The electronic sensor bow: a new gestural control interface

Ben Murphy

University of Wollongong

Recommended Citation
UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.
The Electronic Sensor Bow:

A New Gestural Control Interface

Ben Murphy

University of Wollongong

Master of Arts (Research) Thesis 2011

This thesis is presented in fulfilment of the requirements for the award of the Degree of Master of Arts (Research) University of Wollongong
Abstract

This thesis discusses the ESBow (Electronic Sensor Bow). The ESBow is a traditional violin bow enhanced with electronic sensors for the creation of electronic chamber music. These sensors include two force sensing resistors, a tri-axial accelerometer and a trackball with select. Key issues regarding electronic violin controllers will also be examined in this thesis. This includes a discussion on the significance of extending the legacy of an existing instrument. Other issues discussed include mapping and the composed instrument, and the role of haptic feedback to the performer. Details of the ESBow project and its history will be discussed before the current prototype design is detailed and reviewed. This will include both the technical details of the bow as well as the objectives and ideals behind the bow. The remainder of this thesis will focus on compositional applications of the ESBow. This thesis will only address the use of the ESBow as a solo instrument.
Acknowledgements

The ESBow was made possible with the help of a number of people. I would like to thank Greg Schiemer not only for his guidance and advice but for introducing me to the world of electronic music design and helping me to turn an initial idea into a real and working instrument. I’d also like to thank Houston Dunleavy for his encouragement and advice over the years. I’d like to thank Matthew Ellis for suggesting I research the Arduino, a suggestion which proved a turning point in realising the ESBow. I’d also like to thank Mark Havryliv for his help with the original ESBow project. A great deal of thanks goes to Olena Cullen for all of her help over the years. I’d also like to thank all of the designers of electronic violin controllers that have provided me with inspiration simply by creating their instruments and telling the world about them. Finally, I’d like to thank my loving wife Michelle, for keeping me sane through both my Honours and Masters degrees.
Statement of Originality

All prototype designs for the ESBow described within this thesis are my own. Other instrument designers have influenced my designs through the construction of various electronic violin controllers. None of these have been directly copied and key influences on specific design features have been credited where possible. The hardware sensors and microcontrollers in each design were constructed by various manufacturers and are also credited where possible.

The Arduino codes used during the testing of hardware sensors were of my own design and based on the Arduino example codes available on the Arduino website (Arduino n.d.b). The Arduino codes used to communicate with PD were expanded from the default codes bundled with the Arduino2PD (Arduino2PD n.d.) and SimpleMessageSystem (SimpleMessageSystem n.d.) patches for PD.

The default PD to MIDI interface was expanded from the Arduino2PD patch (Arduino2PD n.d.) which was based on the SimpleMessageSystem patch (SimpleMessageSystem n.d.). The part of my patch that was taken from the Arduino2PD patch are the objects that interface with a communications port and polling of the Arduino’s sensor inputs. The original version of the Arduino2PD patch is shown in Figure 68 in Appendix B.

All other processes presented in PD are of my own design. This includes the preparation and sensitivity selections of each sensor and all mapping techniques. This includes the method of sensing the point of contact in relation to the two FSRs, techniques to differentiate
momentary events from clocked pulses in data streams and the functions of all expression objects. The PD objects themselves are available in the PD library and my original work is in their arrangement for techniques and effects. To the best of my knowledge no PD techniques have been copied from any other source without reference.

All effects used in AudioMulch are native to the software program. Mapping and configurations in each composition and demonstration are of my own design. Figures 15, 18, 19, 20 and 23 were created using AudioMulch automation tools as a makeshift oscilloscope.

All compositions and demonstration patches are my own work. All ideas are my own except where credited. All recorded samples used in compositions and demonstrations are from my own recordings.

The designs discussed in Chapter 4 and Appendices B and C of this thesis were designed and constructed entirely within the Masters timeframe although the concept of an electronic sensor bow was initially explored in my Honours thesis, together with a survey of electronic string instruments and a preliminary evaluation of electronic sensors in bow design. I briefly revisit both the survey - which is summarised in sub-chapter 3.1 of the Masters thesis- and sensor evaluation - summarised and reviewed in sub-chapters 3.3 and 3.4 of the Masters thesis.
Figure 1: The ESBow.
# Contents

Abstract .......................................................................................................................................... i  
Acknowledgements ....................................................................................................................... iii  
Statement of Originality .................................................................................................................. v  
Contents .......................................................................................................................................... ix  
List of Figures .................................................................................................................................. xiii  
Terms and Abbreviations ................................................................................................................ xvii  
DVD-ROM - Video Demonstrations ............................................................................................... xix  
DVD-ROM - Audio Recordings ....................................................................................................... xx  
Prologue ............................................................................................................................................ xxi  

1. The Evolving Violin ....................................................................................................................... 1  

2. Electronic Violin Controllers ................................................................................................... 5  
   2.1 Terms of Reference ................................................................................................................ 5  
   2.2 Extending the Musical Legacy of an Existing Instrument .................................................... 9  
   2.3 Mapping and the Composed Instrument .............................................................................. 11  
   2.4 Haptic Feedback .................................................................................................................. 14  
   2.5 Simulated Haptic Feedback ................................................................................................ 16  
   2.6 Haptic Feedback in Conventional Electronic Interfaces ....................................................... 19  
   2.7 Electronic Chamber Music ................................................................................................ 21  
   2.8 Connecting the Performer with the Music ......................................................................... 26  

3. The ESBow – The Incubation Process ..................................................................................... 29  
   3.1 Origins .................................................................................................................................. 29  
   3.2 The ESBow: An Overview ................................................................................................... 31
5.4 Martelé ................................................................. 83
5.5 Collé ........................................................................ 84
5.6 Spiccato ................................................................... 84
5.7 Jeté ........................................................................... 85
5.8 Sautillé ................................................................. 86
5.9 Chopping ........................................................... 86
5.10 Col Legno Battuto .................................................... 87
5.11 Bowing Surfaces .................................................... 87

6. Epilogue ....................................................................... 93
6.1 Design Observations .................................................. 93
6.2 Ongoing Development ............................................... 94
6.3 User Acceptance ....................................................... 96
6.4 Personal Reflections ................................................... 97

References ...................................................................... 99

Appendix A Compositional Studies for Solo ESBow .................. 111
A.1 Traditional Expectations and the ESBow ......................... 112
A.2 JunoD Improvisations in D minor .............................. 118
A.3 Sound Source Series .............................................. 120
A.4 Without a String to Stand On .................................... 123
A.5 Four Rows of Twelve .............................................. 126
A.6 Violin 2.1 ............................................................ 131
A.7 Kitchen .................................................................. 137
A.8 Composing with the ESBow ....................................... 141
Appendix B  Miscellaneous Diagrams and Listings ................................................................. 143

B.1  Arduino Code .................................................................................................................... 143
B.2  ESBow Daughter Board ................................................................................................. 144
B.3  Pure Data Patches ........................................................................................................... 146
B.3.1  Pure Data to MIDI Interface ....................................................................................... 146
B.3.2  Pure Data Input to MIDI Sub-Patches ....................................................................... 150
B.3.3  Arduino2PD and SimpleMessageSystem ................................................................. 151
B.4  Pure Data Examples ....................................................................................................... 153
B.4.1  Relative Position Sensing .......................................................................................... 153
B.4.2  Displacement of an Axis ............................................................................................ 155
B.4.3  Displacement to actuate two streams ........................................................................ 156
B.4.4  Expression Object ....................................................................................................... 157
B.4.5  Velocity of an Axis ..................................................................................................... 158
B.4.6  Trackball Counter and Looped Sequence .................................................................. 159
B.4.7  Demonstration Video 12 ........................................................................................... 160
B.4.8  Monitoring Trackball Select Duration ........................................................................ 161

Appendix C  The Evolving ESBow ............................................................................................ 163

C.1  Arduino Code .................................................................................................................... 163
C.2  The ESBow Daughter Board .......................................................................................... 165
C.3  Pure Data ....................................................................................................................... 166
C.4  Force Sensing Resistors ................................................................................................ 169
C.5  Accelerometer ................................................................................................................ 170
C.6  Trackball ......................................................................................................................... 172
List of Figures

Figure 1: The ESBow. ........................................................................................................... vii
Figure 2: Mapping strategies. ............................................................................................. 12
Figure 3: ESBow Flow Control. .......................................................................................... 39
Figure 4: The Arduino Diecimila. ....................................................................................... 42
Figure 5: The ESBow daughter board ............................................................................... 43
Figure 6: The Arduino secured in housing box. ................................................................. 44
Figure 7: The Arduino mounted on arm ............................................................................ 45
Figure 8: Flow of gestural data within the default PD to MIDI interface. ....................... 46
Figure 9: The default PD to MIDI interface. .................................................................... 47
Figure 10: Simple binary event detector .......................................................................... 49
Figure 11: Data sampling. ................................................................................................ 50
Figure 12: The FlexiForce force sensing resistor. ........................................................... 52
Figure 13: The FSR foam mount ...................................................................................... 53
Figure 14: Relative position sensing: a composite signal produced by mapping two FSRs. .......... 54
Figure 15: The responsive patterns of the FSRs when bowed slowly along the entire length of the bow with equal pressure. ............................................................. 56
Figure 16: Recalibrating the FSRs ................................................................................... 56
Figure 17: Freescale MMA7260 tri-axial accelerometer. ................................................ 60
Figure 18: Dynamic movement in the Y axis of the ESBow. ............................................ 61
Figure 19: Combined dynamic and static acceleration in the X axis above with rapid dynamic acceleration in the Y axis below. This was achieved by slowly rocking the ESBow on its X axis while performing a tremolo action .................................................................................. 62
Figure 20: Localising the output of the Y axis .................................................................. 62
Figure 21: Monitoring the displacement of an axis .......................................................... 65
Figure 22: Splitting a data stream using an expression object. ................................................................. 66
Figure 23: Acceleration and velocity in the Y axis of the accelerometer .................................................. 67
Figure 24: Determining the velocity of an axis. ......................................................................................... 67
Figure 25: The “BlackBerry” trackball. ..................................................................................................... 68
Figure 26: Ramping the trackball select. ................................................................................................. 70
Figure 27: Strobing an analog data stream with the trackball select. ....................................................... 71
Figure 28: Trackball loop counter. ......................................................................................................... 72
Figure 29: Activating a short looped sequence. ...................................................................................... 72
Figure 30: Monitoring the duration the trackball is selected. ................................................................. 74
Figure 31: The mounted sensors. ........................................................................................................... 75
Figure 32: The sensors of the ESBow detached. ..................................................................................... 77
Figure 33: The ESBow reconfigured with a single FSR. ........................................................................... 78
Figure 34: A.1 User interface. ................................................................................................................ 113
Figure 35: A.1 Bassline. .......................................................................................................................... 113
Figure 36: A.1 Dynamics. ......................................................................................................................... 114
Figure 37: A.1 Timbre 1. .......................................................................................................................... 114
Figure 38: A.1 Timbre 2 ........................................................................................................................... 115
Figure 39: A.1 Reversing the output streams .......................................................................................... 116
Figure 40: A.2 JunoD MIDI control table. ............................................................................................. 119
Figure 41: A.3 User interface. ................................................................................................................ 121
Figure 42: A.3 Overdrive. ........................................................................................................................ 121
Figure 43: A.4 Selecting a scale with the trackball. ................................................................................ 124
Figure 44: A.4 Harmonising and selecting output. .................................................................................. 124
Figure 45: A.5 User interface. ................................................................................................................ 126
Figure 46: A.5 The tone row operative. .................................................................................................. 127
Figure 47: A.5 Select as a structural device. ........................................................................................... 128
Figure 48: A.5 Balance and panning. ....................................................................................................... 129
Figure 49: A.6 Matrixed samples. ........................................................................................................... 131
Figure 50: A.6 Matrix object

Figure 51: A.6 Trackball select as a structural device

Figure 52: A.6 Solo instrument

Figure 53: A.6 Dynamics

Figure 54: A.6 Transposition range of the Y axis and dual effects of the X axis

Figure 55: A.6 Ranged panning

Figure 56: A.6 Accompaniment

Figure 57: A.6 Clocking the chord progression

Figure 58: A.6 Metasurface

Figure 59: A.7 User interface

Figure 60: A.7 Trackball axis control

Figure 61: A.7 Randomising MIDI channel numbers using an urn object

Figure 62: A.7 Timing the trackball select

Figure 63: Daughter board schematic

Figure 64: Single canvas PD to MIDI interface

Figure 65: Default sensor MIDI control numbers

Figure 66: PD Input Sub-patch

Figure 67: PD MIDI_Output Sub-patch

Figure 68: Arduino2PD

Figure 69: Relative position sensing

Figure 70: Monitoring the displacement of an axis

Figure 71: Monitoring the displacement of an axis to actuate two control streams

Figure 72: Monitoring displacement with an expression object

Figure 73: Splitting a data stream using an expression object

Figure 74: Determining the velocity of an axis

Figure 75: Trackball counter and sequence

Figure 76: Chord sequence

Figure 77: Monitoring the duration the trackball is selected
Figure 78: Original daughter board schematic................................................................. 165
Figure 79: The original PD to MIDI interface................................................................. 167
Figure 80: The original accelerometer breakout and intercept boards......................... 171
Figure 81: Comparing the two Freescale accelerometer breakout boards...................... 172
Figure 82: The original cannon miniature trackball....................................................... 173
### Terms and Abbreviations

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerometer</td>
<td>Electronic sensor that monitors tilt and acceleration in one to three axes.</td>
</tr>
<tr>
<td>Arduino</td>
<td>Arduino Microcontroller.</td>
</tr>
<tr>
<td>AudioMulch</td>
<td>AudioMulch Interactive Music Studio Software.</td>
</tr>
<tr>
<td>Augmented Violin</td>
<td>A natural violin fitted with electronic sensors.</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>A form of wireless electronic communication.</td>
</tr>
<tr>
<td>Bowing Surface</td>
<td>Any object with a protruding edge that can be bowed.</td>
</tr>
<tr>
<td>Breakout Board</td>
<td>Circuit board used to house a specific sensor and allow it to interface with other electronic devices.</td>
</tr>
<tr>
<td>Composed Instrument</td>
<td>The instrument and its configuration as a compositional framework for improvisation.</td>
</tr>
<tr>
<td>Daughter Board</td>
<td>A circuit board designed to be mounted onto the back of another board.</td>
</tr>
<tr>
<td>Downstream</td>
<td>All post-processing elements of audio and data signals controlled in real-time.</td>
</tr>
<tr>
<td>Dynamic Acceleration</td>
<td>Accelerometer output due to physical movement along an axis.</td>
</tr>
<tr>
<td>Electronic Chamber Music</td>
<td>The restoration of intimacy normally associated with traditional chamber music in solo and ensemble performances of electronic music.</td>
</tr>
<tr>
<td>ESBow</td>
<td>Electronic Sensor Bow.</td>
</tr>
<tr>
<td>FSR</td>
<td>(Force Sensing Resistor) Sensor that monitors the physical force placed on a small contact area.</td>
</tr>
</tbody>
</table>
Haptic Feedback
Tactile feedback achieved through physical action that provides crucial information to the performer about various aspects of their performance.

Juno D
Roland Juno D synthesiser used in selected audio/video examples.

Mapping
The process of connecting an electronic performance interface to an electronic sound generator.

Microcontroller
Small computer on a single Integrated Circuit.

MIDI
Musical Instrument Digital Interface.

Natural Violin
A violin constructed from traditional materials and building technology that does not rely on electronic amplification.

PD
Pure Data Real-time Audio Software.

Prepared Violin
A natural violin 'prepared' for a composition by modifying its hardware interface, such as by inserting foreign objects in between strings.

Reduced Violin
A representational violin that focuses on a single aspect of the natural violin in detail.

Representational Violin
A violin built on core elements of natural violin performance technique that is liberated from traditional design constraints.

Schizophonia
Sound that is separated from its source via electronic reproduction.

Static Acceleration
Accelerometer output due to tilt in an axis.

Trackball
Sensor that can be clicked and rolled in place to provide data along two axes.

Upstream
All unprocessed audio and data and everything that produces them.

USB 2.0
Universal Serial Bus Revision 2.0 Specification.

Virtual Violin
A violin which has no tangible physical properties and is created using algorithms and/or computer software and/or hardware.
DVD-ROM - Video Demonstrations

Video 01: Actuating pitch and duration with an FSR and the Y axis of the accelerometer.

Video 02: Relative position sensing using the FSRs.

Video 03: Panning using two separate outputs from the FSRs.

Video 04: Combining the outputs of the FSRs.

Video 05: Allocating different processes to each FSR.

Video 06: Tilting the accelerometer.

Video 07: Dividing the output of an axis into separate bands.

Video 08: Using the Z axis for orientation.

Video 09: Navigating a metasurface with the trackball.

Video 10: Ramping the select button of the trackball.

Video 11: Extended metasurface techniques.

Video 12: Using the trackball to progress through a series of interval harmonies.

Video 13: Using the four directions of the trackball to control four pre-defined progressions.

Video 14: Using the four directions of the trackball as toggles.

Video 15: Monitoring the duration that the trackball is selected.
**DVD-ROM - Audio Recordings**

Audio 01: Traditional Expectations and the ESBow (simple).

Audio 02: Traditional Expectations and the ESBow (extended).

Audio 03: Opposing Traditional Expectations and the ESBow (simple).

Audio 04: Opposing Traditional Expectations and the ESBow (extended).

Audio 05: JunoD Improvisations in D minor.

Audio 06: JunoD Improvisations in D minor.

Audio 07: Sound Source Series (sine wave oscillator).

Audio 08: Sound Source Series (white noise generator).

Audio 09: Sound Source Series (electric violin).

Audio 10: Sound Source Series (JunoD synthesiser).

Audio 11: Without a String to Stand On.

Audio 12: Four Rows of Twelve.

Audio 13: Violin 2.1.

Audio 14: Kitchen.
Prologue

This thesis is focused on the performance of electronic music with an intuitive electronic violin controller. The focus stems from my interest in the performance of music with electronic hardware custom built for composition. These interfaces allow for greater depths of expression and nuance than the standard typewriter keyboard and computer mouse associated with modern electronic performance.

The compositional examples in this thesis are based in the software applications Pure Data (version 0.41.4-extended) and AudioMulch (version 2.1.1). These programs use graphical representations of the flow of information from the outlets of an object above to the inlets of another object below.

The first chapter of this thesis introduces the reader to electronic violin controllers with specific mention of the Electronic Sensor Bow, or ESBow. It outlines the aims of the ESBow project and provides a brief summation of the following chapters of the thesis.

The second chapter defines the important terms found in this thesis. These include the details of types of violins and violin-like controllers and a brief look at what they entail. Following this is a discussion of significant issues relating to electronic violin controllers in general. This includes the electronic evolution of the violin, haptic feedback, and the audience’s perception of new controllers. The chapter closes with a discussion of electronic chamber music.
The third chapter introduces the first iteration of the ESBow and reviews its development stages. This includes the approach and initial idea behind the ESBow, the impact of other controllers on my project, and the first prototype design which laid the foundation for creative projects.

The fourth chapter outlines the ESBow’s current design. Electronic specifications are included in this chapter. This chapter also discusses how the sensor data is interpreted by the computer workstation and the application and functional design of each sensor.

The fifth chapter focuses on conventional violin bowing techniques and how these techniques can be translated into new forms of electronic control. A discussion on bowing surfaces and their role with the ESBow is also included in this chapter.

The final chapter summarises the outcomes of the ESBow project and how these reflect on the original intentions of the project. This chapter also considers the future of the ESBow and discusses possible modifications to the hardware interface.

Appendix A includes a collection of compositional studies for the ESBow. These studies are part of the methodology for demonstrating the proof of concept design with a focus on key areas discussed in the thesis. The exercises explore the possibilities the interface design presents to the performer and form the springboard for larger compositional works in the future.

Appendix B consists of detailed illustrations and descriptions of techniques discussed in the fourth chapter of the thesis.
Appendix C discusses the legacy of the current design and the stages of evolution that lead to it.

A companion DVD-ROM accompanies this thesis. It contains video examples of bowing and mapping techniques that have been developed for the ESBow and audio recordings of the compositional studies included in Appendix A. The reader will be referred to the relevant video on the DVD-ROM as each technique is discussed in chapter four of the thesis. The video and audio files are located in the Multimedia folder on the disc and can be accessed via the Multimedia html file. An electronic version of this thesis is also included on the DVD-ROM.
1. The Evolving Violin

Far from being static, the violin has undergone a process of continual evolution and advancement. It has seen the sixteenth century Amati developed into the modern orchestral violin and the traditional pernambuco wood to the modern carbon fibre and fibreglass bow. Even the first four string violin can be traced further back to other bowed string instruments popular at the time of its design. While the specific instruments that most impacted on its design are unknown a number of contemporary instruments have been suggested. The rebec and lira da braccio are two of the most frequently suggested instruments with each bearing significant resemblance to these early violins (Montagu n.d.). These instruments and their contemporaries are yet another evolutionary step in a long line of bowed string instruments. This evolution leads all the way back to the early equestrian cultures of Central Asia such as the Turkic and Mongolian cultures and their two string up-right fiddles featuring horse hair strings and horse hair bows. This plethora of chordophones from cultures around the world all have one thing in common, their sound is actuated by bowing.

In recent years this evolution has seen the inclusion of electronic technology in the design interface of new violins and violin-like controllers. These interfaces seek to extend the capabilities of the violin through electronic enhancement of traditional techniques as well as new techniques only made possible by new interface design. The interfaces include electronic extensions added to traditional violins and bows, new electronic instruments based on aspects of traditional violin design, and electronic controllers that simulate real performance gestures on virtual violins. Electronic sensors are used in each case to monitor various aspects of
traditional violin performance technique. This may include changes in the pressure, position and acceleration of the bow in three dimensions, along with the downward, lateral, torsional and frictional strains on the bow. Other aspects monitored could include the position of the fingers of the left hand, the angle of the right hand wrist, and the amount of bow hair in contact with the strings. The gestural information captured is used in a variety of ways. These often include driving processors and effects or determining the parameters of a synthesised physical model of a violin using conventional performance gestures. The information can even be used to control non-audio material such as a visual display. All of these can be performed in real-time. These techniques are employed in order to provide both the expert and the novice performer with the best possible interface for the natural intuitive control of every aspect of all acoustic and electronic processing during performance. These interfaces are not intended to supplant the traditional violin, but rather to allow its continuing evolution through the creation of new instruments and new techniques.

The driving force behind this thesis and the main component of my creative portfolio has been the development of a new interface for composition. The Electronic Sensor Bow, or ESBow, is a violin controller designed to augment the capabilities presented to a performer through the addition of sensors to the interface of the violin. The ESBow monitors the pressure placed on the hair of the bow in two places, as well as the acceleration and tilt of the bow in all three directions. It also features a trackball with a momentary switch that can be used to control a point in two-dimensional space such as one would with a joystick. As the sensors of the ESBow are mounted exclusively on the bow of the violin it can be used to perform with a violin or be taken away from the violin to be used with any stringed instrument or non-instrumental surface. This is one of the key elements of the ESBow project and provides the performer with a blank canvas upon which to plan a performance.
This thesis will explore the background of electronic violin controllers with detailed attention given to selected areas of significance. Discussion will then proceed to the development of the ESBow from conception to the current prototype build. The efficiency and capabilities of the prototype will be examined with regard to each sensor and the design as a whole. Detail will be given to key areas and techniques that best demonstrate the possibilities that the ESBow creates. Finally, the future of the ESBow will be discussed from a view of what the ESBow can currently achieve and what it can potentially achieve.

This thesis focuses on the use of the ESBow as a solo instrument. Focusing on the ESBow as a solo instrument will allow me to look at the instrument itself in greater depth than would be possible if I were to attempt to cover it in ensemble. Greater emphasis will also be placed on the performance of the ESBow without a violin as this is where my personal interest lies. However, the ESBow will still be performed with a violin and it will be reviewed with this in mind to ensure each possibility is given significant attention and detail.
2. Electronic Violin Controllers

This chapter is focused on key issues relating to electronic violin controllers in general. It opens with a list of terms that assist in the understanding of various types of violins and violin-like controllers. It then proceeds to discuss issues regarding electronic violin controllers and conventional MIDI controllers and how these issues apply to the ESBow. These issues include extending the legacy of an existing instrument, haptic feedback and electronic chamber music.

2.1 Terms of Reference

Hugh Livingston proposes the terms upstream and downstream in his article Paradigms for the new string instrument: digital and materials technology (2000). The term upstream is used to describe all unprocessed audio and data and everything that produces them. This includes the hardware materials of the violin, its sensors and any microphones used to record audio (Livingston 2000). The term downstream refers to all post-processing elements controlled in real-time. This includes all software and hardware for digital signal processing, such as pitch-trackers and feature recognition programs or patches, along with any modification and/or production to the input stream/s (Livingston 2000). The downstream environment is extremely flexible and can easily be tailored to specific compositions. The upstream environment can be tailored to a specific composition through scordatura or by preparing the instrument in the Cagean sense of prepared piano, though this is not as common.
Livingston also distinguishes between various types of violins and violin-like controllers. He separates these interfaces into two main groups: *natural violins* and *representational violins*.

A *natural violin* is a violin constructed from traditional materials and building technology that does not rely on electronic amplification (Livingston 2000). These are the violins used in the string section of any traditional orchestra.

A *representational violin* is a violin built on core elements of natural violin performance technique and is liberated from traditional design constraints (Livingston 2000). Representational violins will usually rely on amplification and offer new levels of flexibility and control to the performer. These violins do not typically feature any of the raw audio associated with the natural violin but simply use its performance technique as a source of control information for electronic processing. An exception to this is the Overtone Violin developed by Dan Overholt which features traditional violin strings (Overholt 2005). Instruments that do not produce raw audio in the upstream environment originate audio in the downstream environment. The user can process this audio in the downstream environment in any way before it is amplified. While a natural violin can be recorded and processed in a live performance, an acoustic violin sound will always be present.

Livingston further divides representational violins into two sub-sections, *feature-rich* and *feature-isolated* representational violins. The feature-rich representational violin combines several physical and functional properties of the natural violin (Livingston 2000), such as right hand bowing technique, left hand fingering technique and the violin’s acoustic properties. An example of a feature-rich representational violin is the Hypercello which is associated with the music of Tod Machover and was constructed by engineers at MIT.
(Paradiso & Gershenfeld 1997). The feature-isolated representational violin is focused on a single aspect of the natural violin in detail (Livingston 2000). Feature-isolated representational violins are often designed for the study of an aspect of violin performance technique. An example of this is the vBow, which focuses on right hand bowing technique (Nichols 2003). Examples of feature-isolated representational violins whose designs were motivated for creative purposes are Dan Trueman’s Fangerbored and Bonge. The Fangerbored is focused on left hand fingering technique and the Bonge contains four bowed sponges based on the four bowed strings of the violin (Trueman 1999). Livingston refers to feature-isolated representational violins as virtual violins (2000). This term, however, could be misleading because the term virtual is typically used in relation to a lack of tangible qualities, such as Virtual Reality. The term reduced violin is better suited to describe the focus on a single aspect of a natural instrument.

It is more appropriate to use the term virtual violin to describe instruments which have no tangible physical properties. Virtual violins are created using algorithms and/or computer software and/or hardware. An example of a virtual violin is the General MIDI violin found in most media players. Virtual violins exist purely in the downstream environment with no upstream portion. As virtual violins have no natural physical interface they need to be pre-determined or directed by an outside source. While electronic violin controllers may be used to drive virtual violins, they are in fact separate entities. This makes it possible to differentiate between the virtual experience and the physical interface used to control it. The model example of this is the synthesised violin replicated through the action of a keyboard. The same key can be pressed to hear any sound, such as a piano, tuba or even a car horn. The physical keyboard directs the audio but is not inextricably linked to the audio engine. A keyboard actuated virtual violin can also be played using a pre-composed MIDI score via an
automated performance. This completely separates the audio from the interface that drives it.

The term *augmented violin* will be used to refer to the natural violin fitted with electronic sensors. An example of this is the so called Augmented Violin developed at IRCAM (Bevilacqua et al. 2006). The augmented violin can be thought of either as both a natural and a representational violin simultaneously, or as a natural violin that has been converted into a representational violin through the addition of gestural sensors. Augmentations are specifically designed to be mounted and removed from the violin without any permanent modification to the instrument. An example of this is the detachable device called the Reflective Optical Pickup constructed at IRCAM (Leroy, Flèty & Bevilacqua 2006). These augmentations can often be accomplished as easily as placing a mute on a violin bridge and provide the option of working with a natural or augmented instrument at will.

The ESBow is a traditional violin bow featuring electronic gestural sensors to monitor performance data. The sensors are mounted in a non-permanent fashion and can be easily removed from the bow without risk of damage. As such the ESBow is an augmented instrument. As it focuses on right hand gestures and performance technique with no left hand component, the ESBow is also a reduced instrument. However, when the ESBow is used with a natural violin it can also be considered a single augmented violin consisting of two halves; the natural violin and the representational bow. This demonstrates that the above terms should be used as a general guide to understanding the nature of various electronic violin controllers and are not rigid classifications. While information from the ESBow could be used to drive a virtual violin, it was not designed with this intention. This has been the focus of other instrument builders whose designs would provide a more realistic traditional performance on a virtual violin. One of the most suitable electronic violin controllers for
driving a virtual violin is Charles Nichols’ vBow (Nichols 2003). The ESBow was designed with the motivation of modifying the upstream environment for different works by choosing to bow various surfaces suited to the composition. This is discussed further in chapter 5.11.

Focusing on right hand bowing technique involves the exclusion of pitch control data from the interface. Pickups or other electronic sensors on the natural violin can be used to obtain more data for processing, though these will often lack the audio quality associated with the natural violin. The Reflective Optical Pickup developed at IRCAM for example, was deemed appropriate to determine the pitch of a violin. However, it produces a weak sound without the full harmonic tone of the natural violin (Leroy, Flèty & Bevilacqua 2006).

### 2.2 Extending the Musical Legacy of an Existing Instrument

The principal motivation in designing an interface based on the action of bowing is twofold. Not only does this allow the design to benefit from the musician’s years of personal training but also the accumulated body of traditional performance techniques developed over many centuries. This is practical and advantageous not only for the performer who plays this interface but also for the composer who wishes to write for it. An instrument that challenges both composers and performers is more likely to develop a repertoire that is played by numerous performers.

New electronic interfaces that are not based on pre-existing instruments may initially generate interest but are likely to fall into disuse as performers move on to the next new interface or revert to the traditional instrument. Modifying an existing performance interface
to accommodate traditional techniques as well as add new techniques enhances its appeal to new performers and composers and sustains their continued interest in the instrument. New techniques may be based on new gestures made possible by the addition of electronic sensors or on electronically monitored conventional techniques that are already mastered.

Many performers that use various real-time electronic signal processing and modification equipment obtain their original sound source from a traditional violin (Murphy 2007). This demonstrates the attraction of the violin and its traditional and innovative techniques in electronic environments. However, incorporating externally controlled processors into a performance introduces additional challenges associated with the hardware. For example, the performer can experience difficulty trying to control a number of effects processors while playing an instrument. In these situations a certain degree of control is usually sacrificed in order to improve the playability of the interface. An example of this is the use of an effect or process with a preset intensity that can be toggled on or off, such as a fixed distortion pedal, rather than control that is fluid and continuous, such as bow pressure or acceleration. However, even foot pedals can be challenging for a performer. This is especially true of cellists who must not only have their feet firmly planted in order to support their instrument with their legs, but also have their view of the pedals blocked by their own instrument.

By integrating controls for these and/or similar processors into the interface of the violin, the performer can manipulate parameters with right and left hand techniques, both traditional and new. This allows the violinist to control raw and processed sound simultaneously. Traditional MIDI controllers devised entirely of buttons and controls generally lack this ability for greater control. While precise slider and turnpot changes can be performed individually, only a limited number of controls can be changed simultaneously using two hands.
2.3 Mapping and the Composed Instrument

The performance interface of a natural violin is intrinsically linked with the generation of the violin’s sound. The interface of an electronic instrument, however, does not directly create sound and must be connected to a sound generator for any sound to be heard. The process of connecting an electronic performance interface to an electronic sound generator is known as mapping. Mapping establishes the link between the upstream and downstream components of an instrument and is just as significant as the interface that is played and the sound generator itself. Not only do these rely on mapping to connect, but the way this connection is defined can completely change the character of the instrument.

There are a number of different types of mapping strategies available. These include:

- One-to-one – one signal to produce one result.
- One-to-many – one signal to produce two or more results.
- Many-to-one – two or more signals used in combination to produce a single result.
- Many-to-many – two or more signals used in combination to produce two or more results.

These strategies are illustrated in Figure 2.
These techniques allow sound to be controlled in a variety of ways. Mounting additional sensors onto the interface can enhance the expressive potential of a new instrument. However, intelligent mapping is required to realise this potential.

Andy Hunt, Marcelo M. Wanderley and Matthew Paradis have conducted a number of experiments exploring different types of mapping strategies. One of these tests consisted of presenting performers with different mappings on identical interfaces with identical sound sources. From the results of these studies they concluded that while interfaces with simple mapping strategies were favoured initially, subjects ultimately favoured more complex mappings which were considered to be more expressive and more like a traditional instrument (Hunt, Wanderley & Paradis 2002).

<table>
<thead>
<tr>
<th>One-to-One</th>
<th>Many-to-One</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Example:</td>
<td>Example:</td>
</tr>
<tr>
<td>Tilt in an axis</td>
<td>Bow pressure and tilt</td>
</tr>
<tr>
<td>to determine the pitch of an oscillator.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>One-to-Many</th>
<th>Many-to-Many</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3.png" alt="Diagram" /></td>
<td><img src="image4.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Example:</td>
<td>Example:</td>
</tr>
<tr>
<td>Tilt in an axis</td>
<td>Bow pressure and tilt</td>
</tr>
<tr>
<td>to determine both the dynamics and timbre of an oscillator.</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 2: Mapping strategies.*
Hunt et al identified the problem that one-to-one mapping strategies in electronic controllers are often unrealistic (2002). The natural violin features complex mapping systems embedded within its interface. Violin dynamics for instance are controlled by a combination of the pressure placed on the hair of the bow, the acceleration of the bow along the Y axis, the tilt of the bow in the X axis and the subsequent amount of hair in contact with the string. Controlling the dynamics of audio produced by a representational violin using a single parameter such as bow pressure or acceleration alone lacks the expressive potential of a real violin.

Their findings were substantiated by additional tests that compared one-to-one mapping strategies to complex mapping strategies based on wind instruments. Subjects who were experienced musicians favoured complex configurations while beginners initially preferred simpler configurations that were easier to play, despite the lack of expressivity (Hunt, Wanderley & Paradis 2002). Simple configurations could therefore be used as a pedagogical tool that allows a novice to graduate to more complex and expressive mapping configurations.

Streams of data output from several sensors do not have to be combined in natural mapping combinations. It is also possible to create other combinations in which sensors interact with one another in ways that have no precedent. For example, it is possible to use the data from one sensor to modify the output of another sensor or to apply various mathematical formulae to combinations of sensor streams to gain the average, sums or multiplications of streams. A simple example of this is discussed in chapter 4.5.1 with the output of the two force sensing resistors combined to gain a linear positioning sensor relative to the position of the two force sensing resistors along the length of the bow. One would need to experiment with various
formulae and mapping combinations in order to see how successful each approach is in a performance.

From a compositional perspective the mapping of electronic instruments provides the composer with seemingly limitless possibilities. Not only may different parameters, such as bow pressure or acceleration, be selected for enhancement on the same controller for different compositions, but the mappings of each parameter can also be different for every composition. These changes can be made to suit the piece, the intent of the composer, or to suit the playing style of the performer. The composer does not have to follow traditional mappings for the violin and has the freedom to modify the behaviour of the violin simply by mapping gestures to different control parameters.

The term composed instrument has been applied to new representational instruments to include the interface, sound generator and mapping for a specific composition (Schnell & Battier 2002). In other words the instrument and its configuration can be a compositional framework in which the performer improvises. The compositions in Appendix A are a demonstration of this compositional framework.

2.4 Haptic Feedback

One of the most important aspects of the ESBow project is the tactile connection between performer and instrument. This connection is referred to as haptic feedback. Haptic is derived from the Greek word haptesta – to touch – (Serafin et al. 2001). The term is applied to physical action such as playing the violin where feedback is the result of the friction
produced by running the bow across a string. This feedback provides crucial information to
the performer about various aspects of their performance. For example, the performer can
feel the pitch through the placement of their left hand fingers on the neck of a violin and feel
the vibrato through the minute movements of each finger. The right hand will also sense the
dynamics and tone quality of a violin from the pressure applied to the string via the bow and
the movement and drag of the bow across the string. This feedback is continuous and gives
the violinist an instantaneous understanding of the response of the instrument allowing them
to achieve intuitive control over the sound.

Haptic feedback is just as important for an instrumentalist as auditory feedback. A bowing
technique is often described in terms of its physical action rather than the sound that is
produced. Spiccato – typically depicted using the terms “bounce” or “spring” – is a clear
example of this as it describes the physical action of the bow and how it feels to the
performer (Nichols 2003). A trained violinist will know instinctively how techniques such as
spiccato will sound purely from the feel of the gesture. Performance techniques are developed
by making minor adjustments based on this physical connection.

Instruments that have no haptic feedback, such as the theremin, can be extremely hard to play
accurately as they do not provide physical reference points to the performer. Even though
there are adept and skilful theremin performers, an inexperienced theremin performer is often
confounded by the lack of physical feedback in a situation where there is no tangible
performance interface. This was demonstrated by Sile O’Modhrain by observing the
beneficial effects of coupling a performer’s hand to a theremin antenna with a simple elastic
band (O’Modhrain 2000).
Haptic feedback plays such a significant role in performance that the quality of an instrument is often judged as much by the physical feedback it delivers rather than its sound alone. For this reason the quality of the violin is better assessed by the performer rather than the listener. While the sound produced by a violin is clearly important, a good violinist can compensate for the tone produced by a poor instrument. The effort needed to produce a good tone on an inferior violin cannot be compensated to the same degree. The ease of producing a good tone on an instrument is referred to as the playability of the instrument. Where a performer has to make less compensation for the quality of the instrument the instrument can be said to be more playable. A number of the studies involving new violin interfaces specifically aim to enhance the playability of both natural and representational violins (Serafin, Smith & Woodhouse 1999). This includes the playability of both current and future instruments.

2.5 Simulated Haptic Feedback

Some representational violins offer very little natural physical feedback. Designers of such instruments will sometimes use the addition of hardware that is able to generate customisable haptic feedback to provide a more stimulating or realistic performance interface. The simulation of haptic feedback features the downstream environment affecting the upstream interface. The vBow, designed by Charles Nichols, is one example of a representational violin that simulates haptic feedback through the use of servomotors (Nichols 2003). However, this method is not restricted solely to violin controllers. The Haptic Carillon Clavier developed at the University of Wollongong is another representational instrument that simulates haptic feedback associated with the actuation of carillon bells (Havryliv, Geiger et al. 2009).
Numerous researchers have explored the beneficial effects of generating haptic feedback to provide physical reference points to the player. This has been conducted with numerous types of instruments and electronic devices. One such device is the Moose, which was developed at CCRMA (Center for Computer Research in Music and Acoustics) by Richard Brent Gillespie and Sile O’Modhrain (Gillespie & O’Modhrain 1995). The Moose is a computer mouse that simulates haptic feedback for the control of musical software. Sile O’Modhrain and Chris Chafe performed a series of tests with the Moose focused on haptic feedback. One test involved adding various types of haptic feedback to the Moose while using it like a virtual theremin and recording the accuracy of the pitches in the melodies performed. The resulting data demonstrated that simulated haptic feedback would improve the playability and accuracy of the performance by an average of twenty three percent (O’Modhrain 2000). Results indicate that instrumentalists typically prefer performing with devices that provide haptic feedback as they offer a greater sense of precision and control.

Devices that generate force feedback are commonly found in computer games. Vibrating game controllers such as rumble paks and dual shock controllers have become quite popular in recent years. The player feels vibrations transmitted through a handheld controller to coincide with events that occur in the game. For example, the Super Nintendo game Zelda, Ocarina of Time will transmit vibrations through rumble pak controllers during a volcanic eruption (Miyamoto 1998). Such coarse tactile cues are intended to reinforce the narrative and produce a more immersive and satisfying game experience. Whereas interfaces designed for expressive musical control are intended to improve the actuation of performance technique. As a result tactile cues used in games are inherently simple while tactile cues for expressive musical control tend to be complex and work on much smaller time scales. Virtual and Augmented Realities could be seen as the definitive outcome for simulated haptic
experiences. This includes virtual violins performed with a physical electronic controller that provides haptic cues to the performer.

Haptic feedback can assist a performer to control a virtual instrument. It can also be used to alter the feel of an electronic musical instrument by modifying existing or even creating entirely new haptic cues. This too can be found in videogames. In the aforementioned Zelda, Ocarina of Time a magic item known as the Stone of Agony causes the controller to rumble without physical provocation when the player character is near hidden items (Miyamoto 1998). This is an example of a new haptic cue suggestive of proximity sensing. A musical example suggested by Charles Nichols is the possibility of multiple layers of virtual strings for the vBow. These would be accessed when the performer pushed the bow hard enough for the initial set of strings to give way to another set beneath. Each set of strings would simulate a different material and winding, such as nylon, steel round-wound or silver flat-wound strings with each layer providing modified haptic feedback to fit their specific material and winding (Nichols 2003). His idea could be extended to simulate stringed instruments from non-western backgrounds, such as the sitar and the erhu. It is also possible to apply this to any object with an edge, including objects that have no traditional foundation in music, such as wire fences which were bowed during the Great Fences of Australia project by Jon Rose and Hollis Taylor (Rose n.d.). This could also include virtual surfaces with no equivalent in the real world, which are designed by a composer for a specific composition.

The haptic feedback of an electronic violin controller can be modified for numerous works. The feedback can also be changed during a piece or even during a single note. As Charles Nichols stated in closing his dissertation, the only limit is the “range of the motion of the performer, and the imagination of the composer” (Nichols 2003). What Nichols says of the
vBow is true of all gestural controllers that feature any form of simulated haptic feedback.

### 2.6 Haptic Feedback in Conventional Electronic Interfaces

Electronic music is often performed with various control interfaces that feature potentiometers (rotary or slider) and buttons (momentary or toggle). These devices offer minimal physical feedback to the perform. One may feel how far a rotary potentiometer has been turned and to what extent a slider has been raised or lowered yet from feel alone have little insight into the impact this will have on the music. Furthermore, it is not possible to ascertain continuous haptic feedback from each input in a conventional control interface. Tactile feedback is only received from the physical input currently held and altered. A violinist is able to maintain intuitive control in a way that someone operating a conventional electronic control interface cannot. These control interfaces rely more on visual feedback than tactile feedback. For a performer the sense of touch can be as significant as the sense of hearing and cutting out haptics is effectively a form of sensory deprivation.

A conventional electronic control interface could have a ‘vibrato’ potentiometer; however, the performer won’t be able to feel vibrato the way a violinist does. They will feel the rotation or shift in the potentiometer but are much more likely to rely on feedback from visual and auditory sources. These methods are often not as effective as tactile feedback. While a great deal of information can be conveyed visually, depending on a visual source can create a division between a performer and the music. The performer may observe that more vibrato is required on slider B and increase the vibrato to a precise setting; however, it would be difficult to perform this action intuitively. A violinist however, can use haptic feedback to
simultaneously detect and rectify problems with technique. A reliance on vision also restricts activities such as reading a score and communicating with other performers. Prudent use of visual feedback however can be of distinct benefit.

The position of the frets on a guitar illustrates the value of haptic feedback over visual feedback. Along the neck of a guitar dots mark the position of certain frets. As a guitar teacher, I have observed beginners tending to focus on these visual markers and watching the placement of every finger before they pluck a string. However, in time the guitarist no longer relies on markers and instead relies on tactile cues to play the guitar as performance becomes more intuitive.

The execution of simultaneous control changes across multiple parameters on an electronic control interface can be extremely difficult to accomplish without triggering pre-programmed sequences of data. A violinist, however, can alter their performance in any number of ways simultaneously. It is also difficult to intuitively update and modify parameters executed on consecutive notes in a phrase or melody performed on a conventional electronic interface at a fast tempo. This is especially true when performing the task while updating other parameters simultaneously without any form of pre-definition. A violinist can consistently and simultaneously update parameters such as tremolo, dynamics and tone colour on consecutive notes.

So why would one use a conventional electronic control interface? Control interfaces featuring potentiometers and buttons are often chosen for their accuracy and convenience. If the note A below middle C is required then a tone with the exact frequency of 440 hertz is produced. However, precise tuning can sometimes deliver a clinical sound without the natural
imperfections found in acoustic music. This also works against traditional violin teaching that features non equal tempered intervals such as perfect fifths. Alternatively, electronic control interfaces can assist classically trained performers to play music that is in some form of Just Intonation tuning.

Electronic control interfaces are versatile and can be customised. In one performance a potentiometer can control the pitch of an oscillator. In the next performance the same potentiometer can control the saturation of a granulator effect. Electronic control interfaces can be adapted to suit a performer’s ability, preferences and idiosyncratic performance style. Performers can also adapt their instrument to a specific piece rather than modify the composition in order to be performed on the instrument. Traditionally these modifications could include transposing the piece to a more suitable key for the instrument, or removing awkward double stops and sustained notes impossible for the instrument in question.

2.7 Electronic Chamber Music

Traditional chamber music is often associated with an intimacy between the performer and the music. This applies to both solo performers and members of a small ensemble. This intimacy is not only common to the performer but also to the audience that is listening to the performance.

Intimacy is a quality not normally associated with modern electronic performance. In modern electronic performances a separation can often be felt between the audience and the performers. A number of issues contribute to this schism. These include the physical
separation between the performers and the audience, the unnatural sound reproduction of a loudspeaker as opposed to the acoustic production of the natural instrument, and the difficulty audiences experience in identifying new performance techniques.

The intimacy of sound can be lost when it no longer emanates from the body of an instrument but is electronically reproduced through loudspeakers. Schizophonia – separated sound – is a term coined by R. Murray Schafer to describe this loss of intimacy. Schafer used the term to describe all sounds dislocated from their source via radio, recordings, telephones or other technologies and depicts this process as ventriloquising modern life (Schafer 1969).

Few composers acknowledge this problem and the use of mono-directional stereo amplifiers has become standard for electronic concerts. Dan Trueman however, has developed multi-channel spherical speaker arrays that mimic the way sound radiates from an acoustic source (Trueman 1999).

Trueman has designed and built a variety of different sized speaker arrays. These spherical speaker arrays are driven by external patch-bay drives with software simulating the “directional tonal radiative qualities” of the violin or other acoustic instruments (Trueman & Cook 1999). By replicating the acoustical properties of a violin the speaker arrays do not simply reproduce sound but rather act as an electronic substitute for the resonating body. Trueman’s work demonstrates that his spherical speaker arrays “[blend] better with acoustic instruments than conventional mono-directional speakers” (Trueman & Cook 1999).

Trueman has taken this idea further by incorporating speaker arrays into the design interface of new instruments. Trueman’s BoSSA, or Bowed Spherical Speaker Array, is a
representational violin consisting of a twelve channel speaker array with two reduced violins: the Bonge, consisting of four bowed sponges based on the four strings of the violin (Trueman & Cook 1999); and the Fangerbored, based on the fingerboard of the violin (Trueman & Cook 1999). A third reduced violin, the R-bow, is used to bow the Bonge of the BoSSA (Trueman & Cook 1999). The R-bow features the force sensing resistor design that influenced the ESBow’s force sensing resistor design as well as a bi-axial accelerometer mounted on the frog of the bow. The BoSSA was created with the explicit purpose of reclaiming the intimacy found in traditional chamber music by returning the sound source of electronic music to the instrument in the player’s hands (Trueman & Cook 1999).

Some composers use schizophonic sound to their advantage by writing music specifically intended to be performed on multi-speaker setups. The position or distribution of the sound source among multiple speakers can be actuated by gestural sensor data. A simple demonstration of this could involve the lateral position of the bow actuating panning between a pair of stereo speakers. As the violinist bows the violin they are also bowing the position of the sound source between the two speakers\(^1\). The position sensors in Camille Goudeseune’s E-violin give him the ability to move the electronic sound source according to his position on stage. The sensors also provide the possibility to ‘fling’ the sound source around the room by pointing the violin rather than physically moving there (Goudeseune 2004).

Connecting the actions of a performance with the sounds produced helps to engage the audience with the music. When this connection is unclear, audiences may feel disengaged from the performance. Disengagement is all the more likely with electronic instruments if an audience is not aware how sound is controlled or cannot distinguish between what is live and

\(^1\) This technique is discussed in chapter 4.5.1 and can be found on video 02 on the accompanying DVD-ROM.
what is pre-recorded. New electronic interfaces based on traditional instruments can allow an audience to recognise traditional gestures even if the effect is not the same as a traditional instrument. A bowed electronic controller allows the physical action of bowing to be easily recognised. Bowing is visible to the audience even when the musical impact of the gesture is not obvious.

Laurie Anderson used this principle in the design of two reduced violins that focused on right hand bowing techniques: the Viophonograph and the Tape Bow Violin. The Viophonograph is a stringless violin with a turntable mounted on the body and a stylus attached to the bow. Different pitches are recorded on the separate bands of a record. These are performed by raising and lowering the stylus as well as scratching across the record using traditional strokes (Goldberg 2000). The Tape Bow Violin is an instrument with a playback head mounted to replace the strings and a collection of violin bows with pre-recorded audiotape instead of horsehair (Goldberg 2000). These instruments both have immediately recognisable techniques with different functions than their counterpart in the natural violin.

There are various ways for a performer to engage an audience. Visual stimuli can enhance an audience’s perception as to how an instrument works. The performer’s gestures are perhaps the most easily recognised visual stimuli for an audience. Larger gestures, such as bowing, are easier to see from a distance than left hand fingering or subtle control of foot pedals, potentiometers and sliders.

Some composers use video projection to show the audience a closer look at what the performer is doing. Projections of computer screens can also be used to provide the audience the same visual feedback as the performer such as a score or software interface. Gestural data
from an electronic instrument can also be used to control a visual display. As the gestural data that drives this display is the same data that drives the sound synthesis the two will be intrinsically linked and the audience will not only hear but also see and feel each expressive transformation the composition may take. These live visual displays should not be confused with pre-recorded video footage that is displayed during a performance. A more formal approach is the use of a program booklet. While less engaging this method can convey detailed information about the instrument, composer, performer, or the composition.

Haptic engagement with the instrument is another important aspect of electronic chamber music. It is hard to imagine how the audience can connect with a piece of music if the performer themselves cannot.

The term electronic chamber music has been used in various contexts. I believe the best use of the term is to describe the restoration of a sense of intimacy normally associated with traditional chamber music in both solo and ensemble performances of electronic music. This can take the form of natural radiative acoustical properties, mutual understanding between participants, or a natural physical connection between performer and instrument. The most important aspect of electronic chamber music is the intimate nature of the performance and the atmosphere of its reception.
2.8 Connecting the Performer with the Music

The ESBow was designed to focus on feedback between performer and instrument and address the shortcomings of conventional electronic control interfaces. The connection between performer and music is reinforced through haptic feedback that comes through the physical act of bowing using the ESBow as opposed to performing with a conventional computer interface. This intuitive haptic feedback makes performing with the ESBow more like performing with a traditional instrument than performing with a conventional electronic interface. The intention was to create an interface that makes the performance of electronic music more like chamber music.

Electronic music has typically involved a performer interacting with a control interface consisting of potentiometers and buttons, a computer mouse or a typewriter keyboard. The interface interacts with the computer running musical software to create and shape the music. The ESBow simplifies this process by assisting the performer to engage directly with the music. It is not simply a connection between a performer and a computer but a connection between a performer and the music.

Electronic control interfaces are not usually known for their playability as an instrument and can often be viewed more as a computer interface. The ESBow is an instrument that is also a computer interface, rather than a computer interface that serves as an instrument. The ESBow offers the adaptability that comes with a programmable electronic interface while retaining the playability of the violin with its identifiable techniques that assist in audience engagement.
The ESBow is independent of the violin. The ESBow can be used to perform on any object with an edge and not just on a stretched string. However, it was not the intention to use the ESBow to bow artificially created surfaces. For this reason the ESBow does not simulate haptic feedback electronically but relies on actual haptic feedback.
3. The ESBow – The Incubation Process

This chapter discusses the origins of the ESBow project and a preliminary ESBow design constructed during my honours research. The preliminary design is referred to as the ESBow 1.0. A discussion of the motivating factors behind this and my current design are detailed in 3.2. A review of the preliminary design conducted at the commencement of my current research is provided in 3.4.

3.1 Origins

My personal interest in electronic violin controllers was aroused when I first considered the possibilities of a violin bow that could sense the point along the hair that it touched the string of the violin. The idea was founded on the performance techniques of a theremin and would see the point of contact along the length of the bow determining the saturation of an effect on the music. I subsequently discovered this longitudinal monitoring of bow movement had previously been accomplished (Paradiso & Gershenfeld 1997) and much more was possible.

A comprehensive survey of electronic bowing interfaces conducted during earlier research, revealed a number of electronic violin controllers constructed in the last two decades (Murphy 2007). These ranged from traditional violins with electronic sensors attached without compromising the structural integrity of the instrument, such as the Augmented Violin Bow developed at IRCAM (Bevilacqua et al. 2006), to violins reconstructed from ground up to include sensors within the interface of the violin, such as Dan Overholt’s
Overtone Violin (Overholt 2005). The sensors used in these instruments monitored a range of techniques from traditional violin performance. These included changes in the pressure placed on the bow, the position and acceleration of the bow, the downward, lateral, torsional and frictional strains on the bow, the position of the fingers of the left hand, the angle of the right hand wrist and the amount of bow hair in contact with the strings.

The motivations for these controllers were also varied. A number were designed for research and the close study of violin performance technique, such as the Augmented Violin Bow developed at IRCAM (Rasmimanana 2004). This research was intended to enhance and further develop violin technology, techniques, playability, and teaching methods. Reduced violins are particularly useful in this area and are often constructed for the study of a specific area of violin performance technique. The results of these studies could lead to improved violin controllers, refined performance techniques and superior synthesis models. Other controllers were built for the realistic synthesis of the violin beyond what standard MIDI controllers could provide, such as Charles Nichols’ vBow (Nichols 2003). Most violins designed purely for composition and performance were primarily intended for the sole use of the designer. Dan Overholt’s Overtone Violin is one example of this (Overholt 2005). The MIT Hyperbow is an exception to this rule and is an example of an instrument designed for multiple users (Young 2006). Nearly all instruments involved in research were also used for composition and performance.

Some designers constructed electronic instruments that were focused solely on the bow of the violin. These included the IRCAM Augmented Violin Bow (Rasmimanana 2004) and the MIDI Bow (Rose n.d.) among others. A bow only controller has the advantage of being as stable and durable as any permanent representational violin without altering the body of the
natural violin.

3.2 The ESBow: An Overview

My own approach has been that of a young, unfunded composer with a minor background in electronics. I was primarily interested in composing for the bow and not to use it as a tool for research. The preliminary ESBow design discussed in 3.3 was intended to test the suitability of each sensor for use in a bowed instrument. The principal motivator throughout my current research is the physical connection between the ESBow and the performer. The performer should be able to feel their performance as they would when playing a traditional violin\(^2\).

The current ESBow prototype is intentionally simple in design and interface. I did not want a controller that would precisely monitor every aspect of performance technique. A simple design would both be easy to construct and easy to use for performers with no prior knowledge of the project. The performer should understand intuitively what each sensor is doing during a performance without having to study the instrument. To achieve this it consists of a small number of sensors that provide a simple numerical output in MIDI format. MIDI is one of the most commonly used methods of controlling audio software and could improve the uptake and acceptance of the ESBow by opening it to a large worldwide community of MIDI users. It also allows all musicians that have experience with electronic music to compose and perform with the ESBow without requiring experience in computer programming.

\(^2\) This idea was further elaborated in the previous chapter.
I felt it significant that the audience as well as the performer should have some understanding of the instrument. If an audience fails to understand how the performer uses a new electronic instrument this creates a barrier between the listener and the music. This is especially true when it is hard to distinguish between what is live and what is pre-recorded. This barrier usually persists regardless of the quality of the composition and can detract from the appreciation of the music. A traditional violin bow makes it possible for the audience to recognise familiar gestures. These gestures can help the audience to associate actions with the music even when the musical impact of the gestures is not apparent. Such gestures are primarily used to bring a natural performance technique to electronic music. They are not used simply for theatrical effect. They do however restore some of the natural theatre that was present in concert music prior to the advent of electronic music.

The decision to focus solely on the bow and not the violin was made for a number of reasons. Aside from the benefits bow controllers possess as previously discussed (2.1), the most compelling aspect of a bow design is the ability to use the bow in contexts that do not require a violin. Not only can the ESBow bow any non-violin stringed instrument, but any non-instrumental surface with an edge. A performer could easily bow a music stand or the desk on which their computer sits. The ESBow’s appeal lies in the ability to perform electronic music with an instrument rather than with a computer interface. The natural bowing movement of the ESBow is extremely versatile in its ability to bow any object with an edge with its sensors mapped in any conceivable way.

The ESBow was not intended to monitor every aspect of violin performance technique, but rather to convey the natural feel and control of the violin. The ESBow was also not intended to drive a perfectly replicated virtual violin. Nor was it intended for the purpose of in-depth
study or the creation of data tables for research. Most importantly, the ESBow was not proposed to replace or outdate the natural violin, but rather to provide a new avenue for its use in electronic music.

There are three ESBow prototype designs discussed in this thesis. The preliminary design (ESBow 1.0) was constructed prior to this project and is discussed in the next section of this chapter. The second prototype (ESBow 2.0) is the design planned for construction at the commencement of my current research. This design is based on the first prototype and is discussed in Appendix C. The final prototype (ESBow 2.1) is an amended version of the second prototype that features modifications that occurred during the construction process. It is discussed in chapter four with detailed illustrations provided in Appendix B.

3.3 The ESBow 1.0

The preliminary version of the ESBow designed during my honours research (ESBow 1.0) was influenced by the intentions discussed earlier in this chapter. The sensors included in this design were affordable and easy to use. They were mounted in a semi-permanent fashion in order to be stable but not do any permanent damage to the violin bow. The design monitored the force applied to the hair of the violin bow and the acceleration and tilt of the bow in three directions. This was performed by two force sensing resistors mounted underneath the hair of the bow and a single tri-axial accelerometer mounted on the frog of the bow. The two force sensing resistors were mounted on light foam in order to rest underneath the hair of the bow without impeding its progress along a string or other bowed surface. This was based on the design of Dan Trueman’s R-bow (1999).
The accelerometer mounted on the frog of the bow was also based on the design of other electronic violin controllers such as the R-bow (Trueman 1999), Jon Rose’s MIDI bow (Rose n.d.) and the MIT Hyperbow (Young 2001). Each of these instruments contains bi-axial accelerometers in a single or dual design. The dual design featured two accelerometers mounted at ninety degree angles to each other in order to monitor a third axis of violin bowing (Young 2001). I decided to proceed with a single tri-axial accelerometer instead of a dual array as it would prove cheaper, lighter and easier to mount.

Along with the sensors monitoring the natural movement of the bow was a miniature trackball with momentary select. The trackball was mounted on the frog of the bow to be manipulated by the middle finger of the right hand. The trackball provided control of two-dimensional space such as that of a joystick or internal rollerball in a computer mouse. It can also be clicked like a pushbutton or toggle. Curtis Bahn included a similar mouse touch-pad under the fingerboard of his Sbass. This offered two axes of continuous control along with several extra buttons (Bahn 2000).

All of the sensors connected to a PIC based MicroCV microcontroller via ribbon cable. The microcontroller transmitted all of the sensor data to the computer workstation via a wireless Bluetooth connection. It also provided power to the sensors which was supplied by a 9V battery. The microcontroller and battery were not located on the bow but strapped to the performers arm to reduce the weight of the violin bow and to help maintain its balance point to assist a more natural performance. The microcontroller was not permanently attached to the bow or the other electronics. This allowed it to be used in other projects and reduced the effective cost of the ESBow. While the ESBow was connected to the electronics strapped to

---

3 The MIT Hyperbow was later upgraded with a single tri-axial accelerometer (Young 2007).
the bowing arm via a ribbon cable it was still considered a wireless setup. The performer has the ability to move freely about the performing area without being tethered to the computer workstation by the cumbersome wires associated with much electronic music. This is akin to the wireless guitarist who has freedom of the stage while a cable connects his guitar to a transmitter attached to the back of his guitar strap.

3.4 A Design Review

The hardware design was never fully functional as a performing electronic bow, but provided the hardware platform necessary for me to undertake exploratory practical evaluation of sensors and determine their suitability for monitoring bowing gestures (Murphy 2007). Financial restraints resulted in the inclusion of specific sensors that were available at no cost. This included sensors for monitoring the pressure placed on the hair of the bow. The ideal sensors for the task output voltage representing pressure applied to a single focal point with a small surface area. The sensors available were variable resistors that monitored the flex of a small surface area approximately one by two centimetres in length. The design of the ESBow was modified to allow for this type of sensor. Various methods for mounting the flex sensors underneath the hair of the bow were attempted. At the time the design was reviewed no method had been satisfactory in providing a consistent and reliable linear output that directly corresponded to the varying degrees of pressure placed on the hair of the bow. There was also scope for further testing the effectiveness of natural bowing movement using the trackball and accelerometer.
What was learnt during the process of sensor evaluation has informed the development of subsequent ESBow designs. ESBow 2.0, found in Appendix C, discusses the ESBow constructed during the current research project. The design is based on a microcontroller that has become widely used for computing applications involving new interface design. The flex sensor was also replaced with a sensor chosen specifically for the job required. Using a sensor designed to monitor the pressure applied to a small surface area removed the unnecessary design problems introduced by the flex sensor. This design was finetuned for the ESBow 2.1 prototype discussed in chapter four.

The prototype discussed in the following chapter remains true to the design of the original prototype. Each sensor is a component that reflects the original design. The data is still received and transmitted to a computer workstation by a microcontroller mounted on the forearm, though this microcontroller is not the same as that used in the original ESBow design. Finally, the sensor data is still converted to MIDI format to be available for use in any musical software application.
4. The ESBow 2.1

A new ESBow design evolved around an Arduino microcontroller. This chapter discusses the reasons for design choices, the functionality of the design and its musical applications. Discussion is focused on specific design elements. Each electronic sensor is discussed in terms of its hardware, functionality and application, together with illustrations of the ESBow hardware and examples of the software that describes its functionality. ESBow software includes Arduino firmware code and composition software written in Pure Data (PD). More detailed documentation of the ESBow hardware and related software can be found in Appendix B. A transitional Arduino design which links the preliminary MicroCV design to the final Arduino design is discussed in Appendix C.

4.1 Design Objectives: A Review

During the transitional design stages I re-assessed my design goals and my approach to the ESBow. I had previously sought to discover what could be achieved through the augmentation of traditional violins with electronic gestural sensors. This involved understanding what others had achieved through the design of electronic violin controllers and developing my own design (Murphy 2007). This resulted in the preliminary design described in the previous chapter. From this design came a transitional design based around the Arduino Diecimila which led to the ESBow 2.1 described in this chapter. Since its inclusion in the design of the ESBow 2.1 prototype, the Arduino Diecimila has been discontinued and replaced by the Arduino Duemilanove and more recently the Arduino Uno.
As each board has been developed to support previous model applications the ESBow 2.1 design would be compatible with all three microcontrollers and any subsequent basic Arduino board⁴.

My current research was focused on the ESBow as a solo instrument for performing electronic chamber music (2.7). It also demonstrates the musical possibilities of using the bow in contact with any object with an edge, a stretched violin string being just one of these.

The sensors included in the design are affordable and practical. They can easily be integrated into a handheld control device and are responsive to gesture in a way that performers can understand and relate to easily. The sensors can be mounted both temporarily and securely on a violin bow without damage to the bow or obstruction to the bowing action.

Figure 3 demonstrates the flow of control from the ESBow to the sound produced through audio speakers. Electronic sensors mounted on the ESBow monitor performance gestures. Data produced is sent via ribbon cable to an Arduino microcontroller where it is multiplexed and sent to the computer workstation via USB 2.0 cable. The software program PD interprets bowing gesture data, which can be used either to control audio within the PD environment or be output as MIDI control messages. MIDI messages can be used to interface with other software applications within the computer or external hardware MIDI devices such as a MIDI synthesiser. Audio from the chosen device or application is then routed to external speakers. Demonstrations in this thesis use the software applications PD (version 0.41.4-extended) and AudioMulch (version 2.1.1), or a hardware Juno-D MIDI synthesiser. Applications were developed over a number of years culminating on a PC running Windows 7.

⁴ Knowledge of the different specifications of each Arduino board is unnecessary to this discussion, but can be found on the Arduino homepage (Arduino n.d.b).
Figure 3: ESBow Flow Control.

Key
- Raw Control Data
- MIDI Data
- Audio
- Computer Workstation
4.2 MIDI

Serial communication is used to transmit sensor data from the Arduino microcontroller to the software program PD. Sensor data is then converted to MIDI control messages which can be transmitted between software applications and external hardware MIDI devices. A physical MIDI port and cable is required to communicate with MIDI hardware outside the computer. A driver such as MIDIYoke is used to communicate with MIDI software applications within Windows. MIDIYoke is a virtual patch driver that allows MIDI to be transferred directly from one program to another (O’Connell n.d.). Once installed it automatically appears in software applications that use MIDI. The same process can be achieved on a machine running Mac OS X using the IAC Driver in the Audio MIDI setup in the computer’s utilities.

Designs that combine microcontrollers and PD often rely on a network interface protocol called Open Sound Control or OSC (Wright 2002). While many developers and instrument designers argue in favour of using this protocol in lieu of MIDI, there are others who regard the OSC protocol as unnecessary demonstrating that the disadvantages of MIDI can be overcome by using a generic network transport such as UDP (Raes 2004) or isochronous packets over IEEE1394 (Schiemer 1999). Unlike OSC, these protocols offer several levels of error checking essential for high speed data rates.

In designing the ESBow, the abundance of available software and hardware options for MIDI control influenced the decision to use MIDI rather than OSC, which is more recent and offers relatively fewer options by comparison. MIDI allows the ESBow to be used with most music software applications and MIDI hardware devices such as synthesisers and sequencers. MIDI has a potential worldwide user base consisting of millions of musicians who already use and
understand MIDI. The relative simplicity of MIDI commands has allowed composers, performers, sound designers, producers, stage and lighting designers, pyro-technicians and others to devise elegant solutions to creative problems by appropriating MIDI for a variety of applications despite its engineering limitations.

Even though MIDI bandwidth is limited by the baud rate of standard MIDI hardware, OSC offers little design advantage. While OSC increases the speed of the communication channel, its packet format is based on a hierarchy resembling the directory structure of a computer file system. OSC packet transmission involves increased overheads such as transmitting back slash characters that separate different levels in the hierarchy in order to deliver a payload that is just as easily transmitted as a MIDI message packet. Any gains made by increased OSC bandwidth tend to be offset by the relative increase in OSC packet size. By comparison MIDI message packets are small, typically one, two or three bytes and MIDI Running Status offers a very efficient way for MIDI to conserve bandwidth even at a baud rate of 31.25 kbaud. As sensor data is sent to the computer workstation over a USB 2.0 transport there is little difference in speed between the MIDI and OSC packages. Moreover PD objects netsend and netreceive also make it possible to take advantage of the smaller packet size of MIDI by transmitting control information as MIDI packets via the UDP protocol.

Further discussion on the benefits of MIDI and OSC communication is beyond the scope of this thesis.
4.3 The Arduino Microcontroller

The most significant change to the initial design described in the previous chapter was the replacement of the MicroCV microcontroller with an Arduino, shown in Figure 4 (Arduino n.d.b). The Arduino was chosen because it is simple, accessible and has a growing community of developers. The Arduino connects to the computer workstation via a USB 2.0 cable. This removes the necessity to provide a 9V power supply for an untethered device. It also bypasses the possibility of experiencing any wireless issues during construction. This simplifies problem solving and decreases the cost of the prototype. The Arduino also has a host of analog and digital inputs and in-built analog to digital conversion with a considerable library of software support.

![Figure 4: The Arduino Diecimila.](image-url)
The Arduino is used to create a multiplexed packet of analog and digital data. This is transmitted serially from the Arduino to the computer via USB 2.0 cable to be read in the software program PD. The Arduino code that performs this operation can be found in Appendix B.1.

4.3.1 The ESBow Daughter Board

The Arduino is used to mount a daughter board which routes data from the sensors to the digital and analog inline sockets on the Arduino, as shown in Figure 5a. Sensor data is transmitted to the daughter board via two ribbon cables shown in Figure 5b. The daughter board contains a small recess that provides unobstructed access to unused digital inputs of the Arduino, as shown in Figure 5c. The daughter board also routes power and ground to the sensors with 220k current limiting resistors on the inputs of the two force sensing resistors.

Figure 5: The ESBow daughter board.

A diagram illustrating the layout and wiring of the daughter board can be found in Appendix B.2.
4.3.2 Housing the Arduino

The Arduino microcontroller hardware is mounted on the bowing arm. It is encased and stored in a small plastic container to protect the electronic hardware. This is shown in Figure 6 and Figure 7. The lid is removed during performance to allow ribbon cable to access the daughter board. The USB 2.0 cable exits the container via a hole drilled into its side. Two elastic straps hold the Arduino to the bowing arm. These exit the container via four small holes drilled into its base. Three Velcro spots on each strap act as fasteners to provide a firm and adjustable fit.

Figure 6: The Arduino secured in housing box.
4.4 Pure Data

Sensor data is collected and processed using Pure Data (Puckette n.d.). PD is an open source real-time graphical programming environment available on several platforms. It was originally developed by Miller Puckette and has since acquired a growing community of developers and is consistently updated. It involves a composition environment suitable for my own creative work and also provides a technical and creative foundation for the ongoing development of the ESBow. One of the principle reasons for using PD in the prototype stage is its ability to not only quickly load a new PD patch but also the ability to modify a patch during a test or performance.

A default patch was developed that interprets all gestural sensor data and outputs each stream as MIDI control messages. This patch was designed to be easily accessible to violinists with no programming experience while providing the user with complete control over the sensor
data streams. It can be extended for future compositions by manipulating or combining data streams in any way before the streams are output as MIDI.

Figure 8 depicts the flow of gestural data within the default PD to MIDI interface from raw sensor input to programmable MIDI output.

![Flow of gestural data within the default PD to MIDI interface.](image)

Figure 9 shows a labelled screenshot of the default PD to MIDI interface used to map ESBow sensor data to MIDI. Operation and calibration tools are contained in sub-patches. This provides the user with a simple and tidy interface.
Figure 9: The default PD to MIDI interface.

The three objects at [A] control the transfer of sensor data from the Arduino to PD through the Input sub-patch at [B]. The left-most object enables or disables data transfer while the communications port is opened and closed by the middle and right-most objects respectively. The Input sub-patch also calibrates sensor input streams. This sub-patch can be found in Appendix B.3.2.

Five calibrated sensor input streams appear on the patch as vertical slider objects at [C]. The slider objects provide a visual indication of the sensor output level rather than a numerical read-out for convenience in live performance. The two left-most slider objects display readings from two force sensing resistors while the third, fourth and fifth slider objects from the left display readings from the X, Y and Z axes of the accelerometer.

Five toggle objects at [D] indicate the state of encoded digital inputs associated with the trackball. These include ‘Sel’ which shows the state of the momentary select button on the trackball, ‘U’ and ‘D’ which show the encoded states of the up and down directions of the trackball, and ‘L’ and ‘R’ which show the encoded states of the left and right directions of the
trackball. The four directions are decoded and combined to create vertical and horizontal *slider* objects that are controlled through the movement of the trackball. The ‘Sel’ input is used to determine the output of a *number box*, which is toggled between 0 and 127.

The output of the *slider* objects and *number boxes* at [C] are assigned different MIDI channel numbers using eight *send* objects at [E].

The *MIDI_Output* sub-patch at [F] receives the data streams from [E] and outputs them as 7-bit MIDI control change messages. This sub-patch can be found in Appendix B.3.2.

A detailed version of this patch with all functions in a single canvas can be found in Appendix B.3.1. The PD examples in this thesis are based on this detailed patch.

Functions that are specific to a composition are inserted into the data streams after the outlets of objects at [C] and [D] and before the inlets of objects at [E]. The row of *toggle* objects underneath [D] is not necessary for the default control of the trackball but can assist with other trackball techniques discussed in chapter 4.7.3.

A significant feature of the PD input to MIDI patch is the way in which it updates its input streams. The patch is not interrupt driven but rather takes readings of the sensors by polling the inputs of the Arduino at a constant rate of fifty times a second -- once every twenty milliseconds. Just as the Nyquist frequency of sampled audio needs to be sufficiently high to avoid audible artefacts known as aliasing (Shannon 1949), snapshots of the electronic sensors also need to occur at a rate that satisfies the sampling condition to capture bowing gesture accurately.
The Arduino is polled using clock pulses produced in PD by a *metro* object in the input sub-patch. The multiplexed packets of analog and digital data are received from the Arduino via serial USB 2.0 connection using the *comport* object. These are de-multiplexed and separated into individual analog and digital data streams by the *unpack* object. Analog data includes readings of both force sensing resistors and all three accelerometer axes. Digital data includes the two shaft-encoders and high/low switch of the trackball.

Each time the *unpack* object is clocked it will transmit the clock pulse into each sensor data stream. These pulses act as periodic events called *bang* events in PD. Consequently, momentary events in a data stream will be triggered with each clock pulse. In order to trigger momentary events as an exclusive event, the clock pulses must first be removed from the data streams.

The digital input streams use a *sel* object to differentiate between clock pulses and a change in state due to physical actuation such as clicking or rolling the trackball. Without this patch it was impossible to produce a *bang* event that was unambiguously associated with the selection of the trackball or the decoding of movement in an axis.

![Figure 10: Simple binary event detector.](image)

Figure 10 shows a Boolean *sel* object that compares its two inlets. It triggers a single *bang* event from its right outlet each time the *toggle* changes state. This patch takes advantage of a sequential delay between the two *patchcords* connecting the *toggle* outlet to the left and right
sel inlets. No noticeable lag is introduced using this technique as the sequential delay is less than one millisecond.

This operation is performed for each directional input of the trackball in its default operation. It can also be used on the momentary select in order to trigger events by clicking the trackball.

The process of removing bang events from analog data streams is not as simple as the Boolean process for digital streams. This is both due to the multiple states to decode and to the nature of number boxes to create a bang event each time they are actuated.

Bang events will not transfer through the right inlet of an object. By connecting an analog stream to the right inlet of a float object this stream can be preserved without the clock pulse produced by the metro object that polls the Arduino. A new metro object is used to strobe the left float inlet causing it to output the analog data as demonstrated in Figure 11. The only bang event that appears in this signal is the clock pulse produced by the new metro object. This clock can be tailored to the work or sensor by synchronising a data stream to the tempo of a piece or controlling the clock regulation with another data stream. The result is a rhythmic stepwise motion in the data stream rather than a glissando-like slide.

![Figure 11: Data sampling.](image-url)
Video 01 on the DVD-ROM features a simple application of this technique to control the pitch of an oscillator. The pitch is determined by a force sensing resistor mounted close to the frog of the ESBow. The tempo of the metronome is set by the Y axis of the ESBow. As the bow is tilted point upwards the changes in pitch occur more rapidly and as the bow is tilted point downwards they occur less rapidly.

Other applications of this technique are discussed in chapter 4.7.3.
4.5 Force Sensing Resistors

Force sensing resistors (FSRs) are included in the ESBow 2.1 design to monitor bow pressure.

4.5.1 FSR Hardware

The Flexiforce pressure sensors used to monitor bow pressure are shown in Figure 12 (Tekscan 2005). These are piezoresistive sensors that monitor the pressure applied to a small circular area at one end of the sensor. The greater the pressure placed on the area, the lower the resistance and higher the output. The FlexiForce sensors are extremely well suited to this task and pressure exerted on the mounted sensors during performance falls within the pressure specifications of the sensor. The diameter of the circular pressure point is also within one millimetre of the width of the hair of the bow.

![Figure 12: The FlexiForce force sensing resistor.](image)
The circular ends of the sensors are mounted underneath the hair of the bow on light foam as shown in Figure 13. One sensor is positioned at the tip end of the bow and another at the frog end of the bow. The foam is firm enough to hold the sensors in place while not damaging the stick of the bow or interfering with the ability to perform smooth bow strokes. The foam is shaped with a wide and stable base that is curved around the stick of the bow. The top of the foam has a smaller flat surface for holding the FSRs in contact with the hair of the bow. Once the foam is cut to size it is compacted by holding it in place in a tightened bow. This ensures the base of the foam is curved to the correct angle and retains the correct shape for use.

Twist ties are used to hold the FSRs and their connecting wires firmly against the stick of the bow and out of the way of the hair of the bow as shown in Figure 13. This is particularly necessary with the FSR situated at the tip end of the bow. While twist ties are not aesthetically pleasing they are easily removable and do not pose the risk of damage to the bow stick. They provide an effective and convenient but temporary solution in a proof of concept prototype.
A single grounded input is used to separate the two FSR inputs on the Arduino. This is to resolve interference associated between the two sensors when mounted adjacent to each other. Details of the tests that arrived at this conclusion are contained in Appendix C.4.

The output of the two FSRs can be combined to monitor the position of pressure along the length of the bow. This position is relative to the position of the two FSRs. The normal region for bowing with this configuration is between the two FSRs. Bowing outside this region will shift to the polar ends of the position sensor. Relative position sensing is achieved by subtracting the output from the FSR at the tip end of the bow from the output of the FSR at the frog end of the bow. The FSRs are effectively working as the one sensor and demonstrating the ESBow as more than the sum of its parts.

Figure 14 shows the two outputs from the FSRs subtracted from each other and adjusted to the MIDI range of 0 -127. This process requires calibration before the positioning sensor can be used to ensure it covers the full MIDI range in practical use. A detailed description of the calibration process can be found in Appendix B.4.1.
A simple demonstration of this sensor involves panning an audio source between two stereo speakers based on the position at which the bow is touched between the two FSRs. A variant of this can be achieved by mapping the output of each FSR to the volume of a separate speaker in a stereo configuration. Where the first technique combines the output of the two FSRs, the second uses the two outputs as separate streams. As each technique approaches the task of panning using a different function, each technique acts in a similar but unique way.

Demonstrations of each technique can be found in videos 02 and 03 on the accompanying DVD-ROM.

4.5.2 FSR Functionality

The FSRs provide consistent, predictable and reliable outputs of a linear nature. The effective range of actuation varies according to the nature of different bowing techniques. The FSRs do not hamper the fluid movement of bow strokes when bowed flat over the sensor. The bow can also be tilted in either direction however the bow cannot be tilted while crossing the sensor due to a slight protrusion from each side of the hair of the bow. It would be possible to trim the edges of the sensor to minimise or remove this protrusion however this was not executed in the prototype design.
As the foam mounts are cut by hand and can be transferred between various bows they are of non-uniform size. A difference of one or two millimetres will impact on the output of the FSR. The FSRs will therefore need to be recalibrated when the foam is replaced or moved to a new bow. This involves a simple modification of the FSR input stream in PD as demonstrated in Figure 16.

The process consists of subtracting from the input stream to set the FSR at rest as close to zero as possible. The *slider* object will limit the output so the FSR at rest will output zero rather than a negative number. The number subtracted should optimally be less than 10. Subtracting a number over 20 will result in the loss of the upper range of the sensor stream. It
is also an indication that pressure is being placed on the hair of the bow via the foam. This could affect the playability of the instrument. Foam that falls into this range requires trimming or active compaction until it is of suitable size.

Standard bowing does not compact the foam underneath the sensor. Bowing heavily with two hands on either end of the bow stresses the foam but does not cause permanent damage to the foam. Bowing heavily while tilted on an angle causes this stress to be focused on one side of the foam and increases the compaction of the foam on that side. After a period of this style of bowing considerable compaction occurs and the foam may eventually need to be replaced.

A comparison was made between the stability of pressure readings produced using new and compacted foam. Once recalibrated, compacted foam provides stable pressure readings with only a minor loss in sensitivity. Compaction that may occur during a performance should not have a negative impact on the performance. For the ESBow, foam can be considered a consumable item just like bow resin or violin strings.

4.5.3 FSR Application

The two FSR outputs can be mapped in a variety of ways to create different effects. One technique is combining their outputs to create a single output for pressure. This signal is consistent along the length of the ESBow.
Video 04 on the accompanying DVD-ROM features an example of this technique. In this video the volume of an oscillator is mapped to the combined output of the two FSRs. A slow stroke is used to produce a stable volume across the length of the bow.

The FSR outputs can also be mapped to two different processes. In this way the performer can bow in the upper or lower half of the bow to insert two different effects on the audio stream. The performer can also merge between effects by bowing between the FSRs. These could be any two processes the composer desires and can be considered an extension of traditional bowing techniques that focus on the upper or lower halves of the bow.

An example of separate streams being mapped in the one bowing action can be found in video 05 on the accompanying DVD-ROM. The video demonstrates the ability to crossfade between audio streams using the bowing action of the ESBow.

A combination of both methods could also be used. The FSRs would be combined to provide one output for pressure and simultaneously provide two separate outputs for each FSR. In this way a performer could bow along its length to cross-fade between two sound sources while simultaneously adjusting the length of a delay effect through the pressure placed on the hair of the bow.
4.6 Accelerometer

A tri-axial accelerometer is included in the ESBow 2.1 design to monitor the acceleration and tilt of the bow in three directions.

4.6.1 Accelerometer Hardware

The accelerometer used in the ES Bow prototype is a Freescale MMA7260 tri-axial accelerometer (Freescale 2008) mounted on a Polulu breakout board as shown in Figure 17 (Polulu n.d.). The three axes of the accelerometer provide independent analog data streams. It is powered by a single 3.3V connection and has four adjustable sensitivity settings of +/- 1.5, 2, 4, or 6g⁵. The sensitivity is manually set using jumpers to ground two pins on the accelerometer breakout board. Both pins of the accelerometer were grounded for the demonstration examples found in this thesis. This sets the sensitivity at +/- 1.5g and allows for the subtle tilt of the ES Bow to have a significant effect on the data stream. Performers retain control of the sensitivity setting through the ability to remove the jumper pins.

An L shaped extension, shown in Figure 17, ensures the ribbon cable for the accelerometer clears the end of the bow and does not interfere with bowing action or disturb the bowed surface.

⁵ A g is the unit used to measure acceleration. A single g is equal to the Earth’s gravity at sea level.
4.6.2 Accelerometer Functionality

Accelerometers monitor the simultaneous influence of two forces upon the sensor. These are known as dynamic and static acceleration. Dynamic acceleration is the result of the acceleration of the sensor in three-dimensional space due to physical movement. Static acceleration is the result of the influence of gravity upon the accelerometer due to the tilt of the sensor relative to a horizontal plane as defined by a spirit level. Dynamic and static acceleration are not separated in the output of an axis, however, these techniques will be discussed separately.

The dynamic movement of the ESBow is associated with the level of displacement in the output stream of each accelerometer axis. Figure 18 shows four deliberate bow strokes represented as MIDI data on a makeshift oscilloscope. The output signals are derived from

Figure 17: Freescale MMA7260 tri-axial accelerometer.
the Y axis of the accelerometer. In the quiescence state the value hovers around the mid-point of the MIDI data range when held in a horizontal position. Down-bow strokes produce a peak followed by a trough and up-bow strokes produce a trough followed by a peak. In the diagram down-bow strokes are indicated with the symbol ∏ and up-bow strokes are indicated with the symbol V. In each case the twin displacements of a single stroke will be approximately equal when no other influence is placed on the axis. The level of displacement will increase with the degree of acceleration of the bow. Despite lack of fine control it is therefore possible to differentiate between large and small accelerations in an axis.

![Graph showing dynamic movement in Y axis of ESBow](image)

**Figure 18: Dynamic movement in the Y axis of the ESBow.**

The accelerometer can also be tilted in the direction of the X or Y axes. This offers the most precise control of the accelerometer. Tilting the bow will not cover the full range of the sensor’s output. The PD patch can be modified so the range of tilting covers the full MIDI range, this is shown in the detailed PD patch in Appendix B.3.1. Consequently dynamic acceleration in these directions can move beyond the available range. Limiting the range of each MIDI stream within the default PD input patch prevents this from creating errors in the output MIDI stream. This will not be a problem if the performer is intending to primarily use a tilting action for this axis. This technique can be useful for the X axis which is not as physically actuated during traditional bowing as the other two axes.
Figure 19: Combined dynamic and static acceleration in the X axis above with rapid dynamic acceleration in the Y axis below. This was achieved by slowly rocking the ESBow on its X axis while performing a tremolo action.

An axis that is primarily used for dynamic acceleration can also be tilted during performance to localise the resultant displacement in the output stream. Figure 20 demonstrates the result of tilting the ESBow along its Y axis while performing tremolo strokes.

Figure 20: Localising the output of the Y axis.

When performing in a traditional violin pose, this is accomplished by bending forward or arching backwards while bowing. Dan Trueman discusses using this technique with the R-bow (Trueman 1999). When bowing a non-conventional object the performer is free to bow from any angle possible. Cylindrical objects that allow the performer to approach from any 360 degree angle offer the most freedom to a performer in this respect. An object could also be chosen because of the limits it places on the range of bowing angles.
The Z axis can also be tilted. It outputs high when upright and progressively decreases when tilted in any direction until held upside down where it outputs low. The motion of turning the bow upside will also impact on either the X or Y axis depending on which side the bow is turned. Due to this the Z axis cannot be actuated through tilt without also actuating one of the other two axes and neither of the other axes can be actuated through tilt without also actuating the Z axis. This limits the practical use of tilting the Z axis in many mapping situations. Using an expression object to determine the orientation of the Z axis allows the performer to swap between bowing upright and upside down as a method of control.

Video 08 on the accompanying DVD-ROM features an application of this technique.

The Z axis can also be dynamically actuated by quickly raising or lowering the ESBow. Like the Y axis this can be localised by tilting the bow. In most cases the ESBow will be held upright when bowed. This will cause the initial output level of the Z axis to be relatively high and should be considered when mapping or preparing the data to be used for performance. Motions such as lifting and dropping the bow onto an object are particularly effective in the Z axis. The frog of the bow can also be dropped while maintaining contact with a surface. This motion swings the bow so it is tilted upwards and will also affect the Y axis of the accelerometer.
4.6.3 Accelerometer Application

The orientation of the bow in the tilt of its X and Y axes is monitored by the accelerometer as static acceleration (4.6.2). The simplest use of tilt is that of a virtual potentiometer. This technique uses tilt orientation to determine the output of an axis.

For example, tilting the bow in the X axis anti-clockwise to the nine o’clock position produces a MIDI value of 0 while tilting the bow clockwise to three o’clock produces a MIDI value of 127. Rotating the bow between these positions produces the full MIDI range with the twelve and six o’clock positions both producing a MIDI value of 63.

An alternative mapping technique monitors the displacement from the twelve o’clock position as a positive integer. In this technique a twelve o’clock position produces a MIDI value of 0 and the nine and three o’clock positions produce a MIDI value of 127. When this technique is applied to the X axis it can be compared to the traditional technique of tilting the bow to alter the amount of bow hair in contact with the string. Figure 21a. shows the process used to determine the displacement of an axis. A description of the technique can be found in Appendix B.4.2.
Positive and negative displacement can also be used to control two separate parameters with a single bowing axis, as shown in Figure 21b. In this technique anti-clockwise rotation towards the nine o’clock position produces negative displacement which increases the MIDI value of parameter A. Clockwise rotation towards the three o’clock position produces positive displacement which increases the MIDI value of parameter B. A description of the technique can be found in Appendix B.4.3.

Demonstrations of the last three techniques can be found in video 06 on the accompanying DVD-ROM.

Another technique can be used to separate the tilt in an axis into separate bands or steps. Holding the bow within one of these bands will allow the composer to set as many or as few parameters desired. This can be achieved using an expression object and essentially consists of as many as nine if-else Boolean statements. If more than nine conditions are required additional expression objects can be used.
In Figure 22 the data stream is separated into five separate streams based on the Boolean statements in the expression object. These compare the variable float ($f1$) input to set parameters. If the condition is true the variable is sent to its outlet. If the condition is false it outputs zero. A detailed description of this diagram and the functions of an expression object can be found in Appendix B.4.4.

A simple demonstration of this technique can be found in video 07 on the accompanying DVD-ROM. In this video the tilt of the Y axis determines the pitch of a tone within a pentatonic scale.

The creative work Kitchen (A.7) uses two sets of band separation. These are used to separate each axis of the trackball into four bands. The four bands are used to determine the sound source of the work and which effect is placed on the audio stream.

The direct displacement of an axis can be used to monitor dynamic movement however this technique is subject to two faults. The first is that the zero crossings in the output result in two peaks being created for each bow stroke. The second is that the dynamic displacement is affected by the orientation of the bow which the displacement process was designed to monitor.
A more useful output can be obtained from the displacement by determining the velocity of each bow stroke using integration. This results in a single peak per bow stroke that is not influenced by the orientation of the bow. Figure 23 compares the displacement of four deliberate down-bow and up-bow strokes to their velocity in the Y axis of the ESBow.

Figure 24 shows the process used to obtain the velocity of an axis. This consists of determining the displacement in an axis over a set period of time. A detailed explanation of this diagram can be found in Appendix B.4.5.
4.7 Trackball

A trackball with momentary select is included in the ESBow 2.1 design to provide two axes of parameter control and a pushbutton without interrupting bowing.

4.7.1 Trackball Hardware

The trackball is mounted on a Sparkfun breakout board as shown in Figure 25 (Sparkfun Electronics n.d.). It contains a momentary select and two rotary encoders with four directional outputs. When rolled in a single direction the trackball causes the relative output to progress along a sequence of binary encoded states. Control of the two axes of the trackball is achieved by tracking the number of state changes in each of the accelerometer’s outputs. The trackball has a single voltage and ground supply.

![Figure 25: The “BlackBerry” trackball.](image)
4.7.2 Trackball Functionality

The trackball rolls smoothly and predictably in each direction. These directions are most useful when combined into vertical and horizontal axes. The two axes can be used to control separate parameters or they can be linked to position a point on a virtual two-dimensional plane or metasurface. Linking the axes in this way allows the trackball to act similar to a joystick. A metasurface can be used in various ways such as positioning a sound source in a surround sound environment, or modifying the effects placed on an instrument by relating them to positions in the two-dimensional environment. A visual aid which provides the exact position and the outer limits of the two-dimensional plane can be of assistance with this technique but is not essential for performance.

Video 09 on the DVD-ROM features an example of the navigation of a metasurface to control the parameters of various effects on a sound source.

The performer can adopt various methods of performance using the trackball. As the trackball acts by progressing along a series of high and low states the performer can use short sharp movements in a single direction to change the state of one of the four outputs as they would with a toggle. This method is vulnerable to occasional glitches where an input will halt in the incorrect state. The issue can be assisted through the aid of a visual display of the input state such as on a computer monitor.

The performer can also combine the X and Y axes to form a single axis. This increases playability and predictability when bowing at certain angles where a level of difficulty may be encountered maintaining two separate axes. Alternatively, bowing with the ESBow held
parallel to the performer rather than pointing away from the performer greatly increases the ease of dual axis trackball techniques.

### 4.7.3 Trackball Application

If the trackball select is used as a high low MIDI toggle the sudden level change when the trackball is released can produce an undesirably sharp cutoff. Smoother transitions can be accomplished by ramping between values over a specified period of time.

![Image](image.png)

**Figure 26: Ramping the trackball select.**

Figure 26 shows two examples of ramping. The first is a pre-determined one second ramp. The second is a variable ramp controlled by the horizontal axis of the trackball. A variable ramp provides the performer with active control over the length of ramp on the trackball throughout performance.

Video 10 on the DVD-ROM demonstrates a comparison of a ramped and non-ramped effect.
The trackball select signal can also be used to update an analog data stream.

![Diagram of trackball select signal](image)

**Figure 27: Strobing an analog data stream with the trackball select.**

In Figure 27 the data stream is strobed into the *float* object whenever the trackball is clicked. In this way the performer can activate steps or jumps in a stream with precise timing.

Video 11 on the DVD-ROM uses two of these processes simultaneously with each applied to one axis of the trackball. Instead of dragging the cursor between locations on a metasurface a performer can locate the cursor by rolling the trackball and then activate the location by clicking the trackball.

By inserting a tally counter after the trackball select input, as shown in Figure 28, a performer can trigger events when the tally reaches certain targets. This technique can be used in a variety of ways with any number of preset events defined.
Figure 28: Trackball loop counter.

Figure 29 shows this technique applied to a short looped sequence. Using this patch the trackball will trigger the next sequential bang object each time it is clicked. When it reaches the end of the sequence it resets the count and the looped sequence will start over. A detailed description of this diagram can be found in Appendix B.4.6.

Figure 29: Activating a short looped sequence.

Video 12 on the DVD-ROM features a chord progression that is activated using this technique. A detailed diagram of this patch can be found in Appendix B.4.7.

The trackball techniques discussed thus far have centred on the use of the trackball’s outputs as two separate axes. A different technique would see each of the four outputs having an entirely separate stream. An extension of the technique shown in Figure 29 would allow for
separate sequences to be allocated to each direction.

This is demonstrated in video 13 on the accompanying DVD-ROM.

The compositional work *Four Rows of Twelve* (A.5) features numerous instances of this technique on the trackball’s select and directional outputs. These are used to control the progression of pitches in a tone row and the overall structure and development of the piece.

Alternatively the directional outputs could be used as four separate on/off toggles. Each direction could control a different effect on the audio or the playback of a looped sound source.

This technique is demonstrated in video 14 on the accompanying DVD-ROM.

One further technique monitors the duration that the trackball is held to trigger different events. This allows the trackball to simultaneously operate on multiple structural layers within a single work. In these cases simple setups with only two or three layers work best. These layers are separated through the use of short, medium, or long duration selections. Figure 30 shows the patch used to implement this technique. A detailed version of the diagram can be found in Appendix B.4.8.
A simple application of this technique is demonstrated in video 15 on the accompanying DVD-ROM.

In this example the audio progresses through five different pitches each time the trackball is selected as a momentary event. If the trackball is held for longer than one second the order of the pitches changes. The trackball can then continue to progress through the pitches as before. This can be repeated as many times as desired. If the trackball is held for longer than three seconds the audio fades to silence.

As can be seen in the video demonstration, each selection will trigger all events of a shorter duration in succession, i.e. a medium length selection will trigger both a short and medium length event and a long length selection will trigger a short, medium and long event. Consequently, the mapping of triggered events must be ordered appropriately so that no undesired event is triggered when a longer duration selection is triggered.
4.8 Reconfiguring the ESBow

The sensors are mounted on the ESBow using Blu-tac as shown in Figure 31. This was initially intended as a temporary solution. However, it eventually became obvious that there were benefits in attaching sensors in this way.

![Figure 31: The mounted sensors.](image)

The most obvious benefit is the quick and easy removal of the sensors from the bow. This had a significant impact in a number of ways. A detachable design allows the electronics to be transferred onto any violin bow. This allows performers to use their preferred bow which may be of a different length, weight, or quality than that which I can provide with the original bow.
More importantly the performer can adjust the trackball position to suit their performance style rather than adapt their playing to accommodate the position that suits my own. The significance of this becomes more apparent when the bow styles of various schools of violin performance technique are considered. Different schools of bowing technique, such as the Franco-Belgian or Russian schools, require different methods of holding the violin bow (Flesch 2000). While the methods are similar the precise position of the fingers of the right hand are slightly different. Blu-tac allows the performer to adjust the placement of the trackball to the exact position where it can be accessed by the middle or ring finger. It can also be moved to a position on the frog where it is less likely to be activated by involuntary movements but is still accessible to the middle and ring fingers. The other implication of this is that the ESBow can be made ambidextrous.

Detach able electronics simplifies travelling with the bow. It allows the Arduino and daughter board to be transported safely in its housing with the sensors in a second secure box. This allows the bow to travel in a conventional violin case separate from the electronics. It is not possible to transport a fully assembled ESBow in a conventional violin case. However, the FSR at the tip end of the bow can be left in place during transport to decrease assembly time.

Blu-tac not only attaches the sensors to the bow but also helps prevent the bow from being scratched by component pins and leads protruding from the solder side of the circuit boards. The Blu-tac also insulates tracks on the circuit board from contact with metal parts of the frog.

The use of Blu-tac conveys an ad hoc appearance. However, this was considered inconsequential at the prototype stage. The durability of Blu-tac was also initially considered
a potential issue. However, this proved not to be the case and its use offered some real design advantages.

Each sensor can also be detached from the header of the ESBow circuit boards. Removing sensors allows them to be tested in other configurations and sensor readings compared with those taken on the bow. Sensors can also be swapped easily when design changes are required or parts updated. The detachable headers of the sensors are shown in Figure 32.

![Figure 32: The sensors of the ESBow detached](image)

The reconfigurable design of the ESBow makes it possible to achieve FSR sensitivity that cannot be accommodated adequately by simple software recalibration. Initial tests with the ESBow and a violin found that the level of sensitivity in the FSRs for bowing non-stringed surfaces was insufficient when bowing strings with correct Helmholtz stick-slip motion. A more suitable output range is achieved by using a single FSR to monitor the force applied to the bow via the index finger as shown in Figure 33. This method was used in previous

---

6 The diagram features the original accelerometer breakout and intercept boards (C.5).

7 Stick-slip is the term used to describe the two-phase periodic motion of a bowed string first observed by Hermann von Helmholtz (Smith & Berdahl 2007). As the bow travels across the string it sets the string in motion producing a transverse wave; the bow ‘sticks’ as it is pulled continuously in one direction to a point where it then ‘slips’ in the opposite direction. Both phases alternate for the duration of a single bow stroke. Violinists apply rosin to the hair of the bow to increase the ‘stick’ or traction of the bow on the string.
designs such as the Hypercello (Paradiso & Gershenfeld 1997). As bow pressure is regulated by the index finger the placement monitors pressure without impeding bowing gestures. This eliminates the possibility of obstructing a tilted bow stroke that is bowed directly over either FSR (4.5.2). Composite relative position sensing is disabled in this design as it requires both FSRs. The single FSR configuration was included because it is better suited for violin playing whereas dual FSRs accommodate alternative bowing techniques used with other instruments or as a method for producing or controlling electronic sound.

To implement this design the foam mount for the FSR at the frog end of the bow is removed and the FSR placed directly on the wood. The sensor can be held in place using Blu-tac or through pressure applied by the index finger. The second FSR at the tip end of the bow is deactivated; it can either be removed or left in place and not read. The bow is calibrated with the index finger at rest. This ensures the FSR does not produce an output until the bow is brought in contact with the string. The performer can reconfigure the instrument between...
compositions or even during a performance. The remainder of this thesis will focus on applications of the dual FSR design.
5. The Gestural Language of the ESBow

This chapter focuses on common bow strokes in traditional violin performance technique and how these can be used with the ESBow. Bow strokes were monitored using a combination of signals produced by the accelerometer and FSRs in dual configuration. Output signals of each sensor were studied for predictable patterns as each bowing technique was performed using a variety of bowing surfaces. The term ‘bowing surface’ is used to include not only a stretched violin string but potentially any object that allows a player to produce pressure signals using the ESBow. This concept is discussed in detail in 5.11. Bow strokes were performed horizontally, i.e. parallel to the floor, in order to simplify the process of calibrating output signals produced by the accelerometer. These signals will change slightly in a violin performance where bowing tends to be performed at an angle to the floor. PD software patches (4.4) allow further calibration to accommodate the bowing action of individual performers.

5.1 Legato

Legato is a bowing technique that produces a smooth sound with no obvious break when the direction of bowing changes.

The action of legato bowing is monitored on the ESBow using a combination of signals produced by the FSRs and accelerometer. Legato bowing produces a smooth output stream in the FSRs. Bowing directly over either sensor produces a momentary spike in the output
stream. Combining FSR signals allows bow pressure to be monitored over the entire length of the bow. However, the optimal region to read pressure in legato bowing is between the two sensors. Changes in bowing direction produce a relatively small peak and trough in the Y axis of the accelerometer that increases with the speed of bowing. There are small fluctuations in the X axis during each directional change and minimal fluctuations in the Z axis throughout each stroke.

5.2 Tremolo

Tremolo involves short rapid alternating strokes focused on a single point of the bow.

The ESBow relies principally on signals produced by the accelerometer to monitor the action of tremolo bowing. It is characterised by rapidly fluctuating output values on the Y axis with some jitter on the X and Z axes. Tremolo can be played using a variety of dynamic levels which the performer controls by varying bow pressure. This produces a consistent output signal in the FSRs. Tremolo combines motion and pressure sensing in a way that offers potential for expressive control in the hands of an experienced violinist.

5.3 Détaché/Detached

The French translation of the term détaché literally means separate bows. A détaché stroke should not be confused with a detached bow stroke. In a détaché passage the direction of each stroke is alternated with a single note performed per stroke. In a detached stroke each note is
separated by stopping the bow on the string to deaden vibrations.

The output of a détaché stroke is similar to that of legato with some noticeable differences. The output of combined FSRs is consistent along the length of the bow with momentary spikes directly over the FSRs. Displacement in the output of the Y axis during a directional change is more pronounced than in a legato stroke. This is due to a possible increase in bowing speed and no attempt to minimise the impact of directional changes. These changes also occur more frequently and emphasis may be placed on the change.

A detached stroke has a unique output in the FSRs and accelerometer, which is derived from the speed and intensity of each halt in the stroke. A detached stroke shows no difference in FSR output from moving bow to stopped bow. The position of each halt can be derived from the relative position of the point of contact as monitored by both FSRs. Movement in the Y axis occurs both at the onset of each stroke and upon the abrupt halting of the bow mid-stroke. The faster the bow is travelling the larger the fluctuation in the output of the Y axis when the bow stops abruptly. There is minimal fluctuation in the outputs of the X and Z axes.

5.4 Martelé

A martelé stroke commences with the bow held against the string with pressure and then stroked forcefully to emphasise the note produced. A slight pause between successive notes allows the performer to apply pressure and produce a distinctive sequence of notes with similar emphasis.
The initial pressure placed on the hair of the bow before the stroke raises the output of the FSRs and drops when lighter pressure is applied through the stroke. The forceful strokes output a significant displacement in the Y axis. The Z axis also outputs distinctive peaks as the bow is raised and lowered on the string. There are minimal fluctuations in the X axis of the accelerometer.

5.5 Collé

Collé is a short stroke starting from a heavily weighted position and is usually performed near the frog of the bow.

Collé strokes produce similar signals in the sensor outputs to those of a martelé stroke. The combined FSRs produce a predictable short sizable spike due to the weighted start of the technique. The short sharp movement produces a significant displacement in the Y axis of the accelerometer. The Z axis produces peaks associated with raising and lowering the bow on the string and minimal fluctuation is present in the output of the X axis.

5.6 Spiccato

Spiccato involves bouncing the bow on a violin string to produce a short note with a distinctive sound envelope. It is typically performed with a single bounce per directional change.
Bouncing techniques such as spiccato produce a flurry of rapid output signals. Spiccato strokes produce a series of short distinct spikes in the output of the FSRs which correlate to the audible sound produced. Strong displacement in the output of the Z axis of the accelerometer corresponds to the acceleration of the bow as it is dropped and bounced off the bowing surface. Significant movement is also found in the Y axis due to the rapid changes in tilt during each bounce and rapid direction changes. There is minimal movement in the X axis.

5.7 Jeté

Jeté, or ricochet, involves the upper half of the bow being allowed to bounce naturally on the string.

As the technique is focused on the upper half of the bow, the FSR at the tip end of the bow produces a more robust signal than the FSR at the frog end of the bow. The signal is also more useful than the combined signals of both FSRs. If the ESBow is allowed to bounce a minimal number of times from a significant height, there are distinct separate spikes in the output of the FSR. When the ESBow is left to bounce numerous times with decreasing bounce height, the FSR does not output as distinct a pattern as can be heard in the audio of a natural violin string. There are considerable fluctuations in the Z and Y axes of the accelerometer as the bow bounces. The size of each fluctuation decreases with each successive bounce. There is minimal movement in the X axis.
5.8 Sautillé

Sautillé involves bouncing rapidly in the middle of the bow. The bouncing is very small and comes naturally from the short transverse movements of the bow.

The combined output of the FSRs produces a rapid sequence of small peaks. These peaks are less distinct than those of previous bouncing techniques such spiccato. The frequent direction changes create large fluctuations in the Y axis of the accelerometer. There is also considerable movement in the Z axis with minimal movement in the X axis.

5.9 Chopping

The modern jazz technique of chopping involves striking the hair of the bow near the frog against the strings to produce a quick scratching sound of indeterminate pitch.

Chopping produces a spike in FSR output equivalent to the pressure exerted on the bow when striking the string. A short and fast chopping motion also produces a sharp displacement in the Y axis of the accelerometer. As the bow is brought down onto the string, a significant disturbance is also produced in the Z axis. The X axis results in no direct output, but is influenced through the movement of the bow.
5.10 Col Legno Battuto

Col legno battuto involves tapping against the strings of the violin with the bow stick rather than the hair.

As col legno battuto does not involve the hair of the bow the FSRs do not produce an output and only the accelerometer is affected. A significant response is produced in the Z axis if the back of the bow stick is tapped or in the X axis if the side of the bow stick is tapped. The Y axis also responds to tilting with each bow strike when the back of the bow is tapped.

5.11 Bowing Surfaces

The ESBow was designed to bow a variety of surfaces. These include the vibrating strings of conventionally bowed violins and other chordophones. It also includes the application of bowing technique to all manner of idiophones where the edges and surfaces of vibrating objects are bowed. In this way applications of the ESBow potentially include many of the musical possibilities for producing sound identified in the Hornbostel-Sachs classification of musical instruments (Hornbostel & Sachs 1961). The concept of bowing surface that inspired the design of the ESBow includes any object with a protruding edge that can be bowed. This may include many objects not found in even the most exhaustive classification of musical instruments.

Objects for bowing can include found objects, such as statues, rocks and pieces of wood, metal or plastic. Objects can be used as found or modified to provide a more responsive
bowing surface. It could also include several objects combined to create a single object with more than one bowing surface or an object constructed specifically to be bowed by the ESBow. These objects may be held like a conventional violin between the chin and the shoulder or they may be freestanding objects placed on a table, a stand or the floor. Free standing objects offer the performer scope to extend the bowing techniques possible with the ESBow.

The act of bowing objects other than conventional instruments allows the performer to emphasise various aspects of bowing technique. These include using emphatic martelé or collé strokes or performing a legato stroke slower than would produce a note on a vibrating string. Bowing techniques that involve no transverse movement in the Y axis are also possible such as bouncing or applying pressure at a single point along the bow. Techniques can also be performed along an axis other than that usually associated with the technique. This could include rapid movement along the X axis to produce a tremolo-like output or a detached stroke performed along the Z axis.

The sensor outputs are more responsive when the ESBow is bounced on wires, rods and objects with sharply curved edges rather than objects with flatter gradual curves and a wider surface area. Bouncing can be performed on flatter surfaces but is not as effective as bouncing on an edged surface where the impact is concentrated on a smaller focal point of the hair of the ESBow. If the ESBow is bounced directly over the FSRs its spring is dampened by the foam mounting. This is not an issue when bounced between or close to the FSRs and is less likely with a compacted foam mount (4.5.2). When relative position sensing (4.5.1) is used with bouncing techniques the output signal will appear to originate midway between the sensors when the bow is not in contact with the string. When the bow makes contact with the
string the output signal will momentarily shift towards the relative location of the point of contact; bouncing near the tip of the bow will spike to the left and bouncing near the frog will spike to the right.

It is possible to bow more than one object simultaneously. The ESBow can be laid across two surfaces with pressure placed upon one or both surfaces. The ESBow can also be dragged across either surface. Alternatively, bowing surfaces can be made with different types of surface materials. This is especially exciting if a number of different bowing surfaces are configured in an array that allows various combinations to be played either simultaneously or consecutively. It should be noted that the method for detecting the point of contact relative to the FSRs will only produce a single output reading when multiple objects are bowed simultaneously. This will be at some location between the two points of contact and depends on the varied pressure placed on the bow at each point.

Most of the bowing surfaces used with the ESBow thus far have been relatively smooth. This was to reduce the likelihood of damage to the ESBow during testing. However, composers can use the ESBow on any object with an edge at all. The ESBow design allows sensors to be attached easily to any bow. This allows a violinist concerned about the risk of possible damage to an expensive bow to attach the sensors to a less expensive bow. However, care should be taken to ensure that FSRs under the hair of the bow are not harmed by bowing potentially damaging surfaces, such as brittle edges that might shred bow hair. If a sensor is damaged it can be replaced easily without replacing the entire ESBow electronics. Alternatively, a single FSR can safely read bow pressure from the index finger (4.8) in order to remove the FSRs from proximity to the potentially damaging surface.
The ESBow can be moved in the air without making contact with a bowing surface yet still produce control data from the accelerometer and trackball. This presents a new range of sensing techniques that can be used in performance simply by lifting the bow away from the bowing surface. As no pressure is applied to the hair of the bow while it is held in mid-air the FSRs will not output any signal. However, a performer may still activate these sensors by plucking or pressing the hair of the bow while moving the ESBow in the air.

Performing without a bowing surface also offers another form of bow control. For example, an extended legato stroke could be played indefinitely by drawing the bow through the air horizontally along the Y axis. This stroke could even be performed through a full rotation around the performer. In the same way a series of detached strokes could also be performed by making a series of short interruptions to the bowing action in mid-air.

A series of unorthodox bowing surfaces are used in the work *Kitchen* (A.7). The work features various bowing surfaces commonly found in any kitchen, such as a kettle, cutlery, tap and a fridge door. This is the first of a series of works that will explore a variety of bowing surfaces. Bowing surfaces will be sought in locations of significance to the composer.

A sculpture I have constructed specifically for bowing consists of bent metal rods that resemble a collection of croquet hoops. Each hoop can be used as a dedicated bowing surface and can be combined with other hoops to form multiple bowing surfaces. The sculpture opens up possibilities for new musical works created collaboratively with sculptors.

Bowing can be used in conjunction with other instruments. For example, bowing the stand of a keyboard with the right hand allows the performer to maintain continuous control over the
sound envelope while using their left hand to control pitch using the keyboard. The keyboard stand is bowed because its smooth round edges can be bowed from many angles.

The work *Violin 2.1* (A.6) uses the back of a violin as a bowing surface to manipulate looped samples of violin recordings. The attraction of this surface comes both from the visual and dramatic impact of bowing the violin. Bowing in a conventional playing position retains traditional techniques in less conventional contexts.
6. Epilogue

The implementation of the ESBow design revealed new possibilities for the application of conventional violin bowing technique used with a variety of bowing surfaces. Applications of sensing technologies in the ESBow design support a strong physical connection between the performer and the music. Physical actuation of sound using the ESBow feels more like playing a musical instrument than operating a typical electronic control interface. Research associated with the prototype design revealed areas for future refinements or upgrades of the hardware and functionality of the ESBow.

6.1 Design Observations

The ESBow can be used easily by someone who, like myself, is not an experienced violinist. It offers a level of sophistication that invites the performer to hone their skills and techniques for expressive control and performance. It also allows the relationship between gesture and the sound produced to be explored. This can involve mapping sensor data which completely changes how a performer might interact with the instrument while preserving the intimate connection between performer and instrument. Expressive performance made possible by the ESBow lays the foundation for chamber music based on electronics (2.7) and introduces new possibilities for compositional use. The techniques presented in chapter four provide the base for further exploration by composers and performers.
There are several distinctive features of the ESBow sensor design. As well as providing information on pressure the dual FSR configuration provides the ability to determine the relative position of the point of pressure along the bow (4.5.1). Alternatively, the single FSR configuration (4.8) allows the ESBow to be used with a natural violin (2.1) without any compromise to conventional bowing technique. The accelerometer provides reliable data on the gestural movement and tilt of the bow throughout each stroke. The trackball takes advantage of the uncommitted middle and ring fingers of the bowing hand.

The default PD to MIDI interface (4.4) provides a simple and reliable method to configure the sensitivity of each sensor individually along with the ability to manipulate or combine data streams before they are exported as MIDI. The use of MIDI allows the ESBow to be used with software applications and MIDI hardware devices. Customising data streams in PD allows the ESBow to control any facet of a MIDI instrument in a natural and intuitive way.

6.2 Ongoing Development

Work described in the preceding chapters lays the groundwork for ongoing ESBow design based on enhanced technology. Such designs might include various hardware and software enhancements.

One avenue of exploration would see the development of Arduino code that converted sensor data to MIDI data packets. These would be sent to the computer workstation using a USB transport. Mapping and sensitivity configurations could be specified in Arduino code with changes to settings achieved by loading new code onto the Arduino. This allows changes in
ESBow functionality to be implemented quickly and easily. Such Arduino code implemented as firmware would eliminate dependence on PD. This allows the ESBow to communicate directly with the chosen software application or hardware device. If a performer chooses to work with PD, the default PD to MIDI interface (4.4) could be replaced by a single PD net receive message. As well as simplifying the performer interface this would eliminate the need to separate momentary events from clock pulses (4.4).

A battery powered Arduino worn by the performer would eliminate the need for cables. Wireless communication could be achieved by upgrading the Arduino Diecimila to a Bluetooth Arduino or a microcontroller that uses a wireless protocol such as IEEE 802.11 or an ISM sub-GigaHertz wireless protocol. This would give the performer complete freedom of movement, untethered by cable to a computer or MIDI device. The conversion of sensor data to MIDI could be performed either in the microcontroller or on the computer workstation.

The ESBow design allows sensors to be upgraded without deconstructing the bow. Lighter and more compact sensors are becoming available making it possible to further minimise the weight of electronics added to the natural bow. Such improvements in size and functionality are now easily affordable.

The ESBow could also be updated using additional sensors such as a gyroscope. A combination of gyroscope and accelerometer would deliver greater accuracy in motion control sensing. Another possibility could be the replacement of bow hair with linear position sensing ribbon. This would provide a simple absolute position sensor that would span the entire length of the bow. It would also make it possible to determine the point of contact while using a single FSR configuration (4.8). However, considerable experimentation may be
required before such material could also be used to bow a violin string effectively.

Conventional electronic sensors could also be added to the on-arm components of the ESBow. The addition of rotary potentiometers or toggle buttons to the daughter board could provide a master control system that does not compromise the gestural interface of the ESBow. The twist ties that secure the FSR sensor wires could also be replaced with a detachable material of minimal weight and height, such as miniature Velcro straps.

### 6.3 User Acceptance

The ESBow was designed to be used by musicians from various instrumental traditions. A website to be launched by the end of 2011 will feature detailed instructions on assembling the ESBow. The website will feature text tips and videos to assist musicians develop a foundation in electronic instrument building and design. This will allow the ESBow to reach a potentially worldwide user base. It will also gain exposure to a large audience through public performance.

The ESBow also has a future in ensemble performance. Initial focus will be on the ESBow in collaborative efforts with other electronic performers using instruments of their own design. The inbuilt LEDs in the trackball (C.6) could provide silent cues during performances. This could prove especially useful during structured improvisations. Different coloured LEDs could act as cues between performers indicating changes in section, key, or any other significant moment in the piece. An ESBow quartet is another intention for future collaboration.
6.4 Personal Reflections

The development of the ESBow design prototype has been focused principally on how the design might use bowing with non-conventional bowing surfaces in order to extend the creative potential of my work as a composer. My creative work will focus on continuing exploration of the ES Bow using performance techniques briefly described in this thesis and how these techniques might lead to the creation of an intuitive form of electronic chamber music (2.7).

The ES Bow demonstrates the ongoing development of bowed instruments. Bowing gestures possible using the ES Bow, allow music to be controlled directly by interaction with physical movement. As mobile technologies become increasingly integrated in future implementations of the design, music created using the ES Bow will be able to reach new levels of musical sophistication. These instruments will be able to interact with each other in ways where music becomes the expression of human cooperation realised through the collective action of bowing.
References

Arduino n.d.a, software, Version 0022, SmartProjects, Italy.


Proceedings of the conference on New Interfaces for Musical Expression, Vancouver,
Canada, 26-28 May 2005.

Industries.


http://www.nealfarwell.co.uk/sensors_interactive.html.


Interaction’, Proceedings of the International Computer Music Conference, Havana,

Interfaces for Musical Expression, Genova, Italy, 5-7 June 2008.


the Conference on New Interfaces for Musical Expression, Paris, France, 4-8 June
2006.


http://brainop.media.mit.edu/Archive/Hyperinstruments/creative.html.


O’Connell, n.d., MIDIYoke, software driver.


Puckette, M n.d., Pure Data, software, Version 0.41.4.


Tekscan n.d., ‘*FlexiForce Data Sheet*’, Rev E, Tekscan, Inc.


Appendix A   Compositional Studies for Solo ESBow

This Appendix consists of a series of compositional studies. Each study is a short composed instrument work that demonstrates various aspects of the ESBow’s interface design and the possibilities it presents to the performer. These studies were designed principally for the purpose of allowing a performer to explore new aspects of the ESBow rather than the purpose of public recital.

Discussion relates to the object and approach of each work, how each work was composed and what each work reveals about the ESBow. Details such as the bowing surface used, preparation of data streams, mapping techniques and the structure of the composition are presented together with discussion of the strengths and weaknesses of the various performance techniques, sensors and mapping systems (2.3) used with each composition. The work is illustrated using recordings presented on the DVD-ROM accompanying this thesis and explained with the help of AudioMulch screen shots, PD patches or tables for the JunoD synthesiser. An initial focus on one-to-one mapping systems in studies composed for the ESBow was intended to examine the role and playability of each sensor in performance.
A.1 Traditional Expectations and the ESBow

Audio 01 - 04

ESBow/PD/AudioMulch

The first short series of works use mapping techniques based on those inherent in the performance interface of the natural violin (2.1). This involves many-to-many mapping systems with sensors used in combination to determine performance attributes.

The works focus on performance techniques that would traditionally maintain or avoid stick-slip\(^8\) motion on a stringed surface. As the work is performed with a bow (with no rosin applied to the bow hair) on a non-stringed surface, stick-slip motion will not actually occur in either case. The sensors determine whether the bowing action uses too great or light pressure, and whether the speed of the bow is too slow or fast to properly engage with the string. Each attribute contributes to the dynamic level and timbre of the audio stream in a manner that resembles their natural counterpart in the violin.

---

\(^8\) Stick-slip is the term used to describe the two-phase periodic motion of a bowed string first observed by Hermann von Helmholtz (Smith & Berdahl 2007). As the bow travels across the string it sets the string in motion producing a transverse wave; the bow ‘sticks’ as it is pulled continuously in one direction to a point where it then ‘slips’ in the opposite direction. Both phases alternate for the duration of a single bow stroke. Violinists apply rosin to the hair of the bow to increase the ‘stick’ or traction of the bow on the string.
Audio is derived from a looped bassline to ensure the intentions of emulating mapping techniques are not confused with the intentions of emulating a violin through the physical actuation of a virtual violin. The pitches of the bassline are randomly determined.

The dynamic level of the bassline is actuated by a combination of bow pressure from both FSRs and the velocity of dynamic movement in the Y axis of the accelerometer. The composite signal is reduced by the displacement in the X axis due to tilting. This reflects the natural violin’s dynamic levels which are the result of bow pressure, speed and tilt.
Two effects are applied to the audio stream based on the traditional performance interface of the natural violin. The first stream controls the saturation of a digital distortion effect to represent the coarse timbre produced when a bow is dragged slowly and heavily across a string. This is achieved by increasing a data stream by the combined pressure of the two FSRs when the output rises above a pre-determined figure denoting a heavy bow stroke. The stream also increases when the velocity of the bow in its Y axis is below a set figure. The stream is decreased by a fraction of the displacement of the bow in its X axis due to tilt.

Figure 36: A.1 Dynamics.

Figure 37: A.1 Timbre 1.
The second stream determines the saturation of a delay effect with mapping based on running the bow across the string too quickly and lightly to engage stick-slip motion. This is achieved by adding the velocity of the bow in the Y axis above a pre-determined figure, the displacement of the bow in the X axis and a figure derived from the two FSRs when they drop beneath a minimum value.

![Diagram](image)

*Figure 38: A.1 Timbre 2.*

The trackball is not featured in this series in order to focus on the actuation of traditional performance techniques with traditional mapping systems. While the trackball could be used to perform acts usually associated with the left hand of the performer it was decided to withhold the trackball from the composition to properly observe the ESBow’s behaviour in traditional bowing.

Two recordings were made of this work. In the first recording performance is focused on simple bow strokes that maintain or avoid traditional stick-slip motion (Audio 01). The second recording introduces extended bowing techniques such as jeté and spiccato (Audio 02).
The second work in the series explores the mapping systems opposite to that of the natural violin. This is achieved in PD by adding an *expression* object to each data stream that reverses the output to lower from 127 rather than rise from 0 as shown in Figure 39. The AudioMulch patch did not require alteration.

![Figure 39: A.1 Reversing the output streams.](image)

Two recordings were made of this work. Like the previous work the first recording is focused on simple bow strokes (Audio 03) while the second recording focuses on extended techniques (Audio 04).

The series of works produced simple audio which would be unlikely to be selected for public performance. However, this simplicity provided the ideal basis for a performer to explore traditional performance techniques with the ESBow and their effects on the audio.
Connecting the mappings of various sensor streams offers a more realistic approach than a one to one mapping system. However, simpler and more direct mapping systems can offer different interface and playability techniques.

When performing with mapping systems that actively oppose the traditional mapping systems of a natural violin I tended to focus on techniques that would traditionally avoid stick-slip motion such as extremely slow and heavy bow strokes. Performing bouncing techniques with the opposing mapping system created a similar result in the audio stream as performing them with traditional mapping systems.

I found the second of the two works more satisfying as a performer. Performing the first work seems to ask the performer to maintain a traditional performance in new surroundings where the second work seemingly asks the performer to evade the traditional. While the strokes are exaggerations of traditional strokes they produce untraditional exciting results. I also felt a greater affinity and connection with the ESBow during the performance of the second work.
A.2 JunoD Improvisations in D minor

Audio 05 - 06

ESBow/PD/Roland JunoD Synthesiser

This work is intended to demonstrate the possibilities of using the ESBow to interface with hardware MIDI devices. The creation and manipulation of audio is entirely controlled within a Roland JunoD synthesiser. The computer is used only to prepare and convert sensor data to MIDI format. The ESBow is used to bow the stand of the synthesiser with the right hand while pitches are selected with the left hand. This combines violin and piano performance techniques.

Two works were composed to demonstrate the ESBow with a JunoD synthesiser. The trackball is not featured in either work as they were composed as proof of concept works early in the construction of the prototype.

In the first work (Audio 05) a Juno Lead MIDI instrument is loaded on the JunoD synthesiser. The X and Y axes of the accelerometer are used to manipulate the cutoff and resonance of the instrument. The dynamic level of the instrument is controlled by the combined outputs of the two FSRs. The left hand improvises in D minor.

In the second work (Audio 06) a Juno Lead MIDI instrument is loaded on the synthesiser. The X and Y axes of the accelerometer are used to manipulate the rate and depth of LFO modulation. The dynamic level of the instrument is controlled by the combined outputs of the two FSRs and the position of the instrument in the stereo mix is determined by the relative
position of the point of contact. The left hand improvises in D minor.

These works could be extended to control any number of audio parameters within the JunoD synthesiser. A full list of possible MIDI control options available with the JunoD are provided in Figure 40. Parameters are selected using the relevant MIDI control number. This demonstrates the vast possibility of control offered by the ESBow when used in combination with any MIDI hardware device.

<table>
<thead>
<tr>
<th>Effect</th>
<th>MIDI Control Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modulation</td>
<td>1</td>
<td>Vibrato</td>
</tr>
<tr>
<td>Porta Time</td>
<td>5</td>
<td>Portamento Time</td>
</tr>
<tr>
<td>Volume</td>
<td>7</td>
<td>Level</td>
</tr>
<tr>
<td>Balance</td>
<td>8</td>
<td>The volume balance of lower and upper tones</td>
</tr>
<tr>
<td>Pan</td>
<td>10</td>
<td>Pan</td>
</tr>
<tr>
<td>Expression</td>
<td>11</td>
<td>Level</td>
</tr>
<tr>
<td>Portamento</td>
<td>65</td>
<td>Portamento Switch</td>
</tr>
<tr>
<td>Sostenuto</td>
<td>66</td>
<td>Holds the sound of the key being pressed</td>
</tr>
<tr>
<td>Soft</td>
<td>67</td>
<td>Softens the tone</td>
</tr>
<tr>
<td>Resonance</td>
<td>71</td>
<td>Tone Filter Resonance</td>
</tr>
<tr>
<td>Release Time</td>
<td>72</td>
<td>Tone Envelope Release Time</td>
</tr>
<tr>
<td>Attack Time</td>
<td>73</td>
<td>Tone Envelope Attack Time</td>
</tr>
<tr>
<td>Cutoff</td>
<td>74</td>
<td>Tone Filter Cutoff</td>
</tr>
<tr>
<td>Decay Time</td>
<td>75</td>
<td>Tone Envelope Decay Time</td>
</tr>
<tr>
<td>LFO Rate</td>
<td>76</td>
<td>Tone LFO Rate</td>
</tr>
<tr>
<td>LFO Depth</td>
<td>77</td>
<td>Tone LFO Depth</td>
</tr>
<tr>
<td>LFO Delay</td>
<td>78</td>
<td>Tone LFO Delay</td>
</tr>
<tr>
<td>Cho Send Level</td>
<td>93</td>
<td>Chorus Send Level</td>
</tr>
<tr>
<td>Rev Send Level</td>
<td>91</td>
<td>Reverb Send Level</td>
</tr>
<tr>
<td>MFX Parameter1</td>
<td>12</td>
<td>The parameter specified by Multi-effect Control 1</td>
</tr>
<tr>
<td>MFX Parameter2</td>
<td>13</td>
<td>The parameter specified by Multi-effect Control 2</td>
</tr>
</tbody>
</table>

Figure 40: A.2 JunoD MIDI control table.
A.3 Sound Source Series

Audio 07 - 10

ESBow/PD/AudioMulch/Violin/Roland JunoD Synthesiser

This was the first series of works for the single FSR configuration of the ESBow (4.8). Like the JunoD improvisations in D minor the series was composed during early construction as a proof of concept work and does not feature the trackball.

The series explores how sound source influences performance with the ESBow. All variables other than the sound source are mirrored between works in the series. Fine tuning the mapping sensitivities to the sound source would improve the playability of each work. However, this is specifically avoided in order to enable comparison.

The sound sources are: a sine wave oscillator of 280Hz (Audio 07), a white noise generator (Audio 08), an electric violin (Audio 09), and a JUNO-D synthesiser with a shakuhachi MIDI instrument loaded (Audio 10). The first two works in the series are bowed using the neck of a square based bottle on its side. The work for violin is bowed in a traditional violinist pose with the violin bowed with the right hand and fingered with the left. The final work for synthesiser features the keyboard stand bowed with the right hand while the left hand uses the keyboard.
In each work the single FSR output is used to determine the dynamics of the audio. The data stream is also split to create a second stream as shown in Figure 42. This second data stream is used to create an overdrive effect in AudioMulch using a pair of *DigiGrunge* objects. The sensitivity of the stream is increased and its origin point reduced below zero. This ensures the second stream only affects the audio stream after the dynamics reach a certain level.
Accelerometer data is used to determine three parameters of a granulator object in AudioMulch. The X axis controls the saturation of the granulator effect on the audio stream. The Y axis determines the stereo pan of the affected audio. The Z axis shifts the pitch of the affected audio.

This series of works successfully proved the capabilities of the single FSR configuration. The third work of the series was also the first work composed for the ESBow and violin. Although it demonstrates the ability to use the ESBow with a violin, it barely scratches the surface of what is possible. As the focus of the thesis was placed on the dual FSR configuration for non-stringed surfaces the single FSR configuration and performance with a violin are not featured again in this Appendix.
A.4 Without a String to Stand On

Audio11

ESBow/PD

This was the first work to be performed without a bowing surface. It demonstrates possibilities available through the accelerometer when not restrained by a dictated surface. To emphasise this, the use of a bowing surface diminishes the audio level of the work.

Movement and tilt in the Y axis of the accelerometer progresses a note along the steps of a scale from tonic to octave. The scale is a natural minor scale by default and switches to a major scale when the button of the trackball is held. The length of each note is determined by tilt in the X axis. This is achieved using a metronome object in PD. Notes are produced on an oscillator while the bow is held upright and switch to a saw tooth generator when the ESBow is held upside down. The combined output of the FSRs reduces the dynamic level of the work. The vertical axis of the trackball determines the tonic of the minor and major scales. The horizontal axis determines the pitch of a second note by increasing the interval between it and the original pitch from unison to octave doubling.
Figure 43: A.4 Selecting a scale with the trackball.

Figure 44: A.4 Harmonising and selecting output.
One technique discovered during the performance of this piece was the ability to manipulate the pressure placed on the hair of the bow with the thumbs of the performer (5.11). The performer’s thumbs could also be used to modify the position of the pressure between the two FSRs. However, relative positioning was not used in the work.
A.5 Four Rows of Twelve

Audio 12

ESBow/PD

This work was composed to explore the use of the trackball axes as four separate counters. Four tone rows were developed using a twelve sided die. The twelve notes ascending from A below middle C were assigned a number between one and twelve and arranged in each row according to the order their number was rolled. Each tone row was then developed into four versions; the original, retrograde, inverted, and inverted retrograde. Each version was linked to a direction on the trackball. The four original tone rows were linked to the upward direction; the four retrograde tone rows were linked to the downward direction; the four inverted tone rows were linked to the left direction; and the four inverted retrograde tone rows were linked to the right direction.

Figure 45: A.5 User interface.
When the state of any direction is altered through physical actuation the audible pitch is progressed along the relevant tone row. Each direction can progress eleven times before reaching the end of the tone row.

![Diagram of tone row operative]

*Figure 46: A.5 The tone row operative.*

Each time the trackball is clicked the tone rows are reset. After a tone row is played through twice, clicking the trackball loads a new set of tone rows into the four directions. Clicking the trackball after the repeat of the fourth set of tone rows ends the work by fading the audio to silence.
Figure 47: A.5 Select as a structural device.

A section can end when a single tone row has reached its final note or when all four rows have reached their end. All decisions as to which direction to actuate and when to actuate are left to the performer. The performer can therefore decide to only progress a single direction along its full length and start a new section without progressing any other direction, or sustain a set of tones without a change in pitch for any length of time.

The tilt of the ESBow in all four directions determines the balance of the four tone rows in the output. If held upright the four tone rows output at equal proportions. The work is intended to be performed with the ESBow held in front of the performer so the bow points towards the left of the performer. In this way tilting down to the left increases the mix of the left or inverted tone row and decreases the mix of the right or inverted retrograde tone row. Tilting down to the frog increases the mix of the right tone row and decreases the mix of the left tone row. Tilting the ESBow towards the performer increases the mix of the upward or original tone row and decreases the mix of the downward or retrograde tone row and tilting the ESBow away from the performer acts in the opposite.
The positions of the paired tones in the stereo mix are determined by the relative position of the point of contact. The positions of the paired tones oppose each other so that as one pair is directed to the left speaker, the other is directed to the right speaker.

Each repeat focuses on a different aspect of the work such as the balance and panning of the tone rows, bowing technique, the rate of tone row progression, and beating between pitches.

The work demonstrates the effectiveness of slow subtle manipulations of sound using the ESBow. The work also demonstrates the ease in which the trackball can be used to control the structure of a composition. As the balance of the tone rows relies on the tilt of the
accelerometer a significant effect is achieved by conducting a minor jitter such as a tremolo motion in the bow along the X or Y axis.
A.6  Violin 2.1

Audio 13

ESBow/PD/AudioMulch

One of the objectives of this work was to play with the audience’s perception of the ESBow. To achieve this, the ESBow is used to bow the back of a violin held upside down in an otherwise traditional violinist pose. The ESBow simultaneously controls a solo instrument and its accompaniment. Audio for both audio streams is sourced from six short pre-recorded samples of violin noises. Each sample has been stretched or contracted without pitch protection. The trackball select progresses through the samples using a pair of AudioMulch matrix objects. A matrix object allows a user to rapidly remap connections between inlets and outlets. When the work has progressed through the final sample the trackball select triggers a closing fadeout on the master mixer.

Figure 49: A.6 Matrixed samples.
The solo instrument consists of the sampled audio running through a series of effects and mixers consisting of a *digigrunge* effect, *granulator*, *delay*, *stereo gain mixer* and a *panning mixer*. 

---

**Figure 50:** A.6 Matrix object.

**Figure 51:** A.6 Trackball select as a structural device.
The dynamics of the solo instrument are determined in the *stereo gain mixer*. The signal is derived from the combined output of the two FSRs and the velocity of the ESBow along the Y axis. Using the velocity of the ESBow ensures axis output only occurs due to movement and is not influenced by the static tilt of the ESBow. This stream is primarily determined by the FSR output. This provides a stable output while still retaining a natural feel.

**Figure 52: A.6 Solo instrument.**

**Figure 53: A.6 Dynamics.**
The tilt of the Y axis transposes the pitch of the audio in the *granulator* effect. The stream is modified to provide a minimum and maximum value for a range of possible transposition values as shown in Figure 54. The X axis determines the saturation of two effects on the solo instrument. If tilted towards the performer the saturation of a *delay* effect is increased. If tilted away from the performer the saturation of a *digigrunge* effect is increased. When held upright neither effect influences the audio stream.

![Figure 54: A.6 Transposition range of the Y axis and dual effects of the X axis.](image)

The relative position of the point of contact is used to determine the position of the solo instrument in the stereo output. The stream is modified to provide two reference points a short distance from each other. This allows the original left and right channels of the solo instrument to retain a degree of separation during panning.

![Figure 55: A.6 Ranged panning.](image)

The accompaniment consists of the looped audio samples running through a pair of *five pitch comb filters*. The pitches of the two filters are cycled through four preset chords in a continuous loop. This is timed and activated in PD. Following the *five pitch comb filters* the
audio is split into three streams. One stream progresses directly to the *stereo mixer*. The second stream runs through a *delay* object and outputs to the *stereo mixer*. The third stream runs through a *pulse comb*. This stream is further divided with one stream output to the *stereo mixer* and the other to a second *delay* object and then output to the *stereo mixer*. The mix of the four streams in the background of the work is determined by the position of a cursor on a *metasurface*. The cursor is controlled by the two axes of the trackball.

![Diagram](image-url)

*Figure 56: A.6 Accompaniment.*
This was the first work to use the ESBow to simultaneously control two instruments, the solo and accompaniment audio streams. It is also the first work to simultaneously monitor the velocity and tilt of a single axis in order to control two separate data streams. It also demonstrates the ability to use one data stream to provide a minimum and maximum value for a parameter as is performed for the Y axis tilt and relative position of the point of contact.
A.7 Kitchen

Audio 14

ESBow/PD/AudioMulch

This work demonstrates the use of the ESBow with various bowing surfaces. In principle the performer may choose to bow any object found in the kitchen. The objects bowed for the recording of this work on the DVD-ROM include a kettle, various pieces of cutlery, a tap and a fridge door. Audio for the recording was sourced from four of the six looped violin recordings used in Violin 2.1 (A.6).

The work also explores the ability to remap the ESBow during performance. Remapping occurs with each change in bowing surface. The trackball is the only sensor where the mapping system remains unchanged. The vertical axis of the trackball controls which sample is used as a sound source. The horizontal axis of the trackball controls which effect is applied to the sound source. These selections are made using a pair of AudioMulch matrix objects as shown in Figure 59. The select button of the trackball triggers a random change in the mapping of the analog streams. This is achieved by swapping MIDI control numbers in PD using an urn object as shown in Figure 61. The urn object randomly outputs a series of integers in a pre-defined range.
Figure 59: A.7 User interface.
Each time the performer changes bowing surface they use the select function to initiate a mapping change. Randomising mapping systems forces the performer to explore each new bowing surface as a new instrument.
Holding the trackball select for two seconds rather than clicking it as a momentary switch initiates a fadeout of the master volume (4.7.3).

![Diagram showing trackball select timing](image)

*Figure 62: A.7 Timing the trackball select.*

As this work requires performers to explore the relationship between gestures and sound with each new bowing surface it is very effective in assisting the performer to gain a deeper understanding of the gestural interface of the ESBow. Future works could be expanded by using various locations as the foundation for the work. Sounds from the location could also be recorded to be used as the original sound sources. In *Kitchen* for example, the violin recordings could be replaced by recordings of a blender, whistling kettle, running water or dicing vegetables. The mapping of sound sources and effects in the matrix objects could also be randomised in future works in the series.
A.8 Composing with the ESBow

The physical relationship between performer and instrument developed over time spent with the ESBow. Initial works composed for the ESBow used an approach similar to that which I have used with other MIDI controllers. This involved considering the physical parameters of the ESBow and which audio parameters would be the most exciting to control. This method was appropriate to composing demonstrative works for the ESBow however it did not use the ESBow to its full capacity as an instrument for composition.

When composing with a natural instrument I often approach a work with the instrument in hand and explore its interface through experimentation. Early works for the ESBow encourage exploration during performance however exploration was not a part of the compositional process.

In later compositions I would use a key idea as the foundation for a work which would be explored during its composition. Techniques and ideas that develop during the compositional process can then be used to extend the composition in new areas. This approach attempts to find the natural connection between the instrument and music within a specific work with as few preconceptions as possible. An example of this is the work *Without a String to Stand On* (A.4). Composition of the work initiated with the key idea of performing without a bowing surface. On experimentation during the composition process I began to use my thumbs on the hair of the bow. The ability to manipulate the audio through the pressure and placement of my fingers was then added into the composition.
After performing compositions with various mapping systems I discovered my personal preference is for less traditional approaches. This is reflected in the majority of compositions for the ESBow thus far. My preference for non-traditional bowing and mapping techniques stems from the physical connection with the ESBow during such performances. This connection feels strongest when I perform with the ESBow in front of my body using a combination of slower subtle movements and larger strokes based on traditional techniques. This connection would not be the same for other performers and each performer would quickly find their own favoured performance techniques, style and mapping systems.

The compositions presented in this Appendix merely hint at the vast possibilities available with the ESBow. The studies demonstrate some of these possibilities; however, an inclusive list of all possibilities would not be possible in one thesis, or indeed one lifetime.
Appendix B    Miscellaneous Diagrams and Listings

This Appendix contains descriptions of diagrams and techniques discussed in chapter four with additional detail.

B.1    Arduino Code

The following code is used to create multiplexed packets of sensor data within the Arduino to be received and decoded in PD. The code is a modified version of that provided by the creator of the Arduino2PD PD patch (Arduino2PD n.d.). The original version was based on the code for the SimpleMessageSystem patch and uses the library created for use with SimpleMessageSystem (SimpleMessageSystem n.d.). The ESBow code limits data packets to only include data for used inputs of the Arduino. This doubles the bandwidth speed of the connection.

```cpp
#include <SimpleMessageSystem.h>

/* Analog/Digital inputs to PD trigger
 * -----------
 * send serial values to PD to trigger something
 */

char firstChar;
char secondChar;

void setup()
{
    Serial.begin(115200);
}
```
void loop()
{

  if (messageBuild()) { // Checks to see if the message is complete
    firstChar = messageGetChar(); // Gets the first word as a character

    if (firstChar == 'r') { // Checking for the character 'r'
      secondChar = messageGetChar(); // Gets the next word as a character
      if (secondChar == 'd') { // The next character has to be 'd' to continue
        messageSendChar('d'); // Echo what is being read

        for (int i=0;i<=5;i++) {
          messageSendInt(analogRead(i)); // Read analog pins 0 to 5
        }

        for (int m=2;m<=6;m++) {
          messageSendInt(digitalRead(m)); // Read digital pins 2 to 6
        }

        messageEnd(); // Terminate the message being sent
      }
    }
  }
}

B.2 ESBow Daughter Board

Figure 63 illustrates the wiring of the daughter board that routes signals from the sensor ribbon cables of the ESBow to the appropriate inline sockets of the Arduino. Blue lines represent the copper tracks of the veroboard. Red lines represent physical wires soldered to the veroboard. Grey squares represent connections between wire and copper track. The copper track has been cut between each connection along either side of the board to isolate the input and power pins.
Figure 63: Daughter board schematic.

The main features of the daughter board shown in Figure 63 are:


[C] Power and ground pins of the Arduino.

[D] Ribbon cable for the trackball and FSRs.

[E] Ribbon cable for the accelerometer.

[F] Current limiting resistors for the two FSRs.
B.3 Pure Data Patches

This section provides a detailed breakdown of the default PD to MIDI interface along with diagrams of the input and output sub-patches and original Arduino2PD patch.

B.3.1 Pure Data to MIDI Interface

The default PD to MIDI interface (4.4) is simplified by dividing functions into sub-patches. An earlier single canvas version of the patch illustrated in Figure 64 will be used to discuss the functions of the interface.
Figure 64: Single canvas PD to MIDI interface.

The main features of the single canvas input to MIDI PD interface shown in Figure 64 are:

[A] The *toggle* object starts or stops the patch reading the Arduino.

[B] The *metro* object sets the clock rate of sensor polling. It is set to read sensor output every twenty milliseconds, i.e. fifty times a second.
These three objects open and close the communications port that the Arduino is connected to and sets the baud rate. In this case it is the third port with a baud rate of 115200.

The *unpack* object is a de-multiplexer. This separates multiplexed data packets from the Arduino into individual data streams for each sensor. The order of the unpacked streams is relevant to the hardwiring of the physical inputs of the Arduino. The order of the X and Z streams of the accelerometer and the Left and Right streams of the trackball are swapped in the PD interface for the ease of the user.

These are the analog data streams.

These are the digital data streams.

*Ctlout* objects convert each stream to 7-bit MIDI control change messages and outputs the stream for use with other software applications and hardware devices. PD uses MIDI channel 1 by default so each object only needs to specify the MIDI control number. *Ctlout* objects also limit each stream to integers within the MIDI range of 0-127.

The row of *division* objects scales analog values from 0-1023 to MIDI values of 0-127. The difference in divisor between the FSR and accelerometer streams is due to the 5V and 3.3V power supplies and subsequent output limit of each sensor.

*Subtraction* and *multiplication* objects prepare the X and Y axes of the accelerometer so tilting will cover the full MIDI range. To read the full dynamic range of the sensor these objects can be bypassed as illustrated in the Z axis.
[J] The row of *slider* objects provides a quick visual reference to analog and digital sensor outputs. This decreases the intensity of any visual dependency during performance or testing.

[K] The object underneath K is a binary inverter. This is necessary to express the state of the trackball select which is naturally high at rest.

[L] The trackball directional inputs trigger a *bang* object when a change in state is detected by a *sel* object (4.4).

[M] *Float and addition* objects tally the number of times the *bang* object is triggered and outputs this to the number box beneath (4.7.3).

[N] Both streams for the vertical and horizontal axes of the trackball are combined to output a single value for each axis. A *bang* object is used to ensure the combined stream is updated when either direction is actuated.

[O] *Multiplication* objects are used to increase the sensitivity of data streams and can be adjusted to suit the performer.

[P] This column of objects controls the multiplier of the trackball data streams. By default this number is two but can be modified or actively controlled by another sensor stream during performance.

[Q] This row of *message* objects allows the user to set a starting point for the trackball axes before a performance. These will set the MIDI output at 0, 32, 64, 95 or 127 which represent 0, 25, 50, 75, and 100%, of the full MIDI range. The streams are divided by the multiplier applied to the trackball data streams to ensure they will set the correct position.
Sensor data streams can be output in any MIDI channel and control number scheme. The following table depicts the default control number of each stream. These are flexible and can be altered for specific pieces.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>MIDI Control Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSR (Frog)</td>
<td>1</td>
</tr>
<tr>
<td>FSR (Tip)</td>
<td>2</td>
</tr>
<tr>
<td>Accelerometer X</td>
<td>3</td>
</tr>
<tr>
<td>Accelerometer Y</td>
<td>4</td>
</tr>
<tr>
<td>Accelerometer Z</td>
<td>5</td>
</tr>
<tr>
<td>Trackball Select</td>
<td>6</td>
</tr>
<tr>
<td>Trackball Vertical Axis</td>
<td>7</td>
</tr>
<tr>
<td>Trackball Horizontal Axis</td>
<td>8</td>
</tr>
<tr>
<td>Relative Position</td>
<td>9</td>
</tr>
</tbody>
</table>

Figure 65: Default sensor MIDI control numbers.

B.3.2 Pure Data Input to MIDI Sub-Patches

The following two diagrams depict the sub-patches of the default PD input to MIDI interface (4.4).

---

9 Relative position sensing is not included in the default Pure Data input to MIDI Interface but is typically designated the MIDI control number of 9 when used in composition.
B.3.3 Arduino2PD and SimpleMessageSystem

The PD input to MIDI interface was expanded from the Arduino2PD patch shown in Figure 68 (Arduino2PD n.d.). The Arduino2PD patch was based on SimpleMessageSystem (SimpleMessageSystem n.d.). These patches were designed to receive analog and digital data from the Arduino. The ESBow patches were expanded from the Arduino2PD patch as it featured a simpler method for unpacking digital data than the SimpleMessageSystem patch.
Figure 68: Arduino2PD.
B.4 Pure Data Examples

This section provides detailed descriptions of techniques discussed in chapter four.

B.4.1 Relative Position Sensing

Figure 69: Relative position sensing.

Figure 69 demonstrates the technique used to determine the point of contact along the length of the bow relative to the position of the two FSRs (4.5.1). The output signal from the FSR at the tip end of the bow at [B] is subtracted from the output signal from the FSR at the frog end of the bow at [A]. This is performed by the subtraction object at [C] and results in a number between +/- 127 at [D]. 127 is added at [E] to ensure a positive output between 0 and 255 at [F]. This is divided by 2 at [G] to shift the existing range to the MIDI range of 0 to 127 at [H]. The horizontal slider at [I] demonstrates the position between the two FSRs.
Calibration ensures each FSR represents opposing ends of the MIDI range. To calibrate the positioning sensor the user places pressure at the FSR at the tip end of the bow and notes the output at [D]. This will be a negative number and is substituted in the number box at [E] as a positive number. Pressure is then placed at the FSR at the frog end of the bow. The output at [F] will indicate the highest point in the available range. If this number is lower than 127 it is divided into 127 and the result substituted as a multiplier at [G]. If the number is higher than 127 it is divided by 127 and the result substituted as a divisor at [G]. The final output at [H] should now provide the full MIDI range of 0 to 127. This sensor needs to be recalibrated every time the FSRs are moved to a new bow, the position of the two FSRs along the length of the bow is changed, or the foam mount underneath the FSRs are changed.

This process can be simplified into a single expression object in PD. The process is represented by the equation:

\[ \frac{f_1 - f_2 + f_3}{f_4} \]

In relation to Figure 69 \( f_1 \) is the output of the FSR at the frog end of the bow at [A], \( f_2 \) is the output of the FSR at the tip end of the bow at [B], \( f_3 \) is the number substituted in the number box at [E], and \( f_4 \) is the range modifier substituted at [G]. This equation works on the assumption that \( f_4 \) is a divisor. The equation must be modified accordingly if \( f_4 \) is a multiplier.
B.4.2 Displacement of an Axis

Figure 70 demonstrates the method used to monitor the displacement of an axis (4.6.3). The slider at [A] is the axis data stream. 63.5 is subtracted from this at [B] so the stream will output zero at [C] when the bow is upright. The moses object at [D] splits the stream into positive and negative outputs. Negative numbers are multiplied by negative one at [E] and the result output to [F]. Positive numbers are output directly to [F]. This ensures the number box at [F] will represent the displacement of the axis from rest in either direction as a positive number. The stream is multiplied by 2 at [G] to shift the range of the output from 0 - 63.5 to the MIDI range of 0 - 127 at [H].
B.4.3 Displacement to actuate two streams

![Diagram showing the process](image)

Figure 71: Monitoring the displacement of an axis to actuate two control streams.

Figure 71 demonstrates the method used to monitor the displacement of an axis in order to control two separate data streams (4.6.3). The objects between [A] and [B] split the data into positive and negative outputs at [C] as described in B.4.2. The positive data stream in the right hand column is multiplied by two at [D] to shift the range of the output to the MIDI range of 0 - 127 at [E]. The negative data stream in the left hand column is multiplied by negative two at [D] to shift the range of the output to the MIDI range of 0 - 127 at [E]. The bang object and message box underneath [F] are clocked every twenty milliseconds to ensure whichever stream is inactive at [C] is reset to zero. This process can be simplified into a single expression object as shown in Figure 72.
B.4.4 Expression Object

Figure 73 demonstrates the use of an expression object to split a data stream into five separate streams (4.6.3). The output stream at [A] is separated into five separate streams at [C] based on the Boolean statements in the expression object at [B]. These compare the variable float ($f1$) from [A] to defined parameters. If the condition is true the variable data stream is sent to the corresponding outlet at [C]. If the condition is false the corresponding outlet produces a zero signal.

Each line in the expression object is a Boolean statement comprised of three basic sections separated with commas. The first section defines the parameters that the variable is compared
to. The second section dictates what will occur should the variable be true to the parameters. The third section dictates what will occur should the variable be false to the parameters. Hence a line reading:

\[
if ($f1 > 25 \&\& f1 <= 50, f1 * 2, 0);
\]

Can be translated: if the input is greater than twenty five and less than or equal to fifty, the stream will be multiplied by two. If it is less than twenty five or greater than fifty the outlet will produce the number zero. The \textit{expression} object shown in Figure 73 will split the stream at specified intervals without modification to the input stream and output zero to all inactive streams.

\section*{B.4.5 Velocity of an Axis}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{image.png}
\caption{Determining the velocity of an axis.}
\end{figure}

Figure 74 demonstrates the process used to obtain the velocity of an axis (4.6.3). The axis output at [A] is delayed using a \textit{pipe} object at [B]. The length of this delay is specified within the object (100 milliseconds). The original and delayed outputs are compared by an \textit{expression} object at [C]. The \textit{expression} object determines the displacement which has occurred in the axis during the period specified at [B]. The \textit{expression} object will then process the result in one of two ways depending on whether the displacement is positive or
negative. Each process results in a positive output at [D] representing the velocity of movement in the axis.

**B.4.6 Trackball Counter and Looped Sequence**

![Figure 75: Trackball counter and sequence.](image)

Figure 75 demonstrates the technique used to tally the number of times the trackball is selected to progress along a looped sequence (4.7.3). The trackball state is differentiated at [A] (4.4). The float object at [B] stores a number that is increased by the operative at [C] each time it is triggered by a change in trackball state. The result is output from the float object to the number box at [D]. 0.5 is added to this number at [E]. This ensures the trackball only activates at [F] when the trackball is selected and not when it is released. The sel object at [F] compares the input stream to each number stated in the object and triggers any outlet that is true. The eleventh outlet is triggered when the input is not true for any number stated in the sel object. This includes all non-integer half steps such as those encountered when the trackball select is released. The sel object at [G] receives this stream of numbers and
compares them to 11. If true this will trigger the *bang* object underneath. This triggers the *zero* object at [H] to reset the *float* object at [B] and restart the sequence.

### B.4.7 Demonstration Video 12

![Figure 76: Chord sequence.](image)

Figure 76 shows the patch used in video 12 on the DVD-ROM to progress through a series of chords (4.7.3). A looped sequence is used between [A] and [B] (B.4.6). The number box at [D] is derived from the output of the FSR at the frog end of the bow\(^{10}\). The object at [E] splits

\(^{10}\) For clarity the video example uses a set pitch of 440 Hertz.
this stream into three outputs. As the sequence progresses it triggers a pair of numbers at [C] which are loaded into the number boxes at [F]. These are added to the latter two streams at [G] and output at [H]. The mtof objects at [I] convert the figures from MIDI note numbers to frequencies which are applied to oscillators at [J] and output to the speakers at [K]. In this way the pitch described by the FSR is harmonised by the interval parameters set at [C] and [F]. The diagram depicts a minor third triad which is used at the start and end of the chord progression.

B.4.8 Monitoring Trackball Select Duration

Figure 77: Monitoring the duration the trackball is selected.

Figure 77 demonstrates the technique used to activate secondary triggers by holding the trackball for a specified length of time (4.7.3). When the trackball is selected and held at [A] the metro object at [C] starts a progressive count using the float and addition objects at [D] and [E]. When the trackball is released the count is halted and reset to zero by the message object at [B]. If the count at [F] reaches the number specified in the sel object at [G] before being reset it will trigger the bang object at [H].
Appendix C  The Evolving ESBow

This Appendix discusses the development of the ESBow from its initial design to the final design of the project. It details all significant modifications and provides the necessity for each change. Each sensor is discussed in a separate sub-section.

C.1 Arduino Code

The following code was used with the original Arduino prototype design. It was modified from the code provided with the Arduino2PD PD patch (Arduino2PD n.d.) in order to place internal pull-up resistors on the digital inputs for the original trackball (C.6). The code reads all analog and digital sensors of the Arduino. This was modified in the code for the final prototype design to only read used sensors and improve performance speed (B.1).

```
#include <SimpleMessageSystem.h>

/* Analog/Digital inputs to PD trigger
   * ------------
   * send serial values to PD to trigger something
   */

char firstChar;
char secondChar;
int SEL = 2;
int South = 3;
int North = 4;
int East = 5;
int West = 6;

void setup()
{

```
pinMode(SEL, INPUT);
digitalWrite(SEL, HIGH);
pinMode(South, INPUT);
digitalWrite(South, HIGH);
pinMode(North, INPUT);
digitalWrite(North, HIGH);
pinMode(East, INPUT);
digitalWrite(East, HIGH);
pinMode(West, INPUT);
digitalWrite(West, HIGH);
Serial.begin(115200);
}

void loop()
{

if (messageBuild()) { // Checks to see if the message is complete
    firstChar = messageGetChar(); { // Gets the first word as a character

    if (firstChar = 'r') { // Checking for the character 'r'
        secondChar = messageGetChar(); // Gets the next word as a character
        if (firstChar = 'd') // The next character has to be 'd' to continue
            messageSendChar('d'); // Echo what is being read

        for (int i=0;i<=5;i++) {
            messageSendInt(analogRead(i)); // Read analog pins 0 to 5
        }

        for (int m=2;m<=12;m++) {
            messageSendInt(digitalRead(m)); // Read digital pins 2 to 12, 13 is onboard LED on Arduino NG
        }

        messageEnd(); // Terminate the message being sent
    }
    
}
}
C.2 The ESBow Daughter Board

The daughter board to route sensor data to the inline sockets of the Arduino was modified throughout the project as sensor hardware was updated. Figure 78 shows the wiring of the original daughter board for the Arduino microcontroller. Blue lines represent the copper tracks of the veroboard. Red lines represent physical wires soldered to the veroboard. Grey squares represent connections between wire and copper track. The copper track has been cut between each connection along either side of the board to isolate the input and power pins.

![Figure 78: Original daughter board schematic.](image)

The main features of the daughter board shown in Figure 78 are:


[C] Power and ground pins of the Arduino.

[D] Ribbon cable for the trackball.

[E] Ribbon cable for the two FSRs.

[F] Ribbon cable for the accelerometer.

[G] Current limiting resistors for the two FSRs.

 Modifications to the daughter board included moving the limiting resistors for the FSRs and shaping the board to occupy less surface area. The analog input between the two FSRs was grounded (C.4) and the sensors included in the final prototype allowed the trackball and FSR ribbon cables to be joined at the daughter board (B.2) rather than separated as shown at [D] and [E].

C.3 Pure Data

The PD to MIDI interface (4.4) was consistently updated throughout the project. This included modifications to allow for changes in sensor hardware and the addition of sensitivity and optimisation controls. The original PD to MIDI interface is shown in Figure 79.
The main features of the original PD to MIDI interface shown in Figure 79 as compared to the final interface (B.3.1) are:

[A] All objects from [A] to [B] have remained unchanged from the original PD to MIDI interface with the exception of the metro and unpack objects. The forty millisecond clock rate of the metro object was changed to twenty milliseconds in the final interface to improve bandwidth speed. This was performed in conjunction with a modification to the Arduino code (B.1) and unpack object.
The *unpack* object was simplified in the final interface by removing outlets for unused inputs. The outlet streams were also altered for hardware design changes. The original interface does not include a grounded input between the two FSRs (C.4) or the necessity to rearrange the streams of the X and Y axes of the accelerometer and horizontal axis of the trackball (B.3.1).

The scaling of data information to MIDI output range had not been optimised for the accelerometer streams in the original interface. This includes both the 3.3V reference and the lack of optional scaling for tilting in an axis (4.4).

*Ctlout* objects are simplified in the final patch by removing message boxes that set the MIDI channel number. This relies on PD using MIDI channel one by default. If another MIDI channel is sought it can be expressed within the *ctlout* object.

Slider objects of the analog streams are not inserted into data streams in the original interface. They act only as a visual reference and do not limit the output of the streams. Slider objects of the digital streams have not changed.

High to low conversion is contained in each digital data stream. This was due to the internal pull-up resistors required with the original trackball (C.6). This was not necessary for the directional inputs of the “Blackberry” trackball, but is still used on the input for the trackball select (B.3.1).

The ability to read each actuation of the digital streams as a separate event using a *sel* object (4.4) was not applied in this early patch. This created a visual dependency to ensure each axis would not trigger an infinite loop when actuated. Other additions to the digital streams that are missing from this early patch are the ability to control the sensitivity of each stream and the ability to set an origin point for each axis (B.3.1).
C.4 Force Sensing Resistors

The FSRs were initially mounted on light foam which rapidly compacted under the pressure of bowing. The foam was mounted on either side of a solid foundation in an unsuccessful attempt to slow compaction while maintaining the safety of the bow stick. Manually compacting foam before mounting slowed the rate of further compaction but offered a less stable mount. The light foam was replaced with foam dense enough to resist compaction in a short period of time while posing no threat of damage to the bow stick (4.5.1).

Initial tests with the two FSRs revealed abnormalities in the output signals. The first FSR provided a linear result that correlated to the pressure applied to the sensor. However, actuating the first FSR also impacted on the output of the second FSR. A second series of tests were conducted with various setups of FSRs and rotary potentiometers. The rotary potentiometers provided independent outputs. However, the outputs of each FSR was affected by the preceding analog input on the Arduino, ie an FSR on analog input two would be affected by the data on analog input one. This was also true of the open analog inputs not connected to any sensor or ground connection. The interference introduced an error margin of approximately five percent. This was resolved by separating the FSR inputs and grounding the input that separates them (B.2).
C.5 Accelerometer

The MMA7260 tri-axial accelerometer (4.6.1) was used throughout the project. However, a different breakout board for the accelerometer was used at the start of the project. The ability to manually set the sensitivity of the accelerometer using jumper pins and the automatic bypassing of the sleep function were not contained in this breakout board. Before the later breakout board had become available I had recognised the necessity for these features and constructed an intercept board to employ them with the original accelerometer breakout board. The original breakout and intercept boards are shown in Figure 80. The jumper pins on the intercept board connect power to the pins. The jumper pins for the updated breakout board grounds the pins which are natively high in the breakout board. A further provision in the new breakout board is a ‘voltage in’ pin that powers the accelerometer using a 5V power supply. This is not necessary in the prototype design as a suitable 3.3V power source is available from the Arduino.
A jumper connection was also used to ground the Z axis. This was used to test the impact of the accelerometer on the input of the first FSR (C.4). As the accelerometer was powered with 3.3V the output range of each axis was limited to 66% of the available analog range. The resulting interference was similarly limited below the previous five percent error margin and did not impact on the playability of the FSR. The ability to ground the Z axis using a jumper connection was discarded when the accelerometer was upgraded due to its impractical nature on the new board.

Aside from the ease of including these features within a single board, the decision to upgrade the breakout board of the accelerometer was ultimately based on the reduced weight and size of the new board. A comparison of the two breakout boards is shown in Figure 81.
C.6 Trackball

The original trackball for the ESBow project was a Cannon miniature trackball with momentary select shown in Figure 82. This was the same trackball used in the preliminary MicroCV design (3.3). This trackball features two digital inputs for each axis and one digital input for the momentary select. Internal pull-ups on each digital input stream were provided in the Arduino (C.1). A separate ground line was also necessary for each axis and the select (C.2). As the trackball was rolled in any direction, the ball progressed along a series of haptic steps.
When a “BlackBerry” trackball became available it was comparison tested with the existing trackball\textsuperscript{11}. The “BlackBerry” trackball rolls smoother in all four directions. This is seemingly due to the absence of apparent steps along each axis. The “BlackBerry” trackball is also near silent unlike the cannon trackball which has a faint but audible click for each step progressed. This click would not be loud enough for an audience to hear but may prove distracting to a performer playing on the audible limit of hearing. The “BlackBerry” button is stiffer than the previous trackball but is once again quieter. The “BlackBerry” trackball also features a simpler wiring process without the requirement of internal pull-up resistors on each digital input stream. This led to the decision to replace the original cannon trackball with the “BlackBerry” trackball.

The “BlackBerry” trackball also features four coloured LEDs that face away from the performer. They would therefore be of little use in solo performance and were not included in the prototype design. This allowed the spare digital inputs to remain free for later additions to the bow or for the possibility of a separate MIDI board to act as a master control during performance.

\textsuperscript{11} The “Blackberry” trackball is not taken from a Blackberry device, but is available as a Blackberry styled trackball from Sparkfun Electronics (Sparkfun Electronics n.d.).
A design featuring two trackballs was also considered for this project. Both trackballs were located on the frog of the bow. The middle two fingers of the right hand would be used to manipulate these. I decided to proceed with the original single trackball design in order to keep the bow simple for prototype use. This was aligned with the original intentions of a simple controller for multiple users. The dual trackball design will be constructed at a later date for personal use.