



Gesture Controlled Musical Conducting

Kieran Flay

Supervised by Dave Marshall
Moderated by Alia I Abdelmoty

Module Information

One Semester Individual Project – 40 Credits
1516-CM3203

School of Computer Science and Informatics, Cardiff University

May 2015

Abstract

This final year project report seeks to answer the question “Can gesture control be used as a means to direct and manipulate music through a Digital Audio Workstation?”. Conventional sound controllers used for the production of digital music can be very expensive, this project will seek to provide an alternate and innovative way of controlling music at a fraction of the cost.

This paper will explore the use of two different gesture-based controllers to see if they are a viable and suitable input method for the purpose of controlling music. Furthermore, if an appropriate gesture controller is found, a software solution will be produced that will explore and test the capabilities of the selected gesture device.

The success of the project will be based on the viability, usability and functionality of the solution produced, in comparison to a standard digital music controller. Evidence will be collected through a user acceptance test at the end of the project to test the solution’s overall capability at answering the main research question.

Acknowledgements

Thank you to my supervisor Dave Marshall for his encouragement, support and advice through this project.

Thanks to the people that tested and provided feedback on the final solution.

The code related to this project can be found at: <https://github.com/Kee43/Gesture-Based-MIDI-Controller>

Contents

1 Introduction	8
1.1 Aims and Goals	8
1.2 Research Questions	9
1.2 Intended Audience	9
1.3 Project Scope	9
1.4 Approach	10
1.5 Assumptions	11
1.6 Project Constraints and Risks	11
2 Background	13
2.1 The Problem	13
2.2 Problem Context	13
2.2.1 A History of Digital Audio	13
2.2.2 Introduction to Gesture Control	14
2.2.3 Leap Motion vs Kinect	15
2.2 Relevant Theory	17
2.2.1 MIDI	17
2.2.2 Types of MIDI Commands	18
2.2.3 MIDI Control Change Messages	19
2.2.4 DAW Software	19
2.2.5 Leap Motion Controller	20
2.2.6 Leap Motion API	21
2.5 Existing Solutions	22
2.5.1 Leap Motion Store	22
2.5.2 Other Existing Solutions	24
2.7 Tools	25
3 Specification and Design	27
3.1 System Overview	27
3.1.1 The Solution	27
3.1.2 Software Overview	29
3.2 User Requirements	31

3.2.1 Functional Requirements	31
3.2.2 Non-functional Requirements	31
3.2 System Design	32
3.3 System Flow.....	33
3.4 System Architecture	36
3.5 User Interface Design	38
3.5.1 First Design	38
3.5.2 First Design - Interface Components	39
3.5.3 Required Changes.....	40
3.5.4 Second Design.....	41
3.5.5 Second Design - Interface Components	42
3.6 Control Design	42
3.6.1 Binary-Finger System	42
3.6.2 Interaction Box/Proximity Selection.....	43
3.7 Remaining Components	45
3.8 System Constraints.....	46
4 Implementation	48
4.1 Introduction.....	48
4.2 Technical Background	48
4.3 Tools Used	49
4.3.1 Language.....	49
4.3.2 Libraries/Add-ons	49
4.3.4 External Software	49
4.4 Critical Components	50
4.4.1 Leap Motion Controller	50
4.4.2 LoopMIDI Configuration	52
4.4.4 MIDI	53
4.4.5 User Interface	55
4.4.6 Control System	56
4.4.7 User Configuration and Saved Settings	60
4.4.8 CC Mapping within Reaper	62
5 Results and Evaluation	64
5.1 Testing	64

5.2 The Demo	64
5.2.1 Overview	64
5.2.2 Effects	66
5.3 User Acceptance Testing	68
5.3.1 Acceptance Testing Environment	68
5.3.2 Survey Design	69
5.3.3 Results Analysis.....	70
5.4 Further Development.....	72
5.5 Requirements Evaluation	73
5.6 Critical Appraisal.....	82
5.6.1 Weaknesses	82
5.6.2 Strengths.....	83
6 Future Work.....	84
6.1 Increased Quantity of Complex Actions	84
6.2 Further Analysis and Testing	85
6.3 Documentation.....	85
6.4 Input Combinations	85
7 Conclusion	87
8 Reflection and Learning	88
8.1 Reflection	88
8.2 Learning.....	88
List of Acronyms	90
Appendices	91
Bibliography.....	98

Table of Figures

Figure 1 - Low sample bit rate vs high sample bit rate	13
Figure 2 - Digital Audio Recording Process	14
Figure 3 - Myo Armband	15
Figure 4 - Gest Glove	15
Figure 5 - The Leap Motion Sensor	15
Figure 6 - The Microsoft Kinect 2.0.....	16
Table 1 - Leap Motion Sensor vs Kinect Comparison.....	16
Figure 7 - Conventional MIDI controller	17
Figure 8 - MIDI Command Types.....	18
Figure 9 - Structure of a MIDI CC Command.....	19
Figure 10 - Reaper User Interface	20
Figure 11 - How the Leap Motion Works.....	21
Figure 12 - The Leap Motion Co-ordinate Detection.....	21
Figure 13 - The GECON MIDI User Interface	22
Figure 14 - The Muse User Interface	23
Figure 15 - Swoosh User Interface	24
Figure 16 - Remedi Glove	25
Figure 17 - The WindowBuilder Interface.....	26
Figure 18 - The Standard MIDI Setup.....	28
Figure 19 - The New MIDI Setup	28
Figure 20 - Screenshot of the Developed Application	29
Figure 21 – loopMIDI Interface	30
Figure 22 – Reaper Interface.....	30
Figure 23 - High Level System Flow	33
Figure 24 - Detailed System Flow.....	34
Figure 25 - System Class Diagram	36
Figure 26 - User Interface Mock-up (first iteration)	38
Figure 27 - User Interface Mock-up (second iteration)	41
Figure 28 – Binary Representation	43
Figure 29 – Gesture to Select Track	43

Figure 30 - The Interaction Box.....	43
Figure 31 - Track Selection Grid	44
Figure 32 - The Track Panel.....	45
Figure 33 - What the Leap Motion Sees	50
Figure 34 - loopMIDI Port Configuration	52
Figure 35 - Solution Port Selection	52
Figure 36 - Reaper Preferences.....	53
Figure 37 - loopMIDI Test Results	55
Figure 38 - The Main User Interface	55
Figure 39 - Implemented Track Selection	56
Table 2 - Axis directional Table	58
Figure 40 - Configuration Table	60
Figure 41 - Action Slot Configuration.....	61
Figure 42 - Reaper 'Learn' Feature	63
Figure 43 - Reaper CC Mapping Feature	63
Figure 44 – MIDI Sequences and Track Slots	65
Figure 45 – Loading Virtual Instruments into Tracks	65
Figure 46 – Grand Piano Commands	66
Figure 47 – Flute Commands	66
Figure 48 – Action Slots	67
Table 3 – Actions Mapped for the Demo.....	67
Figure 49 – User Testing Setup	69
Graph 1 – User Question Response Breakdown for Question 4	71
Graph 2 – User Question Response Breakdown for Question 5	72
Figure 50 - How the Leap Motion Provides Feedback.....	73
Figure 51 - MIDI Port Detected in the Solution	74
Figure 52 - MIDI Port Detected in Reaper	74
Figure 53 - MIDI Port Configured in loopMIDI.....	74
Figure 54 - CC Command Mapped in Reaper.....	75
Figure 55 - Track Preferences	76
Figure 56 - Track Preferences Once Application is Re-opened.....	76
Figure 57 - Possible CC Commands.....	77
Figure 58 - Configuration Table	77

Figure 59 - Possible Axis Configurations	78
Figure 60 - Global Actions	78
Figure 61 - Windows Task Manager	79
Figure 62 - Application and DAW on One Screen	80
Figure 63 - Example of a Home Recording Studio	80
Figure 64 - The User Interface	81
Figure 65 – Actions in Reaper that can Be Mapped	84
Figure 66 – Facial Expression Recognition	86
Figure 67 – The Emotiv Mind Control Device	86

1 Introduction

In 2015, according to a IFPI's Digital Music Report, revenues from digital music services matched those from physical format sales for the first time taking the global revenue to \$14.97 billion [1]. With the creation of digital music being more accessible than ever due to the available tools, artists are looking for innovative ways to create the next chart-topping single to take advantage of the increasing demand.

The aim of this project is to use a gestural input device to allow for the manipulation of music, this means recognizing and mapping appropriate hand gestures into musical 'effects' that can manipulate a piece of music being played on digital audio software. This project essentially provides a new platform for making music through a digital platform. It blurs the lines between digital and physical music by letting the artist express themselves through gestures while playing a digital piece of music.

There are two main devices that have been identified with the correct capabilities to complete the project, these are the Leap Motion Sensor [2] and the Microsoft Kinect [3]. Both devices allow for the recording of user gestures through dedicated built-in sensors; the capabilities of each device will be analyzed in-depth in the document to determine which device is more suitable for the project.

Common uses for gesture technology already include gaming, productivity and education [4]. Controlling music is a largely unexplored area with regards to mainstream gesture control, with many unexplored use cases. If the project is successful, it could help make gesture control a more integrated part in the production of digital music.

1.1 Aims and Goals

The overall objective is to determine whether gesture control can be used for directing music, this can be broken down into sub-objectives as follows:

- Connect to a gestural input device and process the input data into something meaningful.
- Map input gestural data to musical effects.
- Send data to a Digital Audio Workstation to apply musical effects to sounds being played.
- Control system that is well-designed based on the gestural device and easy to use.
- Visually appealing User Interface.

1.2 Research Questions

The aims and goals of the project can be translated into a series of research questions that will help determine if the project has been a successful research paper:

- Is existing gesture control hardware suitable for controlling the production of digital music?
- How difficult is it for the average user to learn how to use gesture control to successfully send MIDI commands?
- Could gesture controlled MIDI controllers ever replace traditional MIDI controllers?
- How likely are professional artists going to use gesture control in their music?

1.2 Intended Audience

The project has a potentially large audience. A typical expert user of the system would be a “tech-savvy” user that is interested in digital music with some background knowledge regarding digital music controllers. As the system is designed to complement and not replace a digital audio workstation it would be preferable to also have some background knowledge, however some workstations can be easier to learn than others so it is not always required.

A medium level user would typically be someone that has some knowledge about music but no knowledge of the digital creation of music. There is a huge number of artists that play physical instruments that do not have any knowledge of the digital creation of music so the tool could be an intuitive and fun way of bridging the gap for some artists by allowing them to physically manipulate digital music.

Despite preferably having some knowledge of digital music, the software will be designed to be accessible by anyone who has an interest in music in general. I would say that this describes myself before this project as I did not have any knowledge of digital music – I simply just had an interest in gesture control and music. The system will also be free to get the most experiencing the potential of what can be achieved with gesture control and music.

As this is both a practical and research-based project, people from industry and education could benefit from the project. As the solution will combine two technologies in an interactive way, there is a potential for further research to be conducted on top of the foundations of the project. An example of an individual in this area could be a University professor or someone with a vested interest regarding the research of gesture control or digital music.

1.3 Project Scope

The system is written in Java and designed to run on a Windows machine, no other platforms will be considered in the scope. Java is a universal language that can be run on a large number of devices including Windows and OS X, so this will keep the development

load to a minimum while being able to reach a large portion of the intended audience. Due to time restrictions, the project will only be built and tested for the Windows platform.

The solution produced will involve creating a virtual controller that connects to digital music software. Existing software that is part of the overall system will not be in the scope, it will just be required to connect the solution up so that its capabilities can be shown and tested.

The project will be tested so that it meets its requirements, however extensive testing will not be conducted due to time constraints in the time-plan. After a minor UX/UI evaluation it could be expected that the software will be ready to publish online through a medium such as the Leap Motion Application store [5], but this publication will not be required as part of the current scope.

The last code build of the system will be expected to meet all of the functional requirements and aims; this will be a required part of the scope in order to demonstrate the full capabilities of the system and to prove if the project has been a success.

1.4 Approach

The project was approached with the Agile methodology; due to the initial lack of knowledge on digital music it was expected that influence from the supervisor along with research would cause the project to change over the course of its development, so it was important to be prepared. I have experience of using Agile from my industrial placement, so it is a methodology I am familiar and comfortable with. The project fell into four main stages with some overlapping, these were research, implementation, experimentation and documentation.

The initial report was focused on the research side of the project, this was to gain a solid base of knowledge regarding the fundamental components that would make up the system, these included the Leap Motion Sensor and digital music concepts such as MIDI and CC Commands. Research continued in parallel to the development stage which contributed to changes being made during implementation.

A feature of Agile is that “it helps teams respond to unpredictability through incremental, iterative work cadences, known as sprints” [6]. Once coding started, it became natural that each week became a sprint due to the arrangement with the project supervisor where meetings are conducted at the start of every week. Continuous discussion with the project supervisor allowed for feedback to be taken forward into the next weekly sprint as the feedback was translated into improvements to align the system closer to its requirements.

The third stage involves collecting user feedback and testing the overall functionality of the system to determine if it has met its requirements. The results of the tests and user evaluation could be used as a starting point for someone willing to continue to develop the project further.

1.5 Assumptions

Several assumptions have been made during the development of the system. Firstly, it is assumed that the system will be used in adequate lighting conditions. According to the documentation, the Leap Motion Sensor achieves its “best performance in an environment without any external infrared light sources”. Having suitable lighting during testing will make sure that test results are not affected.

Secondly, it must be assumed that the user is physically capable of controlling the gesture controller with two hands and that they are able to hear sound. The system will require two hands due to the complexity of the actions, so someone who is handicapped may not be able to use the system as efficiently compared to more capable users.

Another assumption will be that the user has a computer that meets the hardware requirements required to use the Leap Motion Controller [7], these are:

- Windows® 7+ or Mac® OS X 10.7+
- AMD Phenom™ II or Intel® Core™ i3/i5/i7 processor
- 2 GB RAM
- USB 2.0 port
- Internet connection

An internet connection is only required for the controller setup so it will be possible to use the system without an internet connection once the Leap Motion has been set up with the provided configuration software. According to current statistics [8], it is safe to assume that most users of Windows have Windows 7 and above. To run the software, it will also be assumed that the user's computer is running Java, which is commonly installed to run many programs on computers.

1.6 Project Constraints and Risks

Skill Requirements

As stated in previous section, a user does not necessarily need background of digital music knowledge to use the software, but it would be required to use the software to its potential. This is a risk to the project as it means it is partially dependent on the user's skillset. The risk is that if the software is not being used to its potential by users, it may result in poor user feedback simply because a lack of background knowledge. The issue is fairly common as it can be associated with all types of MIDI controllers - users must understand how the controller and digital audio software work in order to use them together effectively.

Interoperability Issues

With all MIDI controllers, there is a dependency on other software being configured correctly so that MIDI commands can be successfully sent to the DAW. There is a risk that the solution can produce commands that may not be compatible with certain digital audio software. This is a constraint that all MIDI controllers/software face and it is hard to control.

Gesture Controlled Musical Conducting

This risk will be minimised by performing extensive research into the most popular commands that are supported by a selection of digital audio software so that an effort can be made to implement them into the solution.

Hardware Not Fit for Purpose

Over the course of the project, there is a risk that research into the Leap Motion Controller could come to the conclusion that the hardware is not fit for the purpose of the project. To combat any limitations of the hardware the official documentation and FAQs will be researched to understand the best practices and optimal conditions of the Leap Motion sensor - so they can be replicated for the project.

Lack of User Feedback

Due to time constraints it will not be possible to conduct extensive user feedback. This is a risk because the users being tested might not have representative views of a larger audience. The risk will be minimised by applying the Agile methodology to the project and consulting with the project supervisor at the start of each week. The weekly periods will allow for the enhancement of the user experience which will help to achieve the goal of making the overall system very robust.

Project Architecture

As there are several components that are required to configure the solution, there is a risk that the architecture might fail. Prior to full implementation, a proof of concept regarding the components of the project will be quickly implemented to check whether the architecture is feasible. If it turns out the architecture is infeasible, then it will save a lot of implementation time in the long run.

2 Background

2.1 The Problem

In order to determine whether gesture control is a viable platform for controlling music, a solution will be produced in the form of a gesture-controlled MIDI controller. Gestures are configured to map to special Control Change commands which are sent through to the Digital Audio Workstation; these commands are used to change and apply 'effects' to the music being played.

In order to see if the platform can be useful, a usability and user acceptance test will be conducted to form the experimental stage of the project. The usability of the solution will be a critical part of the project as the main test is how users can successfully interact with the gesture controller, in comparison to a standard controller.

If successful, the project will enable artists to use alternative way of making music where they will be able to express themselves through the use of their hands. The project is innovative as it seeks to combine the aspects of physical and digital music creation. To make it tailored to each user, commands will be completely customizable to make sure that each user can set up the software in a way that feels natural and comfortable.

2.2 Problem Context

In order to understand the problem background, a few key music-based concepts need to be explained to gain a deeper understanding of the context.

2.2.1 A History of Digital Audio

Digital audio involves using a computer to record physical musical sounds that are processed and stored on a computer as a sequence of 0s and 1s - called bits. The more bits in a recording (the higher the bit rate), the more accurate the sound (shown in Figure 1). The first digital audio medium was the compact disc, which was introduced in the early 1980s [9]. In 2004, worldwide sales of audio CDs, CD-ROMs and CD-Rs reached around 30 billion discs [10].

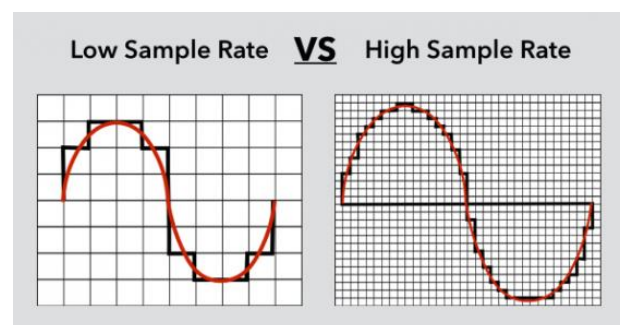


Figure 1 – Low sample bit rate vs high sample bit rate [55]

The process of recording digital audio normally involves connecting the physical instrument to the computer via a cable and playing the instrument so that the computer can record the sound in a digital format. These sounds can be loaded into software to be combined and mixed with other sounds to form a song. Software packages are widely accessible to any

user with internet access and have become widely popular among a range of audiences including serious artist and casual musicians.

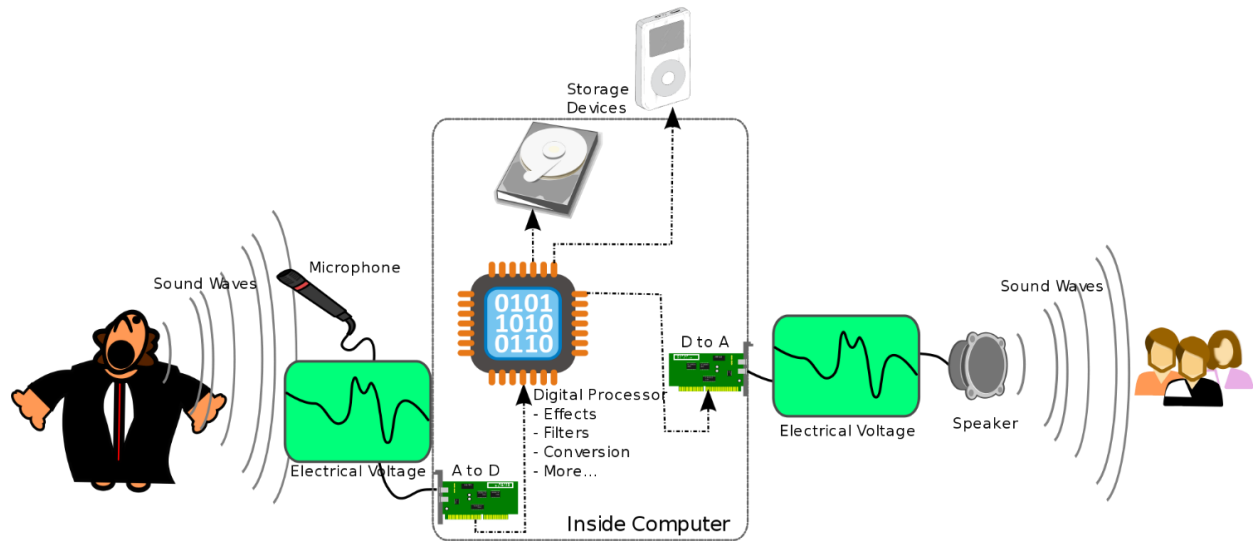


Figure 2 – Digital Audio Recording Process [56]

What does Figure 2 show?

- Complete flow of recording and playing digital music, from the initial input sound waves all the way to the output sound waves.
- Vocal sound waves being converted to voltage through a microphone connected to a circuit board inside the computer, which is subsequently converts the voltage into 0s and 1s by the computers processor.
- Data processed into binary form becomes stored inside the computer on a hard drive or can be loaded onto a digital audio device, such as an iPod.
- The stored music in binary-form can be converted back into voltage where it is output by a speaker.

2.2.2 Introduction to Gesture Control

Gesture control is technology usually associated with the detecting of a user's hand movements or facial expressions as an alternative input to the traditional touch screen or mouse and keyboard. Gesture control often requires a camera to record the user, the most popular example at present is a mobile application called Snapchat [11]. The application gets the user to raise their eyebrows to trigger facial filters that manipulate the users face in interesting ways; this feature is extremely popular and has helped the company reach 100 million active daily users.

Gesture Controlled Musical Conducting

Hand and facial gestures can utilise a camera for input but recently hand gestures have been captured using other innovative devices, this is to mostly enhance the degree of accuracy that can sometimes be lacking through the use of a standard camera. Devices such as the Gest glove (Figure 3) and Myo armband (Figure 4) are worn by the user and have a better degree of accuracy due to the large number of sensors built into the devices. The Myo armband is controlled with arm muscles [12] whereas the Gest glove is controlled with individual sensors on each finger [13].



Figure 3 – Gest Glove [57]



Figure 4 – Myo Armband [58]

2.2.3 Leap Motion vs Kinect

The two gesture devices identified for the project are the Leap Motion Controller and the Microsoft Kinect. Each device is analysed below and the most suitable is used to complete the project:

Leap Motion

The Leap Motion controller [14] is a small USB peripheral device. The sensor is slightly different to the previously described gesture devices; the main difference being is that it combines 'two monochromatic IR cameras and three infrared LEDs' to get an impressive amount of accuracy. The sensor is small in size and designed to capture data from a user's hands (Figure 5).



Figure 5 - The Leap Motion Sensor [59]

Microsoft Kinect

The Microsoft Kinect [15] is a highly advanced webcam peripheral. The device has many built-in sensors (RGB camera, depth sensor and multi-array microphone) that allow users to interact with either a connected device, either the Xbox One or computer. The dedicated sensors enable the user to control the software with voice commands or hand gestures; it is also able to detect the users full body, which is often used for games. The sensor also has a number of primitive functions that include a standard webcam or microphone.



Figure 6 - The Microsoft Kinect 2.0 [60]

Table 1 below is a breakdown of the comparison between the Leap Motion Sensor and Microsoft Kinect:

Feature	Leap Motion	Kinect
User Input	Hands only	Voice, hands & full body
Connects to	Computer	Computer & Xbox One
Connection Type	USB	Official Kinect Adapter for Xbox, USB converter cable needed for PC
Web-camera function	No	Yes
Microphone	No	Yes
Sensors	Two cameras and three infrared LEDs	RGB camera, infrared (IR) emitter, IR depth sensor and multi-array microphone
Field of View	120° vertical 150° horizontal	43° vertical 57° horizontal
Developer Tools	Leap SDK – Available in C++, C#, Java, Objective-C, Python, Javascript, Unity	Kinect for Windows SDK 2.0 – Available in C++, C#, Visual Basic, or .NET
Cost	£35.99	£104.99

Table 1 - Leap Motion Sensor vs Kinect Comparison

Based on the above table and the needs of the project, the Leap Motion Sensor was selected to be used for the project. The Leap Motion Sensor was selected for the following reasons:

1. Sensors on the Leap Motion can process input data from the user's hands, to a very high degree of accuracy.

2. The Leap Motion is a lot more affordable than the Kinect.
3. The SDK supports Java, a programming language that I am most comfortable and experienced with, which is also not supported in the Kinect SDK.
4. There is no need for all of the extra functionality that is possible with the Kinect.
5. The Kinect requires an adapter to be used with the PC, the Leap Motion only requires a USB slot which is present in most computers.

2.2 Relevant Theory

In order to understand how the separate components of the system work, it is important to understand some underlying concepts related to gesture control and the creation process of digital music.

2.2.1 MIDI

MIDI (Musical Instrument Digital Interface) is a protocol that enables instruments and Digital Audio Workstations to communicate with each other [16]. The protocol itself is around 30 years old, but it has kept evolving with High Definition MIDI [17] being developed in 2013. MIDI itself is not sound, it is a series of signals like 'note on', 'note off' and 'pitch'. With MIDI, it is up to the instruments to make the sound - for example a piano and guitar can play the same MIDI sequence and both will sound very different depending on the instrument. Typically, a MIDI system [18] is comprised of:

- **Synthesiser/Sampler** - an electronic musical instrument that generates the sound that gets recorded by the computer, usually this is an electronic keyboard but it can also be virtual.
- **MIDI Control Input Devices** - usually a keyboard with additional control such as sustain, pitch bend, modulation, after touch and other controllers. These devices can come in many forms and can also be virtual-based. Figure 7 is a cross between a MIDI Control Input Device and a Synthesiser, this due to the fact it can make sound through the Keyboard while offering additional MIDI input control through the sliders and knobs.



Figure 7 – Conventional MIDI controller [61]

- **MIDI Interfaces** - physical MIDI devices need to connect to a computer with a MIDI Interface, however as the project will be configured as a virtual MIDI device, this will not be required.

- **Channels** - the number of channels or streams of data that can be sent down one set of MIDI cables to one MIDI instrument. They work similar to television channels, we all have one aerial, but we only receive data for the particular channel we want to watch.
- **Sequencer** - either a stand-alone hardware unit or software running on a computer that enables recording, editing, or music playback. Sequencers usually come as software and are commonly found as part of larger applications that control all aspects of audio and MIDI.
- **Tracks** - used to separate different MIDI sequences within a sequencer.
- **Computer** - the main part of a MIDI system that controls the scheduling, synchronisation and recording of all data. The computer connects to all of the Control Input Devices, interfaces and sequencers (if hardware-based).
- **MIDI Control Output Devices** - MIDI can control a wide range of external devices, common uses include the controlling of lighting or robotics.

2.2.2 Types of MIDI Commands

MIDI commands can be categorised into different types (Figure 8):

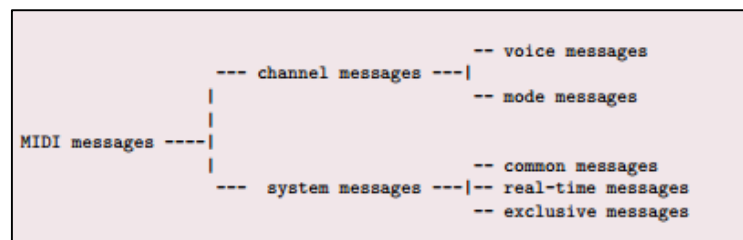


Figure 8 - MIDI Command Types

- **Channel Messages** - apply only to a specific Channel, so the Channel number is also included in the status byte for these messages.
- **System Messages** - not Channel-specific, so no Channel number is indicated in their status bytes.
- **Channel Voice Messages** - carry musical performance data, which makes up most of the traffic in a typical MIDI data stream.
- **Channel Mode Messages** - affect the way an instrument will respond to the Channel Voice messages.

Channel Voice Messages include: Note On, Note Off, Polyphonic Key Pressure, Channel Pressure, Pitch Bend Change, Program Change and Control Change. The Control Change commands will be used for the project.

2.2.3 MIDI Control Change Messages

MIDI CC messages are used to control a wide variety of functions in a synthesizer, these include control functions like volume, pan, vibrato and modulation. Control Change messages also contain the Channel number so that they only effect the mapped channels. The structure of a message also contains the Control Change status, followed by the controller number, specifies the control value. The controller number ranges from 0 to 127 and identifies which function of the synthesizer is to be controlled by the message and the controller value determines the value of the controller number, which also ranges between 0 and 127. For a complete list of the controller numbers, see Appendix 1. Figure 9 below shows an example Control Change command:

Type	Channel	Controller Number	Controller Value
Control Change	1	#7 – volume	50

Figure 9 – Structure of a MIDI CC Command

For the project, gestures will map to the CC commands, so when a gesture is performed by the user it will cause a function in the synthesiser to change; the result will be an audio output that the user can hear.

2.2.4 DAW Software

A Digital Audio Workstation (DAW) is an electronic device or software application used for recording, editing and producing audio performances. DAWs are similar to sequences, but they have extra functionality such as the ability generate and record music.

DAWs generate audio through plug-ins that come in the form of virtual instruments; these can be applied to a MIDI sequence that is loaded into the software so that it will make the sound. The plugins can be very powerful and are both free and paid, paid plugins usually have a larger collection of instruments and effects compared to the free versions. DAWs are used for a wide range of uses including the production and recording of music, radio, television, podcasts and multimedia.

There are so many software packages that have their own features that come at different price points, so the selection of workstation is largely down to user preference as the software can be picked based on individual requirements. The software package Reaper will be used for this project but there are many similar packages that have the same capabilities, Albelton, Garageband, MuLab, Audacity, Steinberg Cubase, Apple Logic Pro X are considered some of the more popular workstations [19].

Within DAWs there is usually a maximum of 16 slots for individual tracks where MIDI sequences are played. Each track is associated with a Channel, which is a number between 1 and 16. Any external sources such as a MIDI controller must be set to the same channel to link the two together so that they can communicate and send commands to each other.

Gesture Controlled Musical Conducting

The DAW Reaper will be used for the project, it has been selected as it has a free version and one of the best interfaces. Reaper also makes it easy to map commands to actions which is important for the configuration stage of the project. The Reaper interface is shown below (Figure 10) and has five tracks open with a MIDI sequence in each:

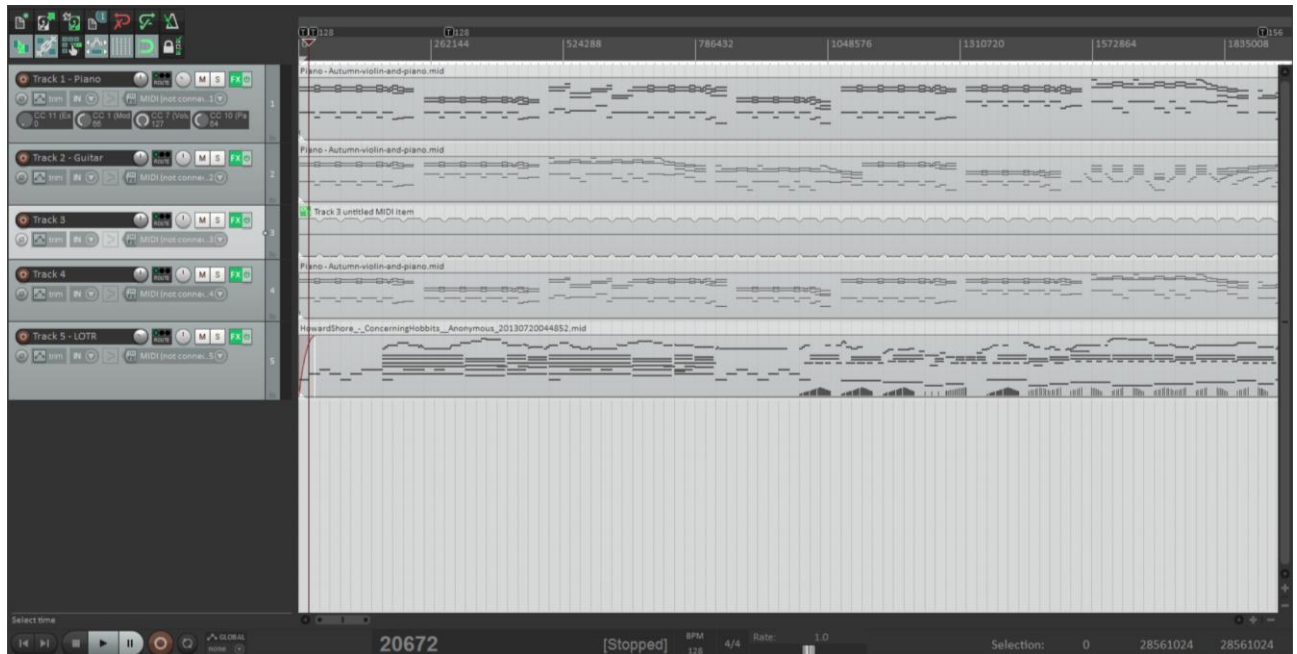


Figure 10 – Reaper User Interface

2.2.5 Leap Motion Controller

The Leap Motion Controller is a device that records hand-based gesture data with technology that combines two cameras with infrared LEDs. According to the official blog [20] this enables the device to ‘track infrared light with a wavelength of 850 nanometres, which is outside the visible light spectrum.’ Figure 11 shows how the user’s bone structure is detected through the sensors; this is how the device is able to achieve a greater degree of accuracy compared to standard cameras.

Raw hand data picked up from the sensors cameras is sent to the devices own local memory, the benefit being the processing load on the computer is reduced to get a quicker response rate. Smoothing algorithms are also performed within the local memory to gain better results from the raw data. The device is compatible with most PCs as only requires a single USB port to connect



Figure 11 – How the Leap Motion works [62]

2.2.6 Leap Motion API

The Leap Motion API makes it extremely easy to get processed raw data from the Leap Motion Controller. The Leap Motion detects a user's hands as a bone structure (shown above) and allows the developer to access useful classes such as: Arm, Bone, Gesture, GestureList, Hand, Finger, FingerList. Accessing classes such as Hand and Finger make it easy to implement a custom control system because you always have access to the user input.

The API is based around a co-ordinate detection system [21], each point in the bone structure (Hand, Arm, Finger etc.) has a co-ordinate associated with it, which is accessible through the API. The sensor tracks each object across the X, Y and Z axis, shown below:

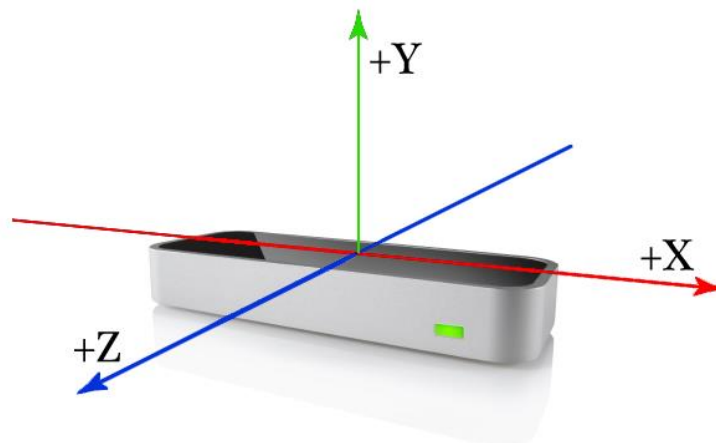


Figure 12 – The Leap Motion Co-ordinate Detection [63]

The documentation for the API is very well presented and it covers a large amount of programming languages including JavaScript, Unity, C#, C++, Java, Python and Objective-C; this makes it simple to implement the Leap Motion Sensor across a wide range of systems.

2.5 Existing Solutions

In recent years, people have been trying to find new ways of making music creatively using the latest technology [22]; normally this involves taking an interesting form of user input and enabling it to control musical effects.

2.5.1 Leap Motion Store

The music section of the Leap Motion store offers some interesting experiments that show off the capabilities of the Leap Motion Controller. The following applications on the store showcase the different areas of music than can be controlled using the Leap Motion Sensor:

GECO MIDI

GECO MIDI [23] is currently on the ‘Top Reviewed Apps’ section of the Leap Motion App Store. It works by using the Leap Motion Sensor to send MIDI CC commands to a digital audio workstation as they are controlled in real-time. The User Interface (Figure 13) is well-designed as it provides a visual representation of the data that outputs the software. Using the interface, the user is able to control 16 different actions and set variables such as output channel, MIDI message, data offset, rest value and mute/solo. The software is currently free on the Leap Motion Application store and has good reviews.

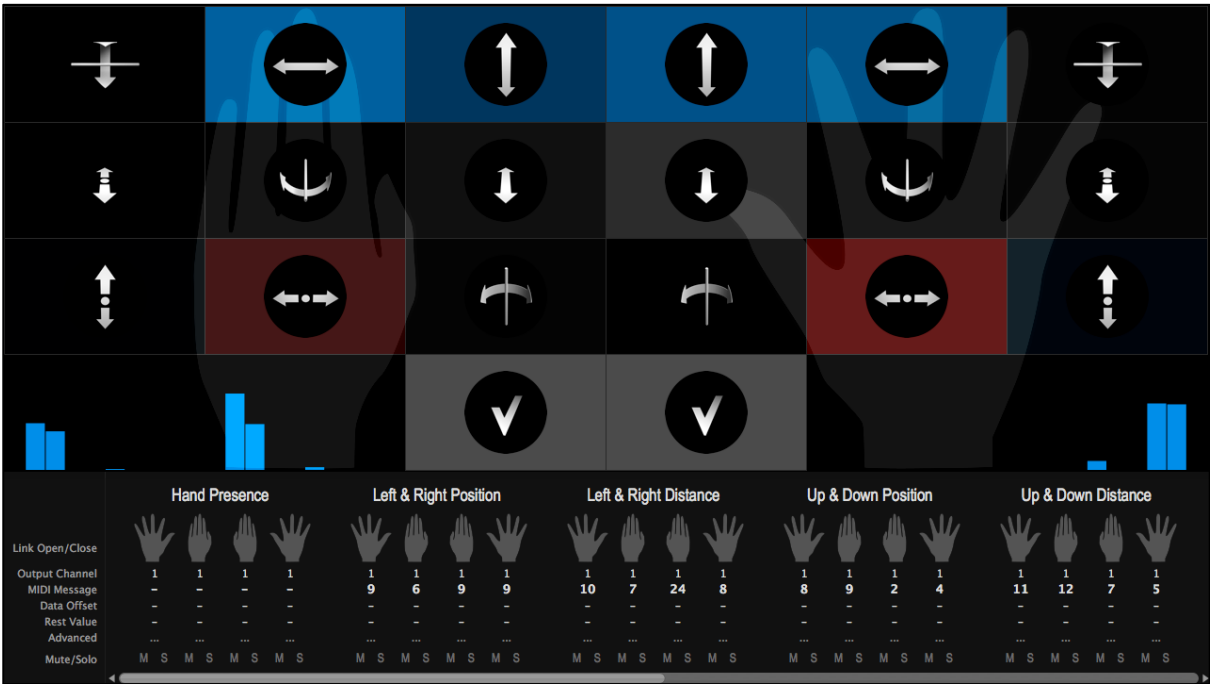


Figure 13 – The GECO MIDI User Interface [64]

GECO MIDI could be considered similar in terms of the overall functionality, however the one problem with GECO is that it is completely real-time to what the user is performing. The limitation of being solely real-time means that the user is unable to perform commands that require time to record user input before the resulting action happens. An example of a

delayed command is tap tempo where the user is recorded over a few seconds to determine the current beats per minute of the selected track. The solution implemented will have the aim of allowing the user to perform a mix of real-time and delayed commands such as tap tempo to allow them more control over the workstation.

Muse

Muse is an application on the Leap Motion Store [24] that is available for Windows and OSX; it is designed to enable the user to create ambient music through an interface where a user selects 'cubes' that are programmed to different ambient sounds. Users hover over cubes using the Leap Motion, and open their fingers to trigger the sounds. The application also enables users to control timbre, echo and reverberation. The interface for Muse is shown below in Figure 14.

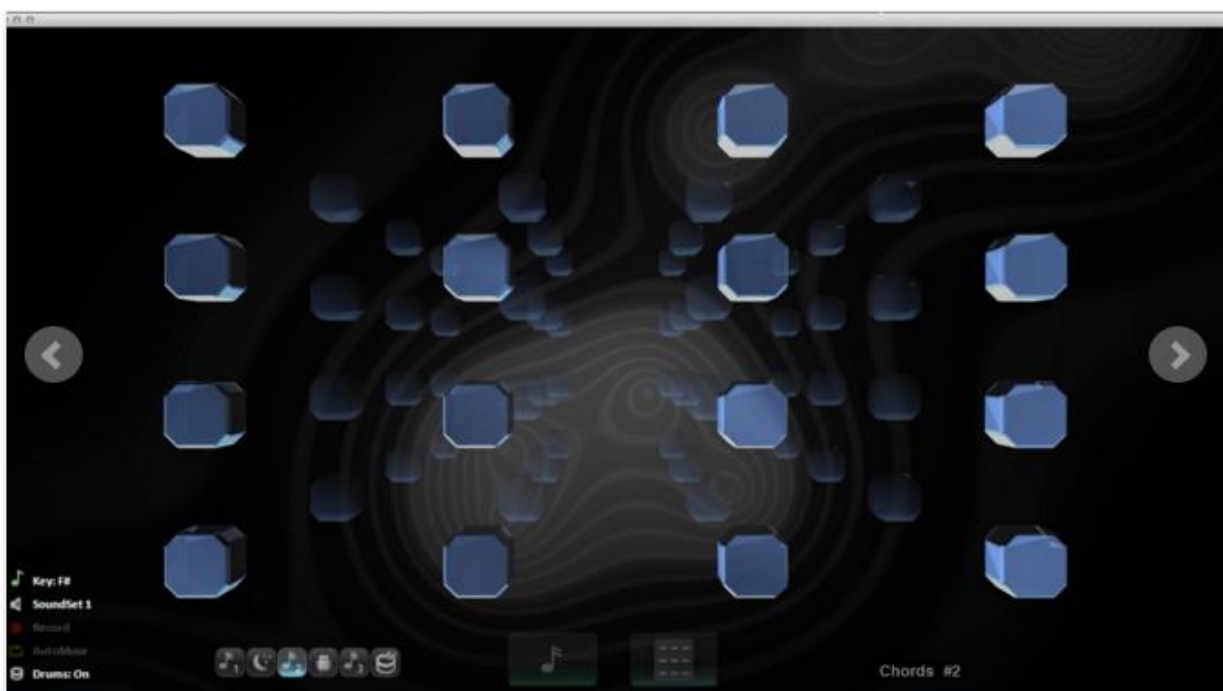


Figure 14 – The Muse User Interface [65]

Tekh Tonic

Tekh Tonic is another popular application on the store [25]. According to the store description “Tekh Tonic is an OSC/MIDI control interface reimagined for today’s high-tekh motion capture input devices. We combine the highly accurate input data from the Leap Motion Controller with the unlimited possibilities of a 3D physics engine to give you a totally new way to control MIDI and OSC devices”. The application features 6 different physics-based controllers and can be used with any device compatible with MIDI.

Swoosh

Swoosh is an add-on for iTunes, it allows the user to simulate vinyl control [26]. As a song is being played a user can reach in and control track playback, audio effects and looping. It enables users to directly connect with iTunes, allowing users to upload Playlists. Despite

looking well-designed, there are many store reviews where users have complained about a lack of documentation or usability issues. The interaction with Swoosh is visualised in Figure 15.



Figure 15 – Swoosh User Interface [66]

2.5.2 Other Existing Solutions

Smart TV

Gesture control is now commonly used in Smart TVs. Users are able to change channels, turn the volume up, change inputs or browse the TV guide using their hands.

Remidi Glove

Another product that combines gesture control with digital audio is a glove by Remedi [28] (Figure 16). It is a Bluetooth enabled MIDI controlled glove that has recently been crow-funded on Kickstarter and is now available for pre-order. The gloves work by assigning fingers to actions so when the user taps a surface it will perform the MIDI command. Use cases for this are very similar to the Leap Motion sensor, however to have controllers for both hands it costs \$669.00 for a pair of gloves so cannot be classed as being in the same price bracket as the Leap Motion Sensor (£35).



Figure 16 – Remedi Glove [67]

2.7 Tools

There are several tools that have been identified will be used to produce the solution:

- **Java 1.8** – Java has been identified as the language to be used for the project development, this is due to experience with the language from previous projects in University and from an internship.
- **Leap SDK** - the Leap Motion Software Development Kit [29] is used to link the input from the Leap Motion Sensor to the application code. The API provides several classes that will enable the implementation of the control system.
- **GitHub** - a code repository tool [30] that will be used to manage the code of the project. The project code will be in a private repository that will become public once the development is complete. The code will be available at: <https://github.com/Kee43/Gesture-Based-MIDI-Controller>.
- **Reaper** - the chosen DAW [31] that will be used to test the project. A sample project will be created that shows the capabilities of the solution developed. A free version of Reaper will be used for the first 60 days of use, if the development time exceeds this then a paid version will be purchased.
- **loopMIDI** - an application that connects the solution to the DAW through a virtual port [32]. This port acts as a pipeline that enables the solution to send MIDI CC commands that will affect the mapped functions within the DAW. A port must will be set up within loopMIDI and this port needs to be mapped in the DAW and solution so that the two components can successfully communicate.

- **WindowBuilder** - a powerful and easy to use bi-directional Java GUI designer [33]. The tool is composed of SWT Designer and Swing Designer which will speed up the development time. Figure 17 below shows the native drag-and-drop style interface that generates the interface code automatically behind the scenes.

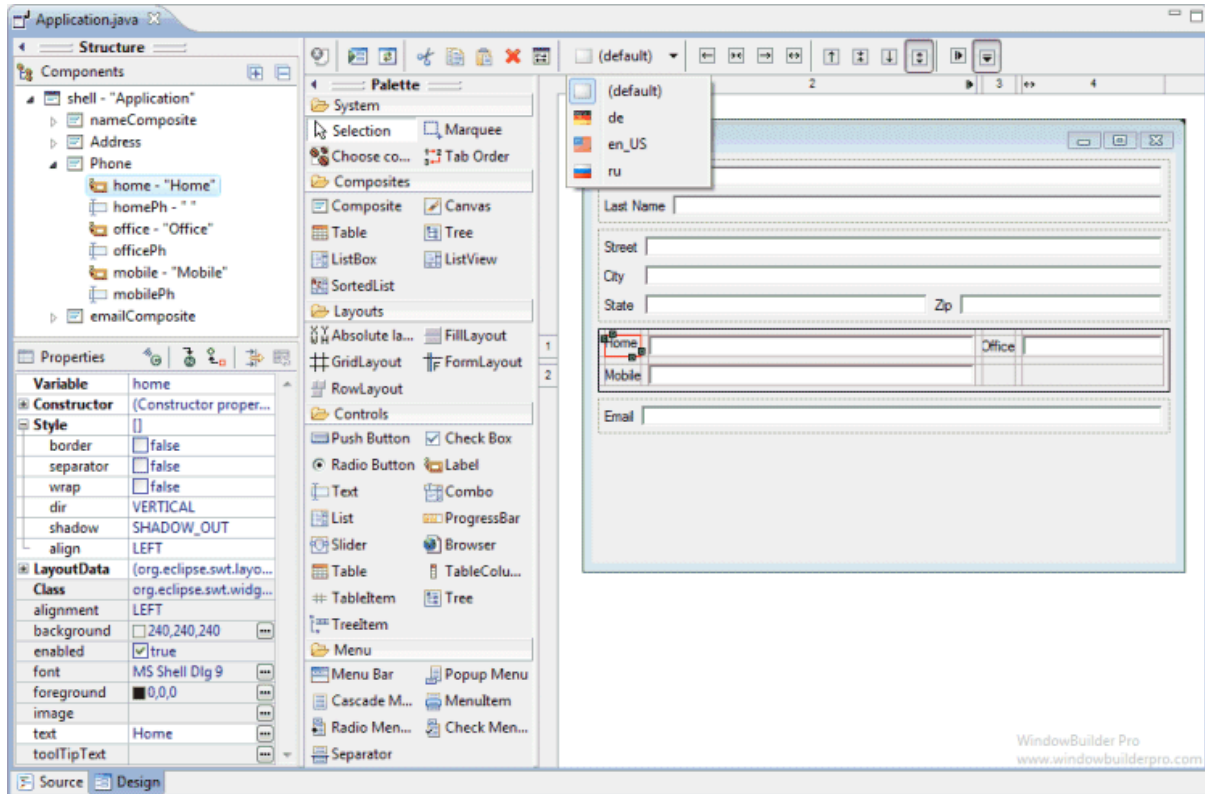


Figure 17 - The WindowBuilder interface [68]

- **Balsamiq** - a mock-ups tool with a well-designed GUI [34]. The tool can be accessed in a web browser or downloaded as a standalone application. I have used this tool in past projects and it will be used to create the interface designs of the solution.
- **Gliffy** - a similar tool to Balsamiq, except it is designed for the creation of diagrams such as system architectures or flow charts [35]. It is only accessible through its online web application and will be used to design the flow charts showing the user journey through the system.
- **Creately** - is another web application that is very similar to Gliffy [36]. This tool is better suited to creating class diagrams, so this will be used to create the class diagrams for the design phase of the project.
- **Eclipse** - the Developer Environment that will be used to program the application [37].

3 Specification and Design

3.1 System Overview

This section will briefly describe the idea for the solution that will later be designed in future sections. The overall system will be composed of three main components:

1. The 'Solution'

The solution will be the part of the system that is developed. It will be a virtual MIDI input controller and will be responsible for mapping the detected gestures from the Leap Motion into CC commands that will be sent to the DAW software. Once mapped correctly, the gestures will have a direct effect on the performance within the Digital Audio Workstation.

2. Reaper

Reaper is the selected DAW that will be used to test and demo the overall system. MIDI CC commands generated from gestures in the solution will have an effect (volume, modulation, vibrato etc.) on the tracks being played in Reaper. The CC commands and track channels will need to be mapped to the solution from Reaper so that the two components can communicate through the pipeline.

3. loopMIDI

LoopMIDI will be used as the pipeline to connect the virtual MIDI input controller to the DAW. The pipeline is the virtual form of the physical cable that connects a conventional MIDI input device to a computer. Channels must be configured on both sides of the connection so that CC commands are sent and received from the different end points.

3.1.1 The Solution

As a comparison, Figure 18 shows what a standard MIDI system looks like and Figure 19 shows how the solution will fit into a similar MIDI system.

The main difference between the new and old systems is the amount of virtual components. Normally in a standard MIDI system the Audio/MIDI interface is a piece of external hardware that has many ports to enable several MIDI devices to connect a computer. As the only MIDI controller will be virtual, it requires a virtual port to connect to the Digital Audio Workstation.

In Figure 18 it is shown that the audio/MIDI interface and MIDI controller (keyboard) are both pieces of hardware. In the new system (Figure 19) the MIDI interface and controller both become virtual.

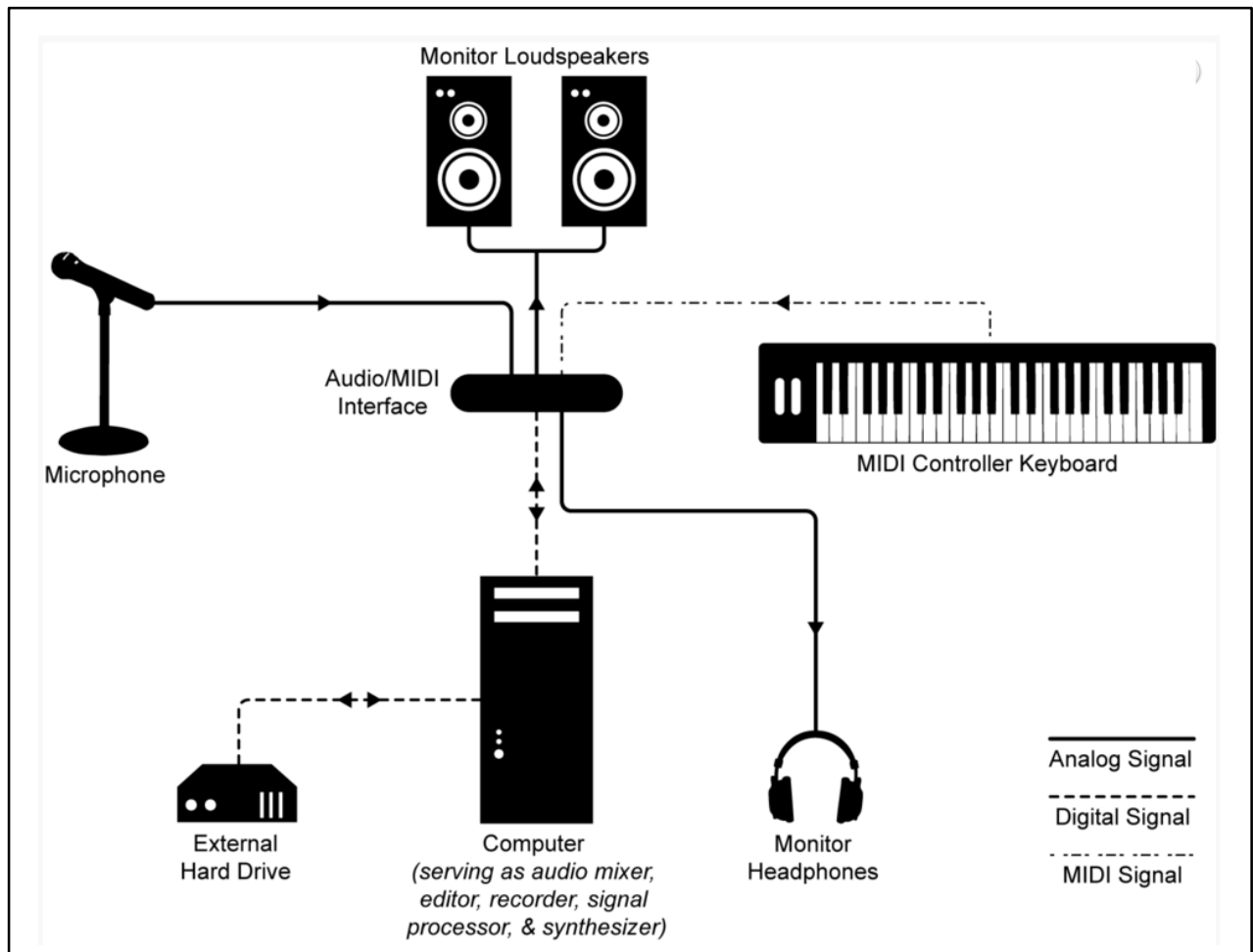


Figure 18 – The Standard MIDI Setup [69]

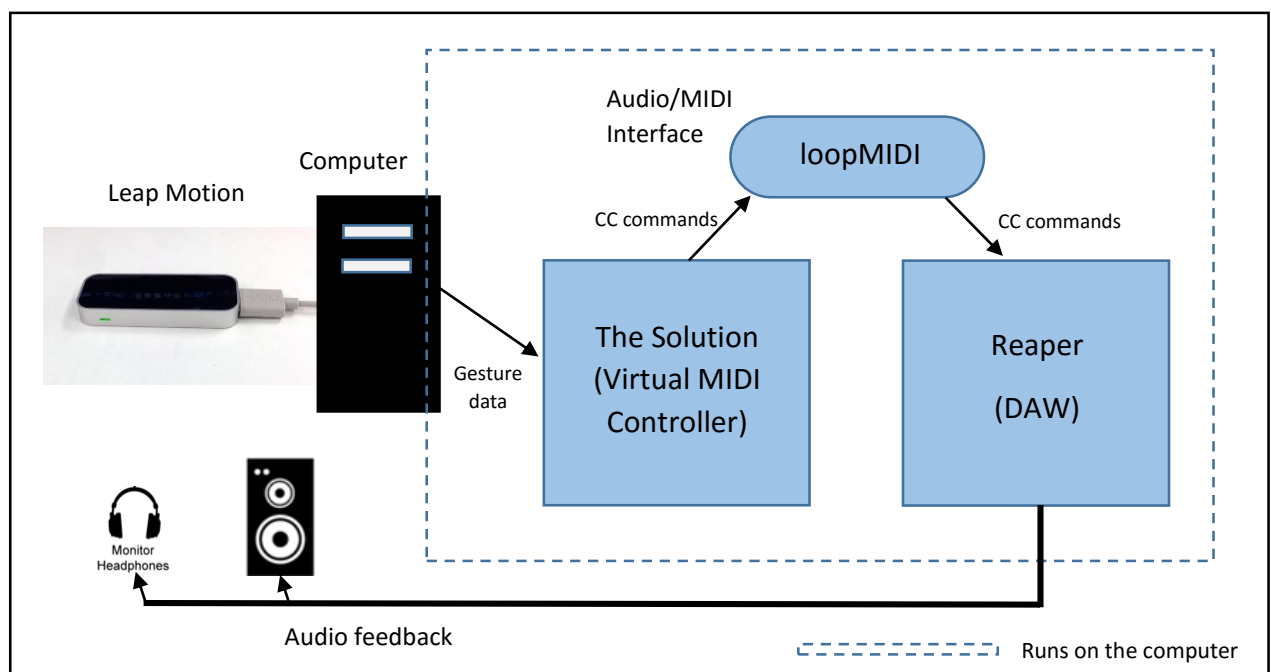


Figure 19 – The New MIDI Setup

3.1.2 Software Overview

Key features of each software component:

The Solution -

- Virtual MIDI controller that runs on a computer.
- Connects to the Leap Motion controller through a computer.
- Converts gestures from the Leap Motion to MIDI CC commands.
- Sends MIDI CC commands to loopMIDI.
- 6 action slots for CC commands per track.
- 16 tracks in total with an additional slot for global commands.
- Displays real-time Leap Motion data.

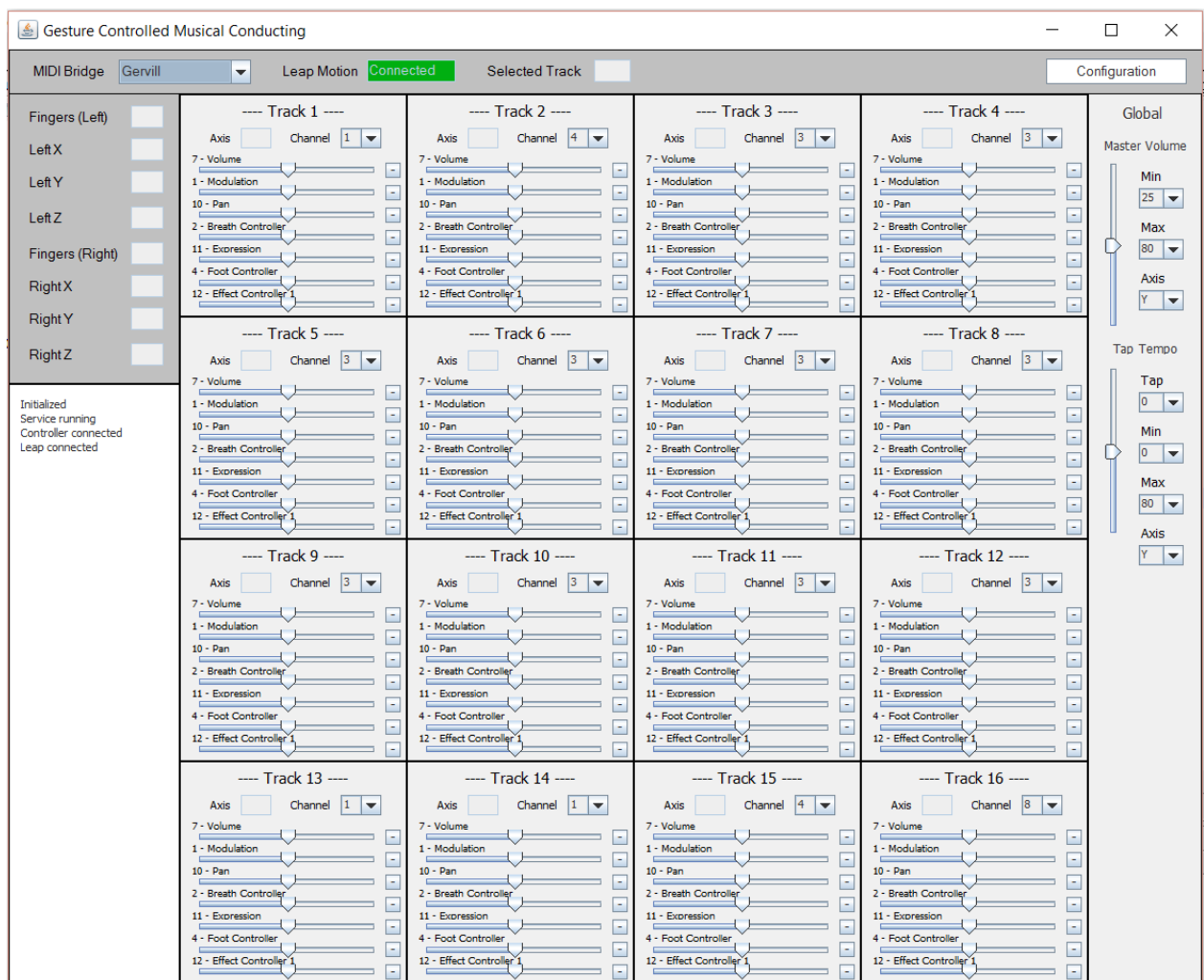


Figure 20 – Screenshot of the Developed Application

loopMIDI -

- Virtual audio interface that replaces the need for external hardware.
- Connects virtual MIDI compatible software together.
- Recieves MIDI CC commands from the solution and forwards them on to the Digital Audio Workstation.

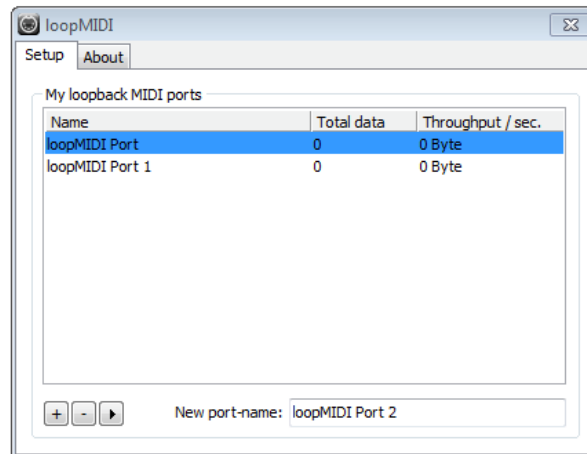


Figure 21 – LoopMIDI Interface [70]

Reaper -

- Digital Audio Workstation that combines an audio mixer, synthesizer, editor, signal processor and recorder into one peice of software.
- Used by professional artists to produce digital music.
- Recieves CC commands from loopMIDI which are mapped to 'effects' that alter the performance playing.
- Sends audio feedback back to the computers speakers or connected headset.



Figure 22 – Reaper Interface [71]

3.2 User Requirements

The aim of this section is to determine the features that the system needs to incorporate. They have been split into functional and non-functional to show the difference between what the system should do and how the system will work. The requirements have also been split into the categories 'must have', 'should have' and 'could have' to show the levels of priority. The requirements have been planned before any implementation so that it can be reviewed once implementation is complete to analyse how successful the produced system is at meeting the requirements.

3.2.1 Functional Requirements

Must Have -

- The system successfully connects to and reads input from the Leap Motion Controller.
- The system sends the correct CC command based on the user selection and gesture.
- The system will be recognized as a virtual MIDI device all compatible digital audio software.
- The system will send MIDI CC commands (volume, modulation etc.) to digital audio software through the MIDI bridge tool.

Should Have -

- The system will save user preferences so that they are retrievable next time the user opens the application.
- MIDI commands for the system are configurable.
- The system has local actions for each track as well as global actions that apply effects to the entire performance.

Could Have -

- The system has an advanced configuration for each MIDI command such as axis changes or custom gestures.

3.2.2 Non-functional Requirements

Must Have -

- The program must run on the Windows operating system.
- The system window must be small enough to see both the system and the digital audio software.

Should Have -

- The controls for the system should feel natural and be easy to learn by the average user.
- The user interface of the system should be simple enough to be used by a user with low computing skills.

3.2 System Design

The purpose of this section will be to describe the approach taken to design the system. As one of the system's most important aims is to provide a good user experience, it will be an underlying factor taken into consideration while each part of the system is designed.

The following three components have been defined as the most important features that need to be designed with the user experience in mind:

1. Hardware

As the hardware being used for the project is a relatively new platform, I cannot expect each user to have a great deal of experience using it. The system will be designed with this in mind, so it will seek to provide the user with a smooth and simple experience, while using the hardware to its potential. There will be features within the interface that update to inform the user that the hardware is working as intended.

2. Controls

The control system will be designed with the user interface in mind, so to make the controls as easy as possible, the user interface will provide essential feedback. The controls for the system will also be affected by any possible hardware limitations that are present in the Leap Motion Controller so these will need to be taken into account during the design.

3. Feedback

The UI will be designed with the thought in mind that user feedback needs to be constant. This is an important feature as it will allow the user to feel that they are in control at all times. Feedback will be sent to the user through the user interface as well as the audio response from the digital audio workstation.

3.3 System Flow

Figure 23 describes the typical user interaction flow when using the system; it illustrates the interaction between the systems three main components: the solution, digital audio workstation and loopMIDI software.

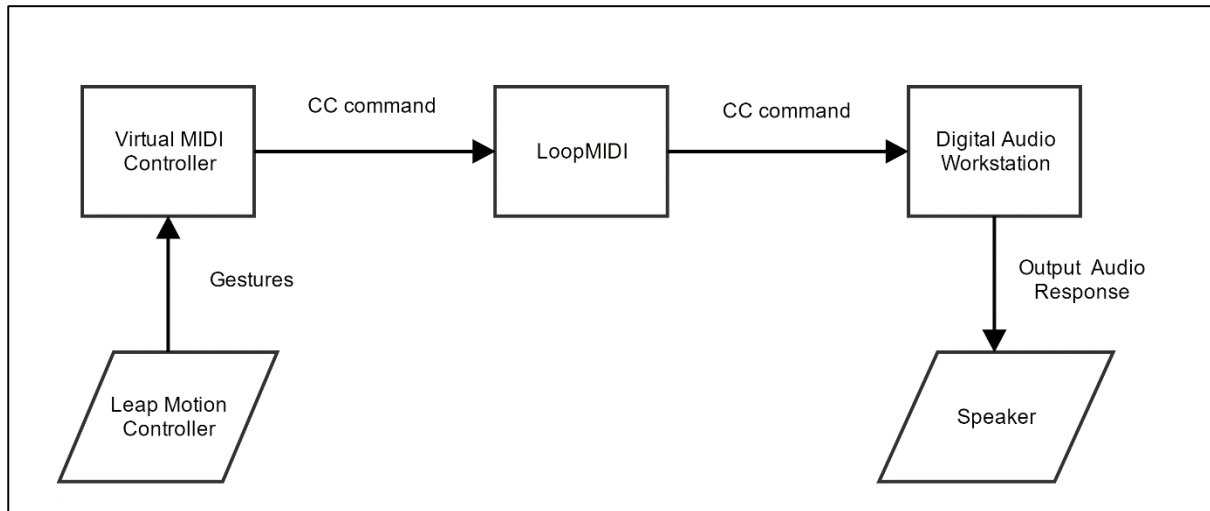


Figure 23 – High Level System Flow

What diagram 3 shows:

- **Leap Motion Controller** - the gesture controller detects the user's hands and processes the raw data by performing smoothing algorithms built into the devices CPU. The data is then sent to the virtual MIDI controller (solution) through the wired USB connection.
- **Virtual MIDI Controller** - the solution to be developed. Using the Leap Motion API, the solution will map the raw input data received from the sensor into actions that will be mapped to CC commands.
- **LoopMIDI** - responsible for the virtual pipeline connecting the MIDI controller to the DAW. The CC commands from the MIDI controller are sent along the pipeline to the DAW, each endpoint must be mapped to the port set up within loopMIDI so that they can send and receive the commands.
- **Digital Audio Workstation** - the software playing the MIDI sequences across separate tracks. Each track will be assigned a channel so that individual commands can be sent to each sequence. Each sequence will be assigned an instrument so that different effects can be tested and performed from the MIDI controller.
- **Speaker** - will output the musical effects being manipulated within the performance in the DAW. The speaker can be an external peripheral, built into the computer or part of headphones being worn by the user.

The second diagram below is a more detailed version of the previous flow diagram. The dashed red lines show the separation between the components of the system:

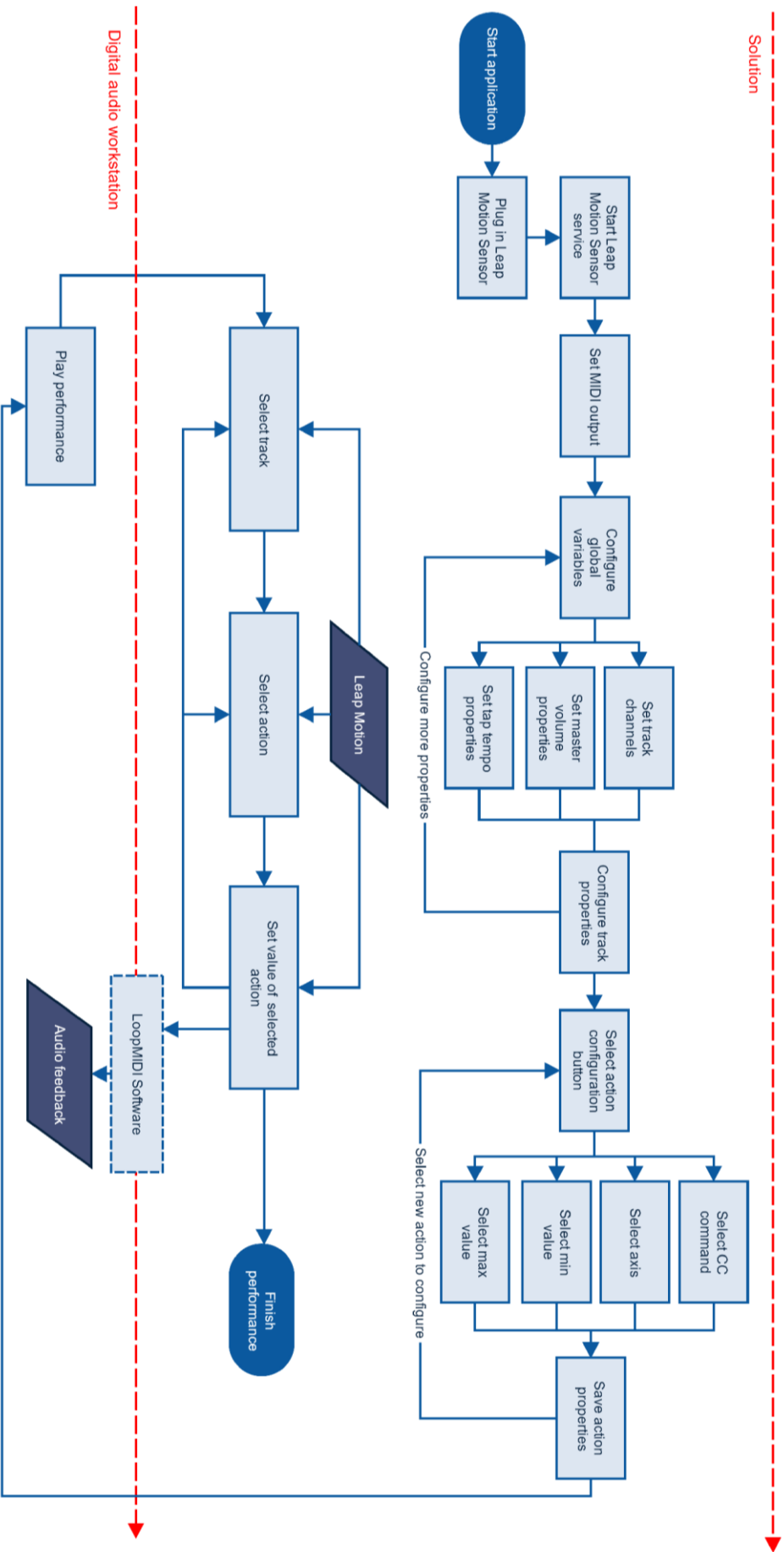


Figure 24 – The Detailed System Flow

Diagram 3 shows:

- **Leap Motion Setup** - to begin using the Leap Motion Sensor, it must be plugged into a USB port and the background service must be started to begin detecting data. From Windows, I managed to start the service from the 'Services' panel of the Windows Task Manager.
- **MIDI Output** - the application needs to be mapped the port created in loopMIDI. So that the commands can be sent through the pipeline.
- **Configuring Global Actions** - the global actions are functions that effect the entire performance. The global actions still need to be mapped to a channel, however other features such as minimum and maximum values are optional.
- **Configuring Local Tracks** - this involves configuring the optional six slots per track that are mapped to actions. Each can be configured by changing the CC command, axis, minimum and maximum values.
- **LoopMIDI** - connects the solution and DAW, which is represented in the diagram. The CC commands sent from the solution get passed along the loopMIDI pipeline to the DAW, which provides the audio feedback.
- **Play Performance** - this action is done within the DAW. Once the user is happy that the application has been configured and mapped to loopMIDI, the performance is played and this enables the tracks to be controlled from the solution.
- **Leap Motion Controller Input** - the user selects the track, action and action value using the Leap Motion Controller with the Leap Motion API acting as the interface between the controller and the solution. Depending on how the control system is implemented, the solution will map the gestures to the three variables before sending the corresponding CC command to the DAW.
- **Audio Output** - the actions performed within the solution have a direct effect on the performance within the DAW, this will cause an audio output response through the speakers connected to the computer.

3.4 System Architecture

This section describes the main architecture of the system. The class diagram below helps to define how the overall architecture works and how the separate components of the larger system interact to exchange data:

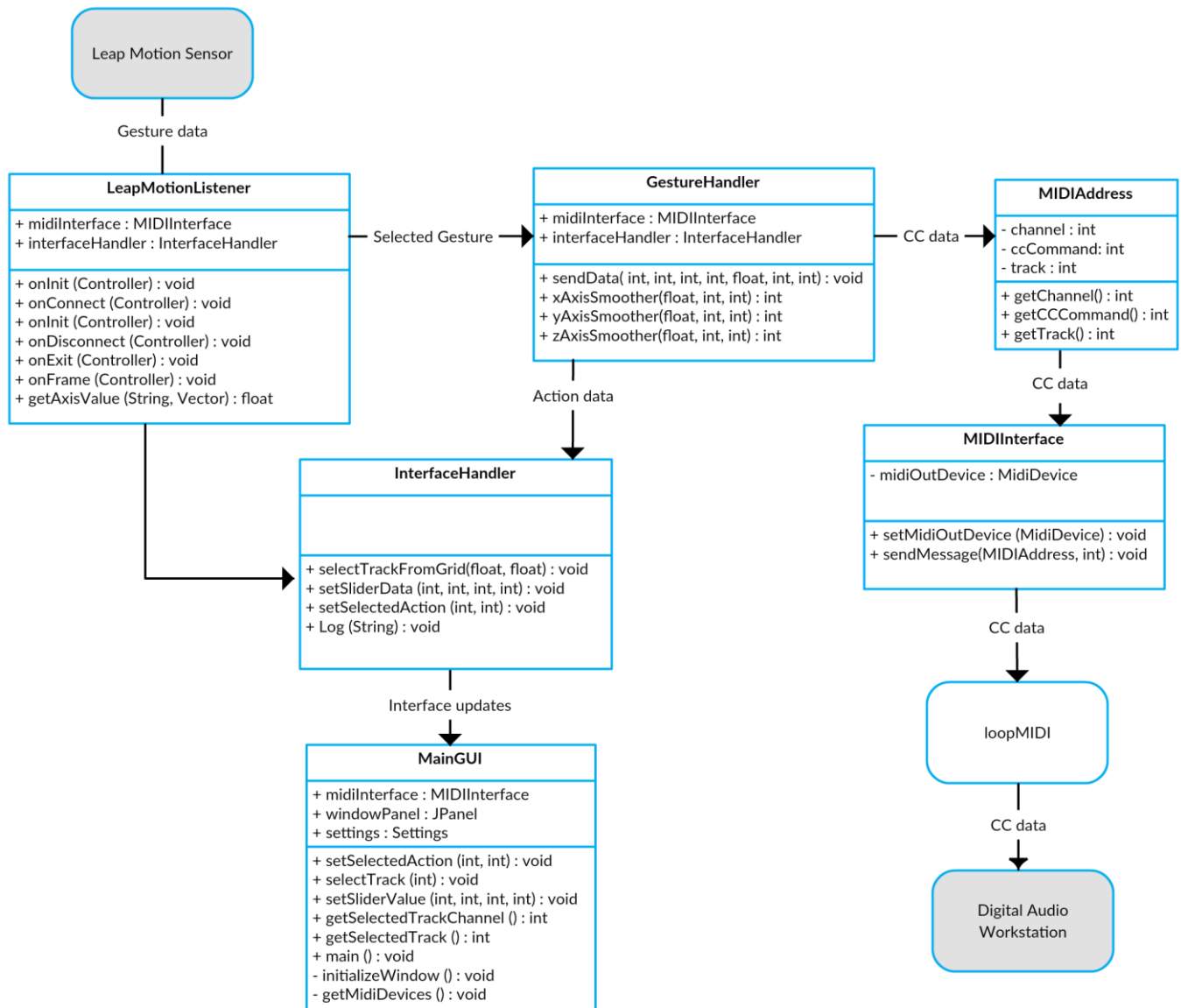


Figure 25 – The System Class Diagram

The above diagram is a combination of a class and flow diagram. It has been documented this way to make it easier to understand the complete architecture of the system by clearly showing the flow of data through the system.

Figure 25 shows some of the methods that will be implemented in the designed class structure; each class has its own responsibilities within the overall solution:

- **LeapMotionListener** – directly communicates with the Leap Motion sensor through the Leap API. If a class implements the Leap API it must implement methods such as `onConnect`, `onDisconnect` and `onFrame`. The overall responsibility of this class is to process the data that gets retrieved from the Leap Motion and send the data to the rest of the system where it is required. The LeapMotionListener also retrieves co-ordinate data from the Leap Motion across the X, Y and Z axis.
- **InterfaceHandler** – the GUI has a class that prepares the data being calling interface updates. Status updates regarding the hardware and gesture data is sent to the InterfaceHandler class so that the data can be prepared for being sent to the user interface. The class includes methods that perform analysis such as deciding which track slot needs to be selected based on the co-ordinate data sent from the LeapMotionListener class.
- **MainGUI** – represents everything the user sees. Once the background processing is performed, status updates are sent through the InterfaceHandler class to notify the user. Real-time updates are performed here through methods such as `setSelectedAction`, `selectTrack`, `setSliderValue` that highlight parts of the interface based on what the user has selected.
- **GestureHandler** – smooths co-ordinate data sent from the LeapMotionListener and turns it into the MIDI value (between 0 and 127). Minimum and maximum configurations are also taken into account here so if the set maximum value is 50 then the co-ordinate data will never translate into anything above 50 for the MIDI value. Once the MIDI value has been calculated it sends the data to the MIDIInterface class where value makes up part of the overall MIDI CC command that is sent from the application.
- **MIDIAddress** – designed to create the structure for the MIDI CC commands, they hold the channel, CC command and track data. The commands are sent as MIDIAddress objects from the GestureHandler to the MIDIInterface.
- **MIDIInterface** – responsible for sending the MIDI CC commands to the DAW through the virtual port. The MIDIInterface collects the MIDI Address objects and MIDI values from the GestureHandler class. MIDI data is sent from this class to the port that the user has selected in the User Interface.

3.5 User Interface Design

This section describes the design process that was conducted in order to achieve the final User Interface.

3.5.1 First Design

This component evolved the most as it went through a couple of iterations after discussions with the project supervisor. The task of the User Interface is to provide feedback to the user through all of the use cases. The first iteration of the UI was developed based on existing knowledge combined with research of MIDI controllers during the initial report.

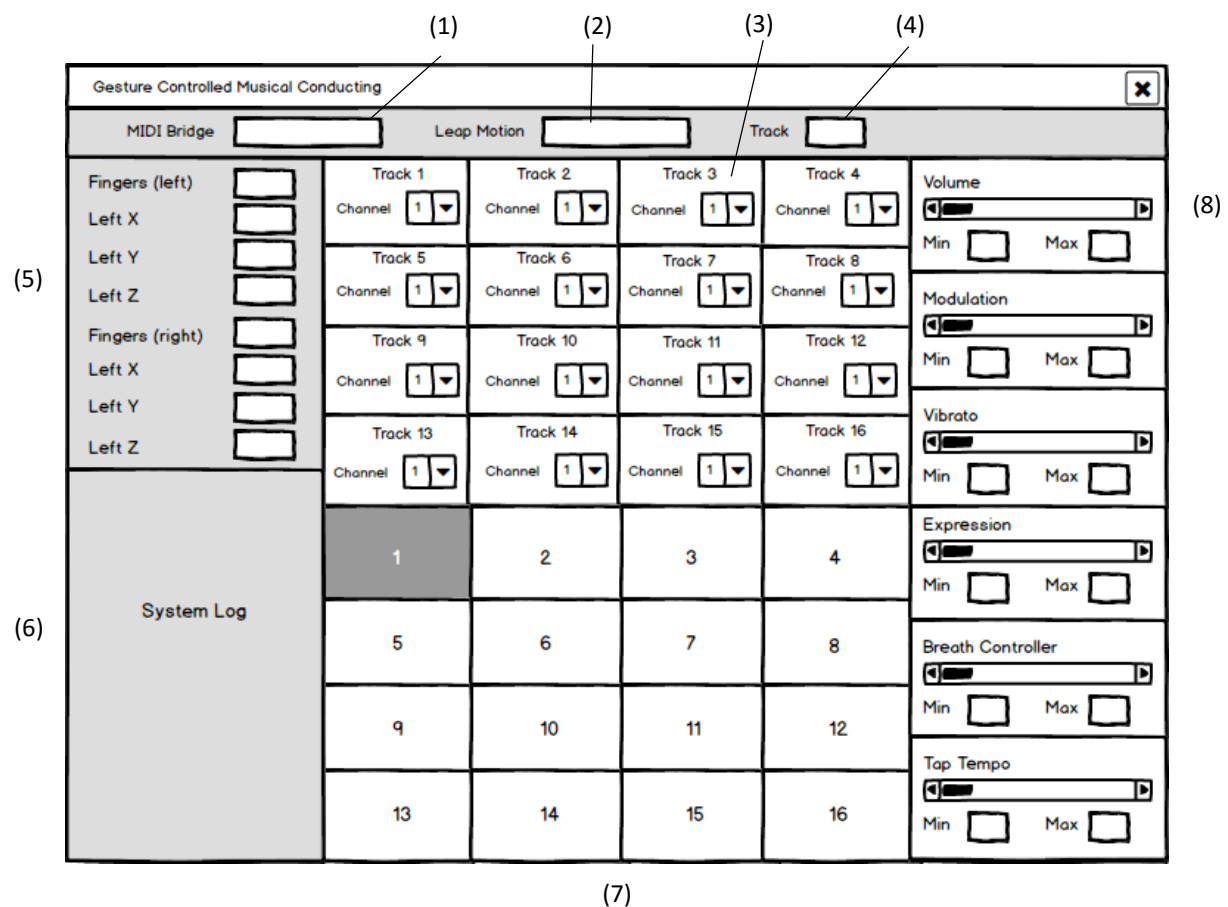


Figure 26 – User Interface Mock-up (first iteration)

Key:

- (1) MIDI bridge - user selects the MIDI output from a dropdown list
- (2) Leap Motion status – Current status of the Leap Motion Sensor
- (3) Configuration panel for the individual track
- (4) Selected track – shows the user's currently selected track
- (5) Leap Motion feedback panel – displays the current information being tracked
- (6) System log – the logs for the system are displayed here
- (7) Visually shows the current track selected (track 1)
- (8) Action panel – lists the actions available to the user

3.5.2 First Design - Interface Components

Main Configuration Panel

The task of the main panel is to provide the user with important information regarding the status of the overall system. It contains the MIDI output selection, Leap Motion status bar and selected track box. The MIDI output selection enables the user to determine which MIDI compatible device the application will send the MIDI CC commands to. The Leap Motion status bar has been added to the interface as it is an easy way of telling the users will let the user know if the Leap Motion Sensor is unplugged or if it is ready to receive input data.

Track Configuration Grid

Each track has its own section in the grid so that the user can easily identify which channel is assigned to each track. The channel needs to be aligned with the same track set in the digital audio workstation so that the MIDI CC commands can be mapped to the correct instrument.

Leap Motion Data Panel

The Leap Motion Sensor panel displays relevant data related to the sensor tracking. The number of extended fingers on each hand and the X, Y, Z coordinates of each hand are shown as a reference if the user needs to understand what is being detected by the Leap Motion Sensor.

Actions Panel

The right hand panel shows the actions that can be performed once a track has been selected by the user. The ability to restrict the CC through setting a minimum and maximum value was implemented after research suggested it may be useful to some artists who want to restrict the range of possible values where necessary.

System Log

The system log displays all of the other system data that is not shown in the interface. This includes displays information such as:

- Current action selected.
- Other gestures detected by the Leap Motion.
- Error logs that may be helpful to the user.
- MIDI values being sent out of the application.

3.5.3 Required Changes

After consultation with the project supervisor and further research, it was decided that the first design needed some changes, these were:

1. Track Actions

After further research into how existing MIDI controllers work, it was apparent that there was a need for individual control actions per track. This is to eliminate the fact that if two tracks may use the action the value would jump, for example if two tracks are being used, the first track could be used to set the value to 60 and if the second track requires the same action then the slider would have to start at where the last track finished using it. By enabling each track to have its own slider means that control of each track will be a lot smoother.

2. Global Actions

I came to the understanding that the global actions should be factored into the interface. This would enable a critical feature of controlling the entire performance compared to the individual tracks. There is often a need by people using digital audio workstations to control the global performance through actions such as master volume or controlling the beats per minute through an action called tap tempo, where a user presses a key in time with a beat to set their own tempo to the performance.

3. Action Slots

Initially, it was thought that the solution would cover the seven most popular actions in the CC table [38], so that at least some actions would be useful to some configured in the DAW. Revisiting the requirements showed that there was a need for an extensive array of customisation, so I made the decision to remove the concept of static actions and replace them with 'slots'. This means that the application will now be implemented so that the user can decide which seven actions from the complete CC list will be applied to the individual tracks. This is a significant change as it will improve the number of actions from 7 to 128, greatly improving the customisable options available to the user.

4. Configuration Table and Configuration Button

Due to the increased customisation, it became required that there should be an easy way of accessing all the applications configuration settings, so a button on the main configuration panel was added to allow the user to open the applications settings. This provides a way of configuring each track by being able to set the seven actions, as well as the minimum, maximum, and axis associated with each action.

A small configuration button was also added to each slider on the interface as an alternative way of dynamically changing the settings for each action slot.

3.5.4 Second Design

Once the required changes were taken into account, the result was the second design for the main interface:

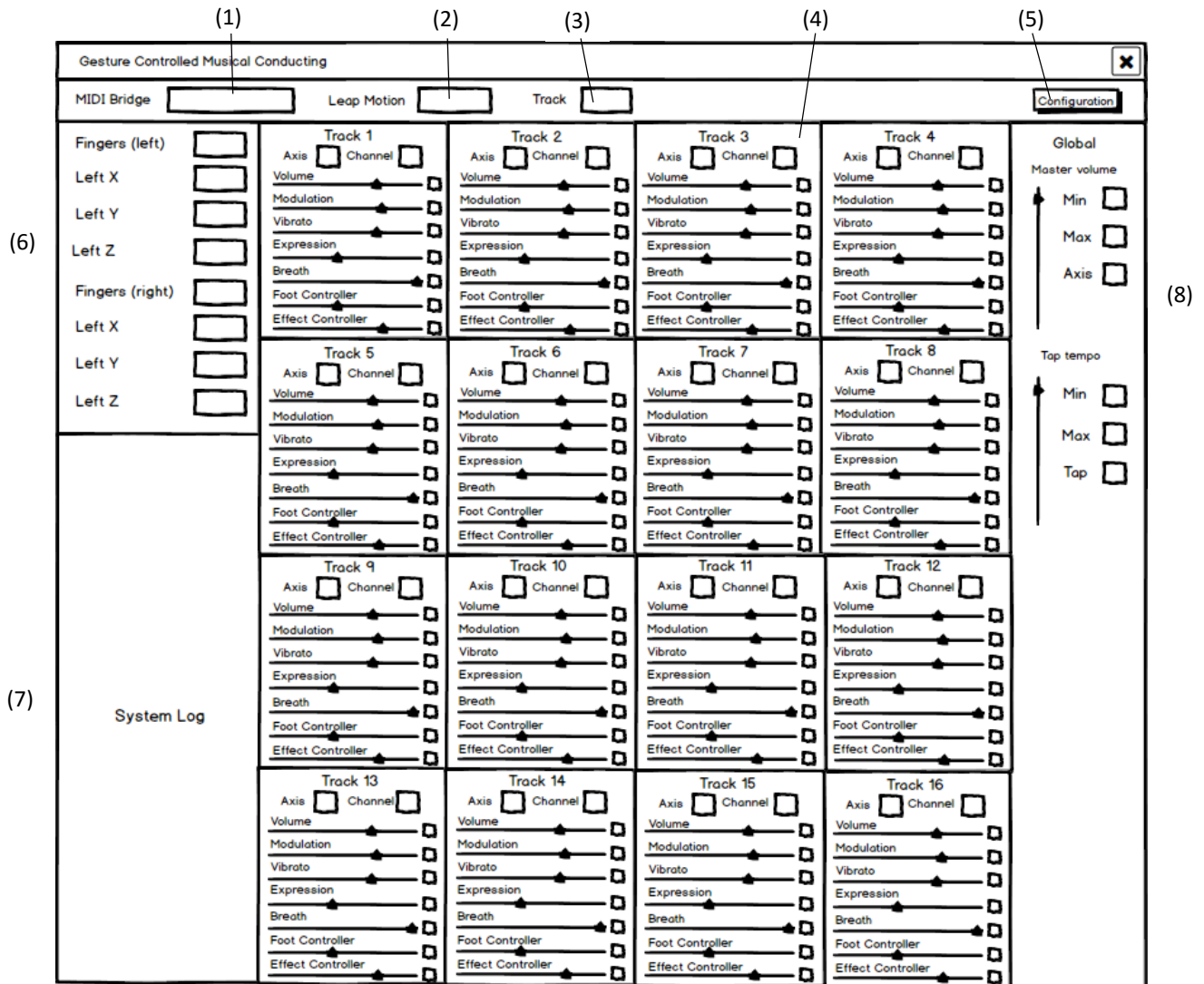


Figure 27 – User Interface Mock-up (second iteration)

Key:

- (1) MIDI bridge - user selects the MIDI output from a dropdown list
- (2) Leap Motion status – Current status of the Leap Motion Sensor
- (3) Selected track – shows the user's currently selected track
- (4) Configuration panel for the individual track
- (5) Configuration button- used to change the system settings
- (6) Leap Motion Musical feedback panel – displays the current information being tracked
- (7) System log – the logs for the system are displayed here
- (8) Global control panel – lists the global actions available to the user

3.5.5 Second Design - Interface Components

The second design was created to accommodate the flaws identified from the first design. Some components have remained the same from the first design but the noticeable differences include:

Configuration Button

The configuration button opens up a table with a list of configurable commands for the entire system. It allows the user to set the commands for each track by changing settings such as CC command, minimum value and maximum value.

Global Action Panel

Once this has been selected by the user, they are able to control the master volume and set the beats per minute for the performance. This is a feature that was important to add to the project to give the user more control over the overall performance compared to just individual tracks.

Track Design and Configuration

The biggest change to the second design was the way the tracks are controlled. To enable better control, each track now had seven slots that are configurable based on the instrument and preference. To enable the user to quickly customise each track, the button next to each slider allows the user to directly configure the associated slot. The axis text box has also been implemented to provide hints to the user regarding the axis they need to use to control the selected action.

3.6 Control Design

The other critical part of the system that had to be designed was the control system. This is by far the most complex component as it determines how the user moves their hands to select the track, action slot and CC value. The CC value is the measure of the slot and is set up to be between 0 and 100. For example, in the slot for volume, if the CC value is set to 100 it would send the CC command to change the volume parameter to maximum and the result would be the volume would be set to its loudest value.

A couple of control methods were discussed with the project supervisor to determine how to get the maximum control out of two hands while maintaining a simple control system.

3.6.1 Binary-Finger System

The first control method discussed was to map the right hand as a binary representation to select the tracks, with the left hand being used to select/control the action and CC value parameters. As there are five fingers on a hand, each finger would represent a bit so the user can select the track number by extending the right sequence of fingers, with any value above 16 being used to control extra functions such as the global commands. As an example, Figure 13 and 14 show how a track will be selected.

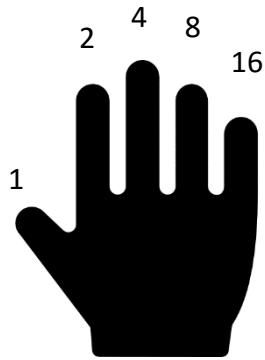


Figure 28 - Binary Representation



Figure 29 – Gesture to Select Track 2

The main issue identified with this design is that some combinations are not physically possible. Most people struggle with certain finger combinations because of the way our fingers are connected, such as extending the index and ring fingers together or the middle and ring fingers only. This mapping would cause problems with performing basic actions so the control system was considered infeasible to implement.

3.6.2 Interaction Box/Proximity Selection

Selecting the Track

Further research was conducted based on the topic of how to use the Leap Motion Sensor to interact with menus. After looking at the design guidelines within the Leap Motion's best practices online documentation an idea was thought of to use a combination of two techniques, an interaction box [39] and proximity-based highlighting [40]. An interaction box (visualised in Figure 30 below) involves creating a grid of a fixed size in front of the user's screen and using a finger to select a point within that grid.

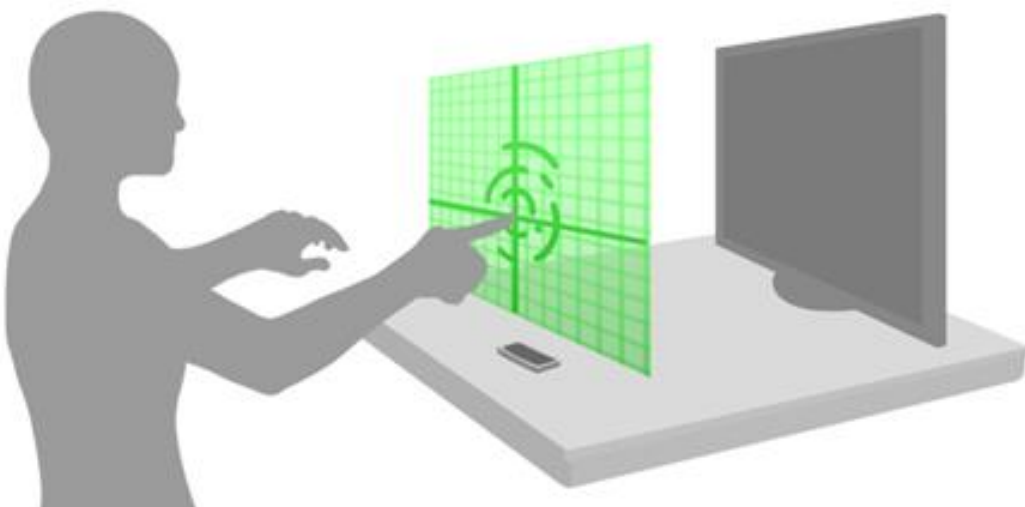


Figure 30 – The Interaction Box [72]

The technique of proximity-based highlighting involves selecting a panel from the grid based on the fingers proximity. This means if the finger falls into a boundary within the grid, then that panel is selected. In this system, the grid covers the entire screen but the track is selected with the right hand, so the interaction box only registers data from the left half of the grid. The grid has been split into a 4 x 4 grid to cover the 16 tracks, so when the finger from the interaction grid falls into a square boundary, the grid panel highlights to represent the selection. Figure 31 shows the visualised grid section of the user interface below, where track 6 is selected.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Figure 31 – Track Selection Grid

Selecting the Action

Once the track has been selected with the left hand, it was logical to use right hand to select one of seven action slots within the track. As a simple way of selecting the slot, seven gestures were implemented:



Action 1



Action 2



Action 3



Action 4



Action 5

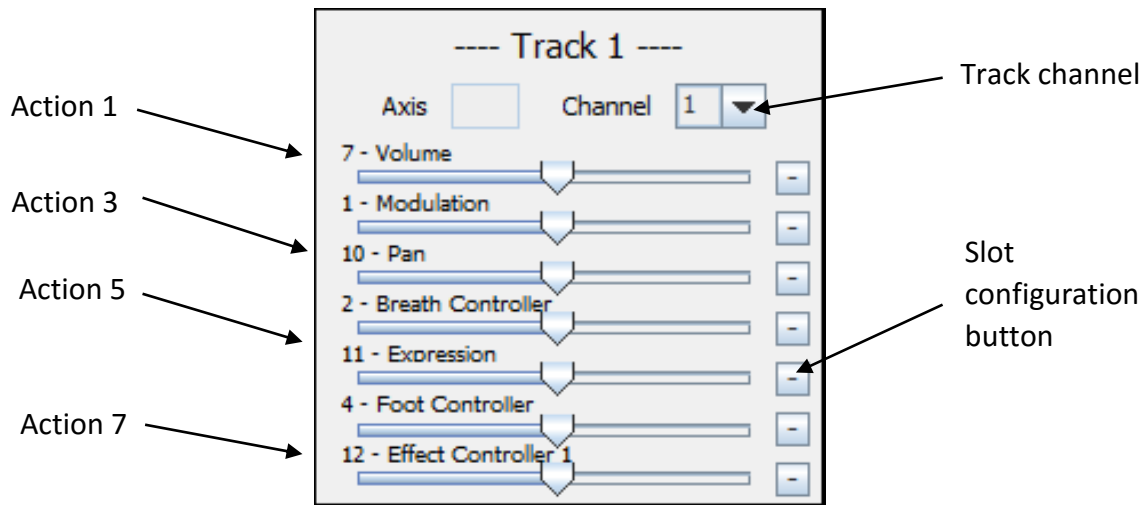


Action 6



Action 7

The gestures are mapped to each of the seven action slots within the tracks as follows:



Originally a single finger on the right hand was used to select action 1, however as the left hand also uses a single finger to select a track, complications may arise where the Leap Motion detects thinks the user is selecting a track by confusing the left and right hand. For the sake of usability, the action slots start with two fingers extended for slot 1 to make it easier for the application and improve the usability for the user.

3.7 Remaining Components

The remaining non-GUI components were designed and implemented as one unit through the implementation stage; this is because the two main components that affect the usability of the application have been designed in this section, so the remaining components are either part of the configuration phase or the handling of data behind the scenes.

3.8 System Constraints

The aim of this section is to describe the possible constraints that could limit the solution so that they are made aware of before the implementation starts. Constraints by nature are things that affect the solution but can be considered out of the developers control. Highlighting possible constraints will make the implementation stage easier as it will help to highlight what might not be feasible throughout the implementation. I have defined the questions below to summarize the constraints:

How will the design of the system affect a user's ability to learn the controls?

The user interface can help the user through real-time feedback, however users will be required to learn the gestures. Showing the gesture associated with each command on the interface could possibly help the user, but that feature would take up a lot of interface space which would conflict with the aim to keep the interface as small as possible.

According to a highly cited psychology paper [41], the number of objects an average human can hold in working memory is 7 ± 2 . As the number of gestures to remember will be around 6, it would be expected this would be no problem for the average user to remember with practice. Any memory issues that the user has would be considered out of control as it would affect their ability to use the system. As the system is designed to be easier to use with practice, it can be seen as cross between a controller and an instrument.

How do you know the system is being delivered in-line with the stakeholder's requirements?

As there is not enough time to perform extensive user feedback, the system has been designed after research and consultation with the project. The requirements have been discussed with the project supervisor, who is also a stakeholder. This has helped to be a good indication of whether or not the system is in-line with the requirements.

If the system meets all of the requirements but a round of detailed feedback suggests that the interface is not suitable, it will be considered out of our current control as the main target is meeting the present requirements. If there is time post-implementation, then the interface will be adjusted to make it more in-line with the stakeholder's requirements.

What if a particular digital audio workstation is not compatible with the system?

There is simply not enough time to test the application on every digital audio workstation, as there are hundreds available online. However, as MIDI is a universal standard, most Digital Audio Workstations are compatible with MIDI.

A very popular Digital Audio Workstation – Reaper, will be used to test the project and is very similar in terms of function to a lot of the other workstations so it will be expected that the solution is compatible with most digital audio workstations. The system has been designed to be compatible with all of the popular DAWs, so if it is not compatible then it will be considered out of control.

Can the system be supported on other operating systems?

Due to time constraints there will not be enough time to test the solution on other operating systems. The system is designed to work on Windows so if it doesn't run properly on other operating systems then it will be considered out of my control. This does not mean that the system could not support further operating systems post-launch.

Does the gesture controller have any hardware limitations that will affect the system?

The main limitation that will affect the Leap Motion Sensor is the lighting condition of the environment, these situations include bright sunlight or where there are bright light sources/reflective surfaces above the device. Users would be expected to meet the Leap Motion's guidelines on environmental conditions to use the hardware effectively. As it is a factor that is up to the user to configure it is considered to be a system constraint due to the fact it is quite hard to control from a developer point of view.

Will the average physically-capable user have any physical limitations regarding controls?

When the control system is implemented it will be designed for capable users with two hands, so any limitations they have will be taken into consideration. For example, it won't be expected that a gesture will include the index and ring fingers being extended because it is physically challenging. If the user has any other physical limitations that are not forced because of the system controls, then it will be considered out of control.

4 Implementation

4.1 Introduction

This section describes the implementation of the various components that make up the overall system. This section will also discuss how the concepts and ideas developed in the design section were later implemented as well as any problems that may have been uncovered during implementation.

4.2 Technical Background

As this project seeks to combine two major technologies, it was logical to split the project into four distinct modules:

1. Leap Motion Sensor input
2. MIDI transmitter
3. Control system
4. User Interface

After splitting the project, it was logical to code the overall solution as separate parts before connecting them together. After this the user experience was fine-tuned by implementing the configuration settings and user preferences. Before any coding was done, a plan of how the project would be developed was produced:

1. Implement the Leap Motion Sensor SDK to successfully get raw data from the controller.
2. Build the MIDI component to successfully send test MIDI data.
3. Implement the user interface based on the agreed designs.
4. Decide on an appropriate control method for selecting the track/actions.
5. Connect the control component to the User Interface.
6. Connect the control component to the MIDI component.
7. Implement user preferences to enable configuration of the system.

4.3 Tools Used

4.3.1 Language

The solution was written in Java (JavaSE 1.8) using the Eclipse Mars Integrated Development Environment [42]. Using one language for the project shortened the coding time as I have a lot of experience with Java and Eclipse from previous development experience.

4.3.2 Libraries/Add-ons

As Java is popular language, there are plenty of external libraries that support the language. There were three main libraries used for the completion of the project:

Leap SDK (Software Development Kit)

The Leap Motion developer kit [43] was used to link the raw input data from the Leap Motion Sensor hardware to the application. After installing the kit within a development environment it became easy to access the data through hand-based classes built into the Java development kit.

Java Sound API

The Java sound API [44] is used for controlling audio playback, audio capture, MIDI synthesis, and basic MIDI sequencing. The API was used to handle the MIDI side of the project as it allows the project to be set up as a MIDI transmitter that can be used to send the MIDI CC commands.

WindowBuilder

WindowBuilder [45] is an add-on to Eclipse that makes building GUI applications easier through its drag-and-drop style interface. It generates interface code making it a lot easier than writing the code manually.

Java Preferences

The Preferences library [46] built into Java enables the application to store small pieces of user preference data, this will mainly be used to improve the user experience.

4.3.4 External Software

Reaper

Reaper [47] is the digital audio workstation that will be used to play the music being manipulated. The interface to Reaper makes it easy to set up, however there is a lot of complex features for more experienced users. The solution being built will connect to Reaper as a virtual MIDI device.

LoopMIDI

LoopMIDI [48] is a free application that connects virtual MIDI controllers to any MIDI compatible software through a virtual connection, for this project it will connect the solution to Reaper.

4.4 Critical Components

The critical components of the system are the parts of the solution that are important to the operational functions. For the stages in which these were implemented see section 4.2.

4.4.1 Leap Motion Controller

Importing the Leap Motion SDK is extremely easy and it allows the application to access the data from the Leap Motion Sensor. Importing it into the application is the following line of code:

```
import com.leapmotion.leap.*;
```

The Leap SDK has several overriding methods that include the status of the controller, these allow for the hardware status to be visible within the User Interface through methods such as `onConnected()` and `onDisconnected()`, which is helpful to the user as they are able to have an awareness that the hardware is working correctly. The Leap Motion controller has its own dedicated CPU which allows for the smoothing and processing of the raw data collected from its Infra-Red cameras. Figure 33 below shows how a user's hands look from the perspective of the Leap Motion Sensor's cameras:



Figure 33 - What the Leap Motion Sees [73]

The SDK collects the processed data and makes use of pre-defined classes to enable the developer easy access to the data. These classes represent what the camera sees and have been used throughout the application in a number of areas.

Frames, Hands & Fingers

Each frame represents a segment of video being detected by the Leap Motion, it works exactly the same as a frame from a normal video camera. Each frame has its own set of objects associated with it that include: Hand, Finger, Arm, Bone and Gesture objects. The fact that the SDK already makes use of these objects means it is very easy to access the data within applications. The following code is all that was needed to access the Hand and Finger objects within the onFrame method:

```
for (Hand hand : frame.hands()) {  
    for (Finger finger : hand.fingers()) {  
        System.out.println("Finger = " + finger.type());  
    }  
}
```

Keeping the user updated on the position of both hands is a useful feature that helps the user check if they are being correctly detected by the hardware, each frame was used to detect the position of both hands using the following lines:

```
hand.palmPosition().getX();  
hand.palmPosition().getY();  
hand.palmPosition().getZ();
```

Once the input was correctly being received from the hardware it was tested using both hands to check if the data was correct. I was surprised at how easy the hardware was to configure and set up within the application as I had no issues.

4.4.2 LoopMIDI Configuration

Before any coding of the MIDI module, the external software loopMIDI was configured so that the project is able to 'see' the virtual port in order to transmit MIDI CC commands. The job of loopMIDI is to receive the command being sent from the solution before immediately transmitting that same command to the DAW (Reaper), connecting a bridge between the two.

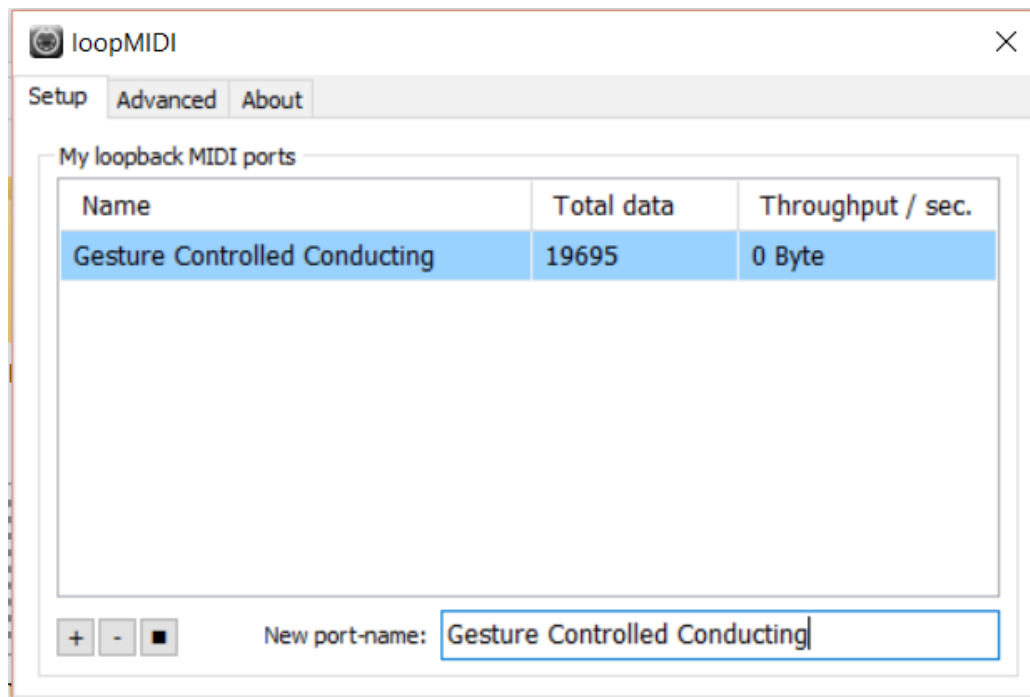


Figure 34 - loopMIDI Port Configuration

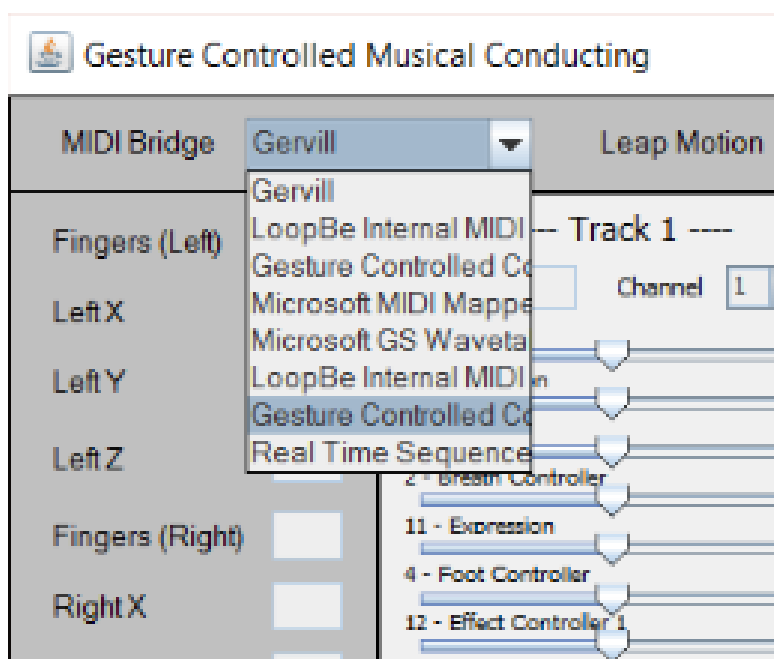


Figure 35 - Solution Port Selection

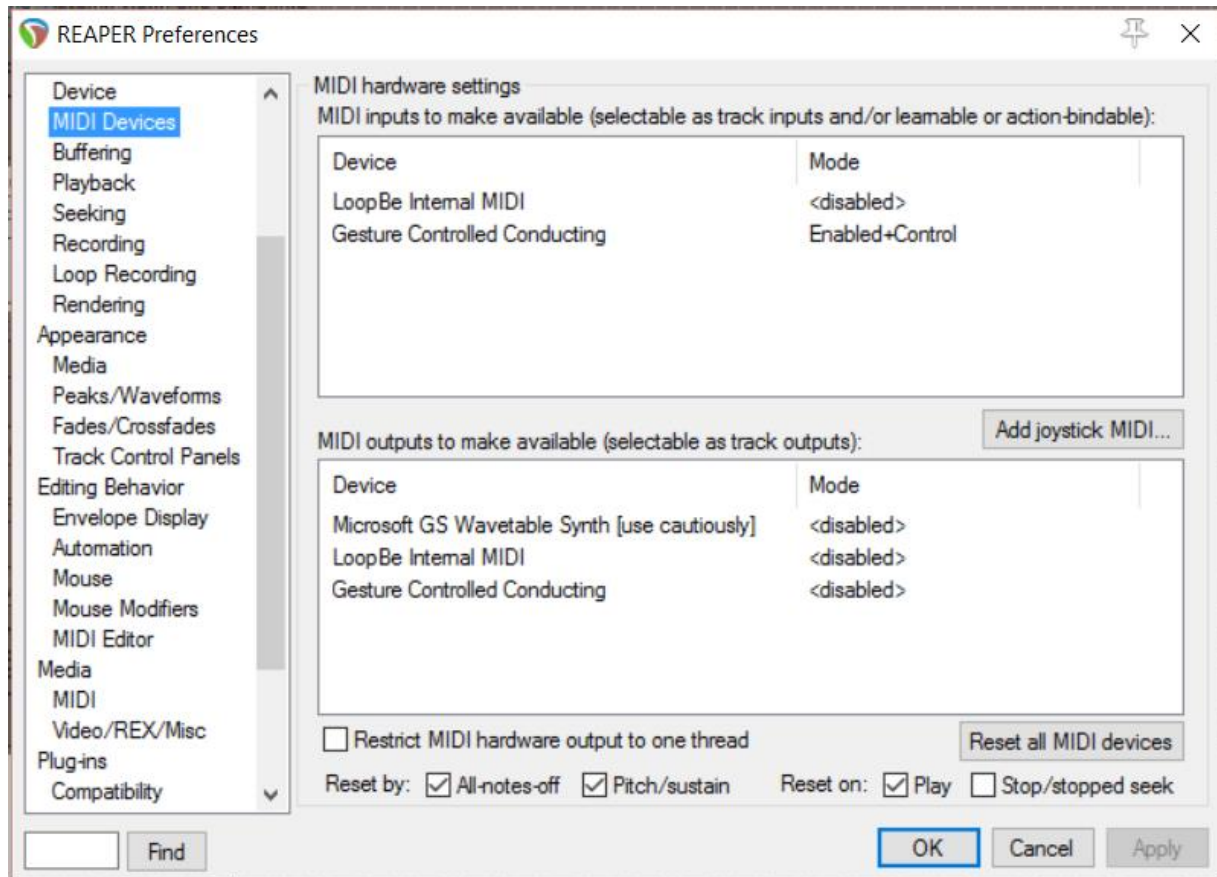


Figure 36 – Reaper Preferences

Figure 34 shows the virtual port being set up in loopMIDI, the port name is simply typed in the text box and the + button is clicked to create the port. The application must run in the background to keep the port open but it can easily be minimised.

Figure 35 shows how to select the port from the application to connect it to loopMIDI. The MIDI Bridge combo box becomes populated with available MIDI devices once the application is opened and the user needs to select the correct port to link up to the DAW.

The last step to connect the three components together is to set up the DAW so it is ready to receive MIDI commands. In the preferences section of Reaper (Figure 36) the MIDI device must be set to the option 'Enabled + Control' for it to be set up so that CC commands can be received.

4.4.4 MIDI

The second task involved developing the MIDI section of the application; this was split into two sub-tasks: set the application up able to send test MIDI CC commands and connect the application to loopMIDI. Setting up the Java MIDI API involved importing the library into the project through the following code:

```
import javax.sound.midi.*;
```

As loopMIDI had already been set up (4.4.2), the system was ready to begin searching for the port. The system is able to detect available devices through the following line:

```
MidiSystem.getMidiDeviceInfo();
```

As there may be more than one available device, the user must select the correct one from a dropdown located on the User Interface. This also allows them to set up multiple ports if they wish to do so. The feature that enables the user to select the MIDI device and assign it to an object was also implemented as follows:

```
public void setMidiOutDevice(Object aDevice) throws
MidiUnavailableException {
    MidiDevice midiOutDevice = MidiSystem.getMidiDevice(
(MidiDevice.Info) aDevice);
    midiOutDevice.open();
    System.out.println("Outputting to " +
aDevice.toString());
}
```

Once a virtual port has been selected by the user (Figure 35), it is possible to send MIDI CC commands to the device to test if any MIDI CC data was being correctly transmitted. To send a CC command, the following method was implemented:

```
public void sendMessage(MIDIAddress aAddress, int aValue) {

    try {
        int timeStamp = -1;
        ShortMessage sm = new
ShortMessage(ShortMessage.CONTROL_CHANGE,
aAddress.getChannel(), aAddress.getCCCommand(), aValue);
        midiOutDevice.getReceiver().send(sm,
timeStamp);

    } catch (InvalidMidiDataException |
MidiUnavailableException e) {
        System.err.println("MIDI interface error:
Invalid MIDI data: " + e.getLocalizedMessage());
    }
}
```

The sendMessage method refers to the MIDIDevice object, this is the device that will be sent the command. For the sake of grouping data, a MIDIAddress class was created to store the data associated with the MIDI CC command such as the channel, CC value and track. To test the MIDI commands being sent to loopMIDI, the port was selected in the application and a test command was sent using the method previously mentioned. According to the Figure below, the total data sent through had changed from 0 to 2187, which means that data had been successfully sent to loopMIDI (Figure 37).

Gesture Controlled Musical Conducting

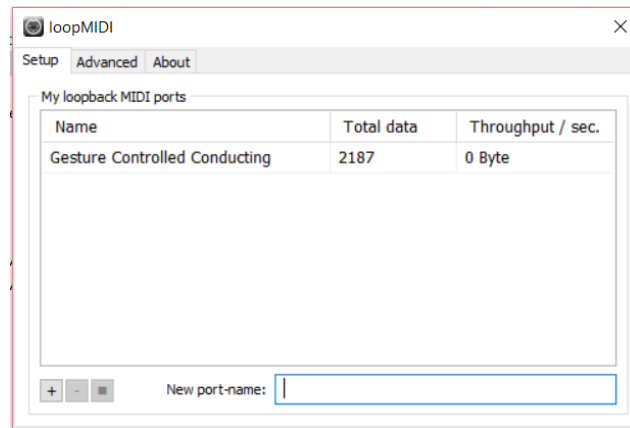


Figure 37 - loopMIDI Test Results

4.4.5 User Interface

The User Interface was implemented based on the designs (3.5.4). It acts as a visual layer that connects the various components of the system and will constantly provide feedback to the user across all of the panels.

Once the Leap Motion and MIDI modules worked successfully the User Interface was implemented, shown in Figure 38.

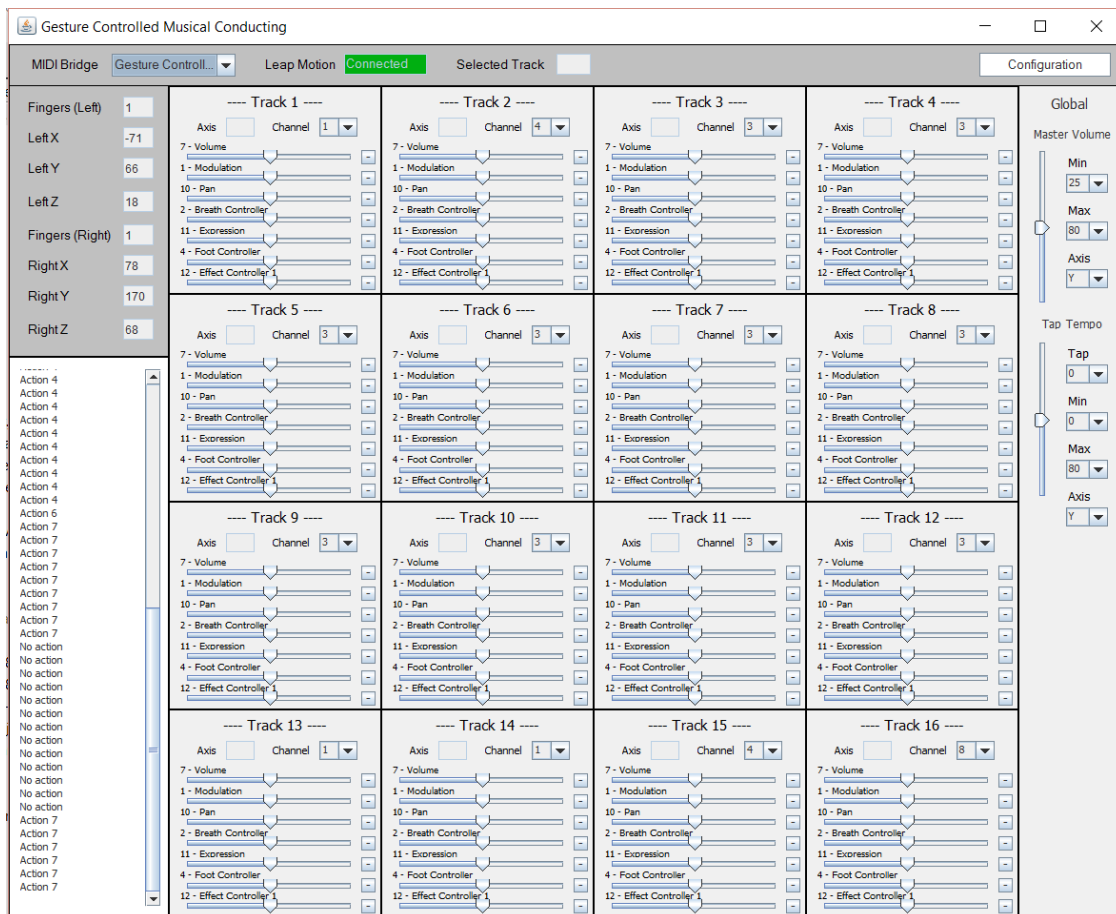


Figure 38 – The Main User Interface

Gesture Controlled Musical Conducting

Once the User Interface shell was implemented, the modules that have been created were connected up to display the relevant information to the user. The Leap Motion module has been connected up to the UI to provide:

- Real-time co-ordinate data added to the top-left panel to help the user understand what is being detected by the system, this will also help provide feedback when the controls are implemented.
- The configuration panel shows the status of the Leap Motion Sensor which turns green when the sensor is connected and red when it is disconnected.

The MIDI module was also connected to the User Interface, this included showing the list of available MIDI ports from loopMIDI on the configuration panel. At this stage most of the MIDI module could not be connected up as it required the control system to be implemented to determine the value of the CC commands being sent. This will also affect the sliders associated with each action as the sliders will change dynamically based on the user input to provide the real-time feedback.

4.4.6 Control System

The control system was translated into code from the designs (created in 3.6). Once it was implemented, the track panels represent the grid (3.6.2) and when a user selects a section of the grid it highlights the mapped track, as shown below:

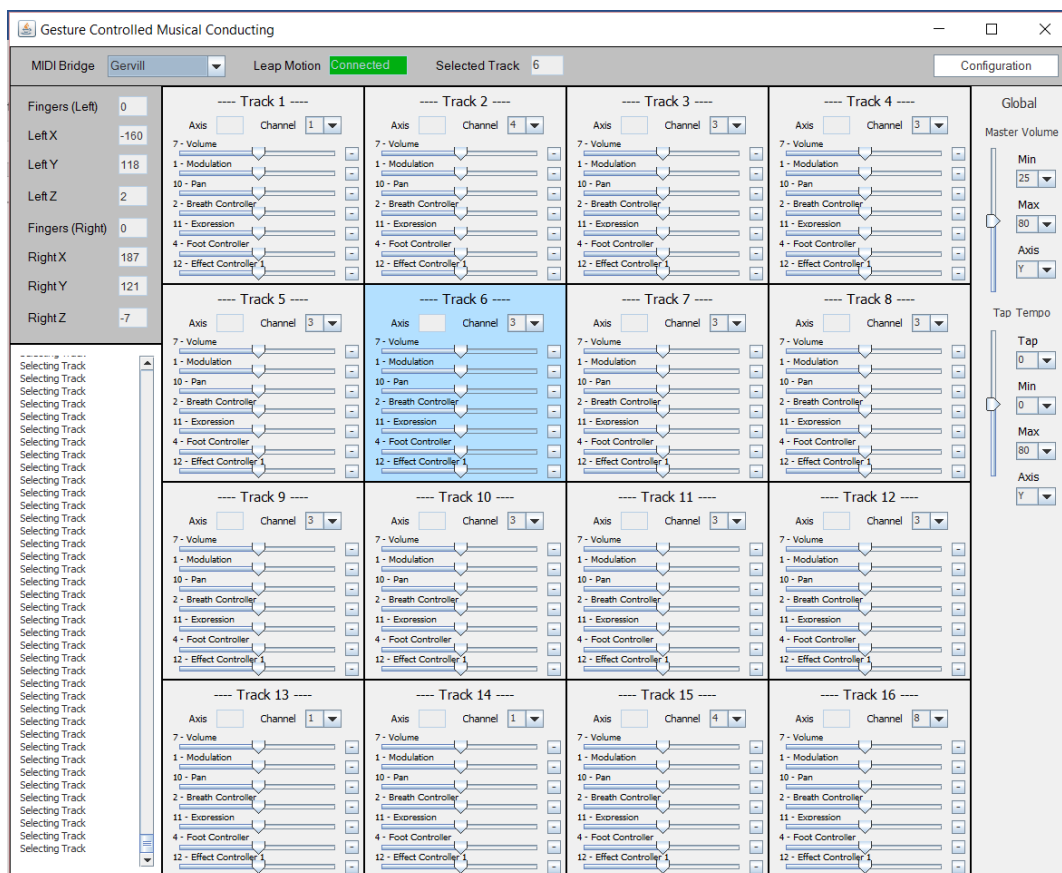


Figure 39 - Implemented Track Selection

The implementation for the grid is shown below. The method `selectTrackFromGrid` is part of the `InterfaceHandler` class which takes the co-ordinates of the finger and selects the track based on its position within the grid. The global action panel is selected by moving the finger to the right of the grid.

```
// Grid dimensions
int appWidth = 800;
int appHeight = 800;

// Setting up the grid and pointer
InteractionBox iBox = aController.frame().interactionBox();
Pointable pointable =
aController.frame().pointables().frontmost();

Vector leapPoint = pointable.stabilizedTipPosition();
Vector normalizedPoint = iBox.normalizePoint(leapPoint, true);

// Position of the finger on the grid
float appX = normalizedPoint.getX() * appWidth;
float appY = (1 - normalizedPoint.getY()) * appHeight;

if (index != null && fingersLeftExtended == 1){
    GUI.Log("Selecting Track");
    GUI.selectTrackFromGrid(appX, appY);
    selectedAction = 1;
}
```

The following code represents a portion of the code that checks for the gestures. The following code contains the lines where the first action is detected:

```
if (thumb == null && index != null && middle != null && ring
== null && pinky == null && fingersRightExtended == 2) {
    if (MainGUI.getSelectedTrack() != 17){
        GUI.Log("Action 1");
    }
    selectedAction = 2;
    data =
Strings.getConfigSettings(MainGUI.getSelectedTrack(), 0);
}
```

The rest of the code responsible for checking the gesture being performed is very similar to the code listed above. Each gesture is based on a combination of fingers being extended to select a particular action slot.

Setting the CC Value

Once the action has been selected, the left hand is used for selecting the value for the CC command. Depending on how it's configured, the axis is used to select the value for the action that is between 0 and 100. The following table shows a breakdown of which axis requires which movement:

Axis	Direction
X	Up and down
Y	Left and right
Z	Forwards and backwards

Table 2 – Axis directional Table

Selecting an Action and Control Change Value

The left hand has been implemented to select both the track and the CC value. When the index finger is extended, the left hand is in selection mode. Any other gesture with the left hand will control the CC value if an action is selected with the right hand.

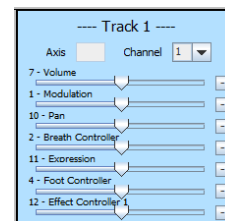
Though the controls may seem complex, when a user practices controlling the system it becomes natural which is similar to learning an instrument for the first time – a user cannot expect to be an expert with zero practice!

Controls Breakdown

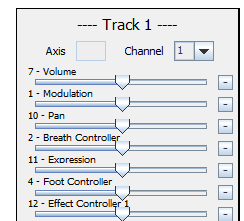
To simplify the controls, a set of instructions were created to help first-time users of the system. The following steps show the process of selecting a track, action and controlling the CC value:

1. Select the track

Using the left hand with the index finger extended, imagine there is a grid the same size as your screen in front of you. Carefully move your finger around until the track you wish to select has turned blue – this will mean it has been selected.



Track selected



Track deselected

2. Select the action

Using the table below, extend your fingers associated with the required action slot:

Action Slot	Thumb	Index	Middle	Ring	Pinky
1		Extended	Extended		
2		Extended	Extended	Extended	
3		Extended	Extended	Extended	Extended
4	Extended	Extended	Extended	Extended	Extended
5		Extended			Extended
6	Extended	Extended			
7	Extended				

3. Set the action value

With a closed fist on your left hand (keep the right hand on the action) – move your hand in the direction according to the set axis to change the value of the CC command being sent through the MIDI receiver, the action slider should change accordingly.

Axis	Hand direction
X	Up and down
Y	Left and right
Z	Forwards and backwards

Table 2 – Axis directional table

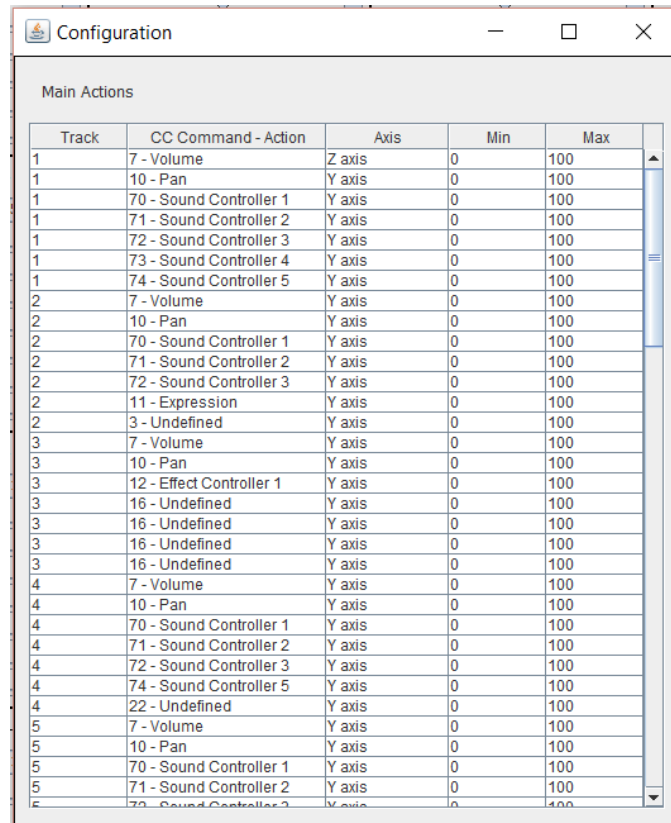
Interface Integration

Once the controls were implemented, an important feature was to provide feedback to the user so that they are aware of the status of the application at all times. Feedback was provided through several features:

- A text box on the configuration panel to show the current selected track.
- Each track panel represents a section on the virtual grid that also highlights each panel blue when the user selects.
- Each action slot has a slider that represents the value being transmitted from the application.

4.4.7 User Configuration and Saved Settings

Once the main modules have been connected, the last feature was to add the ability to set user preferences. To enable the user to tailor the application to their own preferences, each action slot was implemented to be part of a configuration table that is accessed via the configuration button on the main panel:



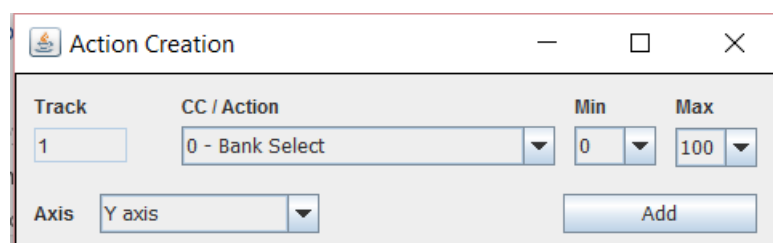
The screenshot shows a window titled "Configuration" with a table of "Main Actions". The table has five columns: Track, CC Command - Action, Axis, Min, and Max. The data is as follows:

Track	CC Command - Action	Axis	Min	Max
1	7 - Volume	Z axis	0	100
1	10 - Pan	Y axis	0	100
1	70 - Sound Controller 1	Y axis	0	100
1	71 - Sound Controller 2	Y axis	0	100
1	72 - Sound Controller 3	Y axis	0	100
1	73 - Sound Controller 4	Y axis	0	100
1	74 - Sound Controller 5	Y axis	0	100
2	7 - Volume	Y axis	0	100
2	10 - Pan	Y axis	0	100
2	70 - Sound Controller 1	Y axis	0	100
2	71 - Sound Controller 2	Y axis	0	100
2	72 - Sound Controller 3	Y axis	0	100
2	11 - Expression	Y axis	0	100
2	3 - Undefined	Y axis	0	100
3	7 - Volume	Y axis	0	100
3	10 - Pan	Y axis	0	100
3	12 - Effect Controller 1	Y axis	0	100
3	16 - Undefined	Y axis	0	100
3	16 - Undefined	Y axis	0	100
3	16 - Undefined	Y axis	0	100
3	16 - Undefined	Y axis	0	100
4	7 - Volume	Y axis	0	100
4	10 - Pan	Y axis	0	100
4	70 - Sound Controller 1	Y axis	0	100
4	71 - Sound Controller 2	Y axis	0	100
4	72 - Sound Controller 3	Y axis	0	100
4	74 - Sound Controller 5	Y axis	0	100
4	22 - Undefined	Y axis	0	100
5	7 - Volume	Y axis	0	100
5	10 - Pan	Y axis	0	100
5	70 - Sound Controller 1	Y axis	0	100
5	71 - Sound Controller 2	Y axis	0	100
5	72 - Sound Controller 3	Y axis	0	100

Figure 40 – Configuration Table

The configuration table enables the user to change all of the application settings in one place. Each of the seven action slots is completely configurable, so each track settings can be adapted based on the type of instrument it is meant to be controlling, for example the breath CC command may be relevant for a flute but will not be needed for a piano or guitar.

The user can also set the CC command, minimum and maximum values or change the axis. As an alternative to the configuration table, each action slot has a button next to it that enables the user to set the settings specific to the slot. Once the button is clicked, a window pops up to allow the changes:



The screenshot shows a window titled "Action Creation" with the following fields and controls:

- Track:** A text input field containing the number "1".
- CC / Action:** A dropdown menu showing "0 - Bank Select".
- Min:** A numeric input field containing "0".
- Max:** A numeric input field containing "100".
- Axis:** A dropdown menu showing "Y axis".
- Add:** A button to confirm the configuration.

Figure 41 – Action Slot Configuration

The final improvement to the user experience included the ability to save the user preferences so they can be loaded the next time the user opens the application; this feature saves the user from having to configure the application each time it is opened.

Two methods were used to implement the user preferences. The Java preferences library was used to implement the feature to save combo box selections within the application, this included the channel selection for each track. If the application is closed and re-opened the selections return to the same state in which they were saved. Saving and retrieving data from Java Preferences is very simple:

Save Data

```
Preferences prefs =  
Preferences.userNodeForPackage(Settings.class);  
prefs.putInt("Track1Channel", aChannel);
```

Retrieve Data

```
return prefs.getInt("Track1Channel", 0);
```

As the Java preferences library is only intended to store small amounts of data, it was not suitable to store the configuration table. The alternative that was used was to store the complete table as an array object within a settings file. This is saved each time an action slot is configured and loaded automatically each time the application is loaded. Saving the table to a file was a lot more complicated than using Java Preferences in terms of the amount of code:

Save Table

```
public static void saveTable(List<Command> aCommands) {  
    ObjectOutputStream objectOutputStream = null;  
    FileOutputStream fileOutputStream;  
  
    try{  
        fileOutputStream = new FileOutputStream(dir, false);  
        objectOutputStream = new  
ObjectOutputStream(fileOutputStream);  
        objectOutputStream.writeObject(aCommands);  
        objectOutputStream.close();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    System.out.println("Table saved");  
}
```

Load Table

```
public static List<Command> loadTable() throws IOException {  
    try {  
        FileInputStream fileInputStream = new  
FileInputStream(dir);  
        ObjectInputStream objectInputStream = new  
ObjectInputStream(fileInputStream);  
        mainTableData.clear();  
        mainTableData = (ArrayList<Command>)  
        objectInputStream.readObject();  
        objectInputStream.close();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    System.out.println("Table data loaded");  
    return mainTableData;  
}
```

4.4.8 CC Mapping within Reaper

Once the MIDI port has been enabled within Reaper (4.4.2), individual CC commands had to be mapped to effects within Reaper. Once a MIDI sequence has been loaded into a track slot in Reaper, it needs to be assigned 'FX' (virtual instruments) to be able to produce sound. This is because as mentioned before, MIDI is not sound and needs an instrument to make it produce music.

Each virtual instrument has its own effects that can be controlled and set differently depending on how the artist wants the instrument to sound. The list of configurable parameters for the virtual piano within Reaper include: decay, octave, patch, presence, soundfont oscillator, sustain and tune.

To map each 'slot' within the solution to each parameter in the DAW, it needs to be mapped to the right CC command. Each slot needs to be mapped through the following steps:

1. Click on the parameter to be mapped and click the 'Param' button (Figure 42).

Gesture Controlled Musical Conducting

2. Send the CC command and Reaper will detect the CC and Channel values being transmitted (Figure 43), once the command appears in the text box shown, it means the DAW has detected it successfully.

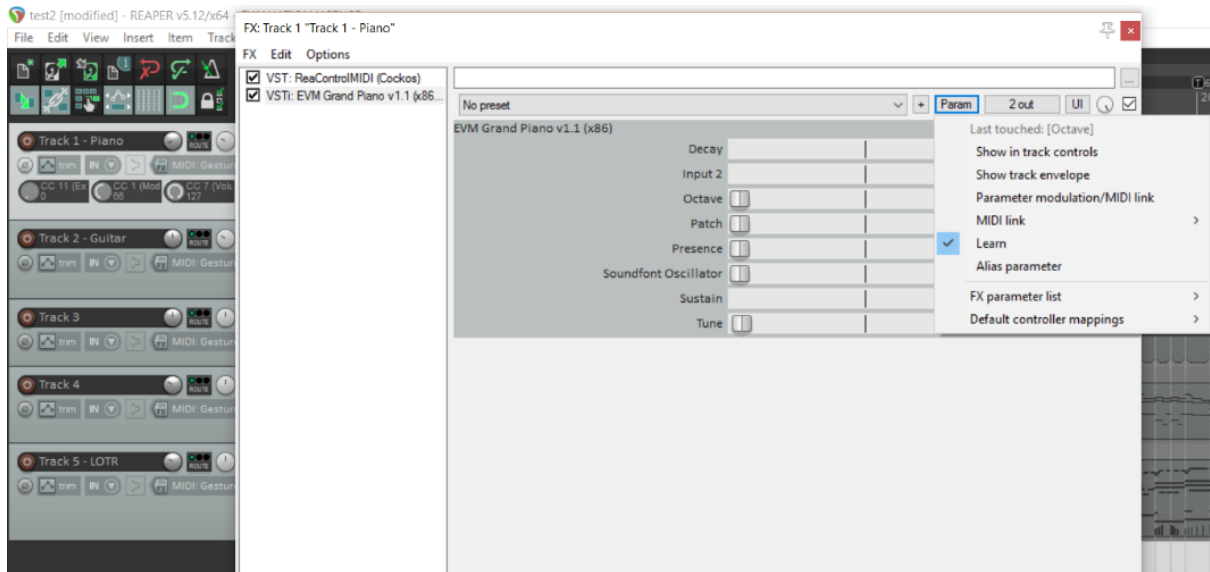


Figure 42 – Reaper 'Learn' Feature

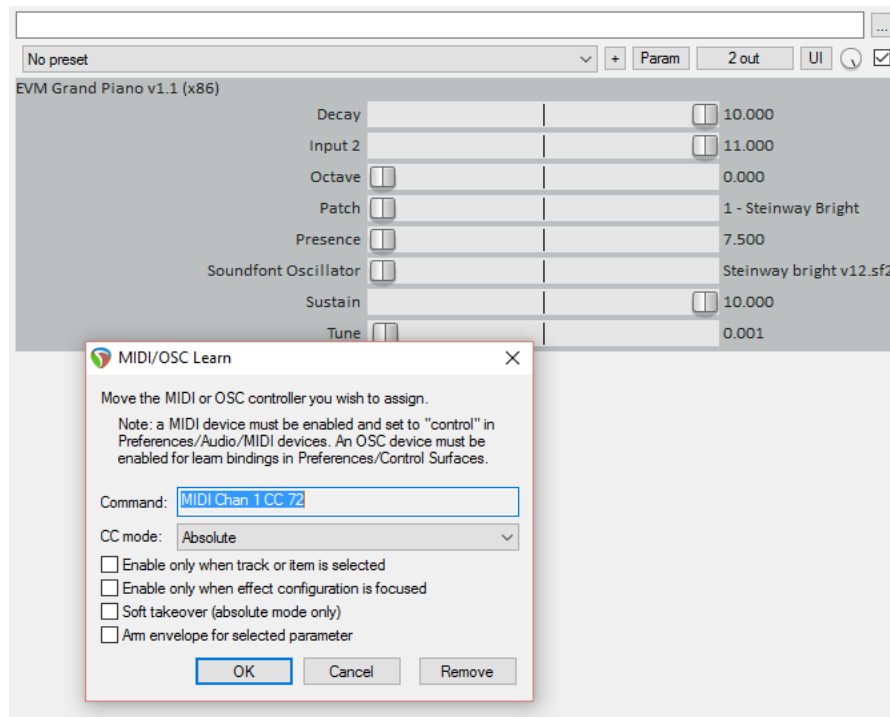


Figure 43 – Reaper CC Mapping Feature

5 Results and Evaluation

5.1 Testing

A common way testing has been conducted on previous projects was to create multiple test cases for each section of functionality and formally document the results with attached evidence. As testing usually takes up large amount of project time, I preferred the approach to perform tests on each module as it was created. As mentioned in a previous section, the development was split into several modules that covered a particular technology or concept, so once each module was completed informal functional tests were performed to determine if the component was working as intended. If a test failed immediate corrections were made to fix the issue. Once I was confident each module was robust they were all integrated into the larger solution.

The formal testing conducted on the project was in the form user acceptance testing [49], this is due to the fact that the user experience is the main factor that is critical in determining the overall quality of the project. The hardware, User Interface and control system needed to be implemented perfectly to provide a complete overall experience.

To finally determine if the project has achieved its intended goals, a requirements evaluation was performed to discuss in detail whether or not each requirement has been met along with any supporting evidence.

5.2 The Demo

5.2.1 Overview

The demo was created to show the users the capabilities of the system during the user acceptance testing. Another purpose was also so that an evaluation could be conducted on the current state of the application (section 5.5) to see if it has met the initial requirements.

The demo includes five different instruments and maps a range of commands that manipulate each instrument in a unique way. The demo was created in the following sequence:

1. Make sure loopMIDI is connected to Reaper and the MIDI port is enabled (shown in 4.4.2 and 4.4.3).
2. Load five MIDI sequences into five track slots within Reaper (Figure 44).
3. Assign a virtual instrument to each track (Figure 45).
4. Map the effects of each instrument to CC commands within the solution (4.4.8).

Gesture Controlled Musical Conducting

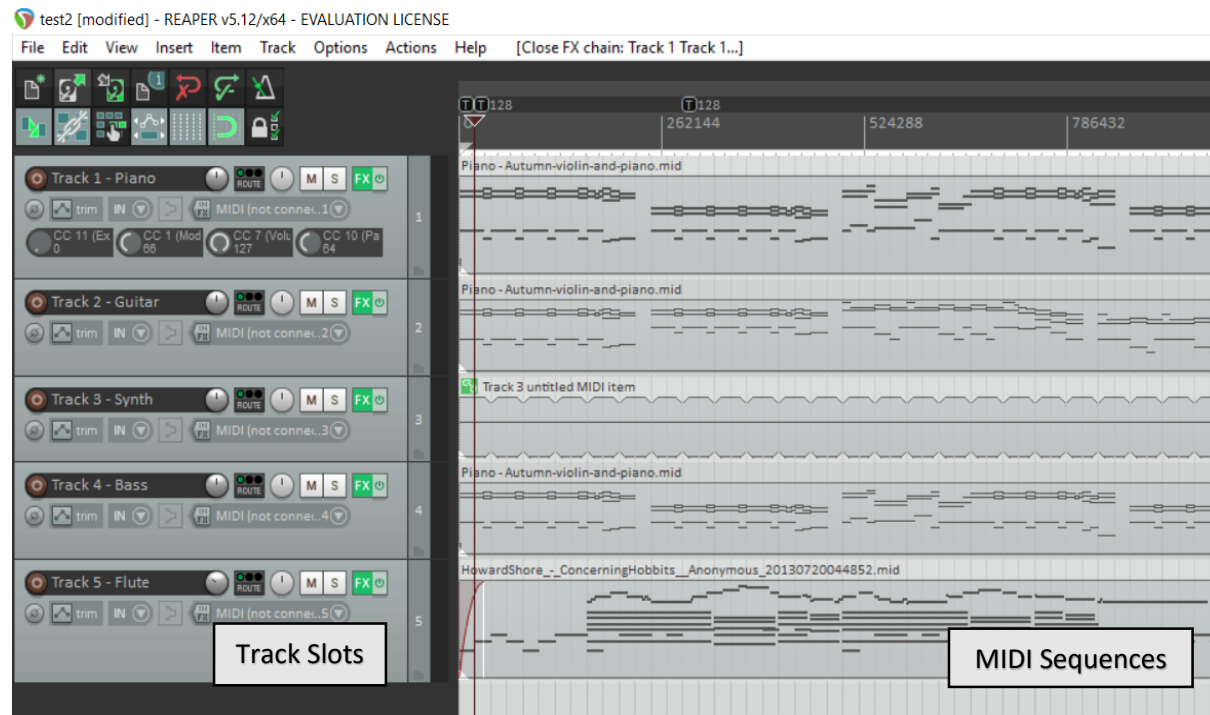


Figure 44 – MIDI Sequences and Track Slots

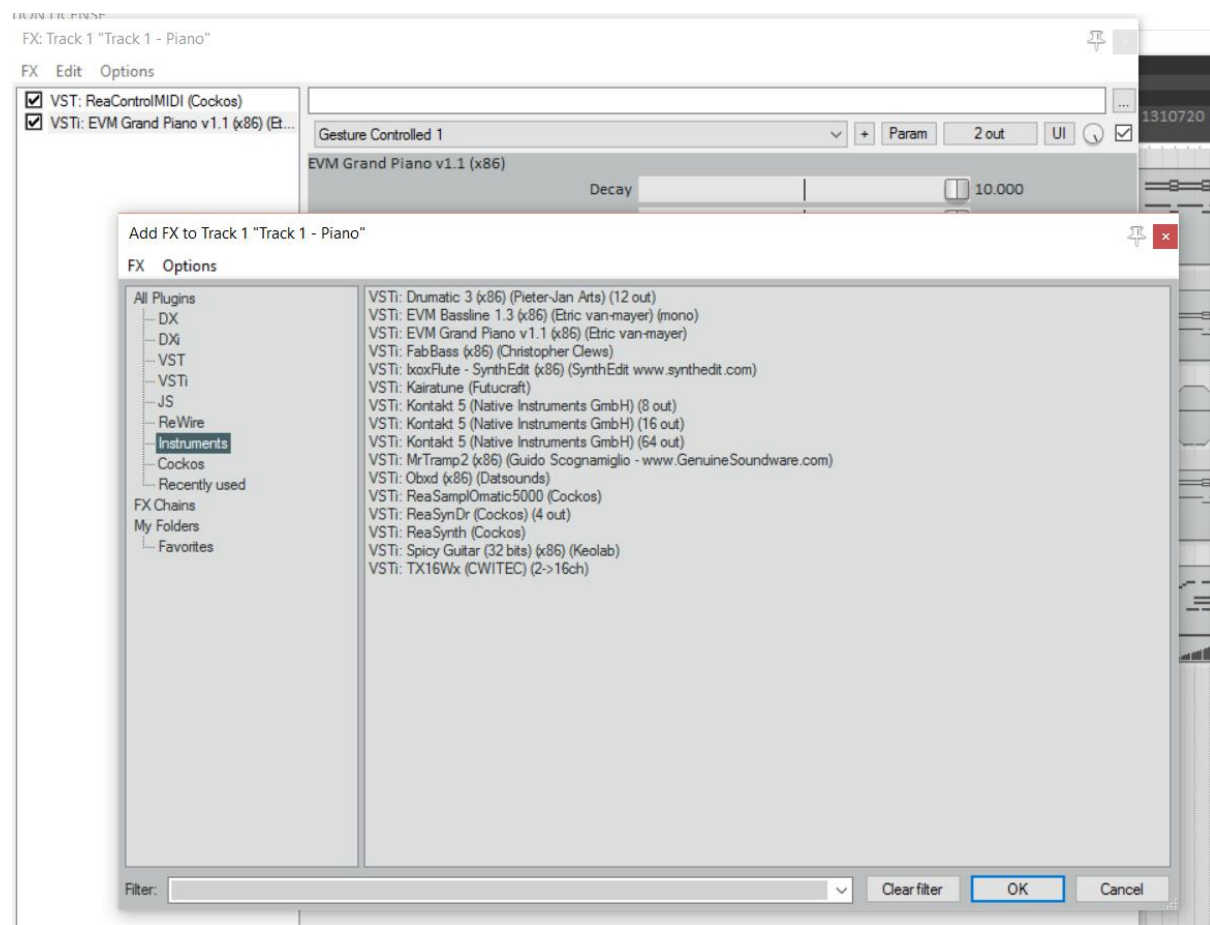


Figure 45 – Loading Virtual Instruments into Tracks

5.2.2 Effects

The list of effects is dependent can be varied depending on the instrument. Figures 46 and 47 below show the difference with possible effects that can be mapped to a Piano and a Flute. The Flute has a lot more effects that can be mapped to change the instrument, the top seven effects (based on user preference) will be mapped to the solution through different CC commands.

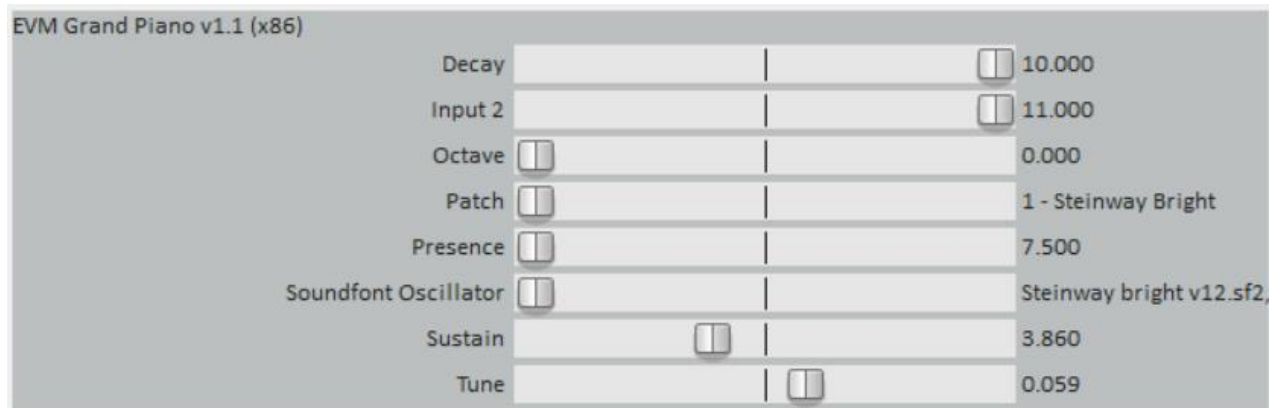


Figure 46 – The Grand Piano Commands

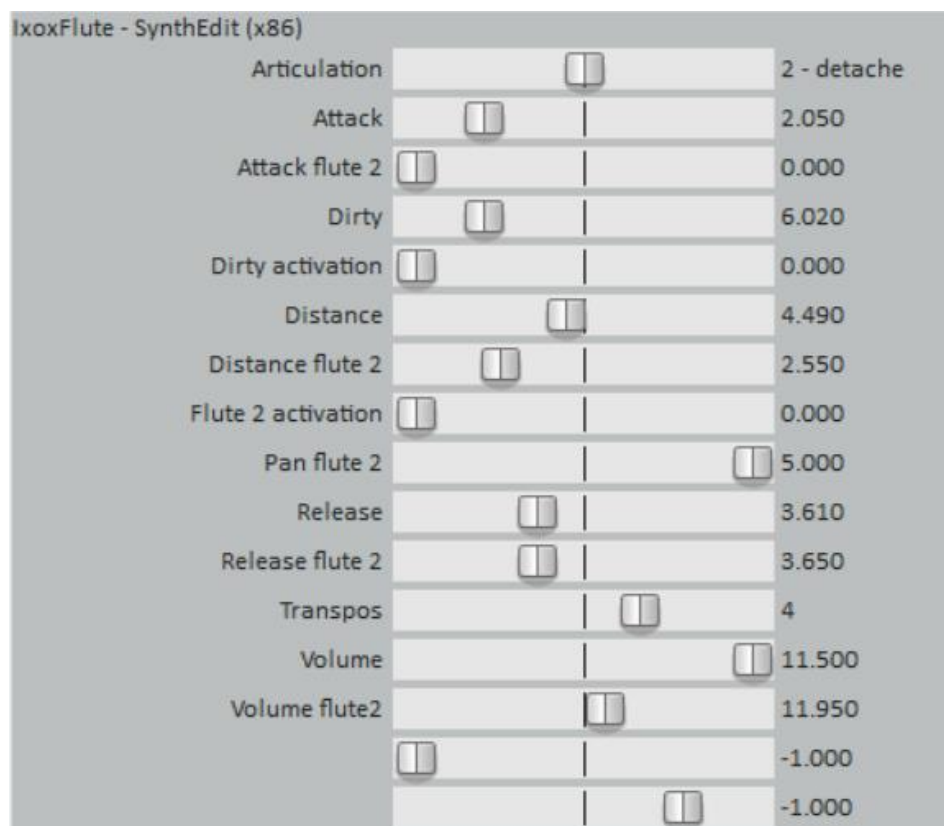



Figure 47 – The Flute Commands

Table 5 shows the commands that have been assigned to each slot and which effect they have been mapped to within the Digital Audio Workstation (see 4.8 for the command mapping process). The table covers a large range of different effects that include modulation, tune and pitch. It was important to cover several instruments to show the different effects for a range of instruments. The five instruments covered in the demo include: Piano, Bass, Guitar, Oscillator and Flute.

In the table below, the action slots represent the commands that can be mapped to each CC command within the solution. Figure 48 shows track 1 that has 7 action slots that can be mapped using the  button to the side of each slot.

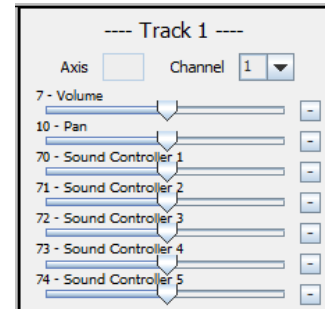


Figure 48 - Action Slots

Table 3 – Actions Mapped for the Demo

Action Slot	Channel and CC Value	Instrument	Mapped Command
Track 1 Slot 1	Channel 1 - 7	Piano	Volume
Track 1 Slot 2	Channel 1 - 10	Piano	Pan
Track 1 Slot 3	Channel 1 - CC 70	Piano	Octave
Track 1 Slot 4	Channel 1 - CC 71	Piano	Patch
Track 1 Slot 5	Channel 1 - CC 72	Piano	Presence
Track 1 Slot 6	Channel 1 - CC 73	Piano	Sustain
Track 1 Slot 7	Channel 1 - CC 74	Piano	Tune
Track 2 Slot 1	Channel 2 - 7	Guitar	Volume
Track 2 Slot 2	Channel 2 - 10	Guitar	Pan
Track 2 Slot 3	Channel 2 - CC 70	Guitar	Colour
Track 2 Slot 4	Channel 2 - CC 71	Guitar	Pitch Bend
Track 2 Slot 5	Channel 2 - CC 72	Guitar	Strumming Speed
Track 2 Slot 6	Channel 2 - CC 73	Guitar	Vibrato Freq.
Track 2 Slot 7	Channel 2 - CC 74	Guitar	Harmonics Toggle
Track 3 Slot 1	Channel 3 - 7	Synth	Volume
Track 3 Slot 2	Channel 3 - 10	Synth	Pan
Track 3 Slot 3	Channel 3 - CC 70	Synth	Detune
Track 3 Slot 4	Channel 3 - CC 71	Synth	Attack
Track 3 Slot 5	Channel 3 - CC 72	Synth	Triangle
Track 3 Slot 6	Channel 3 - CC 73	Synth	Sine2Tun
Track 4 Slot 1	Channel 4 - 7	Bass	Volume
Track 4 Slot 2	Channel 4 - 10	Bass	Pan
Track 4 Slot 3	Channel 4 - CC 70	Bass	Sustain
Track 4 Slot 4	Channel 4 - CC 71	Bass	Tune
Track 4 Slot 5	Channel 4 - CC 72	Bass	Decay
Track 4 Slot 6	Channel 4 - CC 73	Bass	EQ
Track 5 Slot 1	Channel 5 - 7	Flute	Volume
Track 5 Slot 2	Channel 5 - 10	Flute	Pan
Track 5 Slot 3	Channel 5 - CC 70	Flute	Articulation
Track 5 Slot 4	Channel 5 - CC 71	Flute	Distance

5.3 User Acceptance Testing

One of the major components of the system is the user's ability to successfully interact with the system using their hands as input. The audience selected for the testing included people personally known, as it was important to obtain honest feedback, however critical. I like a quote from an individual called Elon Musk – "Seek negative feedback, especially from friends" [50]. This is interesting because it sums up what I was trying to achieve - obtain critically honest feedback.

5.3.1 Acceptance Testing Environment

The tests were conducted in suitable conditions that were the same for every user, these include:

- Quiet area suitable for two people (test subject and the test conductor).
- Room with two monitors, one displaying the digital audio software and one showing the solution.
- Suitable room environment with no distractions.

Before each experiment, the project environment was setup to make it ready for testing. Each user was shown the hardware before being asked to try and use the system without any guidance, this was to test how easy the software is to directly use without any external help. Users were then shown how the control system works through a demonstration before being asked to attempt to interact with the system once more.

The purpose of this test was to understand the effect of being shown how the system works from someone who is considered a system expert. Once the user felt comfortable using the system they were given the chance to experiment with one of the five demo instruments that were already preconfigured. Once the user was happy to end the experiment they were asked to fill out an online survey.

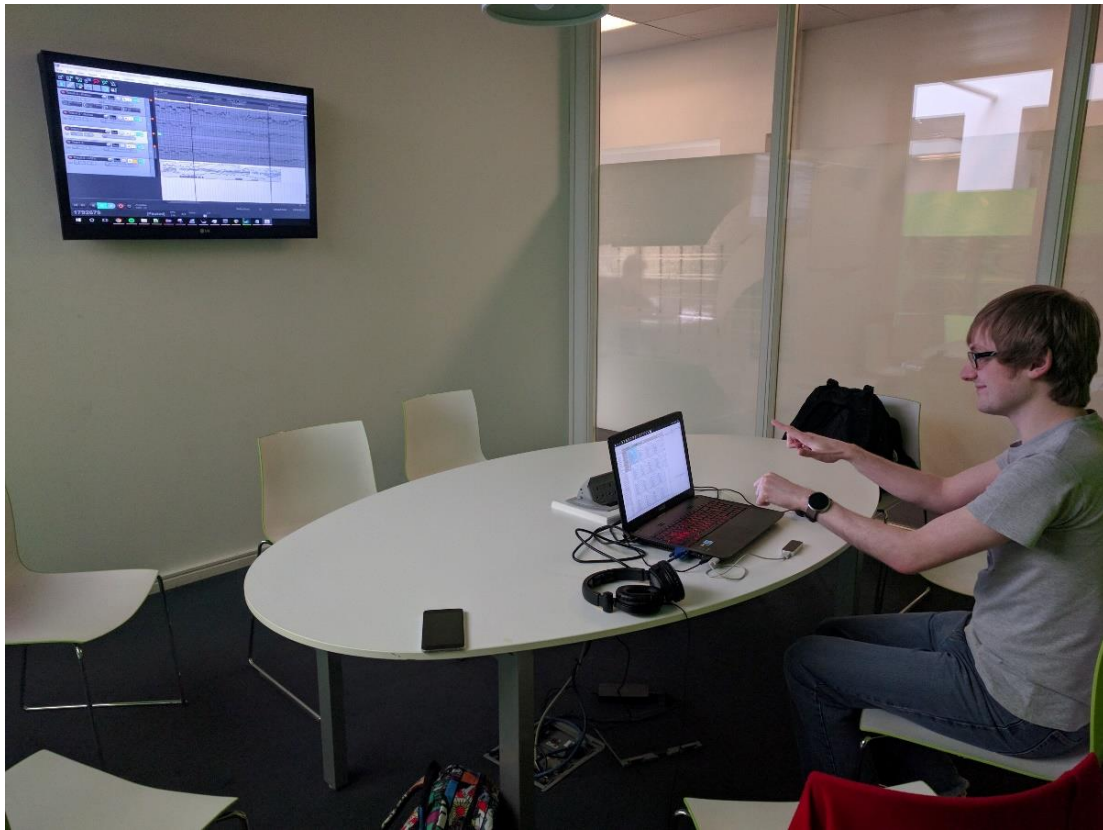


Figure 49 - A User Testing Setup

5.3.2 Survey Design

The overall purpose of the user acceptance testing was to gather an understanding of how easy users found interaction with the system. The aim of the testing was to understand the usability of the application from a range of users from those with a music background to users with no experience in music production. Once the users tested the application they were asked to complete a survey. The questions were designed to understand how enjoyable, usable and comfortable the users felt with aspects of the system such as the controls and User Interface design.

The opening question was designed to get a brief general understanding of what the user thought of the system. It was also a good place for them to describe what happened during the testing and if there were any problems.

1. In your own words, what was your experience of using the software?

The second question was important to get an understanding of how complete the users thought the software was. For the technical users, it was also good to get an understanding if they thought there was anything missing from their experience with digital music in the past.

2. *Is there any feature regarding MIDI control (volume, pitch, tone, expression etc.) that you feel is missing from the project?*

The third question was important to ask to determine if users thought there were any negative aspects of the experience that could be improved, for example better controls or improved interface etc.

3. *In your opinion what else could be improved about the project?*

The next two questions were asked to try and compare the usability of the solution. Asking users to rate the difficulty before and after the demonstration helps to determine if there needs to be more done to help users learn the systems controls.

4. *On a scale of 1 to 10, how easy did you find the project to use BEFORE you received a demonstration of how it worked? (1 being easy and 10 being very hard)*
5. *On a scale of 1 to 10, how easy did you find the project to use AFTER you received a demonstration of how it worked? (1 being easy and 10 being very hard)*

The last question was asked to see if there was anything that the user wanted to comment that had been left out of the above questions.

6. *Any further comments or opinions about the project?*

5.3.3 Results Analysis

In total, 7 people were tested; the group of people included 2 with some background in digital audio or music and 5 people that were not experienced with digital music.

Below is the analysis performed on the responses once they were all collected from test subjects:

1. *In your own words, what was your experience of using the software?*

Most users had an overall enjoyable experience with the software, for all of them it was the first time using a Leap Motion Sensor, although one user had heard of it before. Words like 'fun' and 'good' were used when discussing their first experiences using the Leap Motion. Most of the users said that the software was so much easier to use once showed a demonstration.

2. *Is there any feature regarding MIDI control (volume, pitch, tone, expression etc.) that you feel is missing from the project?*

Users overall felt there was a need for some minor tweaks, but overall felt there was not a lot missing. One user felt it would be useful to 'to filter which types of parameters can be added to particular MIDI tracks' when configuring each track from a particular instrument with set parameters.

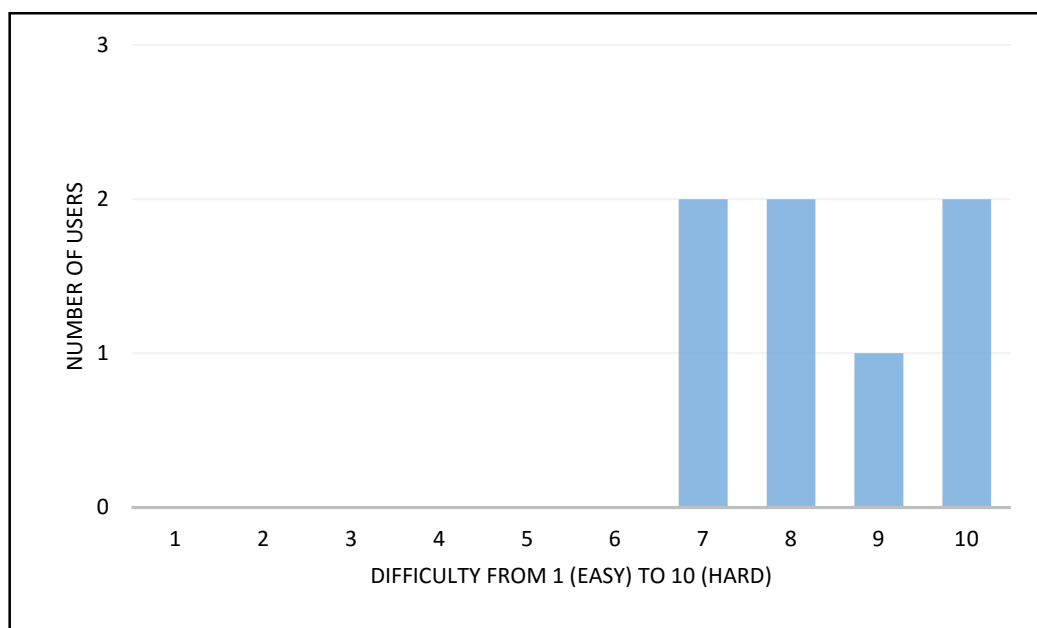
3. *In your opinion what else could be improved about the project?*

The overall opinion from users was that without any tutorials or documentation the solution was very hard to control. Most users said the user interface was well-designed, however there was a need for as one user said - 'clarity over what you're currently changing i.e. bolding a parameter such as volume when you've selected it and are changing it'. One user had the good idea of 'combining voice commands with gestures' to allow for more functionality. An interesting idea was the development of a scaled down version of the GUI that shows the currently selected track, for the user to view when focused on other software.

4. *On a scale of 1 to 10, how easy did you find the project to use BEFORE you received a demonstration of how it worked? (1 being easy and 10 being very hard)*

The next two questions were asked to see if a conclusion could be made regarding the difficulty of the software before and after being taught how to control it. The way the software was designed might cause it to be quite difficult for the average user to pick up and use, this experiment is to see if this is accurately the case.

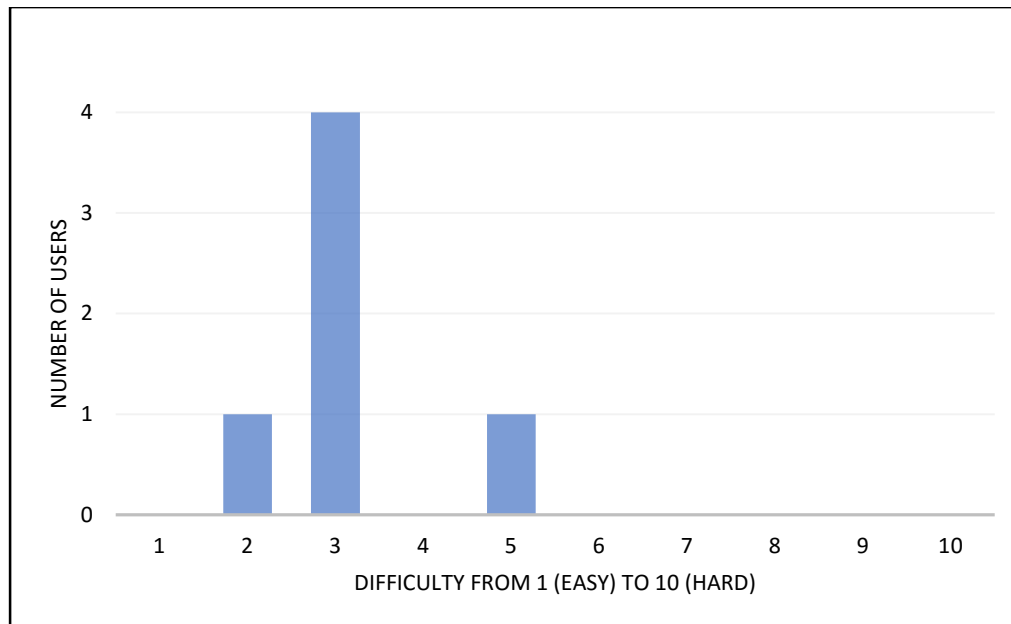
Most users as expected found the solution fairly difficult to use before being shown any help. The breakdown of the opinions can be shown below in Table 3:



Graph 1 – User Question Response Breakdown - Question 4

5. *On a scale of 1 to 10, how easy did you find the project to use AFTER you received a demonstration of how it worked? (1 being easy and 10 being very hard)*

The results from this question are almost the opposite of the previous question and shows that with guidance the project becomes a lot easier to use:



Graph 2 – User Question Response Breakdown - Question 5

6. Any further comments or opinions about the project?

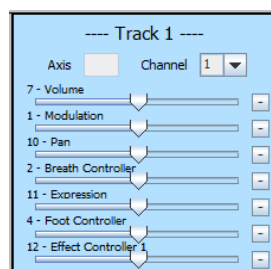
The users expressed their opinion about the difficulty being easier once they were showed the controls. Another user found that the Leap Motion would stop tracking their hands briefly, although admitted that is was probably down to inexperience with the controller.

A complete breakdown of the user responses can be found in Appendix B.

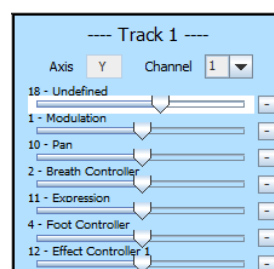
5.4 Further Development

As the development ran ahead of schedule for the implementation stage, there was time at the end to fix any minor bugs or errors that were experienced during the user acceptance tests.

One of the opinions expressed in the feedback was the fact that there is no feedback given for the action slot being selected. As a result, the code was re-visited and the action slots were changed to highlight white when selected to provide the feedback required.



Old selection (slot 1)



New selection (slot 1)

Another issue that became apparent from user testing was that changing some settings on the track would sometimes load in the wrong slot when the application was closed and re-opened. After analysing the code, the problem was identified as an error in the way the slot was saved based on the calculation of the slot. This error was fixed and the patch was uploaded to the code repository.

5.5 Requirements Evaluation

In this section I will describe to what extent the goals for the project were achieved. Evaluating the requirements that were set at the start of the project is a good quality measure to determine if the project was an overall success or failure.

Requirement: The system successfully connects to and reads input from the Leap Motion Controller.

Status: **Passed**

Without the input and connection from the Leap Motion Controller the application is fundamentally useless. The Leap Motion (described in 4.4.1) requires to be working properly in order for the user to control the application with gestures. Figure 50 below show the data presented to the user to show the Leap Motion is working correctly.

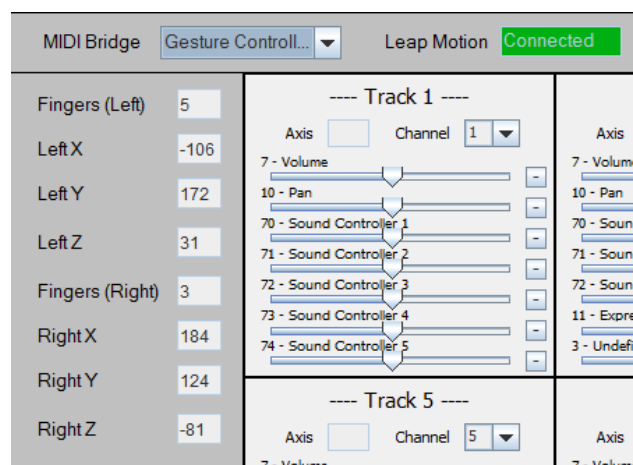


Figure 50 – How the Leap Motion Provides Feedback

Requirement: The system will be recognized as a virtual MIDI device all compatible digital audio software.

Status: **Partially Passed**

The parts of the system have correctly integrated with each other so that the CC commands can be sent and received from the intended components. The system has successfully sent CC data to loopMIDI that is shown on its interface. The implementation where this is discussed further can be read in section 4.4.2.

Gesture Controlled Musical Conducting

The evidence shows both ends of the link being connected to the port set up through the loopMIDI, however the requirement states ‘all’ compatible software. As MIDI is a standard, it would be expected that other software would be compatible, however there are so many digital audio workstations available they cannot all be tested. As the evidence relies on a small assumption, it cannot be fully signed off.

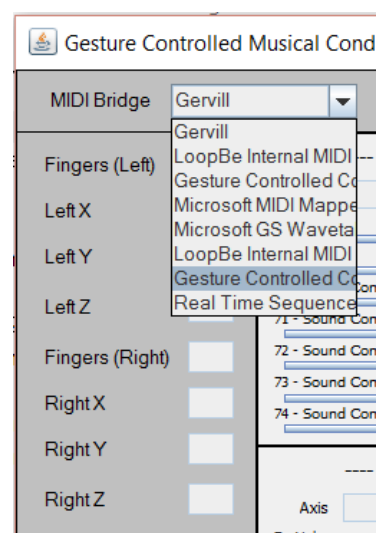


Figure 51 – MIDI Port Detected in the Solution

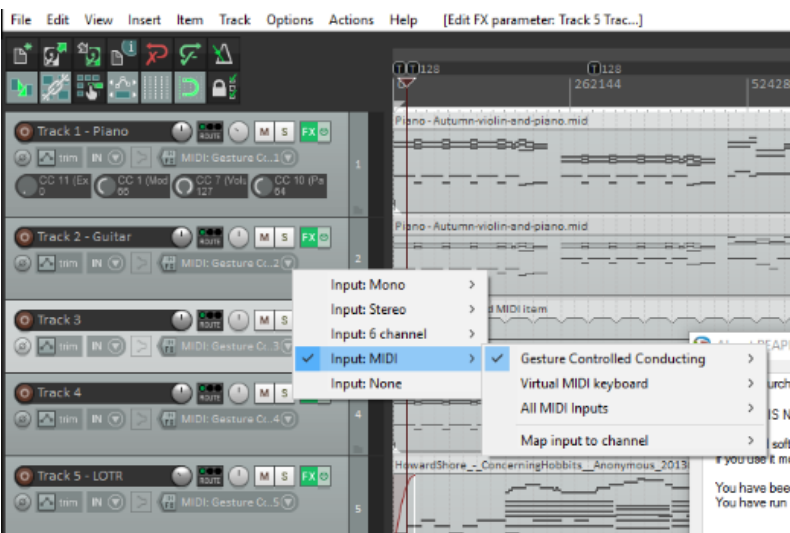


Figure 52 – MIDI Port Detected in Reaper

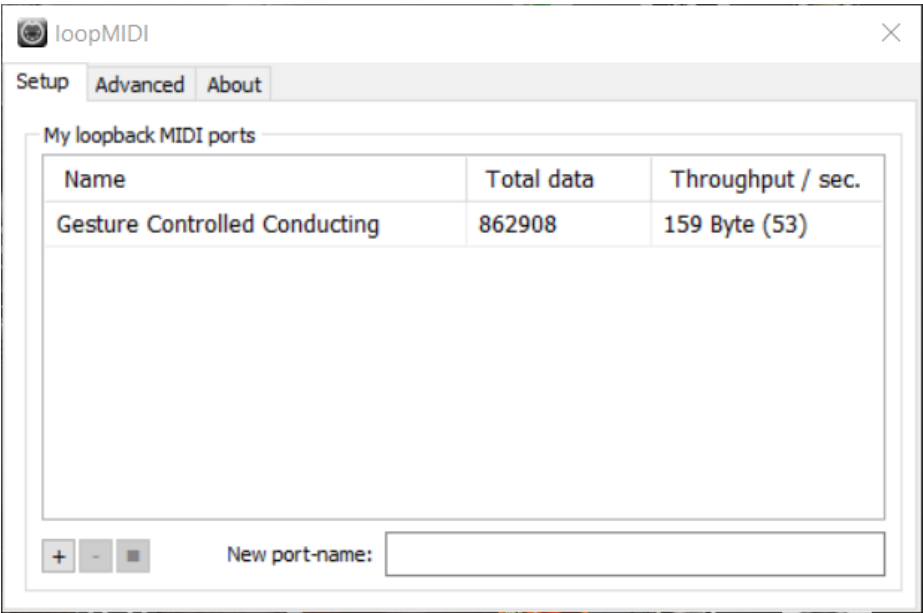


Figure 53 – MIDI Port Configured in loopMIDI

Requirement: The system sends MIDI CC commands (volume, modulation etc.) to digital audio software through the MIDI bridge tool based on the user selection and gesture.

Status: **Passed**

As the CC is sent to the digital audio workstation, depending on the effect the change is noticeable. As it is difficult to evidence audio on paper, the user testing has resulted in this requirement being passed as there were no problems hearing the effects being applied to the music.

When the learn CC command feature is used on the digital audio workstation, the user is required to perform the gesture they wish to map, the receiving software successfully detects which CC command is being sent through the intended channel, which is shown in the Figure below:

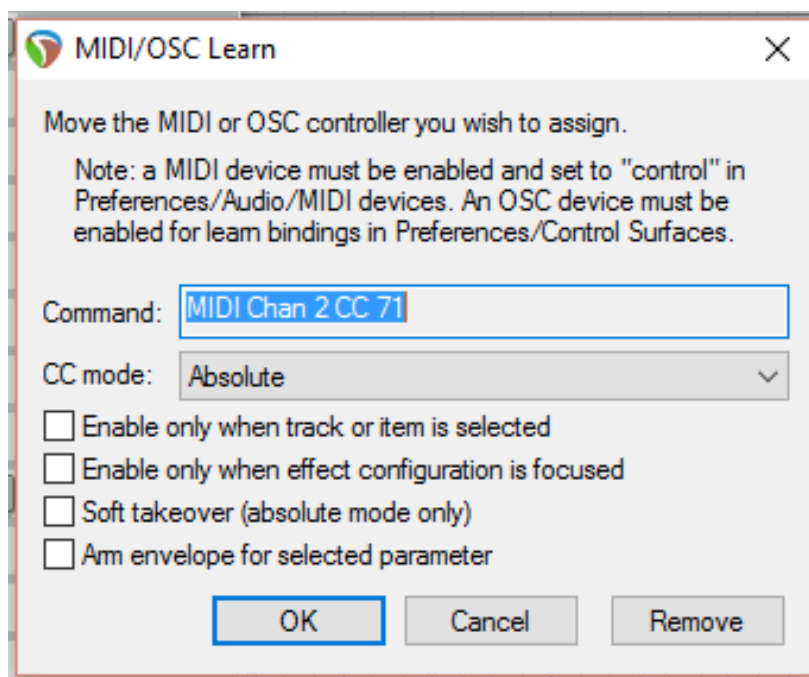


Figure 54 – CC Command Mapped in Reaper

Requirement: The system will save user preferences so that they are retrievable next time the user opens the application.

Status: Passed

The Java preferences library has been successful implemented into the project. When the user changes the default settings and close the application they save to the application preferences. The images below show the non-default configuration before the application is closed and after the application is opened, the last image clearly shows that user preferences have been saved. Figures 55 and 56 below show the preferences being saved for the channel and action slot preferences for each track:



Figure 55 – Track Preferences



Figure 56 – Track Preferences Once Application is Re-opened

Requirement: MIDI commands for the system are configurable.

Status: Passed

This requirement has been met. Each action within the system is configurable to any MIDI command through the configuration table or individual buttons on the panels. The purpose of the requirement was to make sure the user can tailor the application to their instruments they already have set up within the digital software. The figures below show the possible customisation options:

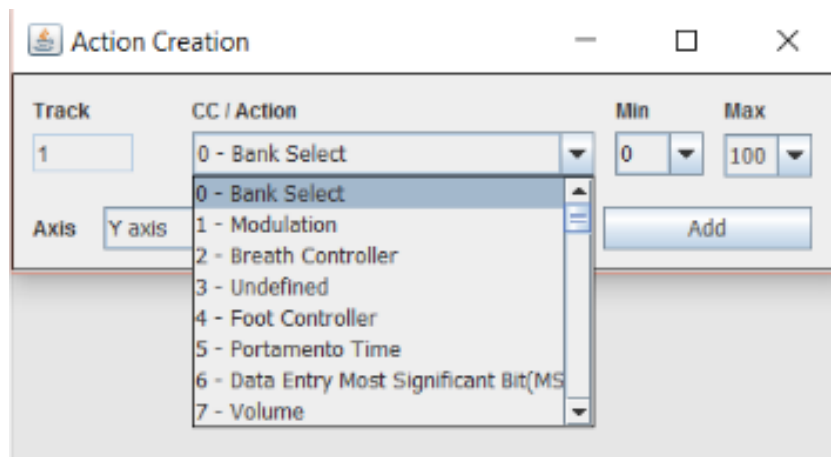


Figure 57 – Possible CC Commands

Main Actions				
Track	CC Command - Action	Axis	Min	Max
1	7 - Volume	Y axis	0	100
1	10 - Pan	Y axis	0	100
1	70 - Sound Controller 1	Y axis	0	100
1	71 - Sound Controller 2	Y axis	0	100
1	72 - Sound Controller 3	Y axis	0	100
1	73 - Sound Controller 4	Y axis	0	100
1	74 - Sound Controller 5	Y axis	0	100
2	7 - Volume	Y axis	0	100
2	10 - Pan	Y axis	0	100
2	70 - Sound Controller 1	Y axis	0	100
2	71 - Sound Controller 2	Y axis	0	100
2	72 - Sound Controller 3	Y axis	0	100
2	11 - Expression	Y axis	0	100
2	3 - Undefined	Y axis	0	100
3	7 - Volume	Y axis	0	100
3	10 - Pan	Y axis	0	100
3	12 - Effect Controller 1	Y axis	0	100
3	16 - Undefined	Y axis	0	100
3	16 - Undefined	Y axis	0	100
3	16 - Undefined	Y axis	0	100
3	16 - Undefined	Y axis	0	100
4	7 - Volume	Y axis	0	100
4	10 - Pan	Y axis	0	100
4	70 - Sound Controller 1	Y axis	0	100
4	71 - Sound Controller 2	Y axis	0	100
4	72 - Sound Controller 3	Y axis	0	100
4	74 - Sound Controller 5	Y axis	0	100
4	22 - Undefined	Y axis	0	100
5	7 - Volume	Y axis	0	100
5	10 - Pan	Y axis	0	100
5	70 - Sound Controller 1	Y axis	0	100
5	71 - Sound Controller 2	Y axis	0	100
5	72 - Sound Controller 3	Y axis	0	100

Figure 58 – Configuration Table

Requirement: The system has an advanced configuration for each MIDI command such as axis.

Status: **Passed**

This requirement is an optional extra to the previous one. The aim was to develop further configuration options (described in 4.4.5) if there was development time left in the project. The 'advanced' configurations implemented into the software include the ability to change the control axis and set a minimum and maximum value for the commands. The evidence below shows the possible axis' the user can select:

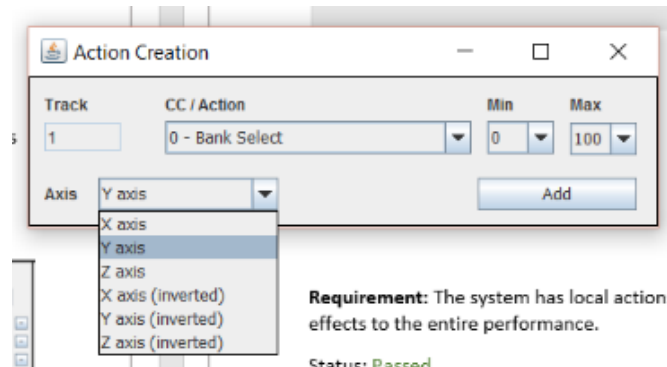


Figure 59 – Possible Axis Configurations

Requirement: The system has local actions for each track as well as global actions that apply effects to the entire performance.

Status: **Partially Passed**

The solution offers two global commands, these are master volume and tap tempo. As there are two only global commands implemented the project needs some work on this side to fully pass the requirement.

The global panel is shown in Figure 60.

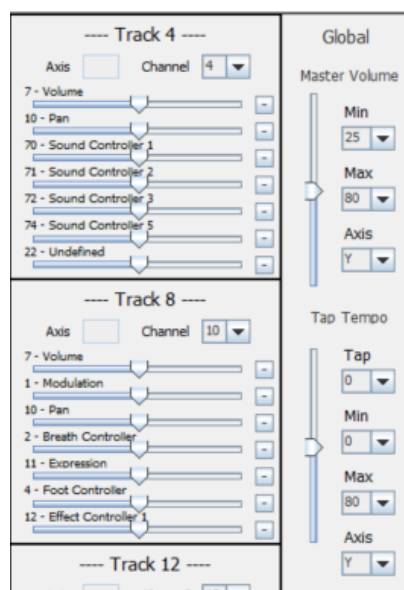


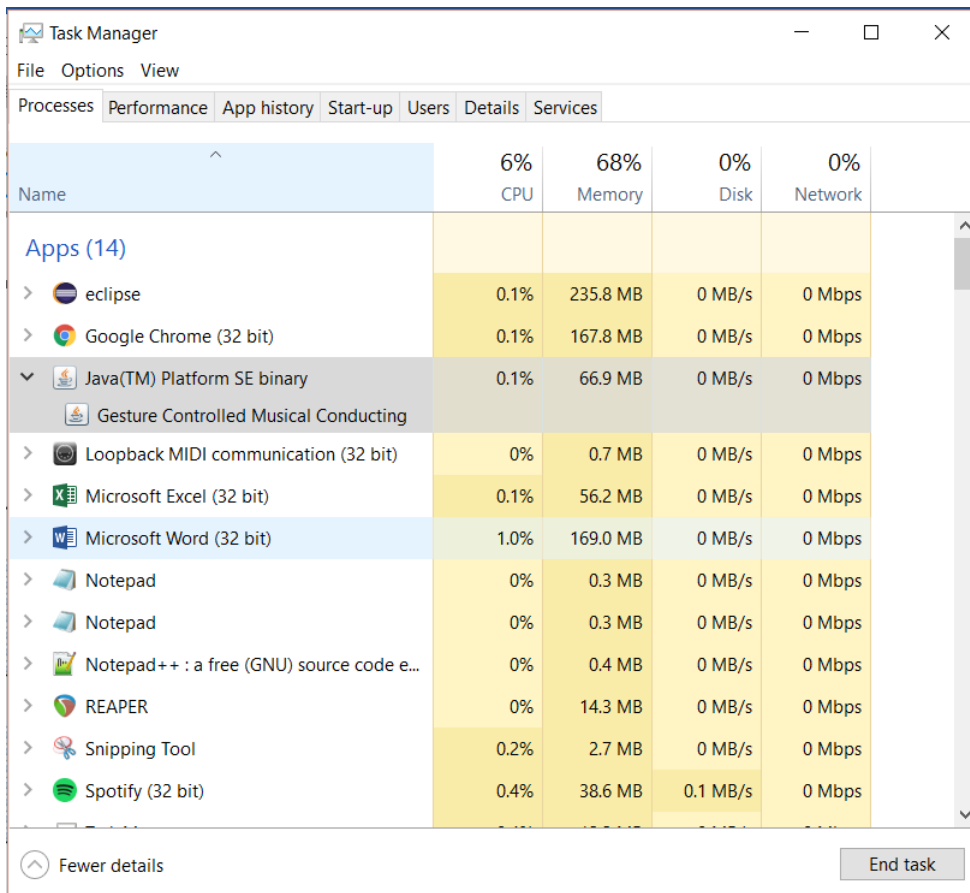
Figure 60 – Global Actions

Gesture Controlled Musical Conducting

Requirement: The program must run on the Windows operating system.

Status: **Passed**

This requirement was one of the easiest to pass. The screenshot below of my task manager shows the solution running comfortably on Windows 10. As the application is written in Java it may be easy to expand the application onto more platforms with some extra development time.



Task Manager				
File Options View				
Processes Performance App history Start-up Users Details Services				
Name	6% CPU	68% Memory	0% Disk	0% Network
Apps (14)				
> eclipse	0.1%	235.8 MB	0 MB/s	0 Mbps
> Google Chrome (32 bit)	0.1%	167.8 MB	0 MB/s	0 Mbps
Java(TM) Platform SE binary	0.1%	66.9 MB	0 MB/s	0 Mbps
Gesture Controlled Musical Conducting				
> Loopback MIDI communication (32 bit)	0%	0.7 MB	0 MB/s	0 Mbps
> Microsoft Excel (32 bit)	0.1%	56.2 MB	0 MB/s	0 Mbps
> Microsoft Word (32 bit)	1.0%	169.0 MB	0 MB/s	0 Mbps
> Notepad	0%	0.3 MB	0 MB/s	0 Mbps
> Notepad	0%	0.3 MB	0 MB/s	0 Mbps
> Notepad++ : a free (GNU) source code e...	0%	0.4 MB	0 MB/s	0 Mbps
> REAPER	0%	14.3 MB	0 MB/s	0 Mbps
> Snipping Tool	0.2%	2.7 MB	0 MB/s	0 Mbps
> Spotify (32 bit)	0.4%	38.6 MB	0.1 MB/s	0 Mbps

Figure 61 – Windows Task Manager

Gesture Controlled Musical Conducting

Requirement: The system window must be small enough to see both the system and the digital audio software.

Status: **Failed**

Initially this was a priority – it was considered important to be able to see both the system and the digital audio software. Unfortunately due to the amount of controls within the solution, it has made making the window size small difficult to implement. Figure 62 below shows a screenshot of my laptop screen and it is quite hard to see both application windows. For the music artists, most professional studios have either a large monitor or multiple monitors so in some circumstances this will not be a problem. Figure 62 shows what the application looks like on my laptop screen and Figure 63 shows what a typical digital music studio may look like with a multi-screen setup:

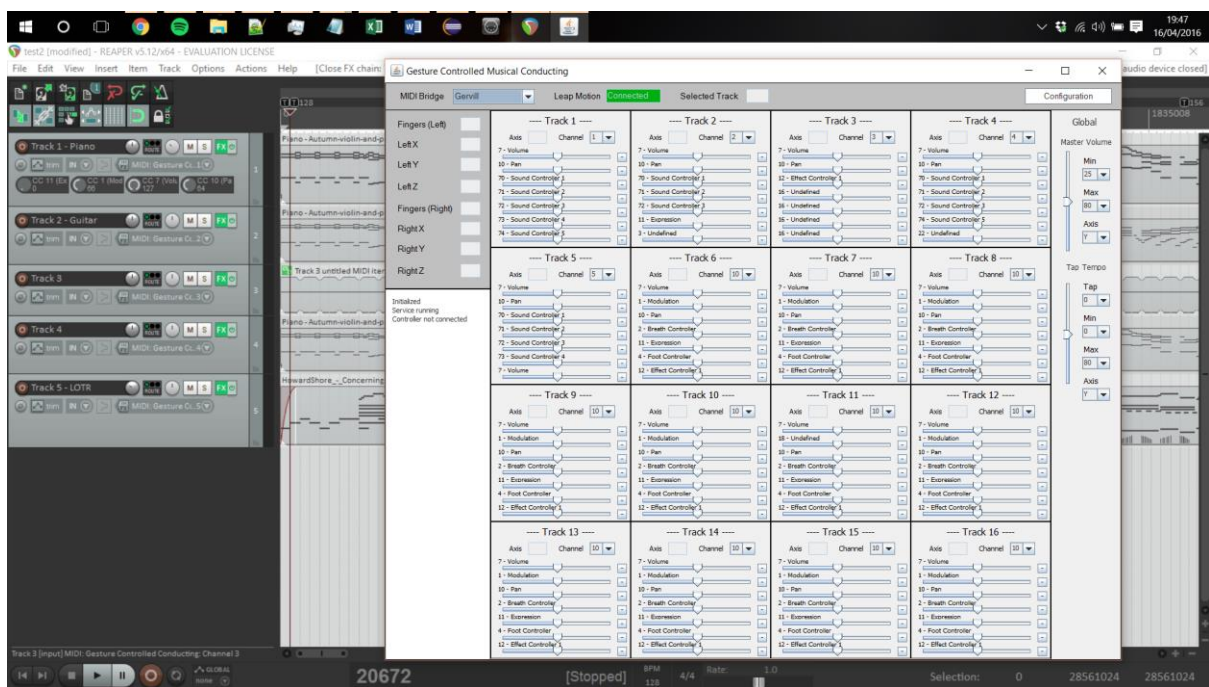


Figure 62 – Application and DAW on One Screen



Figure 63 – Example of a Home Recording Studio [74]

Gesture Controlled Musical Conducting

Requirement: The controls for the system should feel natural and be easy to learn by the average user.

Status: **Partially Passed**

The test results show that the application has an initial steep learning curve. The controls are not obvious when first using the system but once the users were showed how the controls work then they could easily perform the required gestures. The gestures themselves may not need changing, but more work is required to help the users learn the gestures without someone showing them.

Comments from the users relating to the requirement include:

- “Instructions [are needed] to inform the user of how the controls work either in program or documentation”
- “It was good and easy to use after being told how the controls work”

Requirement: The user interface of the system should be simple enough to be used by a user with low computing skills.

Status: **Passed**

Despite being challenging to learn the controls, most users complimented the design of the User Interface. This requirement has been passed most users liked the way that the controls give feedback to the user interface, which helped them control the application. The UI is discussed in-depth in 4.4.3 and can be seen in Figure 64.

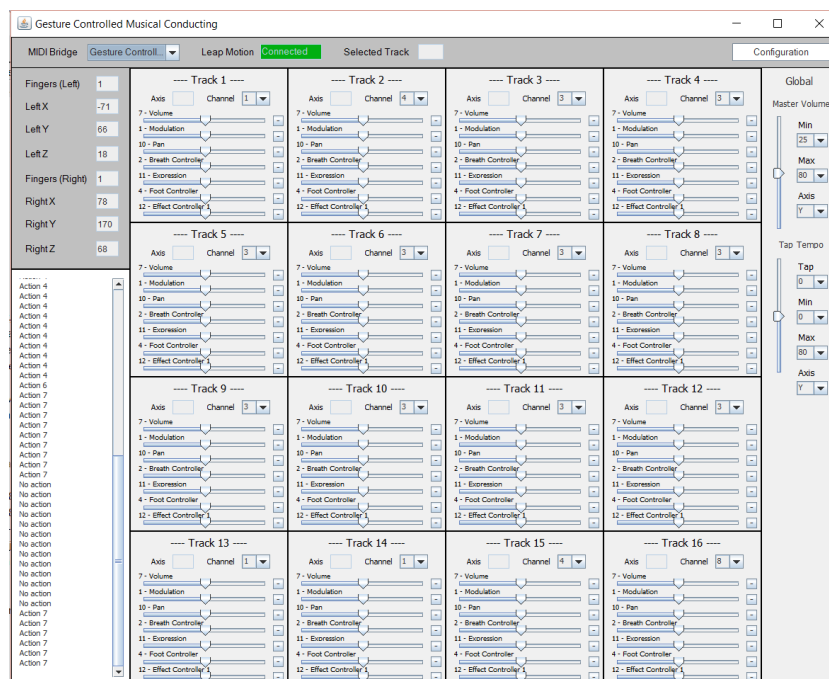


Figure 64 – The User Interface

5.6 Critical Appraisal

This section will present a critical appraisal of the project as a whole. The strengths and weaknesses for the project have come from the test results as they have enabled me to study the system in-depth to determine its strengths and weaknesses. A piece of software is never perfect, so in order to understand where the project needs to be developed we must understand what makes it a good project and how it can be improved.

5.6.1 Weaknesses

- **System learning curve:** The system has been found to be hard to use by users before they receive a demonstration. Some users could perform basic commands without guidance but most didn't know where to start. This is due to a combination of two things, firstly the hardware was completely new to the users as none of the test subjects had interacted with the leap motion sensor, the other factor was down to lack of practice – once the users were shown how to use the control the solution they became a lot more familiar with it. As a comparison, its similar in some aspect to leaning an instrument for the first time – often it requires someone to teach you and a lot of practice.
- **Input reliability:** The Leap Motion Sensor occasionally loses tracking of a hand during a performance. This is a weakness because it could potentially affect the reliability of the system which could cause the user experience to suffer. To eliminate this problem, time will be needed to determine how to optimise the hardware to implement smoother tracking.
- **Configuration period:** Time required spent having to setup and configure the software takes too long, which might be considered a weakness. The problem is that due to the type of solution, it requires configuration across all three components that make up the system. This is unfortunately unavoidable due to the mapping of the commands and the link required to send the data.
- **Single platform:** The system can currently only run on the Windows operating system, which is a weakness as it limits the audience that can experience the software. This is a weakness that most projects face, due to the time restriction it needed to be focused on one platform.
- **Window size:** As the overall interaction is across 2 pieces of software, the user needs to be able to see both the solution and the digital audio workstation on one screen. Due to design limitations this was not possible, which is why one of the requirements has failed. This is a system weakness as it may put people off using the software.

5.6.2 Strengths

- **User Interface:** A functional UI that has received very positive feedback from the test subjects during the acceptance testing. The interface is well designed to integrate with the controls so that appropriate feedback is given at all times.
- **Innovative hardware:** A strength of the system is that it is a good demo to show users who have not experienced the capabilities of gesture-control based hardware. Most users had not experienced gesture control prior to testing the project and they all commented that the hardware was 'cool' and 'enjoyable' to use.
- **Creative controls:** Once the users got over the hurdle of learning the basic controls, the system becomes simple and fluid to use. Most users said the commands are simple enough to perform with two hands and are not over-complicated. If the learning process is taken out, the design of the controls becomes one of the system's main strengths.
- **Adaptable:** The system is configurable which makes it adaptable to a variety of user configurations and purposes. Depending on the performance, an artist can set a variety of actions that are individually configurable; this makes the system compatible with a lot of performance styles which is a big strength of the system.
- **Useful solution:** After experimenting with users to test the capabilities of the system, most users agree that a complete professional would find the system extremely useful to use over the conventional MIDI controller. If a user already has the experience using a digital audio workstation, then it becomes very easy to map the solution to commands.
- **Entertaining system:** An entertaining system that is a pleasure to use by people that interact with it. As the system uses innovative technology to apply fun effects to music it makes using the system enjoyable to use as it is a breath of fresh air compared to the traditional input methods most users are used to. If a system is fun to use it can definitely be considered a strength.
- **Accessible:** The system is very accessible to a wide range of audiences. A Leap Motion Controller is a lot cheaper than the next best alternative and the interaction with the system is also accessible by users that struggle with the conventional control methods due to physical limitations. This was recognised by one of the test subjects who stated that "I can also see lots of applications for composers and musicians with disabilities who struggle with normal keyboards - this application could enable them access to tools in a way that they haven't experienced before."

6 Future Work

This section will discuss in detail the features of the project that can be considered as future work. The scope for the project was quite large and as a result there are parts of the implementation that were not created due to the time restriction of the project. Some ideas for future work have been realised through testing and user feedback. If anyone wishes to develop the project further, then this section will be a useful starting point to understand what could potentially be achieved with more development.

6.1 Increased Quantity of Complex Actions

Digital audio workstations have an extremely large number of commands and actions that can be used to alter the performance. With more development time, more gestures could be implemented to increase the quantity of possible actions that a user can perform. To add further actions, it would require some modification to the implementation of the controls. There is an endless number of commands that could be implemented but maintaining simple controls must be taken into consideration. Having too many possible actions may quickly over-complicate the control system which could affect the users experience. Actions to be implemented could include:

- Solo a track (silence all tracks other than the one selected).
- Mute a track (silence an individual track).
- Group tracks to be able to control them together.
- Reset a track back to default configuration settings.
- Skip or rewind a track.

Once the above features have been implemented into the solution, it is incredibly easy to map them within Reaper. In Reaper, there is a global actions list where DAW commands can be mapped the same was as instrumental effects (4.4.2). There is a number of actions per track that can be mapped to CC commands:

Filter: Track 01		
Shortcut	Description	State
MIDI Chan 1 CC 1	Custom: Track param 01 Adjust track FX parameter 01 (MIDI CC/OSC only)	
	Group: Select all tracks in group 01	
F4	Screensets: Load track view #01	
Shift+F4	Screensets: Save track view #01	
	Track: Select FX envelope 01	
	Track: Select track 01	
	Track: Set mute for track 01 (MIDI CC/OSC only)	
MIDI Chan 1 CC 10	Track: Set pan for track 01 (MIDI CC/OSC only)	
	Track: Set solo for track 01 (MIDI CC/OSC only)	
	Track: Set stereo width (or right channel pan) for track 01 (MIDI CC/OSC o...	
MIDI Chan 1 CC 7	Track: Set volume for track 01 (MIDI CC/OSC only)	
	Track: Toggle FX bypass for track 01	off
	Track: Toggle mute for track 01	off
	Track: Toggle record arm for track 01	off
	Track: Toggle solo for track 01	off

Figure 65 – Actions in Reaper that Can Be Mapped

6.2 Further Analysis and Testing

Due to the restriction of time for the project, extensive testing on a large number of users was not feasible. Increasing the amount of test subjects might help to gain more feedback and analysis so that the solution can be developed further. There are many possible mediums that could be explored to gain more feedback, these could include:

- Releasing the project on the Leap Motion Store [5] might help to get the project tested further. Applications that are submitted to the store get checked before approval which results in a delay, so it would not be possible to gain enough feedback through this platform before the project was completed. Once the store requirements have been met, the application process is straightforward.
- Gaining targeted feedback from artists and professionals who are likely to use the system to gain a better understanding of what is required to improve the software.
- Releasing the code on GitHub (an open-source code repository) will allow other developers to easily download, modify and improve the code. This is very easy to do as it involves uploading the code to a website for others to download and improve.

6.3 Documentation

During the testing, users were taught how to use the solution through demonstration. One of the popular points raised by the testing was the requirement to have documentation, video or tutorial that teaches the controls to the user in the situation when a demonstration is not possible.

It is possible an alternative workaround could be developed into the software where the solution hints at the controls when required. Popular gesture software does this well by showing the action the user needs to perform when it is selected, however in this solution due to the number of controls it was not possible to squeeze control hints into the UI without making the window larger. Further work may be required to understand the best approach to tackle this problem.

6.4 Input Combinations

Combining the solution with other forms of input may be an interesting way of increasing the number of controllable actions. One user from the feedback voiced an idea regarding combining voice commands with the gestures to allow for more control. Adding a new input to the project could potentially require a few more weeks of development time as it would require the developers to use new tools and APIs. Experimentation with other forms of input may be a good way to make the project even more interactive, this could include:

Facial Gestures

This would involve recording the users face and mapping certain facial expressions to MIDI commands [51]. Most laptops have cameras built into them so this may not necessarily require purchasing additional hardware.



Figure 66 – Facial Expression Recognition [75]

A camera can determine the facial expression by mapping points to the users face, machine learning techniques are then used to determine which emotion the user is expressing. An example use for the MIDI controller could be to speed up or slow down the music depending on if the user is displaying happy or sad emotions. Most laptops come with a built-in camera and they are also cheap to buy externally for PCs, costing around £14.99 on Amazon [52]; this means it would not impact the overall cost of the project significantly.

Mind control

Mind control would work similar to facial gestures [53]; experimental devices have been developed to the point where 'someone wearing a headset can manipulate objects on a screen' using mind brain waves. If possible, users could use brain waves to add another form of interaction with the system, with different emotions 'triggering' the particular style of music they represent. This is a very experimental technology that is rather expensive, the Emotiv headset (Figure 67) costs \$399 [54], so compared to the camera for facial recognition it would be an expensive addition to the project.



Figure 67 – The Emotiv Mind Control Device [76]

7 Conclusion

This project has looked at whether gestural input devices could be used to effectively conduct music – overall I believe this has been a success. The usability tests conducted on the final solution proved this, as users classed the solution as fun and enjoyable to use, suggesting they had a positive experience. Despite some flaws, gesture control is a useful and exciting new input platform for controlling music which has been realised through the Leap Motion Controller.

Most of the must-have project requirements were all achieved, the result being a robust system that can show off what the hardware is capable of. Despite being a success in terms of functionality, there is always room for development; to truly understand if the project is useful in a professional setting, further testing will need to be conducted on users that have a high level of experience with digital music production; only once the users in this category are happy then the project can be classed as truly useful.

Users who had no experience with gesture control faced a sharp learning curve when using the solution for the first time. Even though the software was never intended to be mastered without practice, further work will be needed to help users learn how to properly operate the system controls and hardware; this may be achieved through documentation or UI changes; these are completely possible but will require more time spent evolving the project.

One of the research questions asked at the start of the project was ‘Could gesture controllers replace traditional MIDI controllers?’. In my opinion I don’t think that gesture control will replace MIDI controllers, but I feel that this project has proved it should be taken seriously as an alternative. There are many situations when gesture control would be preferable to a standard MIDI controller, such as when the user does not have a lot of money to spend or if they have a disability that makes using standard controllers difficult. Whether it is used solely or to complement existing controllers, gesture control definitely has a place in a digital music studio and I can see gesture control playing a huge part in the future of digital music production.

8 Reflection and Learning

8.1 Reflection

Overall I have really enjoyed this project as I have gained a lot of new skills and lessons to take forward into future projects. When I decided to do the project, it was because I had an interest in exploring the uses of the Leap Motion Controller - I had previously tried the hardware when visiting a gadget convention and it excited me when I saw it being demoed. I had never heard of MIDI, used a digital audio workstation or explored CC commands, which made the project a huge learning experience.

I am glad I was able to study the concepts of digital music as I found it incredibly interesting, over the course of this project I have developed a great appreciation for the different components that go into the creation of digital music.

If I could start the project again I would do a few things differently. Firstly, one of the parts of the project I could have done differently was spent a bit more time on the time-plan. Basic configuration and setup of the three project components took a lot longer than was planned for in the time-plan. A large amount of time was spent learning how to perform basic functions within the digital audio software, this was not taken into account when writing the initial time plan which caused a delay in implementation. In future projects I will make sure to take greater care to evaluate how long it will actually take to make sure the time plan is as accurate as possible.

Secondly, I would have liked to allocate more time for testing stage of the project. User testing was the only formal testing I was able to conduct as I ran out of time after the implementation; overall I was happy with the results but I believe it could have been more efficiently organised.

8.2 Learning

This project has enabled me to learn how to use some interesting new pieces of software. Despite having a steep learning curve due to its sheer complexity, I am glad that I now have a good level of experience using a popular digital audio workstation as it is an incredibly powerful tool.

In terms of the development process, I have gained a lot of skills that can not only be applied to future projects, but also real-world business situations. Transferrable skills such as being able to use Github are extremely important as they are critical to managing project code effectively. My general coding skills have also been developed as I have been able to use libraries for the first time such as WindowBuilder and Java Prefs, enabling me to expand my experience of tools.

This project gave me the opportunity to develop a greater understanding of innovative input methods, especially gesture-control hardware. It was interesting to learn the concepts and

methods that make up the hardware and how they could be utilised within my code. Time was spent learning the recommended implementation styles for the Leap Motion Sensor and in future projects I will be able to save time by implementing them straight away.

On the whole this project has helped me enhance my research, project management and professional writing skills. Based on University and internship experience I feel I already had a solid foundation of basic project skills but I had never experienced having sole responsibility for a large-scale project. Taking on this project has helped me combine everything I been taught and put it into practice by going through the complete lifecycle of a project.

Speaking to the project supervisor, Dave Marshall, on a weekly basis enabled me to clarify what I had learned the previous week and how it could be applied to the project moving forward. Dave's knowledge of digital music is very impressive and it was useful that he was able to guide me with his expertise; this experience has taught me how important is it to speak to subject experts where possible to gain a second opinion.

As a result of managing my own project, I have a greater awareness of the problems that may occur. I have learned some fundamental lessons about project management that have helped me to become a better-rounded developer, these lessons will be taken forward into future projects.

List of Acronyms

MIDI - Musical Instrument Digital Interface

CC - Control Change Messages

UI – User Interface

UX – User Experience

DAW – Digital Audio Workstation

API - Application Program Interface

Appendices

Appendix A – CC Commands Table

This table is a reference to an online MIDI CC table, found at:
<http://nickfever.com/music/midi-cc-list>

CC Number	MIDI CC Purpose	MIDI CC Description
MIDI CC 0	Bank Select	Allows user to switch bank for patch selection. Program change used with Bank Select. MIDI can access 16,384 patches per MIDI channel.
MIDI CC 1	Modulation	Generally this CC controls a vibrato effect (pitch, loudness, brightness). What is modulated is based on the patch.
MIDI CC 2	Breath Controller	Often times associated with aftertouch messages. It was originally intended for use with a breath MIDI controller in which blowing harder produced higher MIDI control values. It can be used for modulation as well.
MIDI CC 3	Undefined	
MIDI CC 4	Foot Controller	Often used with aftertouch messages. It can send a continuous stream of values based on how the pedal is used.
MIDI CC 5	Portamento Time	Controls portamento rate to slide between 2 notes played subsequently.
MIDI CC 6	Data Entry Most Significant Bit(MSB)	Controls Value for NRPN or RPN parameters.
MIDI CC 7	Volume	Control the volume of the channel
MIDI CC 8	Balance	Controls the left and right balance, generally for stereo patches. 0 = hard left, 64 = center, 127 = hard right
MIDI CC 9	Undefined	
MIDI CC 10	Pan	Controls the left and right balance, generally for mono patches. 0 = hard left, 64 = center, 127 = hard right
MIDI CC 11	Expression	Expression is a percentage of volume (CC7).
MIDI CC 12	Effect Controller 1	Usually used to control a parameter of an effect within the synth/workstation.
MIDI CC 13	Effect Controller 2	Usually used to control a parameter of an effect within the synth/workstation.
MIDI CC 14	Undefined	

Gesture Controlled Musical Conducting

MIDI CC 15	Undefined	
MIDI CC 16 – 19	General Purpose	
MIDI CC 20 – 31	Undefined	
MIDI CC 32 – 63	Controller 0-31 Least Significant Bit (LSB)	
MIDI CC 64	Damper Pedal / Sustain Pedal	On/Off switch that controls sustain. (See also Sostenuto CC 66) 0 to 63 = Off, 64 to 127 = On
MIDI CC 65	Portamento On/Off Switch	On/Off switch 0 to 63 = Off, 64 to 127 = On
MIDI CC 66	Sostenuto On/Off Switch	On/Off switch – Like the Sustain controller (CC 64), However it only holds notes that were “On” when the pedal was pressed. People use it to “hold” chords” and play melodies over the held chord. 0 to 63 = Off, 64 to 127 = On
MIDI CC 67	Soft Pedal On/Off Switch	On/Off switch - Lowers the volume of notes played. 0 to 63 = Off, 64 to 127 = On
MIDI CC 68	Legato FootSwitch	On/Off switch - Turns Legato effect between 2 subsequent notes On or Off. 0 to 63 = Off, 64 to 127 = On
MIDI CC 69	Hold 2	Another way to “hold notes” (see MIDI CC 64 and MIDI CC 66). However notes fade out according to their release parameter rather than when the pedal is released.
MIDI CC 70	Sound Controller 1	Usually controls the way a sound is produced. Default = Sound Variation.
MIDI CC 71	Sound Controller 2	Allows shaping the Voltage Controlled Filter (VCF). Default = Resonance - also(Timbre or Harmonics)
MIDI CC 72	Sound Controller 3	Controls release time of the Voltage controlled Amplifier (VCA). Default = Release Time.
MIDI CC 73	Sound Controller 4	Controls the “Attack’ of a sound. The attack is the amount of time it takes forthe sound to reach maximum amplitude.
MIDI CC 74	Sound Controller 5	Controls VCFs cutoff frequency of the filter.
MIDI CC 75	Sound Controller 6	Generic – Some manufacturers may use to further shave their sounds.
MIDI CC 76	Sound Controller 7	Generic – Some manufacturers may use to further shave their sounds.
MIDI CC 77	Sound Controller 8	Generic – Some manufacturers may use to further shave their sounds.
MIDI CC 78	Sound Controller 9	Generic – Some manufacturers may use to further shave their sounds.

Gesture Controlled Musical Conducting

MIDI CC 79	Sound Controller 10	Generic – Some manufacturers may use to further shave their sounds.
MIDI CC 80	General Purpose MIDI CC Controller	Generic On/Off switch 0 to 63 = Off, 64 to 127 = On
MIDI CC 81	General Purpose MIDI CC Controller	Generic On/Off switch 0 to 63 = Off, 64 to 127 = On
MIDI CC 82	General Purpose MIDI CC Controller	Generic On/Off switch 0 to 63 = Off, 64 to 127 = On
MIDI CC 83	General Purpose MIDI CC Controller	Generic On/Off switch 0 to 63 = Off, 64 to 127 = On
MIDI CC 84	Portamento CC Control	Controls the amount of Portamento.
MIDI CC 85 – 90	Undefined	
MIDI CC 91	Effect 1 Depth	Usually controls reverb send amount
MIDI CC 92	Effect 2 Depth	Usually controls tremolo amount
MIDI CC 93	Effect 3 Depth	Usually controls chorus amount
MIDI CC 94	Effect 4 Depth	Usually controls detune amount
MIDI CC 95	Effect 5 Depth	Usually controls phaser amount
MIDI CC 96	(+1) Data Increment	Usually used to increment data for RPN and NRPN messages.
MIDI CC 97	(-1) Data Decrement	Usually used to decrement data for RPN and NRPN messages.
MIDI CC 98	Non-Registered Parameter Number LSB (NRPN)	For controllers 6, 38, 96, and 97, it selects the NRPN parameter.
MIDI CC 99	Non-Registered Parameter Number MSB (NRPN)	For controllers 6, 38, 96, and 97, it selects the NRPN parameter.
MIDI CC 100	Registered Parameter Number LSB (RPN)	For controllers 6, 38, 96, and 97, it selects the RPN parameter.
MIDI CC 101	Registered Parameter Number MSB (RPN)	For controllers 6, 38, 96, and 97, it selects the RPN parameter.
MIDI CC 102 – 119	Undefined	

Gesture Controlled Musical Conducting

MIDI CC 120 to 127 are "Channel Mode Messages."		
MIDI CC 120	All Sound Off	Mutes all sounding notes. It does so regardless of release time or sustain. (See MIDI CC 123)
MIDI CC 121	Reset All Controllers	It will reset all controllers to their default.
MIDI CC 122	Local On/Off Switch	Turns internal connection of a MIDI keyboard/workstation, etc. On or Off. If you use a computer, you will most likely want local control off to avoid notes being played twice. Once locally and twice when the note is sent back from the computer to your keyboard.
MIDI CC 123	All Notes Off	Mutes all sounding notes. Release time will still be maintained, and notes held by sustain will not turn off until sustain pedal is depressed.
MIDI CC 124	Omni Mode Off	Sets to "Omni Off" mode.
MIDI CC 125	Omni Mode On	Sets to "Omni On" mode.
MIDI CC 126	Mono Mode	Sets device mode to Monophonic.
MIDI CC 127	Poly Mode	Sets device mode to Polyphonic.

Appendix B – User Feedback Breakdown

Questions:

1. In your own words, what was your experience of using the software?
2. Is there any feature regarding MIDI control (volume, pitch, tone, expression etc.) that you feel is missing from the project?Gr
3. In your opinion what else could be improved about the project?
4. On a scale of 1 to 10, how easy did you find the project to use BEFORE you received a demonstration of how it worked? (1 being easy and 10 being very hard)
5. On a scale of 1 to 10, how easy did you find the project to use AFTER you received a demonstration of how it worked? (1 being easy and 10 being very hard)
6. Any further comments or opinions about the project?

Table of Responses

Question	Test Subject	Response
1	1	How well it can integrate with existing MIDI applications with minimal set-up needed. - Amount of control given to the user with regards to messages sent - Intuitive hand gestures used to control the software - Well laid out GUI
	2	Kieran showed me to control it and then let me do it myself so I could work it out and see the effect it had on the track.
	3	It was fun to use gestures to interact with software. I had not used a Leap Motion before and it was a good demo software to show what the Leap is capable of.
	4	It was good and easy to use after being told how the controls work
	5	Overall very good, difficult to get to grips with at first but once the basic commands are understood it was easy to use.
	6	It was weird and made my arms ache. It seemed to work well though. Felt like a lot more effort than just using the mouse.
	7	I was introduced first to how the base of the project worked - playing around with hand controls, this helped to get a sense of how to move before using the musical software. I then was introduced to the musical controls, playing around with levels and learning the hand controls for this - which was really fun! I've never had so much fun using a music computer program before!
2	1	Using information from the MIDI software to filter which types of parameters can be added to particular MIDI tracks
	2	Pitch might be useful for remixes.
	3	It would be cool to be able to change the tempo of each track, but generally I felt like all the main MIDI control options were possible.
	4	No
	5	None
	6	I don't know.
	7	I don't think so.
3	1	Clarity over what you're currently change i.e. bolding a parameter such as 'Volume' when you've selected it and are changing it - Adding help to the

Gesture Controlled Musical Conducting

		application such as putting hand gesture symbols next to parameters/tracks - Add a graphic that shows a plot of what the Leap Motion thinks you're doing with your hands instead of/alongside the co-ordinate data currently shown - A scaled down version of the GUI that shows the currently selected track - to view when inside other software
	2	I think the only thing is that the Leap Motion is not great at picking up my hands (although maybe I was just doing it wrong). I would say that perhaps using a similar but more receptive equivalent, or waiting for an upgrade would be good.
	3	I thought that the project did a very good job of allowing me to control the audio with hand gestures. Continuing in the theme of natural/intuitive user interaction, it would be very cool to be able to use voice commands to change the tracks that are being played (however, I am aware that this deviates quite a lot from the core purpose of the project).
	4	Instructions added to inform the user of how the controls work either in program or documentation
	5	Maybe add a way to change instrument timbre with a gesture, for instance, to change violin to pizzicato as a opposed to lagarto and vice versa.
	6	Maybe have photos of the instruments you can point at to control, rather than just having numbered tracks. A tutorial would be useful too.
	7	On the level for modulation, the scale could do with labels to tell you what key you are in/going to - so that you could work out the transposing instruments and whether this is a good key for them. This could also be applied to the other level bars - so that you can tell just how much you are changing the music from its original levels.
4	1	7
	2	10
	3	8
	4	10
	5	9
	6	8
	7	8
5	1	5
	2	3
	3	2
	4	3
	5	3
	6	3
	7	2
6	1	I am not a musical expert at all which is why I found it quite hard to use to start with. However, I think that once you know how to control the software, it would be quite easy to learn how to make it sound good.
	2	None
	3	After being told what all the gestures were to control the software I felt like it didn't take long before they felt natural. However, there were some cases where I found it difficult to get the Leap to recognise my gestures fully.
	4	None
	5	Nope

Gesture Controlled Musical Conducting

	6	I feel like I might have found it easier to use before the demonstration now that I've played around with my own Leap and am familiar with the gestures etc. that apps use. Since only people with Leaps will be able to use the app anyway, people thinking it's too hard before the demonstration might not be so relevant.
	7	I really enjoyed using this, and although hard at first, I think with some practice this would be a much easier way of changing music quickly. I can also see lots of applications for composers and musicians with disabilities who struggle with normal keyboards - this application could enable them access to tools in a way that they haven't experienced before.

Bibliography

- [1] IFPI, IFPI publishes Digital Music Report 2015 [online] Available at: <http://www.ifpi.org/news/Global-digital-music-revenues-match-physical-format-sales-for-first-time> Last Accessed: 18/04/2016
- [2] Leap Motion, Leap Motion Home [online] Available at: <https://www.leapmotion.com/> Last Accessed: 27/04/2016
- [3] Microsoft, Kinect [online] <https://developer.microsoft.com/en-us/windows/kinect> Last Accessed: 27/04/2016
- [4] Uses of Gesture Technology [online] Available at: <http://www.techtree.com/content/features/4254/5-useful-applications-gesture-technology.html> Last Accessed: 27/04/2016
- [5] Leap Motion, Leap Motion App Store [online] Available at: <https://apps.leapmotion.com/> Last Accessed: 18/04/2016
- [6] AgileMethodology.org, The Agile Movement [online] Available at: <http://agilemethodology.org/> Last Accessed: 18/04/2016
- [7] Leap Motion, Leap Motion System Requirements [online] Available at: <https://support.leapmotion.com/entries/39315178-What-are-the-system-requirements-> Last Accessed: 18/04/2016
- [8] Wikipedia, Usage share of operating systems [online] Available at: https://en.wikipedia.org/wiki/Usage_share_of_operating_systems Last Accessed: 18/04/2016
- [9] Wikipedia, Compact Disk [online] Available at: https://en.wikipedia.org/wiki/Compact_disc Last Accessed: 27/04/2016
- [10] Wikipedia, Digital Audio [online] Available at: https://en.wikipedia.org/wiki/Compact_Disc_Digital_Audio Last Accessed: 27/04/2016
- [11] Expanded Ramblings, By the Numbers: 70 Amazing Snapchat Statistics [online] Available at: <http://expandedramblings.com/index.php/snapchat-statistics/> Last Accessed: 18/04/2016
- [12] Myo, Myo Armband [online] Available at: <https://www.myo.com/> Last Accessed: 18/04/2016
- [13] Gizmag, Gest Glove has Gesture Control on Hand [online] Available at: <http://www.gizmag.com/gest-gesture-controller-glove/40174/> Last Accessed: 18/04/2016
- [14] Leap Motion [online] Available at: <https://www.leapmotion.com/> Last Accessed: 27/04/2016

- [15] Quora, How does Microsoft's Kinect work from a technology standpoint? [online] Available at: <https://www.quora.com/How-does-Microsofts-Kinect-work-from-a-technology-standpoint> Last Accessed: 27/04/2016
- [16] Instructables, What is MIDI? [online] Available at: <http://www.instructables.com/id/What-is-MIDI/> Last Accessed: 18/04/2016
- [17] Sequencer, MIDI HD – THE NEW STANDARD! [online] Available at: <http://www.sequencer.de/blog/midi-hd-the-new-standard/21179> Last Accessed: 27/04/2016
- [18] MIDI System [online] Available at: http://www.cs.cf.ac.uk/Dave/Multimedia/PDF/06_MIDI.pdf Last Accessed: 27/04/2016
- [19] Producer Spot, Top 10 Best DAW 2015 – Best Music Software [online] Available at: <http://www.producerspot.com/top-best-daw-2015-best-music-software> Last Accessed: 18/04/2016
- [20] Leap Motion, How Does the Leap Motion Controller Work [online] Available at: <http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/> Last Accessed: 18/04/2016
- [21] Leap Motion, Leap API Overview [online] Available at: https://developer.leapmotion.com/documentation/java/devguide/Leap_Overview.html Last Accessed: 27/04/2016
- [22] Johnny Holland, 10 Interactive Ways to Make Music [online] Available at: <http://johnnyholland.org/2008/12/10-new-interactive-ways-to-make-music/> Last Accessed: 18/04/2016
- [23] Leap Motion, GECO MIDI [online] Available at: <https://apps.leapmotion.com/apps/geco-midi/osx> Last Accessed: 18/04/2016
- [24] Leap Motion Store, Muse [online] Available at: <https://apps.leapmotion.com/apps/muse/windows> Last Accessed: 27/04/2016
- [25] Leap Motion Store, Tekh Tonic [online] Available at: <https://apps.leapmotion.com/apps/tekh-tonic/windows> Last Accessed: 27/04/2016
- [26] Leap Motion Store, Swoosh [online] Available at: <https://apps.leapmotion.com/apps/swoosh/osx> Last Accessed: 27/04/2016
- [27] Synthopia, Audiotouch, The \$200 Multitouch Music Table [online] Available at: <http://www.synthtopia.com/content/2009/05/31/audiotouch-the-200-multitouch-music-table/> Last Accessed: 27/04/2016
- [28] Remidi Pro, Remidi Glove [online] Available at: <http://remidi-pro.com/> Last Accessed: 18/04/2016

- [29] Leap Motion, Leap SDK [online] Available at: <https://developer.leapmotion.com/documentation/java/index.html> Last Accessed: 18/04/2016
- [30] Github, Github Homepage [online] Available at: <https://github.com/> Last Accessed: 18/04/2016
- [31] Reaper, Reaper Homepage [online] Available at: <http://www.reaper.fm/> Last Accessed: 18/04/2016
- [32] Tobias Erichsen, loopMIDI [online] Available at: <http://www.tobias-erichsen.de/software/loopmidi.html> Last Accessed: 18/04/2016
- [33] Eclipse, WindowBuilder [online] Available at: <https://eclipse.org/windowbuilder/> Last Accessed: 18/04/2016
- [34] Balsamiq, Balsamiq Mockups Homepage [online] Available at: <https://balsamiq.com/> Last Accessed: 18/04/2016
- [35] Gliffy, Gliffy Tool [online] Available at: <https://www.gliffy.com/> Last Accessed: 18/04/2016
- [36] Creately [online] Available at: <http://creately.com/> Last Accessed: 18/04/2016
- [37] Eclipse [online] Available at: <https://eclipse.org/> Last Accessed: 18/04/2016
- [38] Nick Fever, MIDI CC List [online] Available at: <http://nickfever.com/music/midi-cc-list> Last Accessed: 18/04/2016
- [39] Leap Motion, Leap Motion Interaction Box [online] Available at: <https://developer.leapmotion.com/documentation/java/api/Leap.InteractionBox.html> Last Accessed: 18/04/2016
- [40] Leap Motion, Menu Design Guidelines [online] Available at: https://developer.leapmotion.com/documentation/java/practices/Leap_Menu_Design_Guidelines.html?proglang=java Last Accessed: 18/04/2016
- [41] Wikipedia, The Magical Number Seven, Plus or Minus Two [online] Available at: https://en.wikipedia.org/wiki/The_Magical_Number_Seven,_Plus_or_Minus_Two Last Accessed: 18/04/2016
- [42] Eclipse Mars, Eclipse [online] Available at: <https://eclipse.org/mars/> Last Accessed: 27/04/2016
- [43] Leap Motion, Leap SDK [online] Available at: <https://www.leapmotion.com/setup> Last Accessed: 27/04/2016
- [44] Java, Java Sound API [online] Available at: <https://docs.oracle.com/javase/tutorial/sound/> Last Accessed: 18/04/2016
- [45] Eclipse, WindowBuilder [online] Available at: <https://eclipse.org/windowbuilder/> Last Accessed: 27/04/2016

[46] Java, Java Preferences [online] Available at:

<http://docs.oracle.com/javase/7/docs/technotes/guides/preferences/> Last Accessed: 18/04/2016

[47] Reaper [online] Available at: <http://www.reaper.fm/download.php> Last Accessed: 27/04/2016

[48] loopMIDI [online] Available at: <http://www.tobias-erichsen.de/software.html> Last Accessed: 27/04/2016

[49] The Practicing IT Project Manager, Types of Testing [online] Available at: <http://blog.practicingitpm.com/2012/05/27/types-of-testing/> Last Accessed: 18/04/2016

[50] Wrike, 15 Project Management Quotes to Live By (Infographic) [online] Available at: <https://www.wrike.com/blog/15-project-management-quotes-to-live-by-infographic/> Last Accessed: 18/04/2016

[51] iBug, Facial Analysis [online] Available at: <http://ibug.doc.ic.ac.uk/research/face-analysis/> Last Accessed: 27/04/2016

[521] Amazon, Logitech C270 HD Webcam [online] Available at: https://www.amazon.co.uk/Logitech-C270-HD-Webcam-Black/dp/B003R1O320/ref=sr_1_2?ie=UTF8&qid=1461756175&sr=8-2&keywords=web+camera Last Accessed: 27/04/2016

[53] HowStuffWorks, How the Emotiv EPOC Works [online] Available at: <http://electronics.howstuffworks.com/emotiv-epoc.htm> Last Accessed: 27/04/2016

[54] Emotiv, Emotiv Headset [online] Available at: <https://emotiv.com/> Last Accessed: 27/04/2016

Image References:

[55] E-Home Recording Studio, The Beginner's Guide to Digital Audio for Music Recording [online] Available at: <http://ehomerecordingstudio.com/digital-audio/> Last Accessed: 27/04/2016

[56] Wikipedia, Digital Audio [online] Available at: https://en.wikipedia.org/wiki/Digital_audio Last Accessed: 27/04/2016

[57] Digital Trends, Gesture Sensing Glove [online] Available at: <http://www.digitaltrends.com/cool-tech/gest-gesture-sensing-glove-kickstarter/> Last Accessed: 27/04/2016

[58] Mashable, Myo Control Armband [online] Available at: <http://mashable.com/2014/07/29/myo-motion-control-armband-hands-on/> Last Accessed: 27/04/2016

[59] PCMag, Leap Motion Controller [online] Available at: <http://www.pcmag.com/article2/0,2817,2422045,00.asp> Last Accessed: 27/04/2016

- [60] Microsoft, Kinect for Xbox One [online] Available at: <http://www.xbox.com/en-GB/xbox-one/accessories/kinect-for-xbox-one> Last Accessed: 27/04/2016
- [61] Vintage King Audio, NOVATION 25 SL MK II USB MIDI CONTROLLER [online] Available at: <http://vintageking.com/novation-25-sl-mk-ii-usb-midi-controller> Last Accessed: 18/04/2016
- [62] Pubnub, Motion-controlled Servos with Leap Motion & Raspberry Pi [online] Available at: <https://www.pubnub.com/blog/2015-08-19-motion-controlled-servos-with-leap-motion-raspberry-pi/> Last Accessed: 18/04/2016
- [63] Leap Motion, Leap Motion Overview [online] Available at: https://developer.leapmotion.com/documentation/java/devguide/Leap_Overview.html Last Accessed: 27/04/2016
- [64] Leap Motion, GECO MIDI [online] Available at: <https://apps.leapmotion.com/apps/geco-midi/windows> Last Accessed: 18/04/2016
- [65] Leap Motion Store, Muse [online] Available at: <https://apps.leapmotion.com/apps/muse/windows> Last Accessed: 27/04/2016
- [66] Leap Motion Store, Swoosh [online] Available at: <https://apps.leapmotion.com/apps/swoosh/osx> Last Accessed: 27/04/2016
- [67] Yahoo, Make any surface into a synthesizer with the Remidi T8 wearable synth controller [online] Available at: <https://www.yahoo.com/tech/surface-synthesizer-remidi-t8-wearable-181547261.html> Last Accessed: 18/04/2016
- [68] Eclipse, Windowbuilder Overview [online] Available at: <https://eclipse.org/windowbuilder/> Last Accessed: 27/04/2016
- [69] Digital Sound & Music, MIDI Components [online] Available at: <http://digitalsoundandmusic.com/6-1-2-midi-components/> Last Accessed: 03/-5/2016
- [70] Infusion Systems, loopMIDI [online] Available at: https://infusionsystems.com/support/quickstart/images/loopmidi_added.png Last Accessed: 03/05/2016
- [71] KVRAudio.com, Cockos releases REAPER 3 [online] Available at: http://www.kvraudio.com/news/cockos_releases_reaper_3_11646 Last Accessed 03/05/2016
- [72] Leap Motion, Interaction Box [online] Available at: <https://developer.leapmotion.com/documentation/java/api/Leap.InteractionBox.html> Last Accessed: 18/04/2016
- [73] Leap Motion, How does the Leap Motion Controller Work [online] Available at: <http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/> Last Accessed: 18/04/2016

[74] Hull University, Student Website [online] Available at: <http://www.hull.ac.uk/php/sbswdr/ecompastwork/0607/xstudentsites0607/321738/News%20Page.htm> Last Accessed: 27/04/2016

[75] iBug, Facial Analysis [online] Available at: <http://ibug.doc.ic.ac.uk/research/face-analysis/> Last Accessed: 27/04/2016

[76] Emotiv, Emotiv Headset [online] Available at: <https://emotiv.com/> Last Accessed: 27/04/2016

Recommended Further Reading:

Brandon Sanders, Mastering Leap Motion – 28 Nov 2014

Paul White, Basic MIDI, 2 Feb 2000

Andrew Brown, Making Music with Java Paperback – 3 May 2009