Expressive Musical Robots: Building, Evaluating, and Interfacing with an Ensemble of Mechatronic Instruments

By:

Jim Murphy

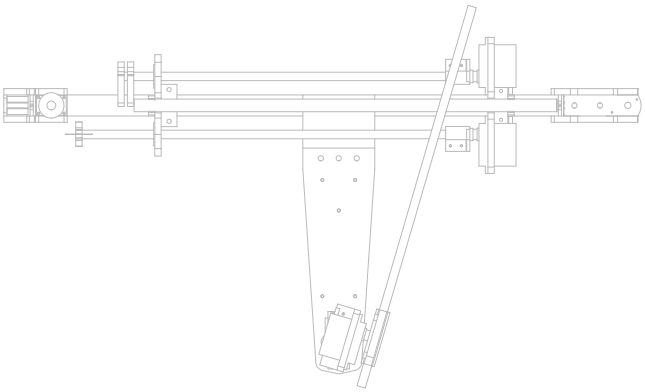
A thesis submitted to the Victoria University of Wellington in fulfilment of the requirements for the degree of Doctor of Philosophy

Victoria University of Wellington

2014

Supervisory Committee

- Supervisor: Dr. Ajay Kapur (New Zealand School of Music, Victoria University of Wellington School of Engineering and Computer Science)
- Co-Supervisor: Dr. Dale A. Carnegie (Victoria University of Wellington School of Engineering and Computer Science)



"Merlin himself, their elderly creator, said he had devoted years to these machines, his favorites, still unfinished."

James Gleick

Abstract

Expressive Musical Robots:
Building, Evaluating, and Interfacing with an Ensemble of Mechatronic Instruments

by Jim Murphy

An increase in the number of parameters of expression on musical robots can result in an increase in their expressivity as musical instruments. This thesis focuses on the design, construction, and implementation of four new robotic instruments, each designed to add more parametric control than is typical for the current state of the art of musical robotics. The principles followed in the building of the four new instruments are scalable and can be applied to musical robotics in general: the techniques exhibited in this thesis for the construction and use of musical robotics can be used by composers, musicians, and installation artists to add expressive depth to their own works with robotic instruments.

Accompanying the increase in parametric depth applied to the musical robotics is an increase in difficulty in interfacing with them: robots with a greater number of actuators require more time to program. This document aims to address this problem in two ways: the use of closed-loop control for low-level adjustments of the robots and the use of a parametric encoding-equipped musical robot network to provide composers with intuitive musical commands for the robots.

The musical robots introduced, described, and applied in this thesis were conceived of as musical instruments for performance and installation use by artists. This thesis closes with an exhibition of the performance and installation uses of these new robots and with a discussion of future research directions.

Acknowledgements

Many thanks to my PhD advisors, who have guided me somewhere I have never traveled. Dr. Ajay Kapur, who has aided and encouraged me in my work, deserves my abundant thanks for his tireless guidance. From my first day at CalArts, he has selflessly helped me to appreciate and realise my artistic and academic goals, and has taught me that the two need not be mutually exclusive.

Thanks also to Dr. Dale A. Carnegie, my secondary supervisor, for the open-mindedness with which my he has approached my inter-disciplinary status. It is my strong belief that his supervision has lent a rigor and dignity to my work of which I am very proud. The attention to detail he pays to his students' work is inspiring: I aspire to treat my own (future) students' work with as much care and thoughtfulness as he treats his.

Through their assistance, the faculty and staff at Victoria University of Wellington's School of Engineering and Computer Science and the New Zealand School of Music have enabled the work presented here to occur. Thanks to Drs. Will Browne, Dugal McKinnon, and Ted Apel for intriguing conversations, opportunities, and ideas, and to Tim Exley, Sean Anderson, and Jason Edwards for deep technical expertise, thoughtful advice, and patience. Thanks also to Victoria University of Wellington's Faculty of Graduate Research for providing me with the Victoria Doctoral Scholarship and to VUW and NZSM as a whole.

My friends and colleagues who I've met during my studies in Wellington have provided me with help and thoughtful ideas pertaining to this work: double thanks to Mo Zareei, Brett Ryan, Blake Johnston, Timothy Barraclough, and Patrick Herd. Thanks also to the SELCT crew for putting up with my ongoing musical robot evaluations and for your artistic and technical help.

Finally, my deepest thanks to Richard Vindriis, Paul Mathews, and James McVay. Your help with my projects has turned all of this work from an abstraction into something real.

I owe special thanks to fellow researchers and artists in the field of musical robotics (and computer music in general). Thanks to Ge Wang and all the developers of ChucK. Thanks to Gil Weinberg for articulating the need for robotic musicianship and to the Logos Foundation for inspiring me with fantastic robotic instruments. Thanks to Eric Singer for paving the way and setting a high standard against which I continue to measure my work. Thanks to Aengus Martin (and the UTS team) for inspiring me with your work on the feedback guitar. Also, thanks to Nicholas Baginsky, Edgar Berdahl,

George Tzanetakis, Andrew McPherson, the Waseda musical robotic team, and to all of the other innovators and researchers whose work I cite in this document.

To my friends and teachers from CalArts (and environs): Michael Darling (for turning me from a "maker" into a sculptor), Curtis Bahn (for showing me that, with enough patience, robots and humans can make beautiful music together), Martijn Zwartjes (for being a good enough friend and teacher to motivate me to finish projects that I start), and Perry Cook (for giving epic ChucK talks and being rapidly responsive to my DSP emails). Also to Carl Burgin (for making me question my work in the best possible way), Dimitri Diakopoulos (for HIDUINO, endless help over the last three years, and being a great friend and generally thoroughly inspiring individual), and to Jordan Hochenbaum and Owen Vallis (both of you were there on the ground floor, helping me to make my PhD into a real thing).

Thanks to Trimpin for his generosity: his openness with his knowledge has catalysed this research. I am humbled and inspired by his work, and am honoured to consider him a friend and mentor.

Mr. Evans, Ms. Chase, and Ms. Kempter: to call you my teachers is insufficient: you are my Zhuangzi; you taught me to wander where there is no path. Such open-minded wandering has led to this work, and for that I owe you my thanks.

To Kate and Chris. For being wonderful people and making me lucky enough to start this thesis-related work with only one sibling and letting me finish with two.

To my parents: while these words can't convey my gratitude to your sacrifices which have directly led to this work, I hope that you see on each of the pages in this work an indelible stamp of my thanks.

Also, Bridget. Without you, I would have been a pair of ragged claws, scuttling across the floors of silent seas...

Contents

List of Figures

List of Tables

To my family, friends, and teachers.

Chapter 1

Introduction

1.1 Expressivity and Musical Robotics: Motivations

Why is expressivity a goal worth striving for in the design of musical robotics? Robotic instruments bring computer music into physical space: "...as opposed to synthesizers, [they] resonate, project, and interact with sound spaces in richer, more complex ways" [1]. Their physicality is what makes them compelling, and the reasons for this are twofold. Sonically, the robot shares the space with the listener. Also, the contours and shifts present within the structure of electronically-composed music can be visually coupled with an object. By creating objects with many parameters, each capable of predictable and precise manipulation of the robot's audio output, these two reasons are accentuated: the visuality and aurality of robots increases along with their mechatronic expressivity.

Expressivity is a word that will appear throughout this work. A potentially subjective word, a definition of its usage in the context of this document is in order: expressivity, in this context, refers to the ability of a mechatronic musical system to affect a wide range of musical parameters (these parameters will be further explored in the following paragraphs). A robotic drummer capable of a single striking intensity at a single place on a drum head is less expressive than one able to vary its strike force and position. A robotic guitar with fixed frets is less expressive than one able to play portamentos up and down the instrument's neck. A robotic harmonium incapable of changing its pumping force is less expressive than one that can. In essence, according to synthetic instrument

builders Xavier Rodet and Christophe Vergez, an instrument capable of "subtle nuances or accentuated features which are under the control of the musician allows for musical expressivity and phrasing" [2]. The building of and interfacing with such instruments is focused upon in this thesis.

As little has been written concerning expressive robotics, a roboticist may turn to traditional musical writing to learn more about the musical expressivity. Patrik N. Juslin and coauthors, in "Feedback Learning of Musical Expressivity" [3], engage in a demystification of musical expressivity in all aspects of music, from compositionally-based to instrument-derived sources. Their work serves to provide useful criteria for any instrument builder concerned with providing musicians and composers with controllable parameters for expressive use. Juslin and his co-authors subdivide musical expression into five categories: piece-related, performer-related, listener-related, context-related, and instrument-related. Juslin states that the factors affecting instrument-related expressivity include "acoustic parameters available, Instrument-specific aspects of timbre, pitch, etc., [and] technical difficulties of the Instrument" [3]. The work conducted in this thesis focuses on those three parameters through the creation of musical robotics endowed with many acoustic and timbral parameters and, with the application of calibration, fail-safety, and networking systems, minimising the technical hurdle of creating compositions for these often-complicated machines.

For those looking to explore expressive instruments of a non-traditional nature, the proceedings of the New Interfaces for Musical Expression (NIME) conferences provide a rich tradition of studies of the implications of technology's role in musical expressivity. In "On Interface Expressivity: A Player-Based Study" [4], Cornelius Poepel references P. N. Juslin's contributions to the field of expressivity in performance, and further attempts to quantify the meaning of the term. He says that "Performers code expressive intentions using expressive related cues. Extensive work has been done to identify the most relevant cues. These cues include: tempo, sound level, timing, intonation, articulation, timbre, vibrato, tone attacks, tone decays, and pauses." These expressive parameters, as with those outlined by Juslin in [3], can provide robot builders with a checklist of parameters from which to draw in order to create expressively interesting robots. By seeking ways to add [2]'s aforementioned "subtle nuances" to these cues, an instrument's expressivity can be enhanced.

A question that might arise when designing expressive mechatronic musical instruments is whether the instruments themselves are expressive. In "The 'E' in NIME: Musical Expressive with New Computer Interfaces," Christopher Dobrian and Daniel Koppelman address this very issue, asking "can a machine be expressive?" Their answer fits with this document's intentions: "Although we may speak of an 'expressive instrument' for the sake of brevity, it is important to recognize that we usually mean 'an instrument that affords expression,' that is, 'an instrument that enables the player to be expressive"' [5]. This affordance of expressivity is what is meant throughout this document when referring to "expressive musical robotics." While it remains the realm of artificial intelligence specialists to pursue "machine expressivity," this work serves to allow performers and composers the capability to explore mechatronic instrument-based music in a more expressive manner than possible with many prior systems.

The musical robots built, tested, and applied in this work are intended as musical instruments to be explored and used by artists. An expressive musical robot will allow a composer to leave a deeper personal imprint upon the work than is possible with a less capable robot and will hopefully lead to a more diverse body of work for that instrument. This thesis contains an exhibition of performance and installation pieces that were created with the goal of exploiting the capabilities of the featured systems. This document's inclusion of a performance/installation "journal" and a user study with musicians chosen as participants highlights the intention of the robots as devices designed to be applied in a real-world setting rather than treated strictly as research tools. This interest in creating performance-ready instruments arises from the author's personal experience with the KarmetiK Machine Orchestra [6], which served as a motivation to engage in further musical robotics-related work.

While this work contains several case studies of specific instruments made expressive through the use of increased degrees of freedom, the underlying motivation for this work is more ambitious. By presenting the conception, design, construction, and use of expressive robots, it is hoped that a new practice of robots as musical instruments will be catalysed. In an era of easy-to-use microcontrollers, rapid prototyping equipment, and domain specific creative coding environments that foster mechatronic exploration, musical roboticists should abandon the status quo of unnecessarily constrained systems and begin to make expressivity a goal. If this work serves to foster the advance of musical robotics toward that goal, then it has succeeded.

1.2 Document Structure and Overview

Structurally, this document is divided into two parts, which are preceded by a history of automated music. The parts divide the work into its two subcategories: that of designing and building the physical mechatronic music systems and that of interfacing with parametrically-rich mechatronic musical instruments. The outline of this thesis is illustrated in Figure 1.1.

Chapter 2 provides an overview of automatic musical instruments. While Chapter 2 serves in part as a literature review, its main role is to familiarise readers with the broad field of musical robotics; more focused chapter-specific literature reviews are contained in each chapter of this thesis. The chapter begins with a section focusing upon the mechanical ancestors of contemporary musical robots, with early self-playing instruments from the Abbasid Caliphate and pre-Industrial Revolution Europe described. The decline of this first "golden age" of automatic musical instruments in the face of the rise of the phonograph and loudspeaker is discussed, and the subsequent rebirth of the field in the 1970's is presented. A second section explores contemporary musical robotics, including modern percussion, keyboard-playing, and chordophone-based instruments. Chapter 2 concludes with a discussion of the means by which the contemporary works presented served as motivations for the new works presented throughout the rest of the thesis.

Chapters 3, 4, and 5 make up Part 1 of this thesis, describing a series of new mechatronic instruments designed to afford users more expressive control than is typical of the state of the art.

Chapter 3 presents Swivel 2 and MechBass. The chapter begins with a systems overview of Swivel 2, a new six-stringed mechatronic slide guitar. Each of Swivel 2's subsystems are explained: the means by which the instrument's strings' pitches are changed is described, as well as the string picking and damping mechanisms. Chapter 3's section on Swivel 2 concludes with a detailed overview of the mechatron's electronics. Following the presentation of Swivel 2, MechBass is similarly described. Each of the mechatronic bass guitar's subsystems are focused upon, and the electronics used to drive the instrument's actuators and communicate with a host controller are detailed. Chapter 3 closes with the presentation of systems evaluations of Swivel 2 and MechBass.

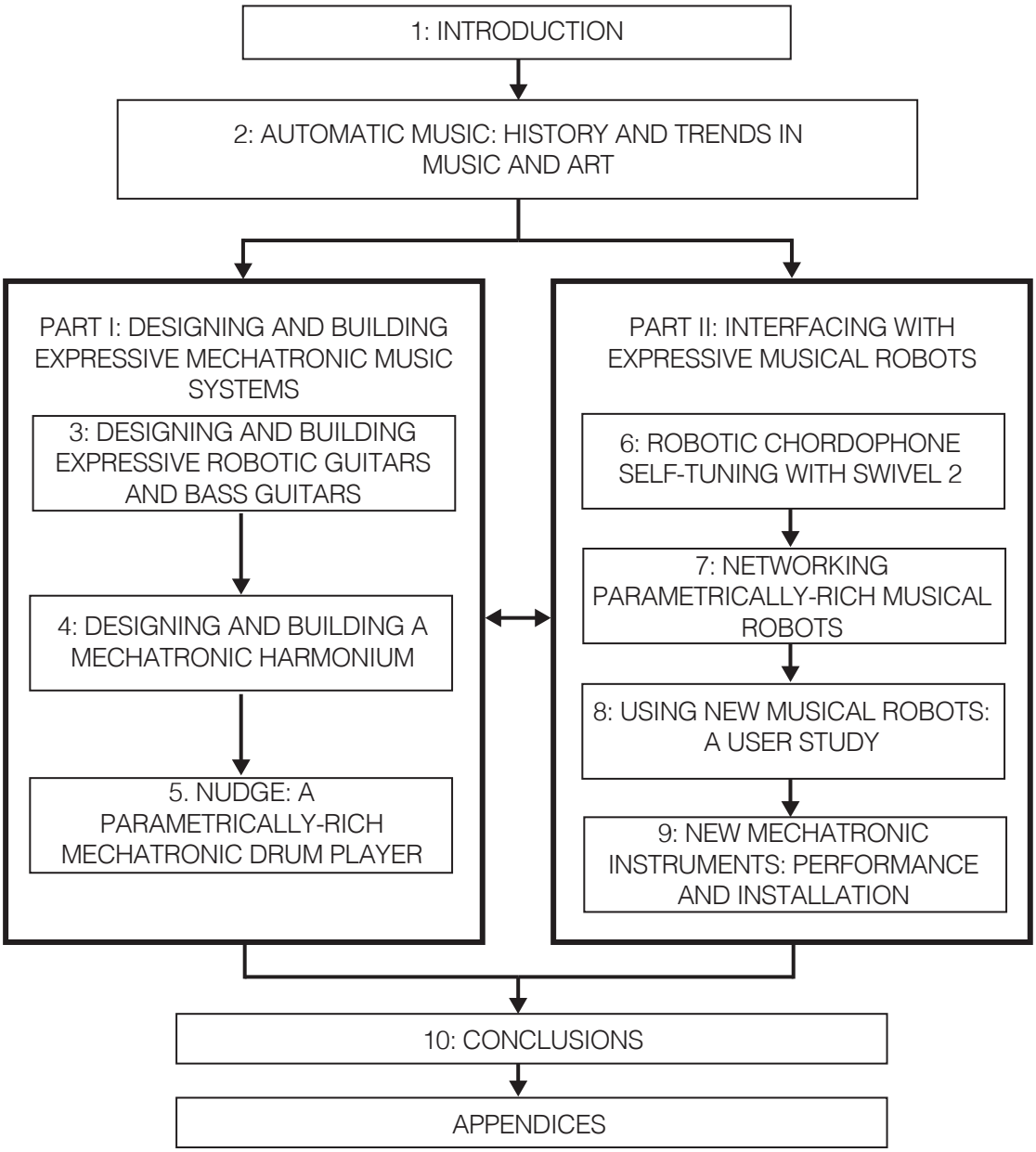


FIGURE 1.1: An illustration of the structure of this thesis. The arrows indicate a suggested order in which the chapters may be read.

Chapter 4 is similar in structure to Chapter 3, shifting from a focus on mechatronic chordophones to mechatronically-augmented keyboard instruments. Chapter 4 opens with a discussion of the motivations for building a mechatronic harmonium player and an overview of the characteristics of traditional Indian-style harmoniums. After introducing Kritaanjli, a mechatronically-augmented harmonium, the design and function of each of its subsystems is explained: the means by which the harmonium's keys are depressed and its bellows pumped are described, and the electronic subsystem that allows actuator control and communication between the mechatronic instrument and a

host computer is presented. After describing the instrument, the results of a series of performance evaluations are shown, presenting the instrument's bellows pumping speed, key repetition rate, and latency between key press and sounding.

Part 1's presentation of a series of new mechatronic instruments concludes with Chapter 5's focus on mechatronic drumming systems. The chapter opens with a discussion of existing mechatronic drummers, exploring the relatively limited parametric control that such devices afford users. After describing a number of preexisting mechatronic drum beaters, Nudge, a new drumming system designed to provide users more expressive control than previous systems, is described. As in Chapters 3 and 4, Chapter 5 provides detailed descriptions of each of the mechanism's subsystems: the means by which Nudge strikes the drum, adjusts its drum-striking intensity, and rotates its beater above a drum are presented. Following the descriptions of Nudge's subsystems, each subsystem's performance is evaluated and the results presented.

Where Part 1 of this document focuses on the construction of new mechatronic apparatus, Part 2's chapters detail means with which the new systems may be interfaced.

Chapter 6 describes a self-tuning system for mechatronic chordophones. Tested and deployed on Swivel 2, this approach is intended to allow users to interact with Swivel 2 by specifying note pitches rather than actuator-specific setpoints. The chapter begins with a detailed review of self-tuning systems for physical instruments, allowing readers to place the approach developed for Swivel 2 in context of prior works. After the review of prior systems, the process by which Swivel 2 self-tunes is described. Following a general description of the approach, the specific implementation on Swivel 2 is detailed, and an evaluation of the system's performance is presented.

Chapter 7 continues Part 2's focus on exploring ways for users to interface with new mechatronic instruments, focusing on networked performance use of the instruments. The chapter begins with a brief survey of network music, focusing on prior systems developed for the networking of mechatronic instruments. After examining the shortcomings of existing systems, Tangle, a client/server application designed for the networked use of mechatronic instruments, is presented. After an overview of Tangle's architecture, the ability to construct custom output behaviours for attached instruments is presented: a number of these "parametric encodings," intended to replace the normal one-to-one mappings afforded by the instruments, are described. After exploring some exemplar

parametric mappings, Chapter 7 describes ways that Tangle can be configured to calibrate and perform fail-safety functions on attached instruments.

Chapters 8 describes selected findings from a user study conducted to gain insight into composers' and performers' use of the new systems described in this document. User impressions of the systems' ease of use and expressivity are presented, along with their opinions regarding compositionally-useful parameters on the instruments and accompanying software. Finally, study participants' experiences with the Tangle client software are examined, and steps taken to improve its user interface are shown.

Chapter **??** presents a number of performances and installations conducted with the instruments described in this thesis document. Beginning with early performances using prototype instruments, the chapter chronologically follows the performance use of the systems through to later installations integrating multiple instruments to the final deployment of a full networked ensemble featuring each of the instruments described in Part 1 of this document. In essence, Chapter **??** provides a narrative of the motivations that led to numerous improvements and developments shown in this thesis.

Following concluding remarks in Chapter 10, this document contains a number of appendices. Appendix A lists and briefly describes the contents of the included supplemental CD-ROM. Appendix B describes the MIDI protocol with an emphasis on its use in the mechatronic instruments presented in Part 1 of this document. Finally, Appendix C lists all of the user study's questions and answers, allowing readers of Chapter 8 to cross-reference the conclusions presented about the study with the actual answers provided by study participants.

1.3 Publications Arising from this Thesis

The following journal and conference publications have been accepted during the course of the research presented in this document. All were accepted via peer-review process. These publications highlight the contributions that the work presented in this thesis has provided to the field of musical robotics.

- **Expressive Robotic Guitars: Developments in Chordophone-Based Musical Robotics** (*To appear*). Jim Murphy, Ajay Kapur, Dale A. Carnegie. Computer Music Journal. MIT Press.

- **Musical Robotics in a Loudspeaker World: Developments in Alternative Approaches to Localization and Spatialization**. Jim Murphy, Ajay Kapur, Dale A. Carnegie. Leonardo Music Journal, Issue 22. MIT Press.

- **Robot! Tune Yourself: Automatic Tuning in Musical Robotics** (*To appear at NIME 2014*). Jim Murphy, Paul Mathews, Dale A. Carnegie, Ajay Kapur. Proceedings of the 2014 Conference on New Interfaces for Musical Expression (NIME). June, 2014. London.

- **Tangle: A Flexible Framework for Performance with Advanced Robotic Musical Instruments** (*To appear at NIME 2014*). Paul Mathews, Jim Murphy, Dale A. Carnegie, Ajay Kapur. Proceedings of the 2014 Conference on New Interfaces for Musical Expression (NIME). June, 2014. London.

- **Swivel: Analysis and Systems Overview of a New Robotic Guitar**. Jim Murphy, Dale A. Carnegie, Ajay Kapur. Proceedings of the 2013 International Computer Music Conference (ICMC). Perth, Western Australia.

- **Designing and Building Expressive Robotic Guitars**. Jim Murphy, James McVay, Dale A. Carnegie, Ajay Kapur. Proceedings of the 2013 Conference on New Interfaces for Musical Expression (NIME). May, 2013. Daejeon, Korea.

- **Swivel 2: A Systems Overview and Evaluation of a New Mechatronic Slide Guitar**. Jim Murphy, Ajay Kapur, Dale A. Carnegie. Proceedings of the 2013 Engineering New Zealand Conference (ENZCon). Auckland, New Zealand.

- **MechBass: A Systems Overview of a New Four-Stringed Robotic Bass Guitar**. James McVay, Jim Murphy, Ajay Kapur, Dale A. Carnegie. Proceedings of the 2012 Engineering New Zealand Conference (ENZCon). Dunedin, New Zealand.

- **Interacting with Solenoid Drummers: A Quantitative Approach to Composing and Performing With Open-Loop Solenoid-Based Percussion Systems**. Jim Murphy, Ajay Kapur, Dale A. Carnegie. In Proceedings of the 2012 Australasian Computer Music Conference (ACMC). Brisbane, Queensland, Australia.

- **Kritaanjli: A Robotic Harmonium for Performance, Pedagogy, and Research**. Ajay Kapur, Jim Murphy, Dale A. Carnegie. Proceedings of the 2012 Conference on New Interfaces for Musical expression (NIME). May, 2012. Ann Arbor, Michigan.

- **Better Drumming through Calibration: Techniques for Pre-Performance Robotic Percussion Optimization**. Jim Murphy, Ajay Kapur, Dale A. Carnegie. Proceedings of the 2012 Conference on New Interfaces for Musical expression (NIME). May, 2012. Ann Arbor, Michigan.

Chapter 2

Automatic Music: History and Trends in Research and Art

This chapter presents a history of automatic music, beginning with early mechanical automata and continuing through to the current state of the art. At each stage in the field's history, those instruments deemed expressive for their time are highlighted and discussed. Additionally, contemporary trends in musical robotics are presented, with focus given to the many subdisciplines present within the field. While each chapter of this document will present works closely related to the chapter's topic, this chapter aims to provide a higher-level overview of automatic music, familiarising readers with the discipline as a whole prior to their reading of the more specifically-focused subsequent chapters.

This chapter opens with a discussion of early musical automata, describing the rich history of self-playing mechanical instruments. After discussing the field's ascendency during the Industrial Revolution and decline in the face of the phonograph and loudspeaker, the field's rebirth as "musical robotics" in the 1970's is examined. Contemporary advances in the subdisciplines that pertain to the new work presented in this thesis are presented, including recent advances in robotic percussion, keyboard instruments, and chordophones.

2.1 Early Musical Automata

In an era of ubiquitous audio recording and playback, the idea of a repeatable musical performance is so familiar as to hardly warrant consideration. Prior to the phonograph, though, inventors and composers turned to musical automata to allow for consistent and convenient performances of compositions. This section details the emergence of automatic instruments, their "golden age", and their decline in the face of modern general-purpose audio reproduction technology. Such a study of historical instruments in a work focusing on contemporary musical robotics is valid for two primary reasons: firstly, musical roboticists may draw inspiration from the centuries of mechanisms used to actuate musical instruments; secondly, the expressivity afforded by mechatronics (a hybrid discipline integrating mechanical and electronic components) is highlighted when contexualised against a history of strictly mechanical instruments.

2.1.1 The Punched Barrel: A Millenium of Applications

Fully automated music arose from the rich history of Greek and Arabic organ-making. Pneumatic and hydraulic organs, while not autonomous, are among the earliest "assistive" instruments, whereby a human player's actions are mediated by water- or air-driven mechanical means. The Banū Mūsā brothers took the mechanical assistance one step further, as explained in great detail in their ninth century treatise "The Instrument Which Plays By Itself." The 1931 translation by Henry George Farmer presents annotated diagrams, showing a pneumatic/hydraulic organ capable of playing songs programmed via cogwheels [7] (reproduced in Figure 2.1). Technology historian Teun Koetsier asserts that this automatic organ is "the earliest known design of a programmable machine" [8].

With the advent of automatic musical instruments, listeners could hear a piece precisely as a composer intended, time and time again. Later centuries saw the rise of automated music in Europe: according to Charles Fowler, pinned-barrel activated "'mechanical carillons' are mentioned in the Low Countries as early as the beginning of the thirteenth century" [9]. Recent articles focusing on the history of automated music in Europe emphasize their ubiquity in the musical landscape of the Classical period and beyond. They were so common that they were often overlooked; according to Terrance Riley in

"Composing for the Machine," "The technology of clockwork and pinned barrel . . . was so widespread and so widely-applied by 1820 that even highly complex symbioses of human and machine were no longer philosophically provocative" [10].

European musical automata consisted largely of pinned or stapled barrels which, when rotated, activated the instruments to which they were attached (as shown in Figure 2.2). These instruments ranged from organs and small flute clocks to carillons. Such instruments were not overlooked by leading composers: Orchestrions, assemblages of multiple automated instruments, attracted the attentions of Beethoven [10]. Prominent automated music scholar W.J.G. Ord-Hume, in "Ornamentation and Mechanical Music" further details compositions by well-known composers, including Haydn and Mozart [11]. Indeed, Ord-Hume and David Fuller [12] emphasize automated music's value as a "time capsule", allowing modern musicologists to examine music at a lower abstraction layer than would be possible with abbreviation-filled musical scores. This ability for an instrument to precisely output a composer's input commands highlights the instruments'

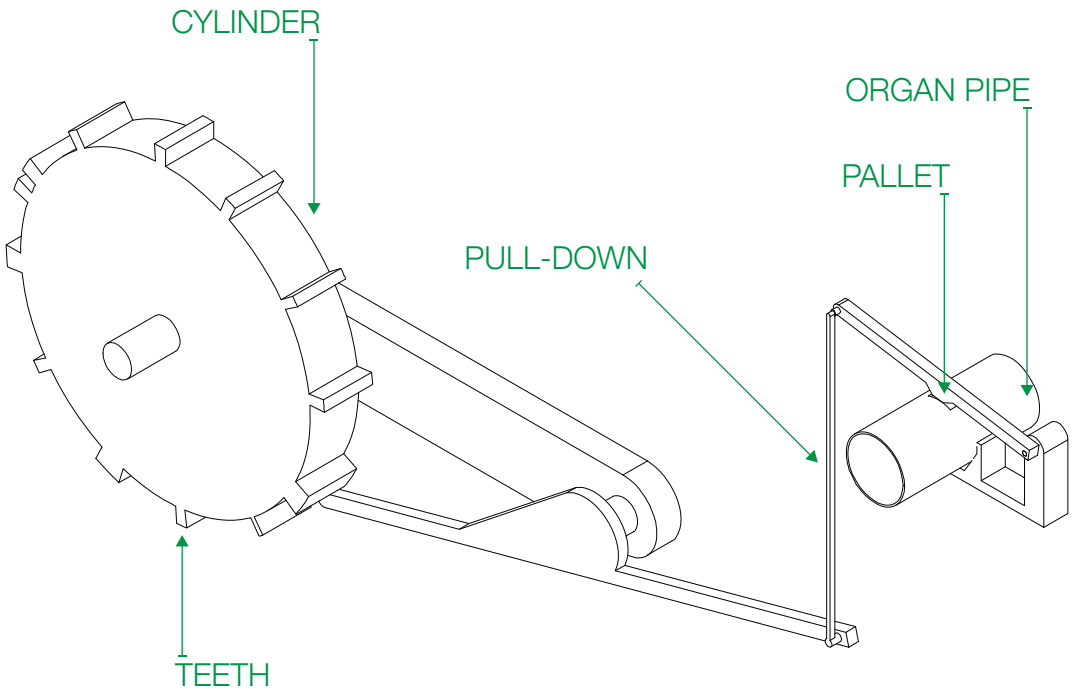


FIGURE 2.1: A diagram of a section of the the Banū Mūsā brothers' hydraulic organ, reproduced here from diagrams originally appearing in [7]. More details about its functionality can be found in [7] and [8]. As the cylinder rotates, contact with the teeth raise and lower the pull-down. The pull-down opens and closes an organ pipe's pallet, changing the instrument's sound.

repeatability.

2.1.2 From Repeatability to Expressivity

The decades leading up to and immediately following the invention of the phonograph saw automatic musical instruments at their popular zenith. The Pianola and similar instruments, described in [13], were mass-produced and appeared abundantly on streets, in bars, and in churches and homes [14]. These mechanically-complicated instruments (an example of which is illustrated in Figure 2.3) began to feature mechanical means of expressive control over parameters such as volume and pitch: an 1888 article appearing in the scientific periodical Science speaks of rapid innovations over the "crude mechanism

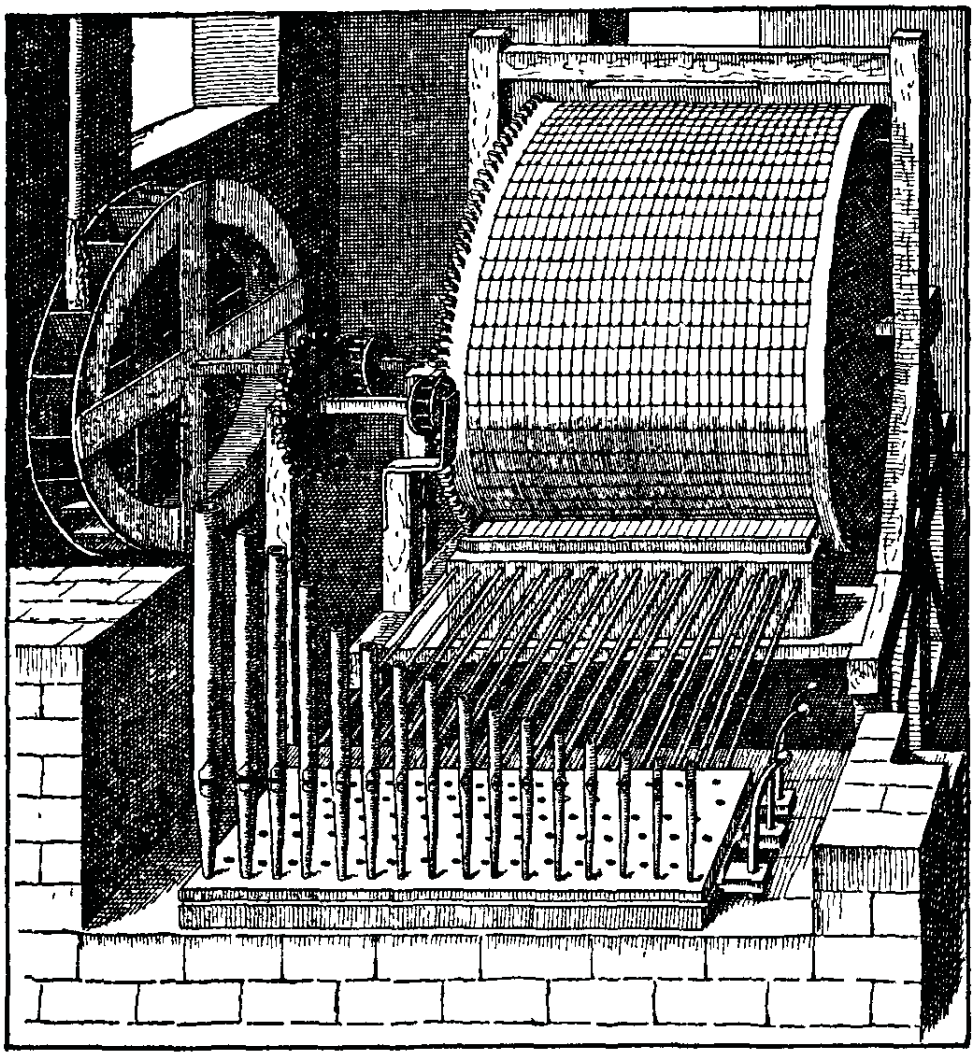


FIGURE 2.2: A 1615 illustration of a barrel organ, showing a punched barrel and hydraulic actuation system, from *Les Raisons des Forces Mouvantes* by Solomon de Caus. This instrument is similar to that described by the Banū Mūsā brothers.

of a few decades ago," and proceeds to describe a simple tremolo mechanism mounted on a mechanical zither [15]. The earlier Componium, described in [10], featured an ability to play variations on themes. It appears as though, with the mechanical aspect nearly mastered, inventors began to search for a deeper level of expressivity in their instruments by exploring the addition of extra parameters for control.

The invention and popularity of the phonograph rendered largely moot the quest for musical repeatability through augmented automatic instruments and shifted much of music technology's focus to that of recorded sound. With the onset of reliable recordings, complicated automatic instruments appeared to be a technological dead end. While composers such as Kurt Weill [11] and Conlon Nancarrow [16] continued to explore automatic musical instruments during the early- and mid-twentieth century, it would

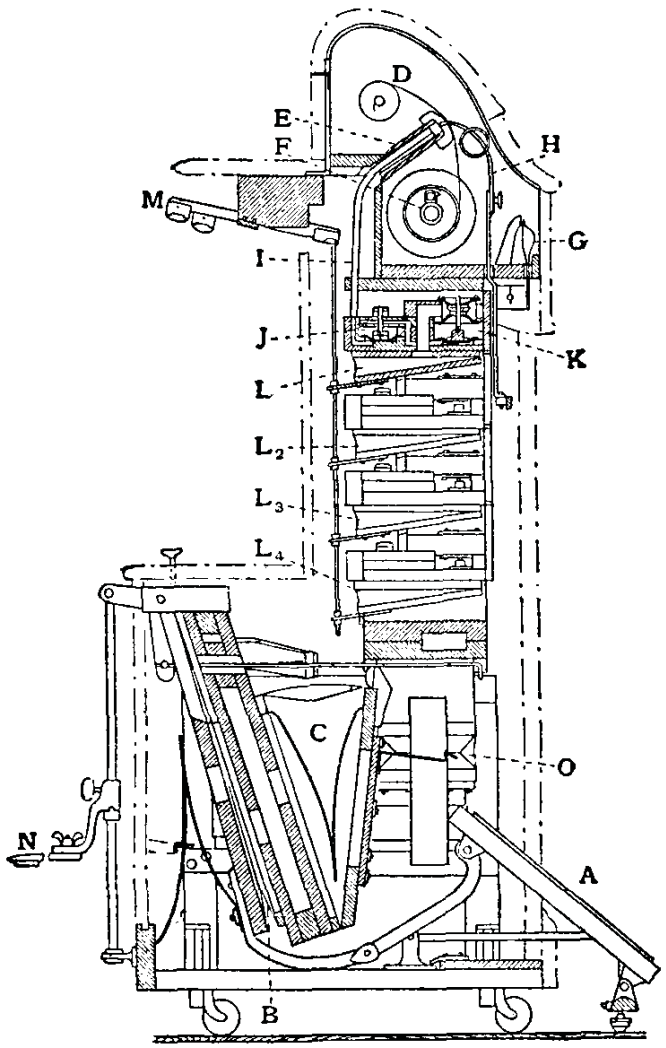


FIGURE 2.3: An early-20th Century automatic piano player, showing its complicated construction.

not be until the early 1970's that a new wave of automated music would appear.

2.2 Musical Robots in a Loudspeaker World

The modern field of musical robotics can be traced to the 1960's. Section 2.2.1 focuses on the earliest work and pioneering contributors, from whom many current practices are derived. Following an examination of the earliest workers in the field, a review of current practitioners, some of whom are interested in the study of robotic expressivity, is undertaken.

The novel contributions of this work focus on robotic guitars, keyboard instruments, and percussion systems. As such, in this section there is special attention devoted to a review of contemporary robotic chordophones, percussion instruments, and keyboard-based instruments. Examinations of the works discussed here have resulted in the expressive advances presented in the rest of this thesis.

2.2.1 The 1970's Renaissance

By the 1970s, few living people in the industrialized world could remember a time before phonographs and subsequent audio emulation technologies. The loudspeaker's domination was complete, and it had transformed the world in the process: loudspeakers were ubiquitous, and humanity was accustomed to experiencing the virtual reality of recorded music played through loudspeakers.

During the 1960's, collectives arose that were ready to overturn existing musical trends. One such collective was the Logos Foundation, founded in Ghent, Belgium, in 1968. Founder Godfried-Willem Raes (shown in Figure 2.4) cites "a desire to conquer the hierarchy of power involving music and its producers" [17] as a motivating factor in creating the collective. While the Logos Foundation originally chose electronic instruments as the means to accomplish the task, by 1972 it was involved in the construction of mechatronic sound sculptures. These sculptures were among the first kinetic sound

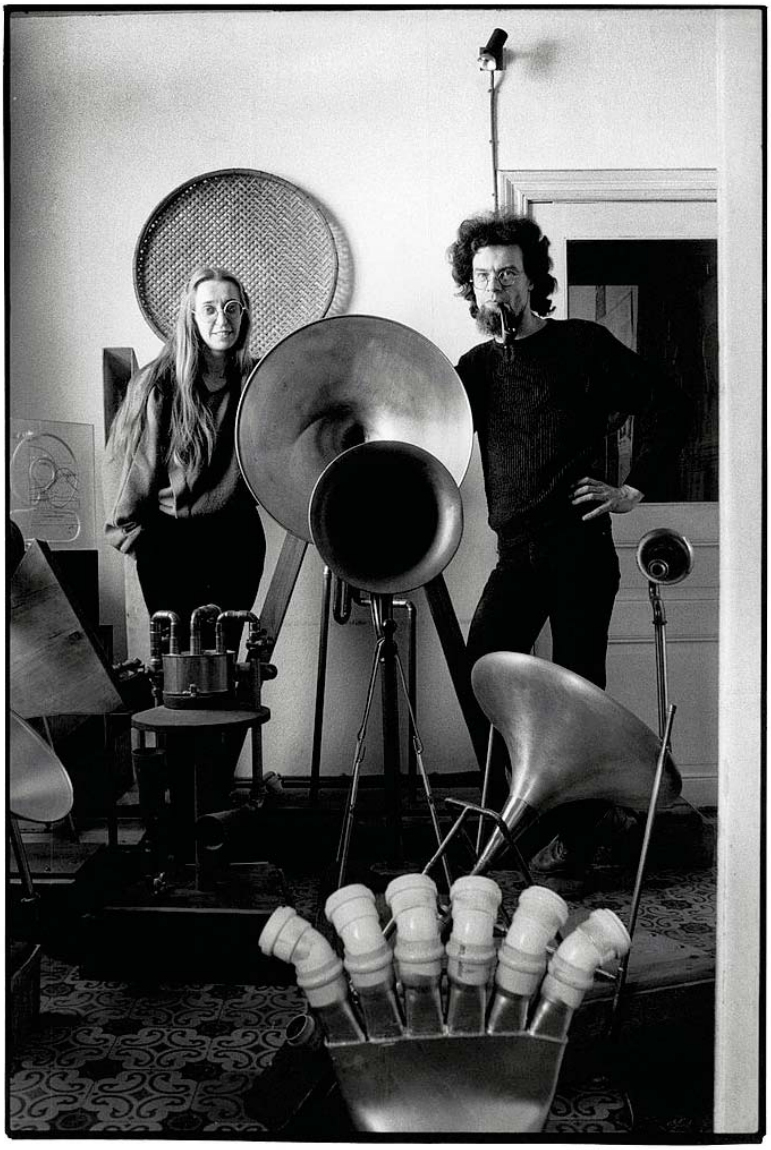


FIGURE 2.4: Godfried-Willem Raes, Moniek Darge, and robotic instruments of the Logos Foundation in 1983 (Image courtesy of Michiel Hendryckx).

sculptures to make use of digital logic, "exclusively realised with electromagnetic components, such as relays and telephone exchange shift-registers and counters" [17][1].

Another early musical roboticist who continues to be a central figure in the community is sound artist Trimpin. Artist Rafael Lozano-Hemmer perhaps sums up Trimpin's early work best by describing it as "adapting the digital to reinterpret the analog" [18]. Trimpin's close work with player-piano composer Conlon Nancarrow has greatly informed his subsequent musical output, much of which combines influences derived

[1]Some of the work of Fluxus artist Joe Jones predates that of Raes: while electromechanical rather than mechatronic, the automated music created by Jones has likely served as inspiration for subsequent musical roboticists. See http://ubu.com/sound/jones.html (retrieved March 4, 2014) for more information.

FIGURE 2.5: Sound artist Trimpin and a prototype of his 2006 sculpture *Shhh*. Image courtesy mit.edu.

from Nancarrow's intricate pattern-based music with a focus on spatialisation (especially evident in his 2000 work *Conloninpurple* [19]).

The decision by Trimpin and Raes to use electronic components, while logical in the latter half of the 20th century, nonetheless marks a significant shift in the history of automated music: automatic music had entered the computer age after centuries of pinned barrels, punched disks, and piano rolls. No longer were compositions for automated instruments a linear affair; near-limitless variations could be explored with conditional branching. Computer-aided actuator control would allow for precise manipulation of large numbers of motors, affording instruments the ability to change their sound output in a repeatable and expressive manner.

With the combination of electronic techniques and automatic musical instruments, the stage was set for a series of innovations over the next decades. While Trimpin and Raes continue to innovate, other workers have emerged who both seek to apply mechatronic music techniques to new genres and who explore the application of enhanced techniques for musical robot expressivity. The following subsections detail significant examples of such work.

2.2.2 Recent Advances in Robotic Percussion

Robotic percussion systems are the most widely-built variety of musical robots. Many advances in robotic musicianship, robotic ethnomusicology, and musical robotic ensembles largely or exclusively feature percussion instruments. The abundance of robotic percussion instruments is likely due to their potential for mechatronic simplicity: unlike chordophones or aerophones, functional percussive effectors (for both ideophones and membranophones) can be built with few moving parts, allowing for workers to focus on compositional rather than engineering goals.

This subsection examines four key contemporary applications in which robotic percussion instruments are used: ensembles of robotic instruments, percussion robots as a means of furthering robotic musicianship, and mechatronic percussion systems in kinetic sound sculpture. Common traits across each of these subdisciplines are examined, and a chronology of each is presented.

Robotic percussion instruments have long played a key role in many historical and contemporary examples of musical robotic ensembles. Pioneers Trimpin and Godfried-Willem Raes, discussed previously in Section 2.2.1, include percussion instruments in many of their sound sculptures and compositions. The Logos Foundation, for example, features at least 20 automatic drumming instruments in their Man and Machine robot orchestra [20]. Trimpin's sculptural ensembles, of which many can be seen in [19], [21], and [22], often utilise mechatronic drumming apparatus: his works *JackBox* and *Laptop Percussion* (detailed in [6]), for example, use solenoid-actuated drum beaters.

Many recent musical robotic ensembles consist either partially or completely of percussion instruments. As of 2004, Eric Singer's League of Musical Urban Robotics (LEMUR) consisted largely of solenoid-actuated ideophones and membranophones [1]. According to Singer et al., these instruments...

> "...provide composers with an immediacy of feedback, similar to composing on synthesizers. However, as opposed to synthesizers, physical instruments resonate, project and interact with sound spaces in richer, more complex ways. Clearly, they have a more commanding physical presence as well" [1].

Many robotic ensembles formed after Singer's LEMUR also make use of percussion instruments. Ensemble Robot[2], a human/robot performance troupe founded by Christine Southworth and Leila Hanson whose first performance was in 2005, extensively use solenoid-based percussion systems. Brighton-based artist Sarah Angliss performs with an array of bell-playing and anthropomorphic robots[3]. Both Felix Thorn, creater of Felix's Machines[4], and Roger Aixut et al., founders of the Cabo San Roque experimental instrument collective and creators of The Mechanical Orchestra of França Xica[5], make use of solenoid-based percussion systems in their robotic sculptures and performance instruments. Most of these ensembles, further described in [23], utilise relatively simple mechatronic systems: actuators with few degrees of freedom are employed, and technical simplicity is favoured over parametric density. Chapter 5 presents a number of these systems, comparing them with more parametrically expressive approaches.

Other recent notable musical robotic ensembles include Carnegie Mellon University's RobOrchestra, directed by Dan Curhan, Nick Yulman's percussion ensemble, and Ajay Kapur and Michael Darling's KarmetiK Machine Orchestra. RobOrchestra includes a wide array of instruments, many of which use solenoid actuators for percussive effects[6]. Yulman's Index Boogie[7] is an assortment of mechatronic percussion instruments. The KarmetiK Machine Orchestra [6], consists of an ensemble of human musicians and robotic percussion instruments. These instruments are outlined in [6] and further detailed in [24], [25], and [26]. While some of these ensembles feature novel approaches to drum effector design, most use solenoid-based drum beaters with limited degrees of freedom.

The reasons for the abundance of percussion instruments in ensembles of musical robotics are likely twofold. Firstly, mechatronic instruments excel at many of what Singer describes as the key reasons to explore robotic music: "...robotic instruments can play in ways that humans can't or generally don't play. Some of these capabilities include speed, pitch and expression granularity, complex polyrhythms and extended duration playing" [1]. These reasons can be explored to some degree with even the simplest, least "expressive" mechatronic instrument. Secondly, composers and inventors seeking

[2]http://www.ensemblerobot.com/ (retrieved September 2, 2013)
[3]http://www.sarahangliss.com/ (retrieved September 2, 2013)
[4]http://www.felixsmachines.com/ (retrieved September 2, 2013)
[5]http://www.cabosanroque.com/ (retrieved September 2, 2013)
[6]http://roboticsclub.org/projects/roborchestra/overview (retrieved September 2, 2013)
[7]http://nysoundworks.org/ (retrieved September 2, 2013)

to create large arrays of robotic instruments likely turn to often-simple robotic percussion apparatus as a rapid means by which to explore masses of instruments on stage or in the gallery. Such simple drum beaters can be inexpensively duplicated en masse in a manner difficult for expensive parametrically-rich drum players (though Nudge, described in Chapter 5, is designed with simplicity and low parts-count as a goal, allowing for easier duplication than similar instruments).

Rather than create ensembles of percussion instruments, some workers use robotic drumming systems as research tools to further what roboticist Gil Weinberg calls "robotic musicianship" [27]. It is in this subdiscipline of musical robotic percussion where most novel work on expressive drum players appears to be emerging. While both Trimpin and Godfried-Willem Raes have long explored novel means of extending robotic percussion techniques, Weinberg himself is an early and important contributor to the field. His 2006 work, coauthored with Scott Driscoll, "Toward Robotic Musicianship" [27] introduces a drumming system with added degrees of freedom. Weinberg's robotic algorithms take advantage of his robots' added degrees of freedom, allowing them to extend their expressivity [28]. Other workers who turn to novel approaches to explore robotic musicianship are the creators of the Expressive Machines[8] ensemble, who have developed a snare drum fitted with a large number of beaters, allowing for composers to selectively strike various regions of the drum head. More recently, Jun Kato [29] and Alyssa Batula, et al. [30] have explored using novel software techniques and control schemes to allow for simpler and more reliable percussion systems. The intentions of these workers on percussion-based "robotic musicianship" appear most similar to those espoused in this document: enhanced expressivity through increased mechatronic and software sophistication.

Quite different from robotics researchers are those who use robotic percussion in kinetic sound sculpture. While many kinetic percussion sculptures, such as those of Zimoun[9] (discussed further in [31]), Pe Lang[10], Chris Kaczmarek[11], and Dan Senn[12], are arguably more electromechanical than robotic, some workers use the same technology developed for other subdisciplines of robotic percussion within their sound sculptures. Nicolas Bernier[13], for example, uses solenoid-based striking systems to actuate tuning forks in

[8] http://www.expressivemachines.com/ (retrieved September 2, 2013)
[9] http://www.zimoun.net/ (retrieved September 2, 2013)
[10] http://www.pelang.ch/ (retrieved September 2, 2013)
[11] http://chriskaczmarek.com/ (retrieved September 2, 2013)
[12] http://www.dan-senn.com/index.html (retrieved September 2, 2013)
[13] http://www.nicolasbernier.com/ (retrieved September 2, 2013)

his 2011 piece *Frequencies*. Making use of augmented instruments, Lukas Flory, Alexandros Konstantaras, and Andreas Gartz use solenoid-based drum beaters to strike frame drums[14]. While such workers are perhaps less scholarly than many in the other subdisciplines in their pursuit of expressive percussion instruments, the works produced by these artists often differ greatly from the popular "solenoid drum beater" paradigm discussed above. Because of this novelty, such percussive sound art can serve as inspiration to those seeking new methods of musical robot percussion expressivity.

Automated percussion is perhaps the most diverse of the subfields of musical robotics: its widespread application in ensembles, robotic ethnomusicology, kinetic sound sculpture, and robotic musicianship research applications is likely due in part to its simplicity when compared to the more mechatronically-complicated subfields of robotic chordophones and aerophones. While works such as [27] highlight innovations in more complicated drumming systems, many implementations opt for simplicity over expressivity: there exist opportunities for roboticists to greatly enhance the expressivity of robotic percussion through the use of feedback and additional degrees of freedom. Chapter 5 of this document presents examples of such new work in musical robotic percussion.

2.2.3 Recent Advances in Robotic Keyboard Instruments

Two parallel trends are evident in automatically-actuated keyboard instruments: that of assistive augmented pianos and that of fully-automated instruments. Assistive augmented pianos consist of instruments created to be played by a human performer; the performer's real-time playing is then altered through mechatronic means. Fully automated instruments are capable of audio playback with no direct human interaction. Occupying a middle ground between assistive and automated keyboard instruments is the Yamaha Disklavier (and similar instruments), an instrument which can be both autonomous and interactive. That such instruments are often arguably more expressive than percussion instruments is likely due to their greater mechanical sophistication: any augmented keyboard instrument will likely have more degrees of freedom than a single mechatronic drum, affording composers more expressive potential.

[14] http://miex.it/ (retrieved September 2, 2013)

Electromechanical keyboard playing aids have been in use through much of the 20th Century. In 1935, Norwegian intonation researcher Elvind Groven created one such device, capable of retuning piano strings [32]. In recent decades, electronic aids made such augmentations more capable and expressive: Edgar Berdahl et al. in [33] present a piano capable of electromagnetic actuation. Per Bloland, inspired by [33], presented an array of compositional ideas for such "electromagnetically-prepared" instruments [34]. Additionally, in 2007, Michael Fabio created the Chandelier, a piano-inspired instrument to extend keyboardists' expressive range [35]. More recently, Andrew McPherson has presented the Resonator Piano, "an augmented instrument enhancing the capabilities of the acoustic grand piano" [36]. The Resonator Piano allows for tremolo and portamento, as well as sustained notes.

A recurring theme of the above instruments is a desire by their creators to extend the expressive capabilities of human keyboardists. Alternately, fully automated keyboards, while unable to directly interact with human performers, can serve as accompaniment devices or as standalone instruments capable of playback or generative music performance[15]. Trimpin's Vorsetzer is an early example of such an instrument[16], and is discussed in more detail in [21] and [37]. The various versions of the Vorsetzer are MIDI-controlled keyboard players, and have been adapted, used, and modified by Godfried-Willem Raes[17]. Another early keyboard-playing robot was developed by the Waseda robotics research group [38], and consists of articulated actuators and a computer vision-assisted score reading system. Like much of the output of the Waseda research group, it is anthropomorphic and arguably oriented toward the field of anthropomorphic robotic research than to performance and installation use.

The Yamaha Disklavier [37] fuses assistive and automated keyboard mechatronics: humans can play concurrently with the solenoid-actuated strings, allowing performers to interact with computer-driven compositions [39]. The Disklavier has a rich history of compositions, with composers such as Horacio Vaggione [40], David Rosenboom, Terry Riley, Michael Pascal, and Morton Subotnick creating pieces for it [41]. More recently,

[15] James Tenney's player piano compositions, including his 1964 *Music for Player Piano*, featured computer-aided composition techniques for a mechanical player piano; these works anticipate many subsequent computer-aided composition techniques for musical robotics and are worthwhile pieces of study for any student of the history of musical robotics. For more information, see http://www.plainsound.org/JTwork.html (retrieved February 5, 2014).

[16] Trimpin has built a wide array of augmented piano instruments, many of which can be seen in [19] and [22].

[17] http://logosfoundation.org/instrum_gwr/playerpiano.html (retrieved September 9, 2013)

composer Jeff Aaron Bryant has explored novel means by which performers can interact with robotic instruments, creating new interfaces for human interactions with Disklaviers [42].

The expressive innovations of the keyboard-playing robots discussed above have served as inspiration for Kritaanjli, a new mechatronic harmonium player presented in Chapter 4. Kritaanjli transfers many of these piano-focused developments to a harmonium, allowing for the harmonium's instrument-specific expressive potential to be mechatronically explored.

2.2.4 Recent Advances in Robotic Guitars, Basses, and String Istruments

Robotic guitars, as with most contemporary robotic musical instruments, have ancestors in pre-loudspeaker automatic musical instruments such as the Orchestrion and Banjorchestra[18]. These instruments typically augmented existing human-playable instruments by placing them beneath pneumatic plucking mechanisms and string fretting systems. Such instruments largely fell out of widespread use in the decades following the introduction of the phonograph and loudspeaker [31].

Starting in the 1970s, alongside other changes discussed in Section 2.1.2, artists and instrument builders began to use mechatronic components to realie more sophisticated robotic guitar systems. Trimpin created numerous iterations of mechatronic guitars during the 1990's and 2000's. Starting in the early 2000's, The Logos Foundation began to build an ensemble of string-based instruments, including 2011's guitar-like Synchrochord monochord[19]. Perhaps best known is LEMUR founder Eric Singer's GuitarBot [43], a four-stringed picked slide guitar-like instrument which gained exposure during its tour with guitarist Pat Metheny[20]. Also oft-mentioned is Nicolas Anatol Baginsky's Aglaopheme guitar from his Three Sirens ensemble, discussed more in [23].

[18] http://www.mechanicalmusicpress.com/history/articles/banjo.htm (retrieved September 9, 2013)
[19] http://logosfoundation.org/instrum_gwr/synchrochord.html (retrieved September 9, 2013)
[20] http://www.patmetheny.com/orchestrioninfo/ (retrieved September 9, 2013)

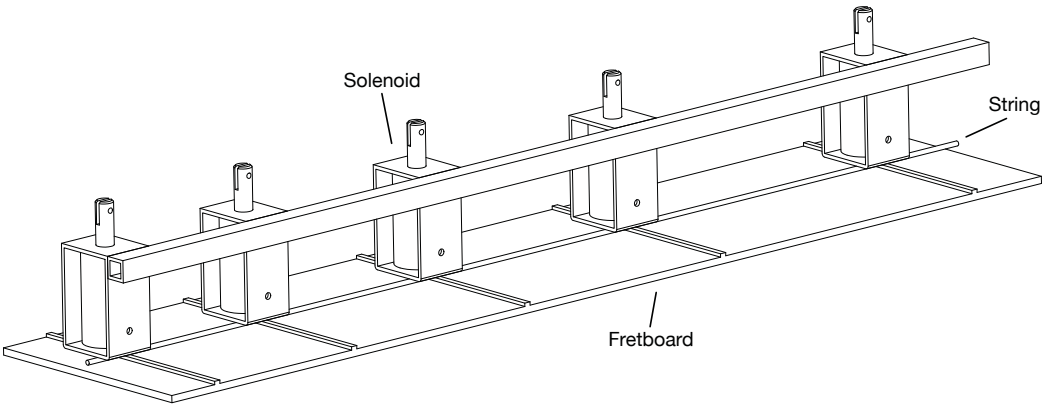


FIGURE 2.6: A solenoid-based string fretter: Each fret on the bass or guitar's fretboard has an accompanying push-type linear solenoid which, when actuated, pinches the string between the next nearest fret and the instrument's bridge.

The following subsections will focus upon the aforementioned instruments in terms of their respective fretting, plucking, and damping subassemblies. A study of these systems is useful when studying the mechanisms shown in Chapter 3.

2.2.4.1 Techniques for Pitch Manipulation

In any robotic guitar, a critical consideration in the instrument's construction is the means by which different notes can be selected. All robotic guitars and basses use one of two methods for accomplishing this: the most popular method involves a linear array of fixed-position solenoids. The second technique utilizes a carriage-mounted "fretter" which traverses the length of the string. Each method will be discussed in detail below.

Trimpin's automatic guitars and basses [19], as well as the Logos Foundation's Synchrochord monochord, the robots of Expressive Machines (discussed in [23]), and instrument builder Vladimir Demin[21] utilize a mechanically simple method of changing the guitar's pitch: solenoids are mounted along a fretboard equipped typically with chromatically-spaced frets. Upon actuation, the solenoids push their plungers onto the string. This action clamps the string at the fret, resulting in a change in the string's pitch. Figure 2.6 illustrates such a mechanism, which is quite similar to the pitch manipulation systems utilized by Industrial Revolution-era automatic instruments[22], about which Arthur W.J.G. Ord-Hume has written much [44].

[21] http://www.youtube.com/watch?v=B40XymKPicA (retrieved September 9, 2013)

[22] An illustration of one such system from 1905 can be viewed at www.millsnovelty.com (retrieved March 4, 2014).

Fixed-position solenoid systems have two key advantages over other pitch manipulation approaches. Musically, they allow for rapid transitions between pitches; conversely, in the linear motion systems discussed below, time must be taken for the slide to reposition the fretting mechanism for different pitches. Additionally, such fixed-position solenoid systems are relatively simple to construct: they consist of a bracket mounted above the string to which stationary solenoids are attached. Instrument builders use several methods to mount solenoids above the guitar's string: Trimpin makes use of T-slot aluminium extrusion in his assemblies, allowing for rapid repositioning of solenoids; Expressive Machines creates custom-fabricated acrylic brackets to which the fretting solenoids are attached. Vladimir Denin and Ragtime West build aluminium plates with arrays of holes drilled to correspond to fret locations.

Such fixed-position solenoid pitch manipulation systems suffer from two primary drawbacks: weight and discreteness. Arrays of numerous solenoids, especially the large number used by Expressive Machines and Vladimir Denin, result in heavy and cumbersome systems. Trimpin addresses this by using multiple custom-built monochords rather than directly augmented guitars: the weight is reduced over each string. Additionally, arrays of string clamping mechanisms are incapable of pitch bends, slides, or other expressive vibrato-like effects: additional actuators would be required to achieve such effects.

To address the drawbacks of the fixed-position solenoid systems, some musical roboticists have utilized linear positioning devices to move a fretting mechanism to any point along the string's length. An early example of such a system is Nicolas Anatol Baginsky's Aglaopheme slide guitar[23], first exhibited in 1992. While the literature lacks any detailed explanation of the instrument, according to Baginsky's website, "a DC motor drives a sledge up and down the guitar neck. This sledge carries the slide and a small solenoid that pushes the slide against the strings if necessary." In spite of its age, Aglaopheme remains a highly advanced musical robot whose mechanical complexity (and artificial neural network-based improvisation scheme) remains largely unsurpassed. The academic literature would benefit from a more in-depth exploration of this somewhat under-documented instrument.

A second influential robotic guitar-based instrument is Eric Singer's GuitarBot [43]. This slide guitar, unlike Baginsky's Aglaopheme, lacks the ability to disengage the slide

[23] http://www.baginsky.de/agl/agl_index.html (retrieved September 9, 2013)

from the string: every pitch transition is thus a portamento. To position the fret along the string's length, a DC servomotor (utilising a PID control scheme) moves a belt to which the fret is attached. Different pitches are established by sending setpoints to the servomotor. GuitarBot's linear positioning scheme and modular configuration served as inspirations to the new chordophones introduced in this work.

Both linear positioning guitar-based robots as well as derivative instruments such as Festo Automation's string quartet Sound Machine 2.0[24], suffer from significantly greater latency between consecutive plucking events than is present in the aforementioned fixed-position solenoid instruments. To address the latency problem, high-speed positioning systems must be used: other methods of positioning, including mechanisms inspired by industrial robotic manipulators, offer potential for the rapid yet accurate positioning required in robotic guitars and bass guitars.

2.2.4.2 Mechatronic String Picking

String actuation systems on automatic guitars are perhaps the most sonically important subsystem on such instruments: they serve as the primary physical interface between the mechatronic components and the string. Two main systems of string plucking exist in the existent robotic guitars and basses: solenoid-based pickers and rotary motor-based pickwheels. Electromagnetic actuation systems have also been built, wherein EBow-style mechanisms are used to excite the string [19]. Additionally, bowing systems (featuring violin bows or bow-wheels) have been deployed on mechatronic violins and may be adapted for guitar use.

Solenoid-based picking systems are perhaps best demonstrated in Trimpin's JackBox and KrautKontrol instruments. Figure 2.7 illustrates such a system: Two solenoids are mounted perpendicular to the instrument's string. A pick is mounted between the two plungers. Upon actuation, one solenoid energizes while the second remains off, pulling the pick across the string. During the next actuation event, the second solenoid energizes while the first remains off, pulling the pick back across the string. With sufficient solenoid plunger damping, solenoid-based picking mechanisms are quiet and fast. Existing systems, though, lack the ability to vary the power of the pick. While future versions could be modified to raise and lower the picking assembly in

[24] http://www.festo.com/cms/en_corp/12712.htm (retrieved September 9, 2013)

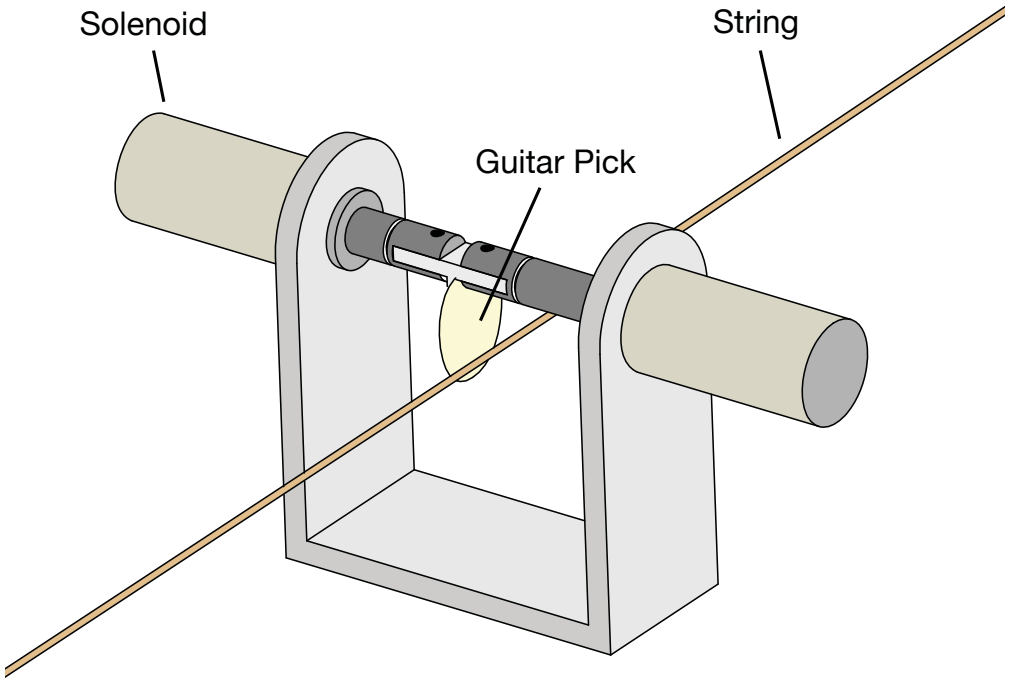


FIGURE 2.7: A solenoid-based picking system. Two solenoids are mounted perpendicular to the string. A pick is attached between the plungers. Upon actuation of one solenoid, the pick is pulled across the string toward the solenoid's plunger.

correspondence with note velocity commands, current systems can only play at a single loudness level.

Rotary motor-mounted pickwheels are a second method used to pluck strings on robotic guitars and bass guitars. [45] shows that these systems are capable of higher speed than solenoid-based systems. Baginsky's Aglaopheme uses a system capable of picking any one of its six strings. According to Baginsky's website, "a short piece of cable tie serves as a very durable plectrum that is moved across the strings by a linear motor to pick selected strings. The entire plectrum-drive assembly can be pulled away from the strings by a solenoid to allow the plectrum to drive to the desired strings without hitting any others"[25]. While Baginsky's approach allows for only a single string to be plucked at once, Eric Singer's GuitarBot uses a pickwheel mechanism on each string, allowing for chords to be picked simultaneously.

[25] http://www.baginsky.de/agl/agl_index.html (Retrieved September 11, 2013)

2.2.4.3 Exotic Robotic Guitars

Rather than attempt to emulate the normal methods of guitar playing, some musical roboticists have explored alternative actuation techniques. Researcher Aengus Martin has published work relating to an augmented guitar whose feedback is automatically adjusted [46]. Instead of plucking the strings, the composer adjusts the amount of feedback from each string through its amplifier. More recently, Sam Ferguson, Andrew Johnston, and Martin have employed novel corpus-based calibration methods in their systems [47], allowing for relatively rapid calibration of the highly nonlinear instrument through the use of pre-performance routines. This work appears to mirror a goal of the research presented in this thesis: increased compositional expressivity in new augmented instruments through the use of novel control schemes.

Additionally, musical interface pioneer Sergi Jordá created Afasia, a "one-man multimedia-band" featuring several robotic instruments [48]. Afasia's electric guitar robot utilizes an array of pneumatic valves which tap the strings, creating an effect similar to electric guitar tapping popular in rock music subgenres. While the focus of this work lies in more traditional playing techniques, the exotic approaches pursued by Jordá and Martin could be used to enhance future robotic guitars, extending their expressivity and timbral range.

2.3 Expressive Musical Robots

The aforementioned recent developments in mechatronic musical instruments indicate that it is a diverse field, populated with workers whose motivations range from robotic ethnomusicology to research into artificial intelligence-driven human/robot interactions. Such a survey reveals a cross-disciplinary lack of instruments that afford composers extensive expressive control over their instruments: mechatronic drumming systems are largely limited to systems with one degree of freedom. Similarly, mechatronically-augmented keyboard playing robots are typically restricted to piano-like instruments (while potentially expressive, it is only with relative difficulty that their inherent amplitude envelopes can be augmented). Also, while mechatronic chordophones possess a relatively wide variety of expressive parameters, these parameters are distributed among

a large number of unrelated systems; there exists a need to combine these and additional parameters into integrated systems.

With the need for additional parameters comes a need for improved human/robot interfaces. While most current mechatronic instrument ensembles use simple MIDI buses or networks that limit parametric mappings to one-to-one systems, improved interfaces can allow composers to more effectively utilise the new parameters available to them. The work presented in the remainder of this document seeks to address these issues.

Part I

Designing and Building Expressive Mechatronic Music Systems

Chapter 3

Designing and Building Expressive Robotic Guitars and Bass Guitars

In the course of the research presented in this thesis, two new musical robotic chordophones were completed: the guitar-like Swivel 2 and the bass-like MechBass. These two instruments were built with the shared primary goal of creating musical robotic chordophone systems with greater amounts of expressive parametric control than in previous systems (such as those described in Section 2.2.4). To achieve such expressivity, the robots' subsystems were built such that they were endowed with further degrees of freedom than in many existent mechatronic guitars and bass guitars.

The following two sections detail the design, construction, and performance evaluation of Swivel 2 and MechBass. The primary subsystems of each robot are detailed: the means by which string pitch is changed and string picking and damping events are performed are focused upon, and the electronics and modularity of each robot is described. This chapter centers on the physical implementation of the robots; applications taking advantage of the robots' expressive parameters are presented in Chapters 6 and 7.

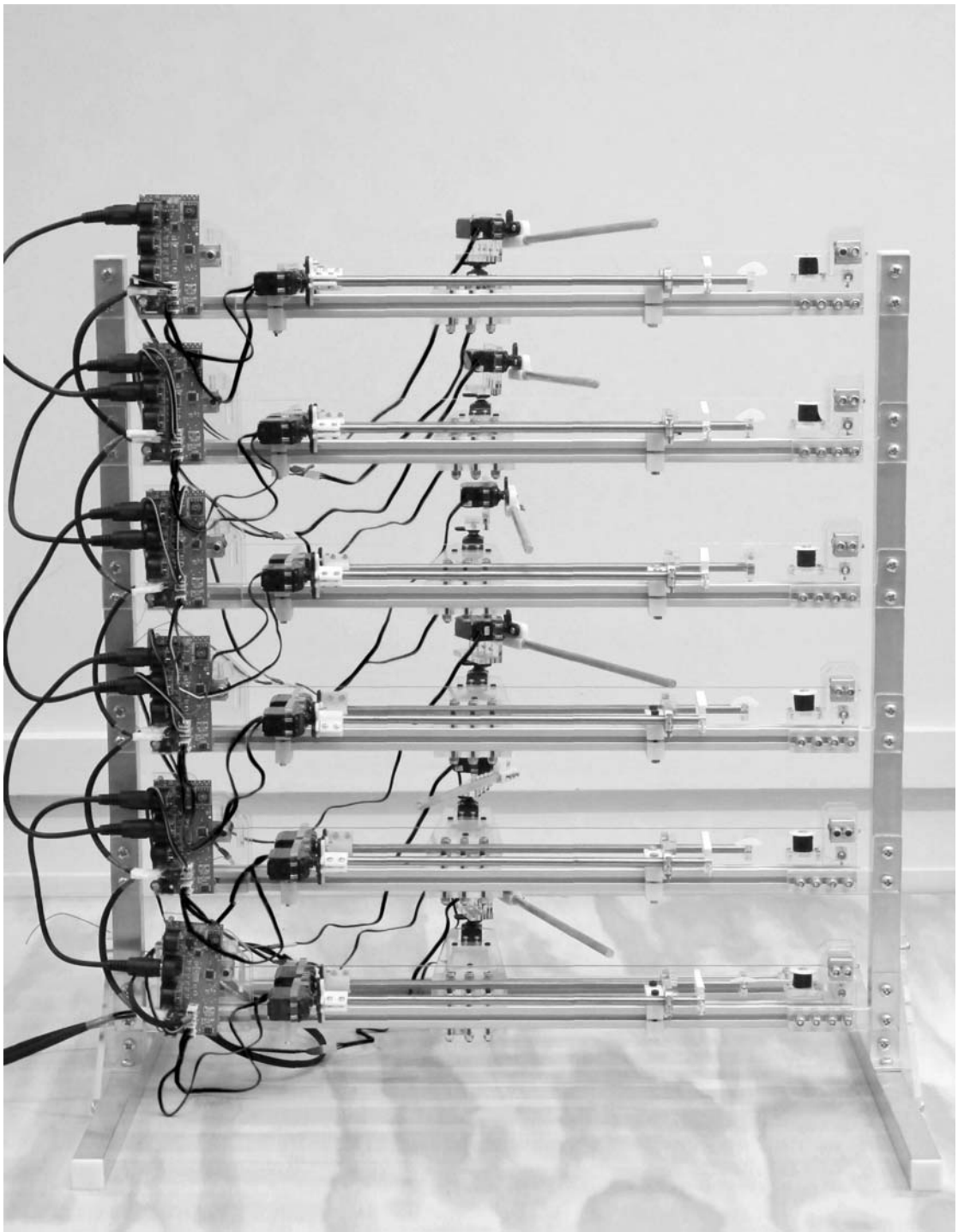


FIGURE 3.1: Swivel 2

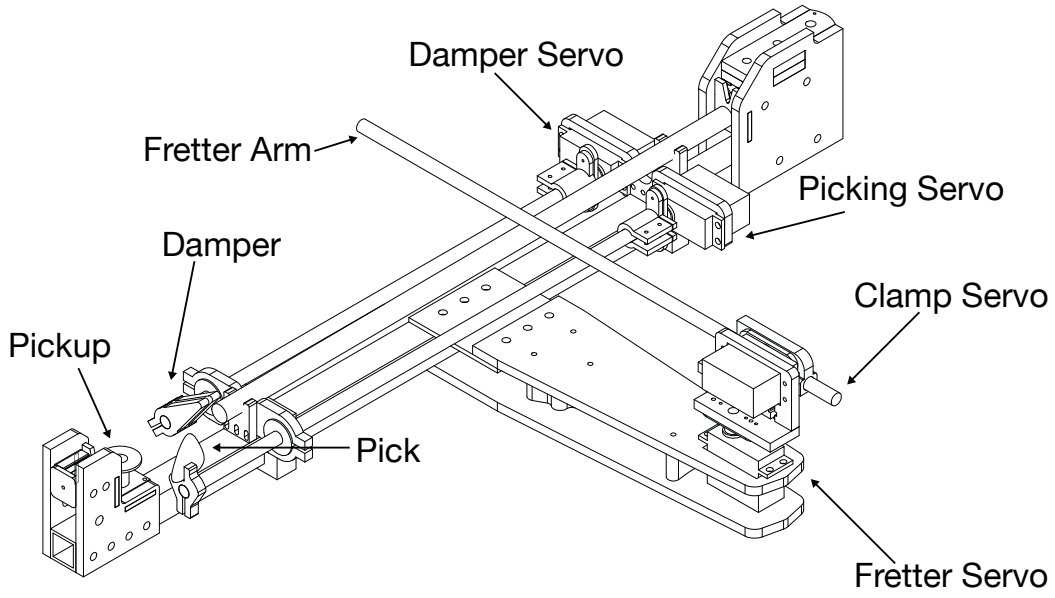


FIGURE 3.2: A Swivel 2 module with major subsystems labeled.

3.1 Swivel 2: Systems, Design, and Construction

Swivel 2 is a relatively lightweight, inexpensive musical robot: due to the number of actuators needed, complicated and heavy stepper motor drives (as used in the prototype discussed in [49]) were foregone in favour of miniature RC-type servomotors for string fretting and picking. These RC servos, further described in [50], function by interpreting a pulse-position modulation (PPM) signal. The PPM signal corresponds to the servo's shaft angle, resulting in a wide range of relatively fine control; compared to solenoid actuators, RC servos allow for a high degree of positional control and the potential for increased musical expressivity.

Each of the six Swivel 2 modules (a drawing of one of which is shown in Figure 3.2) are, in essence, monochords whose vibratory behaviour can be modified by four RC servomotors: one to pick the string, one to damp the string, one to change the string's pitch, and one to vary the force with which the pitch shifter contacts the string. Each string is mounted on an aluminium neck; the bridge, tailpiece, nut, and pickup mount are made of laser-cut acrylic. The actuators are controlled by a microcontroller able to

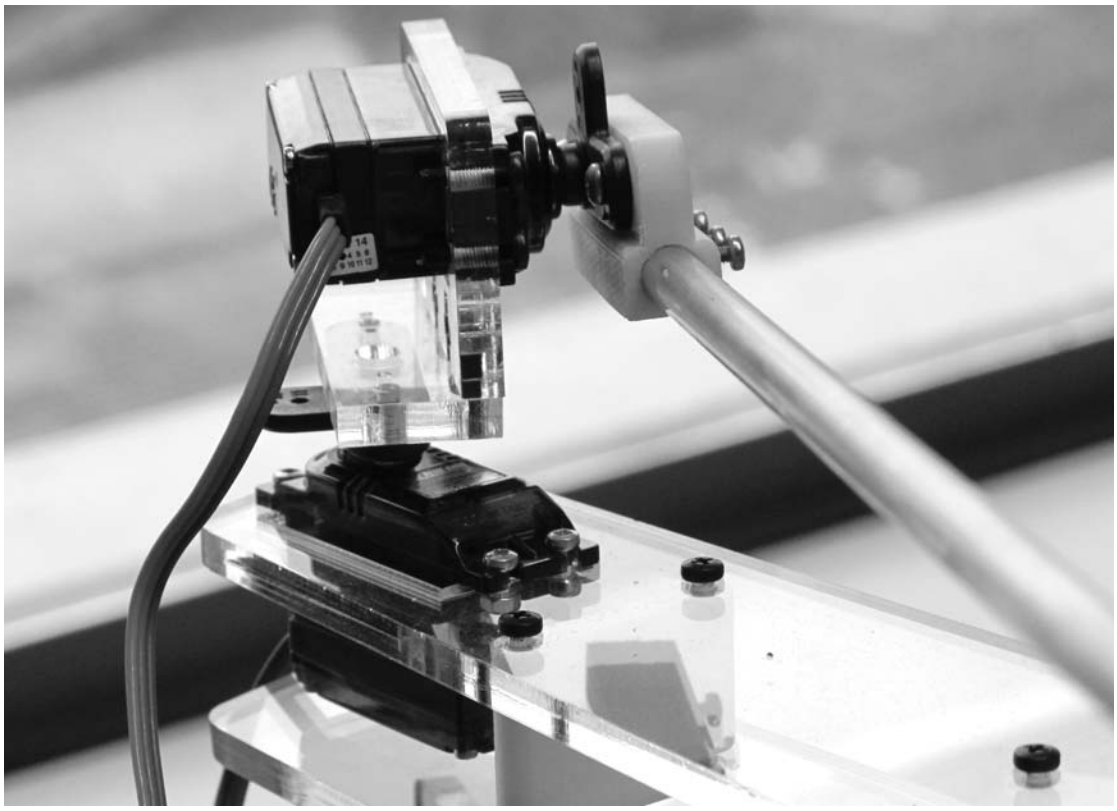


FIGURE 3.3: Swivel 2's fretter system consists of two servos: a fretter and a clamper. The fretter is the lower servo, and rotates the slide and clamper servo along the string's length. The slide, attached to the clamper, can be brought into contact with the string.

receive instructions from a master computer. The following subsections will discuss each of these subsystems in greater detail.

3.1.1 Changing the String's Pitch

Swivel 2 uses a rotary slide mechanism to change the string's pitch. The slide mechanism consists of two servomotors: a fretter servo to move the slide along the string and a clamper servo to vary the force of the slide against the string. This "fretter" assembly is shown in Figure 3.3. The fretter servo's plane of rotation is parallel to the string, allowing it to rotate the clamper servo and slide mechanism along the string's length. The clamper is mounted perpendicularly to the fretter servo's shaft; the aluminium slide is mounted to the clamper servo's shaft in a 3D printed clip. When the clamper servo lowers the slide against the string with sufficient force, the string will vibrate between the fretter and the bridge, altering the string's vibratory frequency.

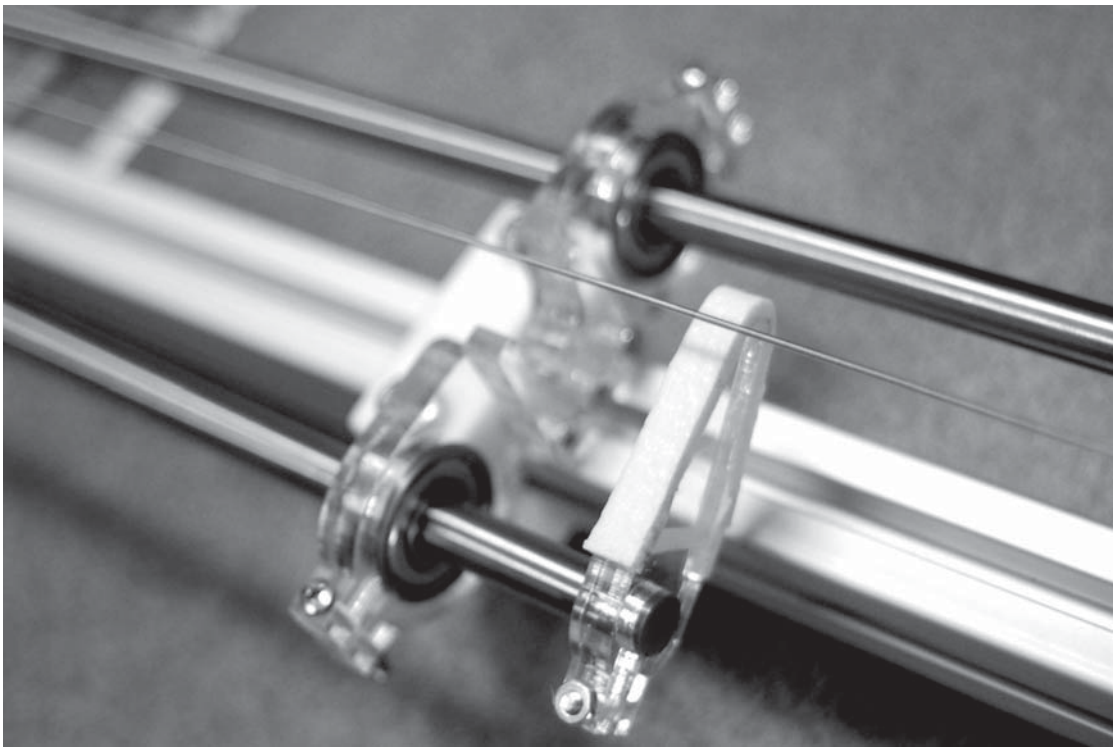


FIGURE 3.4: A damper on a Swivel 2 module. The damper, covered in foam, can be rotated against the string. The damper servo (not shown) is connected to the damper end effector by an extension shaft; this extension shaft is supported by a ball bearing.

Swivel 2's fretting mechanism allows for two approaches to pitch shifting: a continuous or discrete shift. A continuous shift enacts a portamento and is accomplished by maintaining the slide's contact with the string as the fretter servo is moved. A discrete pitch shift requires the slide to be disengaged from the string prior to the fretter servo rotating the fretter assembly between notes. Expressively, this is viewed as an advance over many prior systems that are capable only of discrete steps or portamento-only pitch shifts: where prior systems are restricted to either discrete or continuous shifts, Swivel 2 can conduct either.

3.1.2 Damping the String

To dampen string vibrations on Swivel 2, a damper mechanism can be brought into contact with the string. The damper mechanism consists of a padded damper end effector, shown in Figure 3.4. The end effector is attached to an extender shaft which is in turn connected to the standard-sized damper servomotor. Unlike the solenoid-based damper mechanisms used on many existent guitar systems (discussed in Section 2.2.4),

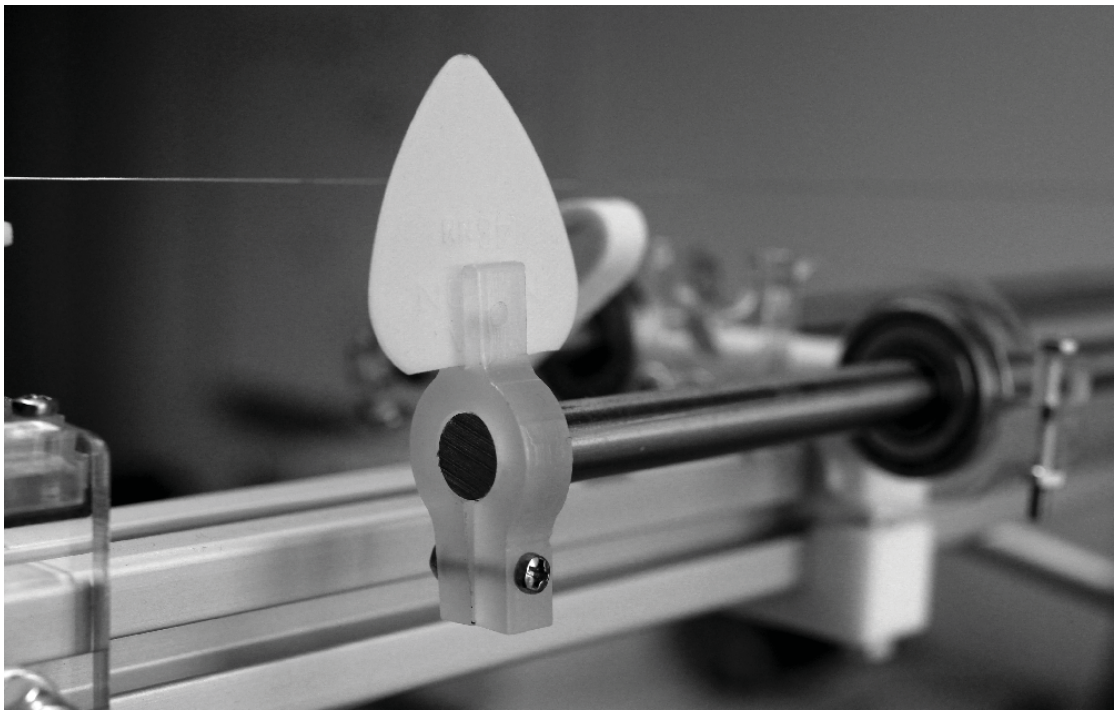


FIGURE 3.5: The Swivel 2 picking mechanism: a guitar pick is attached to a rotating shaft which is connected to an RC servomotor; the motor's reciprocating rotation drags the pick back and forth across the string.

Swivel 2's damper can apply variable pressure to the string, allowing composers to specify the rapidity with which the string's vibrations diminish.

3.1.3 Picking the String

Swivel 2's string picking system consists of a single standard-sized RC servomotor whose driveshaft is connected to a bearing-supported shaft extender, as shown in Figure 3.5. A laser cut acrylic pickholder is attached to the shaft extender; a standard guitar pick can be inserted into the pickholder. The use of the shaft extender allows for the picking servo to be mounted far from Swivel 2's magnetic pickup, out of the range where the electromagnetic signals generated by the servomotor could create undesirable interference on the pickup.

Upon receiving an instruction to conduct a pick event, the picking servo rotates through an arc whose size can be varied depending on the chosen pick's thickness: thicker picks deform less upon contact with the string and therefore require a smaller arc than thinner picks. The pick moves in a reciprocating manner: each pick event moves in a direction opposite to the previous one, dragging the pick back and forth across the string.

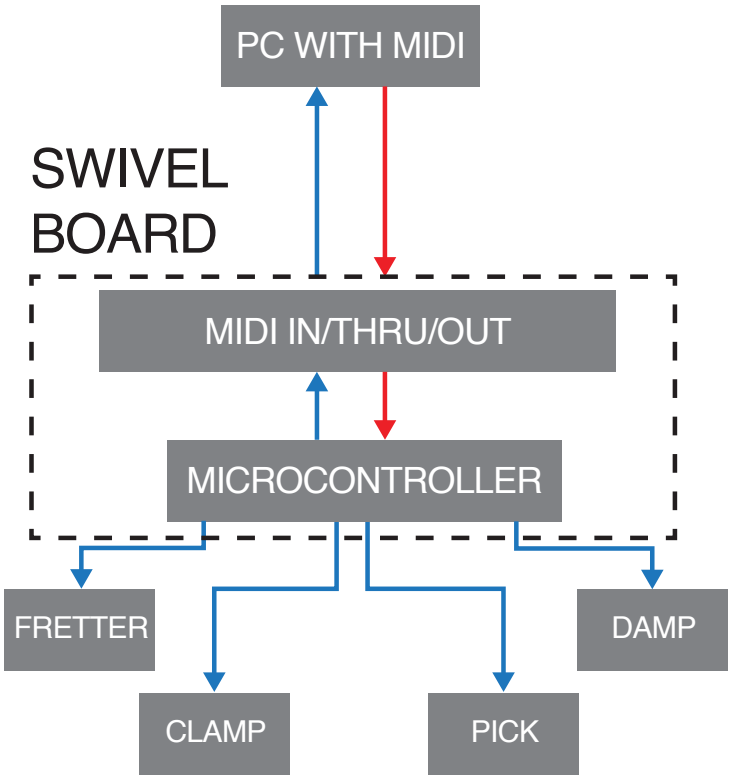


FIGURE 3.6: A block diagram of the Swivel Board.

3.1.4 Electronics and Firmware

Each Swivel 2 module communicates with its host device using the MIDI protocol. The modules are each equipped with a purpose-designed printed circuit assembly (called the Swivel Board, shown in Figure 3.8) capable of receiving MIDI signals, parsing them, and outputting PPM signals to each of the module's actuators. A block diagram of the Swivel Board is shown in Figure 3.6. For ease of programming, an ATMEGA328 microcontroller was chosen: this microcontroller is programmed in the easy-to-use C-based Arduino programming environment[1].

The Swivel Board handles three main tasks: the receiving of MIDI messages, the retransmission of MIDI messages, and the parsing of relevant messages and corresponding PPM output to the module's servomotors. To receive MIDI messages, the Swivel Board implements the MIDI hardware specification[2]. A conventional MIDI cable can be plugged into the Swivel Board, which is connected via the MIDI support circuitry to the UART

[1] http://arduino.cc/ (retrieved October 17, 2013)
[2] http://www.midi.org/techspecs/electrispec.php (retrieved October 29, 2013)

Table 3.1: The Swivel Board's interpretation and mapping of MIDI messages

Message Type	Action	Message Range	Mapped Range
Pitchbend	Fret (Pitch shift)	-8192 - 8191	0-790
CC 7	String pick	0 - 127	0-40
CC 8	String clamp	0 - 127	0-30
CC 9	String damp	0 - 127	0-10

input on the microcontroller. The Arduino MIDI library[3] is used to handle incoming MIDI messages. MIDI messages can be retransmitted either using the MIDI Thru port or the MIDI Out port on the Swivel 2 board.

Each of Swivel 2's actuator functions can be initiated in response to specific MIDI messages. While MIDI messages have a preset range (typically 0-127 for normal messages and -8192 to 8191 for two-byte pitchbend messages), their values have been mapped in software to restrict their range. The reason for this mapping is to prevent the actuators from being instructed to travel outside of their functional region. Such restriction of travel range is critical on Swivel: the system's RC-type servomotors will attempt to move to any specified position, using their proportional feeback to continue applying power to the motor until it reaches the specified destination. If instructed to move to a region made unreachable by an obstacle (such as the robot's chassis or string), the servomotor could be damaged. The MIDI message types, corresponding actions, normal range, and mapped range are shown in Table 3.1.

To control the servomotors, the Arduino Servo library[4] is used. The Arduino Servo library, described in more detail in [51], uses a hardware timer on the target microcontroller to output a PPM wave; on the Swivel Board, the duty cycle of this PPM wave is set in response to the incoming MIDI message.

The above functionality is implemented on the microcontroller in a manner illustrated in Figure 3.7. The microcontroller's firmware has three main operations: initialisation, message receipt, and message parsing and servo output. At initialisation, a small delay corresponding to the device's channel number is performed prior to powering the servos. The delay serves to prevent all of the modules' servos from initializing at once, which could lead to a large instantaneous current demand placed upon the system's power supply. After initialisation, the Swivel Board awaits the receipt of MIDI messages that

[3]http://playground.arduino.cc/Main/MIDILibrary (retrieved October 17, 2013)
[4]http://arduino.cc/en/reference/servo/ (retrieved October 17, 2013)

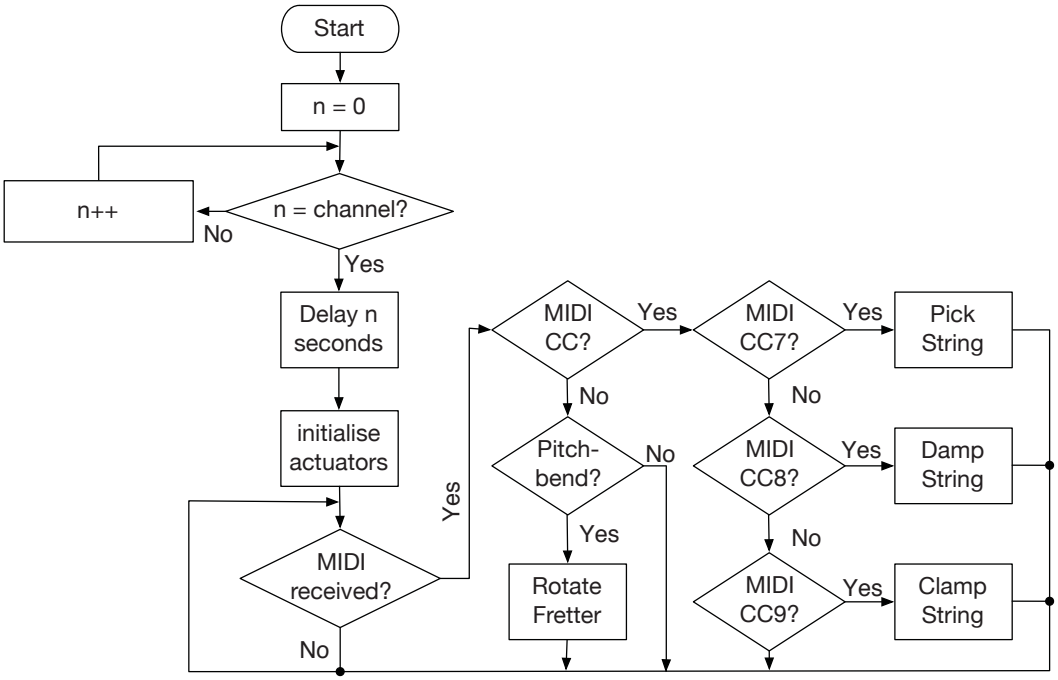


FIGURE 3.7: A Program flow diagram of the firmware on the Swivel 2 board.

correspond to its unique channel number. Upon the receipt of such a message, the message is parsed according to Table 3.1, and suitable values are sent to each servomotor.

3.1.5 Swivel 2: Modularity

Swivel 2 is a modular musical instrument. Each of the six string units in Swivel 2 can be made electronically independent of the others, and all six share a common MIDI bus. The modularity of Swivel 2 allows for flexibility in installation: the distributed actuator control and communications electronics allow for diverse options in setup, potentially differing greatly from the "default" rack-mounted setup discussed below.

As described in this chapter's preceding subsections, each Swivel 2 module consists of four servomotor actuators. As constructed, Swivel 2 contains six separate modules for a total of 24 actuators. To address the individual actuators on a specific module, each module is assigned to a separate MIDI channel (this channel is selectable from a 16-position switch on the Swivel Board). A MIDI cable is connected from the host device to Swivel 2's Channel 1 module's MIDI In port; a second cable connects the Channel 2 module's MIDI In port to the MIDI Through port of the Channel 1 module. This

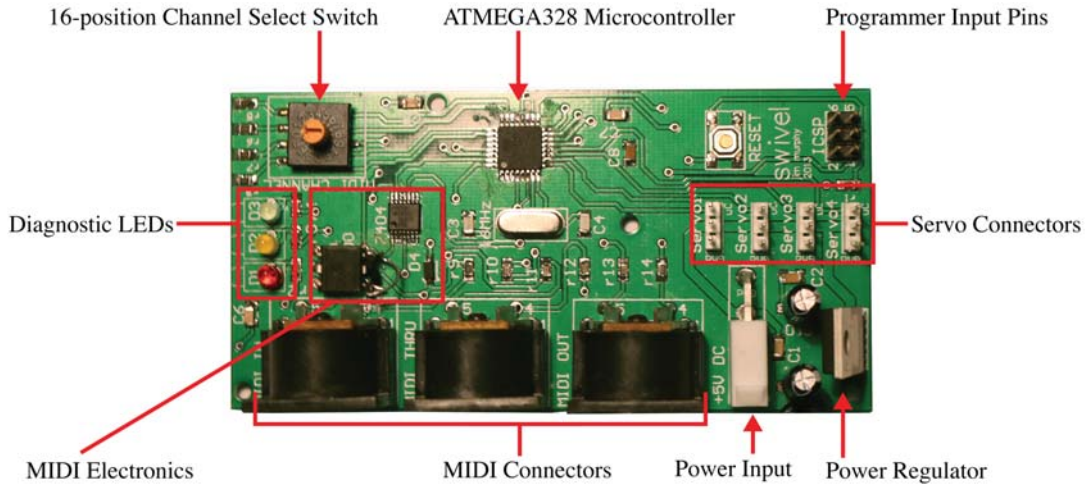


FIGURE 3.8: The Swivel 2 PCB assembly, with major subsections indicated.

process is repeated for all devices, forming a MIDI bus with signal retransmission at each node.

Each module is fitted with a standard steel guitar string; each of the six strings corresponds to a traditional guitar's string, with the strings' vibration frequency shown in Figure 3.9. Every module is equipped with a magnetic pickup to transduce string vibrations; the output from the six pickups is sent to a multiple-input PC audio interface or mixing board.

All six modules of Swivel 2 are mounted on an aluminium frame. This default configuration allows for portability, short cable lengths, and convenience in a laboratory environment. Due to the modular design of Swivel 2, though, alternative configurations can be employed in different gallery or performance settings. Further, additional Swivel 2 modules may be constructed and added to the bus with no modifications needed to the existent modules.

3.2 MechBass: Systems, Design, and Construction

MechBass [52] is a mechatronic bass guitar built by collaborator James McVay under the supervision of the author and the author's PhD supervisors. Like Swivel 2, MechBass was designed in concert with an overarching goal of this thesis: to afford composers greater expressive control over the instrument than had been available in prior works. In a similar manner to Swivel 2, such expressive control was achieved by increasing

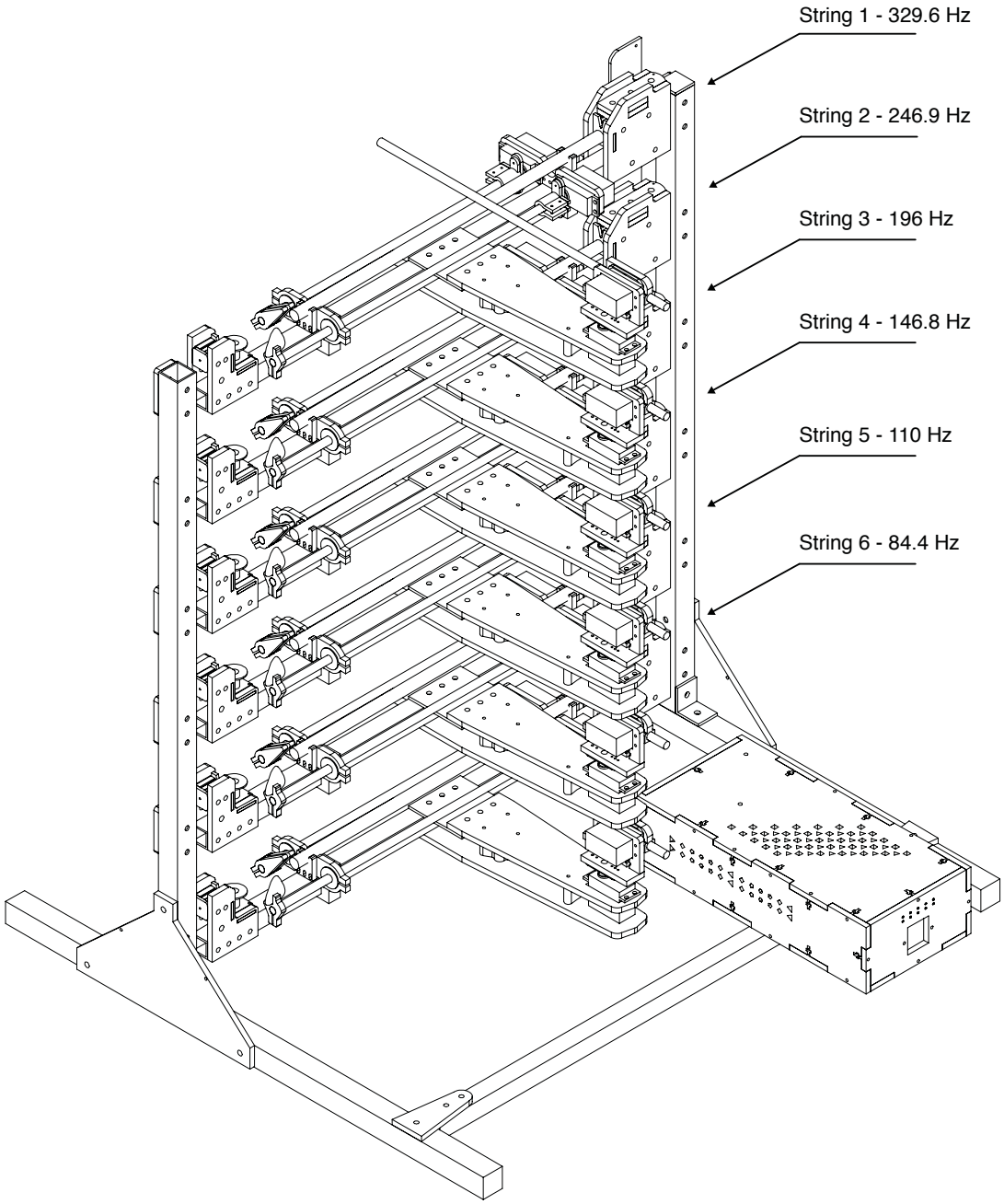


FIGURE 3.9: A drawing of six Swivel 2 units, with each module's string's frequency shown.

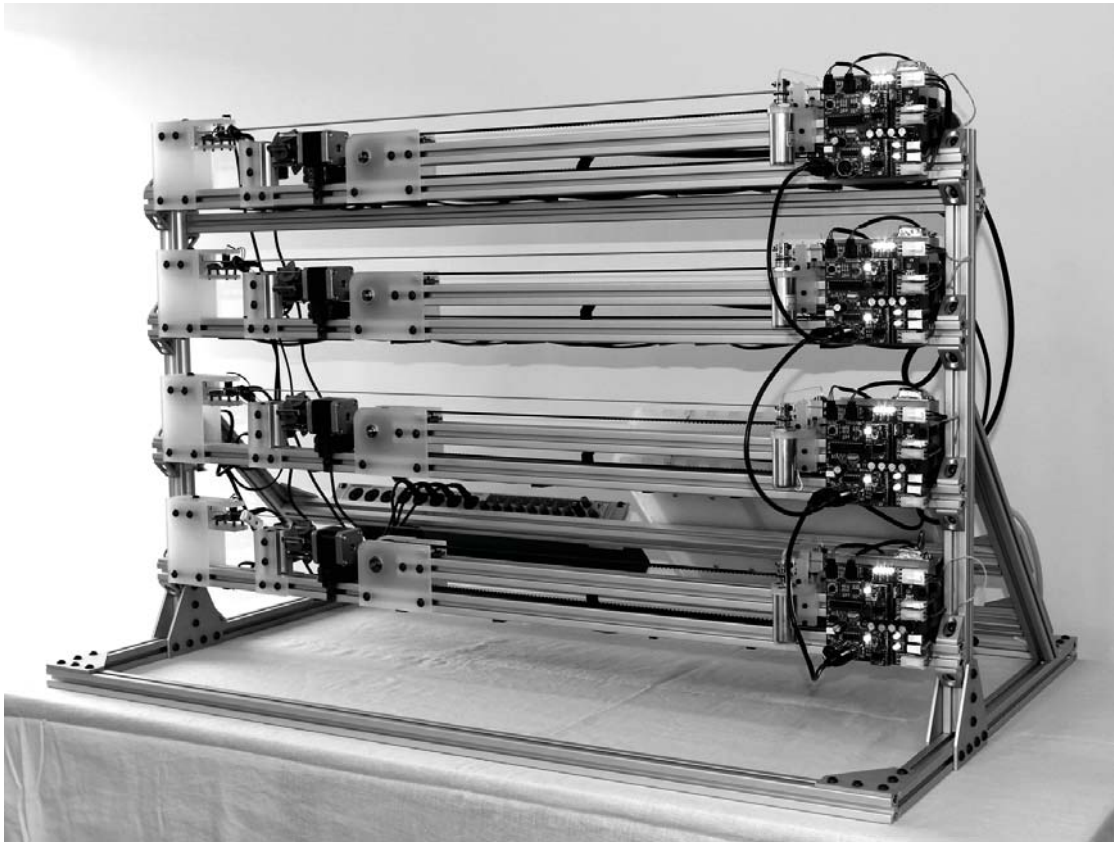


FIGURE 3.10: MechBass (Photograph by James McVay)

the degrees of freedom of the mechatronic systems: where possible, continuous or high-resolution parametric adjustments were chosen over discrete systems.

MechBass, shown in Figure 3.10, consists of four discrete single-string modules. These modules are connected to each other on a MIDI bus which is in turn connected to a MIDI host device. Each module contains three primary subsystems: a pitch shifter, a string damper, and a string picker. These systems will be described in detail below. Additionally, Section 3.3 presents a comparison between design decisions made on MechBass and Swivel 2.

3.2.1 MechBass Pitch Shifting

MechBass uses a belt-driven linear positioner to change the string's pitch. The linear positioner is driven by a NEMA-23 stepper motor connected to an XL-sized timing belt. The timing belt runs beneath the string and is connected to an idler pulley opposite the stepper motor. To change the string's pitch, the stepper motor moves the timing belt a

FIGURE 3.11: MechBass's pitch shifter.

specified number of steps; the motion of the timing belt moves an attached trolley along an aluminium linear slide.

The trolley functions as a movable fret with an accompanying clamping mechanism. Upon reaching the specified position, the linear positioning stepper motor stops turning and two trolley-mounted linear solenoids actuate, lowering a clamper against the string and pinching it between the clamp and the trolley-mounted fret. This action changes the vibratory length of the string, resulting in a pitch change.

The ability to clamp and release the string allows for faster, lower-friction note change events than similar systems whose fret mechanism remains in contact with the string. Additionally, composers are afforded more expressive control of note change events by having the option to not conduct a portamento between every pitch transition.

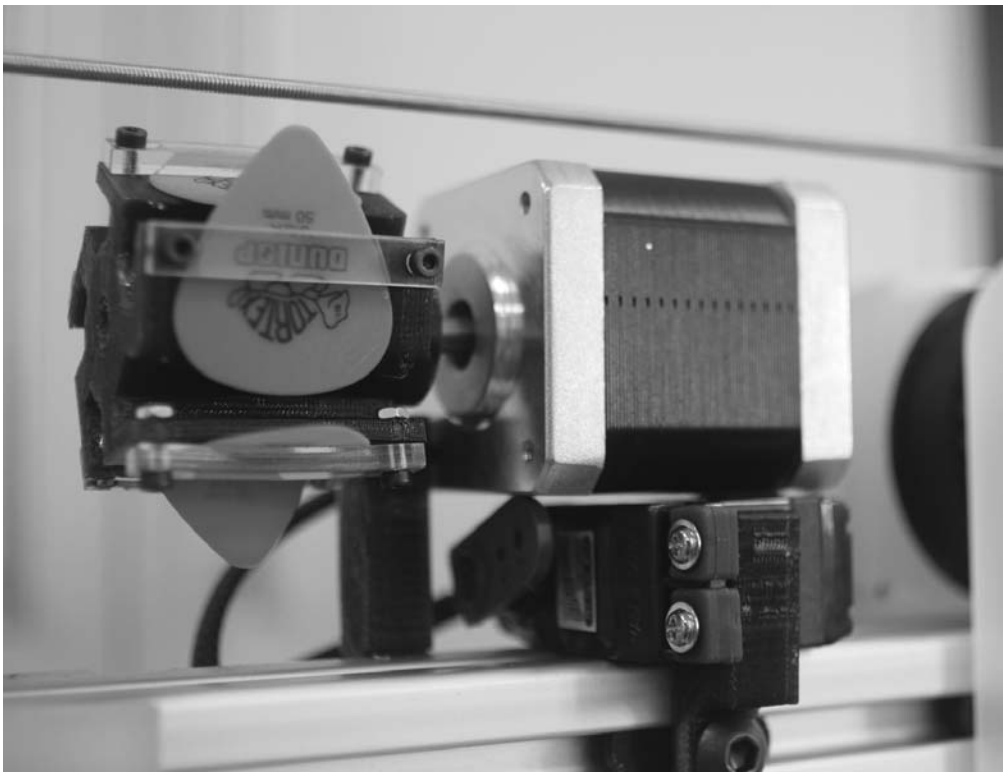


FIGURE 3.12: The string plucking mechansism on MechBass employs an RC-style servomotor which can pivot the stepper motor string picker. It is shown here in its at-rest position.

3.2.2 MechBass String Damping

String vibrations on MechBass modules are dampened in a manner similar to Swivel 2's approach. A felt-covered damper arm is attached to a standard-sized RC servomotor. The damper arm can be brought into contact with the string; as on Swivel 2, the damper mechanisms on MechBass have the potential to be pressed against the string to varying degrees. This variable damping degree lends the damping action more potential expressivity than is afforded by the oft-used dual-state solenoid-based dampers on many existent mechatronic guitar systems (discussed in Section 2.2.4).

3.2.3 Variable-loudness String Picking

After an evaluation of string picking techniques in [45], a rotary string picking system was chosen for MechBass. While mechanically more complicated than the reciprocating picker used in Swivel 2, the single-direction rotary string picker in MechBass offers two

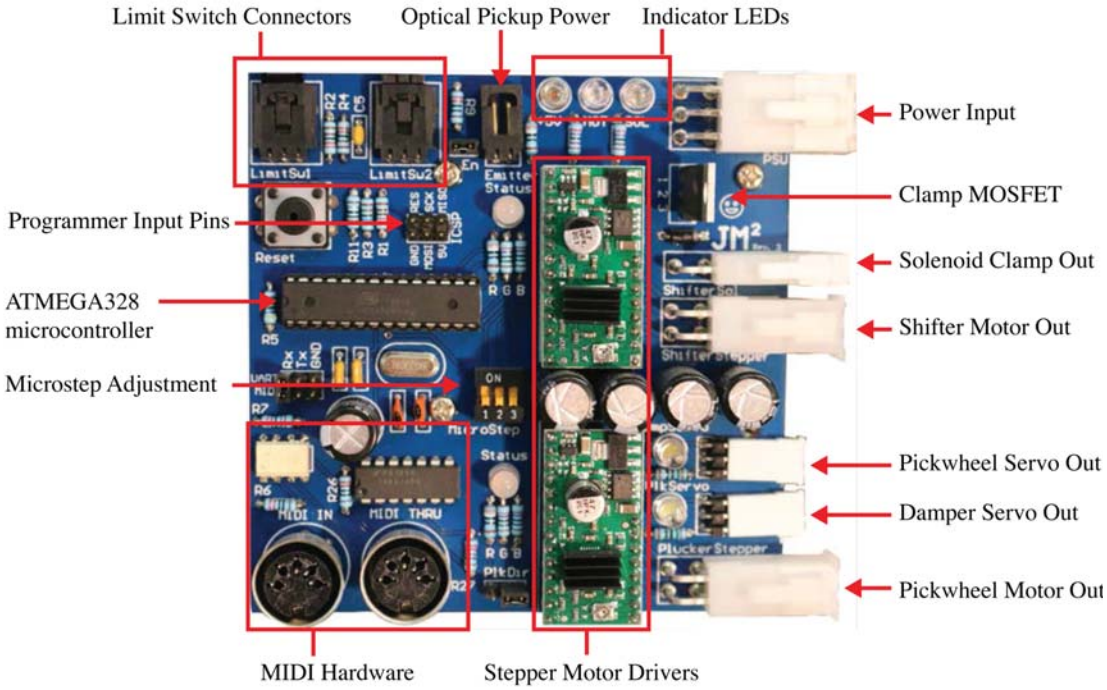


FIGURE 3.13: The JM2 board, with input, output, and major subsections indicated.

advantages: high speed and uniformity of pick sound due to the picks running across the string in a single direction.

While some other existent robotic guitars use rotary string picking systems, none have the ability to change the pick event's intensity. In an effort to add a further dimension of expressive potential to MechBass, a system was developed to allow for the pickwheel to be raised and lowered, changing the picking mechanism's power and loudness against the string. First prototyped by Kapur and Carnegie's honours student Richard Vindriis in [45], it consists of a stepper motor with a pick wheel attached. The stepper motor is constrained by a hinge at its lower corner and can be pivoted on its hinge with a cam attached to a standard-sized RC servomotor.

The note picking routine for MechBass begins with an instruction to raise the RC servomotor's cam to a position corresponding to the desired loudness of the note. After a small delay to allow for the servo to finish its travel, the stepper motor is actuated with $360/(A_s * N_p)$ steps, where A_s is the stepper motor's angle per step and N_p is the number of picks on the pick wheel. This ensures that only one pick per actuation event will strike the string.

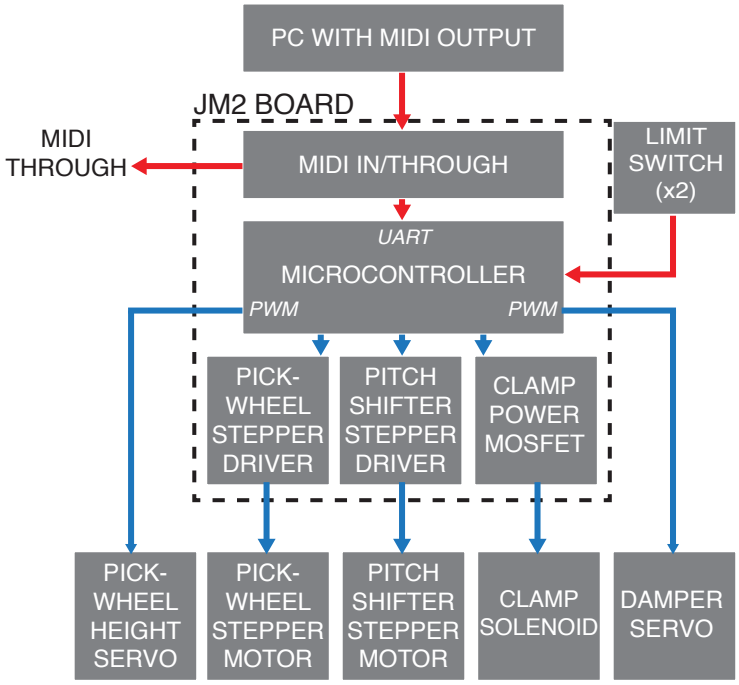


FIGURE 3.14: A block diagram of the JM2 Board and associated connections.

The pick wheel on MechBass is 3D printed and contains sockets for five guitar picks. Picks are clamped against the pickwheel with laser-cut brackets; while the pick clamps can accommodate multiple pick thicknesses, MechBass normally uses picks with a 0.5 mm thickness.

3.2.4 MechBass: Electronics and Firmware

A purpose-built printed circuit assembly is implemented on MechBass. The assembly, dubbed the JM2 Board, contains an ATMega328 microcontroller, a full MIDI implementation, actuator drivers, and limit switches. Figure 3.14 diagrams the inputs, outputs, and on-board systems of the JM2 Board.

In keeping with the communications protocol employed by the other robots built in this thesis and by collaborators, the MIDI protocol is used by MechBass. Each JM2 Board is assigned a MIDI channel; separate MIDI messages are sent to each channel. The MIDI message types corresponding to the MechBass output actions are shown in Table 3.2.

The bus topology of MIDI communications systems allows for additional MechBass modules to be easily added to the system. To add a fifth string to the currently four-stringed

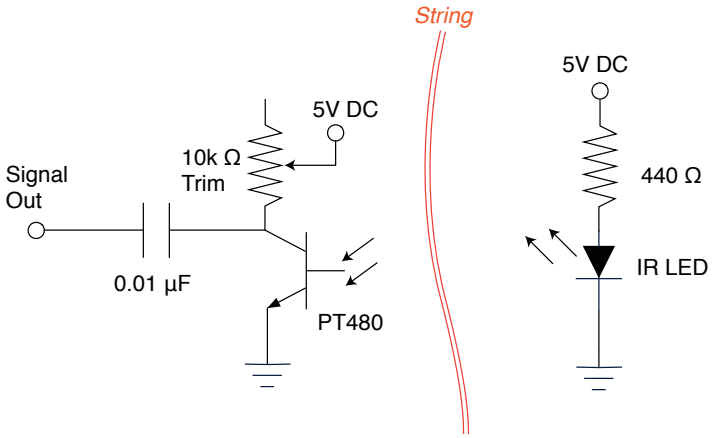


FIGURE 3.15: The optical pickup as used on MechBass. The string's vibrations alter the amount of light from the IR LED allowed to reach the phototransistor.

TABLE 3.2: The JM2 board's interpretation and mapping of MIDI messages

MIDI Message	Action	Message Range	Mapped Range
NoteOn (Note)	Move Fretter, Pick String	0-127	0-127
NoteOn (Velocity)	Rotate Pickwheel Servo	0-127	0-20
NoteOff	Engage Damper	0-127	0-10

MechBass, for example, the new string's JM2 board would simply be programmed to listen for messages broadcast on MIDI channel 5. The communications scheme employed by MechBass is quite similar to that of Swivel 2 and was inspired by Trimpin's mode of communicating with individual string systems [19]).

MIDI messages to each MechBass module are interpreted by the JM2 Board's microcontroller, which subsequently outputs corresponding actuator control signals. Each MechBass string contains six actuators: two clamp solenoids, a NEMA 23 pitch shifter motor, a NEMA 17 pick wheelmotor, a pickwheel velocity adjustment servo, and a damper servo. PPM signals from the microcontroller determine the angle of the velocity adjustment and damper servos. The clamper solenoids are switched by a power MOSFET connected to an output pin of the JM2 Board's microcontroller. Both stepper motors are driven by Allegro A4988 stepper motor driver boards connected to the microcontroller.

MechBass string vibrations are converted to electrical signals by an optical pickup (shown in Figure 3.15); the use of an optical pickup allows for proximal mounting of motors whose electromagnetic interference would disrupt traditional wound-coil guitar pickups.

While novel to mechatronic music scholarly literature, the pickup is similar to those used by Trimpin's guitar sculptures [19], and is evaluated in Section 3.3.3.

3.3 Swivel 2 and MechBass: Performance Evaluation

To understand the musical capabilities of a complicated musical robot, it is useful to perform evaluations of its constituent parts; doing so allows users to become aware of the robot's abilities and limitations and to work within them. Parameters deemed important in the subsystems in both Swivel 2 and MechBass have been evaluated and are discussed in the following subsections.

3.3.1 String Picking Evaluation

To determine the speed, latency, and (in the case of MechBass) dynamic range of the picking systems, the robots' audio output was recorded during a variety of picking actions. The audio was recorded at a sample rate of 44.1 KHz; all FFT operations were performed with a block size of 16,384 samples and Hann windowing. Swivel 2 was found to have a picking rate of 410 picks per minute; MechBass, with its faster and more complicated pickwheel, can pick at a rate of 520 picks per minute. Faster actuators and thicker picks are likely to be able to speed up both robots' picking performance.

The high picking speed of MechBass is due to its multiple picks: to pick the string rapidly, the pickwheel motor needs only to be rotated at the desired rate. The use of multiple picks requires each pick to be uniform: a lack of uniformity in displacement or thickness results in inconsistent pick intensities between pick events.

The performance of the adjustable-height pickwheel is shown in Figure 3.16. To test the adjustable pickwheel, 30 pick events at six evenly-spaced height positions were recorded. The average RMS of the signal was taken and converted to Decibels. As Figure 3.16 shows, MIDI velocity values between 2 and 52 result in relatively consistent pick power increase; After a MIDI velocity of 52, the string is overdriven, resulting in little change as MIDI values are increased. A MIDI velocity of 1 results in no pick (allowing for fretter pre-positioning with no accompanying pick event). Musically, these results mean that composers can work with MIDI values between 2 and 52 for an expressive volume change

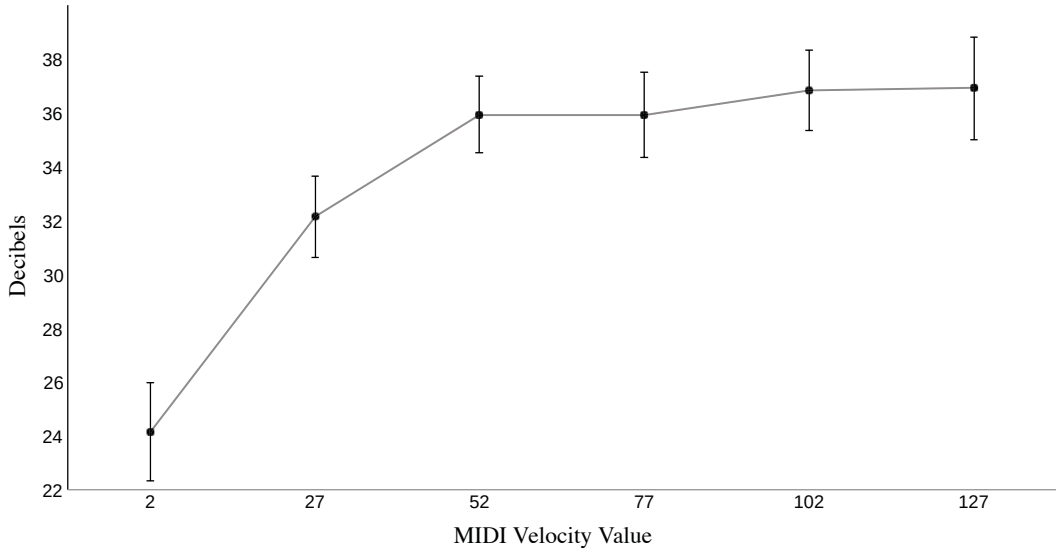


FIGURE 3.16: MechBass Pick intensity at varying MIDI velocity values. The error bars represent the standard deviation over five trials.

with a linear response and with values greater than 52 for a saturated, overdriven sound. Using techniques described in Chapter 7, this response could be linearised in software if desired by users.

3.3.2 Pitch Shifting Performance

Two performance metrics are crucial in determining mechatronic guitar pitch shift performance: speed and repeatability. As each pitch change requires travel time along the string's length, faster pitch shift events are desirable.

The ability for a mechatronic guitar instrument to rapidly change pitch in a repeatable fashion is paramount to the instrument's overall usability. The slow performance of Swivel 1.0 (discussed in [49]) motivated many of the revisions present in Swivel 2.

Fretting speed was measured by instructing the robot to play a single octave consisting of half of the string length. The time taken to move from a home position to a position one octave above that was timed and is reported below. Incoming audio was recorded to a PC and was analyzed in software with 1 ms resolution.

For Swivel 2, the average octave shift time of five pitch shift evaluations was 82 ms with a standard deviation of 1 ms. The most rapid half-string octave shift on MechBass was

found to be 360 ms with a standard deviation of 1 ms. The speed difference between Swivel 2 and MechBass is likely due to two main factors: the smaller size of Swivel 2 and the reduced angle through which Swivel 2's fretter servo needs to rotate to perform the pitch shift.

To measure the pitch shifters' repeatability, the robots' fretters were sent from a random position to a given position. The string's frequency was then recorded and analyzed on an audio tuner with 0.1 Hz resolution. On Swivel 2, five trials returned the same result: from each position, the fretter returned to the given position (which corresponded to 151 Hz) with a standard deviation of 0.5 Hz. In evaluating the fretter accuracy on MechBass, five trials were averaged; no pitch deviation was found greater than five cents from the desired note.

The novel rotary motion pitch shifter employed by Swivel 2 has been found to be suitable for high-speed pitch shift events with good accuracy; for very precise performance, though, the higher-resolution linear actuator on MechBass is desirable.

3.3.3 Pickup performance

Swivel 2 and MechBass use different pickup styles: Swivel 2 uses hand-wound magnetic pickups (consisting of a spool of 0.25 mm enamel-coated wire wrapped around a rare earth magnet core); MechBass uses optical pickups, discussed further in Section 3.2.4.

To evaluate the pickups' performance, a string was picked and the resultant waveform output by the pickup was recorded at a sample rate of 44.1 KHz. The waveform was visually evaluated for signs of clipping and distortion. Sample waveforms are shown in Figure 3.17.

Upon evaluation, the optical pickup was deemed suitable for performance, research, and installation use. Significant pre-performance calibration is required: the emitter and receiver need to be positioned precisely in order to allow the correct amount of light to be received by the phototransistor. Further work on easily calibrated optical pickups, perhaps similar to [53], will reduce the setup time needed for adequate optical pickup performance.

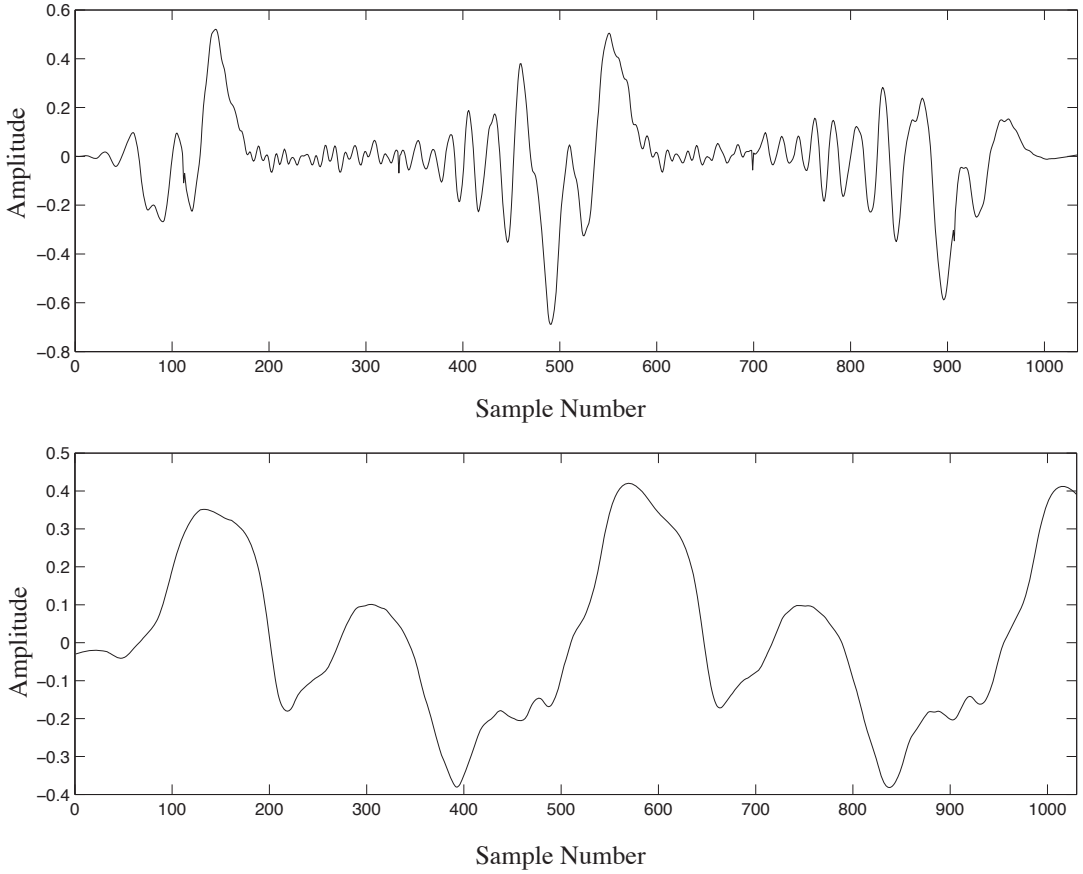


FIGURE 3.17: Pickup waveforms: the top waveform is recorded from Swivel 2; the bottom waveform is recorded from MechBass.

The magnetic pickups used in Swivel 2 are easier to configure than the optical pickups on MechBass. However, they are sensitive to actuator noise and must be placed relatively far from any servo, solenoid, or stepper motor.

3.4 Expressive Mechatronic Chordophones

Mechatronic chordophones are a challenging sub-discipline of musical robotics, requiring the integration of many subsystems in order to produce musical output events. This requirement for subsystems integration has resulted in many prior works including a number of expressive subsystems while foregoing others. The work presented in this chapter has integrated as many expressive parameters from prior works as possible into two new mechatronic systems, allowing composers more control over the instruments' output than can be found in prior systems.

In addition to synthesising prior findings into integrated instruments, this chapter's work introduces a number of novel features. Firstly, a rotary-motion fretting mechanism, developed for Swivel 2, has been described and evaluated. This rotary fretter allows for low-parts-count high-speed positioning along the strings' lengths. Additionally, both mechatronic chordophones are equipped with novel "clamping" mechanisms, allowing the fretter to disengage from the string when commanded. A third novel actuator configuration deployed on MechBass is the instrument's variable-height string picking mechanisms. These mechanisms allow the picks to be raised and lowered in response to a user's commands. Finally, the use of optical pickups on mechatronic instruments is introduced. Novel to the academic literature, this use of optical pickups allows for the placement of noisy actuators in close proximity to the instrument's pickup.

Chapter 4

Designing and Building a Mechatronic Harmonium

4.1 Kritaanjli: A Robotic Harmonium

While the other instruments developed in the course of this thesis allow for the realisation of relatively complicated melodic and rhythmic compositions, they are not as practical for harmonically-dense compositions or those works requiring complicated dynamic envelopes on each note. To address these shortcomings, a harmonium capable of user-defined amplitude envelopes was mechatronically augmented. The resulting instrument, named Kritaanjli (meaning "Two Hands Praying"), is shown in Figure 4.1. As many of the author's and authors' collaborators' prior works are percussive (such as those described in [6]), Kritaanjli allows composers to use a less rhythmic instrument while remaining conceptually within the Indian-themed instrumentation of the KarmetiK Machine Orchestra. In essence, a motivating factor in the decision to create a mechatronic harmonium is to provide the percussion-heavy KarmetiK Machine Orchestra with an instrument capable of producing tonal sounds, dynamically-varying drones, and harmonic content.

A harmonium is a reed organ associated with Indian folk and devotional music, often used as a pedagogical tool or accompaniment for vocalists. Typical harmoniums have three means by which a player interfaces with the instrument: the keyboard, a

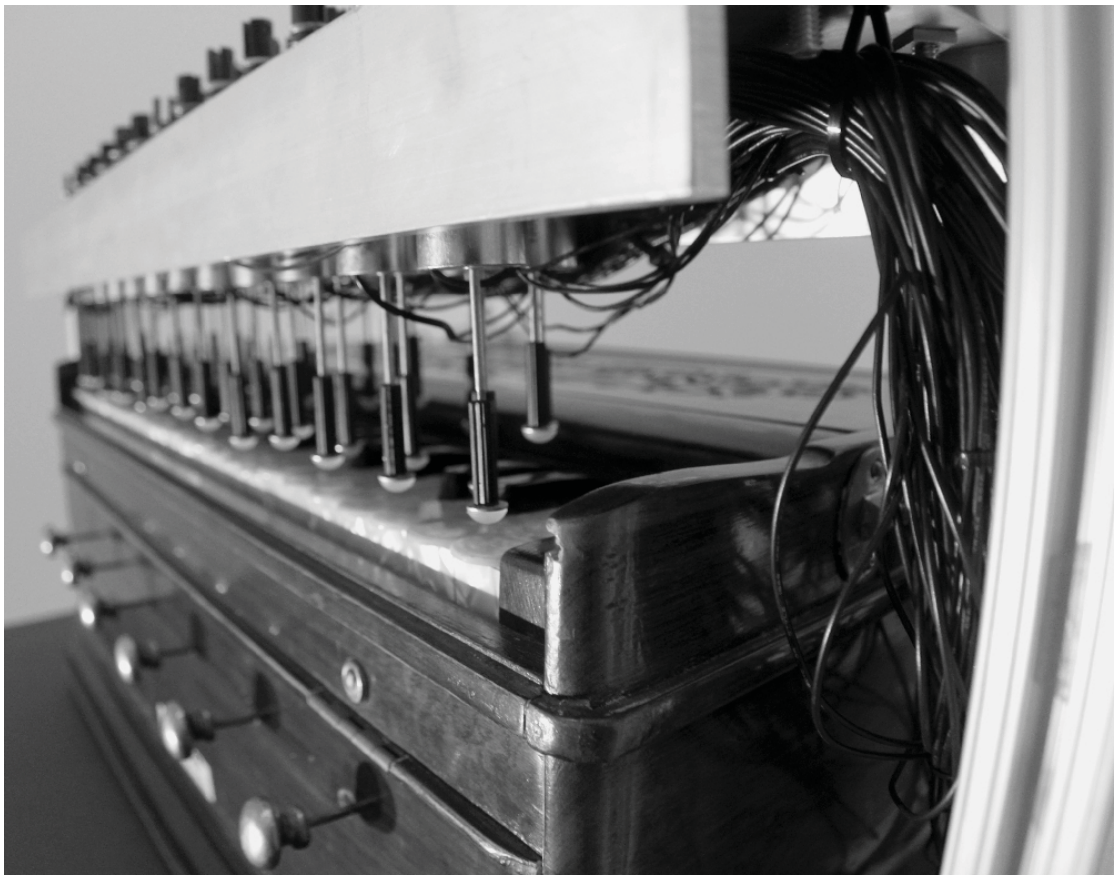


FIGURE 4.1: Kritaanjli: 44 solenoid actuators are positioned above each key on the harmonium.

bellows mechanism, and an array of organ stops. In this chapter, the keyboard and bellows pumping mechanisms will be augmented mechatronically. During play, the player presses keys with one hand while pumping the bellows with the other: the keys serve as momentary switches, allowing the air pumped by the bellows to be forced past reeds opened by the key-press events. While different harmoniums use different bellows configurations, many harmoniums feature a hinged bellows assembly which is opened and closed by the player: dynamics may be adjusted by pumping with varying intensity and stroke. Finally, the organ stops can be employed to activate additional reed banks, changing the timbre of the sound.

To augment an existing harmonium for mechatronic actuation, the aforementioned human interfaces must be coupled to actuators capable of interacting with them in a manner similar to that of a human. Such augmentation presents challenges that were addressed in the design of Kritaanjli: the harmonium is a fragile instrument designed to be played by a trained human musician. Any actuator placement must be conducted

in a manner to avoid inflicting damage or undue wear and tear on the instrument. The harmonium's bellows pump, for example, is hinged by a lightweight brass clip, easily bent or otherwise damaged by pump events too rapid or with excessive stroke length.

An additional challenge arose in the design of Kritaanjli: the rough-hewn nature of the hand-built instrument presented problems related to actuator alignment. For example, in building the keyboard-playing assembly, it was found that the keys of the harmonium are of varying widths, requiring the actuators to be placed in a manner specific to the chosen instrument: if Kritaanjli were to be implemented with a different harmonium, different actuator placement would be required. As such, an easily-reconfigurable chassis was needed.

A final design goal for Kritaanjli was that the harmonium not be permanently augmented: all mechatronic elements must be detachable from the instrument. The reasons for the lack of permanent augmentation are twofold: firstly, it was deemed useful for the instrument to remain human-playable if the augmentation is removed; secondly, the harmonium used in Kritaanjli is an expensive, hand-built instrument with sentimental value. By avoiding permanent augmentation, it may remain in its original state when not being used as a mechatronic instrument.

The following sections detail the means by which the above challenges of making a delicate, hand-built instrument into a mechatronically-augmented system have been addressed, resulting in Kritaanjli, a mechatronically-augmented harmonium with 44-note polyphony and continuously-variable dynamic output. This chapter concludes with an overview and evaluation of Kritaanjli.

4.2 Designing and building Kritaanjli

To mechatronically augment a harmonium, a number of subassemblies have been designed to address the aforementioned design requirements. Combined, these subassemblies allow for a preexisting harmonium to be played through mechatronic means. This section details the design challenges and the means by which they are addressed, beginning with an examination of the actuator systems and concluding with an overview of Kritaanjli's electronics.

FIGURE 4.2: Kritaanjli's chassis and subasssemblies. The extruded t-slot aluminium chassis forms a cage around the harmonium, to which are attached the actuators (centre and upper left) and electronics (right).

Before undertaking a detailed overview of the subassemblies, it is useful to examine the instrument as a whole: an understanding of the instrument's dimensions and the means by which actuators may be attached allows for a keener understanding of the roles of each subsection.

The harmonium chosen to be augmented is a traditional hand-pumped harmonium measuring 31 cm deep, 26.5 cm tall, and 65 cm wide. To allow for actuators to be positioned and aligned with the instrument's keys and bellows in a non-destructive manner, a chassis fitting around the harmonium's perimeter was designed and built. The chassis, shown in Figure 4.2, is larger than the harmonium, fitting over it in a cage-like manner. As shown in Figure 4.2, the chassis consists of T-slot aluminium extrusion, chosen to allow for flexible actuator placement. The T-slot aluminium is manufactured by the 80-20 company[1], and consists of a 25 mm square profile with slots running along each face. The slots can accommodate metric fasteners: all of Kritaanjli's subassemblies are

[1] http://www.8020.net/ (retrieved March 18, 2014)

FIGURE 4.3: Kritaanjli's keyboard player: each of the 44 solenoid actuators is positioned above a key. Upon actuation, the solenoid's plunger extension depresses its key. The width of the solenoids necessitated their staggered placement above the keyboard.

designed to fit directly into the slots of the extrusion, allowing for rapid component alignment relative to the instrument while avoiding unnecessary contact between mounting brackets and the instrument.

4.3 Kritaanjli's Keyboard Player

The role of Kritaanjli's keyboard-playing subassembly is to press each of the harmonium's 44 keys upon receipt of an appropriate command from a composer-controlled host computer. To determine how best to actuate each key, preexisting musical robots were examined: based upon this review of prior works, two potential methods of actuation were chosen for closer study. The first method, used by Gil Weinberg on his Shimon robot [28], consists of a small number of actuators mounted on a linear motion trolley. The trolley is moved along the instrument, stopping and actuating at pre-defined setpoints. Such a system is compact and potentially visually compelling to audiences, but takes relatively large amounts of time to transition between keys and only allows for a small degree of polyphony. The second method, used by Trimpin and many player

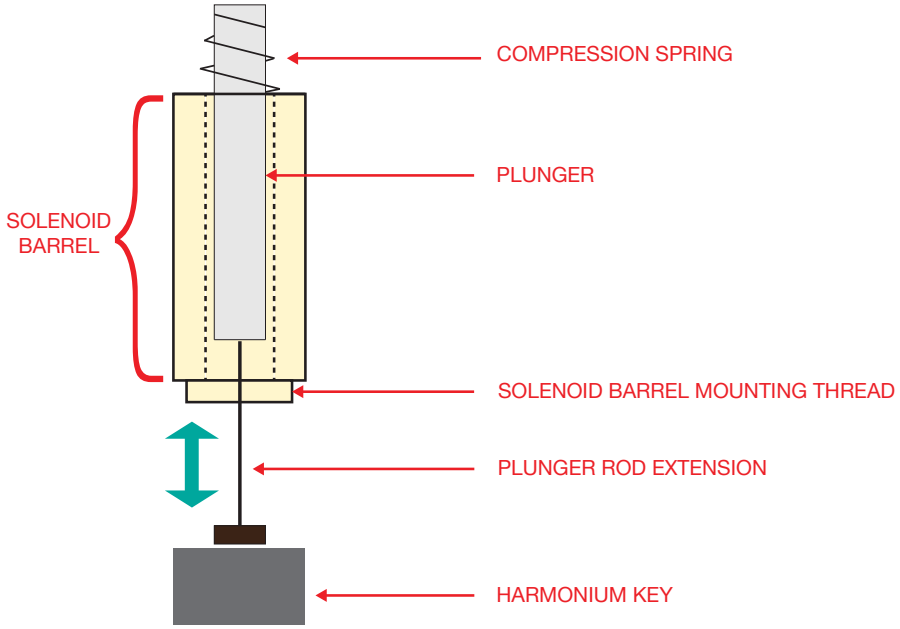


FIGURE 4.4: A drawing of one of Kritaanjli's solenoid actuators. Upon actuation, the plunger is drawn fully into the solenoid barrel and the rod extension is pressed against the harmonium key. Upon deactivation, the compression spring pulls the plunger from the body.

piano systems [19], consists of a large array of actuators, each mounted above a key on the instrument. Each actuator may be addressed individually, allowing for many notes to play concurrently. The second approach was chosen for Kritaanjli for two reasons: the affordance of high degrees of polyphony and ability to play different notes with minimal latency between them. While potentially bulkier and more expensive, this solution is deemed preferable for an instrument designed to allow for harmonically-complicated playing styles.

To implement a large array of actuators on a harmonium, a low-cost actuator with simple requisite driver electronics and significant ease-of-mounting is preferred: actuators lacking these three characteristics will prove impractical to deploy in the number required for Kritaanjli. Two actuator types were considered: miniature RC-style servomotors and linear solenoid actuators. While suitable, the miniature servomotors were found to require more mounting hardware, higher-performance controller hardware, and were of higher cost than solenoid actuators. Linear solenoid actuators were found to address the three actuator criteria, being relatively inexpensive, easy to mount, and requiring simple driver electronics.

In a manner inspired by Trimpin's Vorsetzer piano playing mechanisms [19], an aluminium bracket has been built to align each of the 44 solenoids above the harmonium's keys. While similar to those shown in [19], Kritaanjli's actuators are simpler, requiring a single solenoid and return spring. The solenoid actuators feature a threaded head which allows for secure mounting to the bracket shown in Figure 4.4. The bracket in turn connects to the t-slot aluminium chassis, and is positioned in such a manner as to allow the solenoids to depress the keys of the harmonium upon actuation. Figure 4.3 shows the solenoid bracket: as the solenoids are wider than the keys, careful solenoid positioning is needed to allow each solenoid to interact with its corresponding key. A CAD workflow was utilised to simplify this solenoid layout operation: each solenoid was modelled in AutoCad and positioned above a model of the keyboard. Solenoid mounting holes at these positions were then drilled into the aluminium mounting bracket.

The solenoids used on Kritaanjli are 25 mm diameter linear actuators: when deactivated, the solenoid's plunger rests partially above the barrel on a conical spring. Upon actuation, the plunger is pulled into the barrel. Each solenoid has been modified by attaching a key-press effector to the end of the plunger. As shown in Figure 4.4, when the plunger is pulled into the barrel, the effector pushes against a key, depressing it and allowing air to flow past the key's corresponding reeds. The driver electronics system of the solenoid array is discussed below in Section 4.3.2.

The modified solenoid actuators allow for the pull-type solenoids used on Kritaanjli to behave as push-type solenoids, pressing down on a key upon actuation. If improperly aligned with respect to its key, the solenoid may produce excessive acoustic noise (as it strikes the key) as well as added latency and damage to to the instrument. To address these issues, the key-pressing affectors are threaded to allow for precise positioning above the keys: they may be raised and lowered to allow them to rest directly on the the key when deactivated. Upon actuation, the properly adjusted affector, already resting on the key, simply pushes it down. To further reduce acoustic actuation noise and wear to the instrument, each effector is tipped with a silicone pad.

4.3.1 Kritaanjli: Pumping the Bellows

The keyboard playing subassembly discussed above in Section 4.3 is a simple system: an array of solenoids directly press the keys, with no intermediate linkages or mechanisms needed. To create a mechatronic harmonium capable of playing notes, though, the more complicated bellows must also be augmented to allow for mechanical pumping actions. Bellows augmentation is more complicated than keyboard augmentation, requiring rotational motion with a relatively long stroke. The methods for addressing these challenges and building a mechatronic harmonium pumping mechanism are discussed in this subsection.

The first stage in designing a harmonium pumping mechanism is to undertand the means by which the harmonium may be pumped. Many harmoniums are configured to allow for flexibility in pumping styles, granting the performer the ability to choose how he or she wishes to interact with the instrument. For hand-pumped Indian harmoniums, there are two main pumping methods: one in which the bellows are clamped at one edge and free to rotate at the other, resulting in a pivoting hinge-type bellows, and the other a configuration where both edges remaining unclamped, with the bellows hanging free or rotating at their base[2]. As the harmonium chosen for Kritaanjli can be pumped in either of these ways, they were compared for relative ease of mechatronic augmentation. The pivoting hinged configuration is used on Kritaanjli because it requires fewer components than the free-hanging pumping configuration. With the hinged configuration, the harmonium constrains its bellows at the hinge by itself, allowing for all mechatronic augmentation to interface with the free-hanging corner of the instrument. Conversely, the pumping method where both edges are unclamped requires added mechanical bellows support, rendering the mechanism more complicated.

After deciding that the harmonium would be pumped in its hinged configuration, a pumping mechanism capable of doing so was designed. Aside from pumping the harmonium's bellows, the mechanism needed to fulfill two criteria: first, it must be able to connect a chassis-mounted motor to the bellows in correct alignment and in a non-destructive manner; second, a simple, low-parts-count solution able to be driven with minimal electronics and actuators is deemed preferable. A four-bar linkage was settled

[2]http://www.youtube.com/watch?v=W4XHj8G2HLU illustrates the first style of pumping; http://www.youtube.com/watch?v=jJYEGLk9nSg illustrates the second (retrieved March 13, 2014)

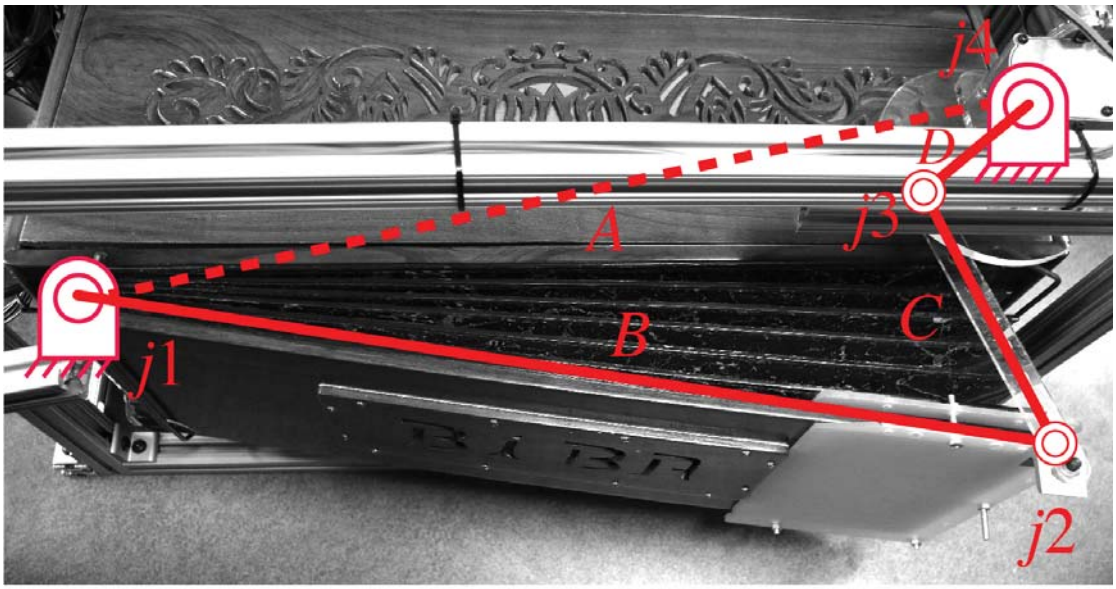


FIGURE 4.5: Kritaanjli's four-bar linkage bellows-pumping mechanism. The crank is visible on the image's right; the rocker (the hinged harmonium bellows) is on the image's left. Overlaid on the image is a drawing of the assembly's elements.

upon as the simplest means of pumping the bellows with a single motor (illustrated and labeled in Figure 4.5). In such a linkage, two of the bars are already provided by the harmonium and its chassis: the chassis forms Figure 4.5's fully constrained Bar A and the bellows forms Bar B. The hinge built into the harmonium's bellows serves as a joint between Bars A and B. Viewed as such, two additional bars are needed to complete a four bar crank rocker capable of converting a motor's full revolutions into reciprocating bellows pumping actions: these two bars (C and D) are provided by a crank mechanism attached to a rotary motor.

Once a four-bar linkage was chosen for the harmonium bellows pumping system, the next step in the bellows design process was to implement it on the instrument. The implementation consisted of two stages: determining the linkage dimensions and, subsequently, designing the physical linkage mechanisms.

As detailed above, two of the linkage's bars were provided by the harmonium's bellows and the chassis, leaving only the crank and its rod to be designed. The lengths of the crank and its rod are dictated by two factors: the desired harmonium bellows stroke length and the motor's placement on the chassis relative to the harmonium.

Deciding upon a stroke length for the harmonium's bellows proved to be a compromise between providing the harmonium's reeds with adequate airflow and preventing damage

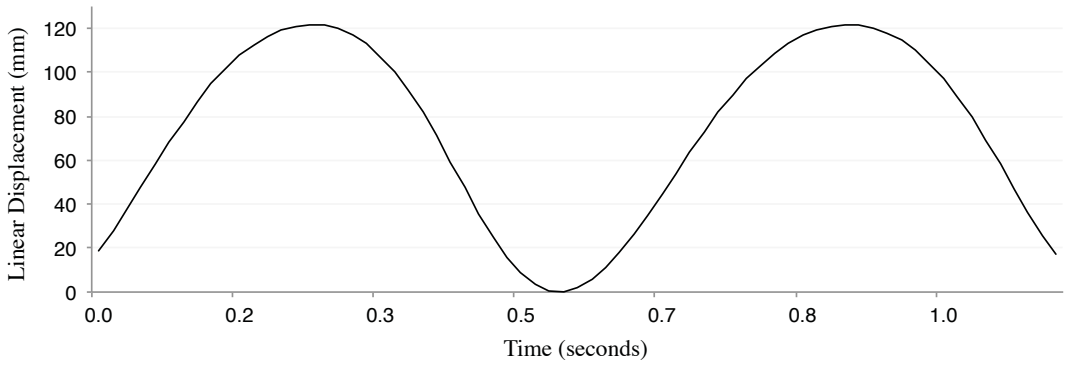


FIGURE 4.6: j_4's simulated displacement relative to Kritaanjli's motor. Two motor revolutions are shown, with a motor speed of 78 rpm.

to the instrument's hinge: if repeatedly opened to the un-hinged bellows edge's maximum displacement of 240 mm, it was decided that the relatively delicate wood, paper, and glue hinge of the harmonium may be damaged. A smaller stroke length of 120 mm was tested by hand-playing the harmonium with such a pumping stroke. Such hand-pumping with a stroke of 120 mm and a rate of one stroke per second provided what was deemed to be acceptable airflow while opening to less than its maximum angle. The crank and its rod were therefore designed to allow the bellows to open up to 120 mm in response to the rotary motion of a chassis-mounted motor.

The design of the crank and its rod were completed in SolidWorks: the harmonium and the chassis were modeled together, and models of the the crank (Bar D in Figure 4.5) and rod (Bar C in Figure 4.5) were built in context of the modeled harmonium and chassis assembly. To allow for a stroke of 120 mm, the rod connects to the crank 60 mm from its centre: a full rotation of the crank (represented by revolute joint $j4$ in Figure 4.5) results in a displacement of 120 mm and a return to its home position. The crank's rod connects the crank (at revolute joint $j3$ in Figure 4.5) to the harmonium bellows: as the crank and its motor are affixed to the chassis, the crank's rod forms the only clamped connection between the mechatronic Kritaanjli assembly and the harmonium. Because of the loose coupling between the chassis and the harmonium, the rod is made of flexible acrylic to allow for slight alignment errors. As a design goal of Kritaanjli was to avoid permanent modifications to the harmonium, the crank's rod is affixed to the harmonium's bellows pump with a removable clamp. The clamp allows the crank's rod to rotate on it (as represented by revolute joint $j2$ in Figure 4.5).

FIGURE 4.7: Kritaanjli's bellows pumping mechanism clamped to the harmonium's bellows.

The bellows pumping mechanism was designed using a CAD workflow. An advantage of the CAD design process is that it allows for parts to be tested for alignment and fit prior to fabrication. To verify that the crank could move the un-hinged edge of the harmonium's bellows through a displacement of 120 mm, a SolidWorks simulation was performed. The output of the simulation, shown in Figure 4.6, shows that revolute joint $j4$ (mounted at the corner of the harmonium's bellows as shown in Figure 4.5) is displaced 120 mm during a rotation of joint $j2$, indicating that the crank assembly will provide suitable performance. After designing and verifying the action of of the crank mechanism in context of the harmonium and chassis in SolidWorks, a DC motor compatible with the crank was chosen. Shown in Figure 4.8, the DC motor used as the Kritaanjli bellows pump must fulfill two criteria: it needs to be compatible with the crank system developed above, and it must be sufficiently powerful to pump the bellows. The chosen DC gearmotor needed to be compact enough to fit on the chassis and easily connect to the crank with a shaft-mounted hub while providing sufficient output power to pump the bellows. A Buehler 12 V DC gearmotor was chosen: the motor's body is 31 mm in diameter and 55 mm long, and drives a 120:1 ratio right-angle gearbox. Its bellows-pumping characteristics are evaluated below, in Section 4.3.3.

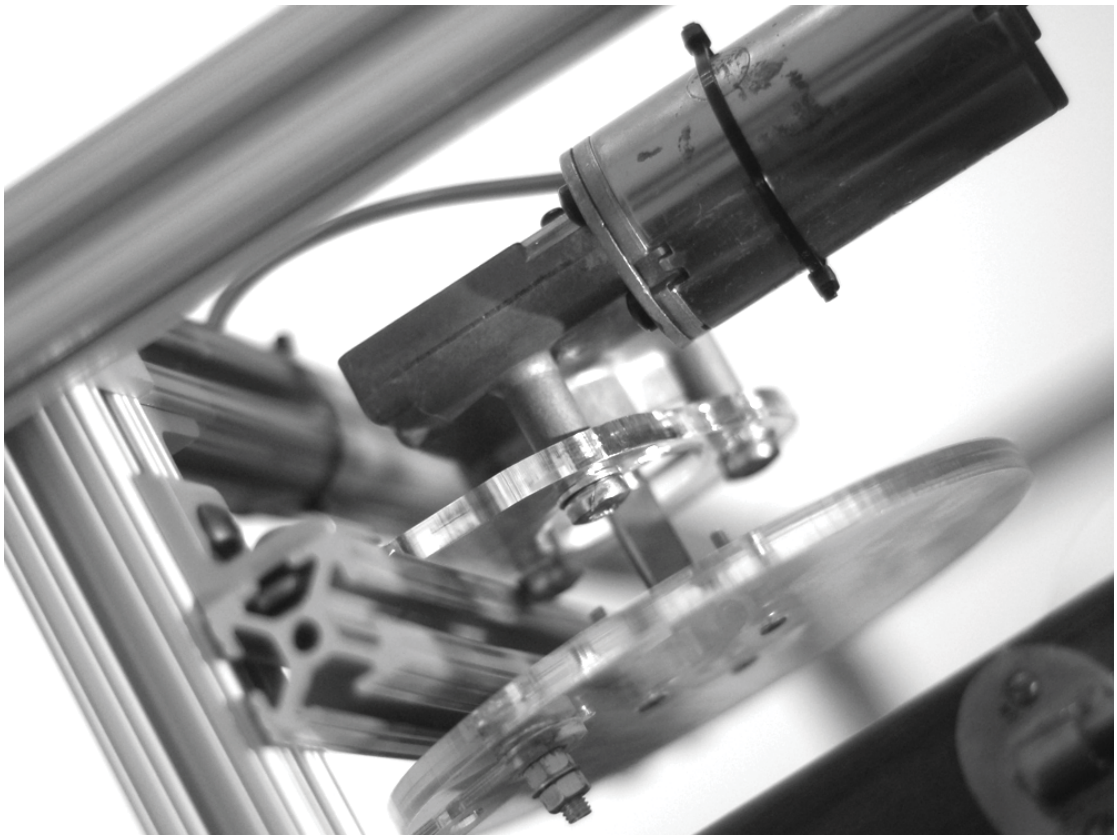


FIGURE 4.8: Kritaanjli's bellows pumping motor, connected to the crank mechanism.

The completed bellows pumping mechanism (shown in Figure 4.7) is capable of pumping the harmonium's bellows in a manner requiring no permanent augmentation to the potentially delicate instrument. Furthermore, the bellows pumping assembly requires simple driver electronics (discussed below in Section 4.3.2), and is inexpensive and of low parts-count: the harmonium bellows clamp, crank, rod, and motor mount and hub are the only required parts[3].

4.3.2 Electronics

To allow Kritaanjli's keyboard player and bellows pumping mechanism to actuate the harmonium in response to a composer's commands, a purpose-built electronics assembly is employed. This assembly, shown in Figure 4.9, contains three sections: a communications section, a microcontroller, and an actuator driver electronics section. Operating in concert, these electronics allow Kritaanjli to receive messages, determine to which actuator the message pertains, and activate or deactivate the actuator.

[3]See https://vimeo.com/51977345 for a video of Kritaanjli's bellows pumping system and keyboard player (URL retrieved March 13, 2014).

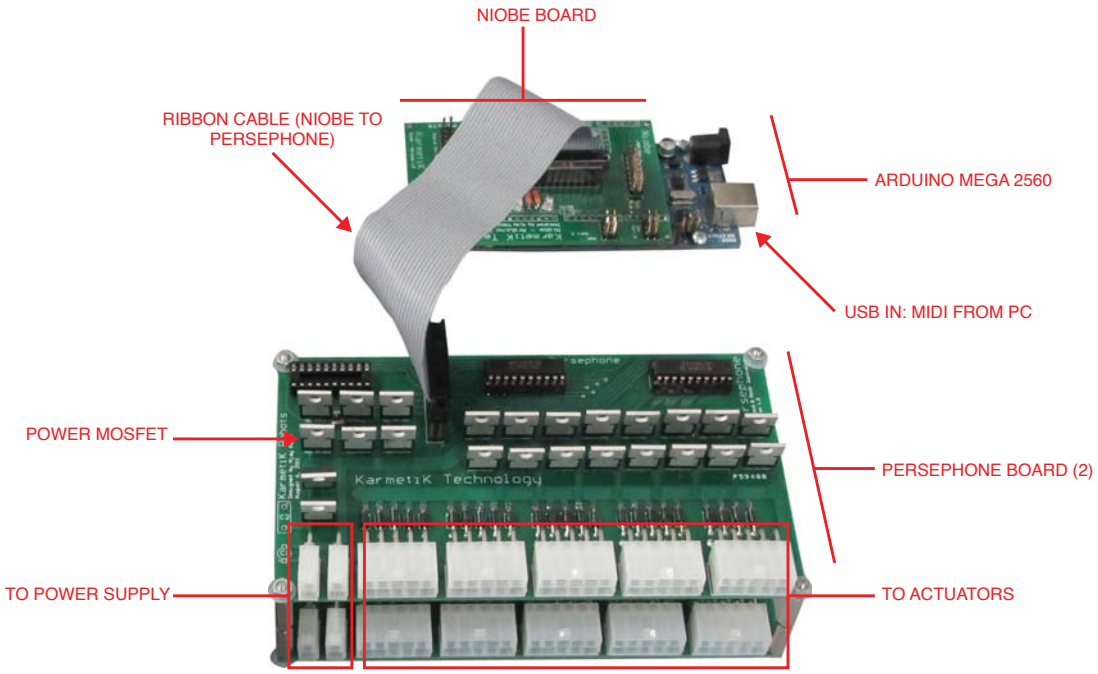


FIGURE 4.9: Kritaanjli's electronics: an Arduino is connected to two 24-output Persephone boards by a Niobe board.

A key part of Kritaanjli's electronics is the provision to allow communication between it and a composer-controlled host device. The MIDI protocol is used, both due to its widespread popularity as a communications protocol for musical instruments and to remain compatible with the other MIDI-equipped systems described in this document and in the author's (and author's collaborators') prior works [6]. A detailed explanation of MIDI as used on the systems described in this document can be found in Appendix B.

As a simple, low-parts-count MIDI implementation is preferred, the HIDUINO firmware is used on Kritaanjli. HIDUINO, developed by Dimitri Diakopoulos [54], allows for Kritaanjli to interface with MIDI host devices as a driverless USB MIDI Human Interface Device (HID). This MIDI HID functionality enables Kritaanjli to interface with computers without the need for dedicated MIDI hardware interfaces: a user need only plug the Kritaanjli electronics into a computer's USB port, at which point Kritaanjli may be controlled using any MIDI host software.

The decision to use HIDUINO dictated the choice of microcontroller employed in Kritaanjli. As HIDUINO requires a USB-equipped ATMEGA microcontroller such as the ATMEGA16U2, the ATMEGA16U2-equipped Arduino Mega 2560 microcontroller development board is used to handle Kritaanjli's communications and actuator control.

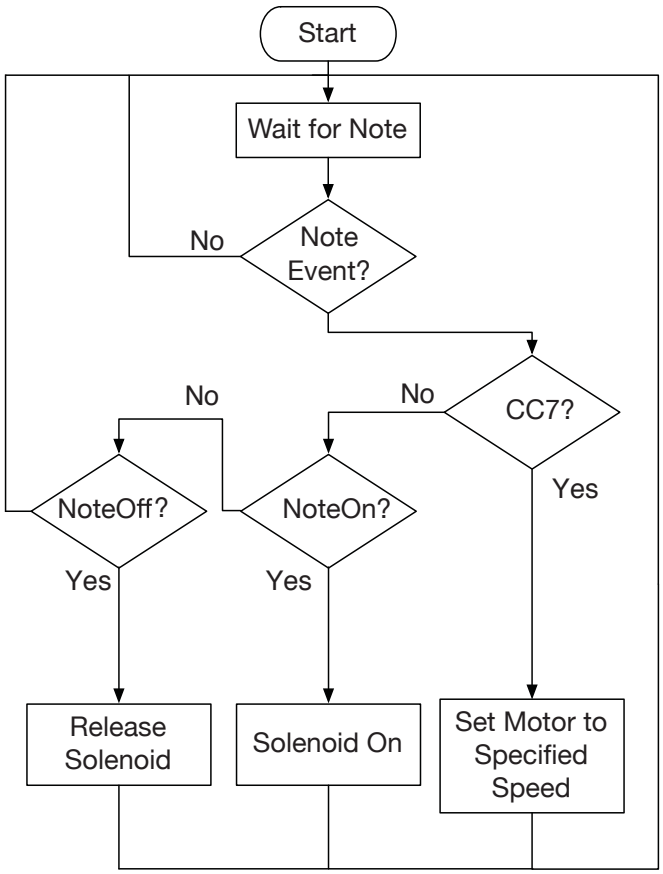


FIGURE 4.10: The program flow of Kritaanjli's firmware.

The Arduino Mega is suitable for Kritaanjli for a second reason: its ability to provide a large number of digital outputs without the need for additional electronics. The Arduino Mega features two microcontrollers: the USB-equipped ATMEGA16U2 and an ATMEGA2560 whose role is to handle analog and digital input and output. Equipped with HIDUINO, the ATMEGA16U2 converts MIDI HID commands to serial commands parsable by the ATMEGA2560. The ATMEGA2560 has 54 digital output pins, 44 of which are used to interface with Kritaanjli's solenoid driver electronics. A pulse width modulation output on the ATMEGA2560 is used to interface with the bellows pumping motor's driver electronics.

The role of Kritaanjli's communications-handling firmware is to receive incoming MIDI commands and affect the system's actuators accordingly. To allow for individual control over each of Kritaanjli's actuators, each actuator is assigned a specific command. 44 MIDI NoteOn commands (with each command's pitch parameter corresponding to its respective harmonium key) are used to activate the solenoids; equivalent NoteOff

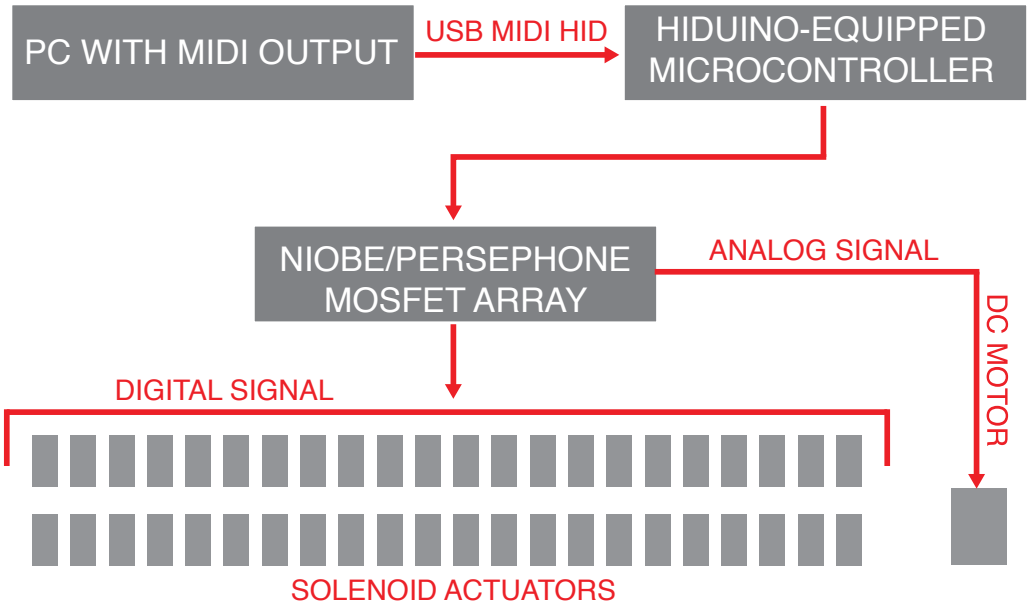


FIGURE 4.11: A block diagram of the Kritaanjli's electronics and actuators.

commands deactivate them. The data parameter of a MIDI Control Change 7 (CC7) command dictates the bellows pumping motor speed. The Arduino MIDI Library is used to handle the incoming MIDI commands: the library was chosen for its easy-to-use interface and use of callbacks, simplifying the act of handling incoming MIDI commands. Figure 4.10 details the ATMEGA2560's program flow: upon receipt of a MIDI command, the command type is determined and its appropriate callback is executed.

The MIDI interface and microcontroller serve as two of the three elements in Kritaanjli's electronics subsystem; the actuator driver electronics form the final element. Together, these three allow a host to send commands to the instrument, resulting in actuation events (as diagrammed in Figure 4.11). As with the mechanical design of the instrument, simplicity was a goal throughout the design process of Kritaanjli: a simple array of power MOSFETs are used to control both the keyboard playing solenoids and the bellows pumping motor. The large number of MOSFETs needed for Kritaanjli dictated the design of a new printed circuit assembly. This assembly, dubbed the Persephone board (shown in Figure 4.9), allows the low-power signals from a microcontroller to switch the high-power actuators. To allow 48 of the Arduino Mega 2560's digital outputs to switch a higher-powered load, the Persephone board is equipped with 24 FDB7030BL N-Channel MOSFETs. Two Persephone boards are used for Kritaanjli; on the two boards, 45 of their outputs are employed (44 to actuate the solenoids, and an additional one for the .

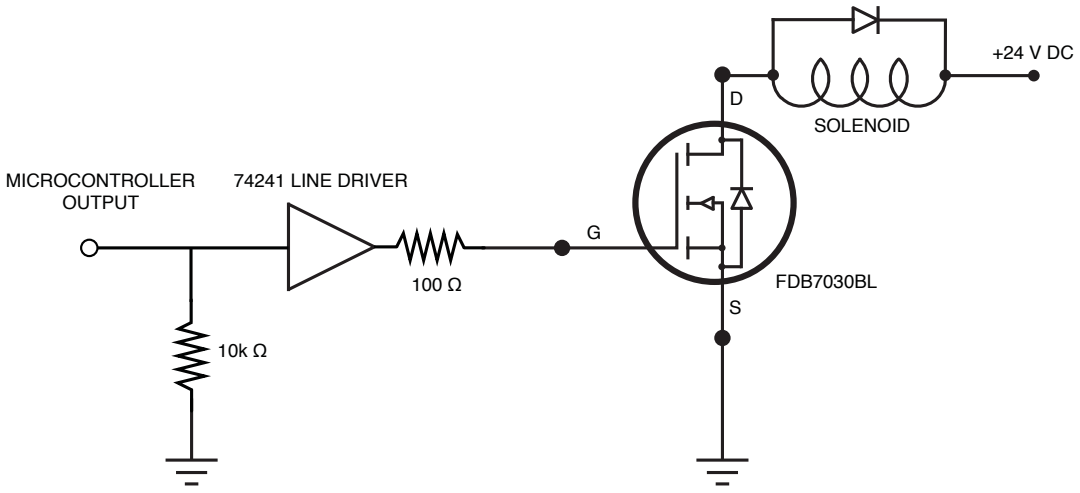


FIGURE 4.12: Kritaanjli's solenoid circuit. A line driver is used to allow for longer cable runs between the microcontroller and the Persephone board. The same circuit is used for Kritaanjli's bellows pumping motor.

The FDB7030BL MOSFET was chosen for three reasons: its drop-in compatibility with Darlington Pair TIP-122 drivers (used on previous prototypes of the Persephone board), its relatively low cost, and institutional familiarity with the device. Figure 4.12 shows an example circuit, of which there are 24 on each Persephone board: an output pin on the ATMEGA2560 is sent through a line driver (to allow for the use of longer ribbon cable); it switches the FDB7030BL power MOSFET, activating a solenoid connected to the MOSFET's drain.

The final element in the actuator driver electronics assembly (shown in Figure 4.9) is a means by which the Arduino microcontroller board can be connected to the Persephone power MOSFET boards. A design objective in the building of the Persephone board was to allow it to be compatible with future microcontroller platforms which may have different interfaces than the Arduino. To allow for such future changes, The Niobe, an intermediate adapter board, is employed to connect the Arduino to the Persephone boards. Configured as an Arduino "shield," the Niobe daughterboard connects directly to the Arduino's output pins, routing their signals to ribbon cables which connect to the Persephone driver board. The Niobe can be replaced with other small adapter PCB assemblies in the event of future microcontroller changes, allowing for the larger and more expensive Persephone boards to be used without modification.

To power Kritaanjli's electronics and actuators, a switched-mode AC to DC power supply is used. Both the bellows pumping motor and the keyboard-playing solenoids require 12

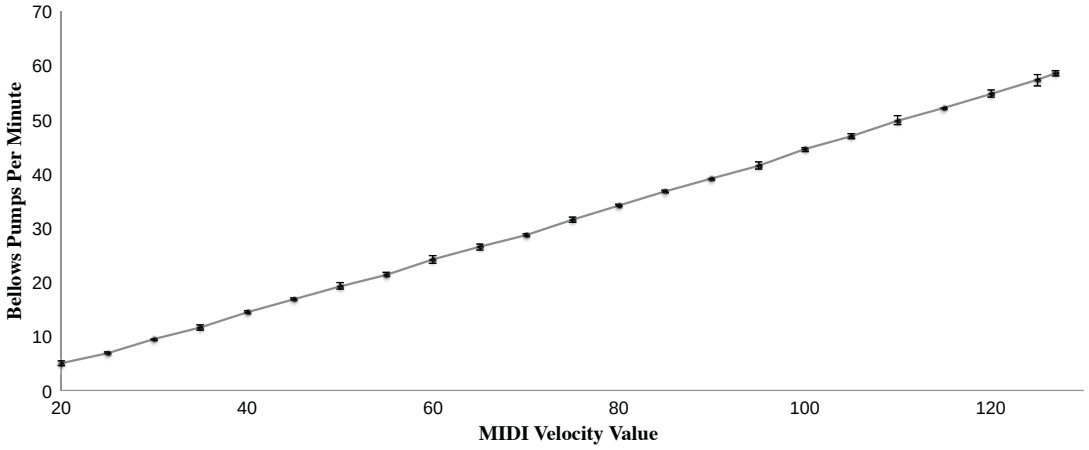


FIGURE 4.13: Harmonium pumps per minute with varying MIDI velocity values sent to the DC bellows motor driver.

V DC. To determine the power supply's required amperage, the actuators' current draw at 12 V DC was measured. The keyboard-playing solenoids each draw up to 0.5 A at 12 V DC; the bellows pumping motor draws up to 0.3 A at 12 V DC. The electronics, which require 5 V DC and draw up to 100 mA, use an Arduino-mounted 7805 linear voltage regulator to obtain their necessary voltage. To allow for 44 note polyphony while pumping the motor and powering the electronics, a 269 W power supply is needed. A 500 W 12 V DC power supply is used, chosen due to its capability to provide sufficient power for Kritaanjli and allowing for the potential future use of additional actuators on the three unused MOSFETs on Persephone.

4.3.3 Kritaanjli: Evaluation

This section evaluates the performance of Kritaanjli, examining the behaviour of its actuator subsystems. An understanding of the performance characteristics of Kritaanjli allows composers to better exploit the device's capabilities. Three performance parameters deemed significant to the device's behaviour are evaluated and discussed below.

It is important for composers working with Kritaanjli's variable-speed bellows pumping motor to understand how the motor responds to varying voltages, specified by varying MIDI command values. Such an understanding allows them to send meaningful MIDI messages to the instrument. To determine the bellows pumping speed of Kritaanjli at different motor voltages, 23 MIDI values are sent from a host device to Kritaanjli. These commands begin at MIDI value 20, a minimal speed for the bellows pump, and

TABLE 4.1: Times to initial sounding and full volume for each octave of Kritaanjli.

Note	Initial Sounding	Full Volume
C2	0.8 s	5.1 s
C3	1.1 s	6.2 s
C4	1.4 s	7.1 s
C5	2.4 s	7.5 s

increase in steps of five to a MIDI value of 125; a final test to evaluate the system at full intensity is performed at a MIDI value of 127. The times required for single pump cycles are recorded, and three trials are averaged. The results of the bellows pumping speed evaluation is shown in Figure 4.13, with the error bars representing the standard deviation over three trials. As shown, the input MIDI value bears a linear relationship to the number of pump events per minute. From a musical perspective, these results mean that a composer could use an input MIDI value of 20 to obtain very slow, quiet pump events. At a MIDI value of 127, nearly one 120 mm stroke pump event per second can be achieved[4]. At high speeds, relatively large volumes of airflow through the harmonium are attainable, allowing many reeds to be excited at once: with high MIDI CC 7 values, extended polyphony can be achieved.

Upon completing construction of Kritaanjli, initial tests showed that a small amount of time is required between the beginning of an actuation event and the beginning of the note's sounding. Additionally, higher notes were found to take longer to begin sounding (and to reach their full volume) than lower notes. These observations make sense in light of the instrument's behaviour: in a harmonium, sound is produced as air is moved across vibrating reeds: some amount of time is required to induce reed vibration. As the size of the reeds decreases, increased airflow is required to excite the reed: higher notes are observed to require longer durations of airflow across the reed to produce sound. To characterize this latency, one note per octave is actuated. The bellows are then instructed to pump with a MIDI command of 127 (resulting in a high speed pumping action, as shown in Figure 4.13). A microphone placed 30 cm from the rear centre of the harmonium records the instrument's output. The recorded results are analyzed in the Ableton Live PC digital audio workstation, and are reported in Table 4.1: lower notes are found to require less time to begin sounding and to reach full volume than

[4] A method of normalising these onset delays is presented in Chapter 7.

higher notes. An awareness of this latency will allow composers to account for it while composing for Kritaanjli.

While the previous two tests focus on the behaviour of Kritaanjli's bellows pumping mechanism, an adequate evaluation of Kritaanjli's performance necessitates also an understanding of the keyboard player's solenoid performance. The rate at which the solenoid could consistently repeatedly press a key is evaluated, allowing performers to further understand the instrument's behaviour. To measure the harmonium's solenoid key press repetition rate, the solenoids are instructed to play increasingly rapidly. The maximum rate at which the solenoids can consistently cycle a key is recorded. Across the instrument's 44 solenoids, the maximum consistent repetition rate is 610 key press events per minute. This high speed is likely due to the spring-return nature of the harmonium's keyboard: when a solenoid's electromagnetic field is deactivated, the key springs back to its home position, returning the plunger to a position in the centre of the solenoid's barrel, where it responds quickly to subsequent actuation commands. Because of the aforementioned time required to excite the instrument's reeds, there will likely be few playing situations requiring these high solenoid repetition rates. Such high speed actuation events could prove compositionally useful in extended technique contexts, however.

The results of the tests presented in this section show that Kritaanjli meets its initial design goals: it is capable of pumping a harmonium's bellows with a range of intensities and actuating its keys at high speeds in a manner that does not permanently augment the instrument. With the mechatronic elements completed and evaluated in this chapter, the means by which an artist can interface with this new mechatronic instrument are presented and discussed in Chapter 7.

4.4 Summary

Kritaanjli provides ensembles of mechatronic instruments with a relatively portable keyboard-playing robot capable of polyphony and user-defined volume envelopes. Originally intended to give percussion-heavy ensembles of robots (such as the KarmetiK Machine Orchestra) a robot capable of harmonic and melodic output, Kritaanjli is also capable of "extended technique" composition styles.

An additional motivation that dictated many design choices undertaken on Kritaanjli was the decision to augment the instrument in a non-destructive manner. As such, Kritaanjli exists as an example of the non-destructive mechatronic augmentation of a relatively-complicated instrument: the bellows pumping mechanism and keyboard-playing assembly are mounted in such a manner as to allow easy removal of the mechatronic system.

By providing composers with a thoroughly-characterised mechatronic harmonium capable of novel amplitude envelope manipulation, high speed note repetition capability, and non-permanent augmentation of the instrument to which it is attached, Kritaanjli allows composers to explore expressive mechatronic composition techniuqes for keyboard-based instruments.

Chapter 5

Nudge: A Parametrically-Rich Mechatronic Drum Player

5.1 Introduction

As with the chordophones presented in Chapter 3 and the keyboard instrument in Chapter 4, robotic percussion systems can gain musical expressivity with the assistance of increased degrees of freedom. To explore this assertion, Nudge, a new mechatronic drum player was designed, built, and evaluated.

As with the other new instruments described in this document, Nudge (shown in Figure 5.1) was built after a detailed review of existing systems: by examining prior robotic drum players, Nudge has been designed and built in a manner that incorporates their strengths while addressing and improving upon their relative lack of expressivity.

This chapter begins with an overview of design objectives. After examining a number of prior mechatronic drumming systems, their expressive shortcomings are discussed and Nudge's resultant design specifications are introduced.

Section 5.3 details the means by which Nudge's design goals are accomplished, focusing upon each of its subsystems. The chapter concludes with an evaluation of Nudge's subsystems' performance.

FIGURE 5.1: Nudge: at left is the DC servomotor connected to the turntable. The turntable pivots the solenoid-mounted drumstick (right) and its accompanying stick height stop.

5.2 Motivations and Design Goals

Mechatronic drum players use electronically-controlled actuators to bring an effector into contact with a percussive object. Compared with the numerous steps required to perform a chordophone or keyboard instrument playing event, mechatronic percussion instruments can be built in a very simple manner, utilising a single actuator mounted in a fixed position above a percussive object [55].

These simple solenoid-based drum beaters (exemplified by those shown in Figures 5.2 and 5.3) have been nearly ubiquitous in mechatronic drumming systems since the birth of mechatronic music in the 1970's. In the subsequent decades, similar systems have been featured in many ensembles, installations, and academic articles. Indeed, with only minor variations, solenoid percussion systems similar to the early works of Trimpin and the Logos Foundation continue to appear in contemporary academic publications [20] [56].

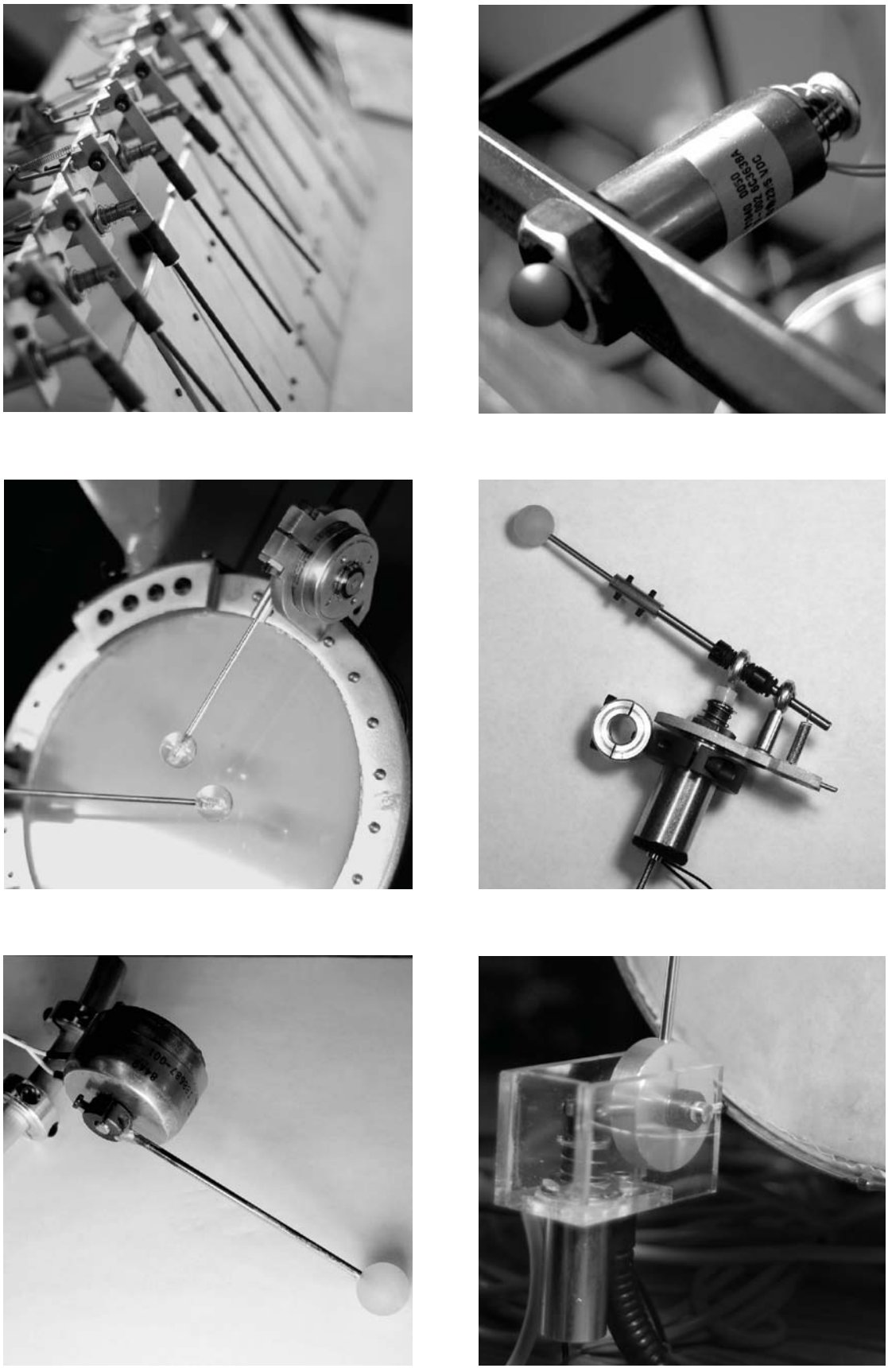


FIGURE 5.2: Machine Orchestra drum beaters. Clockwise from top right: Kapur Finger (on Tammy), KalTron (as used on NotomotoN), DarTron (on MahaDeviBot Rev. 1), TrimpTron (as used on NotomotoN), TrimpTron (with aluminium bracket on MahaDeviBot Rev. 2), and CalTron array (on GlockenBot).

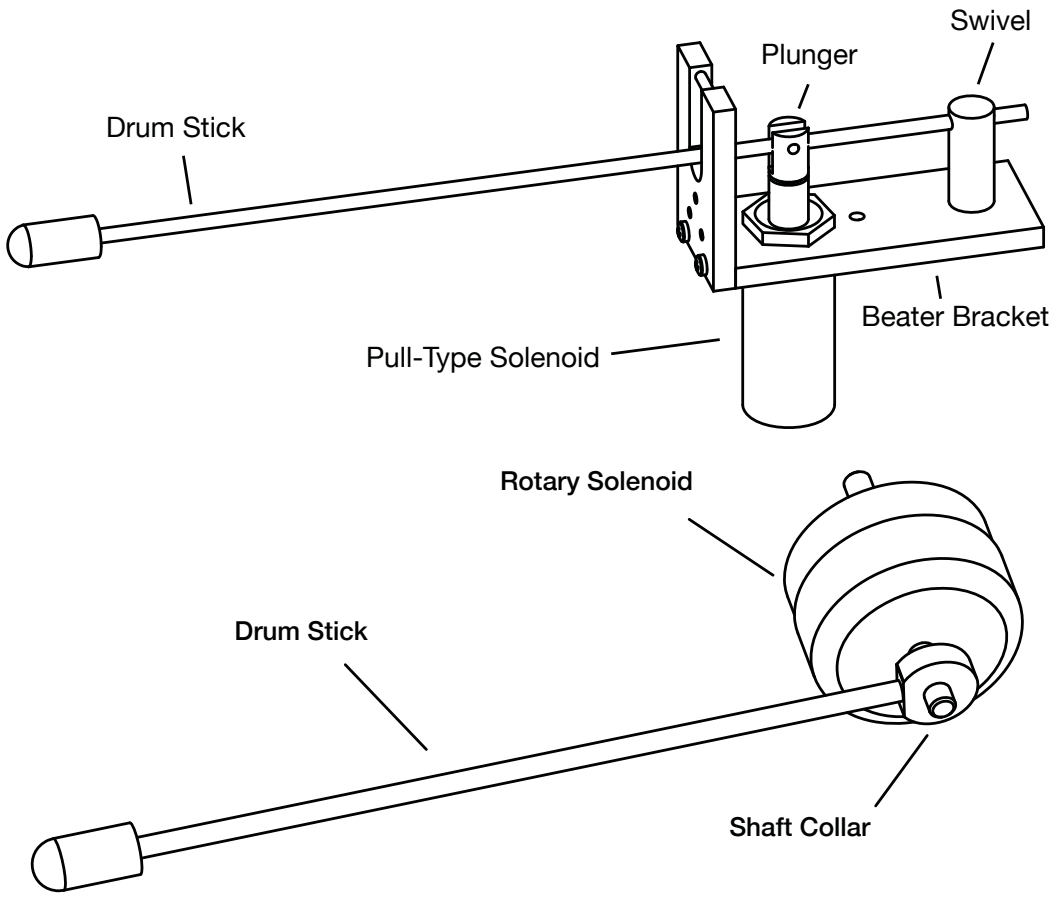


FIGURE 5.3: An illustration of two solenoid beater designs examined for use on Nudge. A system utilising a linear-motion solenoid is illustrated at top. Below is a rotary motion solenoid, similar to the TrimpTron-derived beater used on Nudge.

Such basic automatic drum beaters are not particularly elegant in their simplicity: they lack both the ability to precisely and repeatably play a wide dynamic range (the ability of a mechatronic drummer to play at different loudness levels), temporal range (the ability of a mechatronic drummer to play at varying speeds)[57], and the ability to change their effector's point of contact with the percussive object. By building a system with increased dynamic range, temporal range, and effecter position, composers can be presented with a system capable of enhanced expressivity.

The challenges present at Nudge's inception were to build a drum player capable of the aforementioned expressive parameters while retaining as much as possible the preexisting systems' two largest advantages: high-speed performance and simplicity. While not capable of wide ranges of expressivity, existent solenoid drumming systems perform well when positioned correctly: as shown in [58] and described in [1], even with minimal

electronics and configuration, solenoids can forcefully or, if suitably positioned, rapidly strike a drum head.

After examining the strengths and shortcomings of prior systems (such as those presented in Chapter 2 and in Section 5.3.1), a number of design specifications were settled upon: Nudge should be capable of repositioning its drumstick in relation to a percussive object. Additionally, it should be able to vary both its temporal and dynamic output. Finally, it should be as mechatronically simple as possible, as large numbers of such beaters will potentially be built to replace the existing beaters on mechatronic percussion ensembles. The next section focuses on the ways that these specifications are implemented on Nudge, resulting in a relatively simple expressive mechatronic drumming system.

5.3 Nudge: Systems Overview

Similar to the new instruments described in Chapters 3 and 4, Nudge is an assembly of subsystems, each of which serves to provide a specific musically-expressive trait by addressing a design goal (specified in the previous section). A drawing of these subsystems is shown in Figure 5.4; each are discussed in the following subsections.

5.3.1 Striking a Drum: Nudge's Drum Beater Mechanism

The means by which an effector is brought into contact with a percussive surface is at the core of any mechatronic percussion system, be it a simple or mechatronically-complicated drum player. On Nudge, the drum striking mechanism forms the core of the mechatron: all expressivity-enhancing subsystems are built relative to it, as they all serve to modify the way that the effector strikes the drum. As the other subsystems collectively interface with the drum striker, an examination of Nudge's drum striking system is a sensible place to begin an examination of Nudge as a whole.

The author's (and author's collaborators') relatively extensive performance and installation experience with solenoid-based mechatronic drum ensembles (described in [6], [55], and [57]) resulted in the decision to pursue a solenoid-based drum striker on Nudge[1].

[1] A small number of mechatronic drum systems use miniature servomotors (such as https://www.youtube.com/watch?v=IfVCmPT6l68, retrieved April 14, 2014), which produce very audible actuation sound and are incapable of the speeds attainable by a solenoid actuator. Only by using expensive industrial actuators can such shortcomings be mitigated.

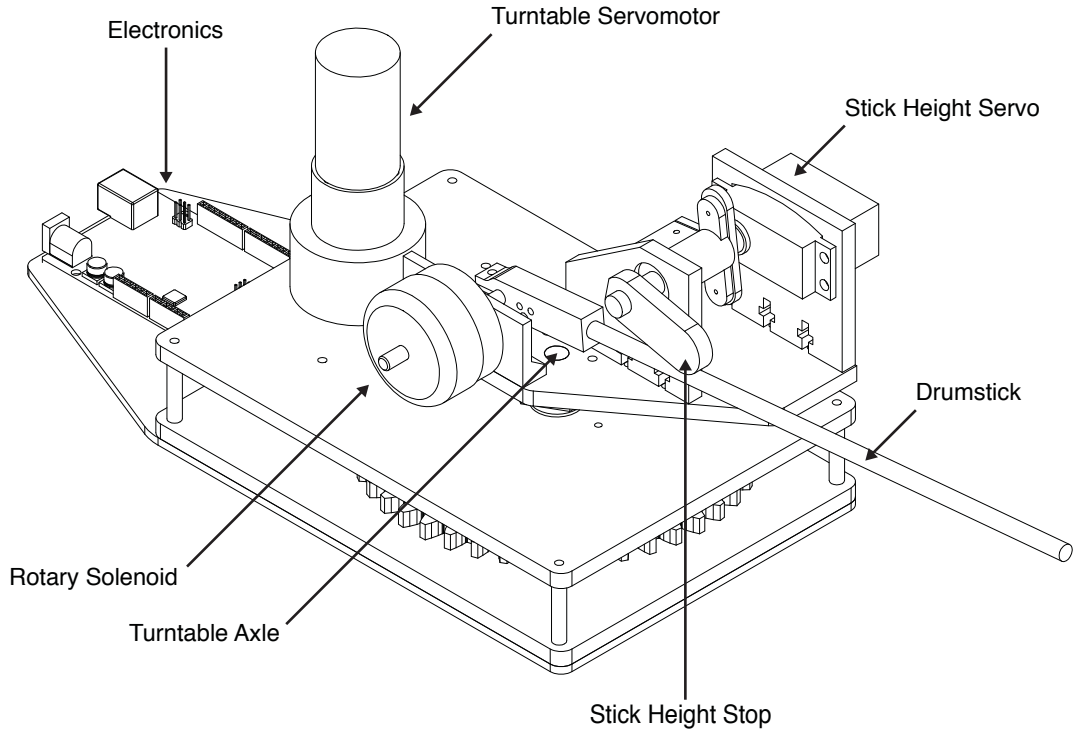


FIGURE 5.4: A drawing of Nudge, with major subsections labeled.

Further, the use of solenoids fulfills the design objectives of simplicity and compatibility with existing systems, requiring only simple driver electronics, power systems, and mounting hardware.

Upon deciding to use a solenoid-based striking actuator, the mechatronic drum beaters which the author and collaborators have used were compared, with the intention of selecting a suitable drum beater variety for augmentation on Nudge. These drum beaters, all of which are used in the KarmetiK Machine Orchestra and subsequent projects [6], fall into two categories: linear motion beaters and rotary motion beaters (both types of which are shown in Figure 5.2 and described in Table 5.1).

Linear motion solenoid beaters are built around a linear solenoid actuator. This actuator can be configured to either directly affect the percussive object or affect the percussive object through a mechanical linkage. The "Kapur Finger" beater [59] is an example of such direct actuation: upon receipt of a DC voltage (typically between 0 and 24 V DC), the solenoid's plunger is pulled toward the solenoid's barrel. Such solenoids are modified in a manner similar to Kritaanjli's key actuators (presented in Chapter 4 and illustrated in Figure 4.4): a hole is drilled through the base of a pull-type solenoid and a rod is

TABLE 5.1: Mechatronic drum beaters used in KarmetiK Machine Orchesta robots. The parts count indicates a minimal configuration: typical configurations might include more complicated drum beater attachments and other additions. All listed solenoids include both solenoid barrel and plunger. See Figure 5.2 and [59] for images of these beaters.

Beater Name	**Solenoid Type**	**Beater Description**	**Mach. Orch. Robots w/ This Beater**	**Number of Components (Minimal Config.)**
BellHop	Linear (Push)	A pull-type solenoid modified to expel its plunger against an object.	MahaDeviBot	4: Solenoid, plunger extension tube, mounting tube, damping material
Boom Beater	Linear (Push)	A large drum beater which pushes a rotating mallet against a drum.	BreakBot	5: Solenoid, spring, drumstick, bracket, rod end
CalTron	Linear (Pull)	A low-parts-count beater with attached drumstick or mallet. Intended to be relatively low cost.	GlockenBot, BreakBot	5: Solenoid, bracket, springs (2), drumstick, drumstick adapter for solenoid
DarTron	Linear (Pull)	A crank-equipped beater which hits a drum upon its return stroke.	GanaPatiBot Rev. 1	9: Solenoid, crank components (5), spring, bracket, drumstick
Kapur Finger	Linear (Push)	A push-type linear solenoid whose plunger directly strikes an object.	MahaDeviBot, Tammy, Kritaanjli	3: Solenoid, spring, plunger pad, bracket piece
KalTron	Linear (Pull)	Similar to the Trimpin Hammer: home height is adjusted by manually adjusting a plunger-mounted stop.	NotomotoN	12: Solenoid, springs (2), bracket pieces (3), rod ends (2), drumstick, retaining clamps (3)
Singer Hammer	Linear (Pull)	A mallet-equipped beater: the mallet is loosely attached to the solenoid with a flexible joint.	MahaDeviBot	7: Solenoid, spring, plastic joint, ball bearings (2), drumstick, bracket
Trimpin Hammer	Linear (Pull)	A dual-rod end beater: one attaches to the frame, the other to the solenoid plunger.	MahaDeviBot	7: Solenoid, rod ends (2), bracket pieces (3), spring, retaining clamps (2)
TrimpTron	Rotary	A rotary solenoid with an integrated spring return; a drumstick or mallet is affixed to the solenoid's shaft.	MalletOTon, NotmotoN, MahaDeviBot Rev. 2	4: Solenoid, bracket mount, shaft clamp, drumstick

threaded to the end of the plunger. The rod is pushed down as the plunger retracts into the beater; attached to the rod is a striker intended to directly actuate a drum or other percussion instrument. The Tammy robot's marimba [6] as well as various attachments on MahaDeviBot [24] feature such actuators.

Linear solenoid-based drum beaters such as the Kapur Finger and BellHop[2] consist of a solenoid plunger which directly interacts with the drum or other percussive object. While simple, inexpensive, and of low parts count, these direct-interaction drum beaters lack the visually-compelling motion of a drumstick-equipped beater as well as such a beater's increased leverage for forceful strike events. Such direct-interaction beaters are not used on Nudge because, in spite of fulfilling the objective of a simple design, their mechanical simplicity dictates that they lack the ability to engage in the wide dynamic range desired in a new expressive drum beater. A linkage-equipped system is required to provide sufficient drumstick displacement from the drumhead (discussed in more detail in Section 5.4.2).

The rejection of such purely linear-motion direct-interaction solenoid mechanisms led to the examination of a variety of rotary-motion drum beater mechanisms for possible augmentation and use on Nudge. A rotary motion solenoid-based drum beater mechanism consists of an an actuator configured to rotate an effector through an arc that intersects with a percussive object. The KarmetiK Machine Orchestra uses numerous such beaters, listed in Table 5.1. Such rotary motion beaters fall into two subcategories: those using a linear-motion solenoid and those using a rotary-motion solenoid. Upon evaluating drumbeater types for augmentation and inclusion in Nudge, systems using linear-motion solenoids were examined and rejected due to their inherent need for the relatively high-parts-count linkages required to convert their linear motion to a rotary motion. Systems using a rotary motion solenoid, such as the TrimpTron beater (shown in Figure 5.2), consist of a rotary solenoid to whose shaft an effector is directly attached. As detailed in Table 5.1, such beater types are of low parts-count and are simple to manufacture and affix to a chassis. By fulfilling the aforementioned criteria for a drum beater, the TrimpTron was selected for implementation on Nudge.

Most mechatronic drumming systems are largely drum-agnostic: the same drum beater mechanism may be employed on a variety of different percussive objects. By using a

[2]The BellHop was developed and it extensively used by Trimpin, particularly in *ConlonInPurple* (2000) [19]

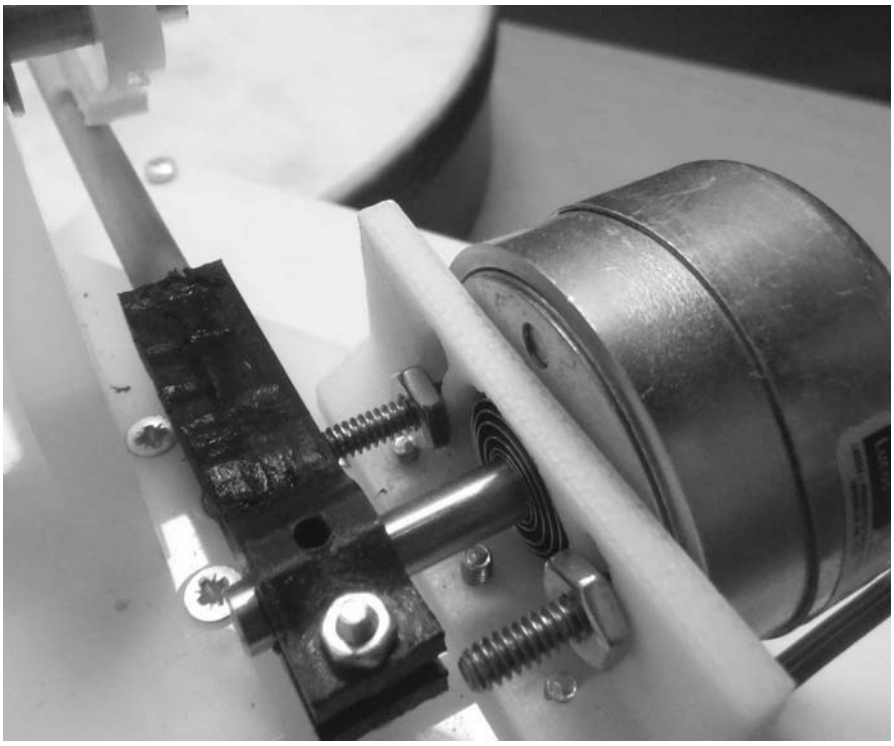


FIGURE 5.5: Nudge's drum beater mechanism: at left is the solenoid At right is the drumstick, affixed to the solenoid with an ABS plastic clamp.

solenoid beater design already proven compatible with a wide variety of percussive materials, Nudge inherits the TrimpTron's demonstrated agnosticism, and can be deployed on traditional drums (as on the KarmetiK NotomotoN [55]) or, equally well, on unusual surfaces (as on Trimpin's computer-striking Laptop Percussion [19]).

The following paragraphs detail the solenoid drum beater as deployed on Nudge, illustrated in Figures 5.4 and 5.5.

The TrimpTron, like many solenoid actuators, is a relatively large and heavy assembly. To adapt it for Nudge, on which the solenoid drum striking system is turntable-mounted (discussed below in Section 5.4), weight-saving measures were undertaken. While the TrimpTron actuators used on the NotomotoN mechatronic drum system are similar to those employed on Nudge, the relatively large 50 mm diameter actuators on NotomotoN were rejected in favour of the smaller and lighter 40 mm diameter Lucas Ledex 369781 rotary solenoid: the smaller actuator was deemed more suitable to mounting on the repositionable Nudge chassis. Aside from the smaller, lighter solenoid actuator, the drum beater mechanism on Nudge differs from prior TrimpTron beaters in both mounting scheme and drumstick attachment. Where previous TrimpTron beaters use steel or aluminium mounting and drumstick-attachment hardware [55], lightweight ABS plastic

fixtures are employed on Nudge. These may be rapidly 3D printed, allowing for different diameter drumsticks to be used with little reconfiguration.

The ABS plastic drumstick attachment, shown in Figure 5.5, was designed in SolidWorks and 3D printed on an Up Mini 3D printer. Designed to allow a 7 mm diameter drumstick to be attached perpendicular to the solenoid's 5.5 mm rotary shaft, the drumstick attachment is 50 mm long and 10 mm wide. The drumstick is held in place on the attachment with the aid of M3 grub screws. While early iterations of the drumstick attachment attached to the solenoid's centre shaft with grub screws, a large amount of slippage was observed; the solenoid shaft grub screws were later replaced with a screw-tightened clamp, greatly reducing the attachment's slippage on the solenoid's shaft.

After mounting the drumstick to the drum beater, both needed to be affixed to the body of Nudge. To accomplish this, an ABS plastic mounting bracket was developed, designed to allow for precise angling of the solenoid. The drum striker's solenoid actuator is equipped with two mounting screws which affix the solenoid to the bracket with the aid of nuts. Rather than simply using mounting holes through which the solenoid's mounting screws are placed, the clamp is fitted with two circular slots, allowing the solenoid to be manually rotated through 30 degrees prior to the tightening of its nuts; such angling (inspired by its application on the NotomotoN) is of use when percussive objects are placed significantly higher or lower than the solenoid actuator.

The drumstick used on Nudge is a 7 mm diameter hardwood dowel. The small diameter was chosen for its lightness: a lighter drumstick will respond more rapidly to the solenoid's actuation and return-spring events. The 7 mm dowel can accommodate a number of pads or dampers at its terminal end: as tested, a round plastic attachment was used. For other percussion applications, mallet-type heads, brushes, or other head types may be easily affixed.

As the drum striker described in this subsection is quite similar to the TrimpTron-style drum beaters used on the NotomotoN and other systems [55], it suffers from the same expressive limitation of those drum beaters: a narrow dynamic range (described in more detail in the author's prior publications [57, 60]) and a fixed timbral range. By augmenting the beater, though, these limitations can be overcome. The following subsections detail the means by which this rotary solenoid mechanism is adapted to

FIGURE 5.6: Nudge with multiple drums: by rotating its motorised turntable, Nudge can move to strike multiple objects within its workspace.

allow for the playback of a significantly wider range of event types than is possible with a traditional TrimpTron beater.

5.3.2 Repositioning the Drum Beater

Mechatronic drumming systems are typically incapable of repositioning their end effector and are only able to strike a single point on a percussive object. As discussed in Section 5.3.1, such systems fail to provide composers with the expressive parameters needed to access the range of timbres afforded by one or more percussive objects. Granting users access to the ability to reposition the end effector above the drum head was a central objective in the intial design stage of Nudge. The means by which other systems provide more than a single timbre to composers were examined, and a resulting system was designed based upon the examination's findings. This subsection begins by discussing the means by which other systems endeavour to widen their timbral range and concludes with a detailed overview of Nudge's own drumstick positioning subassembly, shown in Figure 5.6.

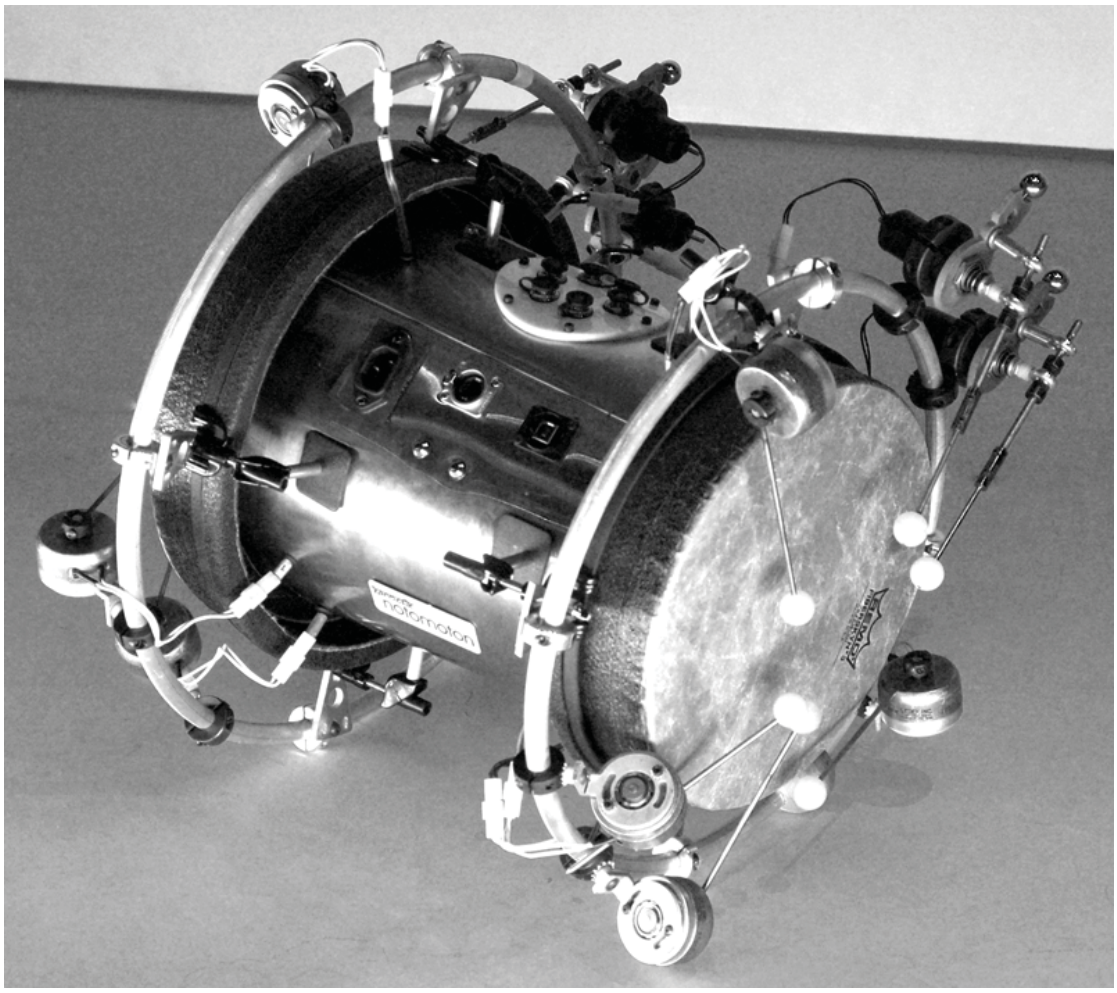


FIGURE 5.7: The KarmetiK NotomotoN, an example of a mechatronic drum player that achieves extended timbral range by utilising an array of pre-positioned solenoid actuators.

While the majority of mechatronic drum players consist of a single drum actuator mounted statically above a drum head, there are a small number of percussion systems that present composers with more timbral options. Two methods are used for this: large arrays of discrete actuators and single actuators on positioners. Both techniques are discussed below.

The KarmetiK NotomotoN (developed by the author's collaborators [55]) serves as a good example of a solenoid-array drum system. Shown in Figure 5.7, the instrument's

drumheads can be struck at a number of positions. To strike at varying positions, different solenoids are addressed[3]. Similar systems have been developed by Expressive Machines[4], Felix's Machines[5], Trimpin [19], Godfried-Willem Raes [20], and others. Mechatronic drum playing systems consisting of arrays of discrete actuators share the advantages offered by other "discrete" mechatronic music techniques (such as solenoid-array chordophone fretting systems and keyboard playing mechanisms, discussed in Chapters 3 and 4): they provide composers with electronically-simple access to a range of timbres while allowing for rapid transitions between actuators and polyphonic output. Their drawbacks are similar to those presented with the solenoid array chordophone fretting assemblies: they are potentially bulky, expensive, and of relatively low resolution. The low resolution of such discrete arrays was of particular concern in the design of Nudge: after developing compositions for NotomotoN, the lack of fine-grained timbral adjustments was found to be frustrating, suggesting that a higher-resolution solenoid positioning system could grant more expressivity.

For greater timbral range, roboticists Gil Weinberg and Scott Driscoll created their robot Haile by affixing a solenoid drum beater to a motorised linear motion stage [27]. This, an early example of a repositionable drum beater, allows a composer to precisely pick where on a drum head the solenoid should strike. While the linear motion system employed by Weinberg and Driscoll is larger and more complicated than is desired for Nudge, their system was a major inspiration in the development of Nudge. Though there are no detailed published evaluations of Haile, video documentation[6] indicates that through the use of its linear positioner, a relatively wide range of drum sounds can be played.

In choosing a means by which a mechatronic drumming system's timbral range could be enhanced, repositionable beaters were focused upon in favour of discrete arrays of solenoid. While the NotomotoN, an instrument with a relatively large array of separate solenoids, can play at 12 discrete locations on a drum head [55], repositionable drum beaters can play at as many unique locations as their motors can discretely reach. Additionally, as illustrated in Figure 5.6, a single repositionable drum beater can be configured to play multiple percussive objects. Due to this enhanced timbral range and flexibility, a repositionable drum beater mechanism was chosen (in favour of discrete

[3] On NotomotoN, each solenoid is addressed with a different MIDI NoteOn command [55]
[4] http://expressivemachines.com/ (retrieved April 15, 2014)
[5] http://felixsmachines.com/ (retrieved April 15, 2014
[6] https://www.youtube.com/watch?v=veQS6tsogAA (retrieved April 15, 2014)

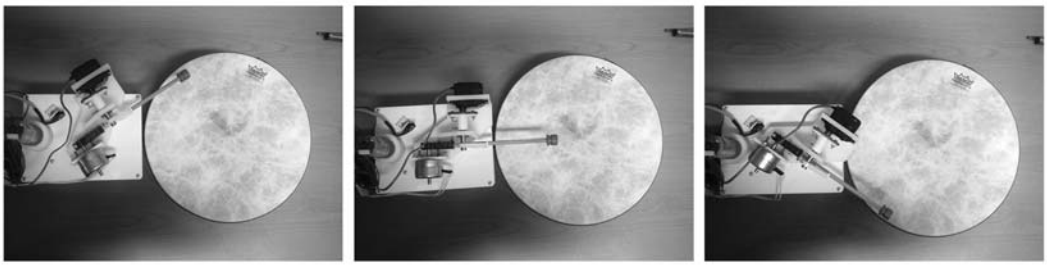


FIGURE 5.8: Nudge's rotary drumstick positioner, shown rotating above a drum; as shown, the drumstick can strike either rim of the drum or an arc along the drum's head.

arrays of solenoids) for Nudge as the method to achieve greater timbral expressivity. Upon deciding to pursue such a system, a number of goals were established in the design of the expressive drum positioner for Nudge. While adhering to the overarching goal of a relatively simple, low parts-count solution, the positioner must provide significantly improved resolution over typical existent systems, as well as the ability to reposition its attached effector rapidly and in a repeatable manner. Finally, a closed-loop system was specified to allow for composers to "teach" Nudge to recall specified positions in its workspace, a process described in Chapter 7.

To fulfill the drum positioning subsystem's design goals, two types of motors were examined: stepper motors and DC servomotors. Stepper motors, though suitable for precise positioning systems, were rejected based upon experiences obtained in the design of MechBass's stepper motor linear positioning system: they were found to be acoustically noisier than alternatives and in need of relatively complicated driver electronics and large power supplies. Further, additional position encoders are necessary to render the stepper motor a closed-loop device. Conversely, DC servomotors can utilise widely-available motor drivers and, thanks to their integrated angular position encoders, allow for simple interfacing with a microcontroller for closed-loop control (described in Section 5.3.4). Additionally, gear-reduced servomotors are readily obtainable at low prices, allowing for high-resolution and high-torque positioning solutions to be implemented.

Upon choosing to use a DC servomotor for Nudge's drumstick positioner, a means by which the motor would rotate the drum was decided upon. As shown in Figures 5.6 and 5.8, the mechanically-complicated linear positioner (as used and shown in [27]) was rejected in favour of a lower parts-count rotary motion positioner. If mounted suitably, the workspace of such a rotary motion positioner can reach both the edges and centre

of one or more percussive objects. As direct-drive DC servomotors of enough torque to position the drum beater at a number of discrete positions above a percussive object were found to be prohibitively expensive and bulky, smaller and more economical geared DC servomotors were examined for suitability. A small 12 V gear-reduced servomotor was chosen, equipped with a 1:51 gear reduction on the output shaft: the servomotor, a CIC30GM gearmotor, features a direct drive shaft-mounted Hall Effect rotary encoder capable of providing 26 pulses per revolution of the motor: after gear reduction, the output shaft generates 1326 pulses per revolution, or one pulse per 0.27 degrees of rotation. Over a drum head encompassing 60 degrees of its travel, positions corresponding to 221 encoder counts are reachable. Such resolution is significantly higher than the relatively small number of positions reachable by discrete solenoid arrays such as those shown in Figure 5.7.

To fabricate a rotary motion drumstick positioner, two systems needed to be built: a DC servomotor mount and a turntable to which the drumbeater assembly is mounted. The DC servomotor mount is a 3D-printed ABS plastic bracket that couples the motor's faceplate to Nudge's stationary chassis. A 3D-printed motor mount was chosen over other materials and fabrication techniques due to the durability of ABS components and the ease with which CAD-designed components may be fabricated with the aid of additive manufacturing tools. Like the DC servomotor, the turntable is coupled with Nudge's 19 cm by 14 cm stationary chassis (as shown in Figure 5.4). The turntable consists of an acrylic plate to which the aforementioned solenoid drum beater assembly and its accompanying solenoid height stop (presented in the following subsection) are mounted. Acrylic, like the 3D-printed ABS plastic, is used due to its ease of manufacture: parts can be designed in CAD software and quickly laser cut from acrylic sheet, allowing for rapid design iterations with little turnaround time.

Upon system startup, the turntable must be returned to a home position to allow the encoder to count from a previously established position rather than one arbitrarily chosen upon initialisation. The turntable's home position corresponds to the location of a static home switch. The normally-open SPST microswitch is mounted on a bracket and closed upon contact with the turntable. The homing procedure is described below in Section 5.3.4.

By foregoing the use of large arrays of solenoids in favour of a single solenoid drum

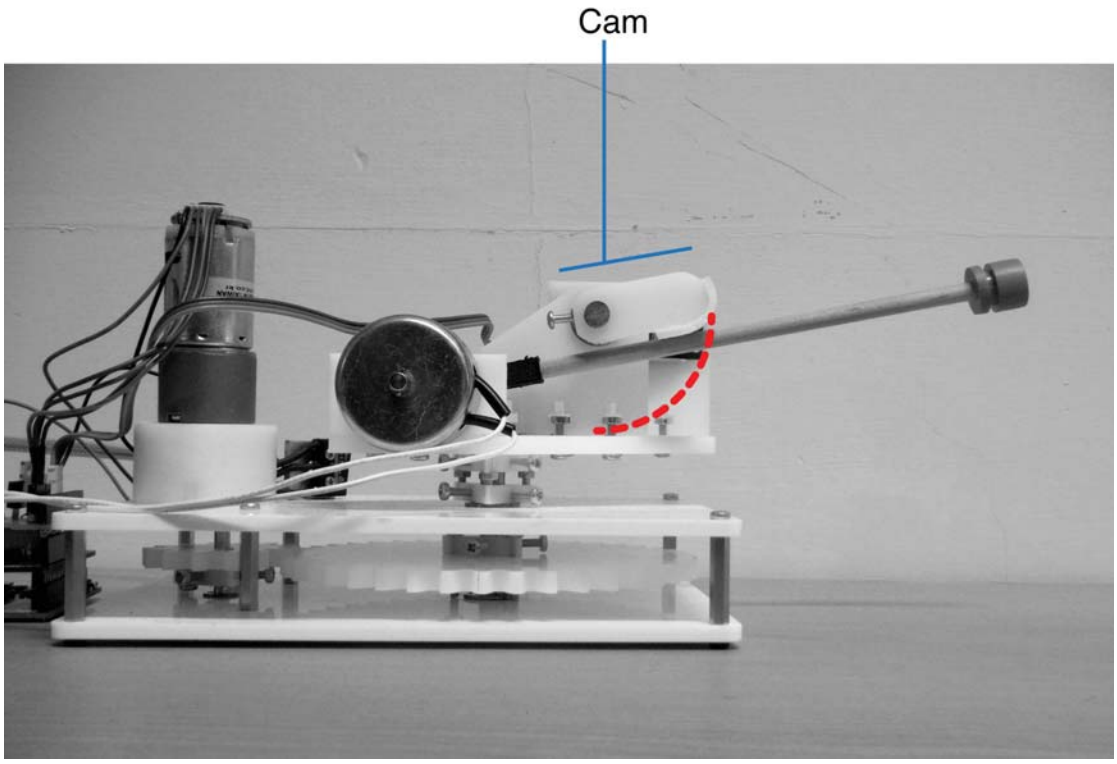


FIGURE 5.9: Nudge's drumstick height stop cam. The red arc illustrates the angles through which the cam can travel, altering the at-rest height of the drumstick.

beater mounted in such a manner as to enable online repositioning above one or more percussive objects, one of the original goals in creating Nudge is accomplished: that of creating a drumming system able to achieve a greater timbral range than is possible with the nearly-ubiquitous static drum beater systems.

5.3.3 Increasing Nudge's Dynamic Range

To accomplish the remaining goals of enhancing a drum beater's dynamic and temporal range, an additional degree of augmentation must be made to the TrimpTron-style rotary solenoid mechanism described in Subsection 5.3.1. In keeping with the accompanying design goal of simplicity, a single integrated solution for enhancing both dynamic and temporal range is preferred.

The author's previous experience with statically-mounted drum striking mechanisms indicates that they can play rapidly and quietly if their drumsticks' at-rest height is

close to the percussive object[7]. Slower, more forceful playing can be achieved if the drumstick's at-rest height is further from the drum head. While a single beater type can be mounted offline in a variety of positions to play a relatively wide range of dynamics and tempos from any single at-rest height, the author's prior published findings show that a fixed-position solenoid beater lacks extensive dynamic and temporal range [60].

By adjusting the drum beater's drumstick's at-rest height in an online manner, the desired wide dynamic and temporal range can be achieved. As the at-rest height changes from high above the drumhead to a position in close proximity to it, solenoid actuation events change from relatively powerful events (repeatable at a slow tempo) to soft drum strike events (repeatable at a high tempo). The implementation challenge on Nudge was to create such a system able to alter the drumstick's at-rest height on the fly while remaining mechanically simple and able to respond rapidly to incoming messages. The development and design of such a system, shown in Figures 5.9 and 5.10, is described in the following paragraphs.

To develop the adjustable solenoid height stop, two ways to incrementally alter a drumstick's at-rest height were examined. Lead screws, whose displacement can be precisely adjusted, were compared with cams (potentially less precise but faster to respond to height-change events). In both cases, the mechanisms are instructed to alter their displacement in response to an actuator's motion, changing the point at which the drumstick rests upon deactivation. As the at-rest height needed to be changed in the middle of musical phrases, the rapid response of a cam-based height stop was deemed preferable when compared to precise (but typically slower) lead screw systems. In addition to their rapid response, cams fulfill the initial goal of simplicity: a cam system requires only a motor and a shaft-mounted cam, while other positioners require additional hardware.

To allow for consistent performance, an actuator capable of repeatably rotating an attached cam to a specified angle is needed. While any feedback-equipped servomotor is suitable for this task, miniature RC-style servomotors (as used on Swivel 2 and MechBass, described in Chapter 3) are selected due to their ease of mounting, minimal necessary driver electronics, low price, and compact size. The solenoid height stop servo is a standard-sized RC servomotor to whose output shaft is affixed a shaft clamp. Attached

[7]Upon deactivation, a rotary solenoid actuator's shaft is returned to a home position by a spring return, a process illustrated in https://www.youtube.com/watch?v=hxWmnkSBX9o (retrieved April 15, 2014).

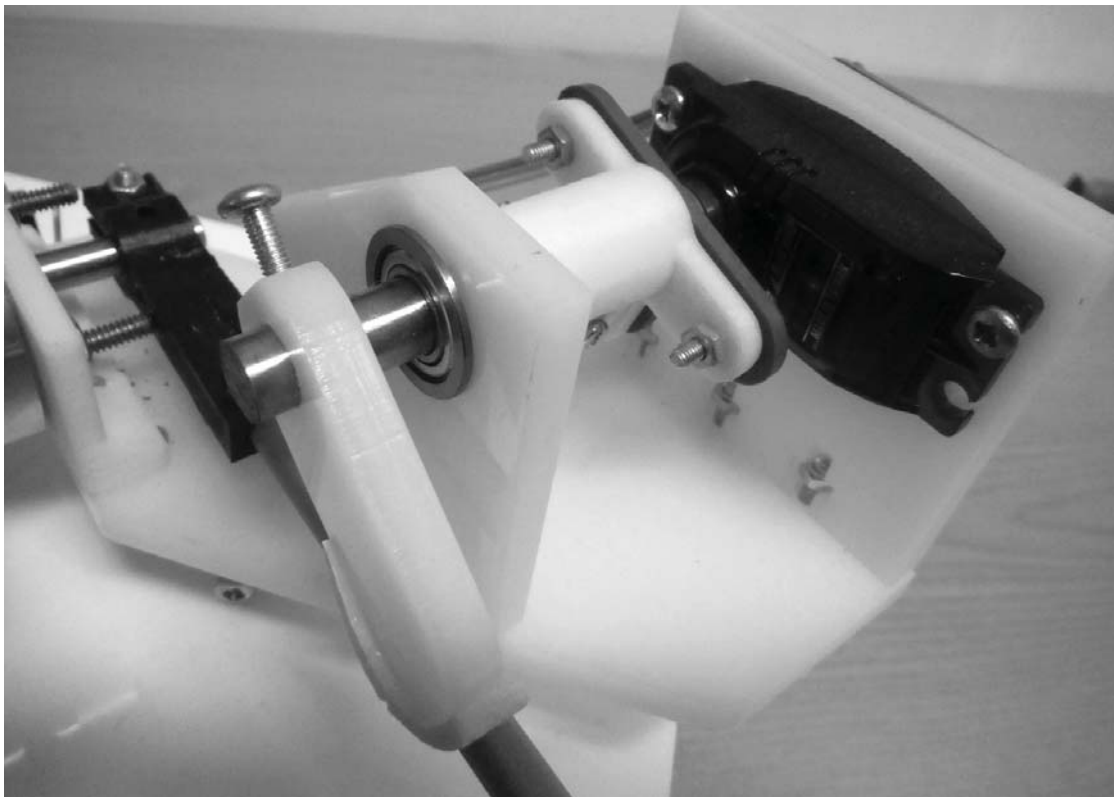


FIGURE 5.10: A detail of Nudge's cam-based solenoid height stop. The miniature servomotor, shown at right, rotates the shaft-mounted cam, changing the drumstick's at-rest height.

to the shaft clamp is a shaft extension, allowing the cam to be positioned parallel above the drumstick. The cam itself, like much of the rest of Nudge, is composed of acrylic. Early trials showed that upon deactivation and returning to its at-rest position under the power of the solenoid's return spring, the drumstick struck the cam with clearly audible force. A small foam pad, visible in Figure 5.10, is affixed to the cam to reduce this acoustic noise.

While the cam may be pivoted through the 180 degrees of rotation of which its servomotor is capable, its rotation on Nudge is restricted to 45 degrees of meaningful travel (angles outside of this range either push the drumstick into the drum or chassis or rotate out of contact with the drumstick). Within these 45 degrees, the cam can force the drumstick to rest either directly on the drum head or at its default at-rest height, as well as 43 positions in between. Section 5.3.4 describes the means by which the servo's angle (and the drumstick's resultant at-rest height) is altered through composer-defined MIDI commands.

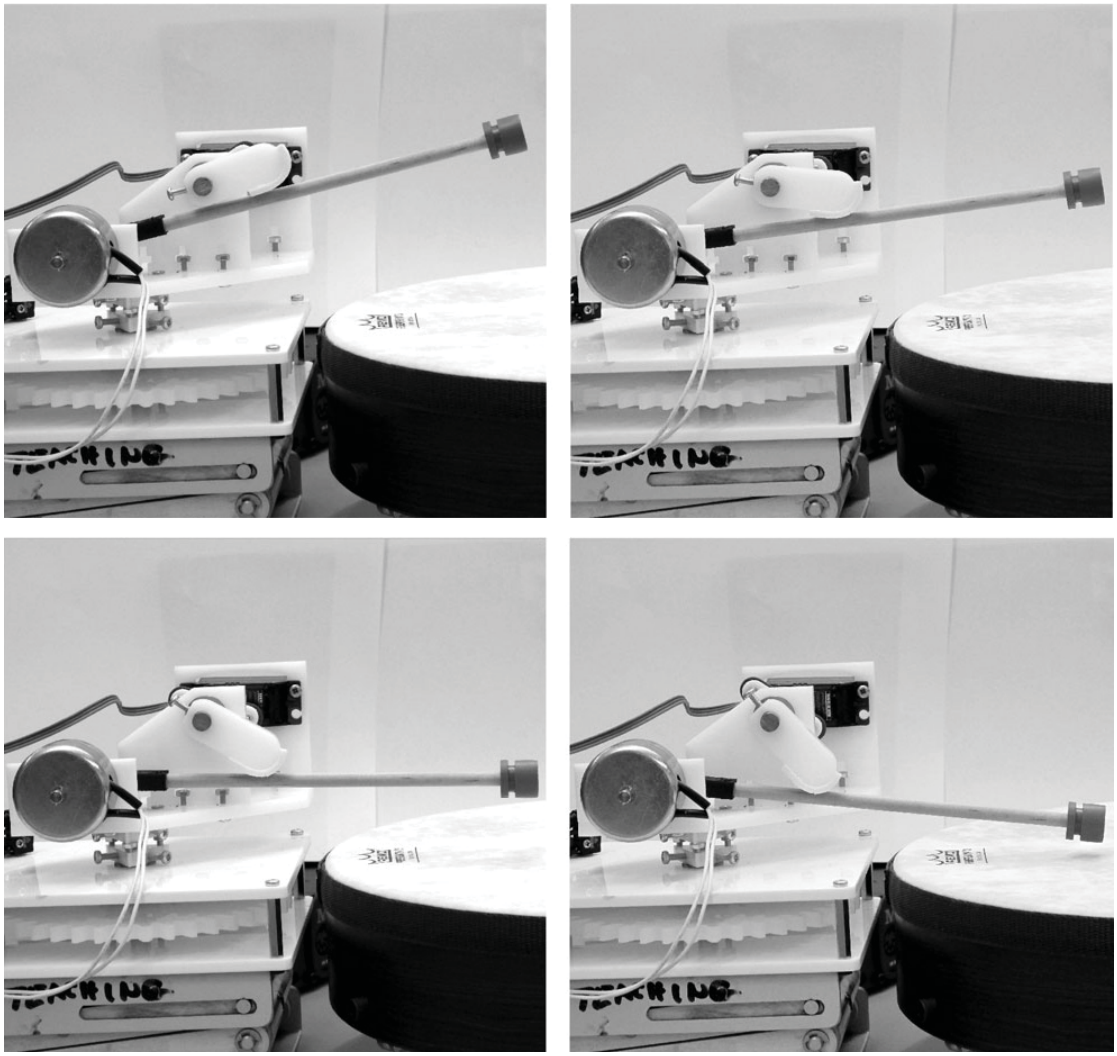


FIGURE 5.11: A sequence of Nudge's height stop servo positions. As the servo rotates clockwise, the drumstick is brought to an at-rest position closer to the drum head.

5.3.4 Nudge: Electronics and Firmware

The mechanical systems described in the preceding sections allow for a mechatronic drum player to play a wide variety of timbres as it rotates above the percussive object and a wide range of dynamics and tempos as its drumstick's proximity to the percussive object is varied. To affect those changes, Nudge employs an electronic system whose role is to receive incoming composer-issued commands, process the commands, and output suitable signals to drive the three actuators. This subsection details these electronics, describing their implementation and the means by which they fulfill Nudge's stated design goals of relative simplicity and compatibility with the author's (and author's collaborators') other mechatronic musical instruments.

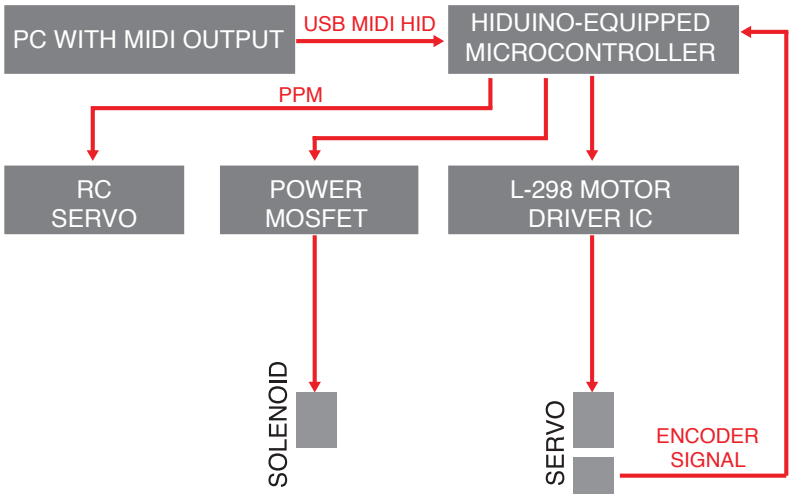


FIGURE 5.12: A block diagram of Nudge's electronics.

Nudge's electronics system (a block diagram of which is shown in Figure 5.12) can be viewed as three subsystems. These subsystems together allow Nudge's actuators to respond to a composer's commands: a communications subsystem receives commands, a processor handles the commands, parsing them and converting them into actuator control signals (the processor further handles the computations required in the rotary turntable's control scheme), and an actuator control electronic subsystem receives the processor's signals and outputs voltages to Nudge's actuators.

Nudge's communications scheme is similar to those used on the other instruments described in this thesis. As with the other instruments, a communications scheme compatible with existing PC music software and the author's own preexisting systems is preferred. With such requirements in place, MIDI was selected as the scheme of choice. While Open Sound Control [61] was considered, its additional requirements for hardware-level implementation were deemed unsuitable in favour of MIDI's simplicity. Additionally, the resolution offered by MIDI is deemed suitable for the actuator output resolution present on Nudge.

After the successful deployment of HIDUINO on Kritaanjli (described in Chapter 4), HIDUINO has been chosen as the MIDI communications scheme for Nudge, a system not in need of the daisy-chain buses used on MechBass and Swivel 2. HIDUINO, described in more detail in [54], is a driverless MIDI HID interface, allowing USB hardware-equipped ATMEGA microcontrollers to be recognized by USB host devices as MIDI HID devices.

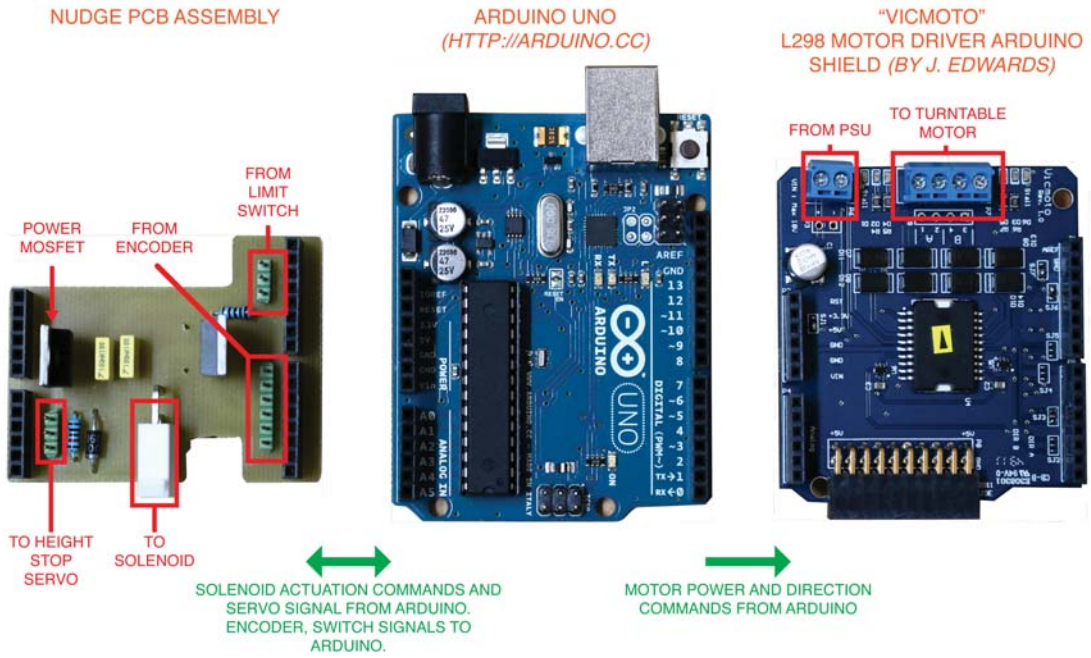


FIGURE 5.13: Nudge's electronics: the purpose-built Nudge PCB assembly and the VicMoto L298 motor driver shield both attach to Nudge's Arduino Uno.

As MIDI HID devices, they may be interfaced with by PC software in a driverless manner, removing the need for additional MIDI hardware interfaces. To use HIDUINO, an Arduino microcontroller board with a USB-capable ATMEGA micontroller is required. The more expensive and bulky Arduino Mega 2560 required to control Kritaanjli's numerous actuators is not needed for Nudge. Instead, a smaller and less expensive Arduino Uno Revision 3 microcontroller board is used (shown in Figure 5.13 and equipped with the USB-compatible ATMEGA16U2 microcontroller and an ATMEGA328 microcontroller).

As the Arduino Uno is equipped with a communications microcontroller as well as a general purpose ATMEGA328 input and output microcontroller, the otherwise-unused ATMEGA328 was examined for its suitability as the electronics' processor subsystem.

The ATMEGA328 can be programmed with the Arduino language, a C-derived language equipped with many useful libraries and helper functions, as well as the ability to control the low-level functionality of the microcontroller by writing C or assembly language code at any point in a program. Because of this ease of use, flexibility, and large number of community-developed actuator control libraries, the decision was made to use the Arduino Uno's ATMEGA328 as Nudge's main processor.

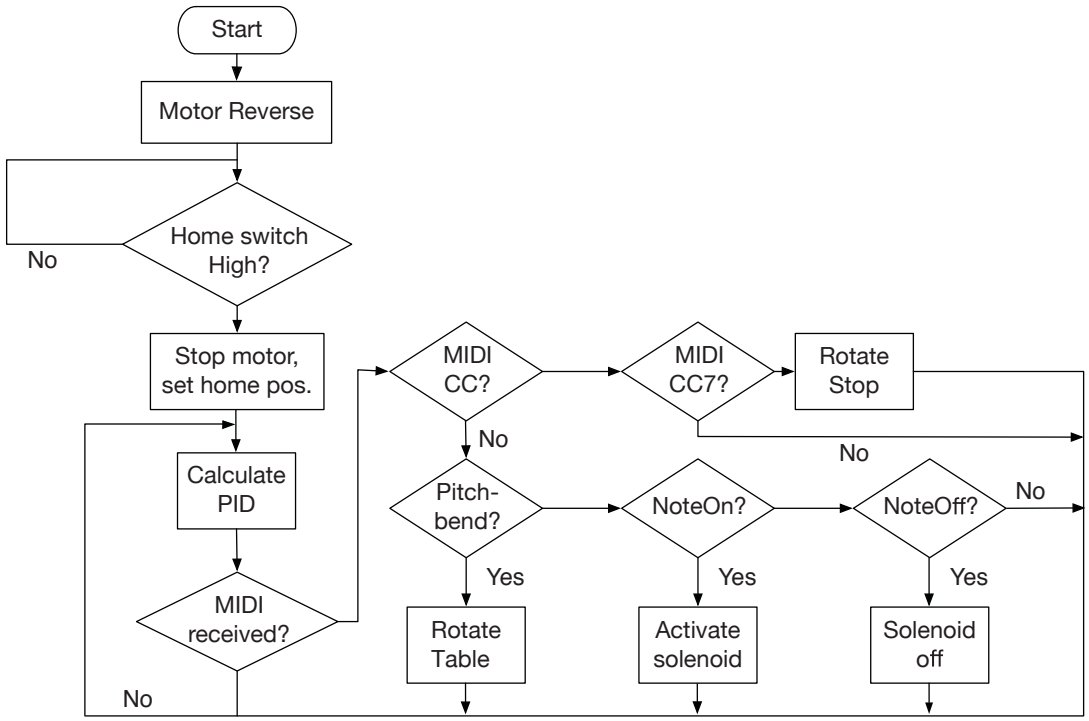


FIGURE 5.14: A program flow diagram of Nudge's firmware.

The main processor's roles are threefold: to initialise Nudge to a state allowing for repeatable and predictable performance, to interpret incoming communications and parse them into actuator control signals, and to perform any closed-loop control computations.

Upon system startup, the first of the processor's roles is to place Nudge's actuators in a state ready for the arrival of MIDI commands. To accomplish this, two main operations are needed: to position the solenoid height stop servo's cam at a predefined angle and to guide Nudge's rotary-motion drumstick positioner to its home position. To position the solenoid height stop servo, a PPM signal is sent from the microcontroller to the RC servomotor's onboard electronics; the onboard electronics convert the PPM signal to an angle setpoint and move the shaft until the angle is reached. To home the drumstick positioner, the positioner's DC servomotor is instructed at startup to rotate anticlockwise at a slow speed until the turntable closes the home switch. Once the home switch is closed, the encoder count is zeroed, ready for the arrival of incoming MIDI commands. Such commands will result in control system setpoints offset relative to the home position.

After the initialisation routine, Nudge awaits the arrival of MIDI commands issued by its MIDI host device. As on Kritaanjli, MIDI messages are handled by the Arduino

Table 5.2: Nudge MIDI messages, their resultant actions, and their mapped and unmapped ranges.

MIDI Message	Action	Message Range	Mapped Range
NoteOn w/ Velocity	Activate rotary solenoid	0-127	0-255
NoteOff	Deactivate rotary solenoid	NA	NA
PitchBend	Rotate turntable	0-16383	0-800
CC 7 (Volume)	Rotate stick height stop	0-127	0-45

MIDI Library (described in detail in Appendix B). Upon the arrival of a command, the process illustrated in Figure 5.14 is executed. Nudge is configured to listen to four MIDI message types: NoteOff, NoteOn, PitchBend, and Control Change messages. Table 5.2 outlines the messages and their corresponding actions and mapped ranges.

Incoming NoteOn and NoteOff messages control the solenoid actuator's driver electronics (discussed below). Control Change 7 messages are mapped to PPM duty cycles to control the solenoid height stop cam's angle. Pitchbend messages are mapped to a range within the rotary drumstick positioner's workspace and are input as setpoints to the positioner's onboard PI controller (discussed in greater detail below). Such MIDI command parsing and motor driver command outputs are the main role of Nudge's ATMEGA328 processor.

In addition to parsing MIDI messages and outputting appropriate instructions to Nudge's actuator drivers, Nudge's ATMEGA328 implements a proportional-integral (PI) controller, closing the loop formed by the DC servomotor's driver and its Hall Effect rotary position encoders.

The need for a closed-loop control scheme was found early in Nudge's design process. Prior to deciding to pursue such a closed-loop scheme, a simple open-loop DC motor control scheme was implemented and found to be incapable of repeatable performance: in this scheme, the motor was instructed to return to its home position and travel for a specified amount of time at a fixed output voltage. In spite of the constant travel time and motor voltage, the system was unable to consistently return to user-specifed points (likely due to inconsistencies in the gear train and axles' friction).

To allow for greater repeatability, closed loop control schemes were implemented. Very simple closed-loop schemes were first evaluated: a setpoint was input to the ATMEGA328 microcontroller, and a constant motor voltage was applied to the servomotor in order to drive it toward the setpoint. Upon reaching the setpoint, the motor was reversed. While

simple, this bang-bang controller produced outputs of sufficient instability to potentially damage Nudge if left online for an extended period of time. A control scheme able to vary the voltage sent to the motor based upon the position of the motor with respect to a setpoint was needed. While steady state error was undesirable, a small amount of overshoot and subsequent settling to the setpoint was deemed acceptable due to the inaudibility of such errors. The Arduino PID Library[8] was found to allow for such control schemes while remaining easy to implement on Nudge's microcontroller.

The Arduino PID Library can implement either a proportional (P), a proportional-integral (PI), or a proportional-integral-derivative (PID) controller. In testing the suitability of the library, the simplest P-only controller was first tested. As expected, the P controller was found to suffer from an unacceptable steady state error for any reasonable value of the P constant.

To overcome steady-state error and allow for smaller proportional gains, a PI controller was implemented using the Arduino PID Library. The PI controller was manually tuned to the system. With a proportional gain of 0.4 and an integral gain of 0.15, the system was found to respond to setpoint changes rapidly, with a small amount of overshoot and very little steady state error. The performance of the implemented PI controller is presented in Section 5.4.1.

The third role of Nudge's electronics subsystems is that of actuator control. Illustrated in Figures 5.12 and 5.13, the actuator control subsystem serves to receive the processor's output signals and convert them into motor actuation signals. Two separate Arduino daughterboards are used: one to drive the DC motor and a second to drive the drum beater solenoid[9].

As with Kritaanjli's keyboard playing solenoids (discussed in Section 4.3), a power MOSFET is used to switch on and off Nudge's solenoid actuator. After successfully employing the FDB7030BL N-Type power MOSFET on Kritaanjli, the same MOSFET was chosen to switch Nudge's drumstick solenoid. The circuit used is the same as as on Kritaanjli: a purpose-built PCB assembly, at left in Figure 5.13, was fabricated to connect the power MOSFET to the microcontroller and solenoid.

[8] http://playground.arduino.cc/Code/PIDLibrary (retrieved on April 15, 2014)

[9] The drum beater driver board also contains inputs for the encoder signals and outputs to the drumstick height stop servo.

The second daughterboard (designed by Victoria University of Wellington technician Jason Edwards) is used to drive the DC servomotor. The daughterboard, shown at right in Figure 5.13, employs an L298 dual motor driver and was selected due to its ease of interfacing with the Arduino Uno.

The miniature RC-style servomotor, like the servomotors used on Swivel 2, contain integrated driver electronics, requiring only a PPM waveform output from the microcontroller.

The final element of Nudge's electronics is its power system. A switched-mode AC to DC power supply is used to power Nudge's actuators and electronics. Two different input voltages are needed on Nudge: the DC servomotor and solenoid require 12 V DC, while the Arduino, motor drivers, encoder, and drumstick height stop miniature servomotor require 5 V DC. To obtain 5 V DC from the power supply's 12 V DC, the Arduino Uno's onboard 7805 voltage regulator is used. To determine the power supply's required amperage, the actuators' current draw at 12 V DC was measured. At 12 V, the drumstick solenoid draws up to 700 mA upon actuation; the DC servomotor draws up to 300 mA at 12 V. In total, Nudge may draw up to 1.5 A. A 36 W 12 V wall-mounted AC to DC power supply is used, chosen for its ability to provide sufficient power to Nudge while not approaching its maximum rated output current.

5.4 Evaluating and Characterising Nudge

To gain an understanding of Nudge's performance, a series of evaluations of the system were performed. This section details the evaluations; the findings presented here will not only demonstrate the characteristics of Nudge but will also provide future users with an awareness of its behaviour, allowing them to compose music in a manner that takes advantage of its capabilities.

In this section, three tests are performed, each of which serve to evaluate a particular design goal established at the start of the Nudge project (presented in Section 5.2).

5.4.1 Nudge Turntable Rotation Rate

It is important for composers working with Nudge's rotary drumstick positioner to understand how the turntable responds to setpoint changes, specified by varying MIDI pitchbend values. Such an understanding allows composers to use the positioner in a musically-meaningful way, utilising it to position the drumstick or other end effector. To gain an understanding of the turntable's performance, four metrics need to be known: the time required to rotate the turntable between different setpoints, the turntable's overshoot past the desired angle, the steady-state error (offset from the setpoint), and the settling time once the turntable has rotated to the setpoint. This section evaluates and discusses the musical implications of the rotary drumstick positioner's performance.

The PI controller discussed above in Section 5.3.2, is used to position the servo-driven turntable. While tuning the controller, it was noted that any tuning parameters would prove to be a compromise between system stability and response time: as a rapid response time with little steady-state error was desired, the proportional and integral gains were adjusted accordingly. As mentioned in Section 5.3.2, such adjustments resulted in a small amount of overshoot and a settling time. To quantify the overshoot and settling time, as well as to verify that minimal steady-state error is present, the servomotor's encoder values output during a series of setpoint changes are recorded and examined.

Figure 5.15 shows the encoder's output in response to setpoint changes: alternating setpoints are sent to Nudge, instructing it to move between encoder position 100 and 600, an angle of 135 degrees. Over six trials, an average time to move from setpoint to setpoint of 305 ms with a standard deviation of 18 ms is observed. Over six trials, an average overshoot of 21.0 degrees with a standard deviation of 0.7 degrees is observed. The average time over four trials required to settle to within five degrees of the setpoint (an angle deemed suitable for drum and percussion instrument performance) is 406 ms with a standard deviation of 22 ms. Finally, over four trials, a steady state error of 6.7 degrees is observed, with a standard deviation of 3.0 degrees.

These findings support the objectives to which the PI control scheme was tuned: the small amount of steady state error enables composers to direct the drumstick to a point in the positioner's workspace and, with an average of 6.7 degrees of error, obtain a desirable result. With the average measured repositioning times of 442 degrees per second, the setpoint transition time is suitably fast for repositioning events to occur

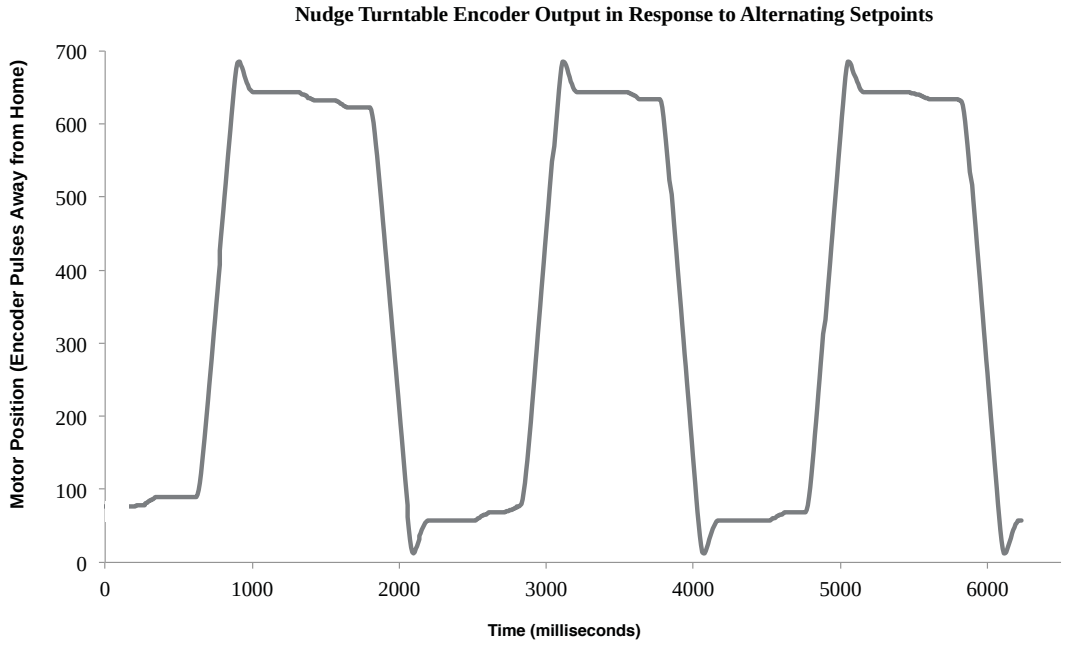


FIGURE 5.15: Nudge's encoder output as the turntable was instructed to move through a 135 degree arc.

within a single musical phrase. As anticipated in a system tuned to transition rapidly between setpoints, Nudge exhibits a degree of overshoot. As the effector is not in contact with the percussive object during occurrences of overshoot, there are few sonic ramifications as long as drumstick solenoid actuation events occur after the turntable has settled to within an acceptable proximity to the setpoint.

5.4.2 Nudge Actuation Rate and Dynamic Range

As prior mechatronic drumming systems were found lacking in variable actuation rates and dynamic ranges [57, 60], it was deemed important to evaluate these parameters on Nudge. After detailing the shortcomings present in preexisting mechatronic drumming systems, this subsection presents the evaluations and conclusions drawn from tests intended to explore Nudge's actuation rates and dynamic intensities at a variety of cam angles.

In the author's prior works (presented in [60], [57], and [55]), an assortment of typical solenoid-actuated drum beaters were evaluated and found to allow no greater than 10 consistent steps of discrete output powers. Even with such low resolution, standard deviations across trials of up to 25 percent of the system's dynamic range were observed

[60]. Additionally, as static devices, trade-offs were observed between dynamic output and actuation rate: as detailed in Section 5.3.1, the static nature of the drumstick's at-rest height resulted in a limited maximum repetition rate and dynamic output. If manually positioned close to a percussive object, actuation rates of greater than 1.35 kHz were observed.

Nudge's variable drumstick height cam, presented in Section 5.3.3, seeks to address the problems associated with statically-mounted solenoid-actuated drumsticks, allowing for online repositioning of the drumstick relative to the drum head, as shown in Figure 5.11. To assess the performance of this mechanism, a drum is first placed below Nudge's drumstick. Subsequently, the drumstick height cam is instructed to travel to a specified angle, at which point the solenoid is instructed to actuate at increasingly high repetition rates. The drum's output is monitored and the highest rate at which discrete repetition events are audibly present is noted. To determine the power of drum strike events, such events at each cam height are recorded at a sample rate of 44.1 kHz. The average RMS of the signal over 16,384 samples is measured and recorded as the output volume at a specified cam angle. As many of the author's (and author's collaborators') mechatronic percussion systems are used to strike frame drums, a similar 30 cm diameter Remo Fiberskyn 3 was used in the evaluations. The Fiberskyn 3 allows a drumstick to "bounce" off of its drumhead, an effect whose musical outcomes are discussed below. For both the dynamic range and actuation rate evaluations, three trials at each cam angle were performed. Ten evenly-spaced MIDI CC 7 messages (specifying cam angles) were sent from a host PC to Nudge; as shown in Figures 5.16 and 5.17, ten samples allow for a detailed examination of Nudge's note temporal range and dynamic range.

Figure 5.16 illustrates Nudge's note repetition rate at varying solenoid cam angles. Each data point presents the average of three trials; no standard deviation larger than 2 beats per minute was observed at any trial. The behaviour illustrated in Figure 5.16 largely matches the performance expectations of Nudge's solenoid height stop: as the cam's angle forces the effector's at-rest position closer to the drum head, the note repetition rate increases. The small standard deviation across multiple trials indicates a higher degree of repeatability than can be accomplished with non-augmented drum beaters [55, 59]. Such repeatability has direct musical implications: by being capable of repeatable performance, pre-composed compositions can be expected to produce similar results over multiple playback events. With such repeatability, composers can fine-tune the

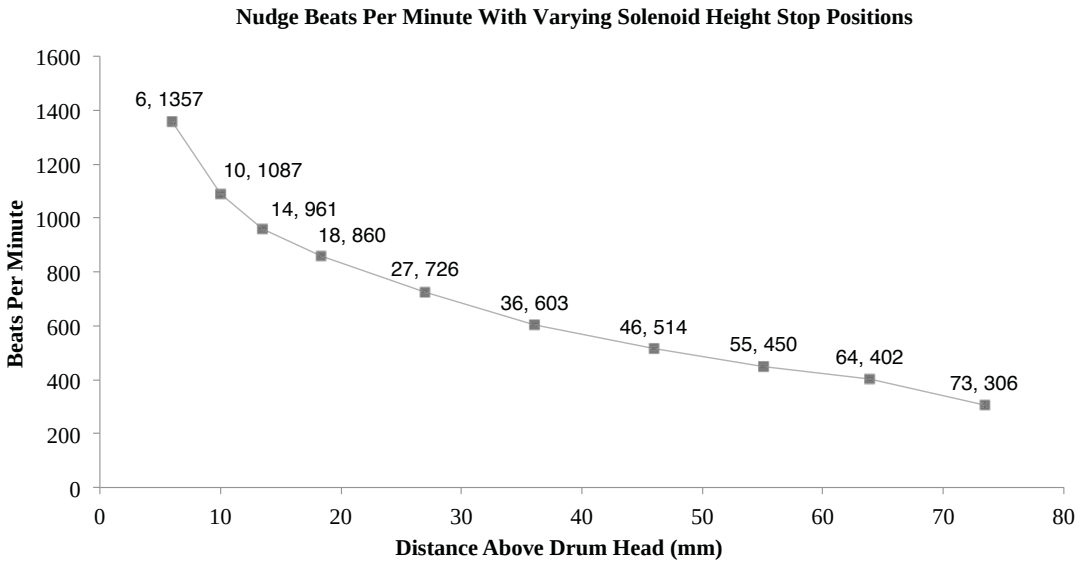


FIGURE 5.16: A sequence of Nudge's height stop servo positions. As the servo rotates clockwise, the drumstick is brought to an at-rest position closer to the drum head.

robot's expressive parameters prior to installation or performance and expect an accurate reproduction of the expressive phrases to be played back.

As shown in Figure 5.16, when the drumstick is positioned between 73 mm and 18 mm above the drumhead, the increase in playing rate as the drumstick is brought closer to the drumhead is largely linear. As the drumstick's at-rest position is set even closer to the drum head, the increase in actuation rate becomes nonlinear. The reason for this nonlinearity is likely due to the drumstick's bounce against the Remo Fiberskyn's drumhead: an actuation event results in the drumstick springing a small distance back off the drumhead. When the at-rest height is close to the drumhead, the bounce allows the drumstick to return to its at-rest height rapidly. The effect diminishes as the at-rest height increases, requiring the drumstick to return to its at-rest height largely with only the aid of its in-built return spring.

While maximum actuation rate increases as the drumstick is brought closer to the drumhead, Figure 5.17 shows that the RMS power of the signal's magnitude spectrum decreases with increasing proximity. To determine this, the drumstick's at-rest position is brought closer to the drum head. At each position, the drum-striking solenoid is actuated and its output recorded. A ChucK-based script was programmed [62], first performing an FFT on the recorded signal and subsequently calculating the RMS power of the signal's magnitude spectrum, a process similar to the means by which MechBass's

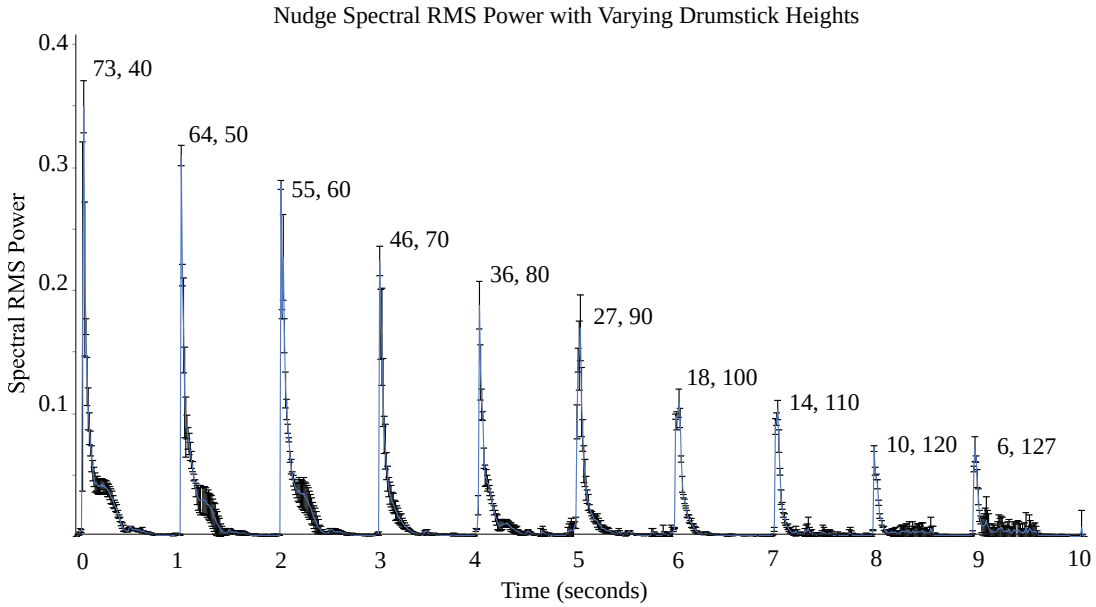


FIGURE 5.17: Nudge's RMS power with varying drumstick at-rest heights. The pairs of numbers at each peak represent the drumstick's height above the drumhead and the MIDI message sent to the servo-mounted cam. The blue line represents an averaged value, while the error bars at each datapoint represent the standard deviation over three averaged trials.

note picking intensity was measured in Chapter 3. The standard deviation from the average of three trials is shown in Figure 5.17. While greater than the standard deviation of the maximum actuation rate, a repeatable diminishing trend is evident in the signal's power as its proximity to the drumhead is increased. The results illustrated in Figure 5.17 are a significant improvement in both range and repeatability over the large errors present in the dynamic range evaluations of statically-mounted TrimpTron-style beaters [60].

The wide ranges of actuation speed and dynamic range options presented to a composer using Nudge suitably fulfill the initial design objectives stated in Section 5.2. While remaining relatively simple and of low-parts count (compared to other augmented percussion systems [27]), the evaluations presented in this section show that Nudge affords composers precise and repeatable control over the drum beater's position in relation to one or more percussive objects and precise and repeatable control over the output tempo and dynamics of the drum beater.

5.5 Summary

Mechatronic drum beaters are nearly ubiquitous in mechatronic music ensembles. While the keyboard playing instruments and chordophones described in previous chapters are relatively unusual, the literature contains many examples of robotic drumming systems. This chapter provides a review of a number of such systems, describing their often-limited degrees of freedom. While their positions may be adjusted manually prior to a performance, their orientation remains fixed during performance or installation use.

Nudge, a new mechatronic drum beater, contains two novel features intended to grant users more expressivity than is typical of robotic drumming systems. Firstly, Nudge's solenoid drum beater is mounted on a turntable, allowing a composer to specify the drumstick's position above a percussive object. Secondly, Nudge's drumstick's at-rest height can be adjusted with the aid of a servo-mounted cam. By adjusting the at-rest height of the drumstick, its repetition rate and dynamic range can be modified.

Part II

Interfacing With Expressive Musical Robots

Chapter 6

Robotic Chordophone Self-Tuning With Swivel 2

6.1 Self-Tuning: Motivations

Musical robots allow performers and composers to explore physical music-making with the use of new instruments that afford levels of precision and accuracy unusual for non-synthetic instruments. As physical objects, though, these instruments require adjustments prior to use: where a digital synthesizer's states are largely immune to the effects of the outside world, musical robots are inherently affected by environmental changes.

Without specialised calibration routines, the instruments often require time-consuming human intervention in order to behave as expected in a performance or installation context. Such intervention is not always practical: long-term installations, for example, render regular expert human calibration impractical. Automated calibration for musical robotics offers a solution: a musical robot capable of adjusting its future output based upon an analysis of its current outputs can be rapidly prepared for performance and remain reliable in long-term installation scenarios, simplifying the means by which humans may interface with it.

The work presented in this chapter focuses upon self-tuning for robotic chordophones. In order to more fully exploit their potential for repeatability and precision, self-tuning

systems can be employed on such instruments, allowing for automated tuning and on-the-fly tuning scheme changes.

The work presented in this chapter was motivated by the aforementioned problem of human intervention in an otherwise largely-automated mechatronic system. It was undertaken with three main beliefs about the functionality of a successful robotic self-tuning system: first, the automated tuning approach should work for any robotic stringed instrument capable of continuous (or high-resolution discrete) variation of string pitch: the tuning system should not be restricted to a single type of actuation, pitch shifting, or transduction approach. Secondly, the automatic tuning procedure should be designed such that it requires relatively little time to complete: a system capable of tuning a string with data received by a single note actuation event is preferred. Finally, a suitable self-tuning system should be able to not only tune the string's tonic pitch but to automatically assign virtual "fret" positions along the string, affording composers the ability to define the intonation of the instrument. With the above goals in mind, the work presented in the remainder of this chapter was undertaken.

The addition of self-tuning capabilities to a parametrically-rich robotic chordophone results in enhanced usability. Compositions for such instruments are not limited only to those wherein offline tuning is practical. Additionally, the ability to switch tuning schemes on the fly enhances an instrument's compositional flexibility and expressive potential. In effect, self-tuning of mechatronic instruments is both a practical and an expressive capability.

This chapter begins with an overview of prior advances in automated tuning for chordophones. After a brief history of self-tuning systems, the string-length adjustment approach developed for (and tested on) Swivel 2 is presented. Following the high-level presentation of the procedure, its implementation on Swivel 2 is shown, followed by an evaluation of the system's performance and a comparison of the new approach with a prior technique.

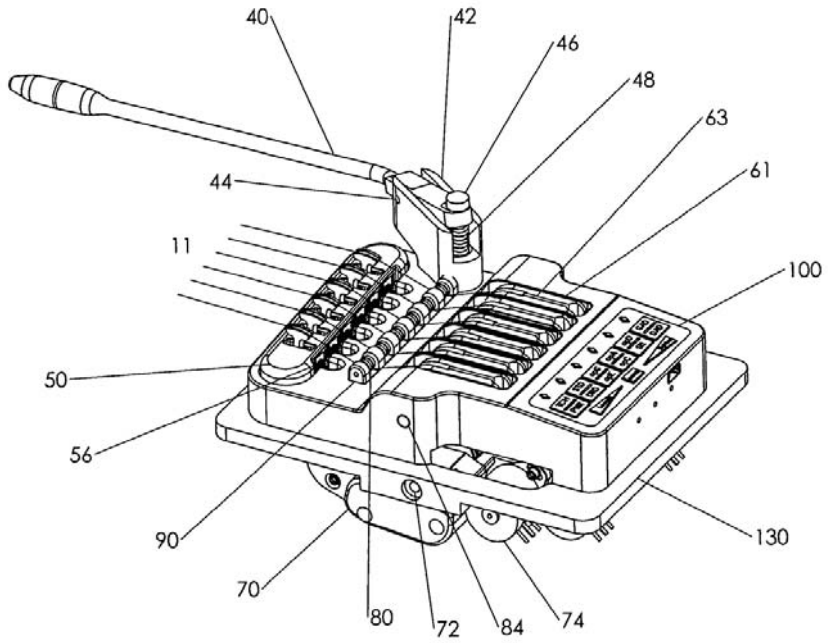


FIGURE 6.1: The AxCent tuning mechanism. Small motors adjust the strings' tension in response to the string's measured frequency. Reprinted from [63].

6.2 Automatic Tuning: Related Work

The idea of a self-tuning chordophone has interested researchers and inventers for decades. While some early examples focused upon self-tuning systems for pianos [32], the contemporary examples discussed below are applied predominantly to guitars. While the majority of the existent techniques have been developed to augment a human-played instrument (freeing a human player from the need to tune his or her instrument), a small number of the systems presented are applied to fully-mechatronic instruments.

The existing schemes can be divided into subgroups: those self-tuning systems applied to fretted or, conversely, fretless instruments, and those schemes that sense the string's pitch via tension sensing or, alternately, audio signal feature extraction. These techniques will be examined in more detail below.

There exist several self-tuning systems developed for human-played and robotic fretted guitars. The patent literature features different methods developed to aid human players in tuning instruments. The technique most closely matching the approach described in

this chapter is employed on guitars built by AxCent Tuning Systems and Transperformance LLC. These systems, described in [63] and [64], adjust the strings' tension by tightening or loosening each string with the aid of an array of small motors (as shown in Figure 6.1). As described in [63], the strings' characteristics are stored in calibration tables, allowing for rapid comparisons between a string's current performance and its desired tuned behavior.

A second commercial self-tuning guitar is marketed as the Gibson Robot Guitar[1]. In spite of its name, the guitar is a human-played instrument, featuring motorised string tensioners similar to those used on the AxCent systems. Unlike AxCent's self tuning system (whose motors are integrated into the guitar's body), the Gibson Robot Guitar uses motorised machine head tuning pegs.

While the aforementioned systems are commercial and designed to augment human-played instruments, the Logos Foundation musical robotic collective (described in [17] and [20]) has shared the design for a fretted mechatronic chordophone tuning system designed for musical robotics applications. The Logos Foundation's approach appears to be similar to the proprietary approach used on the Gibson Robot Guitar[2]: a motorised guitar tuning peg is adjusted based upon the string's frequency. This adjustment changes the string's tension, allowing for rapid re-tuning of the string.

Finally, sound artist Trimpin employs a self-tuning setup on his long-term kinetic sound sculpture *If VI Was IX* [19]. This system uses a modified commercial guitar tuner whose output controls a DC servomotor, and is described in more detail in [22]. *If VI Was IX* represents an ideal usage scenario for self-tuning guitars: as a long-term installation mounted in a difficult-to-access manner, the self-tuning capabilities of the guitars results in greatly simplified long-term maintenance, allowing for compositions that take full advantage of the instrument's expressivity without having to compositionally compromise for lack of regular tuning.

These fretted guitar tuning systems, unlike the robots discussed in this thesis, use fixed-position frets which are designed to closely approximate 12-tone equal temperment tuning. The frets largely predetermine the pitch intervals playable by the string: while

[1] http://www.gibson.com/robotguitar (Retrieved January 25, 2014)
[2] See logosfoundation.org/instrum_gwr/synchrochord (Retrieved January 25, 2014)

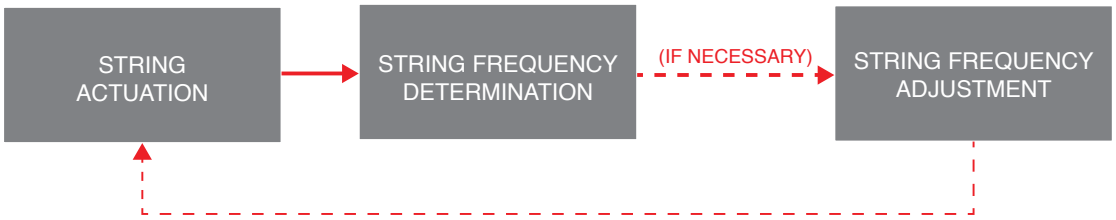


FIGURE 6.2: A typical self-tuning procedure: upon actuation, the string's frequency is measured and, if necessary, the string's tension is adjusted. After adjustment, the string is again actuated and measured.

a string can be tuned to an arbitrary pitch, the fixed-position frets remain immobile, largely restricting the instrument to a traditional chromatic scale.

There exist fewer examples of self-tuning systems for fretless instruments. A notable example is Zhen J. Wang and Cesar Ortega-Sanchez's Electronic Assisting Violin Tuner [65], which augments a traditional violin with a motorized tailpiece for string tensioning and a piezo-based pickup system: an array of small stepper motors is attached to the ends of the violin's strings. In a manner similar to the Logos Foundation's Synchrochord, the stepper motors turn to adjust the strings' tension. However, the fretless nature of Wang and Ortega-Sanchez's innovation, unlike the Syncrochord, allows players to adjust the strings' frequency and play in any intonation scheme they choose.

Self-tuning fretless instruments allow for a further degree of flexibility over fretted systems: after tuning, arbitrary notes can be played along the string's length. For human-played systems, this requires the player to possess the skill and dexterity to play without the aid of frets. For mechatronic systems, though, the lack of frets increases the instrument's flexibility by allowing for the use of "virtual fret" lookup tables, which allow for a very wide variety of tuning schemes and intonation styles to be employed on a single instrument. While the author is unaware of any prior fretless self-tuning fully-mechatronic chordophones, the relative ease of implementing such a system (as introduced in this chapter) indicates that many future robots will employ this scheme.

Automated tuning procedures typically consist of three steps: string actuation, string pitch or tension sensing, and string adjustment (illustrated in Figure 6.2). While actuation and adjustment depend on the instrument in question, there are two methods

often used for determining the string's pitch: frequency detection and string tension determiniation.

To tune a string based upon the string's tension, a transducer is mounted to the string. The transducer's output corresponds to the string's current tension. Combined with a knowledge of the string's gauge, length, and material, the vibratory frequency of the string can be determined. An example and explaination of such a tension-based approach can be found in [66], which details some of the shortcomings of the tension-based string tuning approach: the tension measurements depend on the string material, gauge, and age. Extensive recalibration is needed upon changing strings.

A more popular technique (used in the AxCent systems, as well as [65] and [67]) is to extact string frequency information from the output of the instrument's pickup. This approach is used in the work presented in this chapter: While frequency extraction is likely more computationally expensive than tension determination, an advantage of this approach over tension-based systems is that it typically requires no retrofitting of the instrument with additional transducers.

The academic literature contains few examples of self-tuning systems developed especially for musical robotics. The Stari [67] is one such example, consisting of a simple monochord on a test-bench-style setup: a motorised string tuner adjusts the string's tension. The string is actuated by a rotary-style pickwheel mechanism, and its vibrations are transduced and input to a computer via an audio interface. Where this chapter's research relies on a pre-compiled model of the string's response across many frequencies, Stari iteratively tunes itself from scratch (akin to the corpus-based approach of [47]). A computer script (implemented in the Pd graphical programming environment) instructs the motorised guitar tuner on Stari to gradually tighten or loosen itself based upon incoming pitch extraction data. Due to the digital nature of the motor control, an exact pitch is difficult to obtain and a 3 Hz threshold of acceptable deviation from the expected pitch is used.

To compare this iterative approach with the approach described below, an iterative string tuner was built for Swivel 2 alongside the primary approach presented in this chapter. This iterative tuner, described in Section 6.5.3, was found to require a considerable amount of running time and an undesirable pitch deviation threshold. A model-based approach, allowing for pre-compilation of the string's characteristics and subsequent

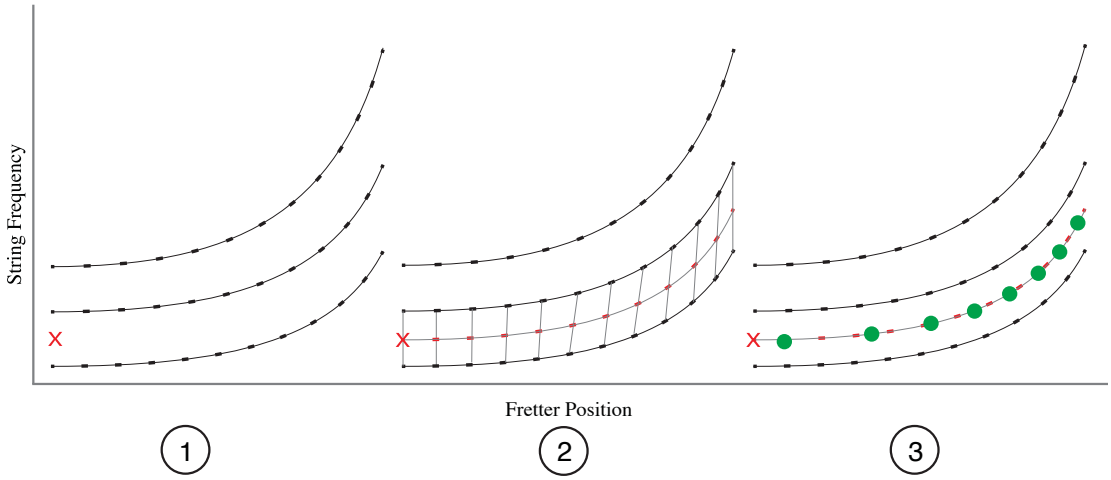


FIGURE 6.3: An illustration of the string tuning and fret assignment process. In *1*, the x represents the measured frequency of the detuned string in relation to the precompiled string characterization tables. In *2*, the detuned string's behaviour is interpolated from the characterisation tables. Finally, in *3*, the virtual frets (represented by circles along the plot) are assigned.

expedited online tuning, was found to allow both for more rapid tuning and for on-the-fly intonation scheme changes. The two approaches are compared in Section 6.5.3.

6.3 Self-Tuning Technique Overview

The self-tuning technique presented in this chapter has been developed for musical robots capable of relatively fine control over string length or tension. This need for fine resolution is due to the potentially small changes in string pitch needed to correctly tune the string. This section presents an overview of the self-tuning technique (an illustration of which presented in Figure 6.3), while the next section focuses upon the implementation of this method on Swivel 2. While Swivel 2 is focused upon in this chapter, the method described is not limited to one particular instrument. With modifications to the approach to account for varying actuation, pitch shifting, and transduction schemes, this approach is likely acceptable for any robotic chordophone capable of relatively high-resolution adjustment of its strings' vibratory frequencies.

The self-tuning scheme consists of two parts: a string characterisation phase and a string detune correction phase. The string characterisation phase is performed offline and results in the population of the string's response to a range of pitch shift events across

multiple fundamental string frequencies. The online string detune correction phase compares the open string's frequency with the pre-recorded string characterisation tables, allowing the string's deviation from the target pitch to be corrected. The following subsections detail both the string characterisation procedure and the detune correction process.

6.3.1 String Characterisation

The self-tuning approach described in this chapter aims not only to set the string to the correct fundamental frequency but also to allow the robot's pitch shifter mechanism to repeatably travel to any attainable frequency along the string's length. To couple the robot's pitch shifter (fretter) position with the string's vibratory frequency, the string's response to various fretter positions must be characterised.

To characterise the string's response to various fretter positions, the fretter is instructed to move to a number of points along the string. At each point, the string is picked or otherwise actuated and the resultant frequency is measured. The process is repeated incrementally along the string, with each value (and the corresponding position value sent to the fretter) stored in a table. After characterisation at one open string frequency, the string is retuned and the process repeated. By repeating the characterisation procedure at numerous fundamental string frequencies, an overall view of the string's response to different fretter positions at many string positions can be formed. Additionally, by interpolating between the string's responses to varying fretter positions at different tensions, a still-clearer picture of the string's overall characteristics across a range of tensions can be inferred. The means by which this procedure is undertaken on Swivel 2 is presented in Section 6.4.1.

This self-tuning approach benefits from large numbers of data points. Repeating the characterisation steps at many points along the string and at many fundamental string frequencies results in less interpolation-related error, as detuned strings are more likely to be close in pitch to a previously characterised string and fret placement is more likely to fall near a characterised point along the string. Implementers of this approach must choose an adequate number of samples to reduce interpolation-related errors to

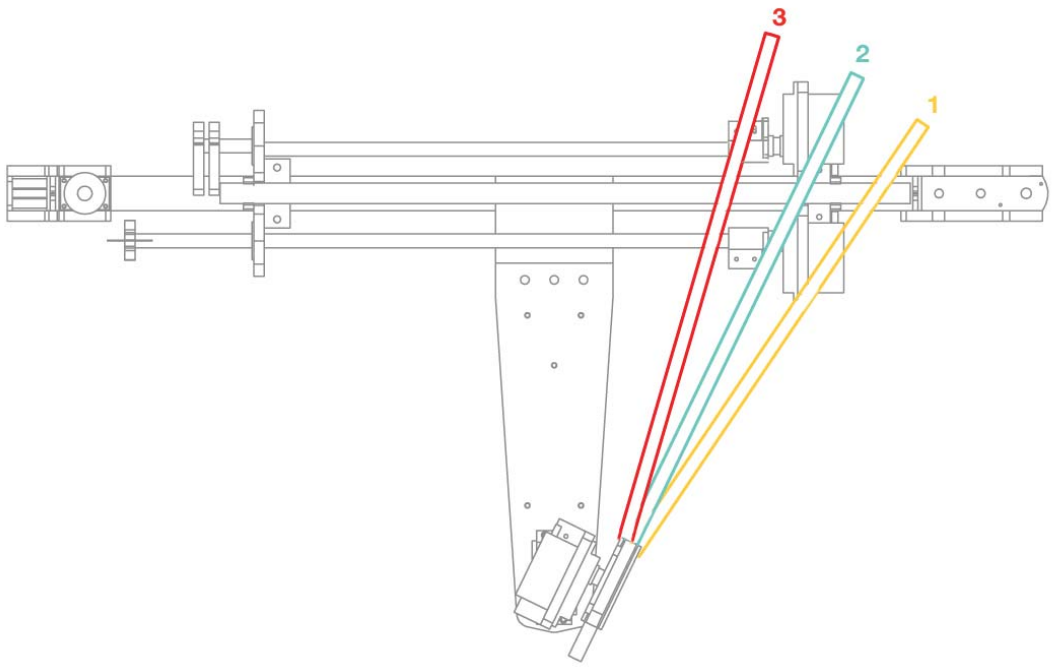


FIGURE 6.4: A drawing of Swivel 2, with its fretter at different home positions. After self-tuning, when instructed to play an "open string," the fretter moves to its home position and clamps against the string, changing the playable fundamental frequency of the string. In this illustration, position *1* would be applied to a tenser string than position *2*, which in turn would be applied to a tenser string than position *3*.

levels deemed acceptable[3]. In practice, an empirical approach is likely suitable. In such an approach, the string is characterised at a user-defined number of samples and, if interpolation-related error is detected, additional samples may be taken or different interpolation schemes tried.

6.3.2 Detune Correction

Once the string has been characterised at a variety of different fundamental frequencies, the string's behaviour at a frequency between two of the characterised frequencies can be inferred by interpolating between the upper and lower neighboring response curves. This is performed first by actuating the string and measuring the string's frequency. The string's current fundamental frequency is then compared with the previously-performed string characterisation tables. The behavior of the fretter in relation to the string at the current frequency can then be determined along the whole string length by interpolating between each of the two nearest characterisation tables.

[3]The threshold beyond which interpolation-related errors become unacceptable will vary from application to application, but should remain below what listeners can perceive.

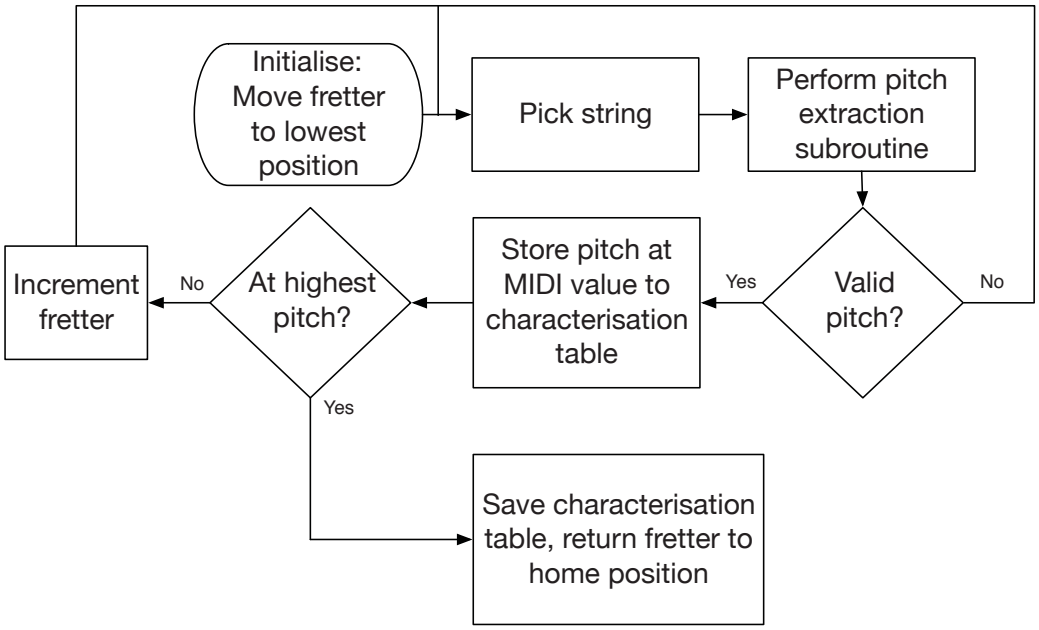


FIGURE 6.5: A program flow chart of the string characterisation as used on Swivel 2. This iterative approach allows for nonlinear fretter-to-string relationships (such as those on Swivel 2) to be easily characterized.

After the robot's fretter actuator position in relation to different string frequencies is known, the software-defined "frets" can be placed along the string length by populating another table with the fretter positions that correspond to the desired frequencies. Unlike physical frets, this table can be easily modified, potentially allowing for intonation changes to occur during performance.

6.4 Self-Tuning on Swivel 2

The self-tuning procedure described in this chapter has been implemented on Swivel 2. Swivel 2, introduced in Chapter 3, is a six-stringed robotic chordophone. To shift pitches, Swivel 2 rotates servo-mounted slides along each of its modules' strings. To change the fundamental frequency of the string, the "home" position of the fretter arm is changed, as illustrated in Figure 6.4.

Swivel 2 was selected as the mechatron on which to test the self tuning scheme presented in this chapter both due to its suitability to the approach and its need for self-tuning.

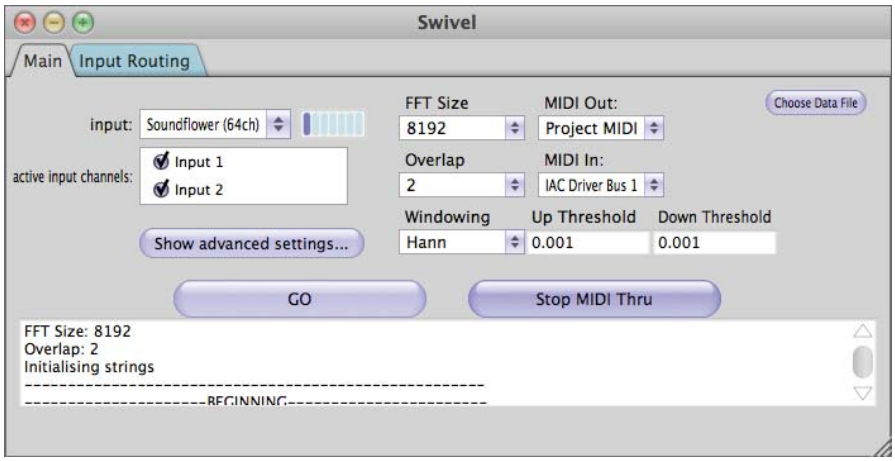


FIGURE 6.6: SwivelSelfTune's GUI: FFT settings, string characterisation files, and MIDI and audio input and output routing can be specified by the user. Once the string is tuned, the user can switch into "running mode" by pressing the "Start MIDI Thru" button.

Swivel 2 is capable of changing its strings' vibratory frequency at high resolutions, allowing the strings' behaviours to be characterised at a large number of points along their length. Additionally, Swivel 2's strings can be easily manually retuned with its integrated machine head tuning pegs, allowing the strings to be detuned and characterised at a number of different fundamental frequencies. Swivel 2 is a good candidate for self-tuning and fret assignment due to its default low-level interface. Without a self-tuning and fret-assigning system, Swivel 2 requires the composer to manually specify the two-byte pitchbend value for each note pitch. The results of a user study (presented in Chapter 8) indicated that users found this to be a slow way to interact with the system. Users requested the implementation of self-tuning and streamlined fret-choosing schemes for Swivel 2.

6.4.1 Characterising Swivel 2

To characterise the six strings of Swivel 2 (a process illustrated in Figure 6.5), a series of MIDI pitchbend commands are sent to each string. The pitchbend command instructs the fretter arm to rotate to an angle corresponding to the command's value (described in Chapter 3), after which the fretter arm clamps against the string and the string is picked. A module's string's vibrations are transduced by its magnetic pickup. The pitch is extracted from this signal. If the pitch lies within an expected range, it and its accompanying MIDI pitchbend value are stored in the string's characterisation table.

After the string has been characterised at a particular open string frequency, the string is retuned and the process repeated.

An FFT-based pitch extractor was built in the ChucK programming language. The pitch extractor's thresholds are manually specified, allowing users to set the extractor to disregard the strings' higher harmonics and any actuator noise tranduced by the strings' pickup. A sample rate of 44.1 kHz is used, along with an FFT size of 16384 samples. The FFT size was chosen to allow for suitable pitch discrimination at low frequencies while remaining able to detect the higher-pitched strings' frequencies before the fundamental frequency's decay below the noise floor.

Due to the mechanical and digital nature of the actuator control, the strings can only be tuned as precisely as the actuators' finest resolution. As such, the string characterisation tables are populated with 25 separate pitches, resulting in a curve found to be sufficiently smooth to avoid hindering the actuators' positioning.

6.4.2 Self-Tuning and Fret Assignment

Once the robot's strings are characterised, a purpose-built application is used to self-tune the strings. The application, dubbed SwivelSelfTune, was adapted from preliminary ChucK and Processing code into an integrated C++ application with assistance from student research assistant Paul F.C. Mathews. The application, whose GUI is shown in Figure 6.6, behaves in a manner illustrated in Figure 6.7.

Upon the user instructing the application to tune Swivel 2, SwivelSelfTune outputs a MIDI CC8 command with a value of 127, resulting in the Swivel 2 module lifting its fretter clear of the string. This allows the string to vibrate at its full detuned open-string length, unencumbered by the presence of the fretter. After a delay of three seconds (to allow the string to fully settle if disturbed by the movement of the fretter away from the string), SwivelSelfTune outputs a MIDI CC7 command with a value of 127, resulting in the module's pick moving across the string. The excited string's pitch is transduced by the module's pickup and is input into an ADC-equipped audio interface (a Presonus Firestudio Project Firewire interface was used to test the application on Swivel 2).

The signal's pitch is extracted in a manner similar to that used by the string characteriser (discussed above in Section 6.4.1). An advantage of the pitch extractor in

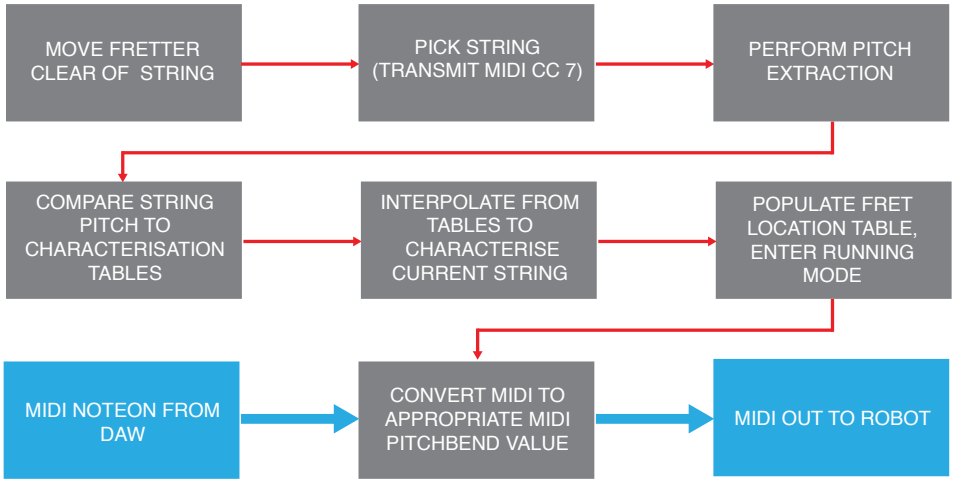


FIGURE 6.7: The SwivelSelfTune procedure.

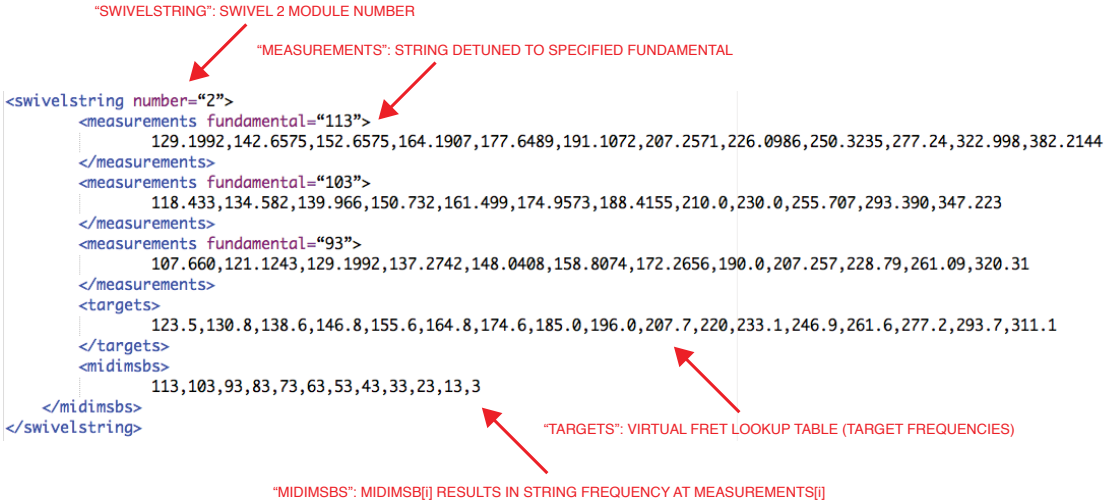


FIGURE 6.8: An annotated illustration of the SwivelSelfTune XML file.

SwivelSelfTune over that used by the string characteriser is that the pitch thresholds of SwivelSelfTune's extractor can be set relatively close to the expected fundamental frequency of the string. While the string characteriser must be configured to detect pitches far from the ideal fundamental as it moves along the string, the single open-string extraction event of SwivelSelfTune meant that emperically-derived thresholds of ± 20 Hz were found suitable, resulting in no detected harmonic-related false-positives.

After determining the pitch, SwivelSelfTune compares the detected pitch to a pre-populated pitch table. The pitch table consists of the results of the string characterisation steps at a variety of open-string frequencies. To allow for easy parsing of the pitch table, an XML-format file is used (an annotated illustration of the XML file is shown in Figure 6.8). In the XML file, each string is specified as a `<swivelstring>` tag, each of which contains a number of `<measurements>` tags (each `<measurements>` tag corresponds to a detuned fundamental frequency whose value is decided in the characterisation procedure). The `<measurements>` tags contain a list of numbers that correspond to the detected pitches at various points along the string. Also included within each `<swivelstring>` tag is a `<midimsbs>` tag that contains a list of the MIDI pitchbend most significant bytes whose index value corresponds to those within the `<measurements>` tag. Finally, the `<swivelstring>` tag contains a `<targets>` tag, which is the "virtual fret" lookup table. Changing the values in the `<targets>` tag is akin to moving the locations of the frets on a fretted guitar.

The detuned string's behaviour is interpolated from the two pre-characterised curves closest to its detected pitch[4]. The result of the interpolation is that a new table is formed, similar to the `<measurements>` tables (shown in Figure 6.8) but containing values interpolated from the nearest characterisation tables above and below the detected frequency. After the interpolated `<measurements>` table is created, the MIDI pitchbend values that correspond to its pitches are known (and are contained in the `<midimsbs>` table). The fretter arm can move to the "fret positions" in the `<targets>` table by interpolating between values in the `<midimsbs>` and the calculated `<measurements>` table.

At this stage, the string has been tuned and its virtual fret intervals set. The user can then instruct the SwivelSelfTune software to enter its running mode. In the running mode, MIDI messages from a digital audio workstation or other MIDI-compatible device are routed through SwivelSelfTune and transformed to MIDI pitchbend values that correspond to the desired frequency[5].

[4] With sufficient characterisation data, linear interpolation is suitable. If few characterisation points are taken, though, the error of such interpolation will likely prove audible to listeners.

[5] See www.github.com/PFCM/SwivelAutotune for full documentation of the GUI (retrieved May 10, 2014).

Table 6.1: Characterization Fundamental Pitches

Swivel Unit	Freq. 1 (Hz)	Freq. 2 (Hz)	Freq. 3 (Hz)
1	70.2	80.6	90.6
2	96.4	106	116
3	130	140	150
4	175	187	200
5	226	246	267
6	306	326	349

6.5 Results and Evaluation

To evaluate the performance of the automated self-tuning scheme, the full procedure was tested on Swivel 2[6]. The robots' strings were characterised and then manually detuned. The detuned strings were then retuned using the SwivelSelfTune application. After being retuned, Swivel 2's fretters were instructed to play chromatic scales (defined in the application's `<targets>` table). The string's actual frequency at each point was compared to the target frequency defined in the `<targets>` table, and the results are reported below, in Figures 6.10 and 6.11.

In addition to evaluating the performance of the SwivelSelfTune application, this section also details the performance evaluation of an iterative self-tuning system. An evaluation of such a system is worthwhile in that it compares the iterative approach (a technique presented in the only other existent mechatronic instrument self-tuning scholarly paper [67]) with the new approach described in this chapter. When compared with the iterative self-tuning technique, the advantages of the approach used for SwivelSelfTune are evident.

6.5.1 Swivel 2 String Characterisation

To characterise each of the six Swivel 2 strings, 25 measurements are taken along the length of the string. The pitch of the string at each of the 25 positions is measured, and this value is stored along with the MIDI pitchbend value which corresponds to the pitch. In this way, the manner in which the instrument's fretter arms affect the string's pitch at varying input values can be determined. The 25 measurements were predicted to produce an output curve of acceptable resolution, lying within the lower

[6]See https://vimeo.com/86156212 for a brief demonstration video of this process (retrieved February 11, 2014)

resolution of the miniature servomotors; on a robot with higher-resolution actuators, more measurements may be taken for increased accuracy.

Each string is characterised at three different open string pitches. These pitches, shown in Table 6.1, produce three different characterisation curves. The pitches were chosen empirically: the lower pitch (depicted by the black curves in Figure 6.9) was the loosest the string could be tuned while still being pickable by Swivel's picking system; the higher pitch (depicted by the blue curves in Figure 6.9) was deemed the tensest that Swivel 2's structure could reasonably withstand; the middle pitch (depicted by the red curves in Figure 6.9) lies halfway between the loose and tight string tensions. Taken together, the three measured string tensions provide an overview of the string's behaviour from its loosest to tensest. The characterisation curves used in SwivelSelfTune's configuration file (and shown in Figure 6.9) are populated with the average pitch obtained from a four second recording of the actuated string at a specified fretter angle. The characterisation pitch extraction is performed in the ChucK programming language with a sample rate of 44.1 kHz and an FFT size of 16,384 samples.

6.5.2 Self-Tuning

Two assumptions were made in characterising Swivel 2's strings: that 25 measurements along each string are sufficient to produce a smooth-sounding characterisation curve and that characterising the string at three different pitches is sufficient to allow for suitably smooth interpolation between them.

To evaluate these assumptions, the SwivelSelfTune application is run with the characterisation data sets described in the preceding subsection. After tuning the string, the robot modules are instructed to play 12-tone equal temperament scales; the output frequency is then measured and compared to the desired frequency.

Figure 6.10 illustrates the results of the evaluation on Swivel 2's A string. For the A string, the test is performed with the characterisation curves from "Unit 2" in Figure 6.9. The test consists of an instruction to move the fretter from a home position clear of the string to a position corresponding with a desired frequency. The string is then picked and its frequency measured with a sample rate of 44.1 kHz and an FFT size of 16,384 samples. The fretter is then instructed to move to and pick the next highest chromatic note along

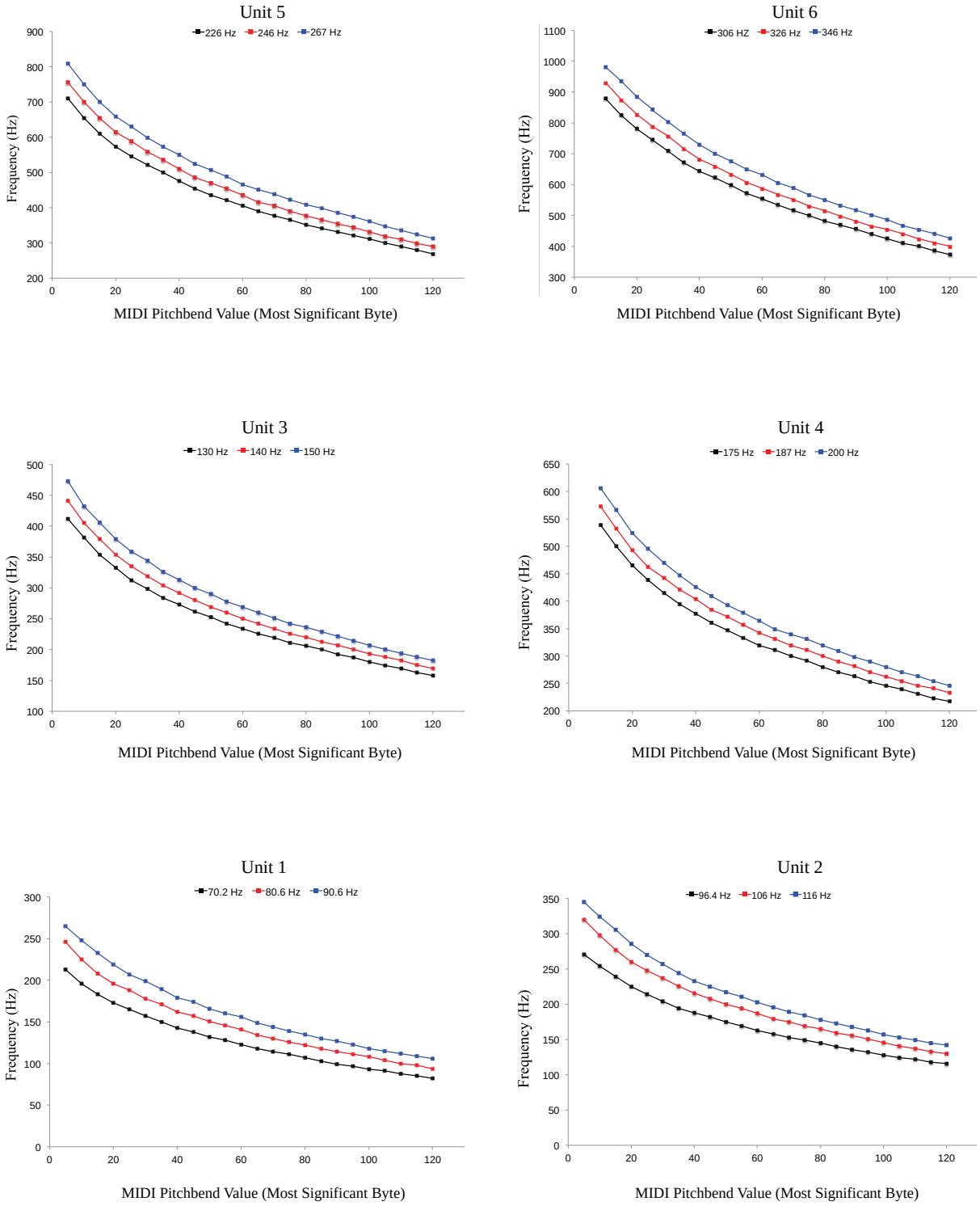


FIGURE 6.9: Swivel 2 characterisation curves. Each different coloured curve represents a different open string frequency (corresponding to the frequencies shown in Table 6.1). Each point is the average of ten FFT pitch extraction events.

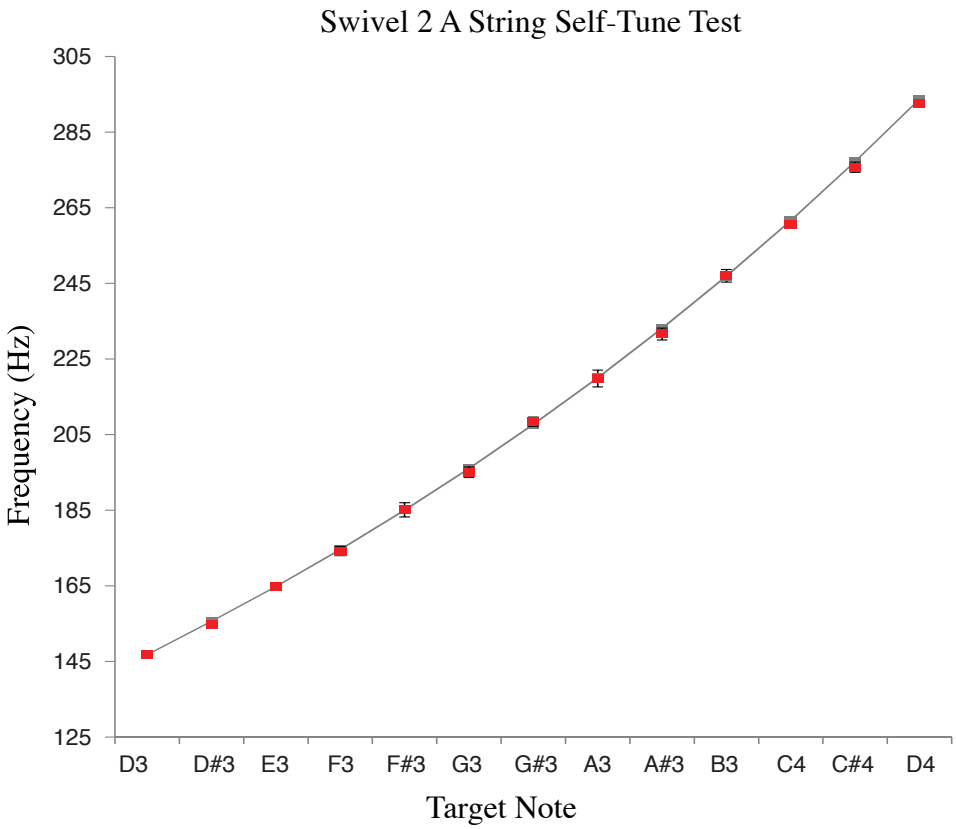


FIGURE 6.10: Swivel 2's A String Self Tuning Results. Error bars represent the standard deviation over three trials.

the string, continuing the process for one octave. This evaluation is repeated three times; the average results are shown, with the results' standard deviation represented by the plot's error bars.

After evaluating Swivel 2's A string, a similar test was performed on Swivel 2's B string. For the B string, the test is performed with the characterisation curves from "Unit 5" in Figure 6.9. The results from the evaluation of SwivelSelfTune's performance on Swivel 2's B string are shown in Figure 6.11.

The results of the tests show that the automatic tuning scheme produces a result that lies quite close to the desired pitches. The error, evidenced by the error bars in Figures 6.10 and 6.11 lies within the resolution of the average listeners' ability to distinguish pitches [68]. Further, the errors are both above and below the target, indicating that they are likely due not to interpolation error but are due to mechanical actuator resolution and placement: the table size of 25 target pitches across three characterization curves is therefore deemed sufficient for Swivel 2.

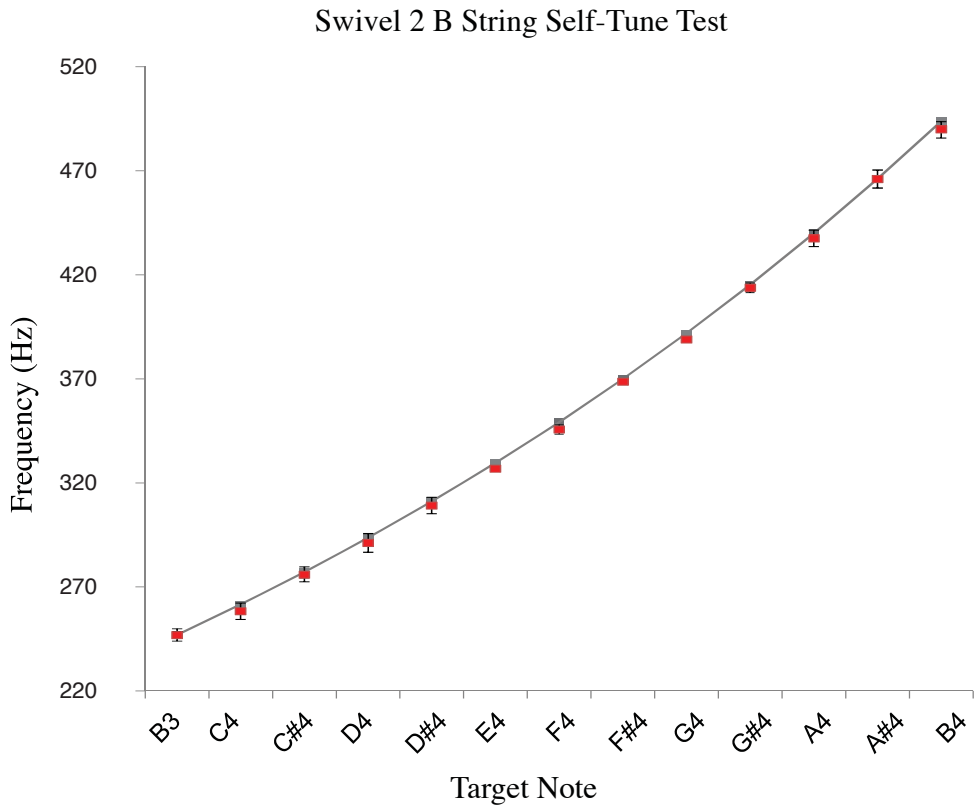


FIGURE 6.11: Swivel 2's B String Self Tuning Results. Error bars represent the standard deviation over three trials.

6.5.3 Iterative Self-Tuning Evaluation

To compare SwivelSelfTune to an iterative scheme (such as the "Stari" approach presented in [67]), a modified version of the Stari system was built for Swivel 2. The main appeal for such a self-tuning scheme is its simplicity: as described in [67], the string's tension is adjusted in a stepwise manner by a stepper motor until the extracted pitch of the string lies within a deadband around a target pitch. The deadband must be large enough to allow any tensioning event to fall within it, making such a tuning scheme necessarily imprecise.

The Stari-inspired iterative tuner for Swivel 2 functions by first determining the string's pitch at a MIDI pitchbend most significant byte value of 127 (see Appendix B for more details about pitchbend messages). An FFT-based pitch extractor with an FFT size of 16,384 samples and a sampling rate of 44.1 kHz is used. Swivel 2's fretter then moves toward the target pitch, checking the string's pitch at every step. The step size was mapped to 1 bit of the MIDI pitchbend message's most significant byte. Figure 6.12

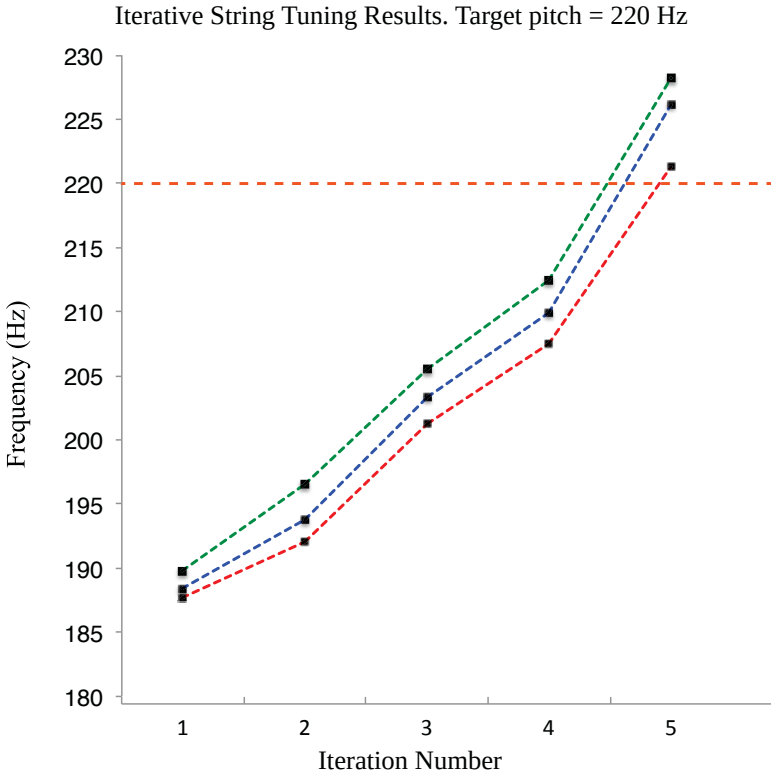


FIGURE 6.12: Three trials of the iterative self-tuning system. A target pitch of 220 Hz was specified. The graphs illustrate the iterative process by which the fretter approaches the target pitch from starting MIDI values between 120 and 127. The red, blue, and green lines indicate initial MIDI values of 127, 125, and 122 respectively.

shows the results of three trials with varying initial pitches and a target pitch of 220 Hz. As shown, five steps were required to move from the initial position to the target pitch. To populate a full array of "virtual frets" instead of the single fundamental frequency illustrated here, a very large number of steps would be necessary.

While this approach lends itself to more efficient search schemes than the linear scheme presented in [67] and replicated here, even a logarithmically-converging binary search approach would require more online search time than the single pick event required by the technique employed by SwivelSelfTune.

6.5.4 Summary and Performance Applications

Parametrically-complicated musical robots require many instructions to reach a desired state. The work presented in this chapter aims to reduce the amount of low-level control required by a human performer, musician, or installation artist. Without the aid of SwivelSelfTune, the string must be tuned by hand and each pitch shift event must

be manually directed by sending a human-derived MIDI pitchbend command, followed by clamping, picking, and damping instructions. Conversely, this self-tuning approach allows for tuned compositions to be made rapidly with relatively few commands and by those less familiar with the robot's interface. Indeed, the work presented in this chapter provies the field of mechatronic music two main contributions: a self-tuning approach requiring little on-line time and a method allowing for software-defined "fret locations" to be dictated by a composer.

To allow for daily self-tuning in long-term installation setups, the SwivelSelfTune application must be configured to run automatically upon system startup. While Swivel 2 has yet to be tested in such an environment, the increased ease of use will allow galleries and other installation venues to display the instrument with little need for human maintenance.

In performance scenarios requiring musician control over all of Swivel 2's parameters, a complicated musical robot's manual operation may be desirable, allowing for flexible human-in-the-loop exploration of the instrument's string behaviour. However, for more traditional performances, the automatic tuning method described in this chapter will likely be employed.

Chapter 7

Networking Parametrically-Rich Musical Robots

The musical robots presented in Part 1 of this thesis were designed with musical expressivity as a primary goal. As a way of achieving such expressivity, the mechatronic systems were built with many actutors mounted in manners allowing for the triggering of musical events. The abundance of actuators presents a challenge to those wishing to compose for the instruments: a large number of actuation parameters can prove difficult to interface with, potentially requiring numerous discrete commands. For a single musical event on Swivel 2, for example, up to five specific instructions may be required. Additionally, a composer or group of composers working with the systems presented in Part 1 would be confronted with a large number of robot-specific messages, requiring the learning of many different schemes for addressing the robots' actuators.

To address these problems, Tangle, a client/server application designed for the networked use of musical robotics, has been built with assistance from Victoria University of Wellington undergraduate student Paul Mathews. Tangle was built to allow for the low-level operational instructions of parametrically-rich musical robots to be encoded as higher-level actions, as well as to allow for the easy addition of robotic instruments to a preexisting Tangle network. An example of such a higher-level musical action would be the encoding of a tremolo-picking mode on MechBass. Previously, such an action would require manual placement of each pick event, with a high-level encoding of the

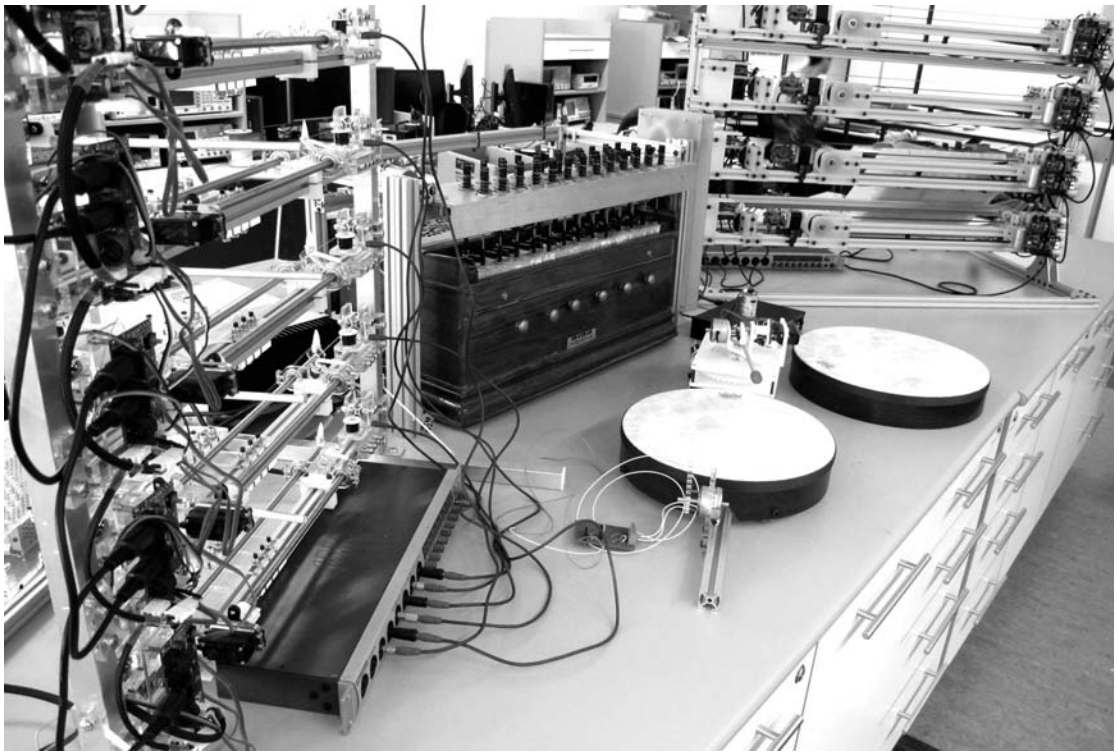


FIGURE 7.1: A typical Tangle deployment, consisting of a number of instruments, each with multiple actuators.

action. With Tangle, the composer need only specify three parameters: when to begin the tremolo pick, when to end it, and the time interval between picks.

Finally, as a third goal, Tangle was designed to allow for calibration and fail-safety functions to be implemented on a server. Such calibration allows for a wide range of diverse musical robots to behave in a relatively consistent manner. Fail-safe functions can be used to prevent the robotic instruments from entering unsafe states or remaining in undesirable states for extended periods of time.

This chapter begins by discussing influential historical musical networks, categorising Tangle based upon their precedent. Following the historical review and classification, an overview of Tangle is presented. Finally, example parametric encodings are shown, along with examples of fail safety and calibration functions.

7.1 Network Music

In [69], Marcelo Mortensen Wanderley and Nicola Orio present seven interaction contexts which are typical of human-in-the-loop computer music. While many of these may

be present in a composition or performance in which Tangle is utilised, Tangle is only responsible for the first of the seven: "note-level control or musical instrument manipulation." It is a feature of Tangle that it does not affect other aspects of a performance: by remaining agnostic of the other performance parameters described by Wanderley and Orio (such as tempo, post-processing, and dancer interaction), Tangle can be deployed in a diverse range of applications in a minimally-invasive manner.

Additionally, it can be useful to visualise Tangle as a networked portal, allowing many users to connect to it to share limited resources. While many network music frameworks (present in some of the output of The League of Automatic Composers [70] and later works by The Hub [71], PLOrk [72], and subsequent laptop and mobile device ensembles) are intended largely as a means by which networked performers may interact with and affect other performers' output, Tangle foregoes a focus on extensive inter-client data sharing and instead seeks to allow many users to control a server-connected range of musical robots in a repeatable and consistent manner.

Tangle derives much inspiration from Signal, a software suite developed by Owen Vallis, Dimitri Diakopoulos, Jordan Hochenbaum, and Ajay Kapur. Signal, detailed in [73], was designed to allow multiple performers to connect to the musical mechatronic instruments of the Machine Orchestra [6]. In the context of this research, however, Signal had three shortcomings which motivated the development of Tangle. The first identified shortcoming is Signal's lack of support for complicated parametric encodings. Such an inability to control many-parametered instruments is understandable when the instruments for which it was designed are examined: most are relatively simple mechatronic percussion devices of the sort detailed in Table 5.1, requiring little coordination of multiple actuation events to achieve musical output. When deployed with more parametrically-rich musical robotic systems (such as the configuration shown in Figure 7.1), such inflexibility results in the need for a composer to manually address each actuator involved in easily-encoded musical phrases rather than utilising a single command to instigate the entire phrase.

A second identified shortcoming of Signal is the lack of fail-safety and calibration routines. Signal serves only to route a client's actuation event to a musical robot, and cannot be configured to adjust its output based upon user-defined states. Empirical experience obtained during Signal evaluations with the KarmetiK Machine Orchestra

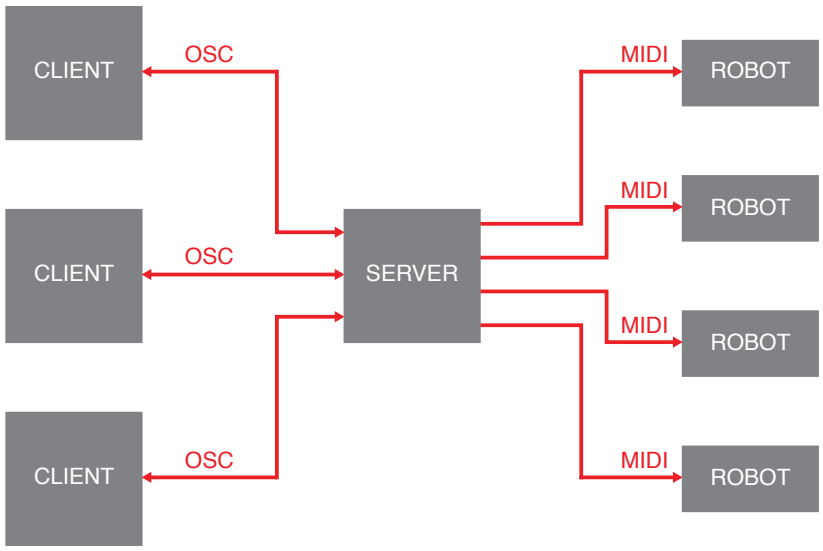


FIGURE 7.2: A diagram of an example Tangle network configuration: four robots are connected to three clients by the server. The clients communicate via the OSC protocol, which is converted to robot-compatible MIDI on the server.

indicated a need for such calibration and fail-safety functionality: in one such situation, an actuator remained in a powered state for longer than desirable, resulting in damage to actuator and robot. The use of fail-safety routines could aid in the avoidance of such damage.

Finally, Signal lacks the ability to alert clients of changes to the server's array of available robots. If additional robots are added to the ensemble, reprogramming of the client's software is required. As the robots described in Part 1 of this thesis may undergo modifications and firmware changes, such a need for client-side updating is impractical. A goal in the building of Tangle was for clients to be informed at runtime of the robots connected to the server and of their capabilities.

These three issues, coupled with the difficulty of maintaining the C++ source code of Signal, led to the development of Tangle. Tangle is a software suite designed with expressive musical robotics in mind. As such, it affords simplified control over robots requiring calibration, fail-safety, and sequences of actuation events. Additionally, Tangle does not forego Signal's (or Signal's predecessors') emphasis on the sharing of musical robotic instruments: as described below in Section 7.2, a large number users may connect to the server and concurrently use an arbitrarily large number of connected musical robots. In

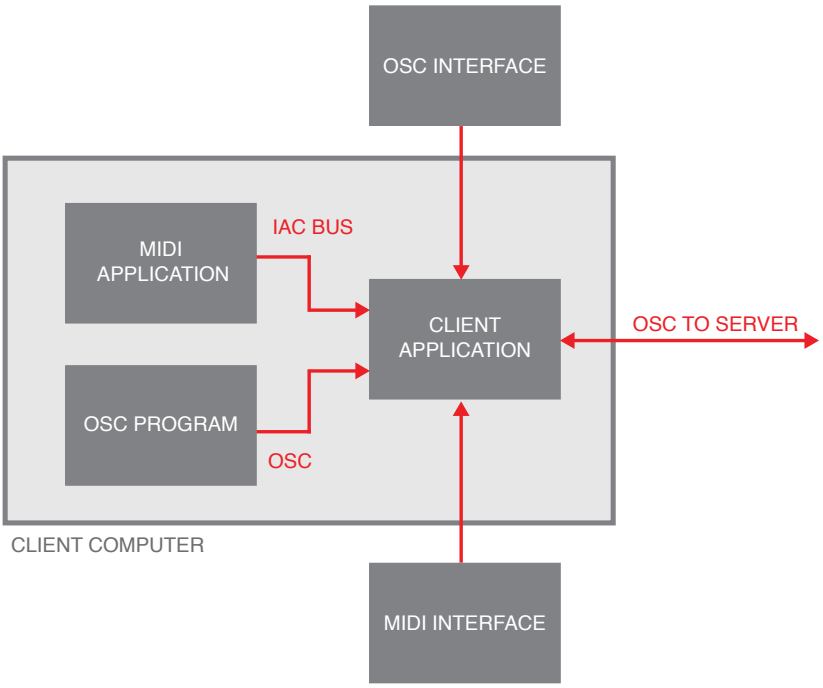


FIGURE 7.3: A diagram of an example Tangle client computer configuration: A digital audio workstation and an OSC application communicate internally to the client application. An OSC-compatible interface and an external MIDI interface also communicate with the client, which in turn sends OSC messages over the network to the server application.

essence, Tangle builds upon experience obtained in the use of Signal and Signal's simpler predecessors, sharing their motivations to afford collaborative use of many musical robots but extending their functionality to accommodate the new expressive systems described in this document.

7.2 Tangle: An Overview

Tangle consists of a client and a server, both written in the ChucK programming language [62]. ChucK is used due to its cross-platform capabilities, ease of maintainence, signal processing capabilities, and native support for OSC [61], MIDI, and audio input and output. It serves as a suitable programming environment for music-oriented projects that seek to interface devices with diverse input, output, and signal processing needs.

In spite of the use of MIDI as a communication protocol throughout this document's projects, OSC was chosen in favour of MIDI for communications between Tangle client and server. Where MIDI is chosen in the projects described in Chapters 3, 4, and 5

due in part to its minimal hardware requirments, the client and server computers used for Tangle are already equipped with the hardware and software stack required for OSC communication. Further, OSC allows for multiple client connections to the server and is easy to implement across an existing ethernet- or WiFi-based network. Because the client connects to the server with OSC, a large number of clients may be connected to the server at any time, allowing Tangle, like Signal, to be used in large-ensemble situations.

The client application is designed to be as low-maintainence as possible with an ability to automatically communicate with any changing number of server-attached musical robots. Such ease-of-maintainence allows users with little familiarity with networked mechatronic instrument performance to deploy Tangle networks. An example network diagram is shown in Figure 7.2, illustrating three clients connected to a server, to which four musical robots are connected.

Upon connection to the server, the client is notified of the names and number of musical robots connected to the server. Based upon this handshaking process between the server and client, a list of OSC messages to be transmitted to the server is created by the client. The server can be reconfigured by adding and removing robotic instruments without the need to update the client software. Such reconfigurability without the need for the patching of client software fulfils an initial goal in the design of Tangle: where Signal required client software updates if the number or behaviour of the robots connected to the server changed, such patching is unnecessary with Tangle. This expedited configuration is useful in performance environments where deployment time is often quite limited.

Users can communicate with the client software application in two ways, both of which are illustrated in Figure 7.3. A user can transmit MIDI to the client from any MIDI-compatible software. The MIDI is typically transmitted via a Mac OS X interapplication communication (IAC) bus[1]; MIDI channels are assigned to the available instruments by the clients and are printed out by the client computer's terminal. Alternatively, OSC can be transmitted directly from the user to the client. Such flexibility in software-to-client communications allows for a wide variety of music composition and performance software and hardware to be used on Tangle networks.

The server application runs on a single computer to which robotic instruments are attached. The primary role of the server is to listen for OSC messages sent via UDP by

[1] See Appendix B

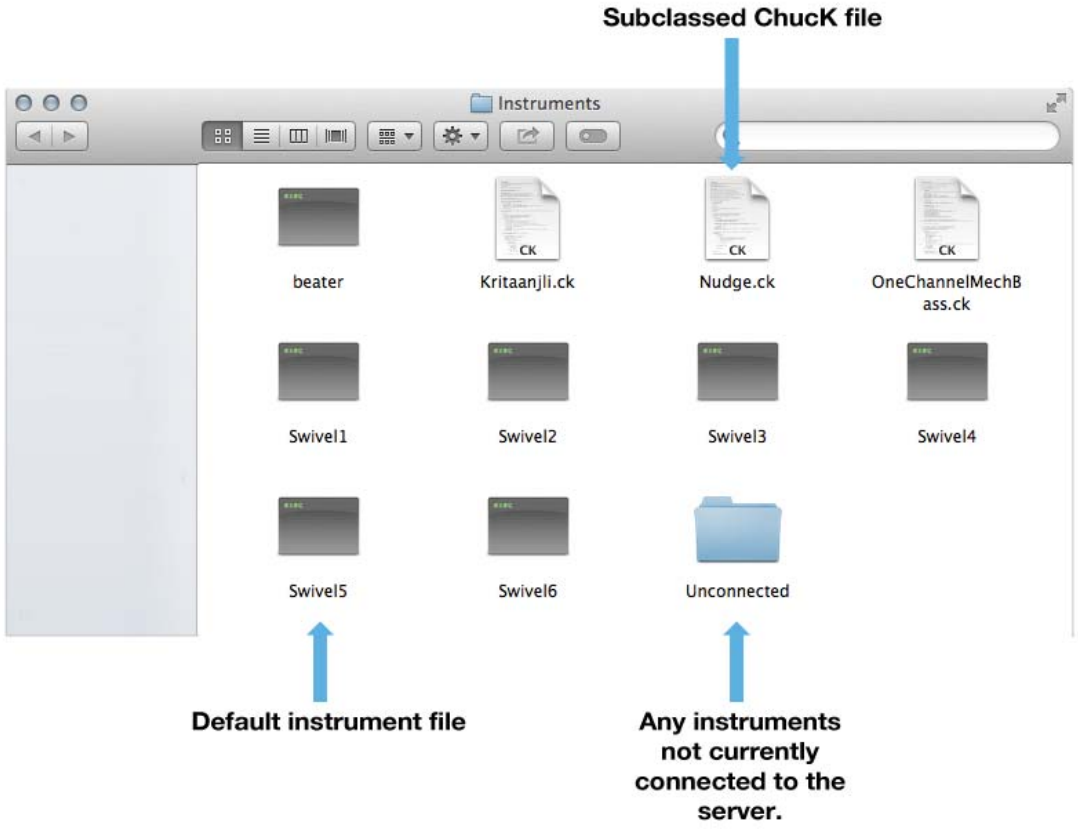


FIGURE 7.4: An example of Tangle's `Instruments` directory, located on the Tangle server. Two filetypes are shown: those with the `.ck` extension are subclassed from the Instrument class; those without an extension are default instrument configuration files. To temporarily remove an unused instrument from the Tangle server, its configuration files may be moved to the `Unconnected` directory.

any connected clients, converting them into commands appropriate for the musical robots connected to it. To fulfil this role, the server first engages in three actions: determination of connected instruments, the determination of those instruments' capabilities, and the sharing of these data with any connected clients.

The configuration files of currently connected instruments are stored inside a folder accessible to the server. This `Instruments` folder (an example of which is shown in Figure 7.4) is assumed to only contain files pertaining to instruments actually connected to the network. Each of the files in the `Instruments` folder describes a connected instrument: its name, MIDI port, and transmitted MIDI output events in response to input OSC messages are specified.

Two types of instrument files may be present in the Instrument directory. The simpler of the two is a "Basic MIDI Config File" that serves as arguments for an instantiation

of the default MIDI instrument class. Such basic configuration files, implemented as text files parsed by the Tangle server, lack the ability to execute any instrument-specific logic more complicated than simple note translations in direct response to incoming OSC events.

The second type of file that may be present in the `Instruments` directory is a subclassed version of the default Instrument class, implemented in the ChucK programming language. Subclassing the Instrument class allows for user-defined logic to be added. An example use case for a subclassed instance of Instrument would involve the use of instrument-specific fail-safe logic: by subclassing the Instrument class, the user can instruct the server to disable an actuator after a set amount of time or to restrict a motor from traveling past a certain point in its workspace. Such subclassing allows for the encoding of complicated musical phrases in response to simpler instructions; examples of such applications are described below.

After determining the config file-specified translations between incoming OSC messages and outgoing MIDI commands, the server sets up an OSC listener for each instrument[2]. By default, each instrument expects two OSC messages as input from the client: `/<instrument name>/note,ii` and `/<instrument name>/control,ii`. Any additional messages (such as pitchbend or noteOff commands) must be specified in the configuration file.

Once the OSC listeners are set up on the server, the server begins to listen for the arrival of clients. Clients connect to Tangle's server by transmitting over OSC `/system/addme,si` (along with the client's IP address and port number). Upon receipt of an `/addme` OSC message, the server sends to the newly connected client the previously constructed list of instruments. For those instruments with additional messages aside from the default two, the server sends the client a message containing the OSC message pattern pertaining to the the additional command. Finally, the server may send instrument-specific snippets of text to the client. This text contains simple human-readable usage instructions for each instrument, familiarising performers with the specific behaviour of particular musical robots connected to the Tangle server.

[2]For more about ChucK's OSC listeners, see http://chuck.cs.princeton.edu/doc/language/event.html (retrieved April 22, 2014)

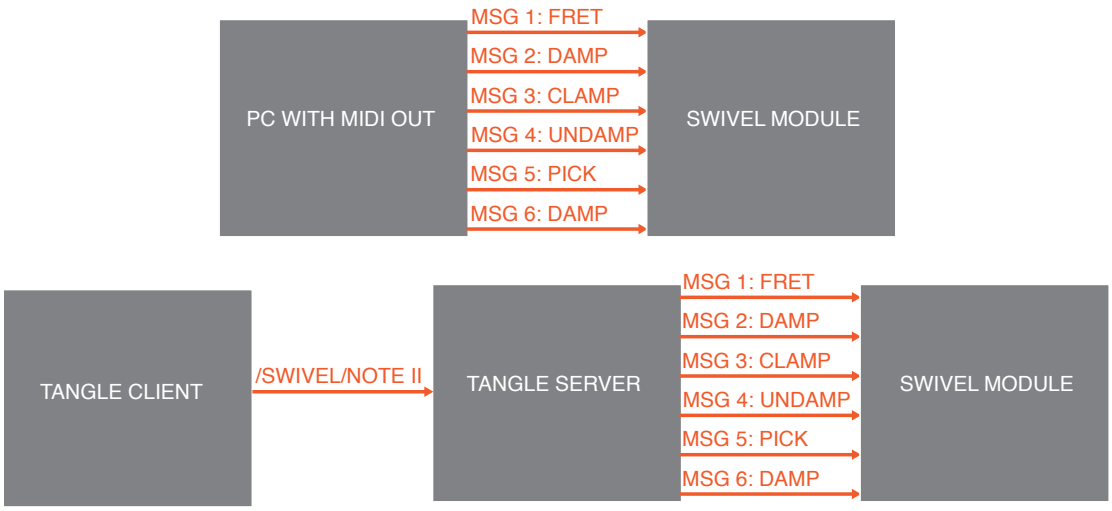


FIGURE 7.5: An example communications scheme featuring Tangle (bottom) presents users with a simplified interface: a single OSC message results in the server automatically outputting five sequenced MIDI messages; the top configuration shows the user's interface without Tangle. Six separate messages must be sent in sequence for a single pitched note picking event to occur.

After receiving the list of instruments and their unique behaviours, the client application creates a translation table, converting any received MIDI or OSC messages to server-appropriate output commands. Once this table is constructed, the server is ready to convert incoming OSC from the client into MIDI commands to control any attached musical robots[3].

Tangle is suited for use with the expressive musical robots described in Part 1 of this thesis for three main reasons: its ease of customisation allows the server to tailor its output messages to the heterogenous inputs required by the musical robots. Additionally, Tangle provides users with a straightforward means of accessing a diverse array of instruments: a user need not connect individually to the potentially large number of robots, instead accessing them via a wired or wireless UDP connection. Finally, Tangle can be configured to serve as an abstraction layer, arbitrarily simplifiying the interface with which users interact with the robots. Rather than send each robot a separate message, for example, Tangle can be configured to allow a single MIDI file to automatically address all of the connected robots.

[3]A detailed overview and description of Tangle's client and server architecture can be viewed at https://github.com/PFCM/Tangle (accessed February 11, 2014).

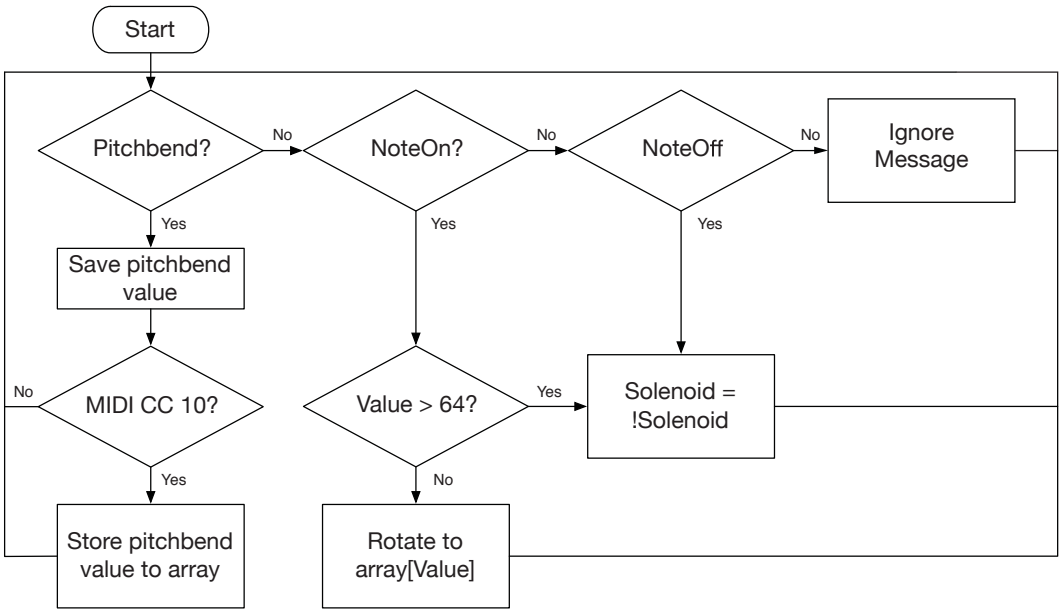


FIGURE 7.6: A program flow chart of Nudge's rotation position recall routine.

7.3 Parametric Encodings in Tangle

Tangle can encode sequences or combinations of MIDI actions, presenting clients with a simplified interface (exemplified in Figure 7.5). Additionally, instruments connected to Tangle can be endowed with server-side logic, allowing for simple sequences of client messages to trigger more complicated outcomes.

This section details a sample of such parametric encodings. The encodings are presented here as examples of means by which the act of interfacing with "parametrically rich" mechatronic instruments may be simplified. In essence, this section focuses on improving the act of interfacing with the complicated systems previously built.

While the examples shown pertain specifically to Swivel 2, MechBass, Kritaanjli, and Nudge, the idea of encoding sequences, combinations, and logical operations for musical robotic instruments may be extended both to simpler instruments and, especially, to future instruments equipped with still further degrees of freedom. The encodings described in this section may thus serve as inspiration to future roboticists wishing to equip their systems with such capabilities.

7.3.1 Nudge Rotation Position Recall with Tangle

As described in Chapter 5, Nudge is a mechatronic drum system capable of rotating its drum striker in response to a MIDI pitchbend instruction. While such drum striker rotation is a potentially expressive feature, allowing a composer to instruct the mechanism to hit a variety of areas on one drum or a number of different percussion objects, the parameter's use requires the repeated input of specific numbers. A means by which "preset" positions may be recalled will allow for a potentially easier-to-use interface.

An instrument subclass has been implemented on Tangle that enables users to save preset Nudge rotation positions and to recall the saved positions using specific user-defined NoteOn messages. This rotation position recall feature is of interest for three reasons: firstly, it allows for "preset"-style storage of rotation angles deemed musically interesting by a composer; secondly, it maps the fine-resolution MIDI pitchbend messages into simplified NoteOn messages, allowing for more rapid programming of rotation sequences; thirdly, it serves as an example of the implementation of relatively complicated instrument-specific logic within a Tangle subclass.

The rotation position recall program flow is illustrated in Figure 7.6. Once Tangle starts listening for messages from the client, it stores the most recent MIDI pitchbend value intended for Nudge. Upon receipt of a MIDI CC 10 message, the most recently received pitchbend value is stored in an array capable of holding up to 64 values. Once the pitchbend value is stored, it can be accessed by the client: to access the pitchbend value, a MIDI NoteOn command is sent from the client to the server. The pitchbend value at the index position of the NoteOn value is then transmitted to Nudge.

7.3.2 Staccato with Tangle and Kritaanjli

Kritaanjli, described in Chapter 4, is a mechatronic harmonium whose dynamic output is determined by a motorised bellows mechanism. As air is forced from the bellows, it escapes past any depressed key's reeds, resulting in a sound. A drawback of this system is that it results in a relatively long attack envelope: any key press and pumping event requires time to begin to vibrate the reeds. Through the use of subclassed instrument files in Tangle, this delay can be addressed, allowing for more abrupt staccato-like events to be played on Kritaanjli.

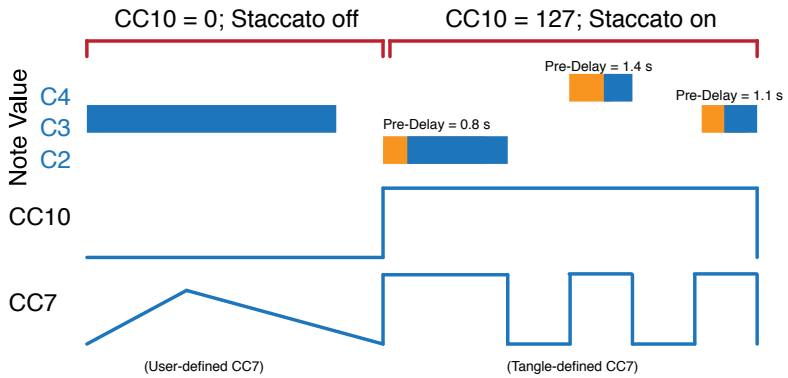


FIGURE 7.7: MIDI inputs and outputs associated with Tangle's Kritaanjli staccato mode. Staccato mode is enabled on the three notes on the right of the figure, with a CC10 value of 127. This results in early pump actuation and pre-delays applied to each note output.

Table 4.1 details the time between an instruction to play a note and to that note's initial sounding. As shown, more time is required for initial soundings of higher-pitched notes, ranging from 0.8 seconds for C2 to 2.4 seconds for C5. As illustrated in Figure 7.7, these delays can be compensated for in Tangle through the use of a subclassed Tangle Instrument class.

Tangle's staccato mode is enabled by a MIDI CC 10 message with a value of 127. Upon receipt of such a message from the client, the Tangle server enables staccato mode: all subsequent noteOn events will be handled by the Staccato function. The Staccato function is disabled again when the Tangle server receives any CC10 with a value lower than 127.

With the staccato mode enabled, a noteOn event "sporks" a "shred"[4]. Upon scheduling of the Staccato shred, Tangle immediately instructs the bellows to begin pumping. After a small delay, the note's solenoid is actuated. The delay's duration is determined by the note's octave and is timed to compensate for the results of Table 4.1.

While simpler in execution than Nudge's rotation position recall, Kritaanjli's staccato mode allows for the composer to explore a different playing style with Kritaanjli with the transmission of a single OSC message from client to server. By adding a pre-delay to the note, a composer must compensate by altering note placement. In situations where

[4]For more about ChucK's concurrency, see http://chuck.cs.princeton.edu/doc/language/spork.html (retrieved February 13, 2014).

a pre-composed song is played back on Kritaanjli, such compensation could be achieved with a DAW tool such as Ableton Live's delay compensation[5].

7.4 Fail-Safety and Calibration in Tangle

This section describes Tangle's fail-safety and calibration routines, providing two case studies for each. While the case studies presented pertain directly to the robots built and described in this thesis, the functionality of the calibration and fail-safety routines allows for easy extensibility to other instruments as they are built and integrated into the Tangle network.

7.4.1 Calibration Routines with Tangle

As physical objects, musical robots often perform best after some form of calibration. Such calibration is undertaken to place the robot's actual output states into agreeance with those expected of it. Calibration routines require feedback: the system's outputs must be altered based upon prior measured outputs. While such feedback may take the form of human-in-the-loop approaches, those presented in this section involve automated audio feature extraction; Tangle servers using such calibration routines therefore require an audio input device.

Tangle's calibration routines can take two forms: those integrated directly into the application's software or those running as separate applications configured to communicate with Tangle. An example of the second scenario is in the use of SwivelSelfTune (presented in Chapter 6) on a Tangle server. In such a configuration, illustrated in Figure 7.8, a Tangle client sends OSC to the server. The fretting-related OSC is converted to MIDI and output as NoteOn MIDI commands over an IAC bus to SwivelSelfTune; from SwivelSelfTune, the NoteOn commands are converted to appropriate pitchbend messages. Any other command types are output directly from Tangle to Swivel 2.

In addition to allowing external calibration software to be used, Tangle can implement internal calibration functions. To demonstrate this capability, a drum latency mitigation routine has been developed and tested.

[5] https://www.ableton.com/en/manual/mixing/#track-delays (retrieved February 13, 2014)

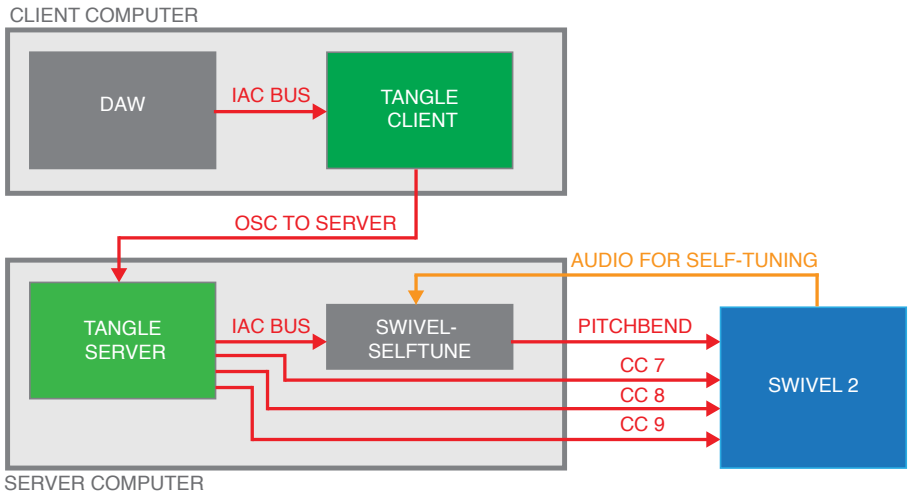


FIGURE 7.8: An illustration of SwivelSelfTune configured to co-function with a Tangle server. Clients communicate with the server as normal: to the client, SwivelSelfTune functions transparently.

Drum latency is a recurring problem among musical robotic ensembles [57]: if these ensembles feature a wide variety of drum actuator types mounted at varying distances from their respective drum heads, simultaneous drum actuation commands can result in drum strikes occuring at different times. To manually correct for such varied latencies is a time-consuming task: the drum actuator with the highest latency in the ensemble must be found, and all other drum beaters must be made to delay their own onset for the amount of time required for the slowest beater to strike the drum minus their own latency.

While manually achievable, such a process is time-consuming and lends itself to automation. As implemented on Tangle, an attached instrument is instructed to perform note actuation events. The output of this event is transduced with a microphone and analysed with an onset detection function implemented in Tangle. Upon detecting an onset, the elapsed time between the command's transmission and the onset is determined, as shown in Figure 7.9.

The process is repeated for each drum beater specified; a table consisting of the delays of all attached drum beaters is thus populated. After all specified beaters have been evaluated, the duration corresponding to the longest time between a drum strike instruction and a detected onset in found. Each instrument's delay is then subtracted from this maximum delay: $delay(i) = duration_{max} - duration_{instrument}$, where $delay(i)$ is the calibrated delay time, $duration_{max}$ is the greatest measured latency, and $duration_{instrument}$

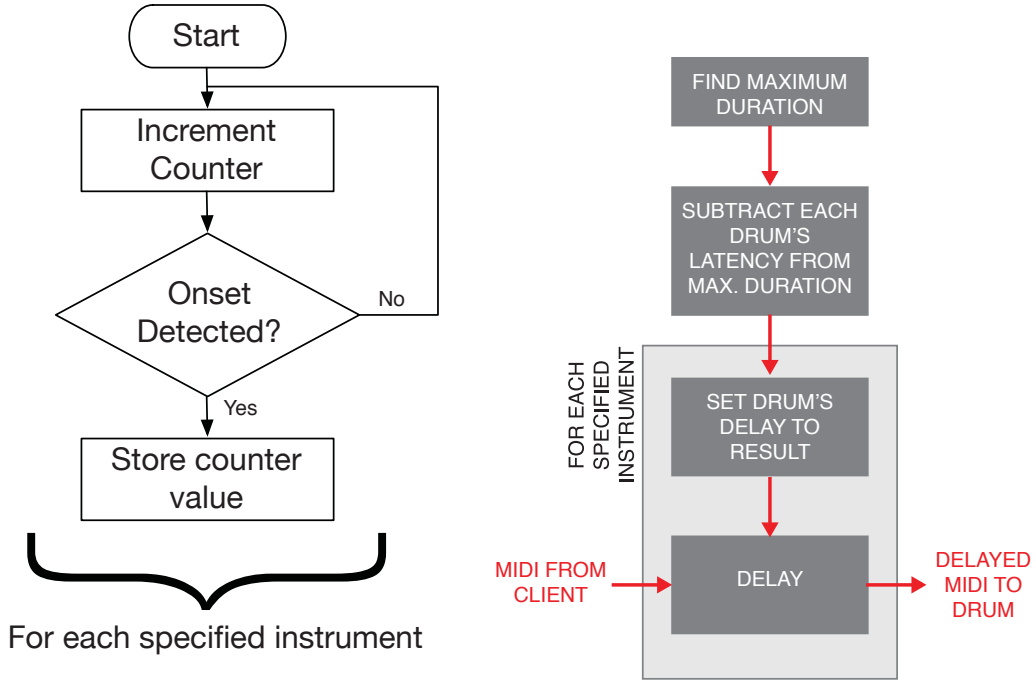


FIGURE 7.9: Illustrations of the Tangle drum latency calibration process. At left, the program flow for an individual drum's latency determination is shown; at right is shown the process of assigning delays to each drum and, subsequently, the delay of a particular incoming MIDI note.

is the particular beater's measured latency.

To evaluate the performance of Tangle's drum latency calibration, a TrimpTron-style drumbeater (described in greater detail in Chapter 5) was programmed to add an additional delay of 100 ms to any incoming MIDI actuation command. Tangle's drum latency calibration was then initialised, with the TrimpTron and Nudge as the two specified drums[6]. The calibration was performed three times on a system with a sample rate of 44.1 kHz; onset detection was performed with the aid of an FFT size of 1,024 samples. The results were recorded and observed. In spite of the 100 ms delay specified on the TrimpTron module, after calibration an average of 0.9 ms between the Nudge strike and the TrimpTron strike was observed, with a standard deviation of 0.1 ms.

The automatic latency calibration procedure is applicable both to the expressive musical robots designed in this thesis and to the simpler actuators currently employed in the Machine Orchestra [6] and other robotic drum ensembles. Additionally, the routine can be applied to non-percussive mechanisms such as the pickwheel on MechBass or Swivel

[6] A video demonstration of this evaluation can be viewed at http://vimeo.com/85770432 (retrieved February 23, 2014).

2. The routine introduces a delay to the actuators and may be undesirable in certain real-time playing conditions; as such, it may be turned off or specified for only specific instruments in an ensemble.

The two calibration routines presented in this subsection are intended not as an exhaustive survey of the possibilities for musical robotic calibration with Tangle-style networks, but rather an an overview of two possible approaches. In both cases, the calibration affords performers, composers, and musicians the ability to work with a large and potentially expressive array of musical robots without the need for low-level manual calibration of the instruments.

7.4.2 Fail-Safe Features in Tangle

A network of musical robots can feature hundreds of separate actuators, some of which can enter states that are undesirable and can lead to an instrument's damage. By monitoring the states of the instruments connected to it, Tangle can take action in the event of such undesirable states being entered.

By automatically monitoring attached instruments, Tangle allows composers and musicians to spend less time manually checking the musical robotic ensemble for faults and more time composing for the instruments. In this regard, Tangle's fail-safe mechanisms futher the software suite's overarching goal of affording greater potential to composers to interface with musical robots in an expressive rather than strictly technical manner.

This subsection details two case studies of fail-safe functions deployed on Tangle. While intended to demonstrate the possible applications of such fail-safe features, the described case studies are not intended to be exhaustive: if deemed necessary for the safety of the instruments or surroundings, with Tangle's instrument subclassing functionality, all of the actuators on the robots built in this thesis may be provided with fail-safe features.

The solenoid actuators used on Nudge and many other mechatronic drum beaters are prone to potentially excessive heating if current is allowed to flow through them for an extended duration. Tangle employs a function to automatically disable a solenoid actuator if a MIDI NoteOff command is not received within a specified time after the receipt of a NoteOn command.

While simple, Tangle's automatic solenoid deactivation functionality serves as a suitable demonstrator for its ability to integrate instrument-specific fail-safe routines. Upon receiving a MIDI NoteOn command (and resultantly actuating Nudge's solenoid), a timer implemented in a ChucK "shred" is "sporked," allowing the timer to run in a non-blocking manner. The timer continues counting until a NoteOff is received, at which point its value is reset and it is removed until the next NoteOn command. If no NoteOff command is received within a user-specified duration, a NoteOff command is sent from Tangle to Nudge's microcontroller, instructing the microcontroller to disable the solenoid. The duration's value is variable to allow for different actuators' different behaviours when left in a powered state.

Prior empirical experience with the array of mechatronic drum actuators of the KarmetiK Machine Orchestra [6] indicates that such solenoid failures can be difficult to visually or aurally detect, especially in large-scale ensembles featuring many actuators positioned on stage or in a gallery space. As such, the fail-safe solenoid deactivation feature removes a previously-required degree of micro-management from the drums' human users.

While the solenoid actuators take many minutes to become potentially hazardous, one of Kritaanjli's failure modes can occur quite quickly. The harmonium used on Kritaanjli is an artisanal instrument built in such a manner that pumping events conducted without accompanying key-press events can cause damage to the instrument's bellows: forcing air into the bellows without allowing it to flow out past a depressed key can cause the bellows to burst. This problem can be prevented with a fail-safe function implemented in Tangle.

Tangle's Kritaanjli bellows protection function works by tallying the number of notes being played. If it detects that no solenoids are being depressed, it deactivates the bellows pump. The bellows protection function has been integrated into Tangle's Kritaanjli instrument subclass; its program flow is shown in Figure 7.10.

A shortcoming of fail-safe implementation on Tangle is that in some cases it fails to account for the physical state of the instrument: as tested on Nudge, Tangle only evaluates the absense of NoteOff commands, failing to account for whether the solenoid has changed state. To remedy this, additional sensors may be attached to the robots,

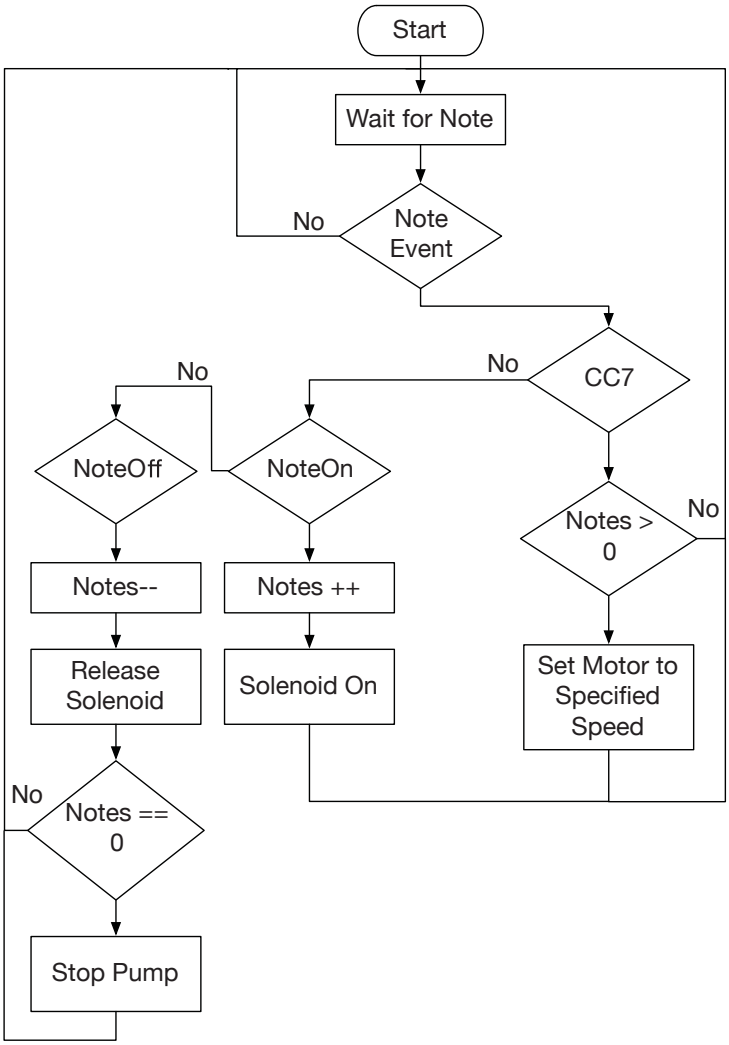


FIGURE 7.10: The program flow of Tangle's Kritaanjli bellows protection scheme, as implemented in the Kritaanjli subclass of Tangle's Instrument class. NoteOn commands are counted; if zero notes are detected, the bellows are prohibited from operating.

allowing Tangle to employ physically-significant inputs in its fail-safety routines. Additionally, redundant fail-safe functionality can be employed on the robots' microcontroller firmware, providing a second "line of defense" against damage to the musical robots.

With the calibration and fail-safety framework provided by Tangle, there exist many opportunities for investigations into more advanced online robot sensing and advanced control and calibration schemes.

7.5 Summary

A problem confronting those who use ensembles of mechatronic instruments is that of interfacing with each of them in a manner that is conducive to a compositional workflow. This problem is compounded when working with actuator-rich mechatronic instruments: an ensemble of such mechatrons may contain hundreds of individual actuators. To address this problem, Tangle, a suite of software for networking musical robotics, has been developed.

Three main design goals were declared at the start of Tangle's development. Tangle's client/server configuration is a result of real-world observations made during use of the KarmetiK Machine Orchestra ensemble of robotic instruments [6]. After using prior systems developed for the KarmetiK Machine Orchestra and noting that client-side updates were time consuming and frustrating to users, it was decided that Tangle must require no client-side updates in the event of the addition or removal of server-connected robots. Additionally, a goal in the design of Tangle was to allow performer/robot mappings to be easily changed. Finally, the ability to perform fail-safety checks and online calibration routines was deemed central to Tangle's functionality.

Tangle fulfils the above goals, providing easily-modified instrument mappings, calibration, and fail-safety functionality without the need for regular client updates.

The above features make Tangle a novel software suite, providing the community of mechatronic musical instrument builders for the first time with a free-to-use and -modify suite of tools that may be adjusted to work with any ensemble of MIDI or OSC-based mechatronic instruments.

Chapter 8

Using New Musical Robots: A User Study

> *"The mere presence of a finely calibrated instrument does not guarantee that it will be put to an expressive use."*
> —Christopher Dobrian and Daniel Koppelman [5].

After the development of MechBass, Swivel 2, Nudge, and Kritaanjli, as well as the Tangle network and all other accompanying interface software and firmware, a number of questions emerged that appeared unanswerable by the characterisation and evaluation of the instruments themselves. To address these questions, enumerated below, two parallel approaches are taken: firstly, a number of new compositions and installations are created for the instruments (described in Chapter **??**), allowing the author to personally experience performance and installation use of the systems. Secondly, as described in this chapter, the robots and software are made available to a number of musicians, and a series of questions intended to gain insight into musicians' use of the instruments are posed.

As parametrically-rich instruments, the musical robots run the risk of presenting users with a confusing array of options, allowing for precise control but potentially unintuitive mapping schemes. Dobrian and Koppelman, in [5], describe the dilemma facing designers of new musical instruments and interfaces: "It is one thing to create a controller with simple mappings that even a novice can use with satisfying results withtout training...

but it is quite another to develop an instrument that provides maximal control, diversity, and efficiency, in order to best enable expression by a skilled user." As described in Chapter 7, what is described in [5] as a "fly-by-wire," "divergent one-to-many" mapping scheme is implemented, "whereby a small amount of control data provides the necessary guidance to a complex system..." A goal of this user study was to examine users' interactions with the mappings, asking them questions as to their opinions and experiences with them. A parallel goal was to examine any differences in the perceived expressive potential between two systems with varying levels of mapping abstraction.

A second goal in the user study was to glean opinions from participants as to what parameters they felt most afforded expressive use and what parameters they would like to see added to the systems to further the instruments' expressive affordances. With a knowledge of the parameters that users found expressive, future expressive mechatronic instruments can be equipped with these parameters and, if not present in all instruments, such parameters can be retrofitted onto already-built instruments.

A final goal of the study was more practical in nature: the usability of the Tangle network software was evaluated. Tangle allows users to input many specific parameters, customising their networking sessions via a command line interface. Users familiar with electronic music composition and performance were asked to connect to the Tangle server and their experiences doing so were evaluated. The results of this evaluation are used to determine the need for changes in Tangle's parametric interface.

8.1 User Study Design and Coordination

The first step in gaining qualitative user input on the mechatronic instruments presented in this document was to identify the "target audience" for the instruments. From such a sampling frame, representative members could be selected for participation in user-based research. To determine the target users, an informal survey of the demographic makeups of three prior mechatronic instrument ensembles were conducted: of the KaremtiK Machine Orchestra's seven founding members, all were familiar with electronic music composition tools such as DAW software and MIDI interfaces [6]; similarly,

the seven founding members of Ensemble Robot possess music and engineering experience[1]. Finally, each of the eight "composer, programmer" musicians currently involved in the longstanding Logos Foundation's Man and Machine robot ensemble have experience with electroacoustic and synthetic composition techniques in addition to their work with the Foundation's mechatronic instruments[2]. While much future work could focus upon simplifying the process by which those unfamiliar with such techniques may interface with mechatronic instruments, it was decided that users with similar backgrounds to those in the aforementioned ensembles would be invited to participate in the study.

After choosing to invite musicians familiar with electroacoustic composition techniques to evaluate the mechatronic instruments, the number of users to involve in the study was decided. A purposive sampling technique was used: eight participants familiar with the aforementioned compositional and musicianship techniques were invited to evaluate the robots. Such purposeful sampling techniques were deemed acceptable given the niche-type skillset desired for participants. The first six questions on the user's study's questionnaire (shown with answers in Appendix C) detail participants' relevent technical experiences.

A survey-based study was designed, administered to participants as a questionnaire to be completed incrementally as they used the system. All questions and answers are presented in Appendix C.

The survey is divided into nine steps intended to follow a user's progress from first connection to the Tangle server to the use of the robots' parameters. In the first step, study participants are asked a number of questions relating to their prior experience with the technology employed in the user study. Participants are then asked to connect to the Tangle server from a client computer; after doing so, they answer a number of questions about their experiences with the Tangle client interface. Once connected to the server, participants are asked to use the Ableton Live DAW and Tangle to instruct Nudge to strike a drum. After gaining some familiarity with the means by which the instruments are addressed with the DAW and Tangle, study participants are invited to create a melody and pattern with the instruments, using all available parameters. After doing so, they are instructed to answer a number of questions on the provided

[1] See www.ensemblerobot.com/artists.html for biographies of the ensemble's artists (Retrieved April 30, 2014).

[2] See www.logotsfoundation.org/mnm/index.html for links to biographies of those involved with the Man and Machine ensemble (Retrieved April 30, 2014).

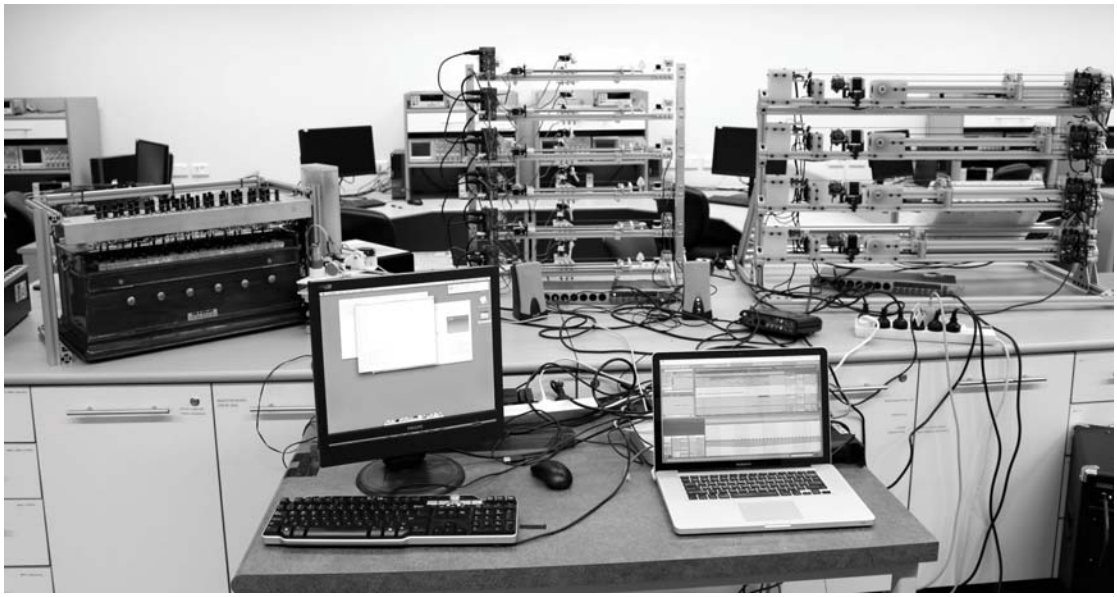


FIGURE 8.1: The user study setup. From left to right, Kritaanjli, Nudge, Swivel 2, and MechBass were controlled by the Tangle server (computer at left) and a client (computer at right, used by study participants).

survey about their experiences with the expressive parameters. Finally, a small number of general questions about the robot ensemble are asked to participants. Through these survey questions, the overarching questions that motivated the creation of this study may be addressed: users' experiences with the Tangle network and their opinions on the role of one-to-one versus one-to-many parametric mappings can be found, as well as the parameters that electroacoustic musicians find inspiring and expressive.

After gaining Victoria University of Wellington Human Ethics Approval[3], the study was conducted in a classroom at Victoria University of Wellington. MechBass, Swivel 2, Nudge, and Kritaanjli were set up in the room's centre. Study participants addressed the robots through a MacBook Pro laptop running Ableton Live 9[4] and the Tangle client. The setup was selected to simulate a likely performance-style configuration, with a number of robots controlled by a Tangle-equipped laptop. The Ableton Live DAW was chosen as it was deemed to allow for relatively simple means by which MIDI could be output to the Tangle client; its user interface is sufficiently similar to other DAWs that any study participants unfamiliar with it may be quickly introduced to its features[5]. The robot setup as used in the user study is shown in Figure 8.1.

[3] See Figure C.1 in Appendix C for more information

[4] www.ableton.com (Retrieved May 1, 2014)

[5] A pictorial quick-start guide to MIDI loop creation with Ableton Live 9 was provided to study participants.

8.2 Expressivity and Ease of Use

The debate between ease of use and parametric richness is an old one in the literature of new musical interface design. While researcher Perry Cook argues in favour of simplicity and ease of use [74], Sile O'Modhrain in [75] presents the case for more open-ended control of interfaces: "...a designer may also wish to leave room in their design for a skilled player to explore the 'corners' of an instrument's sound space, much as a skilled violinist can exploit extended playing technique that expands the range of bowing and fingering gestures."

While the debate about ease of use versus access to O'Modhrain's "corners" centers on digital input interfaces, the question of expressivity versus simplicity and compositional speed can be applied similarly to mechatronic instruments. To go toward answering this question, study participants were asked to interface first with MechBass, whose one-to-many parametric mapping allowed for rapid note-playing events, and then with Swivel 2, whose one-to-one mapping allowed for indpendent control over note pitch, damping, pressure on the string, and picking.

After playing both instruments, study participants were invited to compare their experiences, being asked whether "... Swivel 2 [would] have been easier to use if it was more like MechBass," and whether "Swivel 2 [would have been] as 'expressive' if it was more like MechBass." While users found MechBass easier to use, with two describing it as "very responsive," many agreed that Swivel afforded more expressive control. One user felt that Swivel 2 would be "easier to get going quickly but less expressive" if configured more like MechBass; a second user's comments were similar, saying that Swivel would be "easier to use, but not more expressive."

To address this dichotomy between MechBass and Swivel, one user suggested implementing multiple mappings for each instrument: "I like the level of control in Swivel, but I also like the ease of MechBass. Implement a hybrid..." Another participant advocated for different modes for different composition styles, arguing that Swivel 2 "would benefit from condensed commands," but that "both ways of controlling it would aid different composition techniques." Many other users' responses concurred (as presented in Appendix C). The Tangle network makes such customised mapping quite a straightforward process, as described in Chapter 7: multiple "fly-by-wire" modes could be created for a

single interface, accommodating both those wishing for rapid compositional prototyping and composers seeking fine-grained control over individual parameters.

Some users proposed a second way of addressing the problem of expressivity versus compositional ease-of-use for mechatronic instruments, pointing to the development of new input devices to allow for users to more quickly interface with robotic instruments. The user study's use of Ableton Live worked well for some users familiar with the software, while others found it to be a relatively difficult way of interfacing with the instruments. One participant noted that the DAW served as an adequate interface for offline composition, but would have preferred a "MIDI keyboard" for Kritaanjli or a "custom-designed interface" for Swivel 2. Workers in the field of musical robotics have noted this need: while Ajay Kapur and others have pioneered such input techniques (as described in [24]), there remains a need for custom software and hardware interfaces for parametrically-rich instruments such as those presented in this document.

Based on findings from the user study, one form of mapping that can enhance both ease of use and musical expressivity is that of self-tuning for chordophones. To determine whether to proceed with research into self-tuning systems for mechatronic chordophones, study participants were asked whether "Swivel 2 would be easier to use if it had the ability to tune itself." Such self-tuning is similar to the other mapping schemes discussed above, taking in a user's input and mapping it to a related output value. While one novice-level user stated that such tuning would not be useful at the user's "level of expertise," other participants indicated that the ability to self-tune would increase not only the instrument's ease of use, but also its expressivity: one user mentioned that such self-tuning would make playing "Western scales [and] temperment" easier, while expressing interest in the addition of non-western tuning schemes to the instrument, stating that"some JI [just intonation] tunings would be interesting." Further, users expressed a desire for additional pitch feedback when using Swivel 2. Such interest among study participants in self-tuning for chordophones not only as means by which their ease of use may be enhanced but also as a way to explore new tuning schemes motivated in part the continued development of the self-tuning system presented in Chapter 6.

8.3 Expressive Parameters

Another goal of the user study was to gain insight into what kinds of parameters users found to be expressive. Such insight will help to ensure that these parameters are made available on future mechatronic instruments and, if possible, added to existent instruments.

Many study participants equated continuous control with expressivity: parameters capable of being adjusted through a continuous range of values (rather than those only able to be switched on or off) were named by participants as the instruments' most expressive parameters: one participant noted that the continuous parameters of "balance and pitch-bend were particularly fun to manipulate." A second user indicated that the instruments' ability to adjust their output velocities was "most expressive," stating that "velocity control through all of the instruments seems to be one of the most important [parameters]. This [keeps] them from sounding monotonous and allows for more nuanced interaction between the instruments." Additionally, one user stated that "all of the dynamic controls" were most expressive: "Because these parameters provide the user with full control over the instrument," they allow for "a variety of timbral results."

Study participants' preferences for pseudo-continuous parameters over those capable only of what one user described as highly quantised "step"-style modulation goes toward validating this document's pursuit of expressivity through parametrically-rich mechatronic instruments with many degrees of freedom. Where previous percussion systems, for example, allow for only a single degree of continuous control, Nudge allows continuous control over strike actuation length, actuation intensity, and position on the drumstick (even Gil Weinberg's relatively expressive Haile only grants users two degrees of intensity rather than the continuously-adjustable solenoid height stop of Nudge [27]); similarly, Swivel 2 allows not only for continuously-adjustable portamento events, but also adjustable damping and fretter clamping amount. Kritaanjli and MechBass allow for adjustable output dynamics through variable pumping speed and adjustable pickwheel height, respectively.

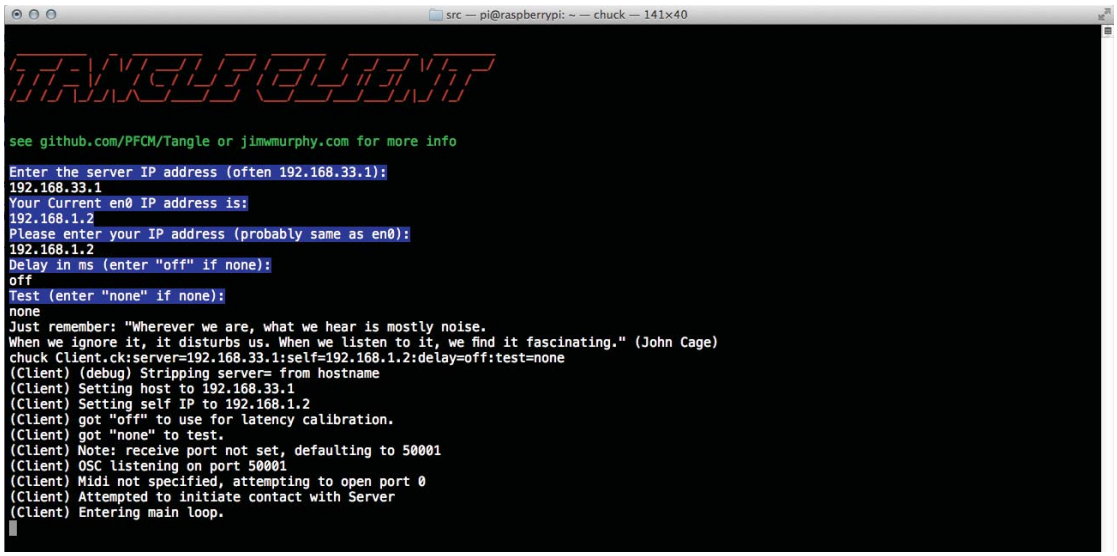


FIGURE 8.2: The Tangle Client Shell Script, developed in response to user study input. Users are prompted for Tangle Client arguments, removing the need to input the potentially complicated Tangle Client initialisation command.

8.4 Tangle: User Experience

A final objective of the user study presented in this chapter and in Appendix C was to gain insight into users' experiences with the Tangle client. The Tangle Client, like many of the instruments themselves, allows for much user-customisable configuration: as described in Chapter 7, users of the Tangle Client may input arguments into their connection request, changing many of the startup behaviours of the application (including latency calibration and delay compensation on a per-instrument basis). These parameters, entered manually into the command line, allow for much flexibility in configuration. To help determine whether users found such an interaction paradigm useful, study participants were asked to connect to the Tangle server from a Tangle client-equipped computer. After doing so, they answered questions regarding their experiences.

Prior to connecting to the server, study participants were asked about their familiarity with command line interfaces in general and the ChucK programming language in particular[6]. All participants were familiar with both ChucK and command line interfaces.

Due likely to their prior experience with command line interfaces and ChucK, most users were able to connect to the Tangle server with little or no assistance from the user study

[6]The ChucK-based Tangle Client is invoked through the command line; a combination of command line experience and some familiarity with ChucK-specific command line usage is therefore necessary to launch the Tangle client.

coordinator. In spite of this, users expressed their desire for an easier-to-use method of connecting (as indicated by their responses to questions 8 and 9 in Appendix C).

To go toward addressing their concerns with the potentially difficult and error-prone method of connecting to the Tangle server, a Bash shell script was written. This script, depicted in Figure 8.2, is prompt-driven, breaking the single long user input down into a number of smaller input events whose purposes are explained as the script runs.

The user study presented in this chapter served to answer to the author's satisfaction the four original questions that motivated its execution. While not all participants' answers to survey questions were featured in this chapter, even those responses not mentioned here are potentially enlightening and worthy of study: they provide insight into the means by which a number of composers view and interact with new mechatronic instruments. These answers, presented in full in Appendix C, contain many exciting ideas for further refinement applicable to current and future mechatronic instruments. Such ideas for future work catalysed in part by this study's findings are presented in more detail in Chapter 10.

8.5 User Study: Summary

An examination of the findings of the user study allow for a number of conclusions to be drawn. These conclusions can be applied retroactively to the robotic instruments built in the course of this thesis or, alternatively, applied to future mechatronic instruments.

Many participants in the user study indicated a desire to use dedicated human-to-robot interfaces for interacting with the mechatronic instruments. Such interfaces (including the eSitar used by Ajay Kapur and Curtis Bahn [6]) have been developed for mechantronically-simpler robots; the findings of this user study reveal a need for related interfaces for the parametrically-rich instruments developed herein.

In addition to indicating a desire among users for dedicated interfaces, the user study revealed participants' desire for multiple mappings on mechatronic instruments. These mappings can be configured to allow one-to-one mappings for "low level" control of the instruments' actuators, as well as potentially-intuitive one-to-many mappings for rapid "sketching" of musical ideas.

Finally, study findings revealed that users preferred the instruments' parameters' ranges to be restricted to values resulting in actual musical output events. On Nudge, for example, the turntable's default mapping allows the drumstick to be rotated past the edges of the drum used in the user study. This setting was chosen to allow for the easy deployment of a larger drum or multiple percussive objects. Users, though, found it easy to unintentionally rotate the drumstick beyond the drumhead. Prior to the user study, all workspace limits were defined in the instruments' microcontroller firmwares, requiring re-flashing of an instrument's firmware to adjust its workspace area. In response to participants' concerns, the Tangle server has been updated to allow for the easy definition of workspace limits, allowing composers to limit the range of the actuators from an easily-editable PC-based software interface.

While the user study supports this thesis's argument that increasing a mechatronic instrument's parametric density results in an increase in its expressivity, it also shows that the act of interfacing with musical robots becomes more complicated as the robots become more parametrically-rich. While an instrument of Nudge's parametric density appears manageable to users, exceedingly actuator-rich instruments such as Swivel 2 require efforts to be made to allow for streamlined user experiences. The steps taken to simplify the means of interfacing with the robots (including the development of the one-to-many mappings for MechBass and Tangle's customisable interfaces) appeared to meet with favourable responses from participants. Any future work on expressive musical robots must be undertaken with this study's findings in mind: at least as much effort should be spent on the development of interfaces for complicated musical robots as on the instruments' mechanisms themselves.

While this user study addresses only the instruments and interfaces presented in this document, its implications are farther-reaching. As the first academic user study of artists' experiences interacting with musical robots, this study should serve as the start of a long-term conversation: as new mechatronic music systems are developed, they should be accompanied not only by the sort of thorough technical evaluations presented in Part 1 of this thesis but also by detailed interviews with those who will end up using them.

Chapter 9

New Mechatronic Instruments: Performance and Installation

An overarching goal of the work presented in this document is to build systems for real-world performance and installation use. This chapter describes a number of installations and performances featuring the new instruments. Not intended to be an exhaustive exposition of the totality of the robots' capabilities, these works serve instead to introduce a number of performance and installation options ripe with potential for further exploration.

Construction of musical robots is an iterative process. While such iterations are typical of any development project, those in this document require regular performance and installation deployment as part of their iterative workflow. To this end, this chapter discusses not only the compositional structure of the performances and installations, but also the technical observations made during the robots' use: in effect, these use cases are "beta tests" for the robots, allowing their creators to observe their performance outside of the laboratory and, if necessary, make changes to them or future systems based upon such observations.

While development of the mechatronic instruments presented in this document has involved much informal setup and use, this chapter presents a number of more conventional performances and installations. The following sections first detail performances of mechatronic chordophone instruments, followed by installations featuring Nudge and

Kritaanjli. The chapter concludes with a section on networked ensemble use of all of the new robots described in this document.

9.1 Mechatronic Chordophones: Performances

9.1.1 *Spatial String*

Spatial String is a piece for mechatronic chordophones and multichannel audio. Intended to explore the notion of diffusion-style positioning of the sounds produced by a mechatronic instrument, *Spatial String* was performed at the Adam Art Gallery in Wellington, New Zealand on October 25, 2012. After a discussion of the motivations for creating the piece, the technical and compositional elements of *Spatial String* are presented. Finally, the technical and performative outcomes of the piece are discussed, along with further work catalysed by the performance's outcome.

In October of 2012, Swivel 1.0, shown in Figure 9.1, was completed. Unlike the resonant drums of Nudge and the vibrating reeds of Kritaanjli, Swivel 1.0's string is mostly inaudible unless amplified, allowing for the audio output of the instrument to be used as a sound source in a diffusion piece. The auditory presence of such an instrument can, in effect, be actively decoupled from its physical presence; *Spatial String* is a performative exploration of such sensory decoupling, involving the positioning of the robot's output in a multichannel sound-space.

Technically, *Spatial String* consists of a single Swivel 1.0 module controlled by a host laptop. The laptop communicates with Swivel 1.0 via MIDI commands; Swivel 1.0's optical pickup is connected to the laptop's audio interface. Input audio is processed with Ableton Live's built-in audio effects. The audio output of the laptop is input into a second laptop; the second laptop runs spatialisation software responsible for adjusting the gains of a quad-channel speaker array, as illustrated in Figure 9.2.

Spatial String (composed in collaboration with Bridget Johnson) is a live performance piece featuring two performers: the first triggers MIDI patterns sent to Swivel 1.0 and applies timbre-shifting affects to the audio; the second is responsible for diffusing the sounds throughout the performance space.

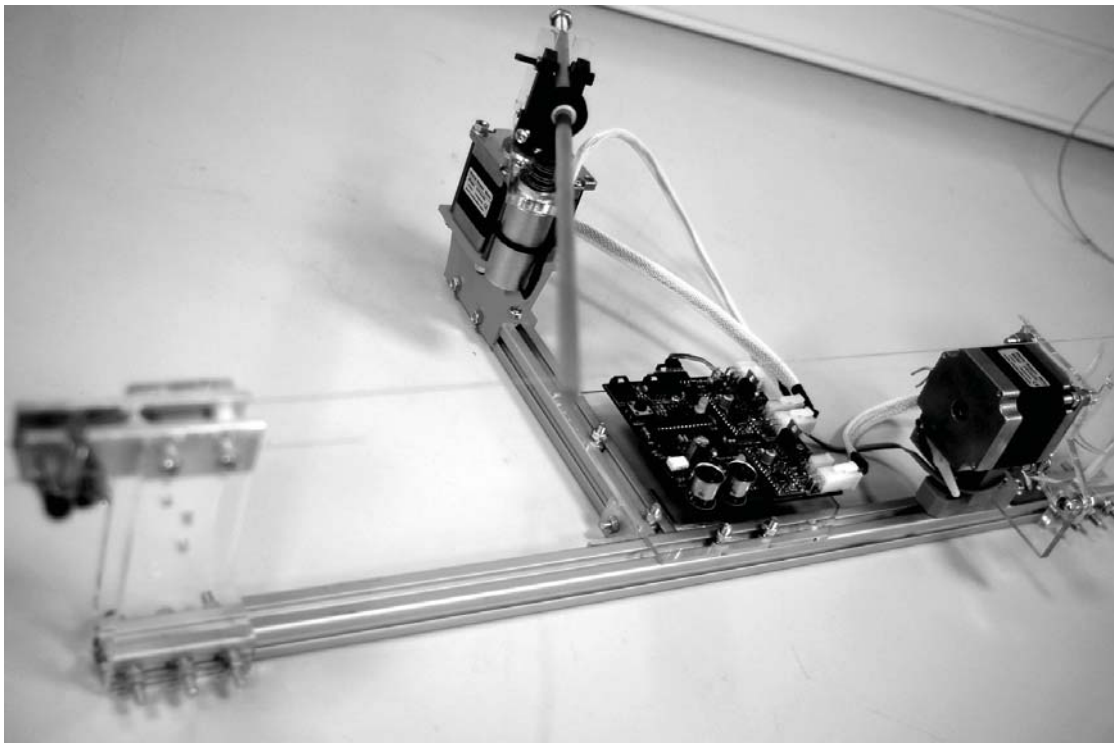


FIGURE 9.1: Swivel 1.0, used in *Spatial String*. Unlike Swivel 2, Swivel 1.0 uses stepper motors for string picking and fret positioning.

Twelve different MIDI patterns were composed for Swivel 1.0. These patterns consisted of a range of chromatic and arbitrarily-chosen string lengths and string picking patterns. Each pattern was configured as a loop, allowing for one pattern to be played repeatedly until the performer chose to switch to another. The incoming audio was modified manually by routing it through reverb, multiband EQ, and compressor effects.

In performance at Wellington's Adam Art Gallery, the loops were played in a sequence of increasing note-density over the course of the 10-minute performance: the performance began with a series of sparse, simple patterns and ended with denser patterns. The increasingly dense structure was chosen to allow the diffusion of sounds to begin in a manner allowing audience distinction of sounds (with discrete points of sound positioned throughout the performance space) and to conclude by presenting listeners with a timbrally-crowded space.

The *Spatial String* performance was a major motivating factor in the decision to pursue the construction of Swivel 2.0. In performance, Swivel 1.0 was found to be bulky and acoustically noisy: actuator sounds were more audible than desired. Additionally, the optical pickup was found to require large amounts of setup time in the performance

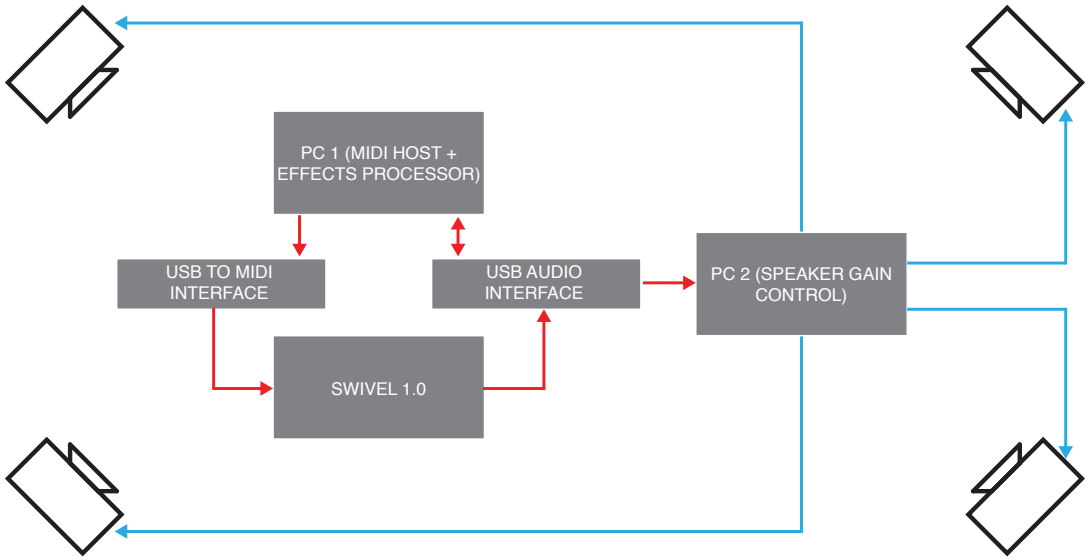


FIGURE 9.2: The signal flow of *Spatial String*.

space: future iterations abandoned the optical pickup in favour of magnetic pickups. Mechatronically, a simpler, more portable robotic chordophone was deemed preferable, resulting in the design and building of Swivel 2 (described in Chapter 3).

9.1.2 *Robotic Consistency*

As the debut performance for Swivel 2, *Robotic Consistinency* served as testbed for the instrument in a performance setting. The piece (whose setup is shown in Figure 9.3) was performed at Wellington City Gallery in Wellington, New Zealand on November 25, 2013. After presenting the motivations for the creation of the performance piece, the piece's technical aspects and compositional elements are detailed. Finally, the outcomes of the performance are discussed.

Robotic Consistency was conceived of as a piece to go toward answering the question of what a performance would be like wherein a human performer attempts to emulate a musical robot's actions. Such a piece would serve to contrast the means by which a human and robot achieve similar musical outcomes. Additionally, it was hoped that the mechanical precision and actuator motions of the robot would highlight the human performer's fluid, silent actions. To address these objectives, a composition containing "call-and-response" elements between robot and human was created.

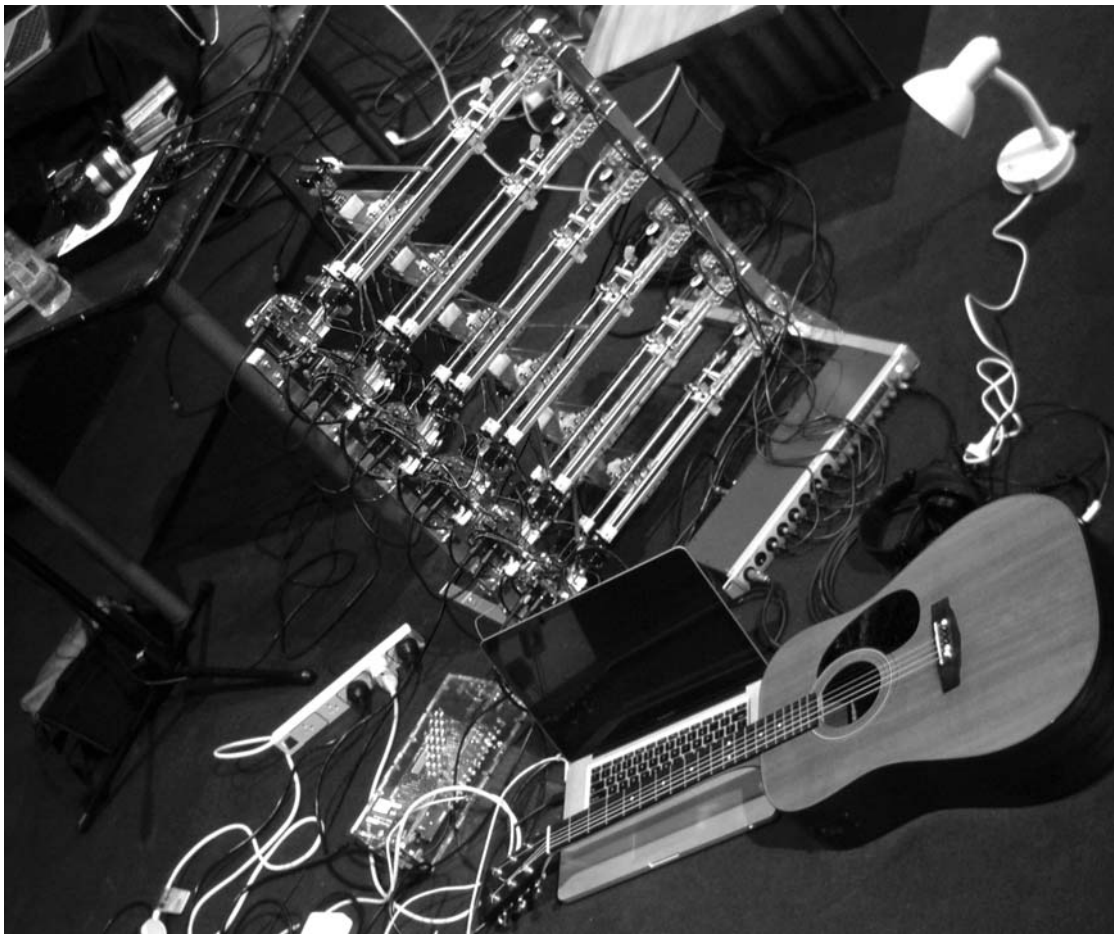


FIGURE 9.3: *Robotic Consistency* on stage at Wellington City Gallery. Visible is Swivel 2, the host computer, and the performer's acoustic guitar.

Other pieces focusing on human/robot performance place the robot in a different position than in *Robotic Consistency*, either placing it in the role of respondant or as an equal "collaborator" with a human performer. In Ajay Kapur's *Digital Sankirna*, for example, the human perfomer serves as the master, creating musical gestures that are elaborated upon by arrays of mechatronic actuators [24]. Composer Jeff Aaron Bryant, in [42], describes similar compositions: sensor-equipped human dancers perform actions that trigger robot actions. Similarly, in [27] and subsequent work, robot builder Gil Weinberg creates collaborative improvisation contexts for humans and robots. In *Robotic Consistency*, by contrast, the human responds to musical gestures created by the robot: as the robot creates difficult-to-play patterns, the human must attempt to emulate them. Artist Tomie Hahn's *Pikapika* served as inspiration to this work: in a 2011 performance of *Pikapika*, Hahn responds in part to a musical robot's actions[1].

[1] www.arts.rpi.edu/ bahnc2/activities/SSpeaPer/pikapika.htm, retrieved October 21, 2014)

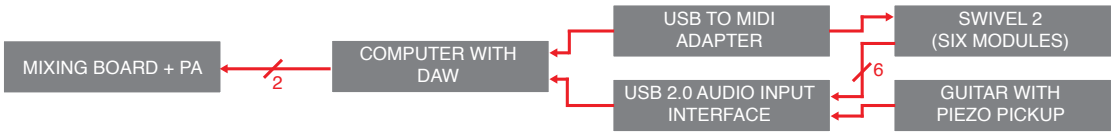


FIGURE 9.4: Signal flow for *Robotic Consistency*.

To direct the human instrumentalist to suitably engage with the robot, textual instructions are provided to performers (in a manner similar to those used by, among others, LaMonte Young in his *Composition #5 1960* [76]). The instructions direct the performer to copy the robot's actions during a segment of the piece:

> *For the first three and a half minutes, sit still next to the robot onstage. Then, with an ordinary guitar and slide, attempt to emulated the actions of the robot's fretter, damper, and picker. Continue this for three minutes; after three minutes of emulation, remain still for the rest of the piece.*

Technically, the piece consists of the Swivel 2 slide guitar robot (presented in Chapter 3) connected to a host PC, as shown in Figure 9.4. A human performer plays a pickup-equipped acoustic guitar. The signals from both Swivel and the acoustic guitar are amplified and, if necessary, processed by the host PC prior to being sent to the performance venue's public address system.

Robotic Consisency contains five sections, of which the human/robot interaction is one: the sections, arranged in ABCDA' form, are illustrated in Figure 9.5. The piece begins with an introduction of the robot: the fretter arms on each of Swivel's modules slide up and down the length of the modules' strings. After a brief melodic interlude that serves as a cue to the human performer to begin playing at its completion, the human/robot interaction section begins. During this section, the robot's fretters move along predefined curves along the strings, pausing after each phrase; during the pauses, the human performer attempts to copy the slide gesture to the best of his or her abilities. After the human/robot interaction section, *Robotic Consistency* features an exhibition of Swivel's expressive parameters: the fretters settle lightly upon the strings, bouncing slightly and producing rhythmic atonal sounds. The piece concludes with final section of slide events up and down the robot's strings: where Section A consisted of serial slides (where each

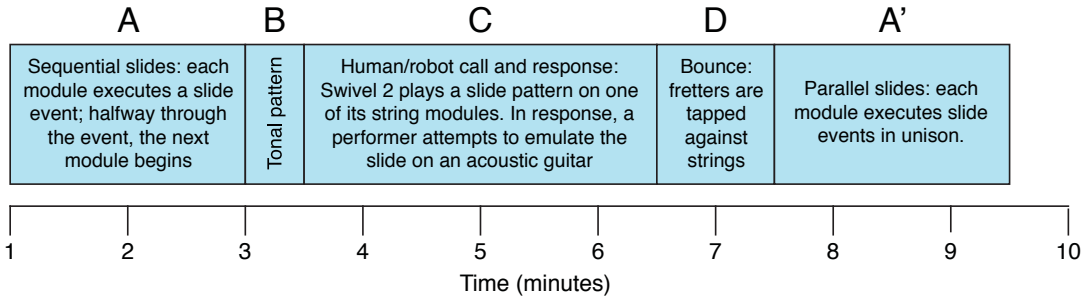


FIGURE 9.5: A timeline of the sections in *Robotic Consistency*.

string's slide event began halfway through the previous module's slide), section A' consists of parallel slides: all fretters slide at the same rate and direction up and down the strings[2].

Robotic Consistency serves as a general exhibition of many of Swivel 2's capabilities. During the performance, the robot was found to be easy to deploy and configure in spite of its large number of actuators. Compositionally, though, the low-level interactions required to program musical gestures was found to be quite time-consuming. To address this, parametric encodings for the musical robots were developed, a number of which are described in Chapter 7. Further, to play chromatically, manual precise positioning of the fretter arms was required: as the arms required pitchbend input, specific 14-bit pitchbend numbers corresponding to each desired pitch had to be found. It is from this tedious task that the idea of a self-tuning scheme for Swivel 2 emerged (a system described in Chapter 6).

9.2 Kritaanjli and Nudge: Sound Installations

9.2.1 *Bhairavi Procedure 1*

Bhairavi Procedure 1 is an installation featuring Kritaanjli, the mechatronically augmented harmonium described in Chapter 4. Created in collaboration with Ajay Kapur to explore the idea of an "endlessly morphing" series of patterns in Raag Bharavi, *Bhairavi Procedure 1* debuted on October 25, 2012 at Wellington, New Zealand's Adam Art Gallery. After a discussion of the motivations for *Bhairavi Procedure 1*, the technical

[2]A video of a segment of the performance can be viewed at https://vimeo.com/87369403 (Retrieved April 25, 2014)

FIGURE 9.6: *Bhairavi Procedure 1* installed at the Adam Art Gallery, Wellington, New Zealand.

TABLE 9.1: Sequence types and actions in *Bhairavi Procedure 1*. Column three lists the subsequent actions accessible by each sequence type: each action is equally likely to occur.

Sequence Type	Sequence Numbers	Action
Beginning sequence	1	Repeat, Next, Random
Middle sequences	2-15	Repeat, Next, Previous, Random, Beginning
End sequence	16	Repeat, Previous, Beginning

and compositional details of the piece are presented. This subsection concludes with a discussion of the results of the installation and the subsequent work motivated by this piece.

As the idea of a long-term composition featuring repeating melodic sequences would have resulted in a prohibitively long traditional performance piece, it was decided to create the piece as an installation. As an installation, gallery visitors could experience the composition for an unspecified amount of time, beginning and ending their experience according to their wishes. Such amorphous starting and stopping points furthered the original concept for *Bhairavi Pattern 1*: that of a ceaseless meditation upon Raag Bhairavi.

Technically, *Bhairavi Procedure 1* is a simple deployment of Kritaanjli: Kritaanjli's microcontroller is plugged into a host laptop's USB port. As shown in Figure 9.6, Kritaanjli is placed on a sculpture plinth with laptop and power supply concealed.

Bhairavi Procedure consists of 16 patterns in *Raag Bhairavi* (an early morning raag with an arohana of *Sa*, *Komal Re*, *Komal Ga*, *Ma*, *Pa*, *Komal Dha*, *Komal Ni* and an *avarohana* of *Sa*, *Komal Ni*, *Komal Dha*, *Pa*, *Ma*, *Komal Ga*, *Komal Re* [77]). The 16 patterns are chosen procedurally, according to rules that vary depending on if a pattern is first, last, or of the middle 14 patterns. The rules, enumerated in Table 9.1, dictate the next pattern chosen based upon the current pattern's type.

As a short-term installation, *Bhairavi Procedure 1* ran for six hours. This extended run allowed for a number of technical observations to be made. During the run, it was noted that the solenoids coming into contact with the instrument's keys created a noise highly audible in the quiet gallery space (as can be noted in the video of the piece[3]). After the installation, this noise was addressed by adjusting the solenoid plunger heights to

[3] viewable at https://vimeo.com/51977345 (retrieved April 25, 2014)

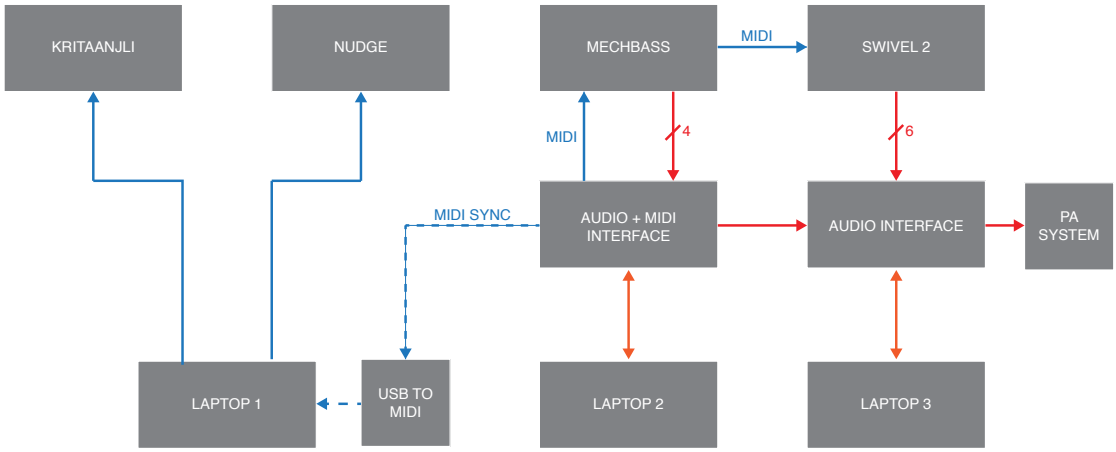


FIGURE 9.7: The signal flow of *Bhairavi Procedure 2*. Its complicated layout and unnecessary redundancy motivated the creation of the Tangle network suite (described in Chapter 7).

remain in contact with the keys even when deactivated, preventing them from travelling and striking the keys from above.

9.2.2 *Bhairavi Procedure 2*

Bhairavi Procedure 2 is an installation featuring Kritaanjli and Nudge with accompaniment by Swivel 2 and MechBass. Conceived of as a demonstration installation for deployment at the 2013 Victoria University Sonic Engineering Expo, *Bhairavi Procedure 2*, like the prior *Bhairavi Procedure 1*, explores the idea of a constantly-morphing composition that allows visitors to arrive, experience a segment of the ever-changing piece, and leave at a time of their choosing. This subsection focuses on the compositional motivations, technical and compositional details, and the subsequent work undertaken as a result of the results of the piece.

The 2013 Victoria University Sonic Engineering Expo was intended in part to serve as an exposition of the works described in this thesis. As such, a single composition featuring all of the instruments was created, with a focus on Kritaanjli and the newly-finished Nudge (presented in Chapter 5).

Technically, *Bhairavi Procedure 2* is a relatively complicated piece necessitating a large number of computers to control the four different musical robots used in the piece. Figure 9.7 illustrates the technically-complicated setup. To allow for the large number of

audio interfaces required to drive and amplify the instruments, three laptops were used, along with three audio/MIDI interfaces. Swivel 2 and MechBass were controlled by a host computer; Nudge and Kritaanjli were connected to a second. A third computer was used to record the output of MechBass and Swivel 2. To allow the two driver computers to play synchronously, one computer was used to provide MIDI sync commands; the second computer, running Ableton Live, was set as a MIDI sync slave.

Like *Bhairavi Procedure 1*, *Bhairavi Procedure 2* consists of 16 patterns in *Raag Bhairavi*. Each pattern is chosen procedurally (based upon the type of pattern picked previously), as described by the rules shown in Table 9.1.

The installation's complicated setup and requirement for a large number of laptops, each running similar software and performing similar tasks, was deemed impractical for further use. While a relatively long setup time was allowed for *Bhairavi Procedure 2*, such a setup would be impractical in other venues, necessitating a more streamlined setup. To address this complexity, a network for mechatronic instruments was developed. This network, dubbed Tangle, is described in detail in Chapter 7.

Additionally, the large number of actuators present on the four musical robots required constant close observation to ensure that they did not remain in unsafe or undesirable states. To address this need for constant human supervision, a number of fail-safety features were integrated into Tangle. These features, also described in Chapter 7, allow the robots to be installed in situations where such supervision is impractical.

9.3 Networked Ensemble Use

9.3.1 *Tangled Expressions*

The complicated setup for *Bhairavi Procedure 2* motivated the development of Tangle, allowing users of the mechatronic instruments to easily access all networked instruments without extensive setup for each connection. After the development of Tangle, a new composition for the networked robots was created. The composition combines the work presented throughout this document, utilising the physical instruments built in Part 1 and the interfacing systems developed in Part 2.

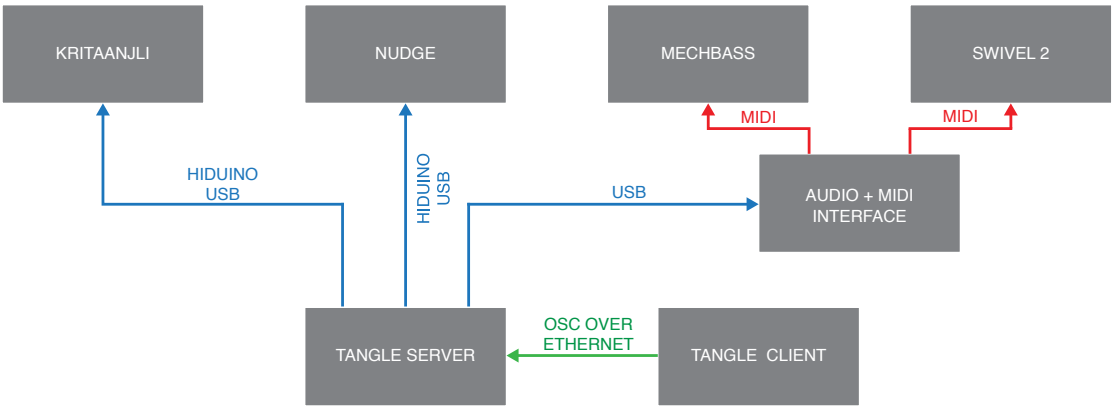


FIGURE 9.8: The signal flow of *Tangled Expressions*. A client computer connects to a server; the client needs only to plug in an ethernet cable to communicate with all the robots.

A compositional goal for the piece, named *Tangled Expressions*[4], was to employ as many of the parameters of the instruments as possible: doing so allows the interactions between different parameters to be explored and the Tangle network to be tested. *Tangled Expressions* exists as a video of a performance executed May 5th, 2014 at Victoria University of Wellington.

Technically, *Tangled Expressions* is a typical deployment of a Tangle client/server network. Figure 9.8 illustrates the configuration: a client machine is plugged via a single ethernet cable into the Tangle server. Attached to the Tangle server are the mechatronic instruments: Nudge and Kritaanjli communicate via USB MIDI HID; MechBass and Swivel 2 are attached to a MIDI hardware interface.

Tangled Expressions is divided into six subsections (shown in Figure 9.9). The piece begins with an exhibition of Nudge's ability to interact in a relatively delicate manner with a frame drum: using Nudge's adjustable solenoid height stop, the drumstick is positioned 3 mm above the drum head and is instructed to strike it at smoothly-changing rates. Following the soft drumstick playing events of the piece's introduction, the remainder of *Tangled Expressions* consists of a series of passages featuring simultaneous playing by all of the instruments: first, a section focusing on interactions between Kritaanjli and Nudge begins. In this passage, Kritaanjli is instructed to rapidly depress its keys while pumping its bellows, creating pulsating tones that accompany Nudge's rhythm.

[4]A video of *Tangled Expressions* can be viewed at https://vimeo.com/95576294 (retrieved May 18, 2014)

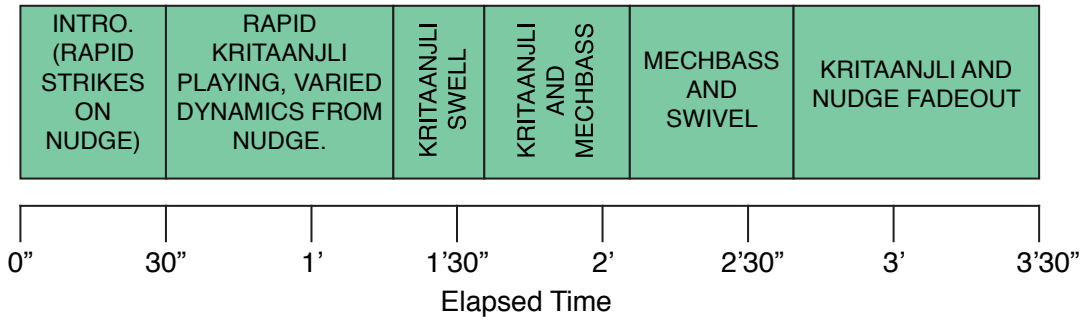


FIGURE 9.9: A timeline of the sections in *Tangled Expressions*.

After the passage featuring Kritaanjli and Nudge, MechBass begins playing rapid note pick events, mirroring Kritaanjli's abrupt keyboard stiking sounds. Finally, MechBass and Swivel play together, with Swivel's modules conducting portamentos and vibrato effects, alternating between tonal and atonal sounds. *Tangled Expressions* concludes with a reiteration of the second section: Kritaanjli and Nudge play together, their rhythms becoming sparser until ceasing entirely.

Throughout its sections, *Tangled Expressions* uses the instruments' parameters in ways not conceived of at the time of the instruments' construction: Kritaanjli's rapidly pulsating key-pressing actions, MechBass's contribution to the piece's rhythm through its actuation noises, Nudge's role as a "delicate" sound-producer, and Swivel's dual role as a producer of tonal and atonal sounds indicate that the instruments, when networked together to allow for simplified addressing of multiple instruments from a single DAW, are capable of expressive extended technique use.

9.4 Summary: Performance and Installation Use

This chapter presents a narrative, beginning with pieces (like *Spatial String*) that used mechatronic instruments still in their prototype stages and progressing through to ensemble compositions featuring an array of networked mechatronic instruments. By following the works from earliest to most recent, a cycle is evident wherein new robots motivate new works, which in turn motivate changes and improvements to the robots.

Of things learnt in the creation and execution of the works described in this chapter, three stand out as key lessons that centrally informed subsequent works. Firstly, after *Spatial*

String, it became clear that the bulky Swivel 1.0 (equipped with complicated power requirements, actuators, and pickup systems) needed to be replaced with a simpler, more portable system. This decision resulted not only in the development of Swivel 2 but also in the decision to create a lightweight, easy-to-build mechatronic drumming system in favour of a heavy system with potentially more degrees of freedom. The final appearance of Swivel 2 and Nudge, then, can be directly traced to the findings of *Spatial String*.

Secondly, after the complicated deployment of Swivel 2, Nudge, Kritaanjli, and Mech-Bass in *Bhairavi Procedure 2*, it became clear that a more streamlined means by which the robots could be used in an installation or performance setup was needed. This finding resulted in the decision to design and implement the Tangle suite of musical robotic networking tools.

Finally, after composing and performing *Robotic Consistency*, it was found that the abundance of actuator-specific commands required to instigate even a single note playing event on Swivel 2 resulted in prohibitively long composition times. As a result of this, to allow musical ideas to more rapidly be explored on Swivel 2, it was decided that the Tangle network should be equipped with the ability to output sequences of actuator instructions in response to a single client input event.

These three main lessons, learnt in the course of the compositions presented in this chapter, so directly informed the work presented throughout this thesis document that it is evident that musical robotic systems need to be tested in performance and installation settings in order for their functionality to be properly understood. As crucial as laboratory-based characterisation is, such performance and installation use allows the lab-derived characterisations to be pitted against the non-ideal conditions encountered in real-world performance and installation contexts. For roboticists seeking to create instruments for performance and installation use, it is imperative that every opportunity be taken to test their in-development systems outside of the laboratory.

The works presented in this chapter serve as the first existent collected body of new compositions and installations for musical robots designed with high degrees-of-freedom as a goal. It is hoped that these new works and the aforementioned lessons gained from them during their execution will allow future artists and composers to work with

such instruments endowed with knowledge of the challenges and artistic affordances that accompany them.

Chapter 10

Summary & Conclusions

During the course of the work presented in this document, four new musical robots were built. These robots are each endowed with actuators placed in such a manner as to afford musical expressivity: MechBass and Swivel allow for portamento or discrete pitch-shifts; MechBass can vary the intensity of its note-picking events; Kritaanjli is capable of playing large numbers of notes at a time with adjustable loudness; Nudge can be set to hit many positions on one or more drums with a drumstick able to adjust its height above the drum head. The robots and their behaviours are described in Part 1 of this thesis.

These parametrically-rich musical robots allow composers to explore more musical ideas than is typical with many existent systems. However, when faced with an ensemble of such actuator-studded mechatrons, a challenge appears: to individually address the 92 actuators present in Nudge, MechBass, Kritaanjli, and Swivel is somewhat akin to writing software in assembly language: every event must be precisely and accurately specified by the user. What is needed is a paradigm for encoding these low-level actions in a musically-sensible manner. Part 2 of this thesis describes a number of methods for doing so: self-tuning for mechatronic chordophones (tested on Swivel 2) and a software suite that enables customised, potentially complicated output in response to simplified user input.

While both Part 1 and Part 2 of this thesis can be seen as freestanding, taken together they go toward realising what was a major goal in this project: to build expressive musical robots usable in live performance and installation contexts. While many previous

works achieved one or the other, the work presented herein endeavours to achieve both: expressive musical robots belong in galleries, in museums, on stages, and in other live contexts, and it is a goal of this work to help to catalyse such a move out of the laboratory.

During the course of this research, a number of conclusions were reached about the technical act of building musical robots, as well as the artisanal act of building musical robotic instruments. The next three sections detail summarise this document's contributions and detail these conclusions, and are followed by a final section containing a number of immediate and long-term ideas for subsequent work.

10.1 Summary of Contributions

The goal of this work has been to create performance-oriented and installation-oriented musical robotic systems which employ new techniques to better function as expressive musical instruments. Toward this goal, the contributions that this work makes to the field of musical robotics fall into two main categories: work focusing on designing and building expressive musical robotic systems, and work focusing on enabling users to interface with these complicated parametrically-rich musical robots. The following list includes design and construction-related contributions. Collectively, they represent a body of new research into the engineering of mechatronic instruments.

- Two new mechatronic chordophones have been presented. These instruments feature a number of subsystems novel to musical robotics, including new fretting systems, one based on a rotary motion-based fretting and the other based upon linear motion with string clamping. Also presented are novel approaches for variable pick intensity on robotic guitar systems.
- A polyphonic robotic harmonium with a novel variable pumping intensity system has been presented. The mechatronic design of this instrument is detailed, and its performance evaluated.
- The design and construction of a novel closed-loop robotic drum system has been detailed. This system can rotate to reposition its drum beater to various striking positions above one or more drums. Additionally, the beater's proximity to the

drum head may be changed in an online manner. Both the drumstick height-adjustment mechanism and the rotary-motion drumstick positioner are novel.

In addition to design and construction contributions, this work contains new research in networking and interfacing with expressive musical robots, as well as calibration and fail-safety systems. The following list highlights these contributions.

- A novel automated tuning system for robotic guitars has been presented. This system is based upon a pre-performance calibration routine wherein a string's response to varying fretter positions is recorded. The string is then tuned at runtime by interpolating its current detuned value between previously-populated characterisation curves. This novel approach allows for user-defined intonation schemes to be played.
- A network for complicated musical robotics has been introduced. This network features numerous novel elements, including:
  - The ability to customise the server's output to a robot in response to input events from a client. These parametric encodings are detailed in Chapter 7. This novel parametric encoding system allows users to create custom mappings from input interfaces.
  - The integration of a number of fail-safe features, novel to musical robotics, intended to prevent the musical robots from entering into or remaining in unsafe states.
  - A latency calibration routine, novel to mechatronic percussion systems, allowing multiple timers with different latencies to synchronise their actuation times.
- Also presented is the first user study in the discipline of musical mechatronics wherein a number of musicians were asked to use musical robots and to express their experiences with the use of parametrically-complicated networked mechatronic instruments.

10.2 Building Musical Robotic Systems

The musical robots built in this thesis are diverse instruments, featuring an array of actuator types intended to expressively stike, pick, and pump instruments of three different families. In spite of their diversity, a number of common technical themes appeared during their construction. These findings are presented in the following subsections.

10.2.1 Communication

The microcontrollers of Nudge, Kritaanjli, Swivel 2, and MechBass are similar in their high-level functionality, receiving incoming MIDI messages, parsing the messages, and changing the robots' actuators' states accordingly. Early versions of the firmwares of these robots were configured to receive simple messages and, from one message, control a number of the robot's actuators. The microcontroller on Swivel 2, for example, was at first configured to respond to a MIDI NoteOn command by damping the string, moving the fretter, undamping the string, picking the string, and re-damping the string. Such built-in encoding proved troublesome when attempting to allow the robot to respond to a wide variety of parametric encodings such as those afforded by Tangle. Similar challenges were encountered with MechBass, Nudge, and Kritaanjli.

Confronting the problem of over-specific microcontroller input conditions, a useful principle of musical robotic design emerged: musical robots' firmware should be as open-ended as possible. By allowing individual control over the actuators from a host machine, the work presented in the first part of this thesis may be most seamlessly integrated with that presented in Part 2: the subassemblies of the relatively actuator-dense robots of Part 1 may be affected individually and in an online manner by software applications such as those described in Part 2 of this document.

Figure 10.1 illustrates the advantage of individual parametric control over a musical robot's actuators. In the figure's top diagram, a single MIDI message is mapped to control the five robot actuators. In this configuration, modifying the timing and sequence in which the actuators are triggered would require a firmware update or other unusual measures. Conversely, with a separate command expected for each actuator (as shown in the lower diagram in Figure 10.1), the server's on-the-fly reconfigurable software can affect the actuator's behaviour.

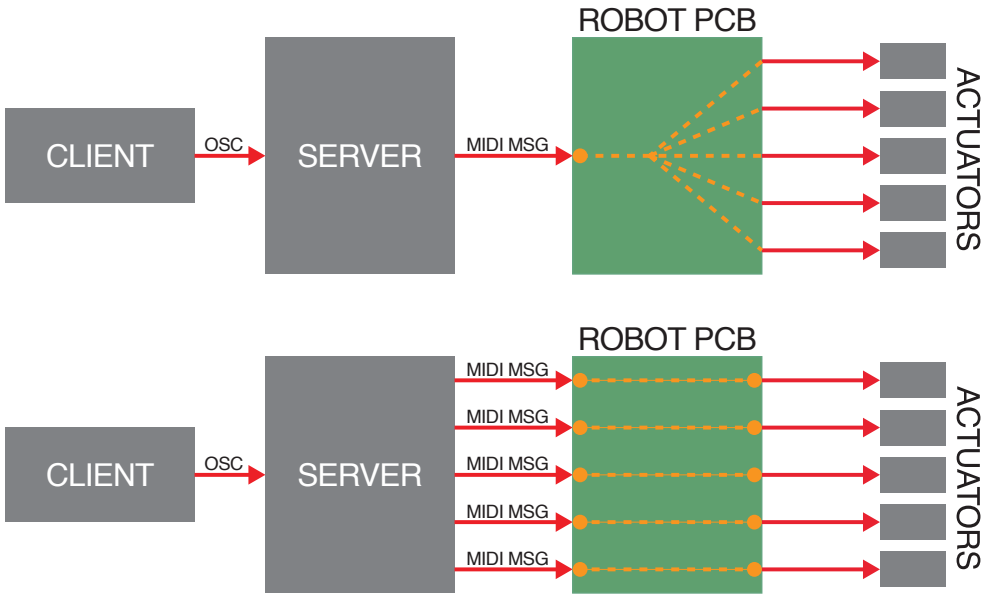


FIGURE 10.1: Two schemes for communicating with musical robots. The top scheme shows a relatively inflexible method; the bottom scheme, used for the robots described in this document, allows for the rapidly-reconfigurable software on the server to affect the robot's individual actuators.

10.2.2 Iterative Design for Musical Mechatronics

In following an iterative design process on MechBass, Swivel 2, Nudge, and Kritaanjli, two additional musical robotic design principles emerged. As the robots are a product of repeated iterations, they benefited greatly from being produced using a Computer Aided Design and Computer Aided Manufacturing (CAD/CAM) workflow. While the CAD/CAM workflow is ubiquitous in many design and engineering applications, only recently have musical robotic systems begun to employ it.

In spite of the robots' apparent diversity, the CAD/CAM approach used on them is quite similar from system to system. The workflow begins with a conceptual drawing which is ported to a solid modelling CAD program (such as SolidWorks[1]). The robot's frame is constructed as an assembly model, and the subsequent parts are modelled in context of the chassis.

[1] www.solidworks.com (retrieved February 20, 2014)

10.2.3 Rapid Prototyping for Musical Mechatronics

Two primary CAM approaches are used on the robots presented herein. A laser cutter is used for parts free of "overhangs:" such parts may be manufactured with a two degrees-of-freedom machine tool. Additive manufacturing techniques are used for parts involving variations on their Z axes. Open source desktop 3D printers are used extensively on the structures of all of the robots. These low-cost 3D printers are able to produce components of suitable integrity to be used as structural elements and end effectors.

When integrated into a computer-aided design process, the use of such CAM approaches allows for the rapid interation of prototypes: pickwheels, drumstick holders, and motor mounts can be tested and improved quickly.

10.2.4 Actuator Selection

A second factor allowing for the rapid prototyping of musical robotics is the use of simple actuators in early designs. While more complex actuator assemblies allow for quieter, faster, and more accurate and precise positioning, such assemblies can be expensive and time-consuming to produce. For conceptual and prototypical musical robots, early versions should be built with the actuators chosen as a compromise between ease of implementation and overall functionality.

Miniature RC-style servos are used extensively on Nudge, MechBass, and Swivel 2. While not as accurate or quiet as more expensive actuator solutions, such servos may be attached to chassis easily and require very simple driver electronics. The use of such simple actuators allows for the rapid realisation of conceptual goals: new actuator placement ideas may be tested and evaluated without the need for complicated support electronics or software. Once the design is settled upon, future iterations can be modified to implement superior actuators.

By treating the design of musical robotics as an iterative process, a methodology inspired by the engineering and design disciplines, expressive parameters can be quickly tested, evaluated, and even used in composition: the use of rapid prototyping equipment and CAD tools allows for the newly-tested designs to be improved upon and changed with minimal turnaround time.

10.3 Applying Expressivity to Musical Robotics

While the act of building musical robots with the intention of enabling their musically expressive output is a device-specific activity, the construction of the numerous musical robots in this thesis, along with their accompanying software followed seven high-level steps. These steps are outlined below, and may serve as a starting point for those seeking to create expressive musical robots.

1. Identify the instrument family.
2. Evaluate the instrument's parameters. A review of the instrument's repertoire (particularly extended technique compositions featuring the instrument) may assist in the act of identifying its expressive parameters.
3. Discern means by which the instrument's previously identified expressive parameters can be actuated by electromechanical means.
4. Determine the placement of actuators.
5. Design a chassis to afford placement of actuators.
6. Design the actuator electronics.
7. Create software allowing compositions to interface with the actuators.
8. Create performance or installation works featuring the instrument, returning to previous steps as problems are identified.

Based upon the above steps, the act of building an expressive musical robot is an interdisciplinary affair, requiring some degree of musical knowledge in the first steps and an amount of mechatronic engineering acumen in the last.

Perhaps the most challenging steps in the process are the second and third: identifying the robot's expressive parameters and then creating mechatronic actuation assemblies to exploit them. An awareness of prior mechanical and mechatronic musical instruments can be of great assistance: the literature contains detailed overviews of parametrically-rich pre-20th Century instruments (many of which are outlined in [31]) which, based upon a lack of citations in academic publications, appear to have been largely forgotten

by many modern musical roboticists. A review of these instruments (such as the one briefly conducted in the second chapter of this thesis) can make apparent a large number of mechanically-ingenious actuation techniques no less useful today than they were when developed. The works of 20th Century musical roboticists can also be studied: Trimpin and Godfried-Willem Raes, for example, have spent decades building and composing for musical mechatronics, and documentation of their work abounds. Finally, research-oriented robots (such as those of Waseda University [38]) can be studied by those seeking to create musical robots for performance and installation use.

As shown by the first steps in the above list, the early decisions made concerning actuator placement and configuration will go far toward dictating the expressive output of the robotic instrument. Viewed in this light, the act of designing and building musical robots is the first stage in the act of composing for them.

In "The 'E' in NIME: Musical Expression with New Computer Interfaces," authors Christopher Dobrian and Daniel Koppelman argue that "it becomes apparent that sophisticated musical expression requires not only a good control interface but also virtuosic mastery of the instrument it controls" [5]. They further argue that musical expression is more than simply access to a large number of parameters. To enhance the expression of modern computer music performance, they encourage, among other things, "...choosing intuitive but complex gesture-sound mappings..." These arguments are quite relevant to the robots built in this thesis: while the mechatronic devices themselves allow for relatively intricate parametric control, they desperately need such "complex gesture-sound mappings" in order to become expressive instruments in the hands of musicians. The interface between the dozens of actuators, each capable of fine-grained control, and the musician hoping to compose for musical robots is as important as the physical instrument. The user study presented in Chapter 8 proved useful in gaining insight into musicians' uses of the instruments.

10.4 Future Work

The work shown throughout this thesis presents numerous opportunities for follow-up research. Future technical work emerging as a result of this thesis will align into three avenues: mechatronic improvements, control systems upgrades, and enhancements to

the means by which users may interact with the systems. These technical improvements and modifications will be discussed below, followed by an overview of future musical applications.

Based upon the experience of building and using the musical robotic systems built in this thesis, a need has emerged for further revisions to be made to each of them. All of the robotic systems will be adapted to use OSC: while MIDI is suitable for most of their actuators' resolutions, OSC's greater extensibility and ability to seamlessly communicate with wide-area networks and increasingly-popular mobile devices paired with the emergence of low-cost UDP-enabled microcontrollers provides ample motivation to upgrade each of the robots.

A second upgrade cycle to be made to all of the robots will focus upon a general reduction of acoustic actuation noise: actuator noise on MechBass and Kritaanjli often remain audible above the instruments' intended output. The miniature servomotors used on Swivel 2 and Nudge can be quite audible in quiet installation and performance contexts. To reduce this actuation noise on some of the subassemblies, larger DC motors will be employed in a direct-drive configuration. The use of direct drive motors will bypass the acoustically-loud gearboxes present in the sevomotors on Swivel and Nudge. Additional damping and acoustic isolation will be added to Kritaanjli's solenoid actuators, and the four pickwheels on MechBass will be augmented with rubber or foam dampening to reduce picking noise.

Swivel 2 will be succeeded by Swivel 3, equipped with an enhanced string-clamping mechanism designed to pinch the string between two roller-bearings rather than against a single fretter arm. The fretter and clamper servos will be supported with additional ball-bearings to allow for more aggressive clamping, and the fretter servo will be equipped with a secondary gear reduction apparatus, greatly increasing the fretter's resolution.

Kritaanjli will feature upgraded electronics, allowing for individual PWM control over each solenoid's power. While deemed acceptable, a speedier bellows pumping mechanism may be implemented to allow for a greater dynamic range. Finally, the harmonium's stops will be automated, permitting on-the-fly timbral adjustments.

Nudge will be made more compact with a workspace of 360 degrees. Its PID firmware will be modified in a manner similar to Eric Singer's GuitarBot [43], allowing for online

tuning adjustments.

The aforementioned opportunities for future iteration and improvement focus upon the work presented in this thesis. During the course of this research, though, the importance of the ideas pursued in Part 2 of this document became increasingly evident. As mechatronic instruments grow more expressive, the need for more effective gesture-sound mappings becomes more obvious. Chapter 8's user study highlighted this, exposing composers' desire for new ways to explore the instruments. With significant progress made in such mappings, musical robots will be one step closer to being fully-realized expressive instruments, able to reapeatedly perform actions while allowing for extensive means by which the systems can interface with the outside world.

10.5 Conclusion

A review of the history of automatic music is one of a quest toward two seemingly-disparate ideals: musical expressivity and precise and accurate consistency. While much of the long history of automated music appears to be on a focus toward repeatability, mechatronics advances in recent decades have opened the doors toward improved expressivity. The work presented in this thesis and by numerous fellow musical roboticists aims to take advantage of methodology derived from engineering disciplines, enhancing automated music by offering the possibility of the realisation of expressive and repeatable systems.

The work presented in this thesis has contributed to the overall body of knowledge about musical robots from both a technical and an applications perspective. Technically, this work has presented a number of original contributions (as mentioned previously in Section 10.1) that will form the basis for a number of subsequent innovations in mechatronic music systems. This is more than merely a technical thesis, though. Prior to Chapter 8's user study, no user evaluations of composers' experiences with complicated networked musical robots had been undertaken. As a result of the user study, future roboticists may base their actions in part upon the user study's findings. Additionally, prior to Chapter 8, no body of work had been presented consisting only of performances and installations using high degree-of-freedom musical robotic systems. The lessons and

outcomes presented in Chapter 8, then, provide a novel examination of the challenges and capabilities of such performance and installation scenarios.

This thesis work began with a question: How might the addition of many actuators enhance musical robotic expressivity? After pursuing answers to that question in the construction and evaluation of a number of musical robots, a second question emerged: How might composers work with increasingly-complicated musical robotic systems in an expressive manner? This thesis work has allowed for the investigation of both questions: complicated, parametrically-rich mechatronic music systems have been built, and a number of means by which composers may interface with these complex systems have been developed.

A third and final question has emerged, made possible by the work presented in this thesis: Now that an ensemble of potentially expressive musical robotics has been built and configured to be interfaced with by composers, what new and unexpected musical outcomes might occur? This final question, the most mysterious and exciting of the three, can only be answered through much experimentation and music-making.

Appendix A

Supplemental CD Contents

The supplemental CD included with this document contains code, circuit diagrams, and engineering drawings related to the projects presented in this thesis. This appendix lists and briefly describes the CD's contents.

`murphy_disseration_cd/Kritaanjli`

- `kritaanjli_solenoid_drawing.dxf` *A drawing of Kritaanjli's keyboard-playing solenoids.*
- `niobe_board.jpg` *An illustration of the Niobe PCB assembly.*
- `persephone_test.zip` *An Arduino sketch used to evaluate Kritaanjli's Persephone PCB assembly.*
- `persephone.jpg` *An illustration of the Persephone PCB assembly.*

`murphy_disseration_cd/MechBass`

- `schematic.pdf` *A schematic of the JM2 board.*

`murphy_disseration_cd/Nudge`

- `nudge_code.ino` *Nudge Arduino code.*
- `nudge_pcb.zip` *Nudge's PCB (Altium Designer 14.2 file).*

`murphy_disseration_cd/Self Tuning`

- `string_characteriser.zip` *ChucK code used to characterise Swivel 2's strings.*
- `swivel_autotune_master.zip` *JUCE application and XCode project of SwivelAutoTune.*

`murphy_disseration_cd/Swivel 2`

- `swivel_electronics.zip` *Swivel 2's PCB (Altium Designer 14.2 file). Contains both the implemented PCB layout and a new (untested) more compact revision of the PCB layout.*
- `swiv_rev2_code.ino` *Swivel 2 Arduino code.*

`murphy_disseration_cd/Tangle`

- `Tangle.zip` *Tangle client and server code.*

`murphy_disseration_cd/Video.zip`

- *This .zip contains a video of a performance of 'Tangled Expressions' and a text document containing a list of links to online videos related to this thesis document.*

Appendix B

MIDI and Musical Robotics

Musical Instrument Digital Interface (MIDI) is a simple-to-implement communications protocol used in many hardware and software synthesizers. While protocols such as Open Sound Control (OSC) offer the potential for all-around improved performance, the universality and ease of implementation of MIDI, coupled with its ability to achieve 13-bit precision with pitchbend commands, has resulted in it remaining a popular interface in many professional and consumer-grade instruments and effects. Rather than provide a general overview of MIDI, this section discusses the ways that MIDI is used by the robotic instruments presented in this thesis document[1].

B.1 MIDI Message Commands and Parameters

NoteOn, NoteOff, Continuous Controller (CC), and Pitchbend commands are used to initiate actions on the muscial robots presented in this thesis. Table B.1 lists these commands and their ranges. Each command has a range of 16, allowing for 16 separate MIDI channels to use that command.

Each MIDI command type has one or two parameters. While these parameters are typically used for musical quantities such as pitch and velocity, they may be used as instructions to affect any musical robotic parameter. All message types except for pitch-bend transmit two 7 bit parameters; pitchbend transmits a single 13 bit parameter as separated least- and most-significant bytes.

[1] For a more general overview of MIDI, see www.midi.org (retrieved February 13, 2014)

Table B.1: MIDI command types

Command Type	Hexadecimal	Decimal
NoteOff	80-89	128-143
NoteOn	90-99	144-159
Cont. Controller (CC)	B0-B9	176-195
Pitchbend	E0-E9	224-239

MIDI pitchbend can be used to bypass the low resolution of most MIDI parameters: for the majority of the actuators used on the robots in this thesis, the 13 bit message is actually of too great a resolution; the pitchbend command is typically mapped to a reduced resolution.

MIDI can be transmitted over traditional DIN5 cables connected to a hardware MIDI interface (such as the Motu MIDI Express 128 used for Swivel 2 and MechBass). A PC connected to such a hardware interface is presented with a number of MIDI ports from which MIDI commands may be transmitted. The port is selected in the composition environment, and any subsequent MIDI messages are sent to the port of choice. This traditional MIDI DIN-5 scheme is useful in systems requiring a bus-type topology: devices may be chained from one MIDI output port to the next device's input port. Additionally, "MIDI thru" ports allow for low-latency retransmission of messages from device to device (this technique is used on Swivel 2).

Nudge and Kritaanjli (as well as many of the authors' other mechatronic instruments) use the USB MIDI HID protocol, wherein MIDI commands are transmitted as human interface device (HID) messages to compatible PCs. On properly-equipped PCs, this approach requires no additional hardware or drivers. HIDUINO (developed by Dimitri Diakopoulos) allows for USB MIDI HID to be implemented on the popular Arduino series of microcontroller prototyping devices.

Tangle and SwivelSelfTune require some degree of MIDI communication within a host PC. To accomplish this, Mac OS X can implement Inter-Application Communication (IAC) buses. These "virtual" MIDI ports can be set to send and receive MIDI from one application to another. IAC buses are particularly useful as a means by which MIDI messages can be sent to an application which converts them to UDP OSC messages for transmission over a network (as in the case of Tangle)[2].

[2]Third-party applications MIDI-Ox and MIDI-Yoke perform similar roles for Windows-based systems. www.midiox.com (retrieved February 17, 2014).

The Arduino MIDI library[3] allows for MIDI messages to be transmitted or received by a UART-equipped Arduino. All of the robots used in this thesis implement this library, which allows for rapid deployment of MIDI compatible devices with minimal low-level communications code needed.

Four steps are needed to receive MIDI messages using the Arduino MIDI library.

1. Include the library and initialise MIDI, specifying the device's channel.
2. Set up any callbacks. A separate callback is needed for each expected MIDI command type.
3. Repeatedly check the microcontroller's serial buffer for new MIDI messages.
4. Use the callbacks to execute command-specific code.

MIDI can be deployed on low-cost microcontrollers, resulting in its use on a wide variety of mechatronic instruments. As the research in this thesis progressed, more hig-resolution applications were identified, resulting in increasing use of the 13 bit pitchbend command. As UDP-capable microcontrollers become more popular, it seems likely that OSC will become a preferred form of communication between composers' computers and musical robots. Due to the large number of legacy MIDI devices, though (and because of MIDI's often-adequate resolution), is is likely that musical robotic ensembles will face a need to support MIDI instruments for some time.

[3] www.playground.arduino.cc/main/MIDIlibrary (retrieved February 13, 2014)

Appendix C

Using New Musical Robots: Survey Questions and Answers

This appendix lists user study questions and answers. The user study, detailed in Chapter 8, consists of 19 questions; the questions are subdivided into nine steps. These steps are listed with the questions and answers provided by the eight participants. The Victoria University of Wellington Ethics Approval form is also included, and is shown in Figure C.1.

Step 1: Please complete Questions 1 through 6 before using Tangle. Question 1: Are you familiar with command line interfaces (for example, "Terminal" on Mac OS X)?

All eight participants indicated that were familiar with command line interfaces.

Question 2: Have you ever used the ChucK programming language?

All eight participants indicated that they had used ChucK.

Question 3: Have you ever used MIDI IAC buses?

Seven of eight participants indicated that they had used MIDI IAC buses.

Question 4: Have you ever used OSC (Open Sound Control)?

All eight participants indicated that they had used OSC.

Question 5: Have you ever used Ableton Live?

Six of eight participants indicated that they had used Ableton Live.

Question 6: Have you ever used musical robots?

Four of eight participants indicated that they had used musical robots.

Step 2: Connect to the Tangle Server. Open up a terminal window.

Step 3: Consult the Tangle Client User Guide[1] and follow its direction to connect to the server. After doing this, please answer Question 7.

Question 7: From 1 (Very difficult) to 5 (Very easy), how difficult did you find connecting to the server?

Four of eight participants rated the connection process a "5." Two rated it a "4." One each rated it "2" and "1."

Step 4: After the client connects to the server, the server will print out diagnostic information. Please navigate to this information in the terminal and answer the following questions:

Question 8: Is the channel information provided by the server easy to find?

If "no," what could improve it?

All eight participants indicated that the channel information was easy to find. Two users added that a GUI would improve the process.

Question 9: What (if anything) would you change about the server connection process?

- *"I would create a GUI with place for IP config, and sort the verbose server feedback by category (i.e., channel info)"*
- *"The string of commands to connect to the server is pretty long (but understandably necessary!) and could be prone to error (although I luckily had none). Not sure how this could be fixed, unless there's a pre-written script that takes in arguments and executes that command? Not a big issue, being picky."*
- *"I would implement it as a shell script or a mac OSX application."*
- *"A simple GUI could aid users less-than-familiar with command line code."*
- *"A GUI"*
- *"Double-clicking a shortcut?"*

[1] Available on www.github.com/PFCM/Tangle (retrieved May 1, 2014)

Step 5: Open Ableton Live. Create a new track that outputs MIDI to the Tangle Client's MIDI input port. This will be accomplished with an IAC bus. Tip: Find the correct MIDI channel by consulting the information printed in the client in Terminal. Instruct Nudge to strike the drum. The necessary MIDI messages to send from Ableton are shown in the Tangle Clients output in the terminal.

Question 10: Were you able to make the robot play a note without any assistance? If not, what assistance did you need?

- *"Some assistance was needed to understand what the MIDI numbers meant and what they responded to (IE pitch bend, etc.)"*
- *"No. I didn't know how to add a note in Ableton. Everything else was fairly straightforward."*
- *"I needed to remember to double click in Ableton to make a new track and also how to create a NoteOn/NoteOff... It's been a long while (and a few versions) since I last used Ableton."*

The remaining five participants were able to connect without assistance.

Step 7: Render a simple pattern on the robots (e.g., Happy Birthday) and answer the following questions:

Question 11: Based upon your experiences with the robots, what are some other musical or usability-enhancing parameters that you would like to see implemented on them? Please provide short answers below pertaining to the robots you used.

MechBass (To save time, you can use a single string rather than all four):

- *"I found MechBass to be a little more confusing due to having to pre-position the fretter, but found it fun to use, although not incredibly expressive compared to other instruments."*
- *"Possibly running off a single channel, so you can write more easily for them. Creating four different tracks in Ableton splits how you write for them (although, this may be a good thing)."*
- *"The MechBass was very responsive and in many ways resembled a live performer."*
- *"The instrument is very responsive and appears to stay in tune even after running for more than 10 minutes. The only issue that comes to mind is the actuation noise which is quite comparable to the amplified bass sounds, and this might be a problem in certain musical contexts."*
- *"String flip in addressing. Musical parameters are sufficient. Good at high-speed picking but tends to desync with Ableton clip."*

Nudge:

- *"Some information about appropriate ranges to send would be helpful, e.g., what positions will actually make contact with the drum."*
- *"I can't think of anything more you can do with this system."*
- *"Nudge was very responsive to user input. It could also be interesting to see if it could incorporate some of its own 'feel.' "*
- *"It would have been more intuitive if CC7 values resulted in lower drumstick positions. Considering the dimensions of the drum, the pitchbend range (angular position of the stick) can be limited to make sure the stick doesn't go off the drum surface. Determining the right drumstick values for certain notes requires some fine-tuning work and is a bit tedious. But overall, having control on both angular position and height of the stick is a huge advantage."*

Kritaanjli:

- *"A bit more info about interacting with the bellows would be helpful for those who have never composed for the instrument before."*
- *"Pumping mechanism: I wasn't so sure about how this operated. Some more instruction or a recommended CC7 pump pattern could be nice."*
- *"Of the three, I found this to be the easiest to use and probably my favourite. I feel like it had the least expressivity but the sound produced was my favourite. I think being able to change the bellows speed would improve the expressivity."*
- *"The bellows being able to be controlled in a repeatable way so that you can get consistency in volume swells and dynamics."*
- *"Kritaanjli could possible incorporate some intuitive interpretation of input. I'm just imagining how different performers may use different bowing for the same piece of music on violin or other bowed instruments."*
- *"Very responsive and capable of producing a full dynamic range of different rhythms. Easy to work with. I used a MIDI arpeggiator to control the instrument and it worked perfectly."*
- *"This might be a comment that is a critique of Ableton - not your system - it'd be fun/easy to work with or use a virtual keyboard synthesizer (e.g. Moog) that would combine all of the control parameters for each of the devices. Knobs, pressure-sensitive touch pads and such could help the user (me!) create compositions. Just a thought-friendlier UI."*
- *"It'll be cool to have a mode whereby the note-on trigger pumps automatically with presets of pump action envelopes."*

Step 8: The user survey coordinator will help you play some notes on Swivel 2. After doing so, answer the following questions:

Question 12: Would Swivel 2 have been easier to use if it was more like MechBass (condensing all of the note change, clamp, damp, and pick commands into a single command)?

- *"I like the level of control in Swivel. But I also like the ease of MechBass. Implement hybrid would be nice."*
- *"Possibly, but I see the need for different actions needing different 'buttons' (CC)."*
- *"Maybe... It may just be the way that Ableton is set out, because you can only see one parameter at once. However, there's more degrees of freedom with Swivel, meaning more ways to control."*
- *"Yes."*
- *"I believe it definitely would have been easier to work with but maybe not as expressive."*
- *"I think Swivel's advantage is the depth of control it allows users to define."*

Question 13: Would Swivel 2 be as "expressive" if it were more like MechBass (with all commands condensed into a single one)?

- *"I think it would be easier to get going quickly but less expressive."*
- *"No, it has a very 'Chinese' instrument feel! It would be nice if picking were done in notes."*
- *"I don't see the UI affecting expressivity."*
- *"I think it would be less expressive."*
- *"Easier to use but probably not more expressive. The MechBass controls work well for its parameters, but there's more to Swivel."*
- *"No."*
- *"Nope."*

Question 14: Would Swivel 2 be easier to use if it had the ability to tune itself?

- *"Definitely."*
- *"Not at my [novice] level of expertise."*
- *"I didn't particularly run into a tuning issue, but it could be a strong and positive feature."*
- *"Yes, for composers who work in the domain of western scales / temperment."*

The remaining users simply indicated "yes."

Question 15: Please add any other comments about Swivel 2 here.

- *"The Ableton curve graphs are disorienting at first, when 0-127 represents the rotation of the clamp or damping. But one can get used to it."*
- *"It was slightly more tricky. (I missed being able to choose my pitches)- but there were more expressive options than the other devices."*
- *"I think that it would benefit from more condensed commands, however both ways of controlling it would aid different composition techniques."*
- *"It seems currently that Swivel is an interesting producer of sound with many forms of interaction. The difficulties it may have at the moment is line-coding + working with pitch-based instruments like MechBass. This could be changed through tuned awareness, but it's not necessary as it can serve other purposes."*
- *"Swivel 2 allows for more expressivity in music making compared to MechBass; it is more difficult to use, though this could possibly be resolved with an interface."*
- *"An intuitive interface for improvisation-setting that maps natural motion to sound. I.E. control gesture = sonic gesture. For composition, a similar sequencer such as Live is good, but it'd be less confusing if all the parameters can be seen simultaneously."*
- *"Display feedback for pitch; this doesn't necessary need to be quantized."*

Step 9: Answer the following general questions about the musical robots.

Question 16: What parameters did you find most expressive? Why were these parameters especially expressive?

- *"The slide and clamp. The clamp variations get a lot of tonal colourations and the slide is great for extended microtonal work."*
- *"Volume.... soft vs. hard beats/picks. Easy to perform and add musicality. When notes are playing at one volume level, systems get boring."*
- *"Balance and pitch bend were particularly fun to manipulate. They didn't like extreme parameters, but I got them to provide a cool output."*
- *"I find the pitch arm on Swivel and the beater distance on Nudge to be the most expressive. I think this is because they had the most continuous control."*
- *"Velocity control through all of the instruments seems to be one of the most important. This helps them from sounding monotonous and allows for more nuanced interaction between the instruments. There's a great range of pitch and polyphony control, meaning harmonic and melodic content is easily abled to be composed."*
- *"The picking and damping of Swivel 2 are quite expressive, as well as the pumping in Kritaanjli."*

- *"All the dynamic controls (i.e. Swivel commands, Nudge's stick positions, Kritaanjli's rhythmic flexibility). Because these parameters provide the user with full control over the instrument, and additionally help achieve a variety of timbral results."*
- *"Sliding mechanism of Swivel 2, rotating drumstick of Nudge. They allow for the pseudo-continuous modulation (vs. step) of tones."*

Question 17: What would be your ideal way of interfacing with the robot(s)?

- *"Maybe a combination of Ableton but also some time of user interface as well to make more intuitive on-the-fly performance."*
- *"Custom GUI with parameters designed specifically for robots' capabilities/notes. I know that's a lot, though!"*
- *"I really want to touch them. They just beg for it. But I know that's not necessarily possible... Maybe some kind of small interaction (equivalent to "warming a gong" before striking it) could eventually be implemented."*
- *"I think that a similar setup as in this study, except with a midi controller or other control surface for handling input to make it more performative."*
- *"For MechBass, I think the current Ableton setup works well. Same with Nudge and Kritaanjli, although the latter could be more intuitive with a MIDI keyboard. Swivel poses a problem of how best to control it due to its many parameters. A custom-designed interface may be needed to get the most out of the instrument in a live, improvised context. However, it its current state, a precomposed piece in Ableton would work well."*
- *"It would be interesting, particularly with Swivel 2 and Kritaanjli, to explore some abstract tangible interface that maps actions to expressivity."*
- *"Ableton Live, using custom-designed Ableton MIDI instrument racks that narrow down all the unnecessary ranges and controls to what is needed."*
- *"An intuitive interface for improvisation-setting that maps natural motion to sound. I.E. control gesture = sonic gesture. For composition, a similar sequencer such as Live is good, but it'd be less confusing if all the parameters can be seen simultaneously."*

Question 18: Do you think the expressive parameters may get in the way of composition? If so, how might this be solved?

- *"I think new users need clear instructions on how to get the standard ranges out of the instruments first, so they can build their expressive parameters from familiar territory."*
- *"Present a simple mode with advanced sub-options. The notes on/off were nice for starters, but I like playing with the other options."*
- *"No- they are important part of composition! (Like being able to use 'p' and 'f' in a traditional instrument's music)."*

- *"I don't think so. Although some of the parameters took a while to understand, I didn't see this as any more difficult than learning "how" to compose for "traditional" instruments."*
- *"Condensing the controls into a common form for all instruments may aid the speed of composition and the workflow, however, it would be at the cost of the possibilities these instruments afford. I think the expressive parameters are what makes these instruments valid and interesting, and what sets them apart."*
- *"No."*
- *"(See above: Q14 ⇒ different modes) [Add an easy and a 'full' mode]."*
- *"It gets in the way of laying down ideas. It might be helpful to create sets of envelopes to be used. IE staccato picking on Swivel = Damping On ⇒ Off. Pick once."*

Question 19: What other parameters would you like to see added?

- *"Keyboard shortcuts for immediate playing/interaction, with sliders/switches/knobs/buttons for parameters. This would help with composing process."*f
- *"Hmm— already mentioned things. Nothing additional to add."*
- *"I think Swivel would benefit from a more traditional discrete-pitch method of composition and the ability to change bellows speed on Kritaanjli."*
- *"Pitch awareness for Swivel would mean that it could become a great melodic and harmonic instrument. Especially if the pitch awareness was accurate enough to do quarter-tones. This could become a fluid instrument that could shift between harmonic series chords."*
- *"Some JI [just intonation] tunings would be interesting."*
- *"Ableton Live instruments would make the user's interaction much more efficient."*
- *"Allow MechBass to do sliding tones. Plucking the strings with different angles."*

Question 20: Please add any additional comments below.

- *"I like the system a lot!"*
- *"I love the inherent 'beat' from triggering the keys on Kritaanjli."*
- *"I think the system would benefit from amplification and mixing to get the instruments to similar volume levels."*
- *"It's an amazing array of instruments."*
- *"Since the terminal keeps track of all the activities and prints them, it's hard to go back and find the initialisation information such as dedicated MIDI channels. The GUI for the server would be very helpful."*
- *"Brilliant set of instruments. A lot of potential for new music ideas."*

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

Phone 0-4-463 5676
Fax 0-4-463 5209
Email Allison.kirkman@vuw.ac.nz

MEMORANDUM

TO	Jim Murphy
COPY TO	Ajay Kapur
FROM	Dr Allison Kirkman, Convener, Human Ethics Committee
DATE	4 May 2014
PAGES	1
SUBJECT	**Ethics Approval: 20913** Using New Musical Robots

Thank you for your application for ethical approval, which has now been considered by the Standing Committee of the Human Ethics Committee.

Your application has been approved from the above date and this approval continues until 20 October 2015. If your data collection is not completed by this date you should apply to the Human Ethics Committee for an extension to this approval.

Best wishes with the research.

Allison Kirkman
Human Ethics Committee

FIGURE C.1: The user study approval form.

Bibliography

[1] Eric Singer, Jeff Fedderson, Chad Redmon, and Bil Bowen. Lemur's musical robots. In *Proceedings of the 2004 Conference on New Interfaces for Musical Expression (NIME)*, Hamamatsu, Japan, 2004.

[2] Xavier Rodet and Christophe Vergez. Nonlinear dynamics in physical models: From basic models to true musical-instrument models. *Computer Music Journal*, 23(3): 35–49, Autumn 1999.

[3] Erwin Schoonderwaldt Patrik N. Juslin, Anders Friberg and Jessika Karlsson. *Musical Excellence*, chapter 13, pages 248–253. Oxford University Press, Oxford, 2004.

[4] Cornelius Poepel. On interface expressivity: A player-based study. In *Proceedings of the 2005 Conference on New Interfaces for Musical Expression*, Vancouver, 2005.

[5] Christopher Dobrian and Daniel Koppelman. The E in NIME: Musical expression with new computer interfaces. In *Proceedings of the 2006 Conference of New Interfaces for Musical Expression*, Paris.

[6] Ajay Kapur, Michael Darling, Dimitri Diakopoulos, Jim Murphy, Jordan Hochenbaum, Owen Vallis, and Curtis Bahn. The machine orchestra: An ensemble of human laptop performers and robotic musical instruments. *Computer Music Journal*, 35(4):1–15, 2011.

[7] Henry George Farmer. *The Organ of the Ancients From Eastern Sources*. New Temple Press, London, 1931.

[8] Teun Koetsier. On the prehistory of programmable machines: Musical automata, looms, calculators. *Mechanism and Machine Theory*, 36:589–603, 2001.

[9] Charles Fowler. The museum of music: A history of mechanical instruments. *Music Educators Journal*, pages 45–49, October 1967.

[10] Terrance Riley. Composing for the machine. *European Romantic Review*, 20(3): 367–379, 2009.

[11] Arthur W. J. G. Ord-Hume. Ornamentation in mechanical music. *Early Music*, 11 (2):185–193, April 1983.

[12] David Fuller. An introduction to automatic instruments. *Early Music*, 11(2):164–166, April 1983.

[13] Arthur W.J.G. Ord-Hume. *Pianola: The History of the Self Playing Piano*. Unwin-Hyman, 1985.

[14] Brian Dolan. *Inventing Entertainment: The Player Piano and the Origins of an American Musical Industry*. Rowman and Littlefield Publishers, Lanham, Maryland, 2009.

[15] Musical boxes. *Science*, XII(306), December 1888.

[16] Kyle Gann. *The Music of Conlon Nancarrow*. Music in the Twentieth Century. Cambridge University Press, Cambridge, 2006.

[17] Godfried-Willem Raes. A personal story of music and technologies. *Leonardo Music Journal*, 2(1):29–35, 1992.

[18] Rafael Lozano-Hemmer. Perverting technological correctness. *Leonardo*, 29(1), 1996.

[19] Anne Focke. *Trimpin: Contraptions for Art and Sound*. Marquand Books, Seattle, Washington, 1st edition, 2011.

[20] Troy Rogers Laura Maes, Godfried-Willem Raes. The man and machine robot orchestra at logos. *Computer Music Journal*, 35(4):28–48, 2011.

[21] Sasha Leitman. Trimpin: An interview. *Computer Music Journal*, 35(4):12–27, 2011.

[22] Peter Esmonde. Trimpin: The sound of invention. Film, 2009.

[23] Ajay Kapur. A history of robotic musical instruments. In *Proceedings of the 2005 International Computer Music Conference*, Barcelona, Spain, 2005.

[24] Ajay Kapur. *Digitizing North Indian Music: Preservation and Extension using Multimodal SensorSystems, Machine Learning and Robotics*. VDM Verlag, 2008.

[25] Ajay Kapur and Michael Darling. A pedagogical paradigm for musical robotics. In *Proceedings of the 2010 Conference on New Interfaces for Musical Expression*, Sydney, Australia, 2010.

[26] Ajay Kapur, Jim Murphy, and Dale Carnegie. Kritaanjli: A robotic harmonium for performance, pedagogy, and research. In *Proceedings of the 2012 Conference on New Interfaces for Musical Expression*, Ann Arbor, Michigan, May 2012.

[27] Gil Weinberg and Scott Driscoll. Toward robotic musicianship. *Computer Music Journal*, 30(4):28–45, Winter 2006.

[28] Guy Hoffman and Gil Weinberg. *Musical Robots and Interactive Multimodal Systems*, chapter 14, pages 233–251. Number 74 in Springer Tracts in Advanced Robotics. Springer, 2011.

[29] Daisuke Sakamoto Jun Kato and Takeo Igarashi. Picode: Inline photos representing posture data in source code. In *CHI '13: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 3097–3100, April 2013.

[30] David K. Grunberg Alyssa M. Batula, Manu Colacot and Youngmoo E. Kim. Using audio and haptic feedback to detect errors in humanoid musical performances. In *Proceedings of the 2013 Conference on New Interfaces for Musical Expression (NIME)*, Daejeon, Korea, 2013.

[31] Jim Murphy, Ajay Kapur, and Dale Carnegie. Musical robotics in a loudspeaker world: Developments in alternative approaches to localization and spatialization. *Leonardo Music Journal*, 22(1):41–48, December 2012.

[32] Elvind Groven. Improvements in a tuning system for pianos. Patent, 1935.

[33] Edgar Berdahl, Steven Backer, and Julius O. Smith III. If I had a hammer: Design and theory of an electromagnetically-prepared piano. In *Proceedings of the 2005 International Computer Music Conference*, 2005.

[34] Per Bloland. The electromagnetically-prepared piano and its implications. In *Proceedings of the 2007 International Computer Music Conference*, 2007.

[35] Michael A. Fabio. The Chandelier : an exploration in robotic musical instrument design. Master's thesis, Massachusetts Institute of Technology, 2007.

[36] Andrew McPherson and Youngmoo Kim. Augmenting the acoustic piano with electromagnetic string actuation and continuous key position sensing. In *Proceedings of the 2010 Conference on New Interfaces for Musical Expression*, Sydney, Australia, 2010.

[37] Alcedo Coenen. Computer-controlled player pianos. *Computer Music Journal*, 16 (4):104–111, 1992.

[38] Curtis Roads. The Tsukuba musical robot. *Computer Music Journal*, 10(2):39–43, 1986.

[39] Jean-Claude Risset and Scott Van Duyne. Real-time performance interaction with a computer-controlled acoustic piano. *Computer Music Journal*, 20(1):62–75, 1996.

[40] Osvaldo Budon and Horacio Vaggione. Composing with objects, networks, and time scales: An interview with Horacio Vaggione. *Computer Music Journal*, 24(3):9–22, 2000.

[41] Jean-Claude Risset. Fifty years of digital sound for music. In *Proceedings of the 2007 Sound and Music Computing Conference*, Lefkada, Greece, 2007.

[42] Jeff Aaron Bryant. Non-affirming strategies in digital correlation. Master's thesis, California Institute of the Arts, Valencia, California, 2012.

[43] Eric Singer, Jeff Fedderson, and David Bianciardi. LEMUR Guitarbot: MIDI robotic string instrument. In *Proceedings of the 2003 International Conference on New Interfaces for Musical Expression*, Montreal, Canada, 2003.

[44] Arthur W. J. G. Ord-Hume. Cogs and crotchets: A view of mechanical music. *Early Music*, 11(2):167–171, April 1983.

[45] Richard Vindriis, Ajay Kapur, and Dale Carnegie. A comparison of pick-based strategies for robotic bass playing. In *Proceedings of the 2011 Electronics New Zealand Conference*, 2011.

[46] Aengus Martin, Sam Ferguson, and Kirsty Beilharz. Mechanisms for controlling complex sound sources: Applications to guitar feedback control. In *Proceedings of*

the 2010 Conference on New Interfaces for Musical Expression, Sydney, Australia, 2010.

[47] Andrew Johnston Sam Ferguson and Aengus Martin. A corpus-based method for controlling guitar feedback. In *Proceedings of the 2013 Conference on New Interfaces for Musical Expression (NIME)*, 2013.

[48] Sergi Jordà. Afasia: the ultimate homeric one-man-multimedia-band. In *Proceedings of the 2002 Conference of New Interfaces for Musical Expression*, 2002.

[49] Jim Murphy, Ajay Kapur, and Dale A. Carnegie. Swivel: Analysis and overview of a new robotic guitar. In *Proceedings of the 2013 International Computer Music Conference*, Perth, Western Australia, 2013.

[50] Dan O'Sullivan and Tom Igoe. *Physical Computing: Sensing and Controlling the Physical World With Computers*. Thomson Course Technology, Boston, 2004.

[51] Martin Evans, Joshua Noble, and Jordan Hochenbaum. *Arduino in Action*. Manning Publications, 2013.

[52] James McVay, Jim Murphy, Ajay Kapur, and Dale Carnegie. Mechbass: A systems overview of a new four- stringed robotic bass guitar. In *Proceedings of the 2012 Electronics New Zealand Conference*, Dunedin, New Zealand, 2012.

[53] Nicolas Leroy, Emmanuel Fléty, and Frederic Bevilacqua. Reflective optical pickup for violin. In *Proceedings of the 2006 Conference of New Interfaces for Musical Expression*, Paris, 2006.

[54] Dimitri Diakopoulos. Hiduino: A firmware for building driverless usb-midi devices using the arduino microcontroller. In *Proceedings of the 2011 Conference on New Interfaces for Musical Expression*, Oslo, Norway, 2011.

[55] Ajay Kapur, Michael Darling, Jim Murphy, Jordan, Dimitri Diakopoulos, and Trimpin. The KarmetiK NotomotoN: A new breed of musical robot for teaching and performance. In *Proceedings of the 2011 Conference on New Interfaces for Musical Expression*, Oslo, Norway, 2011.

[56] Steven Ness, Shawn Trail, Peter Driessen, Andrew Schloss, and George Tzanetakis. Music information robotics: Coping strategies for musically challenged robots. In

Proceedings of the 2011 International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.

[57] Jim Murphy, Ajay Kapur, and Dale Carnegie. Better drumming through calibration: Techniques for pre-performance robotic percussion optimization. In *Proceedings of the 2012 Conference on New Interfaces for Musical Expression*, Ann Arbor, Michigan, June 2012. NIME.

[58] Ajay Kapur. Karmetik machine orchestra. DVD, October 2010.

[59] Ajay Kapur, Trimpin, Eric Singer, Afzal Suleman, and George Tzanetakis. A comparison of solenoid-based strategies for robotic drumming. In *Proceedings of the 2007 International Computer Music Conference*, Copenhagen, 2007.

[60] Jim Murphy, Ajay Kapur, and Dale Carnegie. Interacting with solenoid drummers: A quantitative approach to composing and performing with open-loop solenoid-based robotic percussion systems. In *Proceedings of the 2012 Australasian Computer Music Conference*, Brisbane, Australia, July 2012.

[61] Matt Wright and A. Freed. Open sound control: A new protocol for communicating with sound synthesizers. In *Proceedings of the 1997 International Computer Music Conference*, Thessaloniki, Greece, 1997.

[62] Ge Wang. *The ChucK Audio Programming Language: A Strongly-timed and On-the-fly Environ/mentality*. PhD thesis, Princeton University, 2008.

[63] Neil C. Skinn Stephen J. Freeland, Gerard F. Hallaren. Musical instrument self-tuning system with calibration library. US Patent, 1998.

[64] Steve Kith Bryan S. Johnson Neil Skinn, Frank Strazzabosco. Apparatus and method for self-tuning stringed musical instruments with an accompanying vibrato mechanism. US Patent, 2008.

[65] Zhen J. Wang and Cesar Ortega-Sanchez. Electronic assisting violin tuner. In *TENCON 2012 - 2102 IEEE Region 10 Conference*, Cebu, 2012.

[66] G.B. Minnick. Self-tuning tail piece for string instruments. US Patent 4,584,923, April 1986.

[67] Duncan MacConnell George Tzanetakis Mantis Cheng Shawn Trail, Leonardo Jenkins and Peter Driessen. Stari: A self tuning auto-monochord robotic instrument. In *Proceedings of the 2013 PACRIIM*, Victoria, BC, 2013.

[68] Perry Cook, editor. *Music, Cognition, and Computerized Sound*. MIT Press, Cambridge, Mass., first edition, 1999.

[69] Marcelo Mortensen Wanderley and Nicola Orio. Evaluation of input devices for musical expression: Borrowing tools from HCI. *Computer Music Journal*, 26, 2002.

[70] John Bischoff, Rich Gold, and Jim Horton. Music for an interactive network of microcomputers. *Computer Music Journal*, 2(3):24–29, December 1978.

[71] Scot Gresham-Lancaster. The aesthetics and history of the hub: The effects of changing technology on network computer music. *Leonardo Music Journal*, 8:39–44, 1998.

[72] Dan Trueman, Perry Cook, Scott Smallwood, and Ge Wang. Plork: The Princeton Laptop Orchestra, year 1. In *Proceedings of the 2006 International Computer Music Conference*, New Orleans, 2006.

[73] Owen Vallis, Dimitri Diakopoulos, Jordan Hochenbaum, and Ajay Kapur. Building on the foundations of network music: Exploring interaction contexts and shared robotic instruments. *Organised Sound*, 17(1):62–72, April 2012.

[74] Perry Cook. Principles for designing computer music controllers. In *Proceedings of the 2001 Conference on New Interfaces for Musical Expression*, 2001.

[75] Sile O'Modhrain. A framework for the evaluation of digital musical instruments. *Computer Music Journal*, 35(1):28–42, 2011.

[76] *Electronic and Experimental Music: Technology, Music, and Culture*. Roudledge, 1 edition, 2012.

[77] George Ruckert and Ali Akbar Khan, editors. *The Classical Music of North India: The first years study*. Munshiram Manoharlal Publishers, 1998.