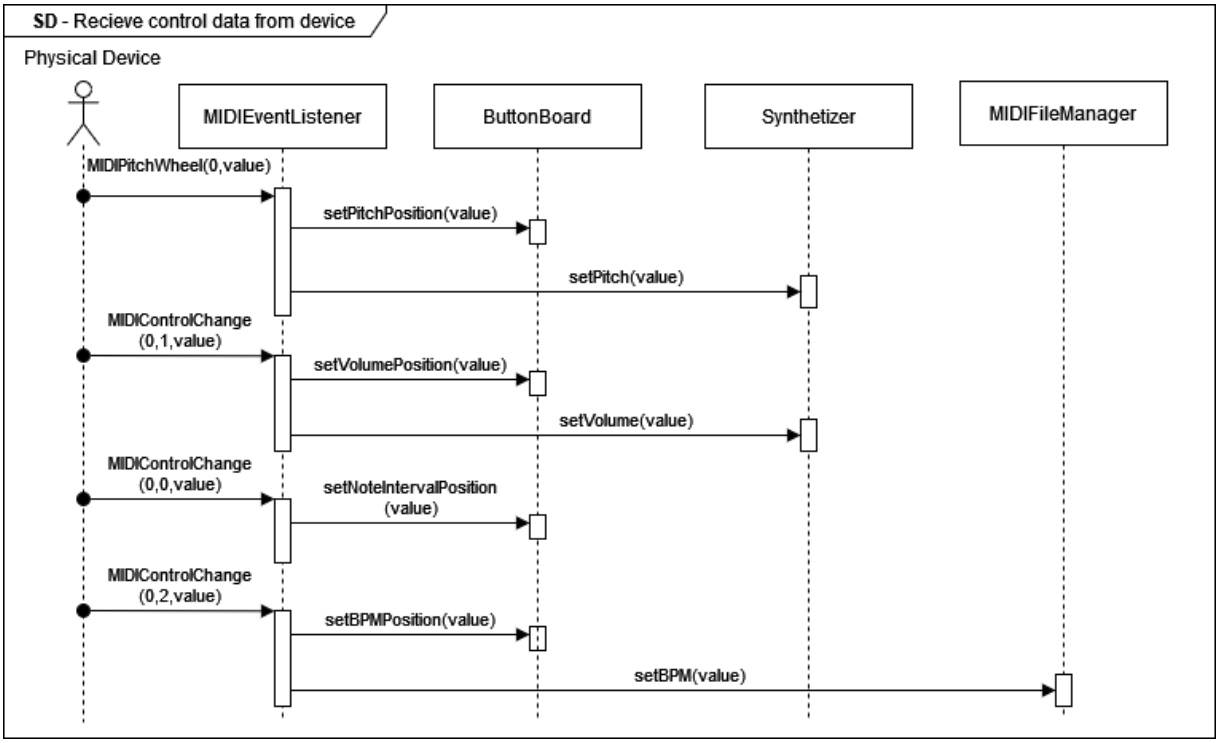


Figure 38. Sequence diagram of the Physical Device actor sending control and pitch changes to the VR experience.

This sequence diagram shows how the physical device can change the Pitch and it will be represented in the synthesizer and in the representation button board. The same happens with volume, assigned to controller number 1 and with BPM, assign to controller 2. The physical device also sends a control change message on control 0 when the user changes the note interval.

4.2.2 Virtual Experience Implementation

Once the experience has been designed, implementation can begin.

Using Unity 3D Professional Edition (with educational license) version 2022.2.17, a new empty project has been created with the Oculus XR plugin. The project consists of a simple scene with only the Oculus Controllers, the button board representation, and a debug console window. The source code for the project can be found in GitHub:

https://github.com/juancarloscp52/TFE-MusicalVirtualRealityGame/tree/main/Virtual%20Experience

MIDI message reception

The experience implements the Bluetooth low energy protocol and the MIDI standard using the BLE-MIDI-for-Android native library. As it is a native Android library, it must be instantiated using Unity's provided AndroidJavaClass invoker. This Android library comes with a specific class that implements compatibility directly with the Unity Engine for Android games. This class allows scanning for devices to connect and receive messages through the MIDIEventListener.

Synthetiser

The experience implements a simple sine wave synthetiser useful for playing notes. It consists of a note buffer and plays each note on the buffer at the same time until the note is removed. As suggested in the article "Procedural Audio with Unity" [37], it makes use of Unity's OnAudioFilterRead to send custom audio data to the available audio channels.

For each audio frame, the sine value at the current phase is compute for each note frequency, all note values are added up and sent to the audio channel.

Note frequencies need to be computed for each note, as each note is received as a value from 0 to 127 as detailed in the MIDI standard. For this translation to

frequency, the following formula detailed in “Midi note numbers and centre frequencies” [] is used.

$$f = 440 * 2^{(n-69)/12}$$

The user is able to modify the sound volume and pitch using the designated potentiometers in real time.

Note Hints

Using SMFLite, a music MIDI file is read, and for each note, a hint is displayed around the button that corresponds to that note, regardless in what note interval it is. Each hint is a coloured circumference over the button to be pressed with an animation making it close over the button. The user can select the hints BPM by changing the BPM potentiometer and pressing the start song trigger.

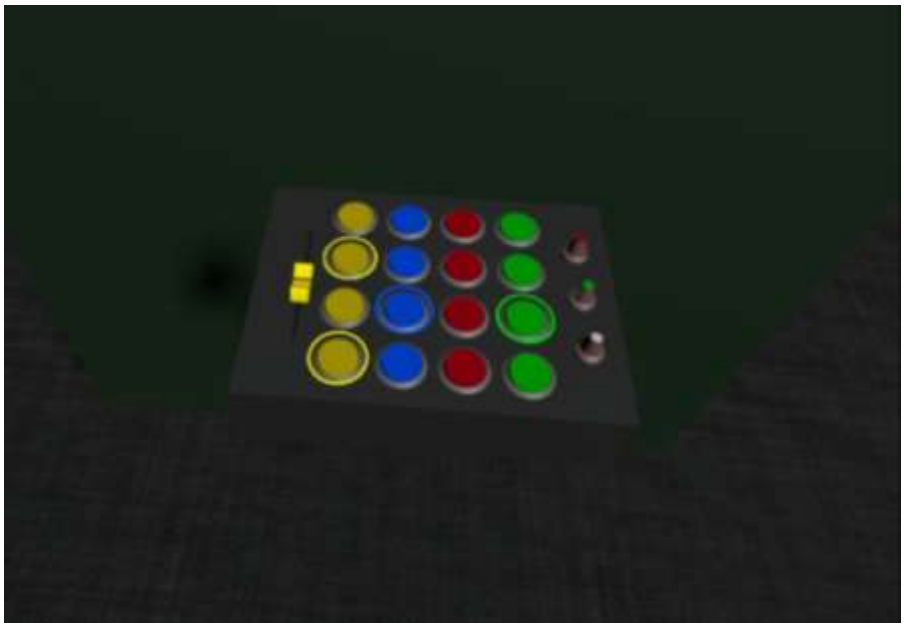


Figure 39. Button board representation with active note hints.

Device and Hand Tracking

The experience makes use of the Oculus XR Unity plugin for tracking the Quest 2 controllers. The left controller is used as anchor for the physical instrument. Additionally, the plugin is also useful for implementation of hand tracking. As the Quest 2 cannot track both controllers and hands at the same time, it switches tracking mode depending on if a button on the controller has been pressed.

The user can position the device and then press the grip trigger and the device position will be set. After a few seconds, the headset switches to hand tracking mode, where the user can see its hands for easy interaction with the instrument.

(a) Hand tracking mode

(b) Controller tracking Mode.

Figure 40. Different Tracking Modes

CHAPTER 5

5 Evaluation

This chapter will explore the evaluation of the implementation, making sure it is conforming with respect to the specified system requirements. To achieve this, a test procedure will be defined and carried out, making sure each requirement is met.

5.1 Conformance testing

Test Procedure Template

The following template will be used for formal test specification.

Identifier	
Name	
Requirement Coverage	
Pre-conditions	
Test Procedure	
Expected Results	
Result	

Table 87. Test Procedure Template

- **Identifier:** Unique alphanumeric string used for test procedure identification. The identifier has as structure TP-XX, where XX is the test procedure number.
- **Name:** Small description of the test procedure.
- **Requirement Coverage:** List of system requirements that the test checks conformity of the system.
- **Pre-conditions:** State of the system needed before the execution of the test. Summarizes the steps that must be taken prior to testing.
- **Test Procedure:** Description about the test and definition of steps to carry out during the test procedure.
- **Expected Results:** What is the expected state of the system after the execution of the test procedure.
- **Result:** States if the test has passed or not passed.

Test Procedures

Identifier	TP-01
Name	MIDI Bluetooth low energy.
Requirement Coverage	SR-FR-01, SR-FR-02, SR-NFR-01, SR-NFR-02, SR-FR-09, SR-FR-10, SR-NFR-08, SR-NFR-09, SR-NFR-10, SR-NFR-11
Pre-conditions	The instrument hardware is powered off.
Test Procedure	1. Connect the instrument device to power. 2. Start the virtual experience on the Meta Quest 2. 3. Press any button on the device
Expected Results	After a few seconds, the instrument and the virtual experience have been paired and the device is able to send messages to the experience.
Result	Passed

Table 88. TP-01

Identifier	TP-02
Name	Compatible with standard MIDI programs
Requirement Coverage	SR-FR-01, SR-FR-02, SR-NFR-01, SR-NFR-02.
Pre-conditions	The instrument hardware is powered off.
Test Procedure	1. Connect the instrument device to power. 2. Start *Piano MIDI BLE USB* app on a Bluetooth capable Android Phone. 3. On the app, go to the connection tab. 4. On the connection tab, press Start Scan. 5. Con Select MIDI Mode, choose Synthesizer option, press OK. 6. Press any button on the device
Expected Results	After a few seconds, the instrument and application have been paired and the device is able to send messages to the app.
Result	Passed

Table 89. TP-02

Identifier	TP-03
Name	Note Messages
Requirement Coverage	SR-FR-03, SR-NFR-03, SR-FR-11, SR-FR-12, SR-FR-14, SR-NFR-10, SR-NFR-12, SR-NFR-07, SR-NFR-11
Pre-conditions	The instrument and experience are connected.
Test Procedure	1. Press each note individually. 2. Press all notes at the same time.
Expected Results	The virtual experience shows a visualization of the board with the buttons pressed. A sound is played when a note is being pressed. When all notes are pressed at once, the experience shows all buttons pressed and plays all notes.
Result	Passed

Table 90. TP-03

Identifier	TP-04
Name	Note Interval Selection
Requirement Coverage	SR-FR-04, SR-FR-06, SR-FR-03, SR-FR-11, SR-FR-12, SR-FR-13, SR-FR-14, SR-FR-15, SR-NFR-10, SR-NFR-12, SR-NFR-07, SR-NFR-11, SR-NFR-15, SR-NFR-06
Pre-conditions	The instrument and experience are connected.
Test Procedure	1. Put the note interval slider at minimum value. 2. Press one button. 3. Put the note interval slider at the middle. 4. Press the same button. 5. Put the note interval slider at maximum value. 6. Press the same button
Expected Results	The virtual experience shows the position of the note interval slider. When a button is pressed, a note on the selected interval is played.
Result	Passed

Table 91. TP-04

Identifier	TP-05
Name	VR Device tracking
Requirement Coverage	SR-FR-05, SR-FR-13, SR-NFR-12, SR-NFR-10, SR-NFR-11, SR-NFR-18
Pre-conditions	The instrument and experience are connected.
Test Procedure	1. Press any button on the attached controller. 2. Move the instrument device. 3. Press the device attached controller grip button.
Expected Results	The virtual experience is able to track the position of the headset after a button on the controller is pressed. When step 3 is done, the current position of the board will remain after switching to hand tracking.
Result	Passed

Table 92. TP-05

Identifier	TP-06
Name	Modify Volume
Requirement Coverage	SR-FR-08, SR-FR-11, SR-FR-13, SR-FR-15, SR-FR-20, SR-NFR-12, SR-NFR-10, SR-NFR-11, SR-NFR-14, SR-NFR-05
Pre-conditions	The instrument and experience are connected.
Test Procedure	1. Put volume potentiometer at maximum value. 2. Press any button. 3. Put volume potentiometer at medium position. 4. Press the same button. 5. Put volume potentiometer at minimum value. 6. Press the same button.
Expected Results	The virtual experience plays a sound when the button is pressed, with the correct volume. The experience shows the position of the volume potentiometer.
Result	Passed

Table 93. TP-06

Identifier	TP-07
Name	Modify Pitch
Requirement Coverage	SR-FR-07, SR-FR-11, SR-FR-13, SR-FR-15, SR-FR-21, SR-NFR-12, SR-NFR-10, SR-NFR-11, SR-NFR-13, SR-NFR-04
Pre-conditions	The instrument and experience are connected.
Test Procedure	1. Put pitch potentiometer at medium value. 2. Press any button. 3. Put pitch potentiometer at max position. 4. Press the same button. 5. Put pitch potentiometer at minimum position. 6. Press the same button.
Expected Results	The virtual experience plays a sound when the button is pressed, with the intended pitch. The experience shows the position of the pitch potentiometer.
Result	Passed

Table 94. TP-07

Identifier	TP-08
Name	Start / Restart song
Requirement Coverage	SR-FR-16, SR-FR-17, SR-FR-18, SR-NFR-12, SR-NFR-10, SR-NFR-11, SR-NFR-16, SR-FR-23, SR-NFR-20, SR-NFR-19, SR-FR-22
Pre-conditions	The instrument and experience are connected.
Test Procedure	• Set the BPM potentiometer. • Press the song trigger. • Change the BPM potentiometer. • Press the song trigger.
Expected Results	The virtual experience starts playing the song hints from the begging in the selected BPM when the trigger is pressed.
Result	Passed

Table 95. TP-08

Identifier	TP-09
Name	Hand Tracking and tracking mode switching
Requirement Coverage	SR-FR-05, SR-FR-13, SR-FR-19, SR-NFR-10, SR-NFR-11, SR-NFR-17
Pre-conditions	The instrument and experience are connected.
Test Procedure	• Move hands in front of the headset. • Press any button on the device attached controller. • Wait 5 to 10 seconds without moving the device. • Move hands in front of the headset
Expected Results	The virtual experience shows a representation of the user hand movements and gestures. After pressing a button on the controller, the hands disappear and controlling tracker starts. After the controller being inactive for 5 to 10 seconds, the user hands are tracked again.
Result	Passed

Table 96. TP-09

5.2 Traceability Matrix

In the next page, shown is Table 97 where all System Requirements are listed on the horizontal axis and all Test Procedures are shown in the vertical axis. Each test procedure covers several system requirements, each one of them marked as covered by a highlighted cell. As shown in Figure 97, the set of Test Procedures covers all defined System Requirements.

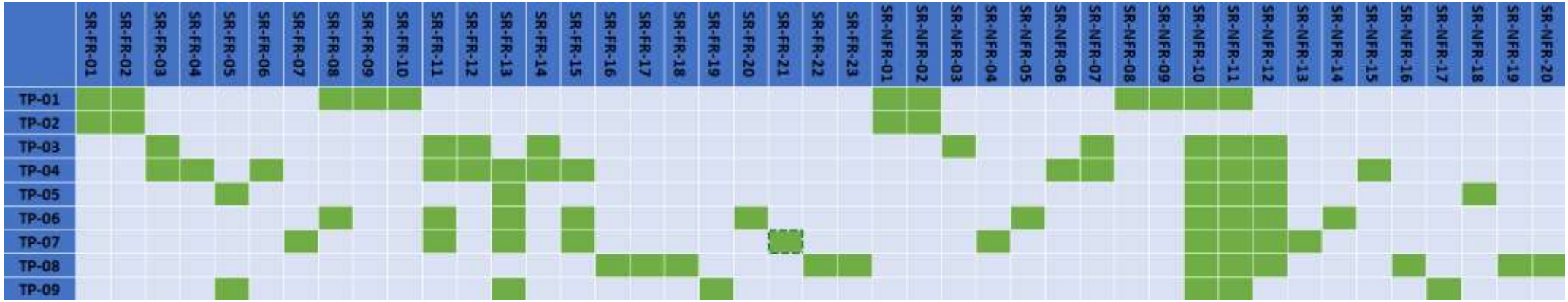


Table 97. Test Procedure – System Requirement Traceability Matrix

CHAPTER 6

6 Project Planning and Budget

This chapter reflects on the planification process for the project, detailing the different phases of the project and finally, discussing the project budget.

6.1 Planning

In Figure 40 the Gantt Chart is detailed with the different phases initially planned. The project is expected to take 30 weeks to complete. The following phases are planned:

- **State of the Art Analysis:** Research and analysis on the current state of the VR ecosystem was conducted. Additionally, in-depth research in instrument communication protocols and wireless technologies has also been carried out.
- **VR in Unity 3D training and experimentation:** This phase is aimed to achieve a greater understanding on how to implement Virtual Reality experiences using Unity 3D, especially in the Meta Quest 2 headset.
- **Planning:** General planification of the project, specification of the expected phases with their respective start and end date.
- **Analysis:** In-depth analysis of the problem, understanding what the required functionality for the user is and translating it into system requirements.
- **Design:** During this phase, the design of the solution is carried out, making sure conformance with system requirements is met, the design of both parts of the project is carried out. This phase is divided in two sub-phases:
    - Physical device design: Where the software and hardware design for the physical instrument is done.
    - Virtual device design: Consists of the design of the virtual experience software, specifying its mechanics, expected components and user interactions.
- **Implementation:** Phase where the implementation of both systems is done, based on the previous specified design. Consists of two sub-phases one for each system.
- **Evaluation and Testing:** The test methodology is defined, the test procedures are specified and carried out, making sure the system meets all requirements specified during Analysis phase.
- **Documentation:** This last phase, the project document is made, including information about these previous phases and results.

In Table 98, the start and end date of each phase is detailed:

Phase	Start Date	End Date	Duration in days
State of the art analysis	08/02/2023	23/02/2023	12
VR in Unity 3D training and experimentation	20/02/2023	21/03/2023	22
Planning	21/03/2023	30/03/2023	8
Analysis	30/03/2023	18/04/2023	14
Design	18/04/2023	08/05/2023	15
Implementation	08/05/2023	26/07/2023	58
Evaluation and testing	26/07/2023	18/08/2023	18
Documentation	17/07/2023	01/09/2023	35

Table 98. Planification Table

6.2 Methodology

The project makes use of iterative incremental delivery taken from Agile Methodologies [38] to iterate over working system deliveries with the ability to receive feedback and include improvements.

The project starts with a small delivery for each system, adding gradually functionality to them. The systems features are structured in a way that they are small portions of functionality quick to implement and easy to test.

When a feature is finished and tested, another feature can be started. During the development of this second feature, changes or improvements to the previous feature can be done, but always making sure it does not break the system.

This methodology usually allows for great communication with the client, having their priorities and feedback always in mind. As this project is developed privately and individually, these deliveries during development are not intended for the client since there isn't one. Instead, these deliveries allow the author to access the state of the project, quickly correct issues and make changes, while incrementing the functionality of the prototype.

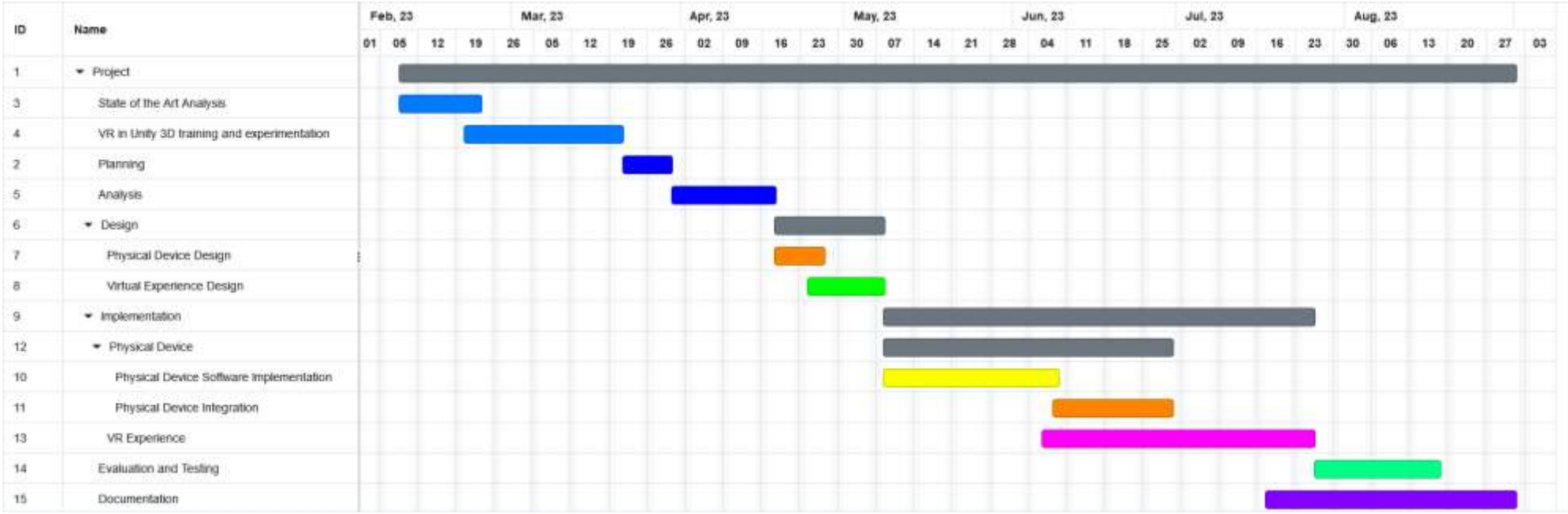


Figure 41. Grant Diagram with project planification.

6.3 Project Budget

The project budget is specified according to the expense in development, material, and possible licenses. This budget is specified as if the project was developed by a company, including budget for different human resources that could have been used for the project.

Labour Cost

Labor cost is specified depending on the labour cost for the role of each employee and the hours of labour. The project has taken 30 weeks for completion, taking about 12 hours per week adding up to 360 hours in total. These hours are broken down on the different roles:

Role	Hours of labour	Labour Cost / Hour	Total Cost
Analyst	62.4	16.41€	1023.98€
Designer	36	15.53€	559.08€
Programmer	139.2	14.62€	2035.10€
Tester	43.2	13.85€	598.32€
Documentation Specialist	84	10.77€	904.68€
Total Cost			5121.17€

Table 99. Human Cost of the project

Hourly rates are average salary for each role in Spain.

Equipment and Component Costs

Table 100 shows the equipment cost of the hardware used to develop the project. To compute the cost of the equipment regarding how much it is used, the price per month is computed over the total price estimating the device lifespan.

Equipment	Price	Lifespan (months)	Price /month	Total cost (over 7 months)
Personal computer	2150€	60	35.83€	250.81€
Meta Quest 2	400€	36	11.11€	77.77€
Soldering Iron kit	25€	48	0.52€	3.64€
Total Cost				332.22€

Table 100. Equipment cost

Component	Price
ESP32 WROOM-32	6.99€
Slider Potentiometer 10k Ohm	2.89€
Potentiometer 10k Ohm with plastic caps (3 units)	5.04€
Case and battery cover 3D print	35€
Micro USB cable	3.5€
Dupont cable 120 pack	8.48€
Arcade Push Button (16 units)	16€
Total	77.9€

Table 101. Component cost

Table 101 shows the cost of the components used for building the instrument device.

Software Costs

In the following table, the cost of the software licenses used during development of the project.

Software Product	License Price
Unity 3D Professional Edition	0€ *
Visual Studio Community	0€

Microsoft Office 365 for business	0€ *
Draw.io	0€
Circuit-diagram.org	0€
onlineGantt	0€
Gimp	0€
Google Drive	0€ *

Table 102. Software Cost

* Free educational licenses provided by UC3M.

As shown in Table 102, all software cost for the project is 0 as mostly free or open-source software has been used. For paid software, educational licenses have been used. This has allowed the total cost of the project to be zero.

Indirect Costs

Indirect costs represent expenses in utilities such as electricity, water, heating, and internet. It is estimated that the overall utility usage of the project is 10%.

Utility	Cost
Internet	22.4€
Electricity	38.5€
Gas	37.5€
Water	33.6€
Total Cost	132€

Table 103. Indirect costs

Total Project Cost

Finally, adding all costs, the project amounts to a total of 5663.29€ (Five thousand six hundred sixty-three Euros and twenty-nice Cents) as shown in table 104

Element	Cost
Labour Cost	5121.17€
Equipment Cost	332.22€
Component Cost	77.9€
Indirect Cost	132€
Total Cost	5663.29€

Table 104. Total Project Cost

CHAPTER 7

7 Legal Framework

The next section, the different legal concerns, regulations that might apply and licensing of third-party software will be discussed.

7.1 Legal and Regulatory Considerations

During the development of the project, several concerns and restrictions might need to be considered.

First, related to technical restrictions, the lack of knowledge in Virtual reality development and know-how on how to build physical devices has required a research and experimentation phase before the development of the project for the author to familiarize with the different systems at play.

Regarding economic restrictions, as this project has been developed without commercialization intend. The only economic restrictions at play are related to the initial cost of equipment use and component cost for building the device, both stated in the Project Budget section.

On the other hand, in order to understand possible legal restrictions, laws that might apply must be taken into account. First, as members of the European Union, the European General Data Protection Regulation (GDPR) [39] must be considered. This regulation, applicable since May 2018 aims to restrict what user data business can handle and aims to give the user more control over how their data is managed. Implementing this regulation in Spain, the “Ley Orgánica de Protección de Datos Personales y garantía de los derechos digitales” [40] was approved by the Spanish Parliament in October of the same year.

The final implementation of the project does not handle any kind of user data, so these regulations do not apply at the moment. But might need to be considered in case of future development. In similar fashion, as the application is developed as a personal project, no privacy policy, or End User License Agreement (EULA) is needed.

Finally, the licensing of third-party software must be taken into account. The project uses three open-source libraries with permissive licenses:

- BLE-MIDI-for-Android: Uses the Apache 2.0 license allowing modification, distribution, and commercial or private use.
- ESP32-BLE-MIDI: Also allows modification, distribution, and commercial or private use through the use of the MIT license.
- SMFLite: Uses a license found on the project GitHub that allows using the software without restriction.

CHAPTER 8

8 Socio-economic Environment

In this chapter the reader can find a description of the impact that the project can have and its usefulness on the current social and economic environments.

8.1 Socio-Economic Impact

This project is intended to be used as a musical learning aid, improving musical skills, and helping the user memorize songs. The project greatly helps the user to identify beats and coordinate notes with the song.

Additionally, the physical device is a cheaper alternative as an instrument device that is compatible with MIDI standard and works with any MIDI receiver and synthesizer. Additionally, it being 3D printed allows for easy replicability and provides a good build quality.

Although the use of a Virtual Reality Headset might not have been historically accessible due to the cost of the device, the use of Meta Quest 2 provides an affordable solution compared to other devices. Additionally, this device is much more accessible due to it being able to run applications standalone without the need of a powerful personal computer.

The integration of the headset with the musical device is made even simpler with the use of one of the headset controllers as tracking for the device, lowering the cost and implementation complexity for new users. And additionally, the use of Hand tracking helps orientate the user and understand how it can interact with the device in a virtual environment.

On the other hand, the project and its documentation are useful for understanding how Virtual Reality development is done and the different things needed to take into consideration. It also provides a good understanding of instrument communication protocol, with an in-depth emphasis on how the MIDI protocol works and how it is used with Bluetooth Low Energy. In general, this document is useful for anyone aiming to replicate or start its own implementation of the MIDI protocol. Additionally, anyone can use the project source-code to build upon and improve the functionality.

CHAPTER 9

9 Conclusion

Finally, in this chapter a retrospective of the project is done, reviewing the fulfilled objectives and detailing future lines of work.

9.1 Achieved Objectives

At the start of the project, a set of three main objectives and a series of sub-objectives were laid out. This document is an example of how these objectives can be achieved and how the project has been completed.

Regarding the first objective, an in-depth study of the currently available instrument communication protocol was successfully conducted. Finally deciding for MIDI as protocol to be implemented. It was also considered what type of wireless connection the protocol can be encapsulated in, deciding for Bluetooth low energy.

The second objective was achieved with the design, construction, and software implementation of a device with multiple buttons and potentiometers, making use of MIDI BLE for sending MIDI messages wirelessly. This objective resulted in the construction of a capable, inexpensive device that met all requirements.

Finally, the third objective of developing a virtual experience was also fulfilled. A virtual experience was created with the intention of helping the user learn how to play songs using the physical device. Resulting in a standalone application for Meta Quest 2 able to connect to the physical device and track it in virtual space.

9.2 Future Work

The project has been completed providing a satisfactory implementation, but it has considerable improvement potential.

For the Virtual Experience, an improved achievement system could be implemented, counting the number of correct notes by the user and giving a score. Additionally, an improved environment with more decoration and lighting responsive to music could be implemented.

The experience provides a MIDI song reader for the hint systems, but only one song is provided. A song selector allowing the user to choose what song to play is an interesting idea worth exploring.

The virtual experience implements a simple sinewave synthesizer for reproducing notes. It can be interesting to explore the possibility to implement a MIDI sampler that uses samples from different instruments.

Finally, it is worth exploring the implementation of a mixed reality experience. With the use of headset cameras, the user can see the real world with an overlay showing note hints and other interactions. This could be specially interesting with the launch of new headset with improved XR capabilities such as the Apple Vision Pro, the Meta Quest Pro or the not yet launched Meta Quest 3.

10 Bibliography

[1] Meta, “We believe in the future of connection in the metaverse,” [Online]. Available: https://about.meta.com/es/metaverse/. [Accessed April 2023].

[2] Apple Inc., “Apple Vision Pro,” [Online]. Available: https://www.apple.com/apple-vision-pro/. [Accessed July 2023].

[3] Amerincan University School of Education, “Virtual Reality in Education: Benefits, Tools, and Resources,” 16 December 2019. [Online]. Available: https://soeonline.american.edu/blog/benefits-of-virtual-reality-in-education/. [Accessed August 2023].

[4] R. Roberts, “You can now use VR to learn piano for free - MusicTech,” 11 August 2022. [Online]. Available: https://musictech.com/news/gear/you-can-now-use-vr-to-learn-piano-for-free/. [Accessed August 2023].

[5] A. N. Bosso, “How Rhythm Games Helped Me as a Learning Musician,” 6 January 2019. [Online]. Available: https://intothespine.com/2019/01/06/how-rhythm-games-helped-me-as-a-learning-musician/. [Accessed August 2023].

[6] M. Schwarz, “Beat Saber: How it began and how it's going,” 07 May 2023. [Online]. Available: https://mixed-news.com/en/beat-saber-how-it-began-how-its-going/. [Accessed August 2023].

[7] Cloudhead Games, “Pistol Whip,” 7 November 2019. [Online]. Available: https://www.cloudheadgames.com/pistol-whip. [Accessed August 2023].

[8] Kluge Interactive, “Synth Riders,” 21 June 2018. [Online]. Available: https://synthridersvr.com/. [Accessed August 2023].

[9] PotamWorks , “Smash Drums,” 2 December 2021. [Online]. Available: https://www.smashdrums.com/. [Accessed August 2023].

[10] Anotherway, “Unplugged - Air Guitar,” 14 December 2021. [Online]. Available: https://unpluggedairguitar.com/. [Accessed August 2023].

[11] P. Danowski, “Sounds Of The Metaverse: A Brief History of Virtual Reality Music Instruments and Virtual Music Venues,” [Online]. Available: https://panopticon.am/a-brief-history-of-virtual-reality-music-instruments-and-virtual-music-venues/. [Accessed August 2023].

[12] S. Serafin, C. Erkut, J. Kojs, N. C. Nilsson and R. Nordahl, “Virtual Reality Musical Instruments: State of the Art, Design Principles, and Future Directions,” *Computer Music Journal,* vol. 40, no. 3, pp. 22-40, 2016.

[13] ZarApps, “Piano Vision,” 2 August 2022. [Online]. Available: https://www.pianovision.com/. [Accessed August 2023].

[14] Thérémix Tech, “Modulia Studio,” 25 April 2019. [Online]. Available: https://www.modulia-studio.com/. [Accessed August 2023].

[15] J. Chadabe, “The Electronic Century Part IV: The seeds of the future,” 1 May 2001. [Online]. Available: https://web.archive.org/web/20120928230435/http:/www.emusician.com/gear/0769/the-electronic-century-part-iv-the-seeds-of-the-future/145415. [Accessed March 2023].

[16] The MIDI Association, “MIDI History Chapter 6-MIDI Begins 1981-1983,” [Online]. Available: https://www.midi.org/midi-articles/midi-history-chapter-6-midi-begins-1981-1983. [Accessed March 2023].

[17] D. Vandenneucker, “MIDI Tutorial Part 1 - MIDI Messages,” [Online]. Available: https://www.cs.cmu.edu/~music/cmsip/readings/MIDI%20tutorial%20for%20programmers.html. [Accessed 2023 March].

[18] “MIDI Cables,” [Online]. Available: http://musictechmusician.weebly.com/midi-cables.html. [Accessed March 2023].

[19] The MIDI Association, “ RTP-MIDI or MIDI over Networks,” [Online]. Available: https://www.midi.org/midi-articles/rtp-midi-or-midi-over-networks. [Accessed March 2023].

[20] The MIDI Association, “ Bluetooth LE MIDI Specification,” 1 November 2015. [Online]. Available: https://www.midi.org/specifications-old/item/bluetooth-le-midi. [Accessed March 2023].

[21] lathoub, “AppleMIDI (aka rtpMIDI) for Arduino,” [Online]. Available: https://github.com/lathoub/Arduino-AppleMIDI-Library. [Accessed April 2023].

[22] T. Erichsen, “rtpMIDI,” [Online]. Available: https://www.tobias-erichsen.de/software/rtpmidi.html. [Accessed April 2023].

[23] DisappointedPig, “DPMIDI,” [Online]. Available: https://github.com/DisappointedPig/DPMIDI. [Accessed April 2023].

[24] M. Green, “Bluetooth MIDI Is Here And Why It’s Important For You,” 4 April 2017. [Online]. Available: https://web.archive.org/web/20170410212126/http:/blog.cakewalk.com/bluetooth-midi-is-here-and-why-its-important-for-you/. [Accessed April 2023].

[25] K. Shoji, “BLE MIDI for Android,” 13 August 2014. [Online]. Available: https://github.com/kshoji/BLE-MIDI-for-Android. [Accessed April 2023].

[26] lathoub, “Arduino BLE-MIDI Transport,” 23 October 2018. [Online]. Available: https://github.com/lathoub/Arduino-BLE-MIDI. [Accessed April 2023].

[27] D. Moura, “rpi-midi-ble,” 22 February 2017. [Online]. Available: https://github.com/oxesoft/rpi-midi-ble. [Accessed April 2023].

[28] K. McMillen, D. Simon and M. Wrigth, “A Summary of the ZIPI Network,” *Computer Music Journal,* vol. 18, no. 4, pp. 74-80, 1994.

[29] M. Wright, “OpenSoundControl Specification 1.0,” 26 March 2002. [Online]. Available: https://opensoundcontrol.stanford.edu/spec-1_0.html. [Accessed April 2023].

[30] Center for New Music and Audio Technologies, “OSC for Arduino,” 24 September 2015. [Online]. Available: https://github.com/CNMAT/OSC. [Accessed April 2023].

[31] “OSC over WiFi,” [Online]. Available: https://www.etcconnect.com/webdocs/Controls/HOG/HTML/en/sect-osc_wifi.htm. [Accessed April 2023].

[32] The MIDI Manufacturers Association, “MIDI 1.0 Detailed Specification V4.2.1,” February 1996. [Online]. Available: https://www.midi.org/specifications/midi1-specifications/midi-1-0-core-specifications/midi-1-0-detailed-specification-2. [Accessed April 2023].

[33] M. André, “ESP32-BLE-MIDI,” 18 August 2020. [Online]. Available: https://github.com/max22-/ESP32-BLE-MIDI. [Accessed April 2023].

[34] Meta, “Install, Uninstall, and Upgrade XR Plugin,” [Online]. Available: https://developer.oculus.com/documentation/unity/unity-xr-plugin. [Accessed April 2023].

[35] Meta, “Creating Your First Meta Quest VR App in Unreal Engine,” [Online]. Available: https://developer.oculus.com/documentation/unreal/unreal-quick-start-guide-quest. [Accessed June 2023].

[36] K. Takahashi, “smflite,” 19 August 2013. [Online]. Available: https://github.com/keijiro/smflite. [Accessed June 2023].

[37] MCVUK, “Procedural audio with Unity,” 20 July 2012. [Online]. Available: https://mcvuk.com/development-news/procedural-audio-with-unity/. [Accessed June 2023].

[38] M. Iqbal, “Incremental Delivery and the Principles of the Agile Manifesto,” 29 March 2022. [Online]. Available: https://www.scrum.org/resources/blog/incremental-delivery-and-principles-agile-manifesto. [Accessed August 2023].

[39] European Parliament, “Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (GDPR),” 27 April 2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=CELEX%3A32016R0679. [Accessed August 2023].

[40] Spanish Parliament, “Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.,” 5 December 2018. [Online]. Available: https://www.boe.es/eli/es/lo/2018/12/05/3/con. [Accessed 2023 August].